```
LOCATION OBJECT CODE LINE     SOURCE LINE

                        ^6801^
                         NAME ^Rev 16 GRW^

                        De_DAISY4 MACRO                          ;Header Rev. 4
                                       .GOTO Ede_DAISY4

                         Project:     NET, 83-101

                         [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
                         [][]                                                      [][]
                         [][]    D A I S Y 4                       ? ? ?           [][]
                         [][]                                                      [][]
                         [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]

                               Rev History
                               Rev.  Date          Name      Change
                               16    83/nn/nn      GRW       ...
                               15    83/nn/nn      ???       ...
                               14    83/nn/nn      ???       ...
                               13    83/nn/nn      ???       ...
                               12    83/nn/nn      ???       ...
                               11    83/nn/nn      ???       ...
                               10    83/nn/nn      ???       ...
                                9    83/nn/nn      ???       ...
                                8    83/nn/nn      ???       ...
                                7    83/nn/nn      ???       ...
                                6    83/nn/nn      ???       ...
                                5    83/nn/nn      ???       ...
                                4    83/nn/nn      ???       ...
                                3    xxjulxxxx     ???       ...
                                2    xxjulxxxx     ???       ...
                                1    xxjulxxxx     ???       converted pseudo code to 6801 code
                                0    xxjulxxxx     ???       Initial Pseudo Code

                        Ede_DAISY4     MEND





                        * Missing page 1 reconstructed by Richard F. Drushel 9603.03
                        * using link map and xref listing, as well as comparison
                        * with other 6801 app title pages.

                        * Actual comments were probably longer, but like the dates and
                        * programmer initials, cannot be recovered.







                        * 6801 EQUATES.

<0000>                  DDR1     EQU    000H             DATA DIRECTION REGISTER FOR PORT 1
<0001>                  DDR2     EQU    001H              "       "        "       "   "   2
<0002>                  PORT1    EQU    002H
```

```
LOCATION OBJECT CODE  LINE     SOURCE LINE

            (0003)     59 PORT2   EQU     003H
            (0004)     60 DDR3    EQU     004H                "       "       "       "   "   3
            (0005)     61 DDR4    EQU     005H                "       "       "       "   "   4
            (0006)     62 PORT3   EQU     006H
            (0007)     63 PORT4   EQU     007H
            (0008)     64 TCSR    EQU     008H            TIMER CONTROL AND STATUS REGISTER
            (0080)     65 TCSR_ICF        EQU     10000000B
            (0040)     66 TCSR_OCF        EQU     01000000B
            (0020)     67 TCSR_TOF        EQU     00100000B
            (0010)     68 TCSR_EICI       EQU     00010000B
            (0008)     69 TCSR_EOCI       EQU     00001000B
            (0004)     70 TCSR_ETOI       EQU     00000100B
            (0002)     71 TCSR_IEDG       EQU     00000010B
            (0001)     72 TCSR_OLVL       EQU     00000001B
            (0009)     73 TIMER   EQU     009H            TIMER REGISTER (16 BITS)
            (000B)     74 OCR     EQU     00BH            OUTPUT COMPARE REGISTER (16 BITS)
            (000D)     75 ICR     EQU     00DH            INPUT CAPTURE REGISTER (16 BITS)
            (000F)     76 P3CSR   EQU     00FH            PORT 3 CONTROL AND STATUS REGISTER
            (0010)     77 RMCR    EQU     010H            RATE AND MODE CONTROL REGISTER FOR SCI
            (0008)     78 RMCR_CC1        EQU     00001000B
            (0004)     79 RMCR_CC0        EQU     00000100B
            (0002)     80 RMCR_SS1        EQU     00000010B
            (0001)     81 RMCR_SS0        EQU     00000001B
            (0011)     82 TRCSR   EQU     011H            TRANSMIT/RECEIVE CONTROL AND STATUS REGISTER
            (0080)     83 TRCSR_RDRF      EQU     10000000B
            (0040)     84 TRCSR_ORFE      EQU     01000000B
            (0020)     85 TRCSR_TDRE      EQU     00100000B
            (0010)     86 TRCSR_RIE       EQU     00010000B
            (0008)     87 TRCSR_RE        EQU     00001000B
            (0004)     88 TRCSR_TIE       EQU     00000100B
            (0002)     89 TRCSR_TE        EQU     00000010B
            (0001)     90 TRCSR_WU        EQU     00000001B
            (0012)     91 RDATA   EQU     012H            RECEIVE DATA REGISTER
            (0013)     92 TDATA   EQU     013H            TRANSMIT DATA REGISTER
            (0014)     93 RAMCR   EQU     014H            RAM CONTROL REGISTER
                       94
            (0002)     95 MOTORS  EQU     PORT1           PORT 1 IS USED FOR STEPPING MOTORS
                       96
            (0006)     97 FUDGE   EQU     PORT3           PORT USED TO FUDGE DESIGN SPEEDS
                       98
            (0007)     99 BITS    EQU     PORT4           PORT 4 IS USED FOR VARIOUS THINGS
            (0080)    100 BITS_TEST       EQU     10000000B       SELF-TEST INPUT BIT (0 = SELF-TEST)
            (0040)    101 BITS_SPOKES     EQU     01000000B       WHEEL HOME INPUT BIT (1 = HOME)
            (0020)    102 BITS_DETENT     EQU     00100000B       PLATEN DETENT INPUT BIT
            (0010)    103 BITS_LEFT       EQU     00010000B       LEFT MARGIN INPUT BIT (1 = LEFT MARGIN)
            (00F7)    104 BITS_BRAKE      EQU     11110111B       PLATEN MOTOR BRAKE OUTPUT (0 = BRAKE ON)
            (00FB)    105 BITS_LF         EQU     11111011B       PLATEN MOTOR ADVANCE OUTPUT (0 = MOTOR ON)
            (00FD)    106 BITS_RIBBON     EQU     11111101B       RIBBON ADVANCE SOLENOID (0 = SOLENOID ON)
            (00FE)    107 BITS_HAMMER     EQU     11111110B       PRINT HAMMER (0 = SOLENOID ON)
                      108
                      109          DATA
                      110
0000                  111 STACKSPACE      RMB     59              ROOM FOR STACK
            (003A)    112 STACK           EQU     $-1             INITIAL VALUE FOR STACK POINTER
003B                  113 BUFFER_POINTER  RMB     2               POINTER TO NEXT CHAR OUT OF BUFFER
                      114                 IF      SHANNON
                      115 BUFFER_COUNT    RMB     2               NUMBER OF CHARS IN BUFFER
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     116                    ELSE
    003D             117 BUFFER_COUNT       RMB     1               NUMBER OF CHARS IN BUFFER
                     118                    ENDIF
    003E             119 BUFFER             RMB     16              THE CHARACTER BUFFER
    004E             120 PLATEN             RMB     1               COMMAND/RESULT FOR PLATEN ROUTINE
    004F             121 PLATSTAT           RMB     1               STATUS OF PLATEN ROUTINE
          (0080)     122 PLATSTAT_RUN1      EQU     10000000B
          (0040)     123 PLATSTAT_RUN2      EQU     01000000B
          (0020)     124 PLATSTAT_STOP      EQU     00100000B
          (0010)     125 PLATSTAT_SETTLE    EQU     00010000B
    0050             126 PLATTICKS          RMB     1               TICK COUNTER FOR PLATEN ROUTINE
    0051             127 FIRE_PROOF         RMB     2               TIMEOUT COUNTER TO SHUT DOWN LF MOTOR
    0053             128 WHEEL              RMB     1               COMMAND/RESULT FOR WHEEL ROUTINE
    0054             129 WHSTAT             RMB     1               STATUS OF WHEEL ROUTINE
          (0080)     130 WHSTAT_BUSY        EQU     10000000B
          (0040)     131 WHSTAT_CW          EQU     01000000B
          (0020)     132 WHSTAT_SETTLE      EQU     00100000B
    0055             133 WHEELTICKS         RMB     1               TICK COUNTER FOR WHEEL ROUTINE
    0056             134 WHEELGONE          RMB     1               DISTANCE WHEEL HAS TRAVELLED
    0057             135 WHEELTOGO          RMB     1               DISTANCE LEFT TO TRAVEL
          (0057)     136 WHEELPOS           EQU     WHEELTOGO       WHEEL POSITION WHEN NOT MOVING
    0058             137 HAMMER             RMB     1               COMMAND/RESULT FOR HAMMER ROUTINE
    0059             138 HAMSTAT            RMB     1               STATUS OF HAMMER ROUTINE
          (0080)     139 HAMSTAT_HFIRE      EQU     10000000B
          (0040)     140 HAMSTAT_HREL       EQU     01000000B
          (0020)     141 HAMSTAT_RFIRE      EQU     00100000B
          (0010)     142 HAMSTAT_RREL       EQU     00010000B
          (0001)     143 HAMSTAT_ODD        EQU     00000001B
    005A             144 HAMTICKS           RMB     1               TICK COUNTER FOR HAMMER ROUTINE
    005B             145 CARRIAGE           RMB     1               COMMAND/RESULT FOR CARRIAGE ROUTINE
    005C             146 CARSTAT            RMB     1               STATUS OF CARRIAGE ROUTINE
          (0080)     147 CARSTAT_BUSY       EQU     10000000B
          (0040)     148 CARSTAT_LEFT       EQU     01000000B
          (0020)     149 CARSTAT_SETTLE     EQU     00100000B
    005D             150 CARTICKS           RMB     1               TICK COUNTER FOR CARRIAGE ROUTINE
    005E             151 CARGONE            RMB     1               DISTANCE CARRIAGE HAS TRAVELLED
    005F             152 CARTOGO            RMB     1               DISTANCE LEFT TO TRAVEL
          (005F)     153 CARPOS             EQU     CARTOGO         CARRIAGE POSITION WHEN NOT MOVING
    0060             154 MODE               RMB     1               CURRENT PRINTER MODE
          (0080)     155 MODE_TEST          EQU     10000000B       SELF-TEST MODE
          (0040)     156 MODE_REVERSE       EQU     01000000B       PRINT REVERSE DIRECTION MODE
    0061             157 IMAGE              RMB     1               IMAGE OF MOTOR OUTPUT PORT
    0062             158 HOME_IMAGE         RMB     1               IMAGE OF MOTOR PORT AT HOME
    0063             159 PHANTOM            RMB     1               PHANTOM POSITION OF CARRIAGE
    0064             160 CLOCK              RMB     2               TIMER TO TURN OFF MOTORS
          (0064)     161 WANT_TIMER         EQU     CLOCK           USED BY CARDELAY & WHEELDELAY
                     162                    IF      SHANNON
                     163 M_SIG              RMB     1
                     164 M_LEN              RMB     1
                     165 NET_IN_BUFF        RMB     16
                     166                    ENDIF
                     167
                     168 * MISC. CONSTANTS.
                     169
          (0008)     170 BS       EQU       008H            ASCII BACKSPACE
          (000A)     171 LF       EQU       00AH            ASCII LINE FEED
          (000B)     172 VT       EQU       00BH            ASCII VERTICAL TAB -- CHAR TO DO 1/2 LINEFEED
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

            (000D)   173 CR       EQU     00DH            ASCII CARRIAGE RETURN
            (000E)   174 SO       EQU     00EH            ASCII SHIFT OUT -- CHAR TO START REVERSE PRINT
            (000F)   175 SI       EQU     00FH            ASCII SHIFT IN -- CHAR TO CANCEL REVERSE PRINT
            (001B)   176 ESC      EQU     01BH            ASCII ESCAPE -- CHAR TO PRINT "SPACE" PETAL
            (0020)   177 SP       EQU     020H            ASCII SPACE
            (007F)   178 DEL      EQU     07FH            ASCII DELETE
                     179
            (0050)   180 PLWIDTH EQU      80              NUMBER OF PRINT COLUMNS ON PLATEN
            (0005)   181 HOME_SPOKE EQU  5                SPOKE NUMBER WHEN IN HOME POSITION
                     182
                     183 * SOME CONSTANTS BASED ON THE FREQUENCY OF THE CRYSTAL.  BECAUSE THE
                     184 * ASSEMBLER IS LIMITED TO 16-BIT ARTHMETIC, THESE CALCULATIONS
                     185 * HAVE TO BE DONE BY HAND (UGH!).
                     186
                     187 *XTAL    EQU     4000000         CRYSTAL FREQUENCY IN HZ
                     188 *E       EQU     XTAL/4          FREQUENCY OF E CLOCK IN HZ
                     189 *TICK    EQU     E/2400          NUMBER OF E CYCLES IN 1/2400 SEC (1 TICK)
            (01A1)   190 TICK     EQU     417
            (001B)   191 MSEC11   EQU     27              NUMBER OF TICKS IN 11 MSEC
            (0024)   192 MSEC15   EQU     36              NUMBER OF TICKS IN 15 MSEC
                     193
                     194 * SPEED AND ENERGY CONSTANTS.
                     195
            (0064)   196 CARSETL EQU    100    ;125       TICK COUNT TO SETTLE CARRIAGE MOTOR
            (0047)   197 CARSPD1 EQU     71               CARRIAGE SPEEDS IN TICKS PER STEP
            (0032)   198 CARSPD2 EQU     50
            (0029)   199 CARSPD3 EQU     41
            (0023)   200 CARSPD4 EQU     35
            (0020)   201 CARSPD5 EQU     32
            (001D)   202 CARSPD6 EQU     29
            (001B)   203 CARSPD7 EQU     27
            (0019)   204 CARSPD8 EQU     25
                     205
            (00A5)   206 WHSETL   EQU     165          TICK COUNT TO SETTLE WHEEL MOTOR
            (0015)   207 WHSPD1   EQU     21               WHEEL SPEEDS IN TICKS PER STEP
            (0012)   208 WHSPD2   EQU     18
            (000F)   209 WHSPD3   EQU     15
            (000D)   210 WHSPD4   EQU     13
            (000B)   211 WHSPD5   EQU     11
            (0009)   212 WHSPD6   EQU     9
            (0008)   213 WHSPD7   EQU     8
            (0007)   214 WHSPD8   EQU     7
                     215
            (0008)   216 E4       EQU     8               TICK COUNTS FOR HAMMER ENERGIES
            (0007)   217 E3       EQU     7
            (0006)   218 E2       EQU     6
            (0005)   219 E1       EQU     5
            (0005)   220 HAMREL   EQU     5               TICK COUNT TO DE-ENERGISE HAMMER SOLENOID
                     221
            (001B)   222 RIBFIRE EQU      MSEC11          TICK COUNT TO ENERGISE RIBBON SOLENOID
            (000A)   223 RIBREL   EQU     10              TICK COUNT TO DE-ENERGISE RIBBON SOLENOID
                     224
            (0006)   225 PL_SPEED EQU     6               PLATEN MOTOR SPEED 0..8
            (007D)   226 PL_BRAKE_COUNT EQU 125; 255      TICK COUNT TO BRAKE PLATEN MOTOR
            (0001)   227 PL_SETTLE_COUNT EQU 1 ;75       TICK COUNT TO LET MOTOR COAST TO A STOP
            (12C0)   228 PL_TIMEOUT    EQU  4800        # OF TICKS IN 2 SECOND TIMEOUT
                     229
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     230          IF       1-SHANNON
                     231          EXT      M_SIG,M_LEN,NET_IN_BUFF,PR_MAC
                     232          ENDIF
                     233
                     234          PROG
                     235
                     236 ********************************************************************************
                     237 * THIS IS THE INITIALISATION CODE.  WE INIT THE MEMORY VALUES AND THE
                     238 * HARDWARE, TAKING INTO ACCOUNT WHETHER WE WILL BE IN SELF-TEST MODE
                     239 * OR NOT.
                     240
0000                 241 RESET
0000 CE0080          242          LDX      #00080H          POINT TO START OF RAM
0003                 243 CLEAR_MEM
0003 6F00            244          CLR      0,X              ZERO IT
0005 08              245          INX                       INC. THE POINTER
0006 8C0100          246          CPX      #00100H          HAVE WE CLEARED ALL RAM YET?
0009 26F8            247          BNE      CLEAR_MEM        LOOP IF NOT
                     248
000B 8E003A          249          LDS      #STACK           INIT THE STACK
                     250
000E 86CC            251          LDAA     #11001100B       INIT THE STEPPING MOTORS AND THEIR IMAGE
0010 9761            252          STAA     IMAGE,D
0012 9702            253          STAA     MOTORS
0014 86FF            254          LDAA     #11111111B       MAKE THE PORT AN OUTPUT
0016 9700            255          STAA     DDR1
                     256
0018 860F            257          LDAA     #00001111B       INIT THE BITS ON PORT 4
001A 9707            258          STAA     BITS
001C 860F            259          LDAA     #00001111B       MAKE SOME BITS OUTPUTS
001E 9705            260          STAA     DDR4
                     261
                     262 ****************** TO SATISFY HP 64000 EMULATOR ****************
0020 8610            263          LDAA     #010H
0022 9701            264          STAA     DDR2             MAKE THE TX DATA BIT AN OUTPUT
                     265
                     266 * SEE IF WE'RE IN SELF-TEST MODE, THEN INIT THE SERIAL PORT ACCORDINGLY.
                     267
                     268          IF       1-SHANNON
0024 9607            269          LDAA     BITS             READ THE REGISTER
0026 8580            270          BITA     #BITS_TEST       LOOK AT THE DESIGNATED BIT
0028 2612            271          BNE      NOT_SELF_TEST    BRANCH IF NOT HELD LOW
                     272          ENDIF
                     273
002A 8680            274          LDAA     #MODE_TEST       RESET THE MODE BYTE
002C 9760            275          STAA     MODE,D
                     276
002E 8604            277          LDAA     #RMCR_CC0        CONFIGURE SCI -- NO INTERRUPTS
0030 9710            278          STAA     RMCR             SET BAUD RATE AND MODE
0032 860A            279          LDAA     #TRCSR_TE.OR.TRCSR_RE
0034 9711            280          STAA     TRCSR
0036 8600            281          LDAA     #0
0038 9708            282          STAA     TCSR             DISABLE TIMER INTERRUPTS
                     283
003A 200C            284          BRA      HOME_THINGS
                     285
003C                 286 NOT_SELF_TEST
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    003C 8604          287          LDAA    #RMCR_CC0       CONFIGURE SCI -- INTERRUPTS ON
    003E 9710          288          STAA    RMCR            SET BAUD RATE AND MODE
    0040 861A          289          LDAA    #TRCSR_TE.OR.TRCSR_RE.OR.TRCSR_RIE
    0042 9711          290          STAA    TRCSR
    0044 8600          291          LDAA    #0
    0046 9708          292          STAA    TCSR
                       293
    0048               294 HOME_THINGS
    0048 0E            295          CLI                     ENABLE INTERRUPTS
    0049 BD0190        296          JSR     CARHOME         HOME THE CARRIAGE
    004C 8601          297          LDAA    #1              INIT THE PHANTOM COLUMN TO MATCH
    004E 9763          298          STAA    PHANTOM,D
    0050 BD01E3        299          JSR     WHEELHOME       HOME THE WHEEL
    0053 9661          300          LDAA    IMAGE,D         SAVE THE HOME IMAGE
    0055 9762          301          STAA    HOME_IMAGE,D
    0057 7C004E        302          INC     PLATEN          SET THIS TO DO 1/2 LINEFEED WHEN INTERRUPTS TURN ON
                       303
    005A DC09          304          LDD     TIMER           GET THE CURRENT TIMER VALUE
    005C C301A1        305          ADDD    #TICK           ADD AN OFFSET TO IT
    005F DD0B          306          STD     OCR             STICK RESULT IN THE COMPARE REGISTER
    0061 8608          307          LDAA    #TCSR_EOCI      ENABLE TIMER INTERRUPT
    0063 9708          308          STAA    TCSR
                       309
                       310 ********************************************************************************
                       311 * THIS IS THE START OF THE MAIN LOOP.
                       312
                       313 * THE FIRST THING WE CAN DO IS SEE IF THERE ARE ANY CHARACTERS FROM
                       314 * THE SERIAL LINK READY TO BE MOVED INTO THE RING BUFFER.
                       315
    0065               316 MAIN
                       317          IF      SHANNON
                       318          LDD     BUFFER_COUNT,D    ARE THERE CHARS TO BE PRINTED?
                       319          ELSE
    0065 7D003D        320          TST     BUFFER_COUNT,D    ARE THERE CHARS TO BE PRINTED?
                       321          ENDIF
    0068 263B          322          BNE     PRINT_ONE       BRANCH IF SO
                       323          IF      1-SHANNON
    006A 7D0000        324          TST     M_SIG           HAS THE MAC SIGNALLED US?
    006D 2615          325          BNE     COPY_BUFFER     BRANCH IF SO
                       326
                       327 * This routine is used to point the buffer to the string if the
                       328 * unit is in self-test mode.  If not, it does nothing.
                       329
    006F               330 MAKE_CHARS
    006F 9660          331          LDAA    MODE,D          GET THE MODE BYTE
    0071 8580          332          BITA    #MODE_TEST      SELF-TEST MODE?
    0073 270C          333          BEQ     JNO_CHAR        BRANCH IF NOT
                       334          ENDIF
    0075 CE04A1        335          LDX     #STRING         ELSE SET THE POINTER TO THE STRING
    0078 DF3B          336          STX     BUFFER_POINTER,D
                       337          IF      SHANNON
                       338          LDD     #STRING_END-STRING SET THE BYTE COUNT
                       339          STD     BUFFER_COUNT,D
                       340          ELSE
    007A 8666          341          LDAA    #STRING_END-STRING SET THE BYTE COUNT
    007C 973D          342          STAA    BUFFER_COUNT,D
    007E 7E00A5        343          JMP     PRINT_ONE       AND GO PRINT IT
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     344
     0081            345 JNO_CHAR
     0081 7E015E     346          JMP      NO_CHAR          GO TIMEOUT MOTORS
                     347
     0084            348 COPY_BUFFER
     0084 F60001     349          LDAB     M_LEN+1          COPY THE BYTE COUNT
     0087 D73D       350          STAB     BUFFER_COUNT,D
     0089 C600       351          LDAB     #0               INIT THE OFFSET
     008B            352 MOVE_ONE
     008B CE0000     353          LDX      #NET_IN_BUFF     POINT TO MAC'S BUFFER
     008E 3A         354          ABX                       POINT TO THE NEXT CHAR
     008F A600       355          LDAA     0,X              GET THE CHAR
     0091 CE003E     356          LDX      #BUFFER          POINT TO MY BUFFER
     0094 3A         357          ABX                       POINT TO NEXT FREE SPACE
     0095 A700       358          STAA     0,X              SAVE THE CHAR
     0097 5C         359          INCB                      INC THE OFFSET
     0098 7A0001     360          DEC      M_LEN+1          DEC THE COUNTER
     009B 26EE       361          BNE      MOVE_ONE         LOOP UNTIL ALL ARE COPIED
     009D CE003E     362          LDX      #BUFFER          INIT MY OWN POINTER
     00A0 DF3B       363          STX      BUFFER_POINTER,D
     00A2 7F0000     364          CLR      M_SIG            TELL THE MAC WE GOT IT
                     365          ENDIF
                     366
     00A5            367 PRINT_ONE
     00A5 DE3B       368          LDX      BUFFER_POINTER,D  GET THE POINTER TO THE NEXT CHAR
     00A7 A600       369          LDAA     0,X              GET THE CHAR FROM THE BUFFER
     00A9 2B25       370          BMI      INVALID          BRANCH IF TOP BIT IS SET
     00AB 8120       371          CMPA     #SP
     00AD 2303       372          BLS      SOME_2
     00AF 7E0128     373          JMP      PRINTABLE        BRANCH IF IT'S A PRINTABLE CHARACTER (INCLUDING DEL)
     00B2            374 SOME_2
     00B2 2730       375          BEQ      SPACE            BRANCH IF IT'S A SPACE
     00B4 810D       376          CMPA     #CR
     00B6 274E       377          BEQ      RETURN           BRANCH IF IT'S A CARRIAGE RETURN
     00B8 810A       378          CMPA     #LF
     00BA 2756       379          BEQ      LINEFEED         BRANCH IF IT'S A LINEFEED
     00BC 8108       380          CMPA     #BS
     00BE 2735       381          BEQ      BACKSPACE        BRANCH IF IT'S A BACKSPACE
     00C0 811B       382          CMPA     #ESC
     00C2 2762       383          BEQ      ESCAPE           BRANCH IF IT'S AN ESCAPE
     00C4 810E       384          CMPA     #SO
     00C6 2714       385          BEQ      REVERSE          BRANCH IF IT'S A SHIFT OUT
     00C8 810B       386          CMPA     #VT
     00CA 274F       387          BEQ      HALF_LINEFEED    BRANCH IF IT'S A VERTICAL TAB
     00CC 810F       388          CMPA     #SI
     00CE 2703       389          BEQ      FORWARD          BRANCH IF IT'S A SHIFT IN
     00D0            390 INVALID
     00D0 7E0153     391          JMP      INC              ELSE IGNORE IT
                     392
     00D3            393 FORWARD
     00D3 9660       394          LDAA     MODE,D           GET THE MODE BYTE
     00D5 84BF       395          ANDA     #0FFH-MODE_REVERSE CLEAR REVERSE MODE BIT
     00D7 9760       396          STAA     MODE,D
     00D9 7E0153     397          JMP      INC
                     398
     00DC            399 REVERSE
     00DC 9660       400          LDAA     MODE,D           GET THE MODE BYTE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

     00DE 8A40          401          ORAA    #MODE_REVERSE   SET REVERSE MODE BIT
     00E0 9760          402          STAA    MODE,D
     00E2 206F          403          BRA     INC
                        404
     00E4               405 SPACE
     00E4 9660          406          LDAA    MODE,D          GET THE MODE BYTE
     00E6 8540          407          BITA    #MODE_REVERSE   CHECK FOR REVERSE MODE
     00E8 2611          408          BNE     GOLEFT          BRANCH IF REVERSE MODE IN EFFECT
     00EA               409 GORIGHT
     00EA 9663          410          LDAA    PHANTOM,D       GET THE PHANTOM COLUMN NUMBER
     00EC 8150          411          CMPA    #PLWIDTH        IS THERE ROOM TO GO TO THE RIGHT?
     00EE 2263          412          BHI     INC             BRANCH IF NOT
     00F0 7C0063        413          INC     PHANTOM         ELSE INC. IT
     00F3 205E          414          BRA     INC
                        415
     00F5               416 BACKSPACE
     00F5 9660          417          LDAA    MODE,D          GET THE MODE BYTE
     00F7 8540          418          BITA    #MODE_REVERSE   CHECK FOR REVERSE MODE
     00F9 26EF          419          BNE     GORIGHT         BRANCH IF REVERSE MODE IN EFFECT
     00FB               420 GOLEFT
     00FB 9663          421          LDAA    PHANTOM,D       GET THE PHANTOM COLUMN NUMBER
     00FD 8101          422          CMPA    #1              IS THERE ROOM TO GO TO THE LEFT
     00FF 2352          423          BLS     INC             BRANCH IF NOT
     0101 7A0063        424          DEC     PHANTOM         ELSE DEC. IT
     0104 204D          425          BRA     INC
                        426
     0106               427 RETURN
     0106 9660          428          LDAA    MODE,D          GET THE MODE BYTE
     0108 84BF          429          ANDA    #0FFH-MODE_REVERSE CANCEL REVERSE MODE
     010A 9760          430          STAA    MODE,D
     010C 8601          431          LDAA    #1              SET THE COLUMN NUMBER
     010E 9763          432          STAA    PHANTOM,D
     0110 2041          433          BRA     INC
                        434
     0112               435 LINEFEED
     0112 964E          436          LDAA    PLATEN,D        GET THE PLATEN COMMAND/RESULT BYTE
     0114 81FF          437          CMPA    #0FFH           IS IT ALREADY FULL?
     0116 27FA          438          BEQ     LINEFEED        WAIT FOR IT IF SO
     0118 7C004E        439          INC     PLATEN          ELSE INC. IT TWICE
                        440
     011B               441 HALF_LINEFEED
     011B 964E          442          LDAA    PLATEN,D        GET THE PLATEN COMMAND/RESULT BYTE
     011D 81FF          443          CMPA    #0FFH           IS IT ALREADY FULL
     011F 27FA          444          BEQ     HALF_LINEFEED   WAIT FOR IT IF SO
     0121 7C004E        445          INC     PLATEN          ELSE INC. IT
     0124 202D          446          BRA     INC
                        447
     0126               448 ESCAPE
     0126 8620          449          LDAA    #SP             GET A SPACE AS THOUGH IT WERE PRINTABLE
                        450
     0128               451 PRINTABLE
     0128 D663          452          LDAB    PHANTOM,D       CHECK THE COLUMN NUMBER
     012A C150          453          CMPB    #PLWIDTH        HAVE WE PAST THE LAST COLUMN?
     012C 2225          454          BHI     INC             BRANCH IF SO -- CAN'T PRINT HERE
     012E 9753          455          STAA    WHEEL,D         GIVE THE CHARACTER TO THE WHEEL ROUTINE
     0130               456 PRINTABLE1
     0130 7D005B        457          TST     CARRIAGE        IS THE CARRIAGE MOVING
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

     0133 26FB          458          BNE     PRINTABLE1       WAIT IF IT IS
     0135 D663          459          LDAB    PHANTOM,D        THEN TELL IT WHERE TO GO
     0137 D75B          460          STAB    CARRIAGE,D
     0139               461 FIRE_WAIT1
     0139 D653          462          LDAB    WHEEL,D          WAIT FOR ALL MOTION TO CEASE
     013B DA5B          463          ORAB    CARRIAGE,D
     013D DA4E          464          ORAB    PLATEN,D
                        465 *        ORAB    HAMMER,D
     013F 26F8          466          BNE     FIRE_WAIT1
                        467
     0141 CE0577        468          LDX     #ENERGIES         POINT TO THE HAMMER ENERGY TABLE
     0144 16            469          TAB
     0145 C020          470          SUBB    #SP               REMOVE PRINTABLE OFFSET
     0147 3A            471          ABX                       ADD OFFSET TO THE TABLE ADDRESS
     0148 A600          472          LDAA    0,X               GET THE ENERGY
     014A 975B          473          STAA    HAMMER,D          TELL THE HAMMER TO FIRE
                        474
     014C               475 FIRE_WAIT2
     014C 7D0058        476          TST     HAMMER            IS THE HAMMER FINISHED
     014F 26FB          477          BNE     FIRE_WAIT2        LOOP IF NOT
     0151 2091          478          BRA     SPACE             THEN MOVE THE CARRIAGE
                        479
                        480 * WHEN WE HAVE FINISHED WITH THE CURRENT CHARACTER, WE COME HERE TO
                        481 * UPDATE THE BUFFER POINTER.
                        482
     0153               483 INC
     0153 DE3B          484          LDX     BUFFER_POINTER,D  GET THE POINTER
     0155 08            485          INX                       INC. IT
     0156 DF3B          486          STX     BUFFER_POINTER,D  UPDATE MEMORY
                        487          IF      SHANNON
                        488          LDX     BUFFER_COUNT,D    DEC. THE CHAR COUNT
                        489          DEX
                        490          STX     BUFFER_COUNT,D
                        491          ELSE
     0158 7A003D        492          DEC     BUFFER_COUNT      DEC. THE CHAR COUNT
                        493          ENDIF
     015B 7E0065        494 JMAIN    JMP     MAIN              AND LOOP FOR THE NEXT CHARACTER
                        495
                        496          IF      1-SHANNON
                        497 * IF THERE ARE NO CHARACTERS IN THE BUFFER TO PROCESS, WE COME HERE.
                        498 * WE MAKE SURE THE CARRIAGE IS IN ITS PROPER POSITION, THEN WE
                        499 * START A TIMER.  IF NO CHARACTERS ARRIVE BEFORE THE TIMER RUNS OUT,
                        500 * WE TURN OFF THE STEPPING MOTORS.
                        501
     015E               502 NO_CHAR
     015E 7D005B        503          TST     CARRIAGE          IS THE CARRIAGE ALREADY IN MOTION
     0161 26F8          504          BNE     JMAIN             BRANCH IF SO -- WE'LL LOOK AGAIN LATER
     0163 9663          505          LDAA    PHANTOM,D         ELSE GET ITS PROPER POSITION
     0165 975B          506          STAA    CARRIAGE,D        TELL CARRIAGE TO GO THERE
                        507
     0167 CCFFFF        508          LDD     #0FFFFH           SET THE TIMER
     016A DD64          509          STD     CLOCK,D
     016C               510 TIMEOUT
     016C 7D0000        511          TST     M_SIG             HAS THE MAC ANYTHING FOR ME?
     016F 26EA          512          BNE     JMAIN             BRANCH TO THE TOP IF SO
     0171 DC64          513          LDD     CLOCK,D           HOW'S THE TIMER DOING
     0173 26F7          514          BNE     TIMEOUT           LOOP IF IT'S NOT ZERO YET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     515
     0175 86FF       516          LDAA    #11111111B      TURN OFF THE STEPPING MOTORS
     0177 9702       517          STAA    MOTORS
                     518
                     519 * NOW WE JUST SIT HERE WAITING FOR A CHARACTER TO ARRIVE.
                     520
     0179            521 AWAIT_CHAR
     0179 7D0000     522          TST     M_SIG           HAS THE BUFFER ANY NEWS?
     017C 27FB       523          BEQ     AWAIT_CHAR      . LOOP IF NOT
                     524
                     525 * WHEN ONE DOES ARRIVE, WE RE-POWER THE MOTORS.  BECAUSE THEY MIGHT
                     526 * HAVE SHIFTED POSITION WHILE THE POWER WAS OFF, WE HAVE TO HOME THE
                     527 * CARRIAGE AND THE DAISY WHEEL.
                     528
     017E            529 GOT_CHAR
     017E 9661       530          LDAA    IMAGE,D         GET THE LAST KNOWN VALUE FOR THE MOTORS
     0180 9702       531          STAA    MOTORS          TURN THEM ON AGAIN
     0182 BD0190     532          JSR     CARHOME         HOME THE CARRIAGE
     0185 BD01E3     533          JSR     WHEELHOME       HOME THE WHEEL
     0188 9662       534          LDAA    HOME_IMAGE,D    FORCE MOTORS TO SAME AS POWER-UP
     018A 9761       535          STAA    IMAGE,D
     018C 9702       536          STAA    MOTORS
     018E 20CB       537          BRA     JMAIN           AND RETURN TO THE TOP OF THE LOOP
                     538          ENDIF
                     539
                     540 ********************************************************************************
                     541 * CARHOME -- THIS ROUTINE MOVES THE CARRIAGE TO THE LEFT MARGIN.
                     542 * THE CARRIAGE IS MOVED AT A CONSTANT, SLOW SPEED WHILE WE WATCH FOR
                     543 * THE LEFT MARGIN SWITCH.
                     544
     0190            545 CARHOME
                     546
                     547 * FIRST WE MOVE THE CARRIAGE TO THE RIGHT, IN CASE IT IS LEFT OF THE
                     548 * LEFT MARGIN SWITCH.
                     549
     0190 8D2D       550          BSR     STEP_RIGHT      MOVE ONE STEP RIGHT
     0192 8D41       551          BSR     CAR_DELAY       LET MOTOR SETTLE
     0194 9607       552          LDAA    BITS            GET THE STATUS BYTE
     0196 8510       553          BITA    #BITS_LEFT      LOOK AT THE LEFT MARGIN SWITCH
     0198 26F6       554          BNE     CARHOME         LOOP IF WE'RE STILL AT LEFT MARGIN
                     555
                     556 * AFTER WE'RE SURE WE'RE TO THE RIGHT OF THE DESIRED POSITION, WE
                     557 * CAN MOVE THE CARRIAGE LEFT UNTIL WE HIT THE LEFT MARGIN SWITCH.
                     558
     019A            559 CARHOME2
     019A 8D0D       560          BSR     STEP_LEFT       MOVE ONE STEP LEFT
     019C 8D37       561          BSR     CAR_DELAY       LET MOTOR SETTLE
     019E 9607       562          LDAA    BITS            GET THE STATUS BYTE
     01A0 8510       563          BITA    #BITS_LEFT      LOOK AT THE LEFT MARGIN SWITCH
     01A2 27F6       564          BEQ     CARHOME2        LOOP IF WE'RE NOT AT LEFT MARGIN
                     565
                     566 * THEN WE UPDATE THE POSITION BYTE TO REFLECT OUR NEW POSITION.
                     567
     01A4 8601       568          LDAA    #1              GET COLUMN NUMBER
     01A6 975F       569          STAA    CARPOS,D        AND SAVE IT
                     570
     01A8 39         571          RTS
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     572
                     573 ********************************************************************************
                     574 * STEP_LEFT -- THIS ROUTINE PERMUTES THE ORDER OF THE BITS IN THE
                     575 * MOTOR PORT FROM 76543210 TO 76540321, THUS MOVING THE CARRIAGE
                     576 * ONE STEP TO THE LEFT.
                     577
    01A9             578 STEP_LEFT
    01A9 9661        579         LDAA    IMAGE,D         GET THE CURRENT PATTERN
    01AB 16          580         TAB                     PUT INTO BOTH ACCUMULATORS
    01AC 44          581         LSRA                    MOVE SOME BITS ONE POSITION RIGHT
    01AD 58          582         LSLB                    MOVE OTHER BITS 3 PLACES LEFT
    01AE 58          583         LSLB
    01AF 58          584         LSLB
    01B0 8407        585         ANDA    #00000111B      PICK OUT THE BITS WE WANT
    01B2 C408        586         ANDB    #00001000B
    01B4 1B          587         ABA                     MIX THEM TOGETHER
    01B5 D661        588         LDAB    IMAGE,D
    01B7 C4F0        589         ANDB    #11110000B      PICK UP THE BITS THAT DON'T MOVE
    01B9 1B          590         ABA                     MIX THEM IN TOO
    01BA 9761        591         STAA    IMAGE,D         UPDATE THE IMAGE
    01BC 9702        592         STAA    MOTORS          UPDATE THE PORT
    01BE 39          593         RTS
                     594
                     595 ********************************************************************************
                     596 * STEP_RIGHT -- THIS ROUTINE PERMUTES THE ORDER OF THE BITS IN THE
                     597 * MOTOR PORT FROM 76543210 TO 76542103, THUS MOVING THE CARRIAGE
                     598 * ONE STEP TO THE RIGHT.
                     599
    01BF             600 STEP_RIGHT
    01BF 9661        601         LDAA    IMAGE,D         GET THE CURRENT PATTERN
    01C1 16          602         TAB                     PUT INTO BOTH ACCUMULATORS
    01C2 48          603         LSLA                    MOVE SOME BITS ONE POSITION LEFT
    01C3 54          604         LSRB                    MOVE OTHER BITS 3 PLACES RIGHT
    01C4 54          605         LSRB
    01C5 54          606         LSRB
    01C6 840E        607         ANDA    #00001110B      PICK OUT THE BITS WE WANT
    01C8 C401        608         ANDB    #00000001B
    01CA 1B          609         ABA                     MIX THEM TOGETHER
    01CB D661        610         LDAB    IMAGE,D
    01CD C4F0        611         ANDB    #11110000B      PICK UP THE BITS THAT DON'T MOVE
    01CF 1B          612         ABA                     MIX THEM IN TOO
    01D0 9761        613         STAA    IMAGE,D         UPDATE THE IMAGE
    01D2 9702        614         STAA    MOTORS          UPDATE THE PORT
    01D4 39          615         RTS
                     616
                     617 ********************************************************************************
                     618 * CAR_DELAY -- A ROUTINE TO DELAY FOR THE APPROPRIATE TIME SO AS TO
                     619 * PROVIDE A SLOW STEPPING RATE TO THE CARRIAGE MOTOR.
                     620
    01D5             621 CAR_DELAY
    01D5 DC09        622         LDD     TIMER           GET THE CURRENT TIMER VALUE
    01D7 C373A7      623         ADDD    #CARSPD1*TICK   CALCULATE DESIRED TIMER VALUE
    01DA DD64        624         STD     WANT_TIMER,D    AND SAVE IT
    01DC             625 CAR_DELAY2
    01DC DC09        626         LDD     TIMER           GET THE CURRENT TIMER VALUE
    01DE 9364        627         SUBD    WANT_TIMER,D    COMPARE IT TO DESIRED VALUE
    01E0 2BFA        628         BMI     CAR_DELAY2      LOOP IF NOT YET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    01E2 39           629         RTS
                      630
                      631 ********************************************************************************
                      632 * WHEELHOME -- THIS ROUTINE MOVES THE WHEEL TO ITS HOME POSITION.
                      633 * THE WHEEL IS MOVED AT A CONSTANT, SLOW SPEED WHILE WE WATCH FOR
                      634 * THE WHEEL HOME SENSOR.
                      635
    01E3              636 WHEELHOME
                      637
                      638 * FIRST WE MOVE THE WHEEL TO MAKE SURE IT IS NOT AT THE HOME POSITION.
                      639
    01E3 8D17         640         BSR     ROT_CCW         MOVE ONE STEP
    01E5 8D41         641         BSR     WHEEL_DELAY     LET MOTOR SETTLE
    01E7 9607         642         LDAA    BITS            GET THE STATUS BYTE
    01E9 8540         643         BITA    #BITS_SPOKES    LOOK AT THE WHEEL HOME SWITCH
    01EB 26F6         644         BNE     WHEELHOME       LOOP IF WE'RE STILL AT HOME POSITION
                      645
                      646 * AFTER WE'RE SURE WE'RE NOT AT THE HOME POSITION, WE CAN ROTATE THE
                      647 * WHEEL INTO THE PROPER POSITION.  THE REASON FOR THE DOUBLE CHECK IS
                      648 * SO THAT WE AVOID ANY ERRORS DUE TO THE WIDTH OF THE LITTLE FINGER
                      649 * WHICH TRIPS THE HOME SENSOR.  BY DOING IT THIS WAY WE ARE ALWAYS
                      650 * LOOKING FOR THE SAME EDGE OF THE FINGER, AND ITS WIDTH HAS NO EFFECT.
                      651
    01ED              652 WHEEL2
    01ED 8D0D         653         BSR     ROT_CCW         MOVE ONE STEP
    01EF 8D37         654         BSR     WHEEL_DELAY     LET MOTOR SETTLE
    01F1 9607         655         LDAA    BITS            GET THE STATUS BYTE
    01F3 8540         656         BITA    #BITS_SPOKES    LOOK AT THE WHEEL HOME SWITCH
    01F5 27F6         657         BEQ     WHEEL2          LOOP IF WE'RE NOT AT HOME POSITION
                      658
                      659 * THEN WE UPDATE THE POSITION BYTE TO REFLECT OUR NEW POSITION.
                      660
    01F7 8605         661         LDAA    #HOME_SPOKE     GET THE HOME SPOKE NUMBER
    01F9 9757         662         STAA    WHEELPOS,D      AND SAVE IT
                      663
    01FB 39           664         RTS
                      665
                      666 ********************************************************************************
                      667 * ROT_CCW -- THIS ROUTINE PERMUTES THE ORDER OF THE BITS IN THE
                      668 * MOTOR PORT FROM 76543210 TO 65473210, THUS MOVING THE WHEEL
                      669 * ONE STEP COUNTER-CLOCKWISE.
                      670
    01FC              671 ROT_CCW
    01FC 9661         672         LDAA    IMAGE,D         GET THE CURRENT PATTERN
    01FE 16           673         TAB                     PUT INTO BOTH ACCUMULATORS
    01FF 48           674         LSLA                    MOVE SOME BITS ONE POSITION LEFT
    0200 54           675         LSRB                    MOVE OTHER BITS 3 PLACES RIGHT
    0201 54           676         LSRB
    0202 54           677         LSRB
    0203 84E0         678         ANDA    #11100000B      PICK OUT THE BITS WE WANT
    0205 C410         679         ANDB    #00010000B
    0207 1B           680         ABA                     MIX THEM TOGETHER
    0208 D661         681         LDAB    IMAGE,D
    020A C40F         682         ANDB    #00001111B      PICK UP THE BITS THAT DON'T MOVE
    020C 1B           683         ABA                     MIX THEM IN TOO
    020D 9761         684         STAA    IMAGE,D         UPDATE THE IMAGE
    020F 9702         685         STAA    MOTORS          UPDATE THE PORT
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0211 39            686          RTS
                       687
                       688 ********************************************************************************
                       689 * ROT_CW -- THIS ROUTINE PERMUTES THE ORDER OF THE BITS IN THE
                       690 * MOTOR PORT FROM 76543210 TO 47653210, THUS MOVING THE WHEEL
                       691 * ONE STEP CLOCKWISE.
                       692
    0212               693 ROT_CW
    0212 9661          694          LDAA    IMAGE,D          GET THE CURRENT PATTERN
    0214 16            695          TAB                      PUT INTO BOTH ACCUMULATORS
    0215 44            696          LSRA                     MOVE SOME BITS ONE POSITION RIGHT
    0216 58            697          LSLB                     MOVE OTHER BITS 3 PLACES LEFT
    0217 58            698          LSLB
    0218 58            699          LSLB
    0219 8470          700          ANDA    #01110000B       PICK OUT THE BITS WE WANT
    021B C480          701          ANDB    #10000000B
    021D 1B            702          ABA                      MIX THEM TOGETHER
    021E D661          703          LDAB    IMAGE,D
    0220 C40F          704          ANDB    #00001111B       PICK UP THE BITS THAT DON'T MOVE
    0222 1B            705          ABA                      MIX THEM IN TOO
    0223 9761          706          STAA    IMAGE,D          UPDATE THE IMAGE
    0225 9702          707          STAA    MOTORS           UPDATE THE PORT
    0227 39            708          RTS
                       709
                       710 ********************************************************************************
                       711 * WHEEL_DELAY -- A ROUTINE TO DELAY FOR THE APPROPRIATE TIME SO AS TO
                       712 * PROVIDE A SLOW STEPPING RATE TO THE WHEEL MOTOR.
                       713
    0228               714 WHEEL_DELAY
    0228 DC09          715          LDD     TIMER            GET THE CURRENT TIMER VALUE
    022A C33AA4        716          ADDD    #MSEC15*TICK     CALCULATE DESIRED TIMER VALUE (IN CYCLES)
    022D DD64          717          STD     WANT_TIMER,D     AND SAVE IT
    022F               718 WHEEL_DELAY2
    022F DC09          719          LDD     TIMER            GET THE CURRENT TIMER VALUE
    0231 9364          720          SUBD    WANT_TIMER,D     COMPARE IT TO DESIRED VALUE
    0233 2BFA          721          BMI     WHEEL_DELAY2     LOOP IF NOT YET
    0235 39            722          RTS
                       723
                       724 ********************************************************************************
                       725 * THIS IS THE START OF THE LOOP WHICH RUNS EACH TIME THERE IS A
                       726 * TIMER COMPARE INTERRUPT (ABOUT 2400 TIMES PER SECOND).
                       727
    0236               728 INT_LOOP
    0236 9608          729          LDAA    TCSR             READ THIS TO START TO CLEAR THE FLAG
    0238 DC09          730          LDD     TIMER            GET CURRENT TIMER CONTENTS
    023A C301A1        731          ADDD    #TICK            CALCULATE WHEN WE WANT ANOTHER INTERRUPT
    023D DD0B          732          STD     OCR              UPDATE COMPARE REGISTER
    023F 7F0008        733          CLR     TCSR             BUT TURN OFF TIMER INTERRUPTS
    0242 0E            734          CLI                      ALLOW SERIAL INTERRUPTS
                       735
                       736 * UPDATE THE CLOCK USED TO TIME THE MOTOR TURN-OFF PERIOD.  IF IT
                       737 * IS ALREADY ZERO, DON'T CHANGE IT.
                       738
    0243               739 DEC_CLOCK
    0243 DC64          740          LDD     CLOCK,D          GET CURRENT CONTENTS
    0245 2705          741          BEQ     DEC_CLOCK_END    EXIT IF ZERO
    0247 830001        742          SUBD    #1               ELSE DECREMENT IT
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    024A DD64          743          STD     CLOCK,D          AND UPDATE IT
    024C               744 DEC_CLOCK_END
                       745
                       746 * SEE IF IT'S TIME TO STEP THE DAISY WHEEL MOTOR.  THIS PIECE OF
                       747 * CODE MAINTAINS ITS OWN STATUS SO THAT IT KNOWS WHAT IT WAS DOING
                       748 * THE LAST TIME WE CAME BY THIS WAY.
                       749
    024C               750 POSWHEEL
    024C 9654          751   .      LDAA    WHSTAT,D         GET THE STATUS BYTE
    024E 8580          752          BITA    #WHSTAT_BUSY     LOOK AT THE ACTIVITY BIT
    0250 263C          753          BNE     WHEEL3           BRANCH IF WE WERE ALREADY DOING SOMETHING
    0252 D653          754          LDAB    WHEEL,D          ELSE SEE IF WE'VE GOTTEN A NEW COMMAND
    0254 2736          755          BEQ     JWH_END          BRANCH IF NOT -- NOTHING TO DO
                       756
                       757 * IF WE HAVE A NEW COMMAND, CONVERT THE ASCII VALUE INTO A SPOKE NUMBER.
                       758
    0256 CE04F7        759          LDX     #SPOKES-SP       POINT TO THE TABLE (MINUS ASCII OFFSET)
    0259 3A            760          ABX                      ADD OFFSET TO TABLE ADDRESS
    025A E600          761          LDAB    0,X              GET THE SPOKE NUMBER
                       762
    025C D057          763          SUBB    WHEELPOS,D       SEE HOW FAR WE HAVE TO MOVE THE WHEEL
    025E 2603          764          BNE     WHEELXX          BRANCH IF WE HAVE TO MOVE IT
    0260 7E02E4        765          JMP     WH_DONE          BRANCH IF IT'S ALREADY WHERE WE WANT IT
                       766
                       767 * WE NOW HAVE A NUMBER IN THE RANGE -95 TO 95 INCLUSIVE (EXCEPT ZERO).
                       768 * TO OPTIMISE WHEEL MOTION WE WANT TO CONVERT IT TO THE RANGE
                       769 * -47 TO 48 INCLUSIVE.  IF IT IS BETWEEN -95 AND -48 WE'LL ADD 96.
                       770 * IF IT IS BETWEEN 49 AND 95 WE'LL SUBTRACT 96.
                       771
    0263               772 WHEELXX
    0263 2B08          773          BMI     NEG              BRANCH IF IT'S A NEGATIVE NUMBER
    0265 C131          774          CMPB    #49
    0267 2D0A          775          BLT     DIR              BRANCH IF 0 < A < 49
    0269 C060          776          SUBB    #96              FIX IT IF A >= 49
    026B 2006          777          BRA     DIR
    026D               778 NEG
    026D C1D0          779          CMPB    #-48
    026F 2E02          780          BGT     DIR              BRANCH IF -48 < A < 0
    0271 CB60          781          ADDB    #96              FIX IT IF A <= -48
                       782
                       783 * NOW, BASED ON THE RESULT FROM ABOVE, WE CAN DECIDE WHICH DIRECTION TO
                       784 * SPIN THE WHEEL.
                       785
    0273               786 DIR
    0273 5D            787          TSTB
    0274 2B08          788          BMI     CW               BRANCH IF WE'RE GOING CLOCKWISE
    0276 8A80          789          ORAA    #WHSTAT_BUSY     SET THE ACTIVITY BIT
    0278 84BF          790          ANDA    #0FFH-WHSTAT_CW  CLEAR THE CLOCKWISE BIT (WE'RE GOING CCW)
    027A 9754          791          STAA    WHSTAT,D
    027C 2005          792          BRA     WH_SETVARS
    027E               793 CW
    027E 8AC0          794          ORAA    #WHSTAT_BUSY.OR.WHSTAT_CW  SET THE BUSY & CW BITS
    0280 9754          795          STAA    WHSTAT,D
    0282 50            796          NEGB                     MAKE THE DISTANCE POSITIVE
                       797
                       798 * NOW WE CAN INIT SOME OF THE VARS WE'LL BE USING WHILE MOVING THE
                       799 * WHEEL.
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     800
                     801 WH_SETVARS
     0283            802          STAB     WHEELTOGO,D      SET HOW FAR WE HAVE TO GO
     0283 D757       803          CLR      WHEELGONE,D      SET HOW FAR WE'VE ALREADY GONE
     0285 7F0056     804          LDAA     #1               FAKE OUT THE COUNTER
     0288 8601       805          STAA     WHEELTICKS,D
     028A 9755       806 JWH_END  BRA      WH_END           THEN EXIT -- THINGS HAPPEN NEXT TIME
     028C 205C       807
                     808 * IF WE WERE ALREADY BUSY DOING SOMETHING, WE COME HERE TO CARRY ON.
                     809
     028E            810 WHEEL3
     028E 7A0055     811          DEC      WHEELTICKS       DEC. THE TICK COUNTER
     0291 2657       812          BNE      WH_END           BRANCH IF IT'S NOT TIME TO DO ANYTHING
     0293 8520       813          BITA     #WHSTAT_SETTLE   IS THE WHEEL SETTLING?
     0295 264D       814          BNE      WH_DONE          BRANCH IF SO -- WE'RE ALL DONE
     0297 8540       815          BITA     #WHSTAT_CW       ELSE LOOK AT THE DIRECTION BIT
     0299 2605       816          BNE      GO_CW            BRANCH IF IT IS CLOCKWISE
     029B BD01FC     817          JSR      ROT_CCW          ELSE STEP THE WHEEL COUNTER-CLOCKWISE
     029E 2003       818          BRA      WHEEL4
     02A0            819 GO_CW
     02A0 BD0212     820          JSR      ROT_CW           STEP THE WHEEL CLOCKWISE
     02A3            821 WHEEL4
     02A3 7C0056     822          INC      WHEELGONE        WE'VE TAKEN ANOTHER STEP
     02A6 7A0057     823          DEC      WHEELTOGO        AND WE'VE GOT ONE LESS TO GO
     02A9 2723       824          BEQ      WH_SETTLE        BRANCH IF WE'VE GOT ZERO TO GO
                     825
                     826 * TO FIGURE HOW LONG TO WAIT BEFORE THE NEXT STEP, WE HAVE TO SEE
                     827 * WHERE WE ARE IN OUR TRAVEL.  CALCULATE THE EXPRESSION:
                     828 * MIN( WHEELGONE, WHEELTOGO, 8 ) AND USE THE RESULT AS AN OFFSET
                     829 * INTO THE SPEED TABLE.  THE NUMBER FROM THE TABLE BECOMES OUR
                     830 * NEW TICK COUNT.
                     831
     02AB D657       832          LDAB     WHEELTOGO,D      GET ONE OF THE TERMS
     02AD D156       833          CMPB     WHEELGONE,D      COMPARE IT TO ANOTHER
     02AF 2302       834          BLS      WHEEL5           BRANCH IF THE ONE WE HAVE IS SMALLER
     02B1 D656       835          LDAB     WHEELGONE,D      ELSE GET THE OTHER ONE
     02B3            836 WHEEL5
     02B3 C108       837          CMPB     #8               COMPARE IT TO THE LAST TERM
     02B5 2302       838          BLS      WHEEL6           BRANCH IF THE ONE WE HAVE IS SMALLER
     02B7 C608       839          LDAB     #8               ELSE GET THE LAST ONE
     02B9            840 WHEEL6
     02B9 CE050E     841          LDX      #WHEELSPEEDS-1   POINT TO THE SPEED TABLE
     02BC 3A         842          ABX                       ADD THE OFFSET
     02BD E600       843          LDAB     0,X              GET THE TICK COUNT
     02BF D755       844          STAB     WHEELTICKS,D     SAVE IT
     02C1 D606       845          LDAB     FUDGE            READ THE FUDGE VALUE
     02C3 53         846          COMB                      COMPLEMENT IT
     02C4 54         847          LSRB                      SHIFT THE NYBBLE DOWN
     02C5 54         848          LSRB
     02C6 54         849          LSRB
     02C7 54         850          LSRB
     02C8 DB55       851          ADDB     WHEELTICKS,D     ADD TO THE CALCULATED VALUE
     02CA D755       852          STAB     WHEELTICKS,D     AND SAVE THE NEW VALUE
     02CC 201C       853          BRA      WH_END           AND LEAVE
                     854
                     855 * IF WE HAVE JUST ARRIVED AT OUR DESTINATION, WE CAN SET UP THE
                     856 * TICK COUNTER FOR THE SETTLING TIME, AND RESET THE STATUS FLAGS.
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     857 * THIS IS ALSO A HANDY TIME TO STORE THE CURRENT WHEEL POSITION.
                     858
     02CE            859 WH_SETTLE
     02CE 86A5       860         LDAA    #WHSETL         SET THE TICK COUNTER
     02D0 9755       861         STAA    WHEELTICKS,D
     02D2 D653       862         LDAB    WHEEL,D         GET THE SPOKE NUMBER AGAIN
     02D4 CE04F7     863         LDX     #SPOKES-SP
     02D7 3A         864         ABX
     02D8 E600       865         LDAB    0,X
     02DA D757       866         STAB    WHEELPOS,D      AND SAVE THAT AS OUR CURRENT POSITION
     02DC 9654       867         LDAA    WHSTAT,D        UPDATE THE STATUS
     02DE 8A20       868         ORAA    #WHSTAT_SETTLE
     02E0 9754       869         STAA    WHSTAT,D
     02E2 2006       870         BRA     WH_END          AND EXIT
                     871
                     872 * IF WE HAVE JUST FINISHED LETTING THE WHEEL SETTLE, OR IF IT HAD
                     873 * NOWHERE TO GO, WE CAN RESET EVERYTHING AND PREPARE FOR THE NEXT
                     874 * TIME AROUND.
                     875
     02E4            876 WH_DONE
     02E4 7F0053     877         CLR     WHEEL           CLEAR THE COMMAND/STATUS BYTE
     02E7 7F0054     878         CLR     WHSTAT          AND SHOW WE ARE DONE
     02EA            879 WH_END
                     880
                     881 * THEN WE ENTER THE CODE WHICH POSITIONS THE CARRIAGE.  THIS CODE
                     882 * MAINTAINS ITS OWN STATUS SO IT CAN RECALL WHAT IT WAS DOING
                     883 * THE LAST TIME IT RAN.
                     884
     02EA            885 POSCAR
     02EA 965C       886         LDAA    CARSTAT,D       GET THE STATUS BYTE
     02EC 8580       887         BITA    #CARSTAT_BUSY   LOOK AT THE ACTIVITY BIT
     02EE 2623       888         BNE     CARRIAGE3       BRANCH IF WE WERE ALREADY DOING SOMETHING
     02F0 D65B       889         LDAB    CARRIAGE,D      ELSE SEE IF WE HAVE GOTTEN A NEW COMMAND
     02F2 271D       890         BEQ     JCAR_END        BRANCH IF NOT -- NOTHING TO DO
                     891
                     892 * IF THIS IS A NEW COMMAND, WE CAN INIT VARIOUS THINGS.  FIRST THING
                     893 * TO DO IS SEE HOW FAR THE CARRIAGE WILL HAVE TO TRAVEL.  NOTE THAT
                     894 * THE COMMAND BYTE IS A COLUMN NUMBER, BUT ALL INTERNAL CALCULATIONS
                     895 * ARE DONE IN MOTOR STEPS.  THERE ARE TWO STEPS PER PRINT COLUMN.
                     896
     02F4 D05F       897         SUBB    CARPOS,D        SEE HOW FAR WE HAVE TO GO
     02F6 2769       898         BEQ     CAR_DONE        BRANCH IF CARRIAGE IS ALREADY THERE
                     899
                     900 * WE NOW HAVE THE DISTANCE IN COLUMNS, IN THE RANGE -79 TO 79.
                     901 * THE SIGN OF THE NUMBER TELLS US WHICH WAY WE'RE GOING TO GO.
                     902 * PRESERVE THAT INFORMATION IN THE STATUS BYTE.
                     903
     02F8 2B08       904         BMI     CAR_LEFT        BRANCH IF GOING LEFT
     02FA 8A80       905         ORAA    #CARSTAT_BUSY   SET THE ACTIVITY BIT
     02FC 847F       906         ANDA    #0FFH-CARSTAT_LEFT SHOW WE'RE NOT GOING LEFT
     02FE 975C       907         STAA    CARSTAT,D
     0300 2005       908         BRA     CAR_SETVARS
     0302            909 CAR_LEFT
     0302 8AC0       910         ORAA    #CARSTAT_BUSY.OR.CARSTAT_LEFT  SHOW WE'RE BUSY GOING LEFT
     0304 975C       911         STAA    CARSTAT,D
     0306 50         912         NEGB                    MAKE THE DISTANCE POSITIVE
                     913
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     914 * NOW WE CAN CONVERT THE DISTANCE (NOW IN THE RANGE 1 TO 79) INTO
                     915 * STEPS BY MULTIPLYING BY 2.  ALTHOUGH THE 6801 HAS A NIFTY
                     916 * MULTIPLY INSTRUCTION, THIS CASE IS EASIER TO DO WITH A SHIFT.
                     917
     0307            918 CAR_SETVARS
     0307 58         919         LSLB                        CONVERT COLUMNS TO STEPS
     0308 D75F       920         STAB    CARTOGO,D           SET HOW FAR WE HAVE TO GO
     030A 7F005E     921         CLR     CARGONE             SET HOW FAR WE'VE ALREADY GONE
     030D 8601       922         LDAA    #1                  FAKE OUT THE TIMER
     030F 975D       923         STAA    CARTICKS,D
     0311 2054       924 JCAR_END BRA    CAR_END             THEN EXIT -- THINGS WILL HAPPEN NEXT TIME
                     925
                     926 * IF WE WERE ALREADY BUSY DOING SOMETHING, WE COME HERE TO CARRY ON.
                     927
     0313            928 CARRIAGE3
     0313 7A005D     929         DEC     CARTICKS            DEC. THE TICK COUNTER
     0316 264F       930         BNE     CAR_END             BRANCH IF IT'S NOT TIME TO DO ANYTHING
     0318 8520       931         BITA    #CARSTAT_SETTLE     IS THE CARRIAGE SETTLING
     031A 2645       932         BNE     CAR_DONE            BRANCH IF SO -- WE'RE ALL DONE NOW
     031C 8540       933         BITA    #CARSTAT_LEFT       LOOK AT THE DIRECTION BIT
     031E 2605       934         BNE     GO_LEFT             BRANCH IF IT IS LEFT
     0320 BD01BF     935         JSR     STEP_RIGHT          ELSE STEP THE CARRIAGE TO THE RIGHT
     0323 2003       936         BRA     CARRIAGE4
     0325            937 GO_LEFT
     0325 BD01A9     938         JSR     STEP_LEFT           STEP THE CARRIAGE TO THE LEFT
     0328            939 CARRIAGE4
     0328 7C005E     940         INC     CARGONE             WE'VE TAKEN ANOTHER STEP
     032B 7A005F     941         DEC     CARTOGO             AND WE'VE GOT ONE LESS TO GO
     032E 2721       942         BEQ     CAR_SETTLE          BRANCH IF WE'VE GOT ZERO TO GO
                     943
                     944 * TO FIGURE HOW LONG TO WAIT BEFORE THE NEXT STEP, WE HAVE TO SEE
                     945 * WHERE WE ARE IN OUR TRAVEL.  CALCULATE THE EXPRESSION:
                     946 * MIN( CARGONE, CARTOGO, 8 ) AND USE THE RESULT AS AN OFFSET
                     947 * INTO THE SPEED TABLE.  THE NUMBER FROM THE TABLE BECOMES OUR
                     948 * NEW TICK COUNT.
                     949
     0330 D65F       950         LDAB    CARTOGO,D           GET ONE OF THE TERMS
     0332 D15E       951         CMPB    CARGONE,D           COMPARE IT TO ANOTHER
     0334 2302       952         BLS     CARRIAGE5           BRANCH IF THE ONE WE HAVE IS SMALLER
     0336 D65E       953         LDAB    CARGONE,D           ELSE GET THE OTHER ONE
     0338            954 CARRIAGE5
     0338 C108       955         CMPB    #8                  COMPARE IT TO THE LAST TERM
     033A 2302       956         BLS     CARRIAGE6           BRANCH IF THE ONE WE HAVE IS SMALLER
     033C C608       957         LDAB    #8                  ELSE GET THE LAST ONE
     033E            958 CARRIAGE6
     033E CE0506     959         LDX     #CARSPEEDS-1        POINT TO THE SPEED TABLE
     0341 3A         960         ABX                         ADD THE OFFSET
     0342 E600       961         LDAB    0,X                 GET THE TICK COUNT
     0344 D75D       962         STAB    CARTICKS,D          SAVE IT
     0346 D606       963         LDAB    FUDGE               READ THE FUDGE PORT
     0348 53         964         COMB                        COMPLEMENT IT
     0349 C40F       965         ANDB    #0FH                ZERO THE HIGH (WHEEL'S) NYBBLE
     034B DB5D       966         ADDB    CARTICKS,D          ADD TO THE CALCULATED VALUE
     034D D75D       967         STAB    CARTICKS,D          AND SAVE THE NEW VALUE
     034F 2016       968         BRA     CAR_END             AND LEAVE
                     969
                     970 * IF WE HAVE JUST ARRIVED AT OUR DESTINATION, WE CAN SET UP THE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      971 * TICK COUNTER FOR THE SETTLING TIME, AND RESET THE STATUS FLAGS.
                      972 * THIS IS ALSO A HANDY TIME TO STORE THE CURRENT CARRIAGE POSITION.
                      973
     0351             974 CAR_SETTLE
     0351 8664        975         LDAA    #CARSETL        SET THE TICK COUNTER
     0353 975D        976         STAA    CARTICKS,D
     0355 D65B        977         LDAB    CARRIAGE,D      GET THE COLUMN AGAIN
     0357 D75F        978         STAB    CARPOS,D        AND SAVE THAT AS OUR CURRENT POSITION
     0359 965C        979         LDAA    CARSTAT,D       UPDATE THE STATUS
     035B 8A20        980         ORAA    #CARSTAT_SETTLE
     035D 975C        981         STAA    CARSTAT,D
     035F 2006        982         BRA     CAR_END         AND EXIT
                      983
                      984 * IF WE HAVE JUST FINISHED LETTING THE CARRIAGE SETTLE, OR IF IT HAD
                      985 * NOWHERE TO GO, WE CAN RESET EVERYTHING AND PREPARE FOR THE NEXT
                      986 * TIME AROUND.
                      987
     0361             988 CAR_DONE
     0361 7F005B      989         CLR     CARRIAGE        CLEAR THE COMMAND/STATUS BYTE
     0364 7F005C      990         CLR     CARSTAT         CLEAR THE FLAGS
     0367             991 CAR_END
                      992
                      993 * THEN WE GET TO THE CODE WHICH FIRES THE HAMMER AND ADVANCES THE RIBBON.
                      994 * THE COMMAND BYTE FROM THE MAINLINE IS USED AS THE INITIAL VALUE FOR
                      995 * THE COUNTER WHEN THE HAMMER SOLENOID IS ENERGISED.  THIS WAY, THE
                      996 * MAINLINE CAN CONTROL THE HAMMER ENERGY.  THE ONLY TRICKY THING WE DO
                      997 * IS THIS -- AFTER FIRING AND RELEASING THE HAMMER WE AUTOMATICALLY
                      998 * FIRE THE RIBBON ADVANCE SOLENOID.  HOWEVER, WE RETURN THE ZERO STATUS
                      999 * TO THE MAINLINE BEFORE ADVANCING THE RIBBON.  THIS WAY, THE MAINLINE
                     1000 * CAN START MOVING THE WHEEL AND CARRIAGE AND PLATEN EVEN WHILE THE
                     1001 * RIBBON ADVANCE SOLENOID IS FIRED, WHICH IS PERFECTLY ALRIGHT.
                     1002
     0367            1003 FIREHAMMER
     0367 9659       1004         LDAA    HAMSTAT,D       GET OUR STATUS BYTE
     0369 85F0       1005         BITA    #HAMSTAT_HFIRE.OR.HAMSTAT_HREL.OR.HAMSTAT_RFIRE.OR.HAMSTAT_RREL
     036B 2616       1006         BNE     CONTINUE        BRANCH IF WE'RE ALREADY DOING SOMETHING
     036D 9650       1007         LDAA    HAMMER,D        ELSE LOOK AT THE COMMAND BYTE
     036F 276A       1008         BEQ     END             BRANCH IF NOTHING TO DO
     0371            1009 HFIRE
     0371 975A       1010         STAA    HAMTICKS,D      ELSE SET THE COUNTER FOR HAMMER FIRE TIME
     0373 9607       1011         LDAA    BITS            GET THE CURRENT PORT VALUE
     0375 84FE       1012         ANDA    #BITS_HAMMER    TURN ON THE HAMMER BIT (0 = ON)
     0377 9707       1013         STAA    BITS
     0379 9659       1014         LDAA    HAMSTAT,D       SHOW THE HAMMER IS FIRING
     037B 8A80       1015         ORAA    #HAMSTAT_HFIRE
     037D 8801       1016         EORA    #HAMSTAT_ODD    FLIP THE EVERY-OTHER BIT
     037F 9759       1017         STAA    HAMSTAT,D       UPDATE THE STATUS
     0381 2058       1018         BRA     END             AND EXIT
                     1019
                     1020 * ARRIVE HERE IF WE WERE ALREADY DOING SOMETHING.  DECREMENT THE
                     1021 * TICK COUNTER TO SEE IF ANYTHING HAS TO BE CHANGED.
                     1022
     0383            1023 CONTINUE
     0383 7A005A     1024         DEC     HAMTICKS        DEC. THE TICK COUNTER
     0386 2653       1025         BNE     END             BRANCH IF NOT TIME TO DO ANYTHING
                     1026
                     1027 * OK, SO WE'VE DECIDED SOME ACTION NEEDS TO BE TAKEN.  CHECK THE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1028 * BITS IN HAMSTAT (CURRENTLY IN A) TO SEE WHAT WE WERE DOING AND
                     1029 * DECIDE WHAT COMES NEXT.
                     1030
     0388 8580       1031          BITA    #HAMSTAT_HFIRE  WERE WE FIRING THE HAMMER?
     038A 260C       1032          BNE     HREL            GO TO RELEASE IT IF SO
     038C 8540       1033          BITA    #HAMSTAT_HREL   WERE WE RELEASING THE HAMMER?
     038E 261A       1034          BNE     RFIRE           GO TO ADVANCE THE RIBBON IF SO
     0390 8520       1035          BITA    #HAMSTAT_RFIRE  WERE WE ADVANCING THE RIBBON?
     0392 262F       1036          BNE     RREL            GO TO RELEASE IT IF SO
     0394 8510       1037          BITA    #HAMSTAT_RREL   WERE WE RELEASING THE RIBBON?
     0396 263D       1038          BNE     DONE            GO TO FINISH UP IF SO
                     1039
                     1040 * HERE WE RELEASE THE HAMMER SOLENOID AND RESET THE TICK COUNTER.
                     1041
     0398            1042 HREL
     0398 8A40       1043          ORAA    #HAMSTAT_HREL   SHOW OUR NEW STATUS
     039A 847F       1044          ANDA    #0FFH-HAMSTAT_HFIRE
     039C 9759       1045          STAA    HAMSTAT,D
     039E 9607       1046          LDAA    BITS            GET THE PORT IMAGE
     03A0 8A01       1047          ORAA    #0FFH-BITS_HAMMER TURN OFF THE HAMMER BIT
     03A2 9707       1048          STAA    BITS
     03A4 8605       1049          LDAA    #HAMREL         SET THE NEW COUNTER
     03A6 975A       1050          STAA    HAMTICKS,D
     03A8 2031       1051          BRA     END             AND EXIT
                     1052
                     1053 * HERE WE CLEAR THE COMMAND/STATUS BYTE AND FIRE THE RIBBON
                     1054 * ADVANCE SOLENOID.
                     1055
     03AA            1056 RFIRE
     03AA 7F0058     1057          CLR     HAMMER          TELL THE MAINLINE TO CARRY ON
     03AD 8501       1058          BITA    #HAMSTAT_ODD    IS THIS THE TIME TO FIRE THE RIBBON?
     03AF 2724       1059          BEQ     DONE            BRANCH IF NOT
     03B1 8A20       1060          ORAA    #HAMSTAT_RFIRE  SHOW OUR NEW STATUS
     03B3 84BF       1061          ANDA    #0FFH-HAMSTAT_HREL
     03B5 9759       1062          STAA    HAMSTAT,D
     03B7 9607       1063          LDAA    BITS            GET THE PORT IMAGE
     03B9 84FD       1064          ANDA    #BITS_RIBBON    TURN ON THE RIBBON ADVANCE BIT
     03BB 9707       1065          STAA    BITS
     03BD 861B       1066          LDAA    #RIBFIRE        SET THE NEW COUNTER
     03BF 975A       1067          STAA    HAMTICKS,D
     03C1 2018       1068          BRA     END             AND EXIT
                     1069
                     1070 * IN THIS SECTION WE RELEASE THE RIBBON ADVANCE SOLENOID.
                     1071
     03C3            1072 RREL
     03C3 8A10       1073          ORAA    #HAMSTAT_RREL   SHOW OUR NEW STATUS
     03C5 84DF       1074          ANDA    #0FFH-HAMSTAT_RFIRE
     03C7 9759       1075          STAA    HAMSTAT,D
     03C9 9607       1076          LDAA    BITS            GET THE PORT IMAGE
     03CB 8A02       1077          ORAA    #0FFH-BITS_RIBBON TURN OFF THE RIBBON ADVANCE BIT
     03CD 9707       1078          STAA    BITS
     03CF 860A       1079          LDAA    #RIBREL         SET THE NEW COUNTER
                     1080
     03D1 975A       1081          STAA    HAMTICKS,D
     03D3 2006       1082          BRA     END             AND EXIT
                     1083
                     1084 * IF WE ARE DONE ALL PHASES OF THIS THING, WE COME HERE TO
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1085 * CLEAR OUR STATUS BYTE.
                     1086
     03D5            1087 DONE
     03D5 9659       1088          LDAA    HAMSTAT,D          TURN OFF THE OLD STATUS BITS
     03D7 8401       1089          ANDA    #HAMSTAT_ODD
     03D9 9759       1090          STAA    HAMSTAT,D
     03DB            1091 END
                     1092
                     1093 * Then we get to this code which controls the linefeed mechanism.
                     1094 * If the contents of PLATEN is non-zero, it means we have to do
                     1095 * some linefeeds.  This routine uses PLATSTAT to remember its
                     1096 * internal states between execution times.
                     1097 * THIS VERSION RUNS THE MOTOR FORWARD UNTIL THE SENSE SWITCH CLOSES,
                     1098 * RUNS UNTIL THE SWITCH OPENS, BRAKES FOR 255 TICKS, THEN IDLES.
                     1099
     03DB            1100 PLATPOS
     03DB 964F       1101          LDAA    PLATSTAT,D      GET OUR STATUS BYTE
     03DD 2610       1102          BNE     PL_CONTINUE     BRANCH IF WE'RE ALREADY DOING SOMETHING
     03DF 964E       1103          LDAA    PLATEN,D        ELSE LOOK AT THE COMMAND BYTE
     03E1 273C       1104          BEQ     PL_END_ISLE     BRANCH IF NOTHING TO DO
                     1105
     03E3 8680       1106          LDAA    #PLATSTAT_RUN1  SET THE STATUS BYTE TO RUN MODE
     03E5 974F       1107          STAA    PLATSTAT,D
     03E7 CC12C0     1108          LDD     #PL_TIMEOUT     INIT FOR 2 SEC. TIMEOUT
     03EA FD0051     1109          STD     FIRE_PROOF      FOR MOTOR RUN
     03ED 2030       1110          BRA     PL_END_ISLE     AND LEAVE
                     1111
     03EF            1112 PL_CONTINUE
     03EF 8580       1113          BITA    #PLATSTAT_RUN1  IS THE MOTOR RUNNING UNTIL CLOSURE?
     03F1 260E       1114          BNE     PL_RUN1         BRANCH IF SO
     03F3 8540       1115          BITA    #PLATSTAT_RUN2  ARE WE LETTING THE SWITCH OPEN?
     03F5 262A       1116          BNE     PL_RUN2         BRANCH IF SO
     03F7 8520       1117          BITA    #PLATSTAT_STOP  IS THE MOTOR STOPPING?
     03F9 2650       1118          BNE     PL_STOP         BRANCH IF SO
     03FB 8510       1119          BITA    #PLATSTAT_SETTLE IS THE MOTOR SETTLING?
     03FD 2661       1120          BNE     PL_SETTLE       BRANCH IF SO
     03FF 201E       1121          BRA     PL_END_ISLE     THIS SHOULD NEVER RUN
                     1122
                     1123 * THIS PART RUNS THE MOTOR UNTIL THE SENSE INPUT GOES LOW (SWITCH CLOSES).
                     1124 * WE USE THE 3 LOWEST BITS OF PLATSTAT AS A COUNTER TO
                     1125 * DETERMINE WHEN TO TURN ON AND OFF THE MOTOR AS WE RUN IT
                     1126 * IN PULSED MODE.
                     1127
     0401            1128 PL_RUN1
     0401 FC0051     1129          LDD     FIRE_PROOF      HAVE WE DIED YET?
     0404 830001     1130          SUBD    #1
     0407 FD0051     1131          STD     FIRE_PROOF
     040A 2765       1132          BEQ     WE_DIED         GO AWAY
     040C BD0486     1133          JSR     PULSE_MOTOR     RUN THE MOTOR IN PULSED MODE
     040F 9607       1134          LDAA    BITS            LOOK AT THE SENSE BIT
     0411 8520       1135          BITA    #BITS_DETENT
     0413 2656       1136          BNE     PL_END          BRANCH IF IT'S STILL OPEN
     0415 8640       1137          LDAA    #PLATSTAT_RUN2  ELSE GO TO RUN2 MODE
     0417 974F       1138          STAA    PLATSTAT,D
     0419 CC12C0     1139          LDD     #PL_TIMEOUT
     041C FD0051     1140          STD     FIRE_PROOF      RESET COUNTER FOR RUN2
     041F            1141 PL_END_ISLE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    041F 204A        1142          BRA     PL_END          AND LEAVE
                     1143
                     1144 * THIS PART CONTINUES TO RUN THE MOTOR UNTIL THE SWITCH CLOSES
                     1145 * AGAIN.
    0421             1146 PL_RUN2
    0421 FC0051      1147          LDD     FIRE_PROOF      HAVE WE DIED YET?
    0424 830001      1148          SUBD    #1
    0427 FD0051      1149          STD     FIRE_PROOF
    042A 2745        1150          BEQ     WE_DIED         GO AWAY
    042C BD0486      1151          JSR     PULSE_MOTOR     RUN THE MOTOR
    042F 9607        1152          LDAA    BITS            READ THE REGISTER
    0431 8520        1153          BITA    #BITS_DETENT    LOOK AT THE SWITCH
    0433 2736        1154          BEQ     PL_END          BRANCH IF SWITCH IS STILL CLOSED
    0435 9607        1155          LDAA    BITS            ELSE BRAKE THE MOTOR
    0437 8A04        1156          ORAA    #0FFH-BITS_LF
    0439 9707        1157          STAA    BITS            (TURN OFF THE DRIVE TRANSISTOR
    043B 01          1158          NOP                      FIRST AND LET IT REST)
    043C 01          1159          NOP
    043D 84F7        1160          ANDA    #BITS_BRAKE
    043F 9707        1161          STAA    BITS
    0441 867D        1162          LDAA    #PL_BRAKE_COUNT SET THE TICK COUNT
    0443 9750        1163          STAA    PLATTICKS,D
    0445 8620        1164          LDAA    #PLATSTAT_STOP  GO TO STOP MODE
    0447 974F        1165          STAA    PLATSTAT,D
    0449 2020        1166          BRA     PL_END
                     1167
    044B             1168 PL_STOP
    044B 7A0050      1169          DEC     PLATTICKS       COUNT DOWN THE REVERSE TIME
    044E 261B        1170          BNE     PL_END          LEAVE IF MORE TO DO
    0450 9607        1171          LDAA    BITS            ELSE TURN OFF THE MOTOR
    0452 8A0C        1172          ORAA    #(0FFH-BITS_BRAKE).OR.(0FFH-BITS_LF)
    0454 9707        1173          STAA    BITS
    0456 8601        1174          LDAA    #PL_SETTLE_COUNT SET THE TICK COUNT
    0458 9750        1175          STAA    PLATTICKS,D
    045A 8610        1176          LDAA    #PLATSTAT_SETTLE GO TO SETTLE MODE
    045C 974F        1177          STAA    PLATSTAT,D
    045E 200B        1178          BRA     PL_END
                     1179
    0460             1180 PL_SETTLE
    0460 7A0050      1181          DEC     PLATTICKS       COUNT DOWN THE REVERSE TIME
    0463 2606        1182          BNE     PL_END          LEAVE IF MORE TO DO
    0465 7F004F      1183          CLR     PLATSTAT        ELSE SHOW WE'RE DONE
    0468 7A004E      1184          DEC     PLATEN          ONE LINEFEED FINISHED
                     1185
    046B             1186 PL_END
                     1187
                     1188 ********************************************************************
                     1189 * THIS IS THE CLEANUP AT THE END OF THE INTERRUPT ROUTINE.
                     1190
    046B 0F          1191          SEI                     DISABLE INTERRUPTS HERE
    046C 8608        1192          LDAA    #TCSR_EOCI      RE-ENABLE TIMER INTERRUPTS
    046E 9708        1193          STAA    TCSR
    0470 3B          1194          RTI
                     1195
                     1196 ********************************************************************
                     1197 * IF WE GOT HERE, THE LF MOTOR TIMEDOUT. KILL ALL INTERRUPTS,
                     1198 * TURN OFF MOTORS, AND GO TO SLEEP
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1199
     0471            1200 WE_DIED
     0471 0F         1201          SEI
     0472 8600       1202          LDAA    #0
     0474 9708       1203          STAA    TCSR              DISABLE TIMER INTERRUPTS
     0476 8604       1204          LDAA    #RMCR_CC0         CONFIGURE SCI -- NO INTERRUPTS
     0478 9710       1205          STAA    RMCR              SET BAUD RATE AND MODE
     047A 860A       1206          LDAA    #TRCSR_TE.OR.TRCSR_RE
     047C 9711       1207          STAA    TRCSR
     047E 9607       1208          LDAA    BITS              TURN OFF THE LINEFEED MOTOR
     0480 8A0C       1209          ORAA    #(0FFH-BITS_BRAKE).OR.(0FFH-BITS_LF)
     0482 9707       1210          STAA    BITS
     0484            1211 INF_LOOP
     0484 20FE       1212          BRA     INF_LOOP          EVEN THOUGH WE HAVE ALL INTERRUPTS
                     1213 *                                  DISABLED, MAKE SURE WE NEVER COME
                     1214 *                                  BACK
                     1215
                     1216 ************************************************************************
                     1217 * THIS ROUTINE RUNS THE PLATEN MOTOR IN PULSED DRIVE MODE.
                     1218 * IT INCREMENTS PLATTICKS AND USES THE BOTTOM THREE BITS
                     1219 * TO DETERMINE WHETHER THE MOTOR SHOULD BE ON OR OFF.
                     1220
     0486            1221 PULSE_MOTOR
     0486 7C0050     1222          INC     PLATTICKS         INC THE PULSE COUNTER
     0489 9650       1223          LDAA    PLATTICKS,D       GET THE COUNT
     048B 8407       1224          ANDA    #00000111B        LOOK ONLY AT THE BOTTOM 3 BITS
     048D 8106       1225          CMPA    #PL_SPEED         COMPARE IT TO OUR SPEED THRESHOLD
     048F 2507       1226          BLO     PL_RUN_ON         BRANCH IF THE MOTOR SHOULD BE ON
     0491            1227 PL_RUN_OFF
     0491 9607       1228          LDAA    BITS              TURN OFF THE MOTOR
     0493 8A0C       1229          ORAA    #(0FFH-BITS_BRAKE).OR.(0FFH-BITS_LF)
     0495 9707       1230          STAA    BITS
     0497 39         1231          RTS
     0498            1232 PL_RUN_ON
     0498 9607       1233          LDAA    BITS              TURN ON THE MOTOR
     049A 8A08       1234          ORAA    #0FFH-BITS_BRAKE (MAKE SURE BRAKE IS OFF)
     049C 84FB       1235          ANDA    #BITS_LF
     049E 9707       1236          STAA    BITS
     04A0 39         1237          RTS
                     1238
                     1239          IF      SHANNON
                     1240
                     1241 ************************************************************************
                     1242 * THIS IS THE STRING TO BE PRINTED IN SELF-TEST MODE.
                     1243
                     1244 STRING
                     1245          FCC     'The head and in frontal attack on english writer    '
                     1246          FCB     LF,SO
                     1247          FCC     'rehtona erofereht si tniop siht fo retcarahc eht taht'
                     1248          FCB     CR,LF
                     1249          FCC     'method for the letters that the time of whoever told  '
                     1250          FCB     LF,SO
                     1251          FCC     'latnorf ni dna daeh ehl  .detcepxenu na rof melborp eht'
                     1252          FCB     CR,LF
                     1253          FCC     'attack on english writer that the character of              '
                     1254          FCB     LF,SO
                     1255          FCC     'taht srettel eht rof dohten rehtona erofereht si tniop siht'
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1256          FCB     CR,LF
                     1257          FCC     'the time of whoever told the problem for an unexpected.'
                     1258          FCB     LF,SO
                     1259          FCC     '   taht retirw hsilgne no kcatta latnorf ni dna daeh ehT'
                     1260          FCB     CR,LF
                     1261          FCC     'the character of this point is therefore another method'
                     1262          FCB     LF,SO
                     1263          FCC     '  -orp eht dlot reveohw fo emit eht taht srettel eht rof'
                     1264          FCB     CR,LF
                     1265          FCC     'blem for an unexpected.'
                     1266          FCB     CR,LF
                     1267 STRING_END EQU  $
                     1268
                     1269          ELSE
                     1270
                     1271 ********************************************************************************
                     1272 * THIS IS THE STRING TO BE PRINTED IN SELF-TEST MODE.
                     1273
     04A1            1274 STRING
     04A1 1B         1275          FCB     ESC
     04A2 2122232425 1276          FCC     '!"#$%&'
     04A7 26
     04A8 27         1277          FCC     "'"
     04A9 28292A2B2C 1278          FCC     '()*+,-./0123456789:;<=>?@ABCDEFGH1JKLMNOPQRSTUVWXYZ[\]^_'
     04AE 2D2E2F3031
     04B3 3233343536
     04B8 3738393A3B
     04BD 3C3D3E3F40
     04C2 4142434445
     04C7 464748494A
     04CC 4B4C4D4E4F
     04D1 5051525354
     04D6 5556575859
     04DB 5A5B5C5D5E
     04E0 5F
     04E1 0A080E7F    1279          FCB     LF,BS,SO,DEL
     04E5 7E7D7C7B7A  1280          FCC     "~}|{zyxwvutsrqponmlkjihgfedcba`"
     04EA 7978777675
     04EF 7473727170
     04F4 6F6E6D6C6B
     04F9 6A69686766
     04FE 6564636261
     0503 60
     0504 0D0A0A      1281          FCB     CR,LF,LF
          (0507)      1282 STRING_END EQU  $
                      1283
                      1284          ENDIF
                      1285
                      1286 ********************************************************************************
                      1287 * THIS TABLE IS USED TO CALCULATE CARRIAGE MOTOR STEPPING TIMES.
                      1288 * THE FIRST ENTRIES ARE USED FOR SLOWER SPEED, AS THE CARRIAGE LEAVES
                      1289 * ITS STARTING POINT AND NEARS ITS DESTINATION.  THE LAST ENTRY IS
                      1290 * TOP SPEED USED DURING MOST OF THE TRANSIT.
                      1291
     0507             1292 CARSPEEDS
     0507 47          1293          FCB     CARSPD1
     0508 32          1294          FCB     CARSPD2
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0509 29          1295         FCB     CARSPD3
    050A 23          1296         FCB     CARSPD4
    050B 20          1297         FCB     CARSPD5
    050C 1D          1298         FCB     CARSPD6
    050D 1B          1299         FCB     CARSPD7
    050E 19          1300         FCB     CARSPD8
                     1301
                     1302 ********************************************************************************
                     1303 * THIS TABLE IS USED TO CALCULATE WHEEL MOTOR STEPPING TIMES.
                     1304 * THE FIRST ENTRIES ARE USED FOR SLOWER SPEED, AS THE WHEEL LEAVES
                     1305 * ITS STARTING POINT AND NEARS ITS DESTINATION.  THE LAST ENTRY IS
                     1306 * TOP SPEED USED DURING MOST OF THE TRANSIT.
                     1307
    050F             1308 WHEELSPEEDS
    050F 15          1309         FCB     WHSPD1
    0510 12          1310         FCB     WHSPD2
    0511 0F          1311         FCB     WHSPD3
    0512 0D          1312         FCB     WHSPD4
    0513 0B          1313         FCB     WHSPD5
    0514 09          1314         FCB     WHSPD6
    0515 08          1315         FCB     WHSPD7
    0516 07          1316         FCB     WHSPD8
                     1317
                     1318 ********************************************************************************
                     1319 * THIS TABLE RELATES ASCII CHARACTERS TO THEIR POSITIONS ON THE PRINT
                     1320 * WHEEL.  THE DATA APPLIES TO 96-CHARACTER PLASTIC WHEELS.  TO USE THE
                     1321 * TABLE, CALCULATE AN OFFSET WHICH IS CHAR) - 020,HAND THE BYTE THERE
                     1322 * IS THE SPOKE NUMBER WHERE THAT CHAR CAN BE FOUND.  ALTHOUGH THE SPOKE
                     1323 * FOR ASCII SPACE IS NOT NORMALLY PRINTED, IT IS INCLUDED HERE BECAUSE A
                     1324 * SPECIAL CODE CAN BE USED TO ACCESS IT.  VALID CHARACTERS ARE IN THE
                     1325 * RANGE 020HTO 07F.H
                     1326
    0517             1327 SPOKES
    0517 0244462E2C  1328         FCB     2,68,70,46,44,47,69,54     to '
    051C 2F4536
    051F 3C3A3D2D03  1329         FCB     60,58,61,45,3,43,5,66    ( to /
    0524 2B0542
    0527 2521222324  1330         FCB     37,33,34,35,36,38,39,40  0 to 7
    052C 262728
    052F 292A0C1F39  1331         FCB     41,42,12,31,57,48,50,65  8 to
    0534 303241
    0537 3E0B080A16  1332         FCB     62,11,8,10,22,15,9,24    @ to G
    053C 0F0918
    053F 11141D1C15  1333         FCB     17,20,29,28,21,6,19,18   H to O
    0544 061312
    0547 1A1B0D0E10  1334         FCB     26,27,13,14,16,23,30,4   P to W
    054C 171E04
    054F 201907353F  1335         FCB     32,25,7,53,63,51,64,55   X to _
    0554 334037
    0557 38544E4F4C  1336         FCB     56,84,78,79,76,83,89,74  ` to g
    055C 53594A
    055F 5755485D4D  1337         FCB     87,85,72,93,77,71,82,80  h to o
    0564 475250
    0567 5A5C515856  1338         FCB     90,92,81,88,86,91,73,0   p to w
    056C 5B4900
    056F 4B5E5F313B  1339         FCB     75,94,95,49,59,67,52,1   x to DEL)
    0574 433401
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1340
                     1341 ********************************************************************************
                     1342 * THIS TABLE GIVES THE HAMMER ENERGY FOR EACH PRINTABLE CHARACTER.
                     1343 * TO USE IT, CALCULATE AN OFFSET WHICH IS CHAR) - 020,HAND
                     1344 * THE BYTE THERE IS THE HAMMER ENERGY FOR THAT CHARACTER.  VALID
                     1345 * CHARACTERS ARE IN THE RANGE 020H TO 07F.H
                     1346
     0577            1347 ENERGIES
     0577 0706060707 1348          FCB     E3,E2,E2,E3,E3,E3,E3,E1   to '
     057C 070705
     057F 0606070605 1349          FCB     E2,E2,E3,E2,E1,E1,E1,E2 ( to /
     0584 050506
     0587 0706070607 1350          FCB     E3,E2,E3,E2,E3,E3,E3,E2 0 to 7
     058C 070706
     058F 0707050606 1351          FCB     E3,E3,E1,E2,E2,E3,E2,E2 8 to
     0594 070606
     0597 0807080708 1352          FCB     E4,E3,E4,E3,E4,E4,E4,E4 @ to G
     059C 080808
     059F 0806070808 1353          FCB     E4,E2,E3,E4,E4,E4,E4,E3 H to O
     05A4 080807
     05A7 0808080707 1354          FCB     E4,E4,E4,E3,E3,E4,E4,E4 P to W
     05AC 080808
     05AF 0808070606 1355          FCB     E4,E4,E3,E2,E2,E2,E1,E2 X to _
     05B4 060506
     05B7 0506080607 1356          FCB     E1,E2,E4,E2,E3,E2,E3,E4 ` to g
     05BC 060708
     05BF 0706070706 1357          FCB     E3,E2,E3,E3,E2,E4,E3,E3 h to o
     05C4 080707
     05C7 0808060706 1358          FCB     E4,E4,E2,E3,E2,E3,E2,E3 p to w
     05CC 070607
     05CF 0708070606 1359          FCB     E3,E4,E3,E2,E2,E2,E1,E1 x to DEL)
     05D4 060505
                     1360
                     1361
                     1362 ********************************************************************************
                     1363 * This is a null interrupt routine that we can aim the unused interrupt
                     1364 * vectors at.  It should never be executed, but better to have it than
                     1365 * to have the machine hang...
                     1366
     05D7            1367 NULL
     05D7 3B         1368          RTI
                     1369
                     1370 ********************************************************************************
                     1371 * THESE ARE THE VECTORS FOR THE 6801.  MANY ARE UNUSED IN THIS SOFTWARE,
                     1372 * SO THEY JUST POINT TO A NULL INTERRUPT ROUTINE.
                     1373
                     1374          IF      1-SHANNON
                     1375          ORG     0FFF0H
     FFF0 0000       1376          FDB     PR_MAC          SCI INTERRUPT
                     1377          ENDIF
                     1378          ORG     0FFF4H
     FFF4 0236       1379          FDB     INT_LOOP        TIMER COMPARE
                     1380          ORG     0FFFEH
     FFFE 0000       1381          FDB     RESET           RESET
                     1382
                     1383
```

LOCATION OBJECT CODE LINE SOURCE LINE

Errors= 0

LINE#	SYMBOL	TYPE	REFERENCES
521	AWAIT_CHAR	P	523
416	BACKSPACE	P	381
99	BITS	A	258,269,552,562,642,655,1011,1013,1046,1048,1063,1065,1076,1078,1134,1152,1155,1157,1161,1171,1173,1208 1210,1228,1230,1233,1236
104	BITS_BRAKE	A	1160,1172,1209,1229,1234
102	BITS_DETENT	A	1135,1153
107	BITS_HAMMER	A	1012,1047
103	BITS_LEFT	A	553,563
105	BITS_LF	A	1156,1172,1209,1229,1235
106	BITS_RIBBON	A	1064,1077
101	BITS_SPOKES	A	643,656
100	BITS_TEST	A	270
170	BS	A	380
119	BUFFER	D	
115	BUFFER_COUNT	D	318,339,488,490
113	BUFFER_POINTER	D	336,368,484,486
151	CARGONE	D	921,940,951,953
545	CARHOME	P	296,532,554
559	CARHOME2	P	564
153	CARPOS	D	569,897,978
145	CARRIAGE	D	457,460,463,503,506,889,977,989
928	CARRIAGE3	P	888
939	CARRIAGE4	P	936
954	CARRIAGE5	P	952
958	CARRIAGE6	P	956
196	CARSETL	A	975
197	CARSPD1	A	623,1293
198	CARSPD2	A	1294
199	CARSPD3	A	1295
200	CARSPD4	A	1296
201	CARSPD5	A	1297
202	CARSPD6	A	1298
203	CARSPD7	A	1299
204	CARSPD8	A	1300
1292	CARSPEEDS	P	959
146	CARSTAT	D	886,907,911,979,981,990
147	CARSTAT_BUSY	A	887,905,910
148	CARSTAT_LEFT	A	906,910,933
149	CARSTAT_SETTLE	A	931,980
150	CARTICKS	D	923,929,962,966,967,976
152	CARTOGO	D	153,920,941,950
621	CAR_DELAY	P	551,561
625	CAR_DELAY2	P	628
988	CAR_DONE	P	898,932
991	CAR_END	P	924,930,968,982
909	CAR_LEFT	P	904
974	CAR_SETTLE	P	942
918	CAR_SETVARS	P	908
243	CLEAR_MEM	P	247
160	CLOCK	D	161,509,513,740,743
1023	CONTINUE	P	1006
	COPY_BUFFER	P	325
173	CR	A	376,1248,1252,1256,1260,1264,1266
793	CW	P	788
56	DDR1	A	255
57	DDR2	A	264
60	DDR3	A	

LINE#	SYMBOL	TYPE	REFERENCES
61	DDR4	A	260
739	DEC_CLOCK	P	
744	DEC_CLOCK_END	P	741
178	DEL	A	
786	DIR	P	775,777,780
1087	DONE	P	1038,1059
219	E1	A	1348,1349,1349,1349,1351,1355,1356,1359,1359
218	E2	A	1348,1348,1349,1349,1349,1349,1350,1350,1350,1351,1351,1351,1351,1353,1355,1355,1355,1355,1356,1356,1356 1357,1357,1358,1358,1358,1359,1359,1359
217	E3	A	1348,1348,1348,1348,1348,1349,1350,1350,1350,1350,1350,1351,1351,1351,1352,1352,1353,1353,1354,1354,1355 1356,1356,1357,1357,1357,1357,1357,1358,1358,1358,1359,1359
216	E4	A	1352,1352,1352,1352,1352,1352,1353,1353,1353,1353,1353,1354,1354,1354,1354,1354,1354,1355,1355,1356,1356 1357,1358,1358,1359
1091	END	P	1008,1018,1025,1051,1068,1082
1347	ENERGIES	P	468
176	ESC	A	382
448	ESCAPE	P	383
1003	FIREHAMMER	P	
127	FIRE_PROOF	D	1109,1129,1131,1140,1147,1149
461	FIRE_WAIT1	P	466
475	FIRE_WAIT2	P	477
393	FORWARD	P	389
97	FUDGE	A	845,963
420	GOLEFT	P	408
409	GORIGHT	P	419
529	GOT_CHAR	P	
819	GO_CW	P	816
937	GO_LEFT	P	934
441	HALF_LINEFEED	P	387,444
137	HAMMER	D	473,476,1007,1057
220	HAMREL	A	1049
138	HAMSTAT	D	1004,1014,1017,1045,1062,1075,1088,1090
139	HAMSTAT_HFIRE	A	1005,1015,1031,1044
140	HAMSTAT_HREL	A	1005,1033,1043,1061
143	HAMSTAT_UDD	A	1016,1058,1089
141	HAMSTAT_RFIRE	A	1005,1035,1060,1074
142	HAMSTAT_RREL	A	1005,1037,1073
144	HAMTICKS	D	1010,1024,1050,1067,1081
1009	HFIRE	P	
158	HOME_IMAGE	D	301,534
181	HOME_SPOKE	A	661
294	HOME_THINGS	P	284
1042	HREL	P	1032
75	ICR	A	
157	IMAGE	D	252,300,530,535,579,588,591,601,610,613,672,681,684,694,703,706
483	INC	P	391,397,403,412,414,423,425,433,446,454
1211	INF_LOOP	P	1212
728	INT_LOOP	P	1379
390	INVALID	P	370
924	JCAR_END	P	890
494	JMAIN	P	504,512,537
	JNO_CHAR	P	333
806	JWH_END	P	755
171	LF	A	378,1246,1248,1250,1252,1254,1256,1258,1260,1262,1264,1266
435	LINEFEED	P	379,438
316	MAIN	P	494
330	MAKE_CHARS	P	

LINE#	SYMBOL	TYPE	REFERENCES
154	MODE	D	275,331,394,396,400,402,406,417,428,430
156	MODE_REVERSE	A	395,401,407,418,429
155	MODE_TEST	A	274,332
95	MOTORS	A	253,517,531,536,592,614,685,707
191	MSEC11	A	222
192	MSEC15	A	716
231	M_LEN	E	
231	M_SIG	E	324,511,522
778	NEG	P	773
231	NET_IN_BUFF	E	
286	NOT_SELF_TEST	P	271
502	NO_CHAR	P	
1367	NULL	P	
74	OCR	A	306,732
76	P3CSR	A	
159	PHANTOM	D	298,410,413,421,424,432,452,459,505
120	PLATEN	D	302,436,439,442,445,464,1103,1184
1100	PLATPOS	P	
121	PLATSTAT	D	1101,1107,1138,1165,1177,1183
122	PLATSTAT_RUN1	A	1106,1113
123	PLATSTAT_RUN2	A	1115,1137
125	PLATSTAT_SETTLE	A	1119,1176
124	PLATSTAT_STOP	A	1117,1164
126	PLATTICKS	D	1163,1169,1175,1181,1222,1223
180	PLWIDTH	A	411,453
226	PL_BRAKE_COUNT	A	1162
1112	PL_CONTINUE	P	1102
1186	PL_END	P	1136,1142,1154,1166,1170,1178,1182
1141	PL_END_ISLE	P	1104,1110,1121
1128	PL_RUN1	P	1114
1146	PL_RUN2	P	1116
1227	PL_RUN_OFF	P	
1232	PL_RUN_ON	P	1226
1180	PL_SETTLE	P	1120
227	PL_SETTLE_COUNT	A	1174
225	PL_SPEED	A	1225
1168	PL_STOP	P	1118
228	PL_TIMEOUT	A	1108,1139
58	PORT1	A	95
59	PORT2	A	
62	PORT3	A	97
63	PORT4	A	99
885	POSCAR	P	
750	POSWHEEL	P	
451	PRINTABLE	P	373
456	PRINTABLE1	P	458
367	PRINT_ONE	P	322
231	PR_MAC	E	1376
1221	PULSE_MOTOR	P	1133,1151
93	RAMCR	A	
91	RDATA	A	
241	RESET	P	1381
427	RETURN	P	377
399	REVERSE	P	385
1056	RFIRE	P	1034
222	RIBFIRE	A	1066
223	RIBREL	A	1079

LINE#	SYMBOL	TYPE	REFERENCES
77	RMCR	A	278,288,1205
79	RMCR_CC0	A	277,287,1204
78	RMCR_CC1	A	
81	RMCR_SS0	A	
80	RMCR_SS1	A	
671	ROT_CCW	P	640,653,817
693	ROT_CW	P	820
1072	RREL	P	1036
16	SHANNON	A	114,162,230,268,317,323,337,487,496,1239,1374
175	SI	A	388
174	SO	A	384,1246,1250,1254,1258,1262
374	SOME_2	P	372
177	SP	A	371,449,470,759,863
405	SPACE	P	375,478
1327	SPOKES	P	759,863
112	STACK	D	249
111	STACKSPACE	D	
578	STEP_LEFT	P	560,938
600	STEP_RIGHT	P	550,935
1244	STRING	P	335,338
1267	STRING_END	P	338
64	TCSR	A	282,292,308,729,733,1193,1203
68	TCSR_EICI	A	
69	TCSR_EOCI	A	307,1192
70	TCSR_ETOI	A	
65	TCSR_ICF	A	
71	TCSR_IEDG	A	
66	TCSR_OCF	A	
72	TCSR_OLVL	A	
67	TCSR_TOF	A	
92	TDATA	A	
190	TICK	A	305,623,716,731
510	TIMEOUT	P	514
73	TIMER	A	304,622,626,715,719,730
82	TRCSR	A	280,290,1207
84	TRCSR_ORFE	A	
83	TRCSR_RDRF	A	
87	TRCSR_RE	A	279,289,1206
86	TRCSR_RIE	A	289
85	TRCSR_TDRE	A	
89	TRCSR_TE	A	279,289,1206
88	TRCSR_TIE	A	
90	TRCSR_WU	A	
172	VT	A	386
161	WANT_TIMER	D	624,627,717,720
1200	WE_DIED	P	132,1150
128	WHEEL	D	455,462,754,862,877
652	WHEEL2	P	657
810	WHEEL3	P	753
821	WHEEL4	P	818
836	WHEEL5	P	834
840	WHEEL6	P	838
134	WHEELGONE	D	803,822,833,835
636	WHEELHOME	P	299,533,644
136	WHEELPOS	D	662,763,866
1308	WHEELSPEEDS	P	841
133	WHEELTICKS	D	805,811,844,851,852,861

LINE#	SYMBOL	TYPE	REFERENCES
135	WHEELTOGO	D	136,802,823,832
772	WHEELXX	P	764
714	WHEEL_DELAY	P	641,654
718	WHEEL_DELAY2	P	721
206	WHSETL	A	860
207	WHSPD1	A	1309
208	WHSPD2	A	1310
209	WHSPD3	A	1311
210	WHSPD4	A	1312
211	WHSPD5	A	1313
212	WHSPD6	A	1314
213	WHSPD7	A	1315
214	WHSPD8	A	1316
129	WHSTAT	D	751,791,795,867,869,878
130	WHSTAT_BUSY	A	752,789,794
131	WHSTAT_CW	A	790,794,815
132	WHSTAT_SETTLE	A	813,868
876	WH_DONE	P	765,814
879	WH_END	P	806,812,853,870
859	WH_SETTLE	P	824
801	WH_SETVARS	P	792

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1 ^6801^
                      2 ;
                      3 ; external routines referenced
                      4 ;
                      5                 EXT     MPR_TR_TRANS
                      6                 EXT     MPR_TR_TCU
                      7
                      8 ;
                      9 ; global data areas defined
                     10 ;
                     11                 GLB     CURRENT_STATE
                     12                 GLB     NET_IN_BUFF
                     13 ;
                     14 ; external data areas used
                     15 ;
                     16                 EXT     M_SIG,M_LEN
                     17
           (0002)    18 NODE_ADDR       EQU     2                       ; PRINTER ADDR = 2
                     19
                     20 ;
                     21 ; message scenarios equates
                     22 ;
           (0002)    23 MN_RESET        EQU     000H*16+NODE_ADDR       ;command.control (reset)
           (0012)    24 MN_STATUS       EQU     001H*16+NODE_ADDR       ;command.control (status)
           (0022)    25 MN_ACK          EQU     002H*16+NODE_ADDR       ;command.control (ack)
           (0032)    26 MN_CLR          EQU     003H*16+NODE_ADDR       ;command.control (clr)
           (0042)    27 MN_RECEIVE      EQU     004H*16+NODE_ADDR       ;command.control (receive)
           (0052)    28 MN_CANCEL       EQU     005H*16+NODE_ADDR       ;command.control (cancel)
           (0062)    29 MN_SEND         EQU     006H*16+NODE_ADDR       ;command.data (send)
           (0072)    30 MN_NACK         EQU     007H*16+NODE_ADDR       ;command.control (nack)
           (00D2)    31 MN_READY        EQU     00DH*16+NODE_ADDR       ;command.control (ready)
                     32
           (0082)    33 NM_STATUS       EQU     008H*16+NODE_ADDR       ;response.control (status)
           (0092)    34 NM_ACK          EQU     009H*16+NODE_ADDR       ;response.control (ack)
           (00A2)    35 NM_CANCEL       EQU     00AH*16+NODE_ADDR       ;response.control (cancel)
           (00B2)    36 NM_SEND         EQU     00BH*16+NODE_ADDR       ;response.data (send)
           (00C2)    37 NM_NACK         EQU     00CH*16+NODE_ADDR       ;response.control (nack)
                     38
                     39 ;
                     40 ; local equates
                     41 ;
           (0001)    42 DATA_AVAIL      EQU     001H
                     43
           (0000)    44 RDRF_INT        EQU     0
           (0001)    45 TDRE_INT        EQU     1
           (0002)    46 IGN_RDRF_INT    EQU     2
                     47
           (0000)    48 CNTRL           EQU     0
           (0001)    49 LENGTHIN_HI     EQU     1
           (0002)    50 LENGTHIN_LO     EQU     2
           (0003)    51 DATAIN          EQU     3
           (0004)    52 CHKSUM_IN       EQU     4
           (0005)    53 STATOUT         EQU     5
           (0006)    54 CHKSUM_OUT      EQU     6
           (0007)    55 WAIT_ACK_NACK   EQU     7
                     56
                     57                 PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      58
                      59                    GLB     PR_MAC
    0000              60 PR_MAC:
    0000 9605         61                    LDAA    INT_FLAG,D              ; WHAT KIND OF INT ARE WE
                      62                                                    ;  EXPECTING ?
    0002 270A         63                    BEQ     EXP_RDRF                ; RDRF ...
    0004 4A           64                    DECA                            ;
    0005 2732         65                    BEQ     EXP_TDRE                ; TDRE ...
                      66
                      67 ;**** HAD AN RDRF THAT WE WANT TO IGNORE ****
                      68
    0007 8600         69                    LDAA    #RDRF_INT               ; NOW WAITING FOR RDRF INT
    0009 9705         70                    STAA    INT_FLAG,D              ;
                      71
    000B 7E0143       72                    JMP     END_PR_MAC              ; DONE
                      73
    000E              74 EXP_RDRF:
                      75 ***************JSR     MPR_TR_REC              ;1   net_byte_in = MPR_TR_REC
                      76
                      77 ;
                      78 ; local equates
                      79 ;
           (000F)     80 ADDR_MASK       EQU     00FH
           (00F0)     81 CMND_MASK       EQU     0F0H
                      82
           (0040)     83 ORFE            EQU     01000000B
                      84
    000E 0D           85                    SEC                             ;1 TOKEN = BYTE FOR THIS NODE
    000F D611         86                    LDAB    011H,D
    0011 9612         87                    LDAA    012H,D                  ;1 NET_BYTE_IN = SCI_RX
                      88                                                    ;1 IF D1_MODE_WORD = CONTROL
    0013 C440         89                    ANDB    #ORFE
    0015 2707         90                    BEQ     NO_ORFE
                      91
                      92                    GLB     BREAK_ORFE
                      93
    0017              94 BREAK_ORFE:
    0017 8600         95                    LDAA    #CNTRL
    0019 9700         96                    STAA    CURRENT_STATE,D
                      97
    001B 7E0030       98                    JMP     SET_WAKEUP
                      99
    001E             100 NO_ORFE:
    001E D600        101                    LDAB    CURRENT_STATE,D         ;    get D1_MODE_WORD
    0020 260C        102                    BNE     ENDIF_CNTRL
    0022 16          103                    TAB                             ;2   SAVE_NBI = NET_BYTE_IN                          /* NOT REC
    0023 840F        104                    ANDA    #ADDR_MASK              ;2   ADDRESS = NET_BYTE_IN .AND. ADDR_MASK           /* LOWER H
    0025 8102        105                    CMPA    #NODE_ADDR              ;2   IF ADDRESS = NODE_ADDR
    0027 2604        106                    BNE     ELSE_NOTADDR
    0029 17          107                    TBA
                     108                                                    ;3     NET_BYTE_IN = SAVE_NBI .AND. CMND_MASK        /* UPPER H
    002A 0D          109                    SEC
    002B 2001        110                    BRA     ENDIF_ADDR
    002D             111 ELSE_NOTADDR:                                      ;2   ELSE
    002D 0C          112                    CLC                             ;3     TOKEN = BYTE NOT FOR THIS NODE
    002E             113 ENDIF_ADDR:                                        ;2   ENDIF
    002E             114 ENDIF_CNTRL:                                       ;1 ENDIF
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    002E 250B          115                 BCS     BYTE_FOR_US             ;1   if net_byte_in = for our address
                       116
    0030               117 SET_WAKEUP:
    0030 D611          118                 LDAB    011H,D
    0032 CA01          119                 ORAB    #00000001B              ; SET WAKE-UP FLAG
    0034 D711          120                 STAB    011H,D
                       121
    0036 7E0143        122                 JMP     END_PR_MAC
                       123
    0039               124 EXP_TDRE:
    0039 9611          125                 LDAA    011H,D                  ; ALLOW WRITE TO TRD
                       126                                                 ;  TO CLEAR TDRE
                       127
    003B               128 BYTE_FOR_US:
                       129                                                 ;2     case CURRENT_STATE of
    003B D600          130                 LDAB    CURRENT_STATE,D          ;        check for control case
    003D 58            131                 LSLB                            ; TABLE LOOK-UP
    003E CE0046        132                 LDX     #STATE_TABLE            ;  INTO STATE_TABLE
    0041 3A            133                 ABX                             ;
    0042 EE00          134                 LDX     0,X                     ; WHERE WE SHOULD GO
    0044 6E00          135                 JMP     0,X                     ;
                       136
    0046               137 STATE_TABLE:
    0046 0056          138                 FDB     CONTROL                 ; NET BYTE IN IS COMMAND
    0048 00B4          139                 FDB     LENGTH_IN_HI            ; EXPECTING HI BYTE OF LENGTH
    004A 00BF          140                 FDB     LENGTH_IN_LO            ;              LO
    004C 00CF          141                 FDB     DATA_IN                 ; EXPECTING DATA BYTES IN
    004E 00EB          142                 FDB     CHK_SUM_IN              ; EXPECTING CHECKSUM IN
    0050 0107          143                 FDB     STAT_OUT                ; SENDING STATUS OUT
    0052 0125          144                 FDB     CHK_SUM_OUT             ; SENDING CHECKSUM OUT
    0054 013B          145                 FDB     GOT_ACK_NACK            ; WAITING FOR ACK/NACK
                       146
    0056               147 CONTROL:                                        ;2       control    : begin
    0056 8102          148                 CMPA    #MN_RESET
    0058 2605          149                 BNE     NOT_RESET
                       150
    005A FEFFFE        151                 LDX     0FFFEH,E
    005D 6E00          152                 JMP     0,X
                       153
    005F               154 NOT_RESET:
    005F 81D2          155                 CMPA    #MN_READY
    0061 2715          156                 BEQ     CHK_CTS
    0063 8162          157                 CMPA    #MN_SEND
    0065 2726          158                 BEQ     REC_DATA
    0067 8112          159                 CMPA    #MN_STATUS              ; REQUEST FOR STATUS
    0069 2729          160                 BEQ     SEND_STATUS
                       161
    006B 86C2          162                 LDAA    #NM_NACK                ; BAD COMMAND TO ME
    006D BD0000        163                 JSR     MPR_TR_TRANS            ;
    0070 2503          164                 BCS     ERR1
    0072 BD0000        165                 JSR     MPR_TR_TCU              ;
                       166
    0075               167 ERR1:
    0075 7E0143        168                 JMP     END_PR_MAC              ; DONE
                       169
    0078               170 CHK_CTS:                                        ;1     chk_cts     : begin
    0078 9600          171                 LDAA    M_SIG,D                 ;2                    if NIM.M_SIG = 0
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    007A 2604          172                 BNE     ELSE_NCTS
    007C 8692          173                 LDAA    #NM_ACK                 ;3                      send ACK
    007E 2002          174                 BRA     ENDIF_CTS
    0080               175 ELSE_NCTS:                                      ;2                  else
    0080 86C2          176                 LDAA    #NM_NACK                  ;3                    send NACK
    0082               177 ENDIF_CTS:                                      ;2                  endif
    0082 BD0000        178                 JSR     MPR_TR_TRANS
    0085 2503          179                 BCS     ERR2
    0087 BD0000        180                 JSR     MPR_TR_TCU
                       181
    008A               182 ERR2:
    008A 7E0143        183                 JMP     END_PR_MAC
                       184
    008D               185 REC_DATA:                                       ;1      rec_data    : begin
    008D 8601          186                 LDAA    #LENGTHIN_HI            ;2                      CURRENT_STATE = length_in
    008F 9700          187                 STAA    CURRENT_STATE,D
    0091 7E0143        188                 JMP     END_PR_MAC                ;1                     end
                       189
    0094               190 SEND_STATUS:
                       191
                       192 ;               DISABLE RDRF INT
                       193 ;               ENABLE TDRE INT
                       194 ;               WAKEUP = 1
                       195
    0094 860F          196                 LDAA    #00001111B
    0096 9711          197                 STAA    011H,D
                       198
    0098 CE0004        199                 LDX     #STAT_PKT_SIZE-1        ; # BYTES IN STATUS PKT
    009B DF01          200                 STX     LOC_LEN,D               ; (WE ALREADY SENT 1)
                       201
    009D 8605          202                 LDAA    #STATOUT                ; WE WILL BE SENDING DATA
    009F 9700          203                 STAA    CURRENT_STATE,D         ;
                       204
    00A1 8601          205                 LDAA    #TDRE_INT               ; TDRE IS SOURCE OF INTS NOW
    00A3 9705          206                 STAA    INT_FLAG,D              ;
                       207
    00A5 8601          208                 LDAA    #1                      ; USE OFFSET BYTE 1 NEXT TIME
    00A7 9704          209                 STAA    DATA_INDEX,D            ;  WE TRANSMIT
                       210
    00A9 7F0003        211                 CLR     CHK_SUM,D               ;
                       212
    00AC B60144        213                 LDAA    STAT_PKT+0              ; FIRST BYTE
    00AF 9713          214                 STAA    013H,D                  ; SEND BYTE
                       215
    00B1 7E0143        216                 JMP     END_PR_MAC              ; BYE BYE ...
                       217
    00B4               218 LENGTH_IN_HI:                                   ;2          length_in : begin
    00B4 9701          219                 STAA    LOC_LEN,D               ;3                      LOC_LEN = net_byte_in
    00B6 9700          220                 STAA    M_LEN,D
                       221                                                 ;3                      CURRENT_STATE = data_in
    00B8 C602          222                 LDAB    #LENGTHIN_LO            ;                         get the data_in constant
    00BA D700          223                 STAB    CURRENT_STATE,D         ;                         store the constant
                       224
    00BC 7E0143        225                 JMP     END_PR_MAC                ;2                     end
                       226
    00BF               227 LENGTH_IN_LO:
    00BF 9702          228                 STAA    LOC_LEN+1,D
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    00C1 9701          229                STAA    M_LEN+1,D
                       230
    00C3 C603          231                LDAB    #DATAIN
    00C5 D700          232                STAB    CURRENT_STATE,D
                       233
    00C7 7F0004        234                CLR     DATA_INDEX,D
    00CA 7F0003        235                CLR     CHK_SUM,D
                       236
    00CD 2074          237                BRA     END_PR_MAC
                       238
    00CF               239 DATA_IN:                                    ;2      data_in    : begin
                       240                                             ;3                   ram_memory [NIM.M_DATA] = net_byte_in
    00CF CE0006        241                LDX     #NET_IN_BUFF         ;                     get the base pointer into ram_memory
    00D2 D604          242                LDAB    DATA_INDEX,D      ;                    get the index from the base
    00D4 3A            243                ABX                          ;                     form the effective address
    00D5 A700          244                STAA    0,X                  ;                     put net_byte_in into ram_memory
                       245
    00D7 9803          246                EORA    CHK_SUM,D           ;3                  csnum = csnum XOR net_byte_in
    00D9 9703          247                STAA    CHK_SUM,D
                       248
    00DB 7C0004        249                INC     DATA_INDEX,D     ;3                   NIM.M_DATA = NIM.M_DATA + 1
                       250
    00DE DE01          251                LDX     LOC_LEN,D
    00E0 09            252                DEX
    00E1 DF01          253                STX     LOC_LEN,D
                       254
    00E3 2604          255                BNE     MORE_DATA_IN         ;3                   if LOC_LEN = 0
                       256                                             ;4                    CURRENT_STATE = chk_sum
    00E5 8604          257                LDAA    #CHKSUM_IN           ;                       get the chk_sum constant
    00E7 9700          258                STAA    CURRENT_STATE,D      ;                       store the constant
                       259
    00E9               260 MORE_DATA_IN:                               ;3                   endif
    00E9 2058          261                BRA     END_PR_MAC              ;2                end
                       262
    00EB               263 CHK_SUM_IN:                                    ;2      chk_sum    : begin
                       264                                             ;3                   if csnum = net_byte_in
    00EB 9103          265                CMPA    CHK_SUM,D            ;                     compare with net_byte_in
    00ED 2608          266                BNE     ELSE_NEQU
                       267
    00EF 8601          268                LDAA    #DATA_AVAIL
    00F1 9700          269                STAA    M_SIG,D
                       270
    00F3 8692          271                LDAA    #NM_ACK              ;4                     send ACK
    00F5 2002          272                BRA     ENDIF_CSNUM
                       273
    00F7               274 ELSE_NEQU:                                  ;3                   else
    00F7 86C2          275                LDAA    #NM_NACK                ;4                      send NACK
                       276
    00F9               277 ENDIF_CSNUM:                                ;3                   endif
    00F9 BD0000        278                JSR     MPR_TR_TRANS
    00FC 2507          279                BCS     ERR3
    00FE BD0000        280                JSR     MPR_TR_TCU
                       281
                       282                                             ;3                   CURRENT_STATE = idle
    0101 8600          283                LDAA    #CNTRL               ;                     get the idle constant
    0103 9700          284                STAA    CURRENT_STATE,D      ;                     store the constant
                       285                                             ;3                   CURRENT_STATE = control
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     286
    0105             287 ERR3:
    0105 203C        288                 BRA     END_PR_MAC
                     289
    0107             290 STAT_OUT:
    0107 CE0144      291                 LDX     #STAT_PKT           ; SOURCE OF STATUS OUT
    010A D604        292                 LDAB    DATA_INDEX,D        ; WHICH BYTE THIS TIME
    010C 3A          293                 ABX                         ; GET INDEXED ADDR
    010D A600        294                 LDAA    0,X                 ; A = DATA BYTE TO GO
    010F 9713        295                 STAA    013H,D              ; SHIP IT
                     296
    0111 9803        297                 EORA    CHK_SUM,D           ; ACCUMULATE CHECKSUM
    0113 9703        298                 STAA    CHK_SUM,D           ;
                     299
    0115 7C0004      300                 INC     DATA_INDEX,D        ; NEXT BYTE NEXT TIME
                     301
    0118 DE01        302                 LDX     LOC_LEN,D           ; DEC COUNT OF BYTES SENT
    011A 09          303                 DEX                         ;
    011B DF01        304                 STX     LOC_LEN,D           ; SAVE NEW BYTE COUNT
                     305
    011D 2604        306                 BNE     MORE_STAT_OUT       ; MORE TO DO
                     307
    011F 8606        308                 LDAA    #CHKSUM_OUT         ; SEND CHECKSUM NEXT TIME
    0121 9700        309                 STAA    CURRENT_STATE,D     ;
                     310
    0123             311 MORE_STAT_OUT:
    0123 201E        312                 BRA     END_PR_MAC
                     313
    0125             314 CHK_SUM_OUT:
    0125 9603        315                 LDAA    CHK_SUM,D           ;
    0127 9713        316                 STAA    013H,D              ; OUTPUT CHECKSUM
                     317
    0129 9611        318                 LDAA    011H,D              ; RESET RDRF (HOPEFULLY...)
    012B 9612        319                 LDAA    012H,D
                     320
                     321 ;               DISABLE TDRE INT
                     322 ;               ENABLE RDRF INT
                     323 ;               WAKEUP = 1
                     324
    012D 861B        325                 LDAA    #00011011B          ;
    012F 9711        326                 STAA    011H,D              ;
                     327
    0131 8600        328                 LDAA    #RDRF_INT           ; NEXT INT SHOULD BE RDRF
    0133 9705        329                 STAA    INT_FLAG,D          ;
                     330
    0135 8607        331                 LDAA    #WAIT_ACK_NACK      ; NEXT BYTE IN SHOULD BE
    0137 9700        332                 STAA    CURRENT_STATE,D     ;  ACK/NACK ON WHAT WE
                     333                                             ;  SENT
                     334
    0139 2008        335                 BRA     END_PR_MAC          ; DONE
                     336
    013B             337 GOT_ACK_NACK:
    013B C600        338                 LDAB    #CNTRL              ; NEXT BYTE IN IS NEW
    013D D700        339                 STAB    CURRENT_STATE,D     ;  SCENARIO
                     340
    013F 861B        341                 LDAA    #00011011B
    0141 9711        342                 STAA    011H
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     343
                     344 ;****************BRA     END_PR_MAC              ;
                     345
     0143            346 END_PR_MAC:                                    ;2      endcase
     0143 3B         347                 RTI
                     348
     0144            349 STAT_PKT:
     0144 82         350                 FCB     82H
     0145 10         351                 FCB     16
     0146 00         352                 FCB     0
     0147 00         353                 FCB     0
     0148 00         354                 FCB     0
                     355
              (0005) 356 STAT_PKT_SIZE   EQU     $-STAT_PKT
                     357
                     358 ;
                     359 ; local data areas defined
                     360 ;
                     361                 DATA
                     362
                     363                 GLB     CURRENT_STATE
                     364
     0000            365 CURRENT_STATE  RMB     1
     0001            366 LOC_LEN        RMB     2                       ;local counter used for data input
     0003            367 CHK_SUM        RMB     1                       ;interim variable used to calculate the check sum
     0004            368 DATA_INDEX     RMB     1                       ; INDEX INTO NET_IN_BUFF
     0005            369 INT_FLAG       RMB     1                       ; WHERE INTS COME FROM
                     370
                     371                 GLB     NET_IN_BUFF
                     372
     0006            373 NET_IN_BUFF    RMB     16                      ;buffer used to store net_byte_in values

Errors=    0
```

LINE#	SYMBOL	TYPE	REFERENCES
80	ADDR_MASK	A	104
94	BREAK_ORFE	P	92
128	BYTE_FOR_US	P	15
52	CHKSUM_IN	A	257
54	CHKSUM_OUT	A	308
170	CHK_CTS	P	156
367	CHK_SUM	D	211,235,246,247,265,297,298,315
263	CHK_SUM_IN	P	142
314	CHK_SUM_OUT	P	144
81	CMND_MASK	A	
48	CNTRL	A	95,283,338
147	CONTROL	P	138
365	CURRENT_STATE	D	11,96,101,130,187,203,223,232,258,284,309,332,339,363
51	DATAIN	A	231
42	DATA_AVAIL	A	268
239	DATA_IN	P	141
368	DATA_INDEX	D	209,234,242,249,292,300
175	ELSE_NCTS	P	172
274	ELSE_NEQU	P	266
111	ELSE_NOTADDR	P	106
113	ENDIF_ADDR	P	110
114	ENDIF_CNTRL	P	102
277	ENDIF_CSNUM	P	272
177	ENDIF_CTS	P	174
346	END_PR_MAC	P	72,122,168,183,188,216,225,237,261,288,312,335
167	ERR1	P	164
182	ERR2	P	179
287	ERR3	P	279
74	EXP_RDRF	P	63
124	EXP_TDRE	P	65
337	GOT_ACK_NACK	P	145
46	IGN_RDRF_INT	A	
369	INT_FLAG	D	61,70,206,329
49	LENGTHIN_HI	A	186
50	LENGTHIN_LO	A	222
218	LENGTH_IN_HI	P	139
227	LENGTH_IN_LO	P	140
366	LOC_LEN	D	200,219,228,251,253,302,304
25	MN_ACK	A	
28	MN_CANCEL	A	
26	MN_CLR	A	
30	MN_NACK	A	
31	MN_READY	A	155
27	MN_RECEIVE	A	
23	MN_RESET	A	148
29	MN_SEND	A	157
24	MN_STATUS	A	159
260	MORE_DATA_IN	P	255
311	MORE_STAT_OUT	P	306
6	MPR_TR_TCU	E	165,180,280
5	MPR_TR_TRANS	E	163,178,278
16	M_LEN	E	220,229
16	M_SIG	E	171,269
373	NET_IN_BUFF	D	12,241,371
34	NM_ACK	A	173,271
35	NM_CANCEL	A	
37	NM_NACK	A	162,176,275

LINE#	SYMBOL	TYPE	REFERENCES
36	NM_SEND	A	
33	NM_STATUS	A	
18	NODE_ADDR	A	23,24,25,26,27,28,29,30,31,33,34,35,36,37,105
154	NOT_RESET	P	149
100	NO_ORFE	P	90
83	ORFE	A	89
60	PR_MAC	P	59
44	RDRF_INT	A	69,328
185	REC_DATA	P	158
190	SEND_STATUS	P	160
117	SET_WAKEUP	P	98
137	STATE_TABLE	P	132
53	STATOUT	A	202
290	STAT_OUT	P	143
349	STAT_PKT	P	213,291,356
356	STAT_PKT_SIZE	A	199
45	TDRE_INT	A	205
55	WAIT_ACK_NACK	A	331

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1 ^6801^
                      3  NAME ^Rev 03 - RPD^
                      4
                      5 De_MPR_TR_TRANS MACRO              ;Header Rev. 4
                      6                 .GOTO Ede_MPR_TR_TRANS
                      7
                      8  Project:      NET, 83-101
                      9
                     10  ************************************************
                     11  *                                              *
                     12  *     MPR_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_MPR_TR_TRANS  MEND
```

LOCATION OBJECT CODE LINE SOURCE LINE

```
25 ************************************************************************
26 *                                                                      *
27 *  MODULE NAME:                                                        *
28 *                                                                      *
29 *    MPR_TR_TRANS                                                      *
30 *                                                                      *
31 *  INPUTS:                                                             *
32 *                                                                      *
33 *    NET_BYTE_OUT (REG_A)                                              *
34 *                                                                      *
35 *  FUNCTION(S):                                                        *
36 *                                                                      *
37 *     1. TO SEND A BYTE OUT OVER THE NETWORK.                          *
38 *                                                                      *
39 *  OUTPUTS:                                                            *
40 *                                                                      *
41 *     NET_BYTE_OUT (LOCATION 13)                                       *
42 *                                                                      *
43 *  CALLS:                                                              *
44 *                                                                      *
45 *    NONE.                                                             *
46 *                                                                      *
47 *  CALLED BY:                                                          *
48 *                                                                      *
49 *    MPR_ACM_SEQ                                                       *
50 *    MPR_NIM_READ                                                      *
51 *                                                                      *
52 *  NOTES:                                                              *
53 *                                                                      *
54 *    NONE.                                                             *
55 *                                                                      *
56 ************************************************************************
57
58 ************************************************************************
59 *                                                                      *
60 *  PSEUDO CODE:                                                        *
61 *                                                                      *
62 *      MPR_TR_TRANS:                                                   *
63 *                                                                      *
64 *        REPEAT_UNTIL_SET:                                             *
65 *                                                                      *
66 *             IF MEM(11).5=0 THEN GOTO REPEAT_UNTIL_SET;               *
67 *             ENDIF;                                                   *
68 *                                                                      *
69 *             MEM(13)=REG_A;                                           *
70 *                                                                      *
71 *       RETURN;                                                        *
72 *                                                                      *
73 ************************************************************************
```

```
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     MPR_TR_TRANS
     0000             85 MPR_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	MPR_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

                      1 ^6801^
                      3  NAME ^Rev 01 - RPD^
                      4
                      5 De_MPR_TR_TCU MACRO              ;Header Rev. 4
                      6                 .GOTO Ede_MPR_TR_TCU
                      7
                      8  Project:      NET, 83-101
                      9
                     10  ************************************************************
                     11  *                                                          *
                     12  *     MPR_TR_TCU                          RPD              *
                     13  *                                                          *
                     14  ************************************************************
                     15
                     16        Rev History
                     17        Rev.  Date         Name      Change
                     18         1    20jul1800p   RPD       created from MPR file
                     19         0    19jul1535p   RPD       Initial Pseudo code and code
                     20
                     21 Ede_MPR_TR_TCU  MEND
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    23 ********************************************************************
                    24 *                                                                  *
                    25 *  MODULE NAME:                                                    *
                    26 *                                                                  *
                    27 *    MPR_TR_TCU (transmit clean up)                                *
                    28 *                                                                  *
                    29 *  INPUTS:                                                         *
                    30 *                                                                  *
                    31 *    none                                                          *
                    32 *                                                                  *
                    33 *  FUNCTION(S):                                                    *
                    34 *                                                                  *
                    35 *     1. Clears the "receive data register full" flag of the       *
                    36 *        6801 SCI after a transmission sequence (1 or more         *
                    37 *        bytes). The flag is set as a result of sending a byte     *
                    38 *        out and receiving the same byte in on the common NET      *
                    39 *        line used for sending and receiving.                      *
                    40 *                                                                  *
                    41 *  OUTPUTS:                                                        *
                    42 *                                                                  *
                    43 *     SCI control/status register bit 7 = 0                        *
                    44 *                                                                  *
                    45 *  CALLS:                                                          *
                    46 *                                                                  *
                    47 *    none                                                          *
                    48 *                                                                  *
                    49 *  CALLED BY:                                                      *
                    50 *                                                                  *
                    51 *    MPR_ACM_R                                                     *
                    52 *    (all routines calling MPR_TR_TRANS)                           *
                    53 *                                                                  *
                    54 *  NOTES:                                                          *
                    55 *     1 - This sequence follows the procedure described in         *
                    56 *         hardware manuals for clearing the flag. Which is:        *
                    57 *           step 1) read the SCI control status register           *
                    58 *           step 2) read the SCI receive data register            *
                    59 *     2 - The MAC modules are responsible for calling this         *
                    60 *         module after doing a transmit function to avoid          *
                    61 *         reading itself when other data is expected.              *
                    62 *                                                                  *
                    63 ********************************************************************
```

```
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     MPR_TR_TCU ;
0000                 86 MPR_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	MPR_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

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1 ^6801^
                      3  NAME ^Rev 00 - DLS^
                      4
                      5 De_D_MAC MACRO              ;Header Rev. 4
                      6                 .GOTO Ede_D_MAC
                      7
                      8  Project:      NET, 83-101
                      9
                     10  ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
                     11  ▒▒                                                  ▒▒
                     12  ▒▒     D_MAC                        DLS             ▒▒
                     13  ▒▒                                                  ▒▒
                     14  ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
                     15
                     16        Rev History
                     17        Rev.  Date         Name      Change
                     18         0    13jul1815   DLS       Initial Pseudo code
                     19
                     20 Ede_D_MAC  MEND
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      22 ****************************************************************
                      23 *                                                              *
                      24 *  MODULE NAME:                                                *
                      25 *                                                              *
                      26 *    D_MAC                                                     *
                      27 *                                                              *
                      28 *  FUNCTION(S):                                                *
                      29 *                                                              *
                      30 *     1.  TO DECLARE THE DATA AREA "NIM_BLOCK."                *
                      31 *     2.  TO DECLARE THE D1_MODE_WORD.                         *
                      32 *                                                              *
                      33 *  NOTES:                                                      *
                      34 *                                                              *
                      35 * 1. NIM_BLOCK IS USED AS THE INTERFACE BETWEEN THE            *
                      36 *    MEDIUM ACCESS CONTROLLER AND THE RESIDENT APPLICATION     *
                      37 *    PROGRAM.                                                  *
                      38 *                                                              *
                      39 * 2. THE INSTALLER IS RESPONSIBLE FOR LOCATING THIS DATA       *
                      40 *    MODULE SO THAT THE LAST BYTE ENDS AT LOCATION 127 (DEC).  *
                      41 *                                                              *
                      42 ****************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      44                GLB     D_MAC
                      45                GLB     D1_MODE_WORD
                      46                GLB     NIM_BLOCK
                      47                GLB     A_LEN
                      48                GLB     A_DATA
                      49                GLB     A_SIG
                      50                GLB     CNFG_WORD
                      51                GLB     M_SIG
                      52                GLB     M_LEN
                      53                DATA
0000                  54 D_MAC:
                      55 ********************************************************************
                      56 *                                                                  *
                      57 *  DATA WORD:                                                      *
                      58 *                                                                  *
                      59 *    D1_MODE_WORD                                                  *
                      60 *                                                                  *
                      61 *   FUNCTION:                                                      *
                      62 *                                                                  *
                      63 *    CONTAINS THE STATE OF SEQUENCER PROCESSING                    *
                      64 *                                                                  *
                      65 ********************************************************************
            (0000)    66 D1_MODE_WORD    EQU             $
0000                  67                 RMB     1
```

LOCATION OBJECT CODE LINE SOURCE LINE

```
69                  INCLUDE I_NIM_PR
 + ************************************************************************
 + *                                                                      *
 + *                        HIERARCHY  CHART                              *
 + *                              OF                                      *
 + *                    MEDIUM  ACCESS  CONTROLLER                        *
 + *                             (MAC)                                    *
 + *                                                                      *
 + *                            (MAC)                                     *
 + *                              |                                       *
 + *                              |                                       *
 + *           +------------------+-------------------------+             *
 + *         (TR)                (NIM)                    (ACM)          *
 + *          |                   |                         |            *
 + *     +----+---+--------+       |                         +            *
 + *  (TRANS)  (REC)    (TCU)     |                        (SEQ)         *
 + *                              |                         |            *
 + *                 +------------+-----------+             |            *
 + *                 |                        |             |            *
 + *              (WRITE)                  (READ)           |            *
 + *                                                        |            *
 + *                                                        |            *
 + *                                              +---------+---------+  *
 + *                                              |                   |  *
 + *                                         (EVENT_PROC)          (RESP)*
 + *                                                                      *
 + ************************************************************************
```

LOCATION OBJECT CODE LINE SOURCE LINE

```
+ ************************************************************************
+ *                INTERFACE MODULE DESCRIPTION                          *
+ *                ----------------------------                          *
+ *                                                                      *
+ *  NAME:                                                               *
+ *                                                                      *
+ *   1_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:                             *
+ *                                                                      *
+ *           +-------------+                                            *
+ *           | M_SIG       |                                            *
+ *           +-------------+                                            *
+ *           | M_DATA  (HI)|                                            *
+ *           +-------------+                                            *
+ *           | M_DATA  (LO)|                                            *
+ *           +-------------+                                            *
+ *           | M_LEN       |                                            *
+ *           +-------------+                                            *
+ ************************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    + ************************************************************************
                    + *                                                                      *
                    + *  DATA ELEMENT DEFINITIONS:                                           *
                    + *                                                                      *
                    + *                                                                      *
                    + *     M_SIG:                                                           *
                    + *     -----                                                            *
                    + *        0- APP READY FOR MORE DATA.          (APP WRITES)             *
                    + *        1- APP IS NOT READY FOR MORE DATA.   (MAC WRITES)             *
                    + *                                                                      *
                    + *     M_DATA                                                           *
                    + *     ------                                                           *
                    + *                                                                      *
                    + *        2 BYTE POINTER TO DATA.                                       *
                    + *                                                                      *
                    + *     M_LEN:                                                           *
                    + *     -----                                                            *
                    + *                                                                      *
                    + *        COUNT OF BYTES IN MAC BUFFER.                                 *
                    + *                                                                      *
                    + *   NOTES:                                                             *
                    + *                                                                      *
                    + *        NONE.                                                         *
                    + *                                                                      *
                    + ************************************************************************
```

```
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 255 (decimal).       *
                      + *                                                          *
                      + ************************************************************
    0001              70 NIM_BLOCK:
    0001              71 A_SIG
    0001              72 A_DATA
    0001              73 A_LEN
    0001              74 CNFG_WORD
    0001              75 M_SIG          RMB     1
    0002              76 M_LEN          RMB     2

Errors=    0
```

LINE#	SYMBOL	TYPE	REFERENCES
72	A_DATA	D	48
73	A_LEN	D	47
71	A_SIG	D	49
74	CNFG_WORD	D	50
66	D1_MODE_WORD	D	45
54	D_MAC	D	44
76	M_LEN	D	52
75	M_SIG	D	51
70	NIM_BLOCK	D	46

FILE/PROG NAME	PROGRAM	DATA	COMMON	ABSOLUTE	DATE	TIME	COMMENTS
DAISY4:pADAMP	F800	0080		FFF0-FFF1 FFF4-FFF5 FFFE-FFFF	Mon,	7 Nov 1983, 15:47	Rev 16 GRW
PR_MAC:pADAMP	FDD8	00E6			Mon,	7 Nov 1983, 16:03	
next address	FF21	00FC					
MPR_TR_TR:pADAMP	FF21				Mon,	7 Nov 1983, 16:02	Rev 03 - RPD
MPR_TR_TC:pADAMP	FF39				Mon,	7 Nov 1983, 16:00	Rev 01 - RPD
next address	FF64						
D_MPR:pADAMP		00FC			Mon,	7 Nov 1983, 15:59	Rev 00 - DLS
next address		0100					

XFER address= 0000 Defined by DEFAULT
absolute & link_com file name=PRA:pADAMP
Total# of bytes loaded= 07EA

SYMBOL	R	VALUE	DEF BY	REFERENCES
A_DATA	D	00FD	D_MPR:pADAMP	
A_LEN	D	00FD	D_MPR:pADAMP	
A_SIG	D	00FD	D_MPR:pADAMP	
BREAK_ORFE	P	FDEF	PR_MAC:pADAMP	
CLEAN_UART_HW	P	FF57	MPR_TR_TC:pADAMP	MPR_TR_TR:pADAMP
CNFG_WORD	D	00FD	D_MPR:pADAMP	
CURRENT_STATE	D	00E6	PR_MAC:pADAMP	MPR_TR_TC:pADAMP
D1_MODE_WORD	D	00FC	D_MPR:pADAMP	
D_MAC	D	00FC	D_MPR:pADAMP	
MPR_TR_TCU	P	FF39	MPR_TR_TC:pADAMP	PR_MAC:pADAMP
MPR_TR_TRANS	P	FF21	MPR_TR_TR:pADAMP	PR_MAC:pADAMP
M_LEN	D	00FE	D_MPR:pADAMP	PR_MAC:pADAMP DAISY4:pADAMP
M_SIG	D	00FD	D_MPR:pADAMP	PR_MAC:pADAMP DAISY4:pADAMP
NET_IN_BUFF	D	00EC	PR_MAC:pADAMP	DAISY4:pADAMP
NIM_BLOCK	D	00FD	D_MPR:pADAMP	
PR_MAC	P	FDD8	PR_MAC:pADAMP	DAISY4:pADAMP