```
LOCATION OBJECT CODE LINE     SOURCE LINE

                          1 ^6801^
                          3  NAME ^Rev 01 - HME^
                          4
                          5 De_TAPE_MAC MACRO                ;Header Rev. 4
                          6                 .GOTO Ede_TAPE_MAC
                          7
                          8  Project:       NET, 83-101
                          9
                         10  ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
                         11  ▓                            ▓
                         12  ▓    TAPE_MAC           HME  ▓
                         13  ▓                            ▓
                         14  ▓    LINKS  INTO  REV_23     ▓
                         15  ▓               8865H        ▓
                         16  ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
                         17
                         18       Rev History
                         19       Rev.  Date          Name      Change
                         20        1    13SEP1034    HME       After sending out a block with a bad cs,
                         21                                    mangle CURRENT_RAM to prevent us from re-transmission
                         22        0    16AUG1601    HME       Initial code
                         23
                         24 Ede_TAPE_MAC     MEND
                         25
                         26 ;
                         27 ; LOCAL EQUATES
                         28 ;
                         29                 GLB   COMMAND_BUFFER
                         30                 GLB   CURRENT_RAM
                         31                 GLB   IO_STATUS_BLOCK
                         32                 GLB   LENGTH_OF_IO_STATUS
                         33                 GLB   TAPE_MAC
                         34                 GLB   TAPE_STATUS0
                         35                 GLB   TAPE_STATUS1
                         36 ;
  <0008>                 37 NODE_ADDRESS    EQU             08H            ; ARE WE NOT TAPE?
  <0008>                 38 MN_RESET        EQU             00H*16+NODE_ADDRESS
  <0018>                 39 MN_STATUS       EQU             01H*16+NODE_ADDRESS
  <0028>                 40 MN_ACK          EQU             02H*16+NODE_ADDRESS
  <0038>                 41 MN_CLR          EQU             03H*16+NODE_ADDRESS
  <0048>                 42 MN_RECEIVE      EQU             04H*16+NODE_ADDRESS
  <0058>                 43 MN_CANCEL       EQU             05H*16+NODE_ADDRESS
  <0068>                 44 MN_SEND         EQU             06H*16+NODE_ADDRESS
  <0078>                 45 MN_NACK         EQU             07H*16+NODE_ADDRESS
  <00D8>                 46 MN_READY        EQU             0DH*16+NODE_ADDRESS
                         47 ;
  <0088>                 48 NM_STATUS       EQU             08H*16+NODE_ADDRESS
  <0098>                 49 NM_ACK          EQU             09H*16+NODE_ADDRESS
  <00A8>                 50 NM_CANCEL       EQU             0AH*16+NODE_ADDRESS
  <00B8>                 51 NM_SEND         EQU             0BH*16+NODE_ADDRESS
  <00C8>                 52 NM_NACK         EQU             0CH*16+NODE_ADDRESS
                         53 ;
                         54 ;STATES
                         55 ;
  <0000>                 56 CNTRL           EQU             0
  <0001>                 57 LENGTH_HI       EQU             1
  <0002>                 58 LENGTH_LO       EQU             2
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

            <0003>     59 IGNORE_JUNK     EQU            3
            <0004>     60 DATAIN          EQU            4
            <0005>     61 CS_IN           EQU            5
                       62 ;
                       63 ; STUFF TO WRITE IN NIM
            <000B>     64 C_READ          EQU            11
            <000C>     65 C_WRITE         EQU            12
            <0052>     66 C_REWIND        EQU            82
                       67 ;
                       68 ; STATII WRITTEN BY APP.
            <0000>     69 S_OK    EQU     0
            <0001>     70 S_BADBLK EQU    1
            <0002>     71 S_NOBLOCK EQU   2
            <0003>     72 S_NOTAPE EQU    3
            <0004>     73 S_NODRIVE EQU   4
                       74 ;
            <0001>     75 LENGTH_OF_IO_STATUS EQU    1
                       76 ; IMPORTANT STUFF
                       77 ;
                       78                 DATA
    0000               79 COMMAND_BUFFER  RMB            5
    0005               80 CURRENT_RAM     RMB            5
    000A               81 COUNT           RMB            2
    000C               82 GO_TO_TAPE      RMB            1
    000D               83 MEM_PTR         RMB            2
    000F               84 CS_BYTE         RMB            1
    0010               85 RAM_STATUS      RMB            4
    0014               86 IO_STATUS_BLOCK RMB            1
    0015               87 TAPE_STATUS0    RMB            1
    0016               88 TAPE_STATUS1    RMB            1
                       89 ;
                       90                 EXT        MTP_TR_TRANS
                       91                 EXT        MTP_TR_TCU
                       92                 EXT        MTP_TR_REC
                       93                 EXT        MTP_NIM_WRITE
                       94                 EXT        CURRENT_STATE
                       95                 EXT        DATA_BUFFER
                       96                 EXT        M_SIG
                       97 ;
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                                     PROG
                             ; MAIN PROGRAM HERE
     0000 BD0000             TAPE_MAC          JSR       MTP_TR_REC
     0003 2508                                 BCS       DATA_FOR_US
                             ; SET WAKEUP BIT
     0005 C61B                                 LDAB      #00011011B
     0007 D711                                 STAB      011H,D

     0009 8600                                 LDAA      #CNTRL              ; BACK TO COMMAND MODE
     000B 9700                                 STAA      CURRENT_STATE,D

     000D 7E019E                               JMP       JUST_RETURN

     0010 D600               DATA_FOR_US       LDAB      CURRENT_STATE,D
     0012 58                                   LSLB
     0013 CE001B                               LDX       #STATE_TABLE
     0016 3A                                   ABX
     0017 EE00                                 LDX       0,X
     0019 6E00                                 JMP       0,X

                             ; JUMP TABLE
     001B 0027               STATE_TABLE       FDB       CONTROL
     001D 00F6                                 FDB       GET_LENH
     001F 00FF                                 FDB       GET_LENL
     0021 012B                                 FDB       GET_JUNK
     0023 0132                                 FDB       GET_DATA
     0025 014C                                 FDB       GET_CS
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                         ***************************************
                         * CONTROL STATE                       *
                         ***************************************
0027                     CONTROL
0027 8108                                CMPA      #MN_RESET
0029 2607                                BNE       NOT_RESET

002B 0D                                  SEC
002C BD0000                              JSR       MTP_NIM_WRITE

002F 7E019E                              JMP       JUST_RETURN
                         ;
0032 8118                NOT_RESET       CMPA      #MN_STATUS
0034 2755                                BEQ       SEND_STATUS
                         ;
0036 8138                                CMPA      #MN_CLR
0038 2775                                BEQ       SEND_DATA
                         ;
003A 8148                                CMPA      #MN_RECEIVE
003C 2617                                BNE       NOT_RECEIVE
                         ; TEST TO SEE IF COMMAND_BUFFER = CURRENT_RAM
003E CE0005                              LDX       #5
0041                     B_TEST
0041 A6FF                                LDAA      COMMAND_BUFFER-1,X
0043 A104                                CMPA      CURRENT_RAM-1,X
0045 2605                                BNE       DONT_HAVE_IT
0047 09                                  DEX
0048 26F7                                BNE       B_TEST
                         ; OK. WE HAVE IT IN RAM
004A 202D                                BRA       SEND_ACK
                         ; WE HAVE TO SPIN UP THE TAPE
004C                     DONT_HAVE_IT
004C 860B                                LDAA      #C_READ
004E 0C                                  CLC
004F BD0000                              JSR       MTP_NIM_WRITE
0052 7E019E                              JMP       JUST_RETURN
                         ;
0055 8168                NOT_RECEIVE     CMPA      #MN_SEND
0057 2607                                BNE       NOT_SEND
                         ; SEND STATE
0059 8601                                LDAA      #LENGTH_HI
005B 9700                                STAA      CURRENT_STATE,D
005D 7E019E                              JMP       JUST_RETURN
0060                     NOT_SEND
0060 81D8                                CMPA      #MN_READY
0062 2715                                BEQ       SEND_ACK
0064 7E019E                              JMP       JUST_RETURN
0067                     SEND_NACK
0067 86C8                                LDAA      #NM_NACK
0069 BD0000                              JSR       MTP_TR_TRANS
006C 2508                                BCS       ERR1
006E BD0000                              JSR       MTP_TR_TCU
0071 2503                                BCS       ERR1
0073 7E019E                              JMP       JUST_RETURN
0076                     ERR1:
0076 7E01AB                              JMP       RETURN_NOW

```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0079              184 SEND_ACK
    0079 8698         185                 LDAA      #NM_ACK
    007B BD0000       186                 JSR       MTP_TR_TRANS
    007E 2508         187                 BCS       ERR2
    0080 BD0000       188                 JSR       MTP_TR_TCU
    0083 2503         189                 BCS       ERR2
    0085 7E019E       190                 JMP       JUST_RETURN
    0088              191 ERR2:
    0088 7E01AB       192                 JMP       RETURN_NOW
                      193 ; SEND OUT STATUS PACKET
    008B              194 SEND_STATUS
                      195 ; COPY THE ROM STATUS PACKET (BYTES 0-3) INTO RAM_STATUS AREA
    008B FC01B7       196                 LDD       STAT_MSG_TBL
    008E DD10         197                 STD       RAM_STATUS,D
    0090 FC01B9       198                 LDD       STAT_MSG_TBL+2
    0093 DD12         199                 STD       RAM_STATUS+2,D
    0095 BD01AC       200                 JSR       ASMB_STATUS
                      201 ; INIT PTRS
    0098 CE0010       202                 LDX       #RAM_STATUS
    009B CC0005       203                 LDD       #STAT_MSG_LEN
    009E DD0A         204                 STD       COUNT,D
    00A0 8688         205                 LDAA      #NM_STATUS
    00A2 970F         206                 STAA      CS_BYTE,D     ; SO THAT CS GETS CLEARED AFTER COMMAND IS SENT
    00A4 BD017E       207                 JSR       LSSD
    00A7 2503         208                 BCS       ERR5
    00A9 7E019E       209                 JMP       JUST_RETURN
    00AC              210 ERR5:
    00AC 7E01AB       211                 JMP       RETURN_NOW
    00AF              212 SEND_DATA
    00AF 7D0004       213                 TST       COMMAND_BUFFER+4,D
    00B2 2705         214                 BEQ       CHK_DR0
                      215 ; SEE IF DRIVE ONE IS EITHER DOWN OR EMPTY
    00B4 B60016       216                 LDAA      TAPE_STATUS1
    00B7 2003         217                 BRA       SD_2
    00B9              218 CHK_DR0
                      219 ; WHAT ABOUT DRIVE 0?
    00B9 B60015       220                 LDAA      TAPE_STATUS0
    00BC 8103         221 SD_2            CMPA      #S_NOTAPE
    00BE 240A         222                 BHS       NO_TAPE
                      223 ; PREPARE DATA FOR OUTPUT.
                      224 ; REG X = PTR TO DATA
                      225 ; COUNT,COUNT+1 = BYTES TO TRANSFER
                      226 ; CARRY SET IF IO_STATUS PRECEDES DATA, CLEAR OTHERWISE
    00C0 CE0000       227                 LDX       #DATA_BUFFER
    00C3 CC0400       228                 LDD       #1024                   ; CONDITIONALLY INCREASE BLOCK SIZE
    00C6 DD0A         229                 STD       COUNT,D
    00C8 200B         230                 BRA       SD_1
    00CA              231 NO_TAPE
    00CA BD01AC       232                 JSR       ASMB_STATUS         ; PUT STATUS BYTE TOGETHER
    00CD CE0014       233                 LDX       #IO_STATUS_BLOCK
    00D0 CC0001       234                 LDD       #1
    00D3 DD0A         235                 STD       COUNT,D
    00D5              236 SD_1
    00D5 BD0166       237                 JSR       LETS_SEND_DATA
    00D8 2519         238                 BCS       ERR4
                      239 ; IT GOT SENT OK, BUT IF WE SENT OUT DATA WITH A BAD CS, THEN BASH
                      240 ; COMMAND_BUFFER SO WE DON'T EVER RESEND IT
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    00DA 7D0004                               TST       COMMAND_BUFFER+4,D
    00DD 2705                                 BEQ       CS_CHK0
                           ; SEE IF DRIVE ONE IS EITHER DOWN OR EMPTY
    00DF B60016                               LDAA      TAPE_STATUS1
    00E2 2003                                 BRA       CS_CHK_COMN
    00E4                   CS_CHK0
                           ; WHAT ABOUT DRIVE 0?
    00E4 B60015                               LDAA      TAPE_STATUS0
    00E7 8101              CS_CHK_COMN        CMPA      #S_BADBLK
    00E9 2605                                 BNE       OK_CS_SENT
                           ; MANGLE COMMAND_BUFFER BEYOND RECOGNITION
    00EB 86FF                                 LDAA      #255
    00ED B70009                               STAA      CURRENT_RAM+4
    00F0                   OK_CS_SENT:
    00F0 7E019E                               JMP       JUST_RETURN
    00F3                   ERR4:
    00F3 7E01AB                               JMP       RETURN_NOW
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      259 ****************************************
                      260 * GET LENGTH_HI STATE                  *
                      261 ****************************************
    00F6              262 GET_LENH
    00F6 970A         263                 STAA      COUNT,D
    00F8 8602         264                 LDAA      #LENGTH_LO
    00FA 9700         265                 STAA      CURRENT_STATE,D
    00FC 7E019E       266                 JMP       JUST_RETURN
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                       268 ****************************************
                       269 * GET LENGTH_LO STATE                  *
                       270 ****************************************
  00FF                 271 GET_LENL
  00FF 970B            272                STAA      COUNT+1,D
                       273 ; 5 BYTE COMMAND PACKET COMING IN?
  0101 8105            274                CMPA      #5
  0103 2604            275                BNE       NOT_5_BYTES
                       276
  0105 960A            277                LDAA      COUNT,D
  0107 2710            278                BEQ       CMD_COMING_IN
                       279
  0109                 280 NOT_5_BYTES
  0109 8603            281                LDAA      #IGNORE_JUNK
  010B 9700            282                STAA      CURRENT_STATE,D
                       283
  010D 8601            284                LDAA      #1
  010F 970C            285                STAA      GO_TO_TAPE,D
                       286
  0111 CC0000          287                LDD       #DATA_BUFFER
  0114 DD0D            288                STD       MEM_PTR,D
                       289
  0116 7E019E          290                JMP       JUST_RETURN
                       291
  0119                 292 CMD_COMING_IN
  0119 8604            293                LDAA      #DATAIN
  011B 9700            294                STAA      CURRENT_STATE,D
                       295
  011D 7F000C          296                CLR       GO_TO_TAPE,D
                       297
  0120 CC0000          298                LDD       #COMMAND_BUFFER
  0123 DD0D            299                STD       MEM_PTR,D
                       300
  0125 7F000F          301                CLR       CS_BYTE
                       302
  0128 7E019E          303                JMP       JUST_RETURN
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      305 ****************************************
                      306 * GET JUNK STATE                       *
                      307 ****************************************
                      308 ; IGNORE THIS BYTE , BUT SET UP FOR SUCKING 1K
    012B              309 GET_JUNK
                      310
    012B C604         311                 LDAB      #DATAIN
    012D D700         312                 STAB      CURRENT_STATE,D
                      313
    012F 7F000F       314                 CLR       CS_BYTE
                      315
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      317 ****************************************
                      318 * GET DATA IN STATE                    *
                      319 ****************************************
   0132               320 GET_DATA
   0132 DE0D          321                 LDX      MEM_PTR,D
   0134 A700          322                 STAA     0,X
   0136 08            323                 INX
   0137 DF0D          324                 STX      MEM_PTR,D
   0139 980F          325                 EORA     CS_BYTE,D
   013B 970F          326                 STAA     CS_BYTE,D
   013D DC0A          327                 LDD      COUNT,D
   013F 830001        328                 SUBD     #1
   0142 DD0A          329                 STD      COUNT,D
   0144 2658          330                 BNE      JUST_RETURN            ; ALL DONE AT 0
                      331 ; NO MORE
   0146 8605          332                 LDAA     #CS_IN
   0148 9700          333                 STAA     CURRENT_STATE,D
   014A 2052          334                 BRA      JUST_RETURN
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      336 ********************************************
                      337 * GET CHECK SUM STATE                      *
                      338 ********************************************
    014C              339 GET_CS
    014C C600         340                 LDAB      #CNTRL
    014E D700         341                 STAB      CURRENT_STATE,D
    0150 910F         342                 CMPA      CS_BYTE,D
                      343 *               BNE       SEND_NACK
    0152 2703         344                 BEQ       GCS_1
    0154 7E0067       345                 JMP       SEND_NACK
                      346 ; WE WIN- CHECK SUM IS OK!
                      347 ; TELL GUY TO DUMP TO TAPE (IF NOT CMD PKT)
    0157 960C         348 GCS_1           LDAA      GO_TO_TAPE,D
                      349 *               BEQ       SEND_ACK               ; SINCE WE DONT WANT THIS ON TAPE,
    0159 2603         350                 BNE       GCS_2                  ; THIS MUST BE A COMMAND PACKET.
    015B 7E0079       351 GCS_SA          JMP       SEND_ACK
    015E 0C           352 GCS_2           CLC
    015F 860C         353                 LDAA      #C_WRITE
    0161 BD0000       354                 JSR       MTP_N1M_WRITE
                      355 *               BRA       SEND_ACK
    0164 20F5         356                 BRA       GCS_SA
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      358 ;
                      359 ; SUBROUTINE TO DUMP DATER OUT ON NET
                      360 ;
  0166                361 LETS_SEND_DATA
                      362 ; ASSUMES PTR IN X, BYTES TO TRANSMIT IN COUNT,COUNT+1
                      363
                      364 ; FIRST, SEND COMMAND TO MASTER
  0166 86B8           365                    LDAA      #NM_SEND
  0168 BD0000         366                    JSR       MTP_TR_TRANS
  016B 2530           367                    BCS       ERR3
                      368
                      369 ; NEXT, HIGH BYTE OF TRANSMISSION LENGTH
  016D 960A           370                    LDAA      COUNT,D
  016F BD0000         371                    JSR       MTP_TR_TRANS
  0172 2529           372                    BCS       ERR3
                      373
                      374 ; NEXT, LOW BYTE
  0174 960B           375                    LDAA      COUNT+1,D
  0176 BD0000         376                    JSR       MTP_TR_TRANS
  0179 2522           377                    BCS       ERR3
                      378
  017B 7F000F         379                    CLR       CS_BYTE
                      380
  017E                381 LSSD
  017E A600           382                    LDAA      0,X
  0180 BD0000         383                    JSR       MTP_TR_TRANS
  0183 2518           384                    BCS       ERR3
                      385
  0185 980F           386                    EORA      CS_BYTE,D
  0187 970F           387                    STAA      CS_BYTE,D
  0189 08             388                    INX
  018A DC0A           389                    LDD       COUNT,D
  018C 830001         390                    SUBD      #1
  018F DD0A           391                    STD       COUNT,D
  0191 26EB           392                    BNE       LSSD
                      393  ; LASTLY, SEND CHECK SUM
  0193 960F           394                    LDAA      CS_BYTE,D
  0195 BD0000         395                    JSR       MTP_TR_TRANS
  0198 2503           396                    BCS       ERR3
  019A BD0000         397                    JSR       MTP_TR_TCU
  019D                398 ERR3:
  019D 39             399                    RTS                          ; ALL DONE HERE
                      400 ***************************************************
                      401 *                  THE END IS NEAR              *
                      402 ***************************************************
  019E                403 JUST_RETURN
                      404 ; IF WE'VE JUST TOLD THE APP TO DO SOMETHING, DISABLE REC INTRPTS.
  019E 7D0000         405                    TST       M_SIG,D
  01A1 2708           406                    BEQ       RETURN_NOW
                      407 ; DISABLE INTRPTS
  01A3 860A           408                    LDAA      #0AH
  01A5 9711           409                    STAA      011H               ; CLEAR THE ENABLE BIT
  01A7 9611           410                    LDAA      011H               ; CLEAR ANY PENDING INTRPT
  01A9 9612           411                    LDAA      012H
  01AB 3B             412 RETURN_NOW         RTI
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                        414 ****************************************************************
                        415 * THIS GUY ASSEMBLES TAPE_STATUS0&1 TOGEHER INTO IO_STATUS_BLOCK *
                        416 ****************************************************************
    01AC                417 ASMB_STATUS
    01AC 9616           418                     LDAA      TAPE_STATUS1,D
    01AE 48             419                     LSLA
    01AF 48             420                     LSLA
    01B0 48             421                     LSLA
    01B1 48             422                     LSLA
    01B2 9A15           423                     ORAA      TAPE_STATUS0,D
    01B4 9714           424                     STAA      IO_STATUS_BLOCK,D
    01B6 39             425                     RTS
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      427 ************************************************************
                      428 *                                                          *
                      429 *  DATA TABLE NAME:                                        *
                      430 *                                                          *
                      431 *     STAT_MSG_TBL                                         *
                      432 *                                                          *
                      433 *  DESCRIPTION:                                            *
                      434 *                                                          *
                      435 *     THIS TABLE CONTAINS THE PACKAGE THAT THIS NODE       *
                      436 *     SENDS TO THE MASTER IN RESPONSE TO THE STATUS        *
                      437 *     COMMAND.                                             *
                      438 *                                                          *
                      439 *  INDEXED BY:                                             *
                      440 *                                                          *
                      441 *     A LOOP COUNTER                                       *
                      442 *                                                          *
                      443 ************************************************************
             (01B7)   444 STAT_MSG_TBL:  EQU          $
    01B7 88           445                FCB          080H+NODE_ADDRESS      ;STATUS.OR.ADDRESS
    01B8 00           446                FCB          000H       ;MAX MSG LENGTH (1K LOW BYTE)
    01B9 04           447                FCB          004H       ;MAX MSG LENGTH (HIGH BYTE)
    01BA 01           448                FCB          001H       ;TRANSMIT CODE=BYTE_MODE.OR.RESERVED
    01BB 00           449                FCB          000H       ;STATUS FLAGS
             (0005)   450 STAT_MSG_LEN:  EQU      $-STAT_MSG_TBL

Errors=    0
```

```
LINE#   SYMBOL          TYPE     REFERENCES

 417  ASMB_STATUS       P  200,232
 149  B_TEST            P  154
 218  CHK_DR0           P  214
 292  CMD_CUMING_IN     P  278
  56  CNTRL             A  107,340
  79  COMMAND_BUFFER    D  29,150,213,241,298
 130  CONTROL           P  120
  81  COUNT             D  204,229,235,263,272,277,327,329,370,375,389,391
  84  CS_BYTE           D  206,301,314,325,326,342,379,386,387,394
 246  CS_CHK0           P  242
 249  CS_CHK_COMN       P  245
  61  CS_IN             A  332
  80  CURRENT_RAM       D  30,151,253
  94  CURRENT_STATE     E  108,112,168,265,282,294,312,333,341
  64  C_READ            A  159
  66  C_REWIND          A
  65  C_WRITE           A  353
  60  DATAIN            A  293,311
  95  DATA_BUFFER       E  227,287
 112  DATA_FOR_US       P  102
 158  DONT_HAVE_IT      P  152
 181  ERR1              P  177,179
 191  ERR2              P  187,189
 398  ERR3              P  367,372,377,384,396
 256  ERR4              P  238
 210  ERR5              P  208
 348  GCS_1             P  344
 352  GCS_2             P  350
 351  GCS_SA            P  356
 339  GET_CS            P  125
 320  GET_DATA          P  124
 309  GET_JUNK          P  123
 262  GET_LENH          P  121
 271  GET_LENL          P  122
  82  GO_TO_TAPE        D  285,296,348
  59  IGNORE_JUNK       A  281
  86  IO_STATUS_BLOCK   D  31,233,424
 403  JUST_RETURN       P  110,137,162,169,173,180,190,209,255,266,290,303,330,334
  57  LENGTH_HI         A  167
  58  LENGTH_LO         A  264
  75  LENGTH_OF_IO_ST   A  32
 361  LETS_SEND_DATA    P  237
 381  LSSD              P  207,392
  83  MEM_PTR           D  288,299,321,324
  40  MN_ACK            A
  43  MN_CANCEL         A
  41  MN_CLR            A  142
  45  MN_NACK           A
  46  MN_READY          A  171
  42  MN_RECEIVE        A  145
  38  MN_RESET          A  131
  44  MN_SEND           A  164
  39  MN_STATUS         A  139
  93  MTP_NIM_WRITE     E  135,161,354
  92  MTP_TR_REC        E  101
  91  MTP_TR_TCU        E  178,188,397
  90  MTP_TR_TRANS      E  176,186,366,371,376,383,395
```

```
LINE#   SYMBOL          TYPE     REFERENCES

   96  M_SIG             E  405
   49  NM_ACK            A  185
   50  NM_CANCEL         A
   52  NM_NACK           A  175
   51  NM_SEND           A  365
   48  NM_STATUS         A  205
   37  NODE_ADDRESS      A  38,39,40,41,42,43,44,45,46,48,49,50,51,52,445
  280  NOT_5_BYTES       P  275
  164  NOT_RECEIVE       P  146
  139  NOT_RESET         P  132
  170  NOT_SEND          P  165
  231  NO_TAPE           P  222
  254  OK_CS_SENT        P  250
   85  RAM_STATUS        D  197,199,202
  412  RETURN_NOW        P  182,192,211,257,406
  236  SD_1              P  230
  221  SD_2              P  217
  184  SEND_ACK          P  156,172,351
  212  SEND_DATA         P  143
  174  SEND_NACK         P  345
  194  SEND_STATUS       P  140
  120  STATE_TABLE       P  114
  450  STAT_MSG_LEN      A  203
  444  STAT_MSG_TBL      P  196,198,450
   70  S_BADBLK          A  249
   71  S_NOBLOCK         A
   73  S_NODRIVE         A
   72  S_NOTAPE          A  221
   69  S_OK              A
  101  TAPE_MAC          P  33
   87  TAPE_STATUS0      D  34,220,248,423
   88  TAPE_STATUS1      D  35,216,244,418
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

 ^6801^
  NAME ^Rev 00 - DLS^

 De_D_MTP MACRO                ;Header Rev. 4
                 .GOTO Ede_D_MTP

  Project:       NET, 83-101

  ******************************
  *                            *
  *    D_MTP              DLS  *
  *                            *
  ******************************

        Rev History
        Rev.  Date          Name      Change

         1    23jul         DTT       MODS FOR TAPE
         0    13jul1815     DLS       Initial Pseudo code




 Ede_D_MTP  MEND
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                       27 Pseudo_code_D_MTP    MACRO  ;Pseudocode macro area
                       28                    .GOTO Ep_D_MTP
                       29
                       30
                       31
                       32 Ep_D_MTP  MEND
```

```
****************************************************************
*                                                              *
*  MODULE NAME:                                                *
*                                                              *
*    D_MTP                                                     *
*                                                              *
*                                                              *
*  FUNCTION(S):                                                *
*                                                              *
*     1.  TO DECLARE THE DATA AREA "NIM_BLOCK."                *
*     2.  TO DECLARE THE D1_MODE_WORD.                         *
*                                                              *
*  NOTES:                                                      *
*                                                              *
* 1. NIM_BLOCK IS USED AS THE INTERFACE BETWEEN THE            *
*    MEDIUM ACCESS CONTROLLER AND THE RESIDENT APPLICATION     *
*    PROGRAM.                                                  *
*                                                              *
* 2. THE INSTALLER IS RESPONSIBLE FOR LOCATING THIS DATA       *
*    MODULE SO THAT THE LAST BYTE ENDS AT LOCATION 127 (DEC).  *
*                                                              *
*                                                              *
*                                                              *
****************************************************************
```

```
LOCATION OBJECT CODE LINE    SOURCE LINE
                       59
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                       61                GLB     CURRENT_STATE
                       62                GLB     D_MTP
                       63                GLB     D1_MODE_WORD
                       64                GLB     NIM_BLOCK
                       65                GLB     CNFG_WORD
                       66                GLB     A_SIG
                       67                GLB     A_DATA
                       68                GLB     M_SIG
                       69                GLB     M_DATA
                       70 *
                       71                GLB     COUNT
                       72                GLB     NODE_ADDRESS
                       73                GLB     CS_WORD
                       74                GLB     DATA_BUFFER
                       75
                       76                DATA
    0000               77 D_MTP:
                       78 ****************************************************************
                       79 *                                                              *
                       80 *  DATA WORD:                                                  *
                       81 *                                                              *
                       82 *    D1_MODE_WORD                                              *
                       83 *                                                              *
                       84 *   FUNCTION:                                                  *
                       85 *                                                              *
                       86 *    CONTAINS THE STATE OF SEQUENCER PROCESSING                *
                       87 *                                                              *
                       88 *                                                              *
                       89 *                                                              *
                       90 ****************************************************************
    0000               91 CURRENT_STATE  RMB     1
    0001               92 D1_MODE_WORD   RMB     1
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE
                       94
                       95
                       96 *
    0002               97 COUNT                          RMB   2
    0004               98 CS_WORD                        RMB   1
             (0008)    99 NODE_ADDRESS                   EQU   8
                      100
                      101                INCLUDE I_NIM
                        + ******************************************************************
                        + *                                                                *
                        + *                     HIERARCHY  CHART                           *
                        + *                           OF                                   *
                        + *                MEDIUM  ACCESS  CONTROLLER                      *
                        + *                         (MAC)                                  *
                        + *                                                                *
                        + *                                                                *
                        + *                                                                *
                        + *                        (MAC)                                   *
                        + *                          |                                     *
                        + *                          |                                     *
                        + *         +----------------+-----------------------+             *
                        + *       (TR)             (NIM)                   (ACM)           *
                        + *        |                 |                       |             *
                        + *    +---+---+             |                       +             *
                        + * (TRANS)  (REC)           |                     (SEQ)           *
                        + *                          |                       |             *
                        + *               +----------+-----------+           |             *
                        + *               |                      |           |             *
                        + *            (WRITE)               (READ)          |             *
                        + *                                                  |             *
                        + *                                                  |             *
                        + *                                        +---------+---------+   *
                        + *                                        |                   |   *
                        + *                                 (EVENT_PROC)             (RESP)*
                        + *                                                                *
                        + *                                                                *
                        + ******************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                              + ******************************************************************
                              + *             INTERFACE MODULE DESCRIPTION                       *
                              + *             ----------------------------                       *
                              + *                                                                *
                              + *  NAME:                                                         *
                              + *                                                                *
                              + *   I_NIM                                                        *
                              + *                                                                *
                              + *  FUNCTION:                                                     *
                              + *                                                                *
                              + *      TO DEFINE THE INTERFACE BETWEEN THE MAC AND APPLICATION   *
                              + *      WITHIN A NODE. EACH AND EVERY NODE (WHERE A PRINTER       *
                              + *      OR KEYBOARD IS AN EXAMPLE OF A NODE) CONSISTS OF TWO      *
                              + *      PARTS: 1)AN APPLICATION PART, I.E., THE SOFTWARE THAT     *
                              + *      HANDLES THE NODE'S REASON FOR EXISTENCE, AND 2) A MAC     *
                              + *      PART, I.E., THE SOFTWARE THAT INTERFACES TO THE NETWORK.  *
                              + *                                                                *
                              + *  DESCRIPTION:                                                  *
                              + *                                                                *
                              + *      A BLOCK OF MEMORY WILL BE SHARED BY THE MAC AND APP,      *
                              + *      WHEREIN DATA AND CONTROL SIGNALS WILL BE PASSED BACK      *
                              + *      AND FORTH BETWEEN THE TWO. A DIAGRAM OF THIS BLOCK        *
                              + *      (REFERRED TO AS NIM_BLOCK) FOLLOWS:                       *
                              + *                                                                *
                              + *             NIM_BLOCK                                          *
                              + *           +-------------+                                      *
                              + *           | M_SIG       |  A(R/RESET), M(W);                   *
                              + *           +-------------+                                      *
                              + *           | M_DATA      |  A(R/RESET), M(W);                   *
                              + *           +-------------+                                      *
                              + ******************************************************************
+ ******************************************************************
+ *                                                                *
+ *  DATA ELEMENT DEFINITIONS:                                     *
+ *                                                                *
+ *                                                                *
+ *     M_SIG:                                                     *
+ *     -----                                                      *
+ *        0- NO SIGNAL (IDLE).                                    *
+ *       170- A COMMAND IS WAITING FOR THE APPLICATION
+ *       255- RESET                                               *
+ *                                                                *
+ *                                                                *
+ *     M_DATA:                                                    *
+ *     ------                                                     *
+ *                                                                *
+ *       11- READ FROM TAPE                                       *
+ *       12- WRITE TO TAPE                                        *
+ *       'R'-REWIND THE TAPE TO THE LEADER                        *
+ *                                                                *
+ *                                                                *
+ *   NOTES:                                                       *
+ *       1. M:= MAC SIDE OF NODE.                                 *
+ *                                                                *
+ ******************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                         + ***************************************************************
                         + *                                                             *
                         + *  NOTES TO INSTALLER OF THIS MAC/APP:                        *
                         + *                                                             *
                         + *   1. THE APP IS RESPONSIBLE FOR INITIALIZING ALL OF RAM.    *
                         + *                                                             *
                         + *   2. THE APP MUST INITIALIZE THE CONTROL AND STATUS REG AT  *
                         + *      LOCATION 0011.                                         *
                         + *                                                             *
                         + *   3. THE D1_MODE_WORD MUST BE SET TO ZERO AT PWR UP BY THE  *
                         + *      APP.                                                   *
                         + *                                                             *
                         + *   4. THE NIM_BLOCK WILL END AT ADDR 127.                    *
                         + *                                                             *
                         + ***************************************************************
   0005                102 NIM_BLOCK:
   0005                103 CNFG_WORD      RMB     0
   0005                104 A_SIG          RMB     0
   0005                105 A_DATA         RMB     0
                       106
   0005                107 M_SIG          RMB     1
   0006                108 M_DATA         RMB     1
                       109
                       110                COMN
   0000                111 DATA_BUFFER    RMB     1024

Errors=    0
```

```
LINE#   SYMBOL          TYPE     REFERENCES

  105  A_DATA            D  67
  104  A_SIG             D  66
  103  CNFG_WORD         D  65
   97  COUNT             D  71
   98  CS_WORD           D  73
   91  CURRENT_STATE     D  61
   92  D1_MODE_WORD      D  63
  111  DATA_BUFFER       C  74
   77  D_MTP             D  62
  108  M_DATA            D  69
  107  M_SIG             D  68
  102  NIM_BLOCK         D  64
   99  NODE_ADDRESS      A  72
```

```
^6801^
 NAME ^Rev 04 - RPD^

De_MTP_TR_REC MACRO               ;Header Rev. 4
                .GOTO Ede_MTP_TR_REC

 Project:      NET, 83-101

 ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
 ▓                             ▓
 ▓    MTP_TR_REC             DLS   ▓
 ▓                             ▓
 ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓

        Rev History
        Rev.  Date          Name      Change
         4    20jul1155a    RPD       added read of control/status to reset RDRF
         3    20jul755p     RPD       removed LIST directives
         2    19jul2104     JIM       Printer MAC started.
         1    13jul750a     RPD       converted pseudo code to 6801 code
         0    12JUL1305     DLS       Initial Pseudo code

Ede_MTP_TR_REC  MEND
```

```
****************************************************************
*                                                              *
*  MODULE NAME:                                                *
*                                                              *
*    MTP_TR_REC                                                *
*                                                              *
*  INPUTS:                                                     *
*                                                              *
*    NET_BYTE_IN (LOCATION 12)                                 *
*    D1_MODE_WORD                                              *
*                                                              *
*  FUNCTION(S):                                                *
*                                                              *
*     1. TO GET A BYTE FROM THE NETWORK.                       *
*                                                              *
*  OUTPUTS:                                                    *
*                                                              *
*     NET_BYTE_IN (REG_A)                                      *
*     TOKEN :  CARRY SET = BYTE FOR THIS NODE.                 *
*              CARRY CLR = NOT FOR THIS NODE.                  *
*                                                              *
*  CALLS:                                                      *
*                                                              *
*    NONE.                                                     *
*                                                              *
*  CALLED BY:                                                  *
*                                                              *
*    MTP_ACM_SEQ                                               *
*                                                              *
*  NOTES:                                                      *
*                                                              *
*    NONE.                                                     *
*                                                              *
****************************************************************
```

```
*****************************************************************
*                                                               *
*  PSEUDO CODE:                                                 *
*                                                               *
*    MTP_TR_REC:                                                *
*                                                               *
*    CARRY=SET;                                                 *
*    REG_A=MEM(12);                                             *
*    IF D1_MODE_WORD<> CONTROL                                  *
*        THEN                                                   *
*          GOTO REC_RTS;   /* RECEIVING DATA MODE */            *
*    ENDIF;                                                     *
*                                                               *
*       SAVE_REG_A = REG_A;                                     *
*       REG_A = $0F.AND.REG_A; /* LOWER HALF = ADDR */          *
*       IF NODE_ADDR <> REG_A                                   *
*          THEN                                                 *
*            CARRY=0;                                           *
*            GOTO REC_RTS;                                      *
*       ENDIF;                                                  *
*       REG_A=$F0.AND.SAVE_REG_A;/* UPPER HALF = CMND */        *
*       SHIFT REG_A TO LOWER NIBBLE;                            *
*  REC_RTS: RETURN;                                             *
*                                                               *
*****************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                       87                 INCLUDE PG0_EQU
                        + ;
                        + ; 6801 internal register equates (page 0)
                        + ;
          <0000>        + P1_DIR          EQU     000H               ;port 1 data direction register
          <0002>        + P1_DATA         EQU     002H               ;port 1 data register
                        +
          <0001>        + P2_DIR          EQU     001H               ;port 2 data direction register
          <0003>        + P2_DATA         EQU     003H               ;port 2 data register
                        +
          <0004>        + P3_DIR          EQU     004H               ;port 3 data direction register
          <0006>        + P3_DATA         EQU     006H               ;port 3 data register
                        +
          <0005>        + P4_DIR          EQU     005H               ;port 4 data direction register
          <0007>        + P4_DATA         EQU     007H               ;port 4 data register
                        +
          <0008>        + T_CNTLSTAT      EQU     008H               ;timer control and status register
          <0009>        + T_CNTRHGH       EQU     009H               ;counter high byte
          <000A>        + T_CNTRLOW       EQU     00AH               ;counter low byte
          <000B>        + T_OCMPHGH       EQU     00BH               ;output compare register high byte
          <000C>        + T_OCMPLOW       EQU     00CH               ;output compare register low byte
          <000D>        + T_ICAPHGH       EQU     00DH               ;input capture register high byte
          <000E>        + T_ICAPLOW       EQU     00EH               ;input capture register low byte
                        +
          <000F>        + P3_CNTLSTAT     EQU     00FH               ;port 3 control and status register
                        +
          <0010>        + SCI_RM          EQU     010H               ;rate and mode control register
          <0011>        + SCI_TR_CS       EQU     011H               ;transmit/receive control and status register
          <0012>        + SCI_RX          EQU     012H               ;receive data register
          <0013>        + SCI_TX          EQU     013H               ;transmit data register
                        +
          <0014>        + RAM_CNTL        EQU     014H               ;RAM control register
                       88
                       89 ;
                       90 ; local equates
                       91 ;
          <000F>       92 ADDR_MASK       EQU     00FH
          <0008>       93 NODE_ADDR       EQU     008H
          <00F0>       94 CMND_MASK       EQU     0F0H
                       95
          <0040>       96 ORFE            EQU     01000000B
                       97
                       98                 EXT     CURRENT_STATE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      100                PROG
                      101                GLB     MTP_TR_REC
    0000              102 MTP_TR_REC:
    0000 0D           103                SEC                             ;1 TOKEN = BYTE FOR THIS NODE
    0001 D611         104                LDAB    SCI_TR_CS,D
    0003 9612         105                LDAA    SCI_RX,D                ;1 NET_BYTE_IN = SCI_RX
                      106                                                ;1 IF D1_MODE_WORD = CONTROL
    0005 C440         107                ANDB    #ORFE
    0007 2702         108                BEQ     NO_ORFE
                      109
                      110                GLB     BREAK_ORFE
                      111
    0009              112 BREAK_ORFE:
    0009 0C           113                CLC                             ; BAD DATA
                      114
    000A 39           115                RTS
                      116
    000B              117 NO_ORFE:
    000B D600         118                LDAB    CURRENT_STATE,D         ;    get D1_MODE_WORD
    000D 260C         119                BNE     ENDIF_CNTRL
    000F 16           120                TAB                             ;2   SAVE_NBI = NET_BYTE_IN                          /* NOT REC
    0010 840F         121                ANDA    #ADDR_MASK              ;2   ADDRESS = NET_BYTE_IN .AND. ADDR_MASK           /* LOWER
    0012 8108         122                CMPA    #NODE_ADDR              ;2   IF ADDRESS = NODE_ADDR
    0014 2604         123                BNE     ELSE_NOTADDR
    0016 17           124                TBA
                      125                                                ;3     NET_BYTE_IN = SAVE_NBI .AND. CMND_MASK          /* UPPER
    0017 0D           126                SEC
    0018 2001         127                BRA     ENDIF_ADDR
    001A              128 ELSE_NOTADDR:                                  ;2   ELSE
    001A 0C           129                CLC                             ;3     TOKEN = BYTE NOT FOR THIS NODE
    001B              130 ENDIF_ADDR:                                    ;2   ENDIF
    001B              131 ENDIF_CNTRL:                                   ;1 ENDIF
    001B 39           132                RTS

Errors=    0
```

```
LINE#   SYMBOL          TYPE      REFERENCES

   92  ADDR_MASK          A   121
  112  BREAK_ORFE         P   110
   94  CMND_MASK          A
   98  CURRENT_STATE      E   118
  128  ELSE_NOTADDR       P   123
  130  ENDIF_ADDR         P   127
  131  ENDIF_CNTRL        P   119
  102  MTP_TR_REC         P   101
   93  NODE_ADDR          A   122
  117  NO_ORFE            P   108
   96  ORFE               A   107
       SCI_RX             A   105
       SCI_TR_CS          A   104
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                          1 ^6801^
                          3  NAME ^Rev 03 - RPD^
                          4
                          5 De_MTP_TR_TRANS MACRO               ;Header Rev. 4
                          6                 .GOTO Ede_MTP_TR_TRANS
                          7
                          8  Project:      NET, 83-101
                          9
                         10  ******************************
                         11  *                            *
                         12  *    MTP_TR_TRANS         DLS    *
                         13  *                            *
                         14  ******************************
                         15
                         16        Rev History
                         17        Rev.  Date         Name      Change
                         18         3    20jul740p    RPD       removed LIST directives
                         19         2    19jul2053    JIM       Printer MAC started.
                         20         1    13jul835a    RPD       converted pseudo code to 6801 code
                         21         0    12JUL1236    DLS       Initial Pseudo code
                         22
                         23 Ede_MTP_TR_TRANS  MEND
```

```
***************************************************************
*                                                             *
*  MODULE NAME:                                               *
*                                                             *
*    MTP_TR_TRANS                                             *
*                                                             *
*  INPUTS:                                                    *
*                                                             *
*    NET_BYTE_OUT (REG_A)                                     *
*                                                             *
*  FUNCTION(S):                                               *
*                                                             *
*     1. TO SEND A BYTE OUT OVER THE NETWORK.                 *
*                                                             *
*  OUTPUTS:                                                   *
*                                                             *
*     NET_BYTE_OUT (LOCATION 13)                              *
*                                                             *
*  CALLS:                                                     *
*                                                             *
*    NONE.                                                    *
*                                                             *
*  CALLED BY:                                                 *
*                                                             *
*    MTP_ACM_SEQ                                              *
*    MTP_NIM_READ                                             *
*                                                             *
*  NOTES:                                                     *
*                                                             *
*    NONE.                                                    *
*                                                             *
***************************************************************

***************************************************************
*                                                             *
*  PSEUDO CODE:                                               *
*                                                             *
*      MTP_TR_TRANS:                                          *
*                                                             *
*        REPEAT_UNTIL_SET:                                    *
*                                                             *
*             IF MEM(11).5=0 THEN GOTO REPEAT_UNTIL_SET;      *
*             ENDIF;                                          *
*                                                             *
*             MEM(13)=REG_A;                                  *
*                                                             *
*       RETURN;                                               *
*                                                             *
***************************************************************
```

```
LOCATION OBJECT CODE LINE    SOURCE LINE

                       75              INCLUDE PG0_EQU
                        + ;
                        + ; 6801 internal register equates (page 0)
                        + ;
          <0000>        + P1_DIR        EQU     000H                    ;port 1 data direction register
          <0002>        + P1_DATA       EQU     002H                    ;port 1 data register
                        +
          <0001>        + P2_DIR        EQU     001H                    ;port 2 data direction register
          <0003>        + P2_DATA       EQU     003H                    ;port 2 data register
                        +
          <0004>        + P3_DIR        EQU     004H                    ;port 3 data direction register
          <0006>        + P3_DATA       EQU     006H                    ;port 3 data register
                        +
          <0005>        + P4_DIR        EQU     005H                    ;port 4 data direction register
          <0007>        + P4_DATA       EQU     007H                    ;port 4 data register
                        +
          <0008>        + T_CNTLSTAT    EQU     008H                    ;timer control and status register
          <0009>        + T_CNTRHGH     EQU     009H                    ;counter high byte
          <000A>        + T_CNTRLOW     EQU     00AH                    ;counter low byte
          <000B>        + T_OCMPHGH     EQU     00BH                    ;output compare register high byte
          <000C>        + T_OCMPLOW     EQU     00CH                    ;output compare register low byte
          <000D>        + T_ICAPHGH     EQU     00DH                    ;input capture register high byte
          <000E>        + T_ICAPLOW     EQU     00EH                    ;input capture register low byte
                        +
          <000F>        + P3_CNTLSTAT   EQU     00FH                    ;port 3 control and status register
                        +
          <0010>        + SCI_RM        EQU     010H                    ;rate and mode control register
          <0011>        + SCI_TR_CS     EQU     011H                    ;transmit/receive control and status register
          <0012>        + SCI_RX        EQU     012H                    ;receive data register
          <0013>        + SCI_TX        EQU     013H                    ;transmit data register
                        +
          <0014>        + RAM_CNTL      EQU     014H                    ;RAM control register
                       76
                       77 ;
                       78 ; local equates
                       79 ;
          <0020>       80 TDRE_MASK     EQU     020H                    ;"transmit_data_register_empty" mask
                       81
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                         83                PROG
                         84                GLB     MTP_TR_TRANS
   0000                  85 MTP_TR_TRANS:
   0000 3C               86                PSHX                                ; SAVE X JUST IN CASE
   0001 CE0016           87                LDX     #(2*160)/(3+3+3+2+3)        ; ALLOW 2 BYTE TIMES
                         88
   0004                  89 REPEAT:                                 ;T         ;1 REPEAT
   0004 09               90                DEX                      ;3         ; TIME UP ???
   0005 270B             91                BEQ     HAVE_TDRE_ERR    ;3         ; YUP
                         92
                         93                                                    ;2   TDRE = SCI_TR_CS .AND. TDRE_MASK
   0007 D611             94                LDAB    SCI_TR_CS,D      ;3         ;      get the control/status byte
   0009 C420             95                ANDB    #TDRE_MASK       ;2         ;      mask in the TDRE bit
   000B 27F7             96                BEQ     REPEAT           ;3         ;1 UNTIL TDRE = TRUE
                         97 *
   000D 9713             98                STAA    SCI_TX,D                    ;1 SCI_TX = NEXT_BYTE_OUT
                         99
   000F 0C              100                CLC
                        101
   0010 2004            102                BRA     END_TR
                        103
   0012                 104 HAVE_TDRE_ERR:
                        105 ;              CLEAN UP UART PORTS
                        106
                        107                EXT     CLEAN_UART_HW
                        108
   0012 BD0000          109                JSR     CLEAN_UART_HW
                        110
   0015 0D              111                SEC
                        112
   0016                 113 END_TR:
   0016 38              114                PULX
   0017 39              115                RTS

Errors=    0
```

```
 LINE#   SYMBOL         TYPE     REFERENCES

   107  CLEAN_UART_HW    E   109
   113  END_TR           P   102
   104  HAVE_TDRE_ERR    P   91
    85  MTP_TR_TRANS     P   84
    89  REPEAT           P   96
        SCI_TR_CS        A   94
        SCI_TX           A   98
    80  TDRE_MASK        A   95
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

^6801^
 NAME ^Rev 01 - RPD^

De_MTP_TR_TCU MACRO                  ;Header Rev. 4
                .GOTO Ede_MTP_TR_TCU

 Project:      NET, 83-101

 ******************************
 *                            *
 *    MTP_TR_TCU         RPD  *
 *                            *
 ******************************

       Rev History
       Rev.  Date        Name      Change
        1    20jul800p   RPD       created from MTP file
        0    19jul535p   RPD       Initial Pseudo code and code

Ede_MTP_TR_TCU  MEND
```

```
****************************************************************
*                                                              *
*  MODULE NAME:                                                *
*                                                              *
*    MTP_TR_TCU (transmit clean up)                            *
*                                                              *
*  INPUTS:                                                     *
*                                                              *
*    none                                                      *
*                                                              *
*  FUNCTION(S):                                                *
*                                                              *
*     1. Clears the "receive data register full" flag of the  *
*        6801 SCI after a transmission sequence (1 or more    *
*        bytes). The flag is set as a result of sending a byte *
*        out and receiving the same byte in on the common NET *
*        line used for sending and receiving.                  *
*                                                              *
*  OUTPUTS:                                                    *
*                                                              *
*     SCI control/status register bit 7 = 0                    *
*                                                              *
*  CALLS:                                                      *
*                                                              *
*    none                                                      *
*                                                              *
*  CALLED BY:                                                  *
*                                                              *
*    MTP_ACM_R                                                 *
*    (all routines calling MTP_TR_TRANS)                       *
*                                                              *
*  NOTES:                                                      *
*     1 - This sequence follows the procedure described in    *
*         hardware manuals for clearing the flag. Which is:   *
*           step 1) read the SCI control status register      *
*           step 2) read the SCI receive data register        *
*     2 - The MAC modules are responsible for calling this    *
*         module after doing a transmit function to avoid     *
*         reading itself when other data is expected.         *
*                                                              *
****************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                       65 ***************************************************************
                       66 *                                                             *
                       67 *  PSEUDO CODE:                                               *
                       68 *                                                             *
                       69 *    begin                                                    *
                       70 *      wait for TDRE = 1                                      *
                       71 *      clear RDRF (from 2nd to the last byte)                 *
                       72 *      wait for RDRF = 1                                      *
                       73 *      read in the received byte (from very last byte)        *
                       74 *    end                                                      *
                       75 *                                                             *
                       76 ***************************************************************
                       77
                       78              INCLUDE PG0_EQU
                        + ;
                        + ; 6801 internal register equates (page 0)
                        + ;
         <0000>         + P1_DIR       EQU     000H                    ;port 1 data direction register
         <0002>         + P1_DATA      EQU     002H                    ;port 1 data register
                        +
         <0001>         + P2_DIR       EQU     001H                    ;port 2 data direction register
         <0003>         + P2_DATA      EQU     003H                    ;port 2 data register
                        +
         <0004>         + P3_DIR       EQU     004H                    ;port 3 data direction register
         <0006>         + P3_DATA      EQU     006H                    ;port 3 data register
                        +
         <0005>         + P4_DIR       EQU     005H                    ;port 4 data direction register
         <0007>         + P4_DATA      EQU     007H                    ;port 4 data register
                        +
         <0008>         + T_CNTLSTAT   EQU     008H                    ;timer control and status register
         <0009>         + T_CNTRHGH    EQU     009H                    ;counter high byte
         <000A>         + T_CNTRLOW    EQU     00AH                    ;counter low byte
         <000B>         + T_OCMPHGH    EQU     00BH                    ;output compare register high byte
         <000C>         + T_OCMPLOW    EQU     00CH                    ;output compare register low byte
         <000D>         + T_ICAPHGH    EQU     00DH                    ;input capture register high byte
         <000E>         + T_ICAPLOW    EQU     00EH                    ;input capture register low byte
                        +
         <000F>         + P3_CNTLSTAT  EQU     00FH                    ;port 3 control and status register
                        +
         <0010>         + SCI_RM       EQU     010H                    ;rate and mode control register
         <0011>         + SCI_TR_CS    EQU     011H                    ;transmit/receive control and status register
         <0012>         + SCI_RX       EQU     012H                    ;receive data register
         <0013>         + SCI_TX       EQU     013H                    ;transmit data register
                        +
         <0014>         + RAM_CNTL     EQU     014H                    ;RAM control register
                       79 ;
                       80 ; local equate
                       81 ;
         <0020>        82 TDRE_MASK    EQU     020H
                       83
                       84              PROG
                       85              GLB     MTP_TR_TCU
  0000                 86 MTP_TR_TCU:
  0000 3C              87              PSHX
                       88
  0001 CE0022          89              LDX     #(3*160)/(3+3+3+2+3)    ; ALLOW 3 BYTE TIMES
                       90
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

     0004                91 REPEAT:
     0004 09             92                 DEX
     0005 2713           93                 BEQ     TDRE_ERR
                         94
     0007 D611           95                 LDAB    SCI_TR_CS,D
     0009 C420           96                 ANDB    #TDRE_MASK
     000B 27F7           97                 BEQ     REPEAT
                         98
     000D D612           99                 LDAB    SCI_RX,D                ;reset RDRF from 2nd to last byte
                        100
     000F               101 REPEAT1:
     000F 09            102                 DEX
     0010 2708          103                 BEQ     TDRE_ERR
                        104
     0012 D611          105                 LDAB    SCI_TR_CS,D             ;1 WAIT FOR RECEIVE DATA REGISTER FULL
     0014 2AF9          106                 BPL     REPEAT1
                        107
     0016 D612          108                 LDAB    SCI_RX,D                ;1 EMPTY RECEIVED DATA REGISTER AND CLEAR RDRF BIT
                        109
     0018 38            110                 PULX
                        111                                         ;reset RDRF from last byte
     0019 39            112                 RTS
                        113
     001A               114 TDRE_ERR:
                        115 ;               CLEAN UP UART PORTS
     001A 8D02          116                 BSR     CLEAN_UART_HW
                        117
     001C 38            118                 PULX
                        119
     001D 39            120                 RTS
                        121
                        122                 GLB     CLEAN_UART_HW
                        123
     001E               124 CLEAN_UART_HW:
     001E D611          125                 LDAB    011H,D
     0020 D612          126                 LDAB    012H,D
                        127
     0022 C61B          128                 LDAB    #00011011B
     0024 D711          129                 STAB    011H,D
                        130
                        131                 EXT     CURRENT_STATE
                        132
     0026 C600          133                 LDAB    #0
     0028 D700          134                 STAB    CURRENT_STATE,D
                        135
     002A 39            136                 RTS

Errors=    0
```

```
LINE#   SYMBOL          TYPE     REFERENCES

  124  CLEAN_UART_HW     P   116,122
  131  CURRENT_STATE     E   134
   86  MTP_TR_TCU        P   85
   91  REPEAT            P   97
  101  REPEAT1           P   106
       SCI_RX            A   99,108
       SCI_TR_CS         A   95,105
  114  TDRE_ERR          P   93,103
   82  TDRE_MASK         A   96
```

```
^6801^
 NAME ^Rev 02 - DLS^

De_MTP_NIM_WRITE MACRO                ;Header Rev. 4
                .GOTO Ede_MTP_NIM_WRITE

 Project:       NET, 83-101

 ******************************
 *                            *
 *    MTP_NIM_WRITE     DLS   *
 *                            *
 ******************************

        Rev History
        Rev.  Date          Name      Change
         2    15jul2130     DLS       FLIPPED OVFL INTERFACE
         1    13jul130p     RPD       converted pseudo code to 6801 code
         0    12JUL1356     DLS       Initial Pseudo code

Ede_MTP_NIM_WRITE  MEND
```

```
*****************************************************************
*                                                               *
*  MODULE NAME:                                                 *
*                                                               *
*    MTP_NIM_WRITE                                              *
*                                                               *
*  INPUTS:                                                      *
*                                                               *
*    RESET FLAG: CARRY SET = RESET                              *
*                CARRY CLR = NO RESET                           *
*    DATA (REG A) with TAPE COMMAND                             *
*                                                               *
*  FUNCTION(S):                                                 *
*                                                               *
*     1. TO PROVIDE DATA AND SIGNALLING INFORMATION TO THE      *
*        NODE APPLICATION.                                      *
*                                                               *
*  OUTPUTS:                                                     *
*                                                               *
*     M_SIG                                                     *
*     M_DATA                                                    *
*                                                               *
*  CALLS:                                                       *
*                                                               *
*    NONE.                                                      *
*                                                               *
*  CALLED BY:                                                   *
*                                                               *
*    MTP_ACM_SEQ                                                *
*                                                               *
*  NOTES:                                                       *
*                                                               *
*    NONE.                                                      *
*                                                               *
*****************************************************************
```

```
***************************************************************
*                                                             *
*  PSEUDO CODE:                                               *
*                                                             *
*   MTP_NIM_WRITE:                                            *
*                                                             *
*      IF CARRY=SET                                           *
*           THEN                                              *
*            M_SIG=$FF; /* RESET SIGNAL */                    *
*            EXIT;                                            *
*      ENDIF                                                  *
*                                                             *
*     M_DATA = TAPE_COMMAND /*POINTER TO INCOMING DATA/*      *
*     M_SIG  = 00H
*                                                             *
* W_RTS: RETURN;                                              *
*                                                             *
***************************************************************
```

```
79               INCLUDE I_NIM
 + ***************************************************************
 + *                                                             *
 + *                    HIERARCHY  CHART                         *
 + *                          OF                                 *
 + *               MEDIUM  ACCESS  CONTROLLER                    *
 + *                        (MAC)                                *
 + *                                                             *
 + *                                                             *
 + *                                                             *
 + *                       (MAC)                                 *
 + *                         |                                   *
 + *                         |                                   *
 + *        +----------------+---------------------+             *
 + *      (TR)             (NIM)                 (ACM)           *
 + *        |                |                     |             *
 + *    +---+---+            |                     +             *
 + * (TRANS)  (REC)          |                   (SEQ)           *
 + *                         |                     |             *
 + *              +----------+----------+          |             *
 + *              |                     |          |             *
 + *           (WRITE)              (READ)         |             *
 + *                                               |             *
 + *                                               |             *
 + *                                      +--------+---------+   *
 + *                                      |                  |   *
 + *                                (EVENT_PROC)           (RESP)*
 + *                                                             *
 + *                                                             *
 + ***************************************************************
+ *****************************************************************
+ *             INTERFACE MODULE DESCRIPTION                      *
+ *             ----------------------------                      *
+ *                                                               *
+ *  NAME:                                                        *
+ *                                                               *
+ *   I_NIM                                                       *
+ *                                                               *
+ *  FUNCTION:                                                    *
+ *                                                               *
+ *      TO DEFINE THE INTERFACE BETWEEN THE MAC AND APPLICATION  *
+ *      WITHIN A NODE. EACH AND EVERY NODE (WHERE A PRINTER      *
+ *      OR KEYBOARD IS AN EXAMPLE OF A NODE) CONSISTS OF TWO     *
+ *      PARTS: 1)AN APPLICATION PART, I.E., THE SOFTWARE THAT    *
+ *      HANDLES THE NODE'S REASON FOR EXISTENCE, AND 2) A MAC    *
+ *      PART, I.E., THE SOFTWARE THAT INTERFACES TO THE NETWORK. *
+ *                                                               *
+ *  DESCRIPTION:                                                 *
+ *                                                               *
+ *      A BLOCK OF MEMORY WILL BE SHARED BY THE MAC AND APP,     *
+ *      WHEREIN DATA AND CONTROL SIGNALS WILL BE PASSED BACK     *
+ *      AND FORTH BETWEEN THE TWO. A DIAGRAM OF THIS BLOCK       *
+ *      (REFERRED TO AS NIM_BLOCK) FOLLOWS:                      *
+ *                                                               *
+ *             NIM_BLOCK                                         *
+ *           +--------------+                                    *
+ *           | M_SIG        |  A(R/RESET), M(W);                 *
+ *           +--------------+                                    *
+ *           | M_DATA       |  A(R/RESET), M(W);                 *
+ *           +--------------+                                    *
+ *****************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                        + ******************************************************************
                        + *                                                                *
                        + *  DATA ELEMENT DEFINITIONS:                                     *
                        + *                                                                *
                        + *                                                                *
                        + *     M_SIG:                                                     *
                        + *     -----                                                      *
                        + *        0- NO SIGNAL (IDLE).                                    *
                        + *       170- A COMMAND IS WAITING FOR THE APPLICATION
                        + *       255- RESET                                               *
                        + *                                                                *
                        + *                                                                *
                        + *     M_DATA:                                                    *
                        + *     ------                                                     *
                        + *                                                                *
                        + *       11- READ FROM TAPE                                       *
                        + *       12- WRITE TO TAPE                                        *
                        + *       'R'-REWIND THE TAPE TO THE LEADER                        *
                        + *                                                                *
                        + *                                                                *
                        + *   NOTES:                                                       *
                        + *       1. M:= MAC SIDE OF NODE.                                 *
                        + *                                                                *
                        + ******************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                         + ****************************************************************
                         + *                                                              *
                         + *  NOTES TO INSTALLER OF THIS MAC/APP:                         *
                         + *                                                              *
                         + *   1. THE APP IS RESPONSIBLE FOR INITIALIZING ALL OF RAM.     *
                         + *                                                              *
                         + *   2. THE APP MUST INITIALIZE THE CONTROL AND STATUS REG AT   *
                         + *      LOCATION 0011.                                          *
                         + *                                                              *
                         + *   3. THE D1_MODE_WORD MUST BE SET TO ZERO AT PWR UP BY THE   *
                         + *      APP.                                                    *
                         + *                                                              *
                         + *   4. THE NIM_BLOCK WILL END AT ADDR 127.                     *
                         + *                                                              *
                         + ****************************************************************
                         80
                         81 ;
                         82 ; local equates
                         83 ;
          <00FF>         84 RESET           EQU     0FFH
          <00AA>         85 SET             EQU     0AAH
                         86                 EXT     M_SIG,M_DATA
                         87
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                        89                PROG
                        90                GLB     MTP_NIM_WRITE
          <0000>        91 MTP_NIM_WRITE: EQU      $
    0000 2406           92                BCC     NOT_RST                   ;RESET IS FALSE
    0002 86FF           93                LDAA    #RESET
    0004 9700           94                STAA    M_SIG,D
    0006 2006           95                BRA     ENDIF_RST
    0008 9700           96 NOT_RST        STAA    M_DATA,D                  ; SAVE DATA IN
    000A 86AA           97                LDAA    #SET
    000C 9700           98                STAA    M_SIG,D
    000E 39             99 ENDIF_RST:      RTS

Errors=     0
```

```
LINE#   SYMBOL          TYPE     REFERENCES

   99  ENDIF_RST         P   95
   91  MTP_NIM_WRITE     P   90
   86  M_DATA            E   96
   86  M_SIG             E   94,98
   96  NOT_RST           P   92
   84  RESET             A   93
   85  SET               A   97
```

```
^6801^
        NAME    ^Rev 15^

De_TAPE_APP MACRO             ;Header Rev. 4
        .GOTO   Ede_TAPE_APP

 Project:       NET, 83-101

 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                                                         *
 *    TAPE_APP                                   HME       *
 *                                                         *
 *    LINKS  INTO  REV_23                                  *
 *                          8865H                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

       Rev History
       Rev.  Date          Name      Change
       15    83/10/04      HME       RS_READ_BIT RE-TIMED
       14    83/09/31      HME       MOVED A MID-CELL TRANSITION TO THE 31 uSEC POINT
                                     TO PROVIDE A SLIGHTLY INCREASED TOLERANCE TO JITTER
       13    83/09/30      HME       EXTENDED BIT CELL TO 70 uSEC
                                     ADDED MANCHESTER+180 SAMPLING
                                     MOTORS STAY RUNNING AFTER TRANSFER
                                     PULLING TAPE CLEARS CURRENT_RAM
                                     USE CHECK SUM INSTEAD OF CRC_16
                                     BE SMARTER IN CASE OF FORWARD STALL
       12    83/08/18      GRW       ADDED RETRY LOOP DECREMENTS TO FIND_BLOCK
       11    83/08/18      GRW       CHANGED STATE AND POSITION OF CIP SWITCHES
                                        BECAUSE THE DESIGNERS FORGOT TO TELL US
                                     .  ABOUT IT AND WE FOUND OUT THE HARD WAY!!
       10    83/08/18      GRW       MOVED CRC CALC. IN WRITE_BLOCK
        9    83/08/18      HME       OFFLINE CONDITION UPDATES CURRENT_RAM
        8    83/08/18      GRW       ADDED TIMEOUT TO STOP ROUTINES
        7    83/08/17      GRW       CHECK ONLY MOTION0 OR MOTION1 IN READ_STUFF
        6    83/08/17      GRW       ADDED CURRENT_RAM
        5    83/08/17      GRW       REASSIGNED BITS TO ACCOMODATE HARDWARE FIXES
        4    08-05-83      HME       added block 0 lockout and included new working subroutines
        3    83/08/01      GRW & HME general fixes and cleanups
        2    27julnoon     GRW       modified to call real application subroutines
        1    26jul1307     HME       modified to be tape test application
        0    17jul440p     DLS       Initial Pseudo code

Ede_APP_START  MEND
```

```
****************************************************************
*                                                              *
*  MODULE NAME:                                                *
*                                                              *
*    TAPE_APP                                                  *
*                                                              *
*  INPUTS:                                                     *
*                                                              *
*    NONE                                                      *
*                                                              *
*                                                              *
*  FUNCTION(S):                                                *
*                                                              *
*     1. LOOP CHECK NIM BLOCK FOR COMMAND AND EXECUTE          *
*        DIRECTLY INTO KNOWN BUFFER LOCATIONS                  *
*                                                              *
*                                                              *
*                                                              *
*  OUTPUTS:                                                    *
*                                                              *
*     NONE                                                     *
*                                                              *
*  CALLS:                                                      *
*                                                              *
*    NONE                                                      *
*                                                              *
*                                                              *
*  CALLED BY:                                                  *
*                                                              *
*    NO ONE                                                     *
*                                                              *
*  NOTES:                                                      *
*                                                              *
*                                                              *
*                                                              *
****************************************************************
```

```
****************************************************************
*                                                              *
*  PSEUDO CODE:                                                *
*                                                              *
****************************************************************

```

```
           * The drive is connected as follows:
           *
           * Port 1:
           *         bit 0   speed            80 ips when high, 20 ips when low
           *         bit 1   stop0            disables servo on drive 0 when high
           *         bit 2   stop1            disables servo on drive 1 when high
           *         bit 3   go fwd           applies forward drive when low
           *         bit 4   go rev           applies reverse drive when low
           *         bit 5   brake            applies brakes to both drives when high
           *         bit 6   write enable 0   enables drive 0 when low
           *         bit 7   write enable 1   enables drive 1 when low
           * Port 2:
           *         bit 0   write data       data to bo:h drives
           *         bit 1   CIP1             high when cassette is in drive 1
           *         bit 2   track select     1 = track A, 0 = track B
           *         bit 3   transmit data    data out to AdamNet
           *         bit 4   receive data     data in from AdamNet
           * Port 3:
           *         bit 0-7 multiplexed address and data to/from external RAM
           * Port 4:
           *         bit 0   A8               address to external RAM
           *         bit 1   A9               address to external RAM
           *         bit 2   A10              address to external RAM
           *         bit 3   motion0          high when tape is moving in drive 0
           *         bit 4   motion1          high when tape is moving in drive 1
           *         bit 5   CIP0             high when cassette is in drive 0
           *         bit 6   unused           always reads as 1
           *         bit 7   read data        data from drives ORed together

           * DATA STRUCTURE DESCRIPTION.
           *
           * Tape block header:
           *         the block proper is preceded by some zeros and a sync byte
           *         2-byte header id. ( 04757h )
           *         2-byte block number ( 0..max )
           *         one's complement of block number
           *         2-byte max block number -- number of blocks on this track ( origin 1 )
           *         checksum -- one-byte one's complement of sum of all above
           *
           * Block/drive numbers (eg. COMMAND_BUFFER, CURRENT_RAM)
           *         4-byte block number with low byte first
           *         1-byte drive number ( 0 or 1 )

                     GLB     ATP_APP

                     EXT     NIM_BLOCK
                     EXT     CS_WORD
                     EXT     TAPE_STATUS0,TAPE_STATUS1
                     EXT     LENGTH_OF_IO_STATUS
                     EXT     DATA_BUFFER
                     EXT     COMMAND_BUFFER
                     EXT     CURRENT_RAM

<0000>     DDR1      EQU     000H             port 1 data direction
<0001>     DDR2      EQU     001H             port 2 data direction
<0002>     MOTOR     EQU     002H             motor control register and write enables
<0003>     MISC      EQU     003H             write data, track select & CIP1
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                <0005>    148 DDR4     EQU      005H               port 4 data direction
                <0007>    149 STATUS   EQU      007H               port 4 data
                <0008>    150 TCSR     EQU      008H               timer control & status
                <0009>    151 TIMER    EQU      009H               16-bit timer register
                <000B>    152 OCR      EQU      00BH               timer output compare register
                <000F>    153 P3CSR    EQU      00FH               port 3 control & status
                <0010>    154 RMCR     EQU      010H               SCI rate & mode control
                <0011>    155 SCSR     EQU      011H               serial control and status
                <0012>    156 RDATA    EQU      012H               serial receive data
                <0013>    157 TDATA    EQU      013H               serial transmit data
                <0014>    158 RAMCR    EQU      014H               RAM control register
                          159
                <0008>    160 MOTION0 EQU       00001000B          bits in STATUS
                <0010>    161 MOTION1 EQU       00010000B
                <0020>    162 CIP0     EQU      00100000B
                <0040>    163 RDDATA0 EQU       01000000B
                <0080>    164 RDDATA1 EQU       10000000B
                          165
                <0004>    166 TRACK    EQU      00000100B          bits in MISC
                <0002>    167 CIP1     EQU      00000010B
                <0001>    168 WTDATA   EQU      00000001B
                          169
                <007F>    170 WENABLE1 EQU      01111111B          bits in MOTOR
                <00BF>    171 WENABLE0 EQU      10111111B
                <00C0>    172 WDISABLE EQU      11000000B
                <00D4>    173 FWDSLOW0 EQU      11010100B         move tape forward slow
                <00D2>    174 FWDSLOW1 EQU      11010010B
                <00D5>    175 FWDFAST0 EQU      FWDSLOW0.OR.1     move tape forward fast
                <00D3>    176 FWDFAST1 EQU      FWDSLOW1.OR.1
                <00CD>    177 REVFAST0 EQU      11001101B         move tape reverse fast
                <00CB>    178 REVFAST1 EQU      11001011B
                <00F4>    179 FWDSTOP0 EQU      11110100B         stop tape in forward direction
                <00F2>    180 FWDSTOP1 EQU      11110010B
                <00EC>    181 REVSTOP0 EQU      11101100B         stop tape in reverse direction
                <00EA>    182 REVSTOP1 EQU      11101010B
                <00DE>    183 STOPPED EQU       11011110B         both drives idle state
                          184
                <0040>    185 OCF      EQU      01000000B         output compare flag in TCSR
                          186
                <0000>    187 M_SIG    EQU     NIM_BLOCK
                <0001>    188 M_DATA   EQU     M_SIG+1
                <000B>    189 C_READ   EQU      11                 COMMAND TO READ TAPE
                <000C>    190 C_WRITE EQU       12                 WRITE TAPE
                <0052>    191 C_REWIND EQU     82                  ASCII 'R'
                <00AA>    192 C_COMMAND EQU     170                NORMAL DRIVE COMMAND -- CHECK M_DATA
                <00FF>    193 C_RESET EQU       255                COMMAND TO RESET NODE
                <0000>    194 S_OK      EQU     0
                <0001>    195 S_BADBLK EQU      1
                <0002>    196 S_NOBLOCK EQU     2
                <0003>    197 S_NOTAPE EQU      3
                <0004>    198 S_NODRIVE EQU     4
                <0016>    199 SYN       EQU     016H               sync character
                <4757>    200 HEAD_ID EQU      04757H              identification word for block header
                <4845>    201 HEAD_ID2 EQU     04845H              alternate block header for middle directory
                <FFFF>    202 STOP_TIMEOUT EQU 0FFFFH              TIME TO ALLOW MOTORS TO STOP
                          203
                          204 ;
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      205 ; * * * BLOCK 0 LOCKOUT CONSTANT- SET TO 1 TO DISABLE WRITES
                      206 ;
           <0000>     207 DISAB_0 EQU      0
           <0001>     208 CS_MODE EQU      1                 USE CHECK SUMS INSTEAD OF CRC16 CHECK
           <0001>     209 BD_MODE EQU      1                 BLOCK DEFINITION MODE- DIRECTORY IN MIDDLE
                      210
                      211         DATA
                      212
  0000                213 ZERO_BYTE RMB    1                 USED TO WRITE ZERO TO TAPE
  0001                214 SYNC_BYTE RMB    1                 USED TO WRITE SYNC TO TAPE
  0002                215 TEMP     RMB     1                 USED BY CRC ROUTINE
  0003                216 BITCOUNT RMB     1                 COUNTS BITS FOR TAPE AND CRC
  0004                217 STUFF_END RMB    2                 BUFFER END ADDRESS WHEN READING STUFF
                      218
                      219 * THE NEXT 3 VARS ARE USED ONLY BY FIND_BLOCK
  0006                220 DRIVE_NUM RMB    1                 CURRENT DRIVE
  0007                221 TRACK_NUM RMB    1                 CURRENT TRACK
  0008                222 BLOCK_NUM RMB    2                 NEXT BLOCK AVAILABLE
                      223
                      224 * USED FOR MANCHESTER+180 ALGORITHM   [2]
  000A                225 LAST_SEEN RMB 1                     [2]
                      226
                      227 * THE NEXT 3 VARS ARE SET BY CALC_PHYS AND USED BY EVERYBODY
  000B                228 WANTED_DRIVE RMB 1                 DESIRED DRIVE
  000C                229 WANTED_TRACK RMB 1                 DESIRED TRACK NUMBER
  000D                230 WANTED_BLOCK RMB 2                 DESIRED BLOCK NUMBER
                      231
                      232 * USED BY THE INACTIVITY TIMER
  000F                233 SHUT_DOWN RMB    1
                      234
                      235        IF     BD_MODE
                      236 * USED BY THE ALTERNATE FORMAT LOGIC
  0010                237 TAPE_TYPE    RMB 1
                      238        ENDIF
                      239
  0011                240 BLOCKS_TRACK RMB 2                 NUMBER OF BLOCKS PER TRACK
  0013                241 FIND_TRIES RMB  1                  RETRY COUNTER FOR FIND_BLOCK
  0014                242 READ_TRIES RMB  1                    "     "     "  CRC ERRORS
           <00FA>     243 QUIET_TIME EQU  250                # of TICKS AFTER WHICH TO SHUT OFF THE MOTORS  [3]
  0015                244 CRC      RMB     2                 CRC BYTES FOR DATA BLOCKS
           <0017>     245 CRC_END EQU      $
  0017                246 HEAD_BUFFER RMB 9                  BUFFER FOR BLOCK HEADERS
           <0020>     247 HEAD_END EQU     $
  0020                248 MOTION_BIT RMB   1                 FOR USE BY READ_STUFF
  0021                249 STACK_SPACE RMB 30
           <003E>     250 STACK    EQU     $-1               INITIAL STACK POINTER VALUE
                      251
           <0400>     252 BUFFER   EQU     0400H             EXTERNAL RAM BLOCK BUFFER
           <0800>     253 BUFFER_END EQU  BUFFER+1024
                      254
                      255         PROG
                      256
                      257 ***********************************************************************
                      258 * The first thing to do is the stack, SCI and I/O port initialization.
                      259 *
                      260
  0000                261 APP_INIT
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0000                262 ATP_APP
    0000 0F             263          SEI                         SET FOR WHEN WE JUMP HERE
    0001 BE003E         264          LDS       #STACK            INITIALIZE THE STACK POINTER
                        265
    0004 86DE           266          LDAA      #STOPPED          set up the port for no motion or writing
    0006 9702           267          STAA      MOTOR
    0008 86FF           268          LDAA      #11111111B        set up the bit directions
    000A 9700           269          STAA      DDR1
                        270
    000C 8615           271          LDAA      #00010101B        set up bit directions for MISC port
    000E 9701           272          STAA      DDR2
                        273
    0010 8607           274          LDAA      #00000111B        set directions for address/status
    0012 9705           275          STAA      DDR4
                        276
    0014 8604           277          LDAA      #04H              INIT RATE AND MODE
    0016 9710           278          STAA      RMCR              TO 62.5K (rate) AND NRZ (mode)
                        279
    0018 861A           280          LDAA      #1AH              also TE AND RE IN THE TRCS REG (enables and rec. int.)
    001A 9711           281          STAA      SCSR
                        282
    001C                283 CLEAR_RAM
    001C CE00FF         284          LDX       #00FFH            POINT TO TOP OF INTERNAL RAM
    001F                285 REPEAT
    001F 6F00           286          CLR       0,X               CLEAR A BYTE
    0021 09             287          DEX                         DEC THE POINTER
    0022 8C0080         288          CPX       #0080H            ARE WE AT THE BOTTOM?
    0025 24F8           289          BHS       REPEAT            LOOP IF NOT
                        290
    0027 7A0004         291          DEC       CURRENT_RAM+4     INVALIDATE CURRENT_RAM
                        292
                        293
    002A 0E             294          CLI                         ALLOW ADAMNET INTERRUPTS
                        295
    002B 7E014C         296          JMP       INIT_TIMER        TO START, MAKE SURE TIMER GETS SET UP PROPERLY
                        297
                        298 ******************************************************************
                        299 * MAIN_LOOP:  THIS IS THE TAPE APPLICATION.
                        300
    002E                301 MAIN_LOOP
                        302 * FIRST SEE IF INACTIVITY TIMER HAS TIMED OUT     [3]
    002E 7D000F         303          TST       SHUT_DOWN         HAVE WE TURNED OFF THE MOTORS?
    0031 2717           304          BEQ       MOTORS_OKAY       BRANCH IF SO
    0033 8640           305          LDAA      #OCF
    0035 9508           306          BITA      TCSR              SET BIT FOR OUTPUT COMPARE
    0037 2711           307          BEQ       MOTORS_OKAY       ONE mSEC HASN'T OCCURRED
    0039 9608           308          LDAA      TCSR              CLEAR OCR FLAG
    003B DC09           309          LDD       TIMER
    003D C307D0         310          ADDD      #2000             ANOTHER TWO mSEC
    0040 DD0B           311          STD       OCR
    0042 7A000F         312          DEC       SHUT_DOWN
    0045 2603           313          BNE       MOTORS_OKAY       HAS THE WHOLE 500 mSEC ELAPSED?
                        314 * KILL MOTORS.
                        315 * ASSUME THAT WANTED_DRIVE IS STILL CORRECT
    0047 BD026A         316          JSR       STOP_FORWARD
    004A                317 MOTORS_OKAY
    004A 9603           318          LDAA      MISC              SEE IF CASSETTE IN PLACE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    004C 8502        319         BITA    #CIP1
    004E 261B        320         BNE     DR1_OK           BRANCH IF SO -- NO PROBLEMS
    0050 9607        321         LDAA    STATUS           CHECK MOTION BIT NEXT
    0052 8510        322         BITA    #MOTION1
    0054 2706        323         BEQ     CHK1_1           BRANCH IF TAPE OUT
    0056 C604        324         LDAB    #S_NODRIVE       IF MOTION AND NO CASSETTE -- NO DRIVE
    0058 D700        325         STAB    TAPE_STATUS1,D
    005A 2019        326         BRA     CHK0
    005C             327 CHK1_1
    005C C603        328         LDAB    #S_NOTAPE
    005E D700        329         STAB    TAPE_STATUS1,D
    0060 7D000B      330         TST     WANTED_DRIVE,D
    0063 2710        331         BEQ     CHK0              IF WE'RE TALKING TO DRIVE 0, WE DON'T WANT TO TRASH CURRENT_RAM
    0065 86FF        332         LDAA    #255                [4A]
    0067 9704        333         STAA    CURRENT_RAM+4,D     [4A]
    0069 200A        334         BRA     CHK0
    006B             335 DR1_OK
    006B 9600        336         LDAA    TAPE_STATUS1,D   SEE WHAT'S ALREADY REPORTED
    006D 8103        337         CMPA    #S_NOTAPE
    006F 2504        338         BLO     CHK0             DON'T CLOBBER LOW MESSAGES
    0071 C600        339         LDAB    #S_OK
    0073 D700        340         STAB    TAPE_STATUS1,D
    0075             341 CHK0
    0075 9607        342         LDAA    STATUS
    0077 8520        343         BITA    #CIP0            IS THERE A CASSETTE?
    0079 2619        344         BNE     DR0_OK           BRANCH IF SO -- ALL IS WELL
    007B 8508        345         BITA    #MOTION0         IS THERE MOTION?
    007D 2706        346         BEQ     CHK0_1           NO -- SHOW NO TAPE
    007F C604        347         LDAB    #S_NODRIVE       ELSE SHOW THERE IS NO DRIVE
    0081 D700        348         STAB    TAPE_STATUS0,D
    0083 2019        349         BRA     CHK_SIG
    0085             350 CHK0_1
    0085 C603        351         LDAB    #S_NOTAPE        SHOW NO TAPE
    0087 D700        352         STAB    TAPE_STATUS0,D
    0089 7D000B      353         TST     WANTED_DRIVE,D
    008C 2610        354         BNE     CHK_SIG           IF WE'RE NOT TALKING TO DRIVE 0, WE DON'T WANT TO TRASH CURRENT_RAM
    008E 86FF        355         LDAA    #255                [4A]
    0090 9704        356         STAA    CURRENT_RAM+4,D     [4A]
    0092 200A        357         BRA     CHK_SIG
    0094             358 DR0_OK
    0094 9600        359         LDAA    TAPE_STATUS0,D   SEE WHAT'S ALREADY REPORTED
    0096 8103        360         CMPA    #S_NOTAPE
    0098 2504        361         BLO     CHK_SIG          DON'T CLOBBER LOW MESSAGES
    009A C600        362         LDAB    #S_OK
    009C D700        363         STAB    TAPE_STATUS0,D
    009E             364 CHK_SIG
    009E 9600        365         LDAA    M_SIG,D          GET THE MAC'S BYTE
    00A0 278C        366         BEQ     MAIN_LOOP        LOOP IF NOTHING TO DO
                     367
    00A2 0F          368         SEI                      DISABLE SINCE WE'RE PROCESSING
    00A3 81FF        369         CMPA    #C_RESET
    00A5 2718        370         BEQ     EXEC_RESET       BRANCH IF RESET COMMAND
    00A7 BD0334      371         JSR     CALC_PHYS        CONVERT LOGICAL DRIVE/BLOCK TO PHYSICAL
    00AA 2403        372         BCC     MAIN_1           BRANCH IF ALL IS WELL
    00AC 7E0134      373         JMP     NO_BLOCK         ELSE JUMP TO SHOW ERROR
    00AF             374 MAIN_1
    00AF 81AA        375         CMPA    #C_COMMAND
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    00B1 2669            376          BNE     CMD_COMP         BRANCH IF INVALID COMMAND
    00B3 9601            377          LDAA    M_DATA,D         FIND OUT WHAT MAC WANTS
    00B5 810B            378          CMPA    #C_READ
    00B7 2721            379          BEQ     EXEC_R           READ THE TAPE
    00B9 810C            380          CMPA    #C_WRITE
    00BB 2749            381          BEQ     EXEC_W           WRITE THE TAPE
                         382 ;        CMPA    #C_REWIND
                         383 ;        BEQ     EXEC_REW         REWIND THE TAPE
    00BD 265D            384          BNE     CMD_COMP         BRANCH IF INVALID OPERAND
                         385
                         386 ***************************************************
    00BF                 387 EXEC_RESET
    00BF 8600            388          LDAA    #0
    00C1 970B            389          STAA    WANTED_DRIVE,D
    00C3 BD0387          390          JSR     CIP              CHECK FOR TAPE IN DRIVE 0
    00C6 2503            391          BCS     CHECK_1          BRANCH IF NOT
    00C8 BD0210          392          JSR     REWIND           ELSE REWIND IT
    00CB                 393 CHECK_1
    00CB 8601            394          LDAA    #1
    00CD 970B            395          STAA    WANTED_DRIVE,D
    00CF BD0387          396          JSR     CIP              CHECK FOR THE OTHER TAPE
    00D2 2503            397          BCS     CHECK_2          BRANCH IF NOT THERE
    00D4 BD0210          398          JSR     REWIND           ELSE REWIND IT
    00D7                 399 CHECK_2
    00D7 7E0000          400          JMP     APP_INIT
                         401 ******************************************************************
                         402 * THIS ROUTINE JUST REWINDS THE TAPE.
                         403 ;
                         404 ;EXEC_REW
                         405 ;        JSR     CIP              SEE IF THERE'S A CASSETTE
                         406 ;        BCS     NO_CASSETTE      BRANCH IF NO TAPE IN THAT DRIVE
                         407 ;        JSR     REWIND           ELSE REWIND THE TAPE
                         408 ;        BRA     CMD_COMP
                         409 ;
                         410 ******************************************************************
                         411 * THIS ROUTINE READS A BLOCK FROM THE TAPE INTO THE BLOCK BUFFER.
                         412
    00DA                 413 EXEC_R
    00DA BD0387          414          JSR     CIP              CHECK FOR CASSETTE
    00DD 2541            415          BCS     NO_CASSETTE      BRANCH IF IT'S NOT THERE
    00DF 8603            416          LDAA    #3               SET RETRY COUNTER
    00E1 9714            417          STAA    READ_TRIES,D
    00E3                 418 RETRY
    00E3 BD015D          419          JSR     FIND_BLOCK       GO LOOK FOR THE BLOCK
    00E6 254C            420          BCS     NO_BLOCK         BRANCH IF IT ISN'T AROUND
    00E8 BD03B9          421          JSR     READ_BLOCK       ELSE CONTINUE TO READ THE DATA & CRC
    00EB DC00            422          LDD     COMMAND_BUFFER,D COPY COMMAND_BUFFER TO CURRENT_RAM
    00ED DD00            423          STD     CURRENT_RAM,D
    00EF DC02            424          LDD     COMMAND_BUFFER+2,D
    00F1 DD02            425          STD     CURRENT_RAM+2,D
    00F3 9604            426          LDAA    COMMAND_BUFFER+4,D
    00F5 9704            427          STAA    CURRENT_RAM+4,D
                         428          IF      CS_MODE
    00F7 BD02D1          429          JSR     CALC_SUM         CALC THE NEW SUM  [4]
                         430          ELSE
                         431          JSR     CALC_CRC         CALC THE NEW CRC
                         432          ENDIF
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   00FA B30015        433          SUBD    CRC                COMPARE TO READ CRC
   00FD 271D          434          BEQ     CMD_COMP           WE'RE FINISHED IF NO ERROR
   00FF 7A0014        435          DEC     READ_TRIES         ELSE DEC RETRY COUNTER
   0102 26DF          436          BNE     RETRY
   0104 202A          437          BRA     CANT_READ          FAILED AFTER RETRYING CRC ERRORS
                      438
                      439 ****************************************************************
                      440 * THIS ROUTINE WRITES THE CONTENTS OF THE BLOCK BUFFER ONTO THE
                      441 * TAPE.
                      442
   0106               443 EXEC_W
   0106 BD0387        444          JSR     CIP                CHECK FOR CASSETTE
   0109 2515          445          BCS     NO_CASSETTE        BRANCH IF SLOT EMPTY
   010B 86FF          446          LDAA    #255               MAKE CURRENT_RAM INVALID
   010D 9704          447          STAA    CURRENT_RAM+4,D
                      448
                      449 ; BLOCK 0 LOCKOUT CODE- CHANGE DISAB_0 TO ALLOW/DISALLOW WRITES
                      450 *        LDAA    COMMAND_BUFFER,D
                      451 *        ORAA    COMMAND_BUFFER+1,D
                      452 *        ORAA    #1-DISAB_0
                      453 *        BEQ     CMD_COMP           TELL THE POOR SAP THAT IT WORKED, EVEN THOUGH WE DIDN'T TRY
                      454
                      455          IF      CS_MODE
   010F BD02D1        456          JSR     CALC_SUM           CALCULATE THE BLOCK'S SUM [4]
                      457          ELSE
                      458          JSR     CALC_CRC           CALCULATE THE BLOCK'S CRC
                      459          ENDIF
   0112 DD15          460          STD     CRC,D              SAVE IT
                      461
   0114 BD015D        462          JSR     FIND_BLOCK         LOOK FOR THE BLOCK
   0117 2518          463          BCS     NO_BLOCK           BRANCH IF IT ISN'T THERE
   0119 BD04CC        464          JSR     WRITE_BLOCK        ELSE GO WRITE THE DATA & CRC
                      465
                      466 ****************************************************************
                      467 * ALL COMMANDS RETURN HERE WHEN THEY COMPLETE.
                      468
   011C               469 CMD_COMP
   011C 8600          470          LDAA    #S_OK              SHOW NO ERROR
   011E 2016          471          BRA     ERR_COMMON
   0120               472 NO_CASSETTE
                      473 ; COPY COMMAND_BUFFER INTO CURRENT_RAM
   0120 DC00          474          LDD     COMMAND_BUFFER,D
   0122 DD00          475          STD     CURRENT_RAM,D
   0124 DC02          476          LDD     COMMAND_BUFFER+2,D
   0126 DD02          477          STD     CURRENT_RAM+2,D
   0128 9604          478          LDAA    COMMAND_BUFFER+4,D
   012A 9704          479          STAA    COMMAND_BUFFER+4,D
   012C 8603          480          LDAA    #S_NOTAPE          SHOW WE'RE MISSING A TAPE
   012E 2006          481          BRA     ERR_COMMON
   0130               482 CANT_READ
   0130 8601          483          LDAA    #S_BADBLK          SHOW WE CAN'T READ THE BLOCK
   0132 2002          484          BRA     ERR_COMMON
   0134               485 NO_BLOCK
   0134 8602          486          LDAA    #S_NOBLOCK
                      487
   0136               488 ERR_COMMON
   0136 7D000B        489          TST     WANTED_DRIVE,D  WHICH DRIVE ARE WE PLAYING WITH?
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0139 2604          490           BNE     ERR_1              BRANCH IF DRIVE 1
    013B 9700          491           STAA    TAPE_STATUS0,D  PUT THE BYTE FOR DRIVE0
    013D 2002          492           BRA     ERR_END
    013F               493 ERR_1
    013F 9700          494           STAA    TAPE_STATUS1,D  PUT THE BYTE FOR DRIVE1
    0141               495 ERR_END
    0141 7F0000        496           CLR     M_SIG,D            GO IDLE NEXT TIME THROUGH
    0144 9611          497           LDAA    SCSR
    0146 9612          498           LDAA    RDATA
    0148 861B          499           LDAA    #1BH
    014A 9711          500           STAA    SCSR               ; ENABLE RCVR INTRPTS
    014C               501 INIT_TIMER
                       502 * SET UP INACTIVITY TIMER FOR 500 mSECONDS  [3]
    014C 9608          503           LDAA    TCSR               CLEAR TIMER FLAG
    014E DC09          504           LDD     TIMER
    0150 C307D0        505           ADDD    #2000              TWO mSEC
    0153 DD0B          506           STD     OCR
    0155 86FA          507           LDAA    #QUIET_TIME
    0157 970F          508           STAA    SHUT_DOWN,D
                       509
    0159 0E            510           CLI                        RE-ENABLE INTERRUPTS
                       511 * BACK FOR MORE ABUSE
    015A 7E002E        512           JMP     MAIN_LOOP
                       513
                       514 ***********************************************************************
                       515 * This subroutine will try to find the block whose number is in
                       516 * WANTED_BLOCK, whose track number is in WANTED_TRACK, and whose
                       517 * drive number is in WANTED_DRIVE.
                       518 * When the block is found, this returns with the tape in motion, with
                       519 * the head between the header and the data block.  If it can't
                       520 * be found, it returns with the tape stopped and the carry set.
                       521
    015D               522 FIND_BLOCK
    015D 8606          523           LDAA    #6                 ALLOW OURSELVES 6 TRIES TO GET THE BLOCK
    015F 9713          524           STAA    FIND_TRIES,D
    0161               525 FIND_BLOK
    0161 7D0013        526           TST     FIND_TRIES,D     HAVE WE USED UP ALL OUR TRIES?
    0164 2602          527           BNE     FIND_AGAIN         BRANCH IF NOT
    0166 0D            528           SEC                        SHOW AN ERROR
    0167 39            529           RTS
                       530
    0168               531 FIND_AGAIN
    0168 960B          532           LDAA    WANTED_DRIVE,D
    016A 9106          533           CMPA    DRIVE_NUM,D      COMPARE TO CURRENT DRIVE
    016C 2606          534           BNE     SET_VARS           BRANCH IF NOT THE SAME
    016E 960C          535           LDAA    WANTED_TRACK,D
    0170 9107          536           CMPA    TRACK_NUM,D      COMPARE TO CURRENT TRACK NUMBER
    0172 271B          537           BEQ     SAME_TRACK         BRANCH IF THE SAME
                       538
                       539 * If the drive number or track number is different from the last
                       540 * time we were called, we'll have to read a header from that
                       541 * desired drive/track to see where it is positioned.
                       542
    0174               543 SET_VARS
    0174 960B          544           LDAA    WANTED_DRIVE,D  UPDATE THE PARAMETERS WE ALREADY KNOW
    0176 9706          545           STAA    DRIVE_NUM,D
    0178 960C          546           LDAA    WANTED_TRACK,D
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    017A 9707          547          STAA    TRACK_NUM,D
    017C               548 FIND_HEAD
    017C BD03E2        549          JSR     READ_HEADER      READ THE NEXT BLOCK NUMBER
    017F 2403          550          BCC     GOT_HEAD
    0181 7E01BF        551          JMP     FWD_STALL        REWIND & TRY AGAIN IF CAN'T GET HEADER
    0184               552 GOT_HEAD
    0184 DC19          553          LDD     HEAD_BUFFER+2,D LOOK AT THE BLOCK NUMBER WE JUST READ
    0186 930D          554          SUBD    WANTED_BLOCK,D  IS THIS THE ONE WE WANT?
    0188 2602          555          BNE     NOT_IT           BRANCH IF NOT
    018A 0C            556          CLC                      RETURN IF SO
    018B 39            557          RTS
    018C               558 NOT_IT
    018C BD026A        559          JSR     STOP_FORWARD     ELSE STOP THE TAPE
                       560
                       561 * Now we know where that drive/track is positioned.
                       562
    018F               563 SAME_TRACK
    018F DC0D          564          LDD     WANTED_BLOCK,D
    0191 9308          565          SUBD    BLOCK_NUM,D      COMPARE TO NEXT BLOCK
    0193 2602          566          BNE     GO_LOOK          BRANCH IF THIS ISN'T IT
    0195 2061          567          BRA     JUST_AHEAD       BRANCH IF WE'RE THERE
    0197               568 GO_LOOK
    0197 4D            569          TSTA
    0198 2B31          570          BMI     BACKUP           BRANCH IF IT'S BEHIND US
    019A 2606          571          BNE     FORWARD          BRANCH IF IT'S A LONG WAY AHEAD
    019C C105          572          CMPB    #5               IS IT LESS THAN 5 BLOCKS AHEAD?
    019E 2402          573          BHS     FORWARD          BRANCH IF NOT -- MOVE TAPE FAST
    01A0 2056          574          BRA     JUST_AHEAD       ELSE JUST GO READ IT
                       575
    01A2               576 FORWARD
    01A2 830004        577          SUBD    #4               SET TO COME OUT OF HYPERSPACE A LITTLE EARLY
    01A5 BD0257        578          JSR     FAST_FORWARD     START THE TAPE FORWARD
                       579
    01A8               580 FWDLOOP
    01A8 BD0315        581          JSR     SKIP_BLOCK       WAIT WHILE A BLOCK PASSES
    01AB BD03A1        582          JSR     CHECK_MOTION     IS THE TAPE STILL ROLLING?
    01AE 250F          583          BCS     FWD_STALL        BRANCH IF NOT
    01B0 830001        584          SUBD    #1               DEC. THE BLOCK COUNT
    01B3 26F3          585          BNE     FWDLOOP          LOOP UNTIL WE GET THERE
    01B5 BD026A        586          JSR     STOP_FORWARD     STOP THE TAPE
    01B8 7A0013        587          DEC     FIND_TRIES
    01BB 26BF          588          BNE     FIND_HEAD        AND SEE WHERE WE ARE
    01BD 0D            589          SEC
    01BE 39            590          RTS
                       591
    01BF               592 FWD_STALL
    01BF BD026A        593          JSR     STOP_FORWARD     TURN OFF THE MOTORS
    01C2 DC11          594          LDD     BLOCKS_TRACK,D  FIGGER OUT HOW FAR BACK TO GO  [5]
    01C4 930D          595          SUBD    WANTED_BLOCK,D
    01C6 BD0294        596          JSR     FAST_REVERSE
    01C9 2008          597          BRA     REVLOOP
                       598
    01CB               599 BACKUP
    01CB 43            600          COMA                     NEGATE THE VALUE TO GET DISTANCE
    01CC 53            601          COMB
    01CD C30005        602          ADDD    #1+4             (SET IT TO COME OUT OF HYPERSPACE A LITTLE LATE)
    01D0 BD0294        603          JSR     FAST_REVERSE     START THE TAPE REVERSE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      604
    01D3              605 REVLOOP
    01D3 BD0315       606          JSR      SKIP_BLOCK       WAIT WHILE A BLOCK PASSES
    01D6 BD03A1       607          JSR      CHECK_MOTION     IS THE TAPE STILL ROLLING?
    01D9 250F         608          BCS      REV_STALL        BRANCH IF NOT
    01DB 830001       609          SUBD     #1               DEC. THE BLOCK COUNT
    01DE 26F3         610          BNE      REVLOOP          LOOP UNTIL WE GET THERE
    01E0 BD02A7       611          JSR      STOP_REVERSE     STOP THE TAPE
    01E3 7A0013       612          DEC      FIND_TRIES
    01E6 2694         613          BNE      FIND_HEAD        AND SEE WHERE WE ARE
    01E8 0D           614          SEC
    01E9 39           615          RTS
                      616
    01EA              617 REV_STALL
    01EA BD02A7       618          JSR      STOP_REVERSE     TURN OFF THE MOTORS
    01ED CC0000       619          LDD      #0               UPDATE THE BLOCK NUMBER
    01F0 DD08         620          STD      BLOCK_NUM,D
    01F2 7A0013       621          DEC      FIND_TRIES       COUNT THIS AS A TRY
    01F5 7E0161       622          JMP      FIND_BLOK        AND TRY AGAIN
                      623
    01F8              624 JUST_AHEAD
    01F8 BD03E2       625          JSR      READ_HEADER      GET THE NEXT HEADER
    01FB 25C2         626          BCS      FWD_STALL
    01FD DC0D         627          LDD      WANTED_BLOCK,D
    01FF 9319         628          SUBD     HEAD_BUFFER+2,D IS THIS THE BLOCK
    0201 270B         629          BEQ      FOUND_IT         BRANCH IF YES
    0203 2AF3         630          BPL      JUST_AHEAD       LOOP IF IT'S JUST AHEAD
    0205 BD026A       631          JSR      STOP_FORWARD     ELSE WE MISSED IT!!
    0208 7A0013       632          DEC      FIND_TRIES       COUNT THAT AS A TRY
    020B 7E0161       633          JMP      FIND_BLOK        AND TRY AGAIN
                      634
    020E              635 FOUND_IT
    020E 0C           636          CLC
    020F 39           637          RTS
                      638
                      639 **********************************************************************
                      640 * This subroutine rewinds the tape.  It checks the value in WANTED_DRIVE
                      641 * to see which drive is being referred to.  It assumes the tape is stopped
                      642 * when it is called.  It exits with the tape stopped, and it zeroes the
                      643 * BLOCK_NUM. This always disables writing when it starts the motor.
                      644
    0210              645 REWIND
    0210 37           646          PSHB
    0211 36           647          PSHA
    0212 7D000B       648          TST      WANTED_DRIVE,D  WHICH DRIVE?
    0215 2604         649          BNE      REW1
    0217 86CD         650          LDAA     #REVFAST0        run the tape in reverse
    0219 2002         651          BRA      REW
    021B              652 REW1
    021B 86CB         653          LDAA     #REVFAST1
    021D              654 REW
    021D 9702         655          STAA     MOTOR
    021F BD02E2       656          JSR      PAUSE            let the sucker get up to speed
    0222              657 REW2
    0222 BD03A1       658          JSR      CHECK_MOTION     check the motion bit
    0225 24FB         659          BCC      REW2             loop if still moving
    0227 BD02A7       660          JSR      STOP_REVERSE     then stop the drive
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    022A BD02E2        661         JSR     PAUSE            let the bouncing stop
    022D BD02E2        662         JSR     PAUSE
    0230 CC0000        663         LDD     #0               zero the block
    0233 DD08          664         STD     BLOCK_NUM,D
    0235 32            665         PULA
    0236 33            666         PULB
    0237 39            667         RTS
                       668
                       669 *******************************************************************
                       670 * This subroutine starts the tape moving in a forward direction.
                       671 * It assumes the tape is stopped when it is called, but it exits
                       672 * with the tape in motion.  It checks the value of WANTED_DRIVE to
                       673 * determine which drive is in question.  This doesn't alter write enable.
                       674
    0238               675 GO_FORWARD
    0238 36            676         PSHA
    0239 37            677         PSHB
    023A 7D000B        678         TST     WANTED_DRIVE,D
    023D 2604          679         BNE     GOF1
    023F 86D4          680         LDAA    #FWDSLOW0        tell the drive to move the tape
    0241 2002          681         BRA     GOF2
    0243               682 GOF1
    0243 86D2          683         LDAA    #FWDSLOW1
    0245               684 GOF2
    0245 D602          685         LDAB    MOTOR
    0247 C4C0          686         ANDB    #11000000B       PRESERVE WRITE ENABLES
    0249 843F          687         ANDA    #00111111B
    024B 1B            688         ABA                      MIX OLD ENABLES WITH NEW MOTORS
    024C 9702          689         STAA    MOTOR
    024E BD02EB        690         JSR     PAUSE100         let the tape get up to speed
    0251 BD02EB        691         JSR     PAUSE100
    0254 33            692         PULB
    0255 32            693         PULA
    0256 39            694         RTS
                       695
                       696 *******************************************************************
                       697 * This subroutine starts the tape moving fast in a forward direction.
                       698 * It assumes the tape is stopped when it is called, but it exits
                       699 * with the tape in motion.  This always disables writing.
                       700
    0257               701 FAST_FORWARD
    0257 36            702         PSHA
    0258 7D000B        703         TST     WANTED_DRIVE,D
    025B 2604          704         BNE     FASTF1
    025D 86D5          705         LDAA    #FWDFAST0        tell the drive to move the tápe
    025F 2002          706         BRA     FASTF
    0261               707 FASTF1
    0261 86D3          708         LDAA    #FWDFAST1
    0263               709 FASTF
    0263 9702          710         STAA    MOTOR
    0265 BD02EB        711         JSR     PAUSE100         let the tape get partly up to speed
    0268 32            712         PULA
    0269 39            713         RTS
                       714
                       715 *******************************************************************
                       716 * This routine brings the tape to a halt from the forward direction.
                       717 * It assumes the tape is in motion forward when it is called, and
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE
                          * exits with the tape stopped.  This always disables writing.

  026A                    STOP_FORWARD
  026A 36                          PSHA
  026B 37                          PSHB
  026C 3C                          PSHX
  026D CEFFFF                      LDX     #STOP_TIMEOUT   INIT TIMEOUT COUNTER
  0270 7D000B                      TST     WANTED_DRIVE,D  ELSE SEE WHICH DRIVE WE'RE USING
  0273 2606                        BNE     SF1             BRANCH IF USING DRIVE 0
  0275 86F4                        LDAA    #FWDSTOP0       ELSE SET FOR DRIVE 0
  0277 C608                        LDAB    #MOTION0
  0279 2004                        BRA     SF
  027B                    SF1
  027B 86F2                        LDAA    #FWDSTOP1       SET FOR DRIVE 1
  027D C610                        LDAB    #MOTION1
  027F                    SF
  027F D507                        BITB    STATUS          IS THE DRIVE ALREADY STOPPED?
  0281 2709                        BEQ     SF_OK           BRANCH IF SO
  0283 9702                        STAA    MOTOR           ELSE APPLY THE BRAKES
  0285                    STOPFWAIT
  0285 D507                        BITB    STATUS          CHECK THE MOTION BIT
  0287 2703                        BEQ     SF_OK           BRANCH IF IT IS STOPPED
  0289 09                          DEX                     DEC. TIMEOUT
  028A 26F9                        BNE     STOPFWAIT       LOOP IF NOT TIMED OUT YET
  028C                    SF_OK
  028C 86DE                        LDAA    #STOPPED        then set everything to idle state
  028E 9702                        STAA    MOTOR
  0290 38                          PULX
  0291 33                          PULB
  0292 32                          PULA
  0293 39                          RTS

                          ***********************************************************************
                          * This subroutine starts the tape moving fast in a reverse direction.
                          * It assumes the tape is stopped when it is called, but it exits
                          * with the tape in motion.  This always disables writing.

  0294                    FAST_REVERSE
  0294 36                          PSHA
  0295 7D000B                      TST     WANTED_DRIVE,D
  0298 2604                        BNE     FASTR1
  029A 86CD                        LDAA    #REVFAST0       tell the drive to move the tape
  029C 2002                        BRA     FASTR
  029E                    FASTR1
  029E 86CB                        LDAA    #REVFAST1
  02A0                    FASTR
  02A0 9702                        STAA    MOTOR
  02A2 BD02EB                      JSR     PAUSE100        let the tape get partly up to speed
  02A5 32                          PULA
  02A6 39                          RTS

                          ***********************************************************************
                          * This routine brings the tape to a halt from the reverse direction.
                          * It assumes the tape is in motion forward when it is called, and
                          * exits with the tape stopped.  This always disables writing.

  02A7                    STOP_REVERSE
```

```
02A7 36                     PSHA
02A8 37                     PSHB
02A9 3C                     PSHX
02AA CEFFFF                 LDX     #STOP_TIMEOUT
02AD 7D000B                 TST     WANTED_DRIVE,D
02B0 2606                   BNE     SR1             BRANCH IF USING DRIVE 0
02B2 86EC                   LDAA    #REVSTOP0       ELSE SET FOR DRIVE 0
02B4 C608                   LDAB    #MOTION0
02B6 2004                   BRA     SR
02B8                SR1
02B8 86EA                   LDAA    #REVSTOP1       SET FOR DRIVE 1
02BA C610                   LDAB    #MOTION1
02BC                SR
02BC D507                   BITB    STATUS          IS THE TAPE ALREADY STOPPED?
02BE 2709                   BEQ     SR_OK           BRANCH IF SO
02C0 9702                   STAA    MOTOR           ELSE APPLY THE BRAKES
02C2                STOPRWAIT
02C2 D507                   BITB    STATUS          CHECK THE MOTION BIT
02C4 2703                   BEQ     SR_OK           BRANCH IF IT IS STOPPED
02C6 09                     DEX                     DEC. TIMEOUT COUNTER
02C7 26F9                   BNE     STOPRWAIT       LOOP IF WE HAVE TIME LEFT
02C9                SR_OK
02C9 86DE                   LDAA    #STOPPED        then set everything to idle state
02CB 9702                   STAA    MOTOR
02CD 38                     PULX
02CE 33                     PULB
02CF 32                     PULA
02D0 39                     RTS

                            IF      CS_MODE
                    ************************************************************************
                    * This routine calculates the sum of the data in the 1k buffer and
                    * returns it in the D register.  The 2 byte buffer (same as the one
                    * used for CRC calculations) is allowed to overflow
                    *
02D1                CALC_SUM
02D1 CC0000                 LDD     #0
02D4 CE0400                 LDX     #BUFFER
02D7                CALC_S2
02D7 EB00                   ADDB    0,X
02D9 8900                   ADCA    #0
02DB 08                     INX
02DC 8C0800                 CPX     #BUFFER_END
02DF 26F6                   BNE     CALC_S2
02E1 39                     RTS

                            ELSE
                    ************************************************************************
                    * This routine calculates the CRC of the data in the 1K buffer and
                    * returns it in the D register.
                    * The algorithm used here calculates CRC16.  The memory buffer is
                    * looked at bit by bit.  For each bit, we EOR it with the bottom
                    * bit of the CRC register.  The result is then EORed with bits
                    * 14 and 1 of the CRC register.  Finally, the CRC register is
                    * shifted right, with the calculated bit being shifted into the
                    * top of the register.

```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     832 CALC_CRC
                     833          LDD     #0             INIT THE CRC
                     834          LDX     #BUFFER        INIT THE BUFFER POINTER
                     835 CRC_BYTE
                     836          PSHA                   GET THE BYTE FROM THE BUFFER
                     837          LDAA    0,X
                     838          STAA    TEMP,D
                     839          LDAA    #8             INIT THE BIT COUNT
                     840          STAA    BITCOUNT,D
                     841          PULA
                     842 CRC_BIT
                     843          PSHB                   EOR TEMP(7) AND REGB(0) INTO CARRY
                     844          LSL     TEMP
                     845          ADCB    #0
                     846          LSRB
                     847          PULB
                     848          BCC     CRC_SHIFT      BRANCH IF RESULT IS ZERO
                     849          EORA    #01000000B     ELSE EOR SOME CRC BITS
                     850          EORB    #00000010B
                     851 CRC_SHIFT
                     852          RORA                   SHIFT CRC, BRING IN NEW TOP BIT
                     853          RORB
                     854          DEC     BITCOUNT       DONE ALL BITS?
                     855          BNE     CRC_BIT        LOOP IF NOT
                     856          INX                    ELSE POINT TO NEXT BYTE
                     857          CPX     #BUFFER_END    ARE WE DONE ALL BYTES?
                     858          BNE     CRC_BYTE       LOOP IF NOT
                     859          RTS
                     860          ENDIF
                     861
                     862 ****************************************************************************
                     863 * This routine just kills some time.
                     864
  02E2               865 PAUSE
  02E2 3C            866          PSHX
  02E3 CEFFFF        867          LDX     #0FFFFH
  02E6               868 PSE1
  02E6 09            869          DEX
  02E7 26FD          870          BNE     PSE1
  02E9 38            871          PULX
  02EA 39            872          RTS
                     873
                     874 ****************************************************************************
                     875 * This routine pauses for 100 milliseconds to let the tape get up
                     876 * to 20 ips.
                     877
  02EB               878 PAUSE100
  02EB 8D00          879          BSR     PAUSE50
  02ED               880 PAUSE50
  02ED 37            881          PSHB
  02EE 36            882          PSHA
  02EF 9608          883          LDAA    TCSR           READ THIS TO CLEAR FLAG JUST IN CASE
  02F1 DC09          884          LDD     TIMER          GET CURRENT TIMER VALUE
  02F3 C3C350        885          ADDD    #50000         ADD 50 MSEC
  02F6 DD0B          886          STD     OCR            PUT RESULT INTO COMPARE REG.
  02F8 8640          887          LDAA    #OCF           SET BIT TO CHECK FOR OUTPUT COMPARE
  02FA               888 PAUSE50WAIT
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

  02FA 9508          889          BITA    TCSR
  02FC 27FC          890          BEQ     PAUSE50WAIT       WAIT FOR OC FLAG
  02FE 32            891          PULA
  02FF 33            892          PULB
  0300 39            893          RTS
                     894
                     895 ***********************************************************************
                     896 * This routine pauses for 1 millisecond (1000 microseconds).  It can
                     897 * be used to lengthen the gap when writing.
                     898
  0301               899 PAUSE1
  0301 37            900          PSHB
  0302 36            901          PSHA
  0303 9608          902          LDAA    TCSR              READ THIS TO CLEAR FLAG JUST IN CASE
  0305 DC09          903          LDD     TIMER             GET CURRENT TIMER VALUE
  0307 C303E8        904          ADDD    #1000             ADD 1 MSEC
  030A DD0B          905          STD     OCR               PUT RESULT INTO COMPARE REG.
  030C 8640          906          LDAA    #OCF              SET BIT TO CHECK FOR OUTPUT COMPARE
  030E               907 PAUSE1WAIT
  030E 9508          908          BITA    TCSR
  0310 27FC          909          BEQ     PAUSE1WAIT        WAIT FOR OC FLAG
  0312 32            910          PULA
  0313 33            911          PULB
  0314 39            912          RTS
                     913
                     914 ***********************************************************************
                     915 * This routine pauses for the length of time that it takes one block
                     916 * to pass under the head at ~90 ips.
                     917 * 15000 BITS @ 714.3 bpi = 21.00 in.
                     918 * At 90 ips, 21.00 in. travels by in 0.222222 sec.
                     919 * 10/4 IT DROPPED OUT TOO SOON- ADDED A LITTLE BIT MORE
                     920
  0315               921 SKIP_BLOCK
  0315 36            922          PSHA
  0316 37            923          PSHB
  0317 3C            924          PSHX
  0318 CE0007        925          LDX     #7
  031B               926 SKIP_LOOP
  031B 8D07          927          BSR     SKIP
  031D 09            928          DEX
  031E 26FB          929          BNE     SKIP_LOOP
  0320 38            930          PULX
  0321 33            931          PULB
  0322 32            932          PULA
  0323 39            933          RTS
                     934
  0324               935 SKIP
  0324 9608          936          LDAA    TCSR              READ THIS TO CLEAR FLAG JUST IN CASE
  0326 DC09          937          LDD     TIMER             GET CURRENT TIMER VALUE
  0328 C37D00        938          ADDD    #32000            ADD THE NECESSARY TIME
  032B DD0B          939          STD     OCR               PUT RESULT INTO COMPARE REG.
  032D 8640          940          LDAA    #OCF              SET BIT TO CHECK FOR OUTPUT COMPARE
  032F               941 SKIPWAIT
  032F 9508          942          BITA    TCSR
  0331 27FC          943          BEQ     SKIPWAIT          WAIT FOR OC FLAG
  0333 39            944          RTS
                     945
```

```
           ************************************************************************
           * This routine converts the logical block number in the command buffer
           * to a physical track & block number in WANTED_TRACK and WANTED_BLOCK.

0334       CALC_PHYS
0334 37              PSHB
0335 36              PSHA
0336 9604            LDAA    COMMAND_BUFFER+4,D COPY THE DRIVE NUMBER OVER
0338 970B            STAA    WANTED_DRIVE,D
033A DC11            LDD     BLOCKS_TRACK,D  CHECK BLOCKS PER TRACK FOR VALIDITY
033C 2608            BNE     CALC_OK         BRANCH IF IT LOOKS OK
033E BD03E2          JSR     READ_HEADER     ELSE GET A REAL NUMBER FROM EITHER TRACK
0341 2525            BCS     CALC_BAD        BRANCH IF WE CAN'T
0343 BD026A          JSR     STOP_FORWARD
0346       CALC_OK
0346 9601            LDAA    COMMAND_BUFFER+1,D GET THE DESIRED BLOCK
0348 D600            LDAB    COMMAND_BUFFER,D
034A 9311            SUBD    BLOCKS_TRACK,D  IS IT ON TRACK ZERO?
034C 2410            BHS     CALC1           BRANCH IF NOT
034E 9601            LDAA    COMMAND_BUFFER+1,D ELSE GET THE BLOCK AGAIN
0350 D600            LDAB    COMMAND_BUFFER,D
0352 DD0D            STD     WANTED_BLOCK,D  AND SET THE BLOCK
                     IF      BD_MODE
0354 BD036F          JSR     MANGLE_NUM      RE-MAP BLOCK# TO ACTUAL #
                     ENDIF
0357 7F000C          CLR     WANTED_TRACK    AND CLEAR THE TRACK
035A 32              PULA
035B 33              PULB
035C 0C              CLC
035D 39              RTS
035E       CALC1
035E DD0D            STD     WANTED_BLOCK,D  SET THE BLOCK MINUS THE EXCESS
0360 8601            LDAA    #1
0362 970C            STAA    WANTED_TRACK,D  AND SET THE TRACK
0364 32              PULA
0365 33              PULB
0366 0C              CLC
0367 39              RTS
0368       CALC_BAD
0368 BD026A          JSR     STOP_FORWARD
036B 32              PULA
036C 33              PULB
036D 0D              SEC
036E 39              RTS

                     IF      BD_MODE
           ************************************************************************
           * This routine handles the re-mapping of BD block numbers to real-live
           * useful block numbers. Currently, we just add BLOCKS_TRACK/2 to the
           * number, and wrap back to 0 on overflow
036F       MANGLE_NUM
036F 7D0010          TST     TAPE_TYPE,D     SEE WHERE THE DIRECTORY IS -
0372 2712            BEQ     MANGL_END       AT BEGINNING. GO AWAY.
0374 DC11            LDD     BLOCKS_TRACK,D
0376 04              LSRD                    DIVIDE BY 2
0377 D30D            ADDD    WANTED_BLOCK,D
0379 DD0D            STD     WANTED_BLOCK,D  SAVE IN CASE WE'RE DONE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    037B 9311           1003           SUBD     BLOCKS_TRACK,D  HAVE WE REQUESTED A NON-EXISTENT BLOCK?
    037D 2401           1004           BHS      MANGL_HI
    037F 39             1005           RTS                      WE'RE OKAY. JUST RETURN
    0380                1006 MANGL_HI
    0380 DC0D           1007           LDD      WANTED_BLOCK,D  SUBTRACT BLOCKS_TRACK TO OFFSET BACK
    0382 9311           1008           SUBD     BLOCKS_TRACK,D
    0384 DD0D           1009           STD      WANTED_BLOCK,D  I HOPE YOU'RE HAPPY NOW, BOZO
    0386                1010 MANGL_END
    0386 39             1011           RTS
                        1012           ENDIF
                        1013
                        1014 ********************************************************************
                        1015 * This routine sees if the drive indicated by the command buffer contains a
                        1016 * cassette.  It returns with the carry clear if it does, and set
                        1017 * if it doesn't.
                        1018
    0387                1019 CIP
    0387 36             1020           PSHA
    0388 7D000B         1021           TST      WANTED_DRIVE,D  LOOK AT THE DRIVE NUMBER
    038B 2608           1022           BNE      CIP_1           BRANCH IF DRIVE 1
    038D 9607           1023           LDAA     STATUS          GET THE DRIVE 0 BIT
    038F 8520           1024           BITA     #CIP0           TEST IT
    0391 270B           1025           BEQ      CIP_9           BRANCH IF IT'S NOT THERE
    0393 2606           1026           BNE      CIP_8           BRANCH IF IT'S THERE
    0395                1027 CIP_1
    0395 9603           1028           LDAA     MISC            GET THE DRIVE 1 BIT
    0397 8502           1029           BITA     #CIP1           TEST IT
    0399 2703           1030           BEQ      CIP_9           BRANCH IF IT'S NOT THERE
    039B                1031 CIP_8
    039B 32             1032           PULA
    039C 0C             1033           CLC
    039D 39             1034           RTS
    039E                1035 CIP_9
    039E 32             1036           PULA
    039F 0D             1037           SEC
    03A0 39             1038           RTS
                        1039
                        1040 ********************************************************************
                        1041 * This routine looks to see if the drive indicated by WANTED_DRIVE is
                        1042 * in motion or not.  It returns the carry clear if there is motion,
                        1043 * and set if not.
                        1044
    03A1                1045 CHECK_MOTION
    03A1 36             1046           PSHA
    03A2 9607           1047           LDAA     STATUS          GET THE MOTION BITS
    03A4 7D000B         1048           TST      WANTED_DRIVE,D
    03A7 2606           1049           BNE      CM1             BRANCH FOR DRIVE 1
    03A9 8508           1050           BITA     #MOTION0        CHECK HERE FOR DRIVE 0
    03AB 2706           1051           BEQ      CM2             BRANCH IF NO MOTION
    03AD 2007           1052           BRA      CM3             BRANCH IF TAPE IS ROLLING
    03AF                1053 CM1
    03AF 8510           1054           BITA     #MOTION1        CHECK HERE FOR DRIVE 1
    03B1 2603           1055           BNE      CM3             BRANCH IF TAPE IS ROLLING
    03B3                1056 CM2
    03B3 0D             1057           SEC                      SHOW NO MOTION
    03B4 32             1058           PULA
    03B5 39             1059           RTS
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

03B6               CM3
03B6 0C                      CLC                         SHOW MOTION
03B7 32                      PULA
03B8 39                      RTS

                   ************************************************************************
                   * This subroutine reads a block of data from tape into the buffer.
                   * It assumes the tape is in the gap between the header and the data
                   * when it is called, and exits with the tape stopped.

03B9               READ_BLOCK
03B9 CE0800                  LDX     #BUFFER_END         INIT THE END POINTER
03BC DF04                    STX     STUFF_END,D
03BE CE0400                  LDX     #BUFFER             INIT THE START POINTER
03C1 BD0440                  JSR     READ_STUFF          READ THE BLOCK
03C4 CE0017                  LDX     #CRC_END            INIT END POINTER AGAIN
03C7 DF04                    STX     STUFF_END,D
03C9 CE0015                  LDX     #CRC                INIT START POINTER AGAIN
03CC BD0440                  JSR     READ_STUFF          READ THE CRC BYTES
03CF BD03A1                  JSR     CHECK_MOTION        SEE IF THE TAPE JAMMED
03D2 2509                    BCS     RB_ERROR            BRANCH IF SO
                   *         JSR     STOP_FORWARD        ELSE STOP THE TAPE * FACE *
                             IF      CS_MODE
03D4 BD02D1                  JSR     CALC_SUM            [4]
                             ELSE
                             JSR     CALC_CRC            GET THE CRC
                             ENDIF
03D7 9315                    SUBD    CRC,D               COMPARE IT TO THE ONE WE READ
03D9 2605                    BNE     RB_ERROR2           BRANCH IF NOT A MATCH
                   ;         LDD     WANTED_BLOCK
                   ;         STD     HAVE_BLOCK
03DB 0C                      CLC                         SHOW NO ERROR
03DC 39                      RTS

03DD               RB_ERROR
03DD BD026A                  JSR     STOP_FORWARD        TURN OFF THE MOTORS
03E0               RB_ERROR2
03E0 0D                      SEC                         SHOW THERE WAS A JAM
03E1 39                      RTS

                   ************************************************************************
                   * This routine reads the next block header from tape into the header buffer.
                   * It assumes the tape is stopped when it is called, and exits with
                   * the tape moving and in the gap between the header and the data.
                   * If there was no trouble, the carry is clear.  If it finds that the
                   * tape jammed while it was reading, it returns with the carry set.

03E2               READ_HEADER
03E2 BD0238                  JSR     GO_FORWARD
03E5               READ_H2
03E5 BD03A1                  JSR     CHECK_MOTION        SEE IF THE TAPE IS REALLY MOVING
03E8 2551                    BCS     RH_STALLED          BRANCH IF NOT
03EA CE0020                  LDX     #HEAD_END           SET THE END ADDRESS
03ED DF04                    STX     STUFF_END,D
03EF CE0017                  LDX     #HEAD_BUFFER        SET THE START ADDRESS
03F2 BD0440                  JSR     READ_STUFF          READ THE HEADER
03F5 2544                    BCS     RH_STALLED          BRANCH IF THE TAPE JAMMED
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1117
                     1118 * Now that we have read some data, let's see if it really was a
                     1119 * block header.  If so, the first two bytes should be the block
                     1120 * identifier, the third byte should be the complement of the
                     1121 * fifth, the fourth should be the complement of the sixth,
                     1122 * and the sum of all 9 of them should be -1.
                     1123
                     1124          IF      BD_MODE
  03F7 7F0010        1125          CLR     TAPE_TYPE,D
                     1126          ENDIF
  03FA FC0017        1127          LDD     HEAD_BUFFER      GET THE FIRST TWO BYTES
  03FD 834757        1128          SUBD    #HEAD_ID         IS THIS A HEADER?
                     1129          IF      BD_MODE
  0400 270A          1130          BEQ     VALID_HEAD
                     1131          ELSE
                     1132          BNE     READ_H2          TRY AGAIN IF NOT RIGHT
                     1133          ENDIF
                     1134 * TRY AGAIN- USE ALTERNATE HEAD_ID
                     1135          IF      BD_MODE
  0402 DC17          1136          LDD     HEAD_BUFFER,D
  0404 834845        1137          SUBD    #HEAD_ID2
  0407 26DC          1138          BNE     READ_H2
  0409 7C0010        1139          INC     TAPE_TYPE,D
                     1140          ENDIF
  040C               1141 VALID_HEAD
  040C DC19          1142          LDD     HEAD_BUFFER+2,D CHECK THE COMPLEMENTARY BYTES
  040E 43            1143          COMA
  040F 53            1144          COMB
  0410 931B          1145          SUBD    HEAD_BUFFER+4,D
  0412 26D1          1146          BNE     READ_H2          TRY AGAIN IF WRONG
  0414 9617          1147          LDAA    HEAD_BUFFER,D    CALCULATE THE SUM
  0416 9B18          1148          ADDA    HEAD_BUFFER+1,D
  0418 9B19          1149          ADDA    HEAD_BUFFER+2,D
  041A 9B1A          1150          ADDA    HEAD_BUFFER+3,D
  041C 9B1B          1151          ADDA    HEAD_BUFFER+4,D
  041E 9B1C          1152          ADDA    HEAD_BUFFER+5,D
  0420 9B1D          1153          ADDA    HEAD_BUFFER+6,D
  0422 9B1E          1154          ADDA    HEAD_BUFFER+7,D
  0424 9B1F          1155          ADDA    HEAD_BUFFER+8,D
  0426 4C            1156          INCA
  0427 26BC          1157          BNE     READ_H2          BRANCH IF SUM IS WRONG
                     1158
                     1159 * As a courtesy to the other subroutines, we will put the number
                     1160 * of the next block into BLOCK_NUM and the number of blocks per
                     1161 * track into BLOCKS_TRACK.
                     1162
  0429 DC19          1163          LDD     HEAD_BUFFER+2,D
  042B C30001        1164          ADDD    #1
  042E DD08          1165          STD     BLOCK_NUM,D
                     1166
  0430 DC1D          1167          LDD     HEAD_BUFFER+6,D
  0432 DD11          1168          STD     BLOCKS_TRACK,D
                     1169
  0434 BD03A1        1170          JSR     CHECK_MOTION     SEE IF THE TAPE JAMMED WHILE WE WERE BUSY
  0437 2502          1171          BCS     RH_STALLED       BRANCH IF SO
  0439 0C            1172          CLC                      SHOW NO JAM
  043A 39            1173          RTS
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1174
    043B             1175 RH_STALLED
    043B BD026A      1176         JSR     STOP_FORWARD     TURN OFF THE MOTORS
    043E 0D          1177         SEC                      SHOW THERE WAS A JAM
    043F 39          1178         RTS
                     1179
                     1180 *****************************************************************
                     1181 * This routine will read a block of stuff (file header, data block,
                     1182 * or CRC bytes) from a drive.  It should be called with the start
                     1183 * memory buffer address in X and the end address plus 1 in STUFF_END.
                     1184
    0440             1185 READ_STUFF
                     1186
                     1187 * FIRST WE MUST SET THE TRACK NUMBER.
    0440 9603        1188         LDAA    MISC             GET CURRENT STATE
    0442 84FB        1189         ANDA    #0FFH-TRACK      ASSUME WE WANT TRACK ZERO
    0444 7D000C      1190         TST     WANTED_TRACK,D   SEE IF WE WERE RIGHT
    0447 2702        1191         BEQ     TK_OK            BRANCH IF SO
    0449 8A04        1192         ORAA    #TRACK           ELSE CHOOSE TRACK 1
    044B             1193 TK_OK
    044B 9703        1194         STAA    MISC
                     1195
                     1196 * THEN WE SET THE MOTION BIT TO WATCH.
    044D 8608        1197         LDAA    #MOTION0         ASSUME IT WILL BE DRIVE 0
    044F 7D000B      1198         TST     WANTED_DRIVE,D
    0452 2702        1199         BEQ     DR_OK            BRANCH IF WE WERE RIGHT
    0454 8610        1200         LDAA    #MOTION1         ELSE CHANGE OUR MIND(S)
    0456             1201 DR_OK
    0456 9720        1202         STAA    MOTION_BIT,D
    0458 8608        1203         LDAA    #8               INIT THE COUNTER
    045A 9703        1204         STAA    BITCOUNT,D
                     1205
                     1206 * The first thing we have to do is look for a SYNC byte.
                     1207 * We just keep shifting bits into a byte (in A) until we recognise
                     1208 * the sync.
                     1209
    045C 4F          1210         CLRA                     SET TO NON-SYNC
    045D             1211 RS_SYNC
    045D D607        1212         LDAB    STATUS           3 GET INITIAL INPUT STATE
    045F             1213 RS_CLOCK1
    045F D107        1214         CMPB    STATUS           3 COMPARE TO CURRENT STATE
    0461 27FC        1215         BEQ     RS_CLOCK1        3 3 LOOP UNTIL WE SEE CLOCK EDGE OR MOTION CHANGE
                     1216
                     1217 * MAKE SURE WE SPEND AT LEAST 42 uSEC BEFORE WE GO BACK TO RS_SYNC AGAIN
                     1218
    0463 D80A        1219         EORB    LAST_SEEN,D      3 6 GRAB THE PREVIOUS DATA BIT
    0465 05          1220         LSLD                     3 9 STORE IT AWAY
    0466 D607        1221         LDAB    STATUS           3 12 DID WE STALL?
    0468 D70A        1222         STAB    LAST_SEEN,D      3 15
    046A D520        1223         BITB    MOTION_BIT,D     3 18
    046C 273B        1224         BEQ     RS_STALLED       3 21 IF SO, SIGNAL ERROR
    046E 01          1225         NOP                      2 23 WE CAN'T LEAVE LOOP UNTIL AT LEAST
    046F 01          1226         NOP                      2 25 42 uSEC HAVE GONE BY
    0470 01          1227         NOP                      2 27
    0471 01          1228         NOP                      2 29
    0472 01          1229         NOP                      2 31
    0473 01          1230         NOP                      2 33
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0474 01             1231         NOP                     2 35
    0475 01             1232         NOP                     2 37
    0476 21FE           1233         BRN      $              3 40
    0478 8116           1234         CMPA     #SYN           2 42 HAVE WE FOUND SYNC YET?
    047A 26E1           1235         BNE      RS_SYNC        3 45 BRANCH FOR ANOTHER BIT IF NOT
                        1236
    047C                1237 RS_READ_BIT
    047C D607           1238         LDAB     STATUS         3 GET INITIAL INPUT STATE
    047E                1239 RS_CLOCK2
    047E D107           1240         CMPB     STATUS         3 COMPARE TO CURRENT STATE
    0480 27FC           1241         BEQ      RS_CLOCK2      3 3 LOOP UNTIL WE SEE CLOCK EDGE OR MOTION CHANGE
                        1242
                        1243 * MAKE SURE WE SPEND AT LEAST 42 uSEC BEFORE WE GO BACK TO RS_READ_BIT AGAIN
                        1244
    0482 D80A           1245         EORB     LAST_SEEN,D    3 6 GET THE PREVIOUS DATA BIT
    0484 05             1246         LSLD                    3 9 STORE IT AWAY
                        1247
    0485 D607           1248         LDAB     STATUS         3 12
    0487 D70A           1249         STAB     LAST_SEEN,D    3 15 SAVE FOR NEXT BIT
    0489 D520           1250         BITB     MOTION_BIT,D   3 18 IS TAPE STILL MOVING?
    048B 271C           1251         BEQ      RS_STALLED     3 21 BRANCH IF NOT
    048D 7A0003         1252         DEC      BITCOUNT       6 27 ARE THERE ANY BITS LEFT IN THE PREV. BYTE?
    0490 2609           1253         BNE      RS_WAIT        3 30 BRANCH IF YES
    0492 A700           1254         STAA     0,X            4 34 ELSE SAVE THE PREVIOUS BYTE
    0494 8608           1255         LDAA     #8             2 36 RE-INIT BIT COUNT
    0496 9703           1256         STAA     BITCOUNT,D     3 39
    0498 08             1257         INX                     3 42 INC. DATA POINTER
    0499 20E1           1258         BRA      RS_READ_BIT    3 45 BRANCH FOR ANOTHER BIT
                        1259
    049B                1260 RS_WAIT
    049B 01             1261         NOP                     2 32
    049C 01             1262         NOP                     2 34
    049D 01             1263         NOP                     2 36
    049E 9C04           1264         CPX      STUFF_END,D    6 42 IS THE BUFFER FULL?
    04A0 26DA           1265         BNE      RS_READ_BIT    3 45 GET ANOTHER BIT IF NOT
                        1266
    04A2 BD03A1         1267         JSR      CHECK_MOTION   SEE IF THE TAPE JAMMED WHILE WE WERE BUSY
    04A5 2502           1268         BCS      RS_STALLED     BRANCH IF SO
    04A7 0C             1269         CLC                     ELSE SHOW NO JAM
    04A8 39             1270         RTS                     RETURN TO CALLER
                        1271
    04A9                1272 RS_STALLED
    04A9 BD026A         1273         JSR      STOP_FORWARD   TURN OFF THE MOTORS
    04AC DC11           1274         LDD      BLOCKS_TRACK,D FIGYUR OUT HOW FAR TO BACK UP
    04AE 2714           1275         BEQ      REALLY_LOST    WE DON'T SEEM TO KNOW HOW MANY BLOCKS/TRACK
    04B0 930D           1276         SUBD     WANTED_BLOCK,D
    04B2 BD0294         1277         JSR      FAST_REVERSE   GENTLEMEN, START YOUR ENGINES
    04B5                1278 RS_BACKING
    04B5 BD0315         1279         JSR      SKIP_BLOCK
    04B8 BD03A1         1280         JSR      CHECK_MOTION   FULLY REWOUND?
    04BB 250A           1281         BCS      RS_EXIT
    04BD 830001         1282         SUBD     #1             ONE MORE BLOCK
    04C0 26F3           1283         BNE      RS_BACKING
    04C2 2003           1284         BRA      RS_EXIT
    04C4                1285 REALLY_LOST
    04C4 BD0210         1286         JSR      REWIND
    04C7                1287 RS_EXIT
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE
  04C7 BD02A7                JSR     STOP_REVERSE    STOP THE MOTORS
  04CA 0D                    SEC                     SHOW THERE WAS A JAM
  04CB 39                    RTS

                      ******************************************************************
                      * This subroutine writes the 1K bytes of data in the buffer to a
                      * block on the tape.  Note that WRITE_BLOCK and WRITE_BYTE, as a
                      * team, agree to use B only as an image of the port.  This routine
                      * assumes the tape is in the gap between the header and the data
                      * when it is called, and it exits with the tape stopped.
                      * This routine looks at WANTED_DRIVE and goes to WRITE_BLOCK0 or WRITE_BLOCK1
                      * accordingly.

  04CC                WRITE_BLOCK

                      * FIRST WE MUST SET THE TRACK NUMBER.
  04CC D603                  LDAB    MISC            GET CURRENT STATE
  04CE C4FB                  ANDB    #0FFH-TRACK     ASSUME WE WANT TRACK ZERO
  04D0 7D000C                TST     WANTED_TRACK,D  SEE IF WE WERE RIGHT
  04D3 2702                  BEQ     TK_OK_TOO       BRANCH IF SO
  04D5 CA04                  ORAB    #TRACK          ELSE CHOOSE TRACK 1
  04D7                TK_OK_TOO
  04D7 D703                  STAB    MISC

  04D9 7D000B                TST     WANTED_DRIVE,D
  04DC 2608                  BNE     WRITE_BLOCK1    BRANCH IF USING DRIVE 1

  04DE 9602                  LDAA    MOTOR
  04E0 84BF                  ANDA    #WENABLE0       TURN ON WRITE ENABLE
  04E2 9702                  STAA    MOTOR
  04E4 2006                  BRA     WRITE_COMMON

  04E6                WRITE_BLOCK1
  04E6 9602                  LDAA    MOTOR
  04E8 847F                  ANDA    #WENABLE1       TURN ON WRITE ENABLE
  04EA 9702                  STAA    MOTOR

  04EC                WRITE_COMMON
  04EC BD0301                JSR     PAUSE1          LEAVE A LITTLE ROOM
  04EF 7F0000                CLR     ZERO_BYTE       SET UP THE PREAMBLE BYTES
  04F2 8616                  LDAA    #SYN
  04F4 9701                  STAA    SYNC_BYTE,D
  04F6 D603                  LDAB    MISC            GET THE IMAGE OF THE PORT WITH WDATA IN IT

                      * Ready to start -- write a couple of zero bytes and the sync byte.

  04F8 CE0000                LDX     #ZERO_BYTE
  04FB BD0573                JSR     WRITE_BYTE
  04FE 7F0000                CLR     ZERO_BYTE       6
  0501 01                    NOP                     2
  0502 01                    NOP                     2
  0503 BD0573                JSR     WRITE_BYTE      6
  0506 7F0000                CLR     ZERO_BYTE       6
  0509 01                    NOP                     2
  050A 01                    NOP                     2
  050B BD0573                JSR     WRITE_BYTE      6
  050E 7F0000                CLR     ZERO_BYTE       6
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

     0511 01             1345          NOP                      2
     0512 01             1346          NOP                      2
     0513 BD0573         1347          JSR      WRITE_BYTE      6
     0516 CE0001         1348          LDX      #SYNC_BYTE      3
     0519 21FE           1349          BRN      $               3
     051B 01             1350          NOP                      2
     051C 01             1351          NOP                      2
     051D BD0573         1352          JSR      WRITE_BYTE      6
     0520 CE0400         1353          LDX      #BUFFER         3
     0523 21FE           1354          BRN      $               3
     0525 01             1355          NOP                      2
     0526 01             1356          NOP                      2
                         1357
     0527                1358 WBNEXT_BYTE
     0527 BD0573         1359          JSR      WRITE_BYTE      6 WRITE A DATA BYTE
     052A 08             1360          INX                      3 INC. THE POINTER
     052B 8C0800         1361          CPX      #BUFFER_END     4 IS THAT THE END OF DATA?
     052E 26F7           1362          BNE      WBNEXT_BYTE     3 BRANCH IF NOT
                         1363
     0530 CE0000         1364          LDX      #ZERO_BYTE      3 WRITE A ZERO BYTE
     0533 8D3E           1365          BSR      WRITE_BYTE      3
     0535 7F0000         1366          CLR      ZERO_BYTE       6 AND WRITE ANOTHER
     0538 01             1367          NOP                      2
     0539 01             1368          NOP                      2
     053A 8D37           1369          BSR      WRITE_BYTE      6
     053C 8616           1370          LDAA     #SYN            2 WRITE A SYNC BYTE
     053E B70001         1371          STAA     SYNC_BYTE       3
     0541 CE0001         1372          LDX      #SYNC_BYTE      3
     0544 01             1373          NOP                      2
     0545 8D2C           1374          BSR      WRITE_BYTE      6
     0547 CE0015         1375          LDX      #CRC            3 WRITE THE CRC HIGH BYTE
     054A 01             1376          NOP                      2
     054B 01             1377          NOP                      2
     054C 21FE           1378          BRN      $               3
     054E 8D23           1379          BSR      WRITE_BYTE      6
     0550 08             1380          INX                      3 WRITE THE CRC LOW BYTE
     0551 21FE           1381          BRN      $               3
     0553 01             1382          NOP                      2
     0554 01             1383          NOP                      2
     0555 8D1C           1384          BSR      WRITE_BYTE      6
     0557 CE0000         1385          LDX      #ZERO_BYTE      3 WRITE ANOTHER ZERO AS JUNK
     055A 01             1386          NOP                      2
     055B 01             1387          NOP                      2
     055C 01             1388          NOP                      2
     055D 01             1389          NOP                      2
     055E 01             1390          NOP                      2
     055F 8D12           1391          BSR      WRITE_BYTE      3
                         1392
     0561 9602           1393          LDAA     MOTOR
     0563 8AC0           1394          ORAA     #WDISABLE       DISABLE WRITING
     0565 9702           1395          STAA     MOTOR
     0567 BD03A1         1396          JSR      CHECK_MOTION    SEE IF THE TAPE JAMMED WHILE WE WERE BUSY
     056A 2502           1397          BCS      WRSTALLED       BRANCH IF SO
                         1398 *        JSR      STOP_FORWARD    ELSE STOP THE TAPE   * FACE *
     056C 0C             1399          CLC                      SHOW THERE WAS NO JAM
     056D 39             1400          RTS
                         1401
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

     056E               1402 WBSTALLED
     056E BD026A        1403          JSR      STOP_FORWARD    TURN OFF THE MOTORS
     0571 0D            1404          SEC                      SHOW THERE WAS A JAM
     0572 39            1405          RTS
                        1406
                        1407 * This subroutine writes out the byte pointed to by X.  Note that it
                        1408 * zeroes the memory location and the clobbers the registers.  It assumes that A
                        1409 * is already set up with the port state.  We write the first clock edge as soon
                        1410 * as we can so as to maximize time available to the calling routine.
                        1411 * If the caller wants to write two adjacent bytes, it has 16 cycles
                        1412 * between calls (including the JSR or BSR).
                        1413
     0573               1414 WRITE_BYTE
     0573 C801          1415          EORB     #WTDATA         2 FLIP THE DATA BIT
     0575 D703          1416          STAB     MISC            3 WRITE IT OUT TO MAKE CLOCK EDGE
                        1417 *        ----------------         TAKE EXACTLY 31 CYCLES TO MAKE DATA EDGE
     0577 8608          1418          LDAA     #8              2 SET THE BIT COUNTER
     0579 2007          1419          BRA      WBENTER         3 ENTER THE NORMAL LOOP
                        1420
     057B               1421 WRITE_BIT
     057B C801          1422          EORB     #WTDATA         2 FLIP THE DATA BIT
     057D D703          1423          STAB     MISC            3 WRITE IT OUT TO MAKE CLOCK EDGE
                        1424 *        ----------------         TAKE EXACTLY 31 CYCLES TO MAKE DATA EDGE
     057F 01            1425          NOP                      2
     0580 21FE          1426          BRN      $               3
     0582               1427 WBENTER
     0582 01            1428          NOP                      2
     0583 01            1429          NOP                      2
     0584 01            1430          NOP                      2
     0585 01            1431          NOP                      2
     0586 01            1432          NOP                      2
     0587 01            1433          NOP                      2
     0588 6800          1434          LSL      0,X             6 ROTATE OUT THE DATA BIT
     058A 2416          1435          BCC      WBZERO          3 BRANCH IF NO DATA EDGE NEEDED
     058C C801          1436          EORB     #WTDATA         2 ELSE FLIP THE DATA BIT
     058E D703          1437          STAB     MISC            3 WRITE IT OUT TO MAKE DATA EDGE
                        1438 *        ----------------         TAKE EXACTLY 39 CYCLES TO MAKE CLOCK EDGE
     0590               1439 WBBOTH
     0590 01            1440          NOP                      2
     0591 01            1441          NOP                      2
     0592 01            1442          NOP                      2
     0593 01            1443          NOP                      2
     0594 4A            1444          DECA                     2 DEC. THE BIT COUNT
     0595 270E          1445          BEQ      WBDONE          3 EXIT IF FINISHED THIS BYTE
     0597 01            1446          NOP                      2
     0598 01            1447          NOP                      2
     0599 01            1448          NOP                      2
     059A 01            1449          NOP                      2
     059B 01            1450          NOP                      2
     059C 01            1451          NOP                      2
     059D 01            1452          NOP                      2
     059E 01            1453          NOP                      2
     059F 01            1454          NOP                      2
     05A0 20D9          1455          BRA      WRITE_BIT       3 GO WRITE OUT THE NEXT BIT
                        1456
                        1457 * This bit of code must take the same time as the bit which writes the
                        1458 * data edge for a ONE bit.
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1459
    05A2              1460 WBZERO
    05A2 01           1461         NOP                   2
    05A3 20EB         1462         BRA     WBBOTH        3 GO RE-JOIN THE MAIN CODE
                      1463
    05A5              1464 WBDONE
    05A5 39           1465         RTS                   5

Errors=    0
```

```
LINE#   SYMBOL          TYPE     REFERENCES

  261   APP_INIT          P   400
  262   ATP_APP           P   134
  599   BACKUP            P   570
  209   BD_MODE           A   235,968,991,1124,1129,1135
  216   BITCOUNT          D   1204,1252,1256
  240   BLOCKS_TRACK      D   594,955,963,999,1003,1008,1168,1274
  222   BLOCK_NUM         D   565,620,664,1165
  252   BUFFER            A   253,812,1073,1353
  253   BUFFER_END        A   817,1071,1361
  976   CALC1             P   964
  984   CALC_BAD          P   958
  960   CALC_OK           P   956
  950   CALC_PHYS         P   371
  813   CALC_S2           P   818
  810   CALC_SUM          P   429,456,1083
  482   CANT_READ         P   437
  393   CHECK_1           P   391
  399   CHECK_2           P   397
 1045   CHECK_MOTION      P   582,607,658,1079,1110,1170,1267,1280,1396
  341   CHK0              P   326,331,334,338
  350   CHK0_1            P   346
  327   CHK1_1            P   323
  364   CHK_SIG           P   349,354,357,361
 1019   CIP               P   390,396,414,444
  162   CIP0              A   343,1024
  167   CIP1              A   319,1029
 1027   CIP_1             P   1022
 1031   CIP_8             P   1026
 1035   CIP_9             P   1025,1030
  283   CLEAR_RAM         P
 1053   CM1               P   1049
 1056   CM2               P   1051
 1060   CM3               P   1052,1055
  469   CMD_COMP          P   376,384,434
  141   COMMAND_BUFFER    E   422,424,426,474,476,478,479,953,961,962,965,966
  244   CRC               D   433,460,1077,1087,1375
  245   CRC_END           D   1075
  208   CS_MODE           A   428,455,804,1082
  137   CS_WORD           E
  142   CURRENT_RAM       E   291,333,356,423,425,427,447,475,477
  192   C_COMMAND         A   375
  189   C_READ            A   378
  193   C_RESET           A   369
  191   C_REWIND          A
  190   C_WRITE           A   380
  140   DATA_BUFFER       E
  144   DDR1              A   269
  145   DDR2              A   272
  148   DDR4              A   275
  207   DISAB_0           A
  358   DR0_OK            P   344
  335   DR1_OK            P   320
  220   DRIVE_NUM         D   533,545
 1201   DR_OK             P   1199
  493   ERR_1             P   490
  488   ERR_COMMON        P   471,481,484
  495   ERR_END           P   492
```

```
LINE#   SYMBOL          TYPE     REFERENCES

  413  EXEC_R            P   379
  387  EXEC_RESET        P   370
  443  EXEC_W            P   381
  709  FASTF             P   706
  707  FASTF1            P   704
  763  FASTR             P   760
  761  FASTR1            P   758
  701  FAST_FORWARD      P   578
  755  FAST_REVERSE      P   596,603,1277
  531  FIND_AGAIN        P   527
  522  FIND_BLOCK        P   419,462
  525  FIND_BLOK         P   622,633
  548  FIND_HEAD         P   588,613
  241  FIND_TRIES        D   524,526,587,612,621,632
  576  FORWARD           P   571,573
  635  FOUND_IT          P   629
  175  FWDFAST0          A   705
  176  FWDFAST1          A   708
  580  FWDLOOP           P   585
  173  FWDSLOW0          A   175,680
  174  FWDSLOW1          A   176,683
  179  FWDSTOP0          A   727
  180  FWDSTOP1          A   731
  592  FWD_STALL         P   551,583,626
  682  GOF1              P   679
  684  GOF2              P   681
  552  GOT_HEAD          P   550
  675  GO_FORWARD        P   1108
  568  GO_LOOK           P   566
  246  HEAD_BUFFER       D   553,628,1114,1127,1136,1142,1145,1147,1148,1149,1150,1151,1152,1153,1154,1155,1163,1167
  247  HEAD_END          D   1112
  200  HEAD_ID           A   1128
  201  HEAD_ID2          A   1137
  501  INIT_TIMER        P   296
  624  JUST_AHEAD        P   567,574,630
  225  LAST_SEEN         D   1219,1222,1245,1249
  139  LENGTH_OF_IO_ST   E
  374  MAIN_1            P   372
  301  MAIN_LOOP         P   366,512
  996  MANGLE_NUM        P   969
 1010  MANGL_END         P   998
 1006  MANGL_HI          P   1004
  147  MISC              A   318,1028,1188,1194,1304,1310,1330,1416,1423,1437
  160  MOTION0           A   345,728,782,1050,1197
  161  MOTION1           A   322,732,786,1054,1200
  248  MOTION_BIT        D   1202,1223,1250
  146  MOTOR             A   267,655,685,689,710,736,744,764,790,798,1315,1317,1321,1323,1393,1395
  317  MOTORS_OKAY       P   304,307,313
  188  M_DATA            E   377
  187  M_SIG             E   188,365,496
  136  NIM_BLOCK         E   187
  558  NOT_IT            P   555
  485  NO_BLOCK          P   373,420,463
  472  NO_CASSETTE       P   415,445
  185  OCF               A   305,887,906,940
  152  OCR               A   311,506,886,905,939
  153  P3CSR             A
```

```
LINE#   SYMBOL          TYPE     REFERENCES

  865  PAUSE             P   656,661,662
  899  PAUSE1            P   1326
  878  PAUSE100          P   690,691,711,765
  907  PAUSE1WAIT        P   909
  880  PAUSE50           P   879
  888  PAUSE50WAIT       P   890
  868  PSE1              P   870
  243  QUIET_TIME        A   507
  158  RAMCR             A
 1094  RB_ERROR          P   1080
 1096  RB_ERROR2         P   1088
  156  RDATA             A   498
  163  RDDATA0           A
  164  RDDATA1           A
 1070  READ_BLOCK        P   421
 1109  READ_H2           P   1138,1146,1157
 1107  READ_HEADER       P   549,625,957
 1185  READ_STUFF        P   1074,1078,1115
  242  READ_TRIES        D   417,435
 1285  REALLY_LOST       P   1275
  285  REPEAT            P   289
  418  RETRY             P   436
  177  REVFAST0          A   650,759
  178  REVFAST1          A   653,762
  605  REVLOOP           P   597,610
  181  REVSTOP0          A   781
  182  REVSTOP1          A   785
  617  REV_STALL         P   608
  654  REW               P   651
  652  REW1              P   649
  657  REW2              P   659
  645  REWIND            P   392,398,1286
 1175  RH_STALLED        P   1111,1116,1171
  154  RMCR              A   278
 1278  RS_BACKING        P   1283
 1213  RS_CLOCK1         P   1215
 1239  RS_CLOCK2         P   1241
 1287  RS_EXIT           P   1281,1284
 1237  RS_READ_BIT       P   1258,1265
 1272  RS_STALLED        P   1224,1251,1268
 1211  RS_SYNC           P   1235
 1260  RS_WAIT           P   1253
  563  SAME_TRACK        P   537
  155  SCSR              A   281,497,500
  543  SET_VARS          P   534
  733  SF                P   729
  730  SF1               P   726
  742  SF_OK             P   735,739
  233  SHUT_DOWN         D   303,312,508
  935  SKIP              P   927
  941  SKIPWAIT          P   943
  921  SKIP_BLOCK        P   581,606,1279
  926  SKIP_LOOP         P   929
  787  SR                P   783
  784  SR1               P   780
  796  SR_OK             P   789,793
  250  STACK             D   264
```

```
LINE#   SYMBOL         TYPE     REFERENCES

  249  STACK_SPACE       D
  149  STATUS            A  321,342,734,738,788,792,1023,1047,1212,1214,1221,1238,1240,1248
  737  STOPFWAIT         P  741
  183  STOPPED           A  266,743,797
  791  STOPRWAIT         P  795
  720  STOP_FORWARD      P  316,559,586,593,631,959,985,1095,1176,1273,1403
  774  STOP_REVERSE      P  611,618,660,1288
  202  STOP_TIMEOUT      A  724,778
  217  STUFF_END         D  1072,1076,1113,1264
  199  SYN               A  1234,1328,1370
  214  SYNC_BYTE         D  1329,1348,1371,1372
  195  S_BADBLK          A  483
  196  S_NOBLOCK         A  486
  198  S_NODRIVE         A  324,347
  197  S_NOTAPE          A  328,337,351,360,480
  194  S_OK              A  339,362,470
  138  TAPE_STATUS0      E  348,352,359,363,491
  138  TAPE_STATUS1      E  325,329,336,340,494
  237  TAPE_TYPE         D  997,1125,1139
  150  TCSR              A  306,308,503,883,889,902,908,936,942
  157  TDATA             A
  215  TEMP              D
  151  TIMER             A  309,504,884,903,937
 1193  TK_OK             P  1191
 1309  TK_OK_TOO         P  1307
  166  TRACK             A  1189,1192,1305,1308
  221  TRACK_NUM         D  536,547
 1141  VALID_HEAD        P  1130
  230  WANTED_BLOCK      D  554,564,595,627,967,977,1001,1002,1007,1009,1276
  228  WANTED_DRIVE      D  330,353,389,395,489,532,544,648,678,703,725,757,779,954,1021,1048,1198,1312
  229  WANTED_TRACK      D  535,546,971,979,1190,1306
 1439  WBBOTH            P  1462
 1464  WBDONE            P  1445
 1427  WBENTER           P  1419
 1358  WBNEXT_BYTE       P  1362
 1402  WBSTALLED         P  1397
 1460  WBZERO            P  1435
  172  WDISABLE          A  1394
  171  WENABLE0          A  1316
  170  WENABLE1          A  1322
 1421  WRITE_BIT         P  1455
 1301  WRITE_BLOCK       P  464
 1320  WRITE_BLOCK1      P  1313
 1414  WRITE_BYTE        P  1335,1339,1343,1347,1352,1359,1365,1369,1374,1379,1384,1391
 1325  WRITE_COMMON      P  1318
  168  WTDATA            A  1415,1422,1436
  213  ZERO_BYTE         D  1327,1334,1336,1340,1344,1364,1366,1385
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                           1 ^6801^
                           3  NAME ^Rev 04 - MJM^
                           4
                           5 De_SR_PU MACRO                ;Header Rev. 4
                           6                .GOTO Ede_SR_PU
                           7
                           8  Project:      Tau, 83-101
                           9
                          10  ******************************
                          11  *                            *
                          12  *    SR_HIMEM           MJM  *
                          13  *                            *
                          14  ******************************
                          15
                          16        Rev History
                          17        Rev.  Date          Name      Change
                          18         6    23JUL1600     HME       Changed software I/O intrpt to
                          19                                      show MTP_ACM_SEQ and ATP_APP
                          20         5    23JUL1401p    MJM       This copy is taken from the KB_68
                          21                                      directory ORANGE system to be used
                          22                                      in the tape mac software package
                          23
                          24         4    20jul955a     RPD       created SR_HIMEM2, removed added SCI vector
                          25         3    18jul1000a    RPD       added SCI interrupt vector
                          26         2    7jul1130a     RPD       replaced unused vectors with RET_VECTOR
                          27         1    16jun940a     JIM       Corrected errors.
                          28         0    15jun320p     JIM       Entered data.
                          29
                          30 Function:         Define the interupt vectors that are in the high memory
                          31                   of the 6801 located at FFF0H. Also defined is the RET_VECTOR
                          32                   interrupt service routine.
                          33
                          34 Ede_SR_PU  MEND
                          35
                          36 ;Subroutines called (referenced, but not executed)
                          37 ;
                          38                  EXT      TAPE_MAC
                          39                  EXT      ATP_APP
                          40
                          41 ;
                          42 ; dummy interrupt service routine
                          43 ;
   0000 3B                44 RET_VECTOR:      RTI                         ;unused vector interrupt service routine
                          45
   0001 0000              46                  FDB      TAPE_MAC           ;Serial i/o interupt vector
   0003 0000              47                  FDB      RET_VECTOR         ;Timer overflow interupt vector
   0005 0000              48                  FDB      RET_VECTOR         ;Output compare interupt vector, i. e. timer interupt
   0007 0000              49                  FDB      RET_VECTOR         ;Input capture interupt vector
   0009 0000              50                  FDB      RET_VECTOR         ;IRQ1 - maskable interupt vector
   000B 0000              51                  FDB      RET_VECTOR         ;Software interupt vector
   000D 0000              52                  FDB      RET_VECTOR         ;Non-maskable interupt vector
   000F 0000              53                  FDB      ATP_APP            ;Reset interupt vector

Errors=    0
```

```
LINE#   SYMBOL         TYPE     REFERENCES

   39  ATP_APP           E  53
   44  RET_VECTOR        P  47,48,49,50,51,52
   38  TAPE_MAC          E  46
```

```
HP 64000 Linker                                   Mon,  7 Nov 1983, 10:53          Page   1

FILE/PROG NAME                PROGRAM   DATA      COMMON    ABSOLUTE        DATE                     TIME   COMMENTS
-------------------------------------------------------------------------------------------------------------------------
TAPE_MAC:pADAMT               F800      0080                                Mon,  7 Nov 1983, 10:28  Rev 01 - HME
D_MTP:pADAMT                            0097      0400                      Mon,  7 Nov 1983, 10:32  Rev 00 - DLS
MTP_TR_RE:pADAMT              F9BC                                          Mon,  7 Nov 1983, 10:34  Rev 04 - RPD
MTP_TR_TR:pADAMT              F9D8                                          Mon,  7 Nov 1983, 10:35  Rev 03 - RPD
MTP_TR_TC:pADAMT              F9F0                                          Mon,  7 Nov 1983, 10:37  Rev 01 - RPD
MTP_NIM_W:pADAMT              FA1B                                          Mon,  7 Nov 1983, 10:38  Rev 02 - DLS
TAPE_APP:pADAMT               FA2A      009E                                Mon,  7 Nov 1983, 10:41  Rev 15
next address                  FFD0      00DD      0800

SR_HIMEM:pADAMT               FFEF                                          Mon,  7 Nov 1983, 10:40  Rev 04 - MJM
next address                  0000

XFER address= 0000      Defined by  DEFAULT
absolute & link_com file name=TPA:pADAMT
Total# of bytes loaded= 0C3E
```

```
SYMBOL           R VALUE    DEF BY                        REFERENCES
----------------------------------------------------------------------------------------------------------------------------
ATP_APP          P FA2A     TAPE_APP:pADAMT               SR_HIMEM:pADAMT
A_DATA           D 009C     D_MTP:pADAMT
A_SIG            D 009C     D_MTP:pADAMT
BREAK_ORFE       P F9C5     MTP_TR_RE:pADAMT
CLEAN_UART_HW    P FA0E     MTP_TR_TC:pADAMT              MTP_TR_TR:pADAMT
CNFG_WORD        D 009C     D_MTP:pADAMT
COMMAND_BUFFER   D 0080     TAPE_MAC:pADAMT               TAPE_APP:pADAMT
COUNT            D 0099     D_MTP:pADAMT
CS_WORD          D 0098     D_MTP:pADAMT                  TAPE_APP:pADAMT
CURRENT_RAM      D 0085     TAPE_MAC:pADAMT               TAPE_APP:pADAMT
CURRENT_STATE    D 0097     D_MTP:pADAMT                  MTP_TR_TC:pADAMT  MTP_TR_RE:pADAMT  TAPE_MAC:pADAMT
D1_MODE_WORD     D 0098     D_MTP:pADAMT
DATA_BUFFER      C 0400     D_MTP:pADAMT                  TAPE_APP:pADAMT  TAPE_MAC:pADAMT
D_MTP            D 0097     D_MTP:pADAMT
IO_STATUS_BLOCK  D 0094     TAPE_MAC:pADAMT
LENGTH_OF_IO_ST  A 0001     TAPE_MAC:pADAMT               TAPE_APP:pADAMT
MTP_NIM_WRITE    P FA1B     MTP_NIM_W:pADAMT              TAPE_MAC:pADAMT
MTP_TR_REC       P F9BC     MTP_TR_RE:pADAMT              TAPE_MAC:pADAMT
MTP_TR_TCU       P F9F0     MTP_TR_TC:pADAMT              TAPE_MAC:pADAMT
MTP_TR_TRANS     P F9D8     MTP_TR_TR:pADAMT              TAPE_MAC:pADAMT
M_DATA           D 009D     D_MTP:pADAMT                  MTP_NIM_W:pADAMT
M_SIG            D 009C     D_MTP:pADAMT                  MTP_NIM_W:pADAMT  TAPE_MAC:pADAMT
NIM_BLOCK        D 009C     D_MTP:pADAMT                  TAPE_APP:pADAMT
NODE_ADDRESS     A 0008     D_MTP:pADAMT
TAPE_MAC         P F800     TAPE_MAC:pADAMT               SR_HIMEM:pADAMT
TAPE_STATUS0     D 0095     TAPE_MAC:pADAMT               TAPE_APP:pADAMT
TAPE_STATUS1     D 0096     TAPE_MAC:pADAMT               TAPE_APP:pADAMT
```

```
 emulate
external
no
no
yes
0 thru 0FFFFH user ram
end
no
no

reset
wait 1

modify io_port 78H to 0
wait  1
modify io_port 78H to 1
load N_EOS_05:N_EOS
display memory _HARD_INIT mnemonic
load 8NEW   :TOS_MM


display memory 0
load TAPE

; Coldstart load
run from 0

; Overlay 1
run until address 1004H data 81H status memory_write
wait measurement_complete
break
modify memory 1004H to 0
display registers
run

; Overlay 2
run until address 1004H data 81H status memory_write
wait measurement_complete
break
modify memory 1004H to 0
run

; Overlay 3
run until address 1004H data 81H status memory_write
wait measurement_complete
break
modify memory 1004H to 0
run

; Overlay 4
run until address 1004H data 81H status memory_write
wait measurement_complete
break
modify memory 1004H to 0
run

; Overlay 5
run until address 1004H data 81H status memory_write
wait measurement_complete
break
modify memory 1004H to 0
run
```

```
; Overlay 6
run until address 1004H data 81H status memory_write
wait measurement_complete
break
modify memory 1004H to 0
run

; Overlay 7
run until address 1004H data 81H status memory_write
wait measurement_complete
break
modify memory 1004H to 0
run

; Overlay 8
run until address 1004H data 81H status memory_write
wait measurement_complete
break
modify memory 1004H to 0
run

; Overlay 9
run until address 1004H data 81H status memory_write
wait measurement_complete
break
;load OVL_9
modify memory 1004H to 0
run


run until address 1004H data 81H status memory_write
wait measurement_complete
end

X:A132DT
```