```
                46  ***     H17ROM - H17 ROM CODE LISTING.
                47  *
                48  *       JGL, OCT 77
                49  *
                50  *       COPYRIGHT (C) HEATH CO., 1977
```

```
                                53
                                54
030.000                         55          ORG     30000A
                                56
030.000  303 014 037           57          JMP     BOOT            BOOT CODE


                                59  **      MEMORY DIAGNOSTIC.
                                60  *
                                61
030.003  041 300 377           62          LXI     H,-64
030.006  071                   63          DAD     SP              (HL) = END
030.007  353                   64          XCHG                    (DE) = END+1
030.010  041 100 040           65          LXI     H,40100A        (HL) = START
030.013  166                   66          HLT                     PAUSE FOR ADJUSTMENT
                                67
                                68
                                69  *       (HL) = START
                                70  *       (DE) = END
                                71
                                72  *       ZERO TEST AREA
                                73
030.014  042 076 040           74          SHLD    40100A-2
030.017  066 000               75  MEM1    MVI     M,0
030.021  043                   76          INX     H
030.022  315 216 030           77          CALL    $CDEHL
030.025  302 017 030           78          JNE     MEM1
                                79
                                80  *       START TESTING MEMORY, INCREMENT EACH BYTE IN TURN, AND COMPARE
                                81  *       THAT RESULT TO THE EXPECTED VALUE
                                82
030.030  006 000               83          MVI     B,0             (B) = EXPECTED VALUE
030.032  052 076 040           84  MEM2    LHLD    40100A-2
030.035  004                   85          INR     B
                                86
030.036  064                   87  MEM3    INR     M
030.037  176                   88          MOV     A,M             (A) = VALUE
030.040  270                   89          CMP     B
030.041  312 046 030           90          JE      MEM4            IS OK
                                91
                                92  *       HAVE ERROR. (HL) = ADDRESS OF BYTE IN ERROR
                                93
030.044  166                   94          HLT
030.045  000                   95          NOP
                                96
030.046  043                   97  MEM4    INX     H
030.047  315 216 030           98          CALL    $CDEHL
030.052  302 036 030           99          JNE     MEM3            NOT AT END OF PASS
030.055  303 032 030          100          JMP     MEM2            AT END OF PASS
```

```
                        105X **      $COMP - COMPARE TWO CHARACTER STRINGS.
                        106X *
                        107X *       $COMP COMPARES TWO BYTE STRINGS.
                        108X *
                        109X *       ENTRY   (C) = COMPARE COUNT
                        110X *               (DE) = FWA OF STRING #1
                        111X *               (HL) = FWA OF STRING #2
                        112X *       EXIT    'Z' CLEAR, IS MIS-MATCH
                        113X *                (C) = LENGTH REMAINING
                        114X *                (DE) = ADDRESS OF MISMATCH IN STRING#1
                        115X *                (HL) = ADDRESS OF MISMATCH IN STRING #2
                        116X *               'C' SET, HAVE MATCH
                        117X *                (C) = 0
                        118X *                (DE) = (DE) + (0C)
                        119X *                (HL) = (HL) + (0C)
                        120X *       USES    A,F,C,D,E,H,L
                        121X
                        122X
030.060  032            123X $COMP   LDAX    D
030.061  276            124X         CMP     M                COMPARE
030.062  300            125X         RNE                      NO MATCH
030.063  023            126X         INX     D
030.064  043            127X         INX     H
030.065  015            128X         DCR     C
030.066  302 060 030    129X         JNZ     $COMP            TRY SOME MORE
030.071  311            130X         RET                      HAVE MATCH


                        133X **      $DADA - PERFORM (H,L) = (H,L) + (0,A)
                        134X *
                        135X *       ENTRY   (H,L) = BEFORE VALUE
                        136X *               (A) = BEFORE VALUE
                        137X *       EXIT    (H,L) = (H,L) + (0,A)
                        138X *               'C' SET IF OVERFLOW
                        139X *       USES    F,H,L
                        140X
                        141X
030.072  325            142X $DADA   PUSH    D
030.073  137            143X         MOV     E,A
030.074  026 000        144X         MVI     D,0
030.076  031            145X         DAD     D
030.077  321            146X         POP     D
030.100  311            147X         RET                      EXIT
```

```
                                150X **      $DADA. - ADD (0,A) TO (H,L)
                                151X *
                                152X *       ENTRY   NONE
                                153X *       EXIT    (HL) = (HL) + (0A)
                                154X *       USES    A,F,H,L
                                155X
                                156X
     030.101  205               157X $DADA.  ADD     L
     030.102  157               158X         MOV     L,A
     030.103  320               159X         RNC
     030.104  044               160X         INR     H
     030.105  311               161X         RET


                                164X **      $DU66 - UNSIGNED 16 / 16 DIVIDE.
                                165X *
                                166X *       (HL) = (BC)/(DE)
                                167X *
                                168X *       ENTRY   (BC), (DE) PRESET
                                169X *       EXIT    (HL) = RESULT
                                170X *               (DE) = REMAINDER
                                171X *       USES    ALL
                                172X
                                173X
     030.106  172               174X $DU66   MOV     A,D             TWOS COMPLEMENT (DE)
     030.107  057               175X         CMA
     030.110  127               176X         MOV     D,A
     030.111  173               177X         MOV     A,E
     030.112  057               178X         CMA
     030.113  137               179X         MOV     E,A
     030.114  023               180X         INX     D
     030.115  172               181X         MOV     A,D
     030.116  263               182X         ORA     E
     030.117  312 205 030       183X         JZ      DU665           IF DIVIDE BY 0
     030.122  257               184X         XRA     A
                                185X
                                186X *       SHIFT (DE) LEFT UNTIL:
                                187X *
                                188X *       1) DE > BL
                                189X *       2) OVERFLOW
                                190X
     030.123  142               191X DU661   MOV     H,D
     030.124  153               192X         MOV     L,E
     030.125  011               193X         DAD     B
     030.126  322 143 030       194X         JNC     DU662           IS TOO LARGE
     030.131  074               195X         INR     A               COUNT SHIFT
     030.132  142               196X         MOV     H,D
     030.133  153               197X         MOV     L,E
     030.134  051               198X         DAD     H
     030.135  353               199X         XCHG                    (DE) = (DE)*2
     030.136  332 123 030       200X         JC      DU661           IF NOT OVERFLOW
                                201X
                                202X *       (DE) OVERFLOWED, PUT IT BACK.
```

```
                          203X
030.141  353            204X          XCHG
030.142  075            205X          DCR     A               REMOVE EXTRA COUNT
                          206X
                          207X *        READY TO START SUBTRACTING. (A) = LOOP COUNT
                          208X
030.143  140            209X DU662    MOV     H,B             (H,L) = WORKING VALU
030.144  151            210X          MOV     L,C
030.145  001 000 000    211X          LXI     B,0             (BC) = RESULT
030.150  365            212X DU663    PUSH    PSW             SAVE (A)
030.151  031            213X          DAD     D
030.152  332 163 030    214X          JC      DU664           IF SUBTRACT OK
030.155  175            215X          MOV     A,L             ADD BACK IN
030.156  223            216X          SUB     E
030.157  157            217X          MOV     L,A
030.160  174            218X          MOV     A,H
030.161  232            219X          SBB     D
030.162  147            220X          MOV     H,A
030.163  171            221X DU664    MOV     A,C
030.164  027            222X          RAL
030.165  117            223X          MOV     C,A
030.166  170            224X          MOV     A,B
030.167  027            225X          RAL
030.170  107            226X          MOV     B,A
                          227X
                          228X *        RIGHT SHFT (DE)
                          229X
030.171  067            230X          STC
030.172  172            231X          MOV     A,D
030.173  037            232X          RAR
030.174  127            233X          MOV     D,A
030.175  173            234X          MOV     A,E
030.176  037            235X          RAR
030.177  137            236X          MOV     E,A
030.200  361            237X          POP     PSW
030.201  075            238X          DCR     A
030.202  362 150 030    239X          JP      DU663           IF NOT DONE
030.205  353            240X DU665    XCHG                    (D,E) = REMAINDER
030.206  140            241X          MOV     H,B             (HL) = RESULT
030.207  151            242X          MOV     L,C
030.210  311            243X          RET


                          246X **       $HLIHL - LOAD HL INDIRECT THROUGH HL.
                          247X *
                          248X *        (HL) = ((HL))
                          249X *
                          250X *        ENTRY   NONE
                          251X *        EXIT    NONE
                          252X *        USES    A,H,L
                          253X
030.211  176            254X $HLIHL   MOV     A,M
030.212  043            255X          INX     H
```

```
030.213  146          256X          MOV     H,M
030,214  157          257X          MOV     L,A
030.215  311          258X          RET


                      261X **       $CDEHL - COMPARE (DE) TO (HL)
                      262X *
                      263X *        $CDEHL COMPARES (DE) TO (HL) FOR EQUALITY.
                      264X *
                      265X *        ENTRY   NONE
                      266X *        EXIT    'Z' SET IF (DE) = (HL)
                      267X *        USES    A,F
                      268X
                      269X
030.216  173          270X $CDEHL   MOV     A,E
030.217  255          271X          XRA     L
030.220  300          272X          RNZ                     IF DIFFERENT
030.221  172          273X          MOV     A,D
030.222  254          274X          XRA     H
030.223  311          275X          RET


                      278X **       $CHL - COMPLEMENT (HL).
                      279X *
                      280X *        (HL) = -(HL)            TWO'S COMPLEMENT
                      281X *
                      282X *        ENTRY   NONE
                      283X *        EXIT    NONE
                      284X *        USES    A,F,H,L
                      285X
                      286X
030.224  174          287X $CHL     MOV     A,H
030.225  057          288X          CMA
030.226  147          289X          MOV     H,A
030.227  175          290X          MOV     A,L
030.230  057          291X          CMA
030.231  157          292X          MOV     L,A
030.232  043          293X          INX     H
030.233  311          294X          RET
```

```
                        297X **     $INDL - INDEXED LOAD.
                        298X *
                        299X *      $INDL LOADS DE WITH THE TWO BYTES AT (HL)+DISPLACMENT
                        300X *
                        301X *      THIS ACTS AS AN INDEXED FULL WORD LOAD.
                        302X *
                        303X *      (DE) = ( (HL) + DSPLACEMENT )
                        304X *
                        305X *      ENTRY   ((RET)) = DISPLACMENT (FULL WORD)
                        306X *              (HL) = TABLE ADDRESS
                        307X *      EXIT    TO (RET+2)
                        308X *      USES    A,F,D,E
                        309X
                        310X
030.234  343            311X $INDL  XTHL                    (HL) = RET, ((SP)) = TBL ADDRESS
030.235  136            312X        MOV     E,M
030.236  043            313X        INX     H
030.237  126            314X        MOV     D,M             (DE) = DISPLACEMENT
                        315X
030.240  043            316X        INX     H
030.241  343            317X        XTHL                    ((SP)) = RET, (HL) = TBL ADDRESS
030.242  353            318X        XCHG                    (DE) = TBL ADDRESS, (HL) = DISPLACEMENT
030.243  031            319X        DAD     D               (HL) = TARGET ADDRESS
030.244  176            320X        MOV     A,M
030.245  043            321X        INX     H
030.246  146            322X        MOV     H,M
030.247  157            323X        MOV     L,A             (HL) = ((HL))
030.250  353            324X        XCHG                    (DE) = VALUE, (HL) = TABLE ADDRESS
030.251  311            325X        RET


                        328X **     $MOVE - MOVE DATA
                        329X *
                        330X *      $MOVE MOVES A BLOCK OF BYTES TO A NEW MEMORY ADDRESS.
                        331X *      IF THE MOVE IS TO A LOWER ADDRESS, THE BYTES ARE MOVED FROM
                        332X *      FIRST TO LAST.
                        333X *
                        334X *      IF THE MOVE IS TO A HIGHER ADDRESS, THE BYTES ARE MOVED FROM
                        335X *      LAST TO FIRST.
                        336X *
                        337X *      THIS IS DONE SO THAT AN OVERLAPED MOVE WILL NOT 'RIPPLE'.
                        338X *
                        339X *      ENTRY   (BC) = COUNT (MUST BE < 32768)
                        340X *              (DE) = FROM
                        341X *              (HL) = TO
                        342X *      EXIT    MOVED
                        343X *              (DE) = ADDRESS OF NEXT FROM BYTE
                        344X *              (HL) = ADDRESS OF NEXT *TO* BYTE
                        345X *              'C' CLEAR
                        346X *      USES    ALL
                        347X
                        348X
030.252                 349X $MOVE  EQU     *
```

```
030.252  170            350X          MOV     A,B
030.253  261            351X          ORA     C
030.254  310            352X          RZ                      NONE TO MOVE
030.255  175            353X          MOV     A,L             COMPARE *FROM* TO *TO*
030.256  223            354X          SUB     E
030.257  174            355X          MOV     A,H
030.260  232            356X          SBB     D
030.261  332 311 030     357X          JC      MOV2            IS MOVE DOWN (TO LOWER ADDRESSES)
                         358X
                         359X *        IS MOVE UP (TO HIGHER ADDRESSES)
                         360X
030.264  013            361X          DCX     B
030.265  011            362X          DAD     B               (HL) = *TO* LWA
030.266  345            363X          PUSH    H               SAVE *TO* LIMIT
030.267  353            364X          XCHG
030.270  011            365X          DAD     B               (HL) = *FROM* LWA
030.271  345            366X          PUSH    H               SAVE *FROM* LIMIT
                         367X
030.272  176            368X MOV1     MOV     A,M             MOVE BYTE
030.273  022            369X          STAX    D
030.274  033            370X          DCX     D               INCREMENT *TO* ADDRESS
030.275  053            371X          DCX     H               INCREMENT *FROM* ADDRESS
030.276  013            372X          DCX     B               DECREMENT COUNT
030.277  170            373X          MOV     A,B
030.300  247            374X          ANA     A
030.301  362 272 030     375X          JP      MOV1            MORE TO GO
030.304  321            376X          POP     D               (DE) = *FROM* LIMIT
030.305  341            377X          POP     H               (HL) = *TO* LIMIT
030.306  023            378X          INX     D
030.307  043            379X          INX     H
030.310  311            380X          RET                     DONE
                         381X
                         382X *        IS MOVE DOWN (TO LOWER ADDRESSES)
                         383X
030.311  032            384X MOV2     LDAX    D               MOVE BYTE
030.312  167            385X          MOV     M,A
030.313  043            386X          INX     H               INCREMENT *FROM*
030.314  023            387X          INX     D               INCREMENT *TO*
030.315  013            388X          DCX     B               DECREMENT COUNT
030.316  170            389X          MOV     A,B
030.317  261            390X          ORA     C
030.320  302 311 030     391X          JNZ     MOV2            IF NOT DONE
030.323  311            392X          RET                     DONE


                         395X **       $MU10 - MULTIPLY UNSIGNED 16 BIT QUANTITY BY 10.
                         396X *
                         397X *        (HL) = (DE)*10
                         398X *
                         399X *        ENTRY   (DE) = MULTIPLIER
                         400X *        EXIT    'C' CLEAR IF OK
                         401X *                (HL) = PRODUCT
                         402X *                'C' SET IF ERROR
```

```
                        403X *       USES    D,E,H,L,F
                        404X
                        405X
    030.324  353        406X $MU10   XCHG                 (HL) = MULTIPLIER
    030.325  051        407X         DAD     H           (HL) = X*2
    030.326  330        408X         RC
    030.327  124        409X         MOV     D,H
    030.330  135        410X         MOV     E,L
    030.331  051        411X         DAD     H           (HL) = X*4
    030.332  330        412X         RC
    030.333  051        413X         DAD     H           (HL) = X*8
    030.334  330        414X         RC
    030.335  031        415X         DAD     D           (HL) = X*10
    030.336  311        416X         RET


                        419X **      $MU66 - UNSIGNED 16X16 MULTIPLY.'
                        420X *
                        421X *       ENTRY (BC) = MULTIPLICAND
                        422X *             (DE) = MULTIPLIER
                        423X *       EXIT  (HL) = RESULT
                        424X *             'Z' SET IF NOT OVERFLOW
                        425X *       USES  ALL
                        426X
                        427X
    030.337  257        428X $MU66   XRA     A
    030.340  365        429X         PUSH    PSW         SAVE OVERFLOW STATUS
    030.341  041 000 000 430X        LXI     H,0         (HL) = RESULT ACCUMULATOR
                        431X
    030.344  170        432X MU661   MOV     A,B
    030.345  037        433X         RAR
    030.346  107        434X         MOV     B,A
    030.347  171        435X         MOV     A,C
    030.350  037        436X         RAR
    030.351  117        437X         MOV     C,A
    030.352  322 364 030 438X        JNC     MU662       IF BIT CLEAR
    030.355  031        439X         DAD     D
    030.356  322 364 030 440X        JNC     MU662       IF NOT OVERFLOW
    030.361  361        441X         POP     PSW
    030.362  074        442X         INR     A
    030.363  365        443X         PUSH    PSW
    030.364  170        444X MU662   MOV     A,B
    030.365  261        445X         ORA     C           SEE IF MULTIPLIER 0
    030.366  312 005 031 446X        JZ      MU663       IS ZERO; AM DONE
    030.371  353        447X         XCHG
    030.372  051        448X         DAD     H           (D,E) = (DE)*2
    030.373  353        449X         XCHG
    030.374  322 344 030 450X        JNC     MU661       IF NOT OVERFLOW
    030.377  361        451X         POP     PSW
    031.000  074        452X         INR     A
    031.001  365        453X         PUSH    PSW         FLAG OVERFLOW
    031.002  303 344 030 454X        JMP     MU661       PROCESS NEXT BIT
                        455X
```

```
031.005  361            456X MU663   POP     PSW             (A,F) = OVERFLOW STATUS
031.006  311            457X         RET


                        460X **      $MU86 - MULTIPLY 8X16 UNSIGNED.
                        461X *
                        462X *       $MU86 MULTIPLIES A 16 BIT VALUE BY A 8
                        463X *       BIT VALUE.
                        464X *
                        465X *       ENTRY   (A) = MULTIPLIER
                        466X *               (DE) = MULTIPLICAND
                        467X *       EXIT    (HL) = RESULT
                        468X *               'Z' SET IF NOT OVERFLOW
                        469X *       USES    A,F,H,L
                        470X
                        471X
031.007  041 000 000    472X $MU86   LXI     H,0             (HL) = RESULT ACCUMULATOR
031.012  305            473X         PUSH    B
031.013  104            474X         MOV     B,H             (B) = OVERFLOW FLAG
031.014  267            475X MU860   ORA     A               CLEAR CARRY
                        476X
031.015  037            477X MU861   RAR
031.016  322 026 031    478X         JNC     MU862           IF NOT TO ADD
031.021  031            479X         DAD     D
031.022  322 026 031    480X         JNC     MU862           NOT OVERFLOW
031.025  004            481X         INR     B
031.026  267            482X MU862   ORA     A
031.027  312 044 031    483X         JZ      MU863           IF DONE
031.032  353            484X         XCHG
031.033  051            485X         DAD     H
031.034  353            486X         XCHG
031.035  322 015 031    487X         JNC     MU861           LOOP IF NOT OVERFLOW
031.040  004            488X         INR     B
031.041  303 014 031    489X         JMP     MU860
                        490X
031.044  260            491X MU863   ORA     B               SET *Z* FLAG IF NOT OVERFLOW
031.045  301            492X         POP     B               RESTORE (BC)
031.046  311            493X         RET


                        496X **      $RSTALL - RESTORE ALL REGISTERS.
                        497X *
                        498X *       $RSTALL RESTORES ALL THE REGISTERS OFF THE STACK, AND
                        499X *       RETURNS TO THE PREVIOUS CALLER.
                        500X *
                        501X *       ENTRY   (SP) = PSW
                        502X *               (SP+2) = BC
                        503X *               (SP+4) = DE
                        504X *               (SP+6) = HL
                        505X *               (SP+8) = RET
                        506X *       EXIT    TO *RET*, REGISTERS RESTORED
```

```
                        507X *       USES    ALL
                        508X
                        509X
    031.047  361       510X $RSTALL POP     PSW
    031.050  301       511X         POP     B
    031.051  321       512X         POP     D
    031.052  341       513X         POP     H
    031.053  311       514X         RET


                        516X **      $SAVALL - SAVE ALL REGISTERS ON STACK.
                        517X *
                        518X *       $SAVALL SAVES ALL THE REGISTERS ON THE STACK.
                        519X *
                        520X *       ENTRY   NONE
                        521X *       EXIT    (SP) = PSW
                        522X *               (SP+2) = BC
                        523X *               (SP+4) = DE
                        524X *               (SP+6) = HL
                        525X *       USES    H,L
                        526X
                        527X
    031.054  343       528X $SAVALL XTHL                    PUSH H, (HL) = RETURN ADDRESS
    031.055  325       529X         PUSH    D
    031.056  305       530X         PUSH    B
    031.057  365       531X         PUSH    PSW
    031.060  351       532X         PCHL                    RETURN TO CALLER


                        535X **      $TJMP - TABLE JUMP.
                        536X *
                        537X *       USAGE
                        538X *
                        539X *       CALL    $TJMP           (A) = INDEX
                        540X *       DW      ADDR1           INDEX = 0
                        541X *       .         .
                        542X *       .         .
                        543X *       .         .
                        544X *       DW      ADDRN           INDEX = N-1
                        545X *
                        546X *       ENTRY   (A) = INDEX
                        547X *       EXIT    TO PROCESSOR
                        548X *               (A) = INDEX*2
                        549X *       USES    A,F
                        550X
                        551X
    031.061  007       552X $TJMP   RLC                     (A) = INDEX*2
                        553X
    031.062            554X $TJMP.  EQU     *
    031.062  343       555X         XTHL                    (HL) = TABLE ADDRESS
    031.063  365       556X         PUSH    PSW             SAVE INDEX*2
    031.064  315 101 030 557X         CALL    $DADA.
```

```
031.067  176          558X          MOV     A,M
031.070  043          559X          INX     H
031.071  146          560X          MOV     H,M
031.072  157          561X          MOV     L,A
031.073  361          562X          POP     PSW             (A) = INDEX*2
031.074  343          563X          XTHL                    ADDRESS ON STACK
031.075  311          564X          RET                     JUMP TP PROCESSOR


                      567X **       $TBRA - BRANCH RELATIVE THOUGH TABLE.
                      568X *
                      569X *        $TBRA USES THE SUPPLIED INDEX TO SELECT A BYTE FROM THE
                      570X *        JUMP TABLE. THE CONTENTS OF THIS BYTE ARE ADDED TO THE
                      571X *        ADDRESS OF THE BYTE, YEILDING THE PROCESSOR ADDRESS.
                      572X *
                      573X *        CALL    $TBRA
                      574X *        DB      LAB1-*          INDEX = 0 FOR LAB1
                      575X *        DB      LAB2-*          INDEX = 1 FOR LAB2
                      576X *        DB      LABN-*          INDEX = N-1 FOR LABN
                      577X *
                      578X *        ENTRY   (A) = INDEX
                      579X *                (RET) = TABLE FWA
                      580X *        EXIT    TO COMPUTED ADDRESS
                      581X *        USES    F,H,L
                      582X
                      583X
031.076               584X $TBRA    EQU     *
031.076  343          585X          XTHL                    (HL) = TABLE ADDRESS
031.077  325          586X          PUSH    D
031.100  137          587X          MOV     E,A
031.101  026 000       588X          MVI     D,0
031.103  031          589X          DAD     D               (HL) = ADDRESS OF ELEMENT
031.104  136          590X          MOV     E,M
031.105  031          591X          DAD     D               (HL) = PROCESSOR ADDRESS
031.106  321          592X          POP     D
031.107  343          593X          XTHL
031.110  311          594X          RET


                      597X **       $TBLS - TABLE SEARCH
                      598X *
                      599X *        TABLE FORMAT
                      600X *
                      601X *        DB      KEY1,VAL1,
                      602X *        .       .
                      603X *        .       .
                      604X *        DB      KEYN,VALN
                      605X *        DB      0
                      606X *
                      607X *        ENTRY   (A) = PATTERN
```

```
                                608X *              (H,L) = TABLE FWA
                                609X *        EXIT  (A) = PATTERN IF FOUND
                                610X *              'Z' SET IF FOUND
                                611X *        USES  A,F,H,L
                                612X
                                613X
031.111  305                    614X $TBLS    PUSH  B
031.112  107                    615X          MOV   B,A
031.113  176                    616X $TBL1    MOV   A,M             (A) = CHARACTER
031.114  270                    617X          CMP   B
031.115  312 133 031            618X          JZ    $TBL2           IF MATC
031.120  247                    619X          ANA   A
031.121  043                    620X          INX   H
031.122  043                    621X          INX   H               SKIP PAST
031.123  302 113 031            622X          JNZ   $TBL1           IF NOT END OF TABLE
031.126  053                    623X          DCX   H
031.127  053                    624X          DCX   H
031.130  264                    625X          ORA   H               CLEAR 'Z'
031.131  076 000                626X          MVI   A,0             SET (A) = 0 FOR OLD USERS
                                627X
                                628X *        DONE
                                629X
031.133  301                    630X $TBL2    POP   B
031.134  043                    631X          INX   H
031.135  311                    632X          RET


                                635X **       $TYPTX - TYPE TEXT.
                                636X *
                                637X *        $TYPTX IS CALLED TO TYPE A BLOCK OF TEXT ON THE SYSTEM CONSOLE.
                                638X *
                                639X *        IMBEDDED ZERO BYTES INDICATE A CARRIAGE RETURN LINE FEED,
                                640X *        A BYTE WITH THE 200Q BIT SET IS THE LAST BYTE IN THE MESSAGE.
                                641X *
                                642X *        ENTRY (RET) = TEXT
                                643X *        EXIT  TO (RET+LENGTH)
                                644X *        USES  A,F
                                645X
                                646X
031.136  343                    647X $TYPTX   XTHL                  (HL) = TEXT ADDRESS
031.137  315 144 031            648X          CALL  $TYPTX.         TYPE IT
031.142  343                    649X          XTHL
031.143  311                    650X          RET
                                651X
031.144  176                    652X $TYPTX.  MOV   A,M
031.145  346 177                653X          ANI   177Q
031.147  377 002                654X          DB    SYSCALL,.SCOUT
031.151  276                    655X          CMP   M
031.152  043                    656X          INX   H
031.153  312 144 031            657X          JE    $TYPTX.         MORE TO GO
031.156  311                    658X          RET
```

```
H17ROM - H17 ROM CODE LISTING.                                  HEATH H8ASM V1.4  01/20/78        PAGE    14
COMMON DECKS                                                    $UDD              15:54:22  11-MAY-78

                                661X **      $UDD - UNPACK DECIMAL DIGITS.
                                662X *
                                663X *       UDD CONVERTS A 16 BIT VALUE INTO A SPECIFIED NUMBER OF
                                664X *       DECIMAL DIGITS. THE RESULT IS ZERO FILLED.
                                665X *
                                666X *       ENTRY   (B,C) = ADDRESS VALUE
                                667X *               (A) = DIGIT COUNT
                                668X *               (H,L) = MEMORY ADDRESS
                                669X *       EXIT    (HL) = (HL) + (A)
                                670X *       USES    ALL
                                671X
                                672X
    031.157                     673X $UDD    EQU     *
    031.157  315 072 030        674X         CALL    $DADA
    031.162  345                675X         PUSH    H               SAVE FINAL (H,L) VALUE
                                676X
    031.163  365                677X UDD1    PUSH    PSW
    031.164  345                678X         PUSH    H
    031.165  021 012 000        679X         LXI     D,10
    031.170  315 106 030        680X         CALL    $DU66           (H,L) = VALUE/10
    031.173  345                681X         PUSH    H
    031.174  301                682X         POP     B               (B,C) = REMAINDER
    031.175  341                683X         POP     H
    031.176  076 060            684X         MVI     A,'0'
    031.200  203                685X         ADD     E               ADD REMAINDER
    031.201  053                686X         DCX     H
    031.202  167                687X         MOV     M,A             STORE DIGIT
    031.203  361                688X         POP     PSW
    031.204  075                689X         DCR     A
    031.205  302 163 031        690X         JNZ     UDD1            IF MORE TO GO
    031.210  341                691X         POP     H               RESTORE H
    031.211  311                692X         RET                        RETURN


                                695X **      $ZERO - ZERO MEMORY
                                696X *
                                697X *       $ZERO ZEROS A BLOCK OF MEMORY.
                                698X *
                                699X *       ENTRY   (HL) = ADDRESS
                                700X *               (B) = COUNT
                                701X *       EXIT    (A) = 0
                                702X *       USES    A,B,F,H,L
                                703X
                                704X
    031.212  257                705X $ZERO   XRA     A
    031.213  167                706X ZRO1    MOV     M,A
    031.214  043                707X         INX     H
    031.215  005                708X         DCR     B
    031.216  302 213 031        709X         JNZ     ZRO1            IF MORE
    031.221  311                710X         RET
                                711
```