Pascal-M
KIM-1
Interpreter Listing
1978, 1979

G.J. van der Grinten

W. van Gelderen

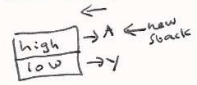
```
 1                      .NAM    'MACRO  INTEPRETER FOR PASCAL-M'
 2 2000                 *       =       $0000 ;KIM MEMORY STARTS AT ZERO
 3                ; --- MONITOR DEFINITIONS
 4      4F1C      COMMAN =        $1C4F
 5      5A1E      TTYIN  =        $1E5A
 6      A01E      TTYOUT =        $1EA0
 7                ; COPYRIGHT BY G.J.V.D.GRINTEN 1978.1979
 8                ;
 9                ; LOW CORE CONSTANTES
10 0000           PC      *       =       *+2 ;PROGRAM COUNTER
11 0002           HP      *       =       *+2 ;HEAP POINTER
12 0004           MP      *       =       *+2 ;MARK POINTER
13 0006           MPSAVE  *       =       *+2 ;MARK SAVE
14 0008           JTABP   *       =       *+2 ;JUMP TABLE POINTER
15 000A           JTAB2P  *       =       *+2 ;JUMP TABLE 2
16 000C           STRPTR  *       =       *+2 ;STRING PRINT POINTER
17 000E           TMP1L8  *       =       *+8 ;8 WORD TEMP FIELD
18 0016           TMP2L2  *       =       *+2 ;2 WORD TEMP
19 0018           TMP3L2  *       =       *+2 ;2 WORD TEMP
20 001A           EOL     *       =       *+1 ;END OF INPUT LINE BOOLEAN
21 001B           OP      *       =       *+1 ;LAST OP CODE
22 001C           ADDR    *       =       *+2 ;LOADER ADDRESS POINTER
23 001E           ASSEM   *       =       *+1 ;ASSEM A WORD
24 001F           CHKSUM  *       =       *+1 ;CHECKSUM
25 0020           STACK   *       =       *+2 ;SOFTWARE STACK POINTER
26 0022           EOFB    *       =       *+1 ;END OF FILE BOOLEAN
27 0023           SETIN   *       =       *+1 ;INPUT SWITCH
28 0024           SETOUT  *       =       *+1 ;OUTPUT SWITCH
```

```
30 0025                    *       =       *+$2C-* ;FREE ROOM
31 002C 07 30    BASE      .WOR    BASBAS  ;BASE MEMORY
32 002E FF 9F    ENDCOR    .WOR    $9FFF   ;END CORE
33 0030 50 41 53
33 0033 43 41 4C
33 0036 2D 4D 20
33 0039 4C 4F 41
33 003C 44 45 44
33 003F 0D       NAME      .BYT    'PASCAL-M       LOADED',$0D
34 00C0                    *       =       $C0     BECAUSE THE COMPILER KNOWS THIS
35 00C0 A9 00    PROC1     LDA     #$00    ;PROCEDURE SETTTYIN ;EXTERN = $0040;
36 00C2 85 23              STA     SETIN
37 00C4 4C DC 00           JMP     PROCEX
38 00C7 EA                 NOP
39 00C8 A9 01    PROC2     LDA     #$01    ;PROCEDURE SETHSR ;EXTERN = $0048;
40 00CA 85 23              STA     SETIN
41 00CC 4C DC 00           JMP     PROCEX
42 00CF EA                 NOP
   00D0 A9 00    PROC3     LDA     #$00    ;PROCEDURE SETTPYOUT ;EXTERN = $0050;
44 00D2 85 24              STA     SETOUT
45 00D4 4C DC 00           JMP     PROCEX
46 00D7 EA                 NOP
47 00D8 A9 01    PROC4     LDA     #$01    ;PROCEDURE SETHSP ;EXTERN = $0058;
48 00DA 85 24              STA     SETOUT
49 00DC 20 7B 20 PROCEX    JSR     PULL2
50 00DF 4C 45 23           JMP     LOOP
```

```
52 00E2                    *       =       $200 ;PROCEDURE BUFFER
53 0200              PRCBUF *       =       *+$00C8 ;ROOM FOR 100 PROCEDURES
54 02C8                    *       =       $2000 ;START POINT
55 2000 4C 1F 22     ENTRY  JMP     LOADIN
56 2003 4C 5D 22     CONTIN JMP     PGO     ;ENTER HERE FOR REPEAT PROGRAM(OR RE
57 2006 A6 23        RDT    LDX     SETIN
58 2008 D0 03                BNE     RDHSR
59 200A 4C F6 2D     RDT1   JMP     READ    ;READ SYSTEM DEVICE
60 200D 4C 06 2E     RDHSR  JMP     INALL   ;ALTERNATE INPUT READ
61 2010 A6 24        WRT    LDX     SETOUT
62 2012 D0 03                BNE     WRHSP
63 2014 4C FE 2D     WRT1   JMP     OUTPUT  ;SYSTEM OUTPUT DEVICE
64 2017 4C 4F 2E     WRHSP  JMP     OUTALL  ;ALTERNATE OUTPUT
65 201A A9 0D        CRLF   LDA     #$0D
66 201C 20 10 20             JSR     WRT
67 201F A9 0A                LDA     #$0A
68 2021 20 10 20             JSR     WRT
69 2024 A6 24                LDX     SETOUT
   2026 D0 0E                BNE     CRLF2   ;IF OUTPUT TO HIGH SPEED
71 2028 98                   TYA             ;SAVE Y CONTENT
72 2029 48                   PHA
73 202A A0 08                LDY     #$08
74 202C A9 00        CRLF1  LDA     #$00    ;SEND FILLER CHAR
75 202E 20 14 20             JSR     WRT1    ;WE KNEW WICH DEVICE
76 2031 88                   DEY
77 2032 D0 F8                BNE     CRLF1
78 2034 68                   PLA
79 2035 A8                   TAY             ;RESTORE Y
80 2036 60           CRLF2  RTS
81 2037 4C 4F 1C     EXIT   JMP     COMMAN
```

```
 83                     ;
 84                     ; USEFULL ROUTINES
 85                     ;
 86 203A 18        INCR   CLC              ;INCREMENT A POINTER (16)BIT
 87 203B B5 00             LDA     00,X     ;ENTER WITH X POINTING TO POINTER
 88 203D 69 01             ADC     #01
 89 203F 95 00             STA     00,X
 90 2041 90 06             BCC     INCR1    ;IF NO CARRY CARRY ON
 91 2043 B5 01             LDA     01,X
 92 2045 69 00             ADC     #00
 93 2047 95 01             STA     01,X
 94 2049 A2 00     INCR1   LDX     #$00
 95 204B 60                RTS
 96 204C 38        DECR    SEC             ;DECREMENT A POINTER 16 BIT
 97 204D B5 00             LDA     00,X     ;ENTER POINTER IN X
 98 204F E9 01             SBC     #01
 99 2051 95 00             STA     00,X
100 2053 B0 06             BCS     DECR1    ;IF CARRY SET NO ADJUST
  1 2055 B5 01             LDA     01,X
102 2057 E9 00             SBC     #00
103 2059 95 01             STA     01,X
104 205B A2 00     DECR1   LDX     #$00
105 205D 60                RTS
106 205E 18        INCPC2  CLC             ;ADD FASTLY TWO TO PC
107 205F A5 00             LDA     PC
108 2061 69 02             ADC     #$02
109 2063 85 00             STA     PC
110 2065 90 06             BCC     INCPCA   ;IF NO NEED TO ADJUST UPPER
111 2067 A5 01             LDA     PC+1
112 2069 69 00             ADC     #$00     ;ADJUST WITH CARRY
113 206B 85 01             STA     PC+1
114 206D A2 00     INCPCA  LDX     #$00     ;GENERAL CONVENTION LEAVE X =00
115 206F 60                RTS
116 2070 C9 3A     HEXIT   CMP     #$3A
117 2072 08                PHP
118 2073 29 0F             AND     #$0F
119 2075 28                PLP
120 2076 90 02             BCC     HEXIT1   ;0..9
121 2078 69 08             ADC     #$08     ;A..F CARRY WAS SET
122 207A 60        HEXIT1  RTS
```

```
                        ;
                        ; STACK OPERATIONS PULL AND PUSH
                        ; PULL2 AND PUSH2
                        ; ENTER OR EXIT WITH VAL IN A OR A + Y
                        ; A CONTAINS UPPER 8 BITS OF VALUE ON STACK? OPERATIONS
                        ; Y CONTAINS LOWER 8 BITS ....
207B 20 7F 20  PULL2    JSR     PULL    ;GET Y FIRST
207E A8                 TAY
207F A2 20     PULL     LDX     #STACK
2081 20 3A 20           JSR     INCR
2084 A1 20              LDA     (STACK,X)
2086 60                 RTS
                        ;
2087 A2 00     PUSH2    LDX     #00     ;CLEAR X FIRST TO BE SHURE
2089 20 8D 20           JSR     PUSH    ;SAVE A FIRST
208C 98                 TYA
208D 81 20     PUSH     STA     (STACK,X)
208F A2 20              LDX     #STACK
2091 20 4C 20           JSR     DECR
2094 60                 RTS
2095 85 0D     STRING   STA     STRPTR+01       ;HIGH PART OF ADDRESS
2097 86 0C              STX     STRPTR  ;LOW PART
2099 A0 00              LDY     #00
209B 84 24              STY     SETOUT  ;RESET OUTPUT DEVICE
209D 20 1A 20           JSR     CRLF    ; GIVE A CLEAR LINE
20A0 B1 0C     STRINA   LDA     (STRPTR),Y
20A2 C9 0D              CMP     #$0D    ;TEST FOR EOM (CR)
20A4 F0 06              BEQ     STINB
20A6 20 14 20           JSR     WRT1    ;PRINT CHAR
20A9 C8                 INY
20AA D0 F4              BNE     STRINA  ;LEAVE NO HOLE IN THIS ROUTINE
20AC 20 1A 20  STINB    JSR     CRLF
20AF 4C 37 20           JMP     EXIT    ;GO TO MONITOR
                        ;
20B2 20 43 48
20B5 45 43 4B
20B8 53 55 4D
20BB 20 45 52
20BE 52 4F 52
20C1 0D         MSG001  .BYT    '       CHECKSUM ERROR',$0D
20C2 20 45 4E
20C5 44 20 50
20C8 41 53 43
20CB 41 4C 0D  MSG002  .BYT    '       END PASCAL',$0D
20CE 20 42 41
20D1 44 20 49
20D4 4E 53 54
20D7 52 55 43
20DA 54 49 4F
20DD 4E 0D     MSG003  .BYT    '       BAD INSTRUCTION',$0D
20DF 20 57 52
20E2 4F 4E 47
20E5 20 52 45
20E8 43 4F 52
20EB 44 0D     MSG004  .BYT    '       WRONG RECORD',$0D
20ED 20 53 54
20F0 41 43 4B
20F3 20 4F 56
20F6 45 52 46
20F9 4C 4F 57
```



—> leaves X=0 stack →

```
20FC 0D       MSG005 .BYT   '       STACK OVERFLOW',$0D
20FD 20 43 41
2100 53 45 20
2103 49 4E 44
2106 45 58 20
2109 4F 55 54
210C 20 4F 46
210F 20 52 41
2112 4E 47 45
2115 0D       MSG006 .BYT   '       CASE INDEX OUT OF RANGE',$0D
2116 20 4E 4F
2119 54 20 49
211C 4D 50 4C
211F 45 4D 45
2122 4E 54 45
2125 44 0D    MSG007 .BYT   '       NOT IMPLEMENTED',$0D
2127 20 44 49
212A 56 49 53
212D 49 4F 4E
2130 20 42 59
2133 20 5A 45
2136 52 4F 0D MSG008 .BYT   '       DIVISION BY ZERO',$0D
              ;
              ;*
              ; BASEAD COMPUTE BASE ADDRESS OF LEVEL (OP-X)
              ; LEAVE RESULT IN TMP2L2
              ;
2139 A5 04    BASEAD LDA    MP      ;MARK POINTER IS DEFAULT (LEVEL = 0
213B 85 16           STA    TMP2L2
213D A5 05           LDA    MP+1
213F 85 17           STA    TMP2L2+1
2141 A5 1B           LDA    OP      ;CHECK LEVEL
2143 29 0F           AND    #$0F    ;ISOLATE LEVEL
2145 F0 1E           BEQ    BAS2    ;IF LEVEL = 0 THEN WE ARE FINISHED
2147 AA              TAX            ;X NOW SAYS HOW DEEP THE LEVEL IS
2148 38       BAS1   SEC            ;OFFSET FOR MARK POINTER IN STACK
2149 A5 16           LDA    TMP2L2
214B E9 01           SBC    #$01    ;POINT TO BASE MP
214D 85 16           STA    TMP2L2
214F A5 17           LDA    TMP2L2+1
2151 E9 00           SBC    #$00    ;ADJUST UPPER ADDRESS
2153 85 17           STA    TMP2L2+1
2155 A0 01           LDY    #$01    ;POINT TO UPPER PART OF INDIRECT MP
2157 B1 16           LDA    (TMP2L2),Y      ;THIS IS THE HIGH PART
2159 48              PHA            ;SAVE IT ON STACK
215A 88              DEY
215B B1 16           LDA    (TMP2L2),Y      ;GET LOW PART
215D 85 16           STA    TMP2L2
215F 68              PLA
2160 85 17           STA    TMP2L2+1        ;THIS IS THE HIGH PART
2162 CA              DEX            ;LEVEL  = LEVEL-1
2163 D0 E3           BNE    BAS1    ;NOT FINISHED
2165 60       BAS2   RTS            ;READY
              ;
              ; ADDGET GET A ADDRESS OF LEVEL X 16 BIT
              ;
2166 20 39 21 ADDGET JSR    BASEAD  ;GET BASE ADDRESS OF PROPER LEVEL
2169 A0 01           LDY    #$01    ;GET LOW PART OF ADDRESS FIRST
216B 38              SEC
216C A5 16           LDA    TMP2L2  ;PC POINTS TO HIGH PART
```

```
204 216E F1 00          SBC     (PC),Y  ;LOW PART ADD
205 2170 85 16          STA     TMP2L2  ;REPLACE
206 2172 88             DEY             ;GET HIGH PART
207 2173 A5 17          LDA     TMP2L2+1
  1 2175 F1 00          SBC     (PC),Y
  2 2177 85 17          STA     TMP2L2+1        ;TMP2L2 NOW POINTS TO ADDRESS
  3 2179 A2 00          LDX     #PC
  4 217B 20 3A 20       JSR     INCR    ;INCREMENT PC BY TWO
  5 217E A2 00    ADDG1 LDX     #PC
  6 2180 20 3A 20       JSR     INCR
  7 2183 A2 16          LDX     #TMP2L2 ;SUBTRACT 1 FROM ADDRESS BECAUSE LOAF
  8 2185 20 3A 20       JSR     INCR    ;THIS IS CORRECT FOR ALL CALLS
  9 2188 60             RTS             ;FINIETO MON AMIE
 10                     ;
 11                     ; SADGET GET SMALL ADDRESS OF PROPER LEVEL (SAME AS'ADDGET)
 12                     ;
 13 2189 20 39 21 SADGET JSR     BASEAD
 14 218C 38              SEC
  1 218D A0 00           LDY     #$00
 16 218F A5 16           LDA     TMP2L2  ;GET ADDRESS POINTER
 17 2191 F1 00           SBC     (PC),Y
 18 2193 85 16           STA     TMP2L2
 19 2195 A5 17           LDA     TMP2L2+1
 20 2197 E9 00           SBC     #$00
 21 2199 85 17           STA     TMP2L2+1
 22 219B 4C 7E 21        JMP     ADDG1
 23 219E 3B 24    JTAB   .WOR    LEQ2    ;2 BYTE LESS THAN OR EQUAL TEST
 24 21A0 4E 24           .WOR    FOR     ;FOR LOOP PROCESSING INSTRUCTION
 25 21A2 E5 24           .WOR    LEQM    ;LESS OR EQUAL TEST FOR ARRAYS
 26 21A4 FA 24           .WOR    LES2    ;LESS THAN TEST FOR 2-BYTE DATA ITEMS
 27 21A6 1F 25           .WOR    LEQ8    ;IS CONTAINED IN TEST FOR SETS
 28 21A8 4C 25           .WOR    LESM    ;LIKE LEQM
 29 21AA 60 25           .WOR    EQU2    ;EQUAL TEST FOR 2-BYTE
 30 21AC 75 25           .WOR    GEQ8    ;CONTAINS TEST FOR SETS
 31 21AE A6 25           .WOR    EQUM    ;EQUAL TEST FOR ARRAYS AND RECORDS
 32 21B0 B9 25           .WOR    EQU8    ;EQUAL TEST FOR SETS
 33 21B2 E4 25           .WOR    IND1    ;INDERECTLY LOAD 1-BYTE DATA ITEM
 34 21B4 F4 25           .WOR    IND2    ;,, ,, 2 ,, ,,
 35 21B6 09 26           .WOR    IND8    ;,, 8 ,, ,,,
 36 21B8 19 26           .WOR    STO1    ;,, STORE 1 BYTE DATA ITEM
 37 21BA 25 26           .WOR    STO2    ;,, ,, 2 ,,
 38 21BC 3A 26           .WOR    STO8    ;,, ,, 8 ,, ,, ,,
 39 21BE 57 26           .WOR    LDC     ;LOAD 2-BYTE CONSTANT
 40 21C0 70 26           .WOR    RETP    ;RETURN FROM PROCEDURE OR FUNCTION
 41 21C2 91 26           .WOR    ADI     ;ADD INTEGER
 42 21C4 A4 26           .WOR    ANDB    ;BOOLEAN AND
 43 21C6 B4 26           .WOR    DIF     ;SET DIFFERENCE
 44 21C8 C8 26           .WOR    DVI     ;INTEGER DIVIDE
 45 21CA EF 26           .WOR    INN     ;TEST IF ELEMENT IN SET
 46 21CC 22 27           .WOR    INT     ;SET INTERSECTION
 47 21CE 34 27           .WOR    IOR     ;INCLUSIVE OR
 48 21D0 42 27           .WOR    MOD     ;MODULUS FUNCTION
 49 21D2 69 27           .WOR    MPI     ;INTEGER MULTIPLY
 50 21D4 90 27           .WOR    NGI     ;NEGATE INTEGER
 51 21D6 AE 27           .WOR    NOT     ;NEGATE BOOLEAN
 52 21D8 B9 27           .WOR    SBI     ;SUBTRACT INTEGER
 53 21DA CC 27           .WOR    SGS     ;GENERATE SINGLETON SET
 54 21DC FB 27           .WOR    UNI     ;SET UNION
 55 21DE 0D 28           .WOR    LNC     ;LOAD NEGATIVE CONSTANT
 56 21E0 26 28           .WOR    FJP     ;FALSE JUMP
```

```
 57 21E2 33 28            .WOR    UJP     ;UNCONDITIALY JUMP
 58 21E4 41 28            .WOR    DECB    ;DECREMENT
 59 21E6 5D 28            .WOR    INCB    ;INCREMENT
 60 21E8 71 28            .WOR    ENT     ;ENTER BLOCK
 61 21EA 87 28            .WOR    CAS     ;CASE STATEMENT PROCESSOR
 62 21EC F6 28            .WOR    MOV     ;MOVE STORAGE
 63 21EE 2B 29            .WOR    DEC1    ;DECREMENT BY 1
 64 21F0 3D 29            .WOR    INC1    ;INCREMENT BY 1
 65 21F2 48 29            .WOR    LDCS    ;LOAD SET CONSTANT (8-BYTES)
 66 21F4 62 29            .WOR    CAP     ;CALL ASSEMBLY PROCEDURE
 67 21F6 79 29            .WOR    LCA     ;LOAD CONSTANT ADDRESS
 68 21F8 99 29            .WOR    CSP     ;CALL STANDARD PROCEDURE
 69 21FA CE 29            .WOR    CUP1    ;CALL USER PROCEDURE SIMPLE
 70 21FC D9 29            .WOR    CUP2    ;COMPLEX CALL USER PROCEDURE
 71 21FE 2B 2A            .WOR    FIX21   ;CLEAN UP STACK AFTER CALL TO SINGLE
 72 2200 37 2A            .WOR    LNS     ;LOAD NULL SET
 73 2202 86 23    C2      .WOR    NOTIMP  ;DUMMY FOR DEBUG ERROR STOP
 74 2204 45 23    C3      .WOR    LOOP    ;NOP INSTRUCTION FOR DEBUG
    2206 00 00    C4      .WOR    0       ;ANOTHER DUMMY
 76                       ; HERE YOU CAN PUT ADITIONAL ADDRESSES FOR ROUTINES
 77 2208 9C 23    JTAB2   .WOR    LDCIS   ;LOAD SMALL INTEGER CONSTANT (0,.15)
 78 220A A6 23            .WOR    LDAS    ;LOAD SHORT ADDRESS
 79 220C B3 23            .WOR    LDAD    ;LOAD ADDRESS
 80 220E C0 23            .WOR    MSTO    ;MARK STACK WITHOUT RETURN BYTES
 81 2210 E2 23            .WOR    MSTN    ;MARK STACK WITH RETURN BYTES
 82 2212 F6 23            .WOR    LOD1    ;LOAD 1 BYTE DATA ITEM ONTO STACK
 83 2214 05 24            .WOR    LOD2    ;LOAD 2 BYTE DATA ITEM ONTO STACK
 84 2216 1A 24            .WOR    STR1    ;STORE 1 BYTE DATA ITEM INTO MEMORY
 85 2218 26 24            .WOR    STR2    ;STORE 2 BYTE DATA ITEM INTO MEMORY
 86                       ; NO MORE CAN GO INTO JTAB2 BECAUSE RANGE 0,.8
 87 221A 30       MSTART  .BYT    $30     ;MARK STACK
 88 221B BE 00            .BYT    $BE,0   ;CALL PROC 0
 89 221D BD 0B            .BYT    $BD,$0B ;STOP -END PASCAL-
 90 221F A5 2C    LOADIN  LDA     BASE    ;INIT VARIUS CELLS
 91 2221 85 1C            STA     ADDR
 92 2223 A5 2D            LDA     BASE+1
 93 2225 85 1D            STA     ADDR+1
    2227 A2 00            LDX     #$00
 95 2229 86 24            STX     SETOUT
 96                       ; HERE STARTS THE LOAD OF M CODE
 97 222B 20 2D 2E         JSR     INOT    ;SELECT INPUT DEVICE TO LOAD PASCAL
 98 222E A9 00    LOADER  LDA     #00
 99 2230 85 1F            STA     CHKSUM
100 2232 20 0D 20         JSR     RDHSR   ;GET A CHAR FROM HSR
101 2235 C9 50            CMP     #'P'    ;IS IT P
102 2237 D0 F5            BNE     LOADER
103 2239 20 0D 20         JSR     RDHSR   ;GET RECORD ID
104 223C C9 31            CMP     #'1'
105 223E F0 4C            BEQ     REC1
106 2240 C9 32            CMP     #'2'
107 2242 F0 6D            BEQ     REC2
108 2244 C9 34            CMP     #'4'
109 2246 D0 03            BNE     LOADA
110 2248 4C E1 22         JMP     REC4
111 224B C9 39    LOADA   CMP     #'9'
112 224D D0 07            BNE     LOADB
113 224F A9 00            LDA     #>NAME
114 2251 A2 30            LDX     #<NAME
115 2253 4C 95 20         JMP     STRING  ;ENTER MONITOR ON COMPLETION OF LOAD
116 2256 A9 20    LOADB   LDA     #>MSG004
```

```
11? 2258 A2 DF          LDX     #<MSG004
118 225A 4C 95 20       JMP     STRING  ;JUMP INTO MONITOR
119 225D A9 22    PGO   LDA     #>MSTART
120 225F 85 01          STA     PC+1
121 2261 A9 1A          LDA     #<MSTART
122 2263 85 00          STA     PC
123 2265 A5 1C          LDA     ADDR
124 2267 85 02          STA     HP
125 2269 A5 1D          LDA     ADDR+1
126 226B 85 03          STA     HP+1
127 226D A5 2E          LDA     ENDCOR
128 226F 85 20          STA     STACK
129 2271 85 04          STA     MP
130 2273 A5 2F          LDA     ENDCOR+1
131 2275 85 21          STA     STACK+1
132 2277 85 05          STA     MP+1
133 2279 A9 00          LDA     #00
134 227B 85 1A          STA     EOL
13? 227D 85 22          STA     EOFB
136 227F 85 23          STA     SETIN
137 2281 85 24          STA     SETOUT
138 2283 20 2D 2E       JSR     INOT
139 2286 20 1A 20       JSR     CRLF
140 2289 4C 45 23       JMP     LOOP
141 228C 20 0B 23 REC1  JSR     GETHEX
142 228F 85 1B          STA     OP      ;SAVE NO OF BYTES TO LOAD
143 2291 20 0B 23 REC1X1 JSR    GETHEX
144 2294 C6 1B          DEC     OP
145 2296 F0 0B          BEQ     REC1X2
146 2298 A0 00          LDY     #00
147 229A 91 1C          STA     (ADDR),Y
148 229C A2 1C          LDX     #ADDR
149 229E 20 3A 20       JSR     INCR
150 22A1 F0 EE          BEQ     REC1X1  ;SHOULD WORK OF LDX 0
151 22A3 E6 1F    REC1X2 INC    CHKSUM
152 22A5 D0 03          BNE     LODERR
153 22A7 4C 2E 22       JMP     LOADER
15? 22AA A9 20    LODERR LDA    #>MSG001
155 22AC A2 B2          LDX     #<MSG001
156 22AE 4C 95 20       JMP     STRING
157 22B1 20 0B 23 REC2  JSR     GETHEX
158 22B4 85 0F          STA     TMP1L8+1
159 22B6 20 0B 23       JSR     GETHEX
160 22B9 18             CLC
161 22BA 65 2C          ADC     BASE
162 22BC 85 0E          STA     TMP1L8
163 22BE A5 0F          LDA     TMP1L8+1
164 22C0 65 2D          ADC     BASE+1
165 22C2 85 0F          STA     TMP1L8+1
166 22C4 20 0B 23       JSR     GETHEX
167 22C7 A2 00          LDX     #0
168 22C9 81 0E          STA     (TMP1L8,X)
169 22CB A2 0E          LDX     #TMP1L8
170 22CD 20 3A 20       JSR     INCR
171 22D0 20 0B 23       JSR     GETHEX
17? 22D3 A2 00          LDX     #$00    ;LOAD X BECAUSE GETHEX DESTROIS X
17? 22D5 81 0E          STA     (TMP1L8,X)
174 22D7 20 0B 23       JSR     GETHEX
175 22DA E6 1F          INC     CHKSUM
176 22DC D0 CC          BNE     LODERR
```

```
177 22DE 4C 2E 22          JMP     LOADER
178 22E1 20 0B 23 REC4     JSR     GETHEX
179 22E4 0A                ASL             A
180 22E5 A8                TAY
181 22E6 A5 1C             LDA     ADDR
182 22E8 99 00 02          STA     PRCBUF,Y
183 22EB C8                INY
184 22EC A5 1D             LDA     ADDR+1
185 22EE 99 00 02          STA     PRCBUF,Y
186 22F1 A0 00             LDY     #$00    ;8 CHAR FROM NAME TO SAVE IN NAME
187 22F3 20 0D 20 REC4X1   JSR     RDHSR   ;GET A CHAR
188 22F6 99 30 00          STA     NAME,Y
189 22F9 20 21 23          JSR     ADCHKS  ;ADD TO CHECKSUM
190 22FC C8                INY
191 22FD C0 08             CPY     #$08
192 22FF D0 F2             BNE     REC4X1
193 2301 20 0B 23          JSR     GETHEX  ;GET CHECKSUM
194 2304 E6 1F             INC     CHKSUM  ;FINALIZE CHKSUM
195 2306 D0 A2             BNE     LODERR  ;ERROR EXIT
196 2308 4C 2E 22          JMP     LOADER
197 230B 20 0D 20 GETHEX   JSR     RDHSR   ;GET A CHAR
198 230E 20 70 20          JSR     HEXIT
199 2311 0A                ASL             A
200 2312 0A                ASL             A
201 2313 0A                ASL             A
202 2314 0A                ASL             A
203 2315 85 1E             STA     ASSEM
204 2317 20 0D 20          JSR     RDHSR
205 231A 20 70 20          JSR     HEXIT
206 231D 05 1E             ORA     ASSEM
207 231F 85 1E             STA     ASSEM   ;SAVE IT
208 2321 18        ADCHKS  CLC
209 2322 65 1F             ADC     CHKSUM
210 2324 85 1F             STA     CHKSUM
211 2326 A5 1E             LDA     ASSEM
212 2328 60                RTS
213                ;
214                ;* HERE STARTS THE PASCALM MICRO CODE INTEPRETER
215                ;
216                ;TEST IF STACK WILL OVERFLOW IN NEAR FUTURE
217                ;
218 2329 38        TSTSTK  SEC             ;TEST IF STACK AND HEAP WILL MEET IN N
219 232A A5 20             LDA     STACK
220 232C E5 02             SBC     HP
221 232E 85 16             STA     TMP2L2
222 2330 A5 21             LDA     STACK+1
223 2332 E5 03             SBC     HP+1
224 2334 B0 07             BCS     TST1X1  ;IF CARRY SET IT OVERFLOWED ALREADY
225 2336 A9 20     OVFL    LDA     #>MSG005
  1 2338 A2 ED             LDX     #<MSG005        ;SEND ERROR MSG AND GO HOME
  2 233A 4C 95 20          JMP     STRING
  3 233D D0 06     TST1X1  BNE     LOOP
  4 233F A5 16             LDA     TMP2L2
  5 2341 C9 30             CMP     #$30
  6 2343 90 F1             BCC     OVFL
  7                ;
  8                ;
  9                ; --- MAIN LOOP STARTS HERE
 10 2345 A2 00     LOOP    LDX     #0
 11 2347 A1 00             LDA     (PC,X)
```

```
12 2349 85 1B           STA     OP
13 234B E6 00           INC     PC
14 234D D0 02           BNE     LOOP1
15 234F E6 01           INC     PC+1
16 2351 C9 C3    LOOP1  CMP     #$C3
17 2353 90 07           BCC     LOOP2   ;IF IN RANGE
18 2355 A9 20           LDA     #>MSG003
19 2357 A2 CE           LDX     #<MSG003
20 2359 4C 95 20        JMP     STRING
21 235C C9 90    LOOP2  CMP     #$90    ;TEST IF DIRECT OP
22 235E B0 14           BCS     PACOP   ;PACKED OPCODE
23 2360 4A              LSR             A
24 2361 4A              LSR             A
25 2362 4A              LSR             A
26 2363 29 1E           AND     #$1E
27 2365 A8              TAY             ;USE Y FOR INDEX
28 2366 B9 08 22        LDA     JTAB2,Y
29 2369 85 08           STA     JTABP
   236B C8              INY
31 236C B9 08 22        LDA     JTAB2,Y
32 236F 85 09           STA     JTABP+1
33 2371 6C 08 00        JMP     (JTABP) ;ENTER ROUTINE
34 2374 E9 90    PACOP  SBC     #$90    ;CARRY WAS SET REMEMBER ?
35 2376 0A              ASL             A
36 2377 A8              TAY
37 2378 B9 9E 21        LDA     JTAB,Y
38 237B 85 08           STA     JTABP
39 237D C8              INY
40 237E B9 9E 21        LDA     JTAB,Y
41 2381 85 09           STA     JTABP+1
42 2383 6C 08 00        JMP     (JTABP) ;JUMP INDIRECT
43 2386 A9 21    NOTIMP LDA     #>MSG007
44 2388 A2 16           LDX     #<MSG007
45                      ;          SEND NOT IMPLEMENTED MESSAGE
46 238A 4C 95 20        JMP     STRING  ;AND ABORT PASCAL RUN
47                      ;
48                      ; SETUP1 AND SETUP2 ARE SUBROUTINES TO SETUP
                        ; VARIABLES FOR VARIOUS FUNCTIONS (ADD SUBTR ECT)
50                      ;
51 238D 20 7B 20 SETUP2 JSR     PULL2   ;GET THE FIRST OPERAND
52 2390 85 19           STA     TMP3L2+1
53 2392 84 18           STY     TMP3L2
54 2394 20 7B 20 SETUP1 JSR     PULL2   ;NEXT ITEM GOES INTO TMP2L2
55 2397 85 17           STA     TMP2L2+1
56 2399 84 16           STY     TMP2L2
57 239B 60              RTS
58                      ;
59                      ; NEXT ARE THE SIMPLE PASCAL ROUTINES
60                      ;
61                      ; LDCIS ( OX ) LOAD SMALL INTEGER CONSTANT
62                      ;
63 239C A4 1B    LDCIS  LDY     OP      ;LOAD SMALL SCALAR CONSTANT
64 239E A9 00           LDA     #$00    ;PUSH SINGLE BYTE OF 0 (DUMMY)
65 23A0 20 87 20        JSR     PUSH2
66 23A3 4C 45 23        JMP     LOOP    ;EXIT   <- error should be TSTSTK
                        ;
                        ; LDAS (1X YY) LOAD SHORT ADDRESS
69                      ;
70 23A6 20 89 21 LDAS   JSR     SADGET  ;LOAD SMALL ADDRESS
71 23A9 A5 17           LDA     TMP2L2+1        ;GET HIGH
```

```
 72 23AB A4 16          LDY     TMP2L2  ;AND LOW ADDRESS FOR INDIRECT USE
 73 23AD 20 87 20       JSR     PUSH2   ;Y=LOW AND A=HIGH PART
 74 23B0 4C 29 23       JMP     TSTSTK
 75                     ;
 76                     ; LDA (2X YY YY) LOAD ADDRESS
 77                     ;
 78 23B3 20 66 21 LDAD  JSR     ADDGET  ;LOAD 16 BIT ADDRESS
 79 23B6 A5 17          LDA     TMP2L2+1        ;GET HIGH PART OF INDIRECT A
 80 23B8 A4 16          LDY     TMP2L2  ;LOW PART
 81 23BA 20 87 20       JSR     PUSH2
 82 23BD 4C 29 23       JMP     TSTSTK
 83                     ;
 84                     ; MSTO ( 3X) MARK STACK WITHOUT RETURN BYTES
 85                     ;
 86 23C0 20 39 21 MSTO  JSR     BASEAD  ;GET BASE ADDRESS OF PROPER LEVEL (X
 87 23C3 A5 20          LDA     STACK   ;MOVE STACK POINTER TO MPSAVE
 88 23C5 85 06          STA     MPSAVE
 89 23C7 A5 21          LDA     STACK+1
 90 23C9 85 07          STA     MPSAVE+1
 91 23CB A5 17          LDA     TMP2L2+1        ;PUSH BASE ADDRESS
 92 23CD A4 16          LDY     TMP2L2  ;WAS SAVED BY BASEAD
 93 23CF 20 87 20       JSR     PUSH2
 94 23D2 A5 05          LDA     MP+1    ;SAVE MARK POINTER
 95 23D4 A4 04          LDY     MP
 96 23D6 20 87 20       JSR     PUSH2
 97 23D9 A9 00          LDA     #$00    ;PUSH DUMMY FOR P COUNTER SAVE
 98 23DB A8             TAY
 99 23DC 20 87 20       JSR     PUSH2
100 23DF 4C 29 23       JMP     TSTSTK  ;TEST FOR OVERFLOW
101                     ;
102                     ; MSTN (4X YY) MARK STACK WITH RETURN BYTES FOR FUNCTION
103                     ;
104 23E2 A0 00     MSTN LDY     #$00
105 23E4 B1 00          LDA     (PC),Y  ;GET OPERAND (NO OF BYTES TO FILL)
106 23E6 A8             TAY             ;SAVE IT
107 23E7 A2 00          LDX     #PC     ;INCREMENT PC
108 23E9 20 3A 20       JSR     INCR
109 23EC 8A        MSTN1 TXA            ;GET A ZERO IN A REG
110 23ED 20 8D 20       JSR     PUSH    ;PUSH A DUMMY ONTO STACK
111 23F0 88             DEY
112 23F1 D0 F9          BNE     MSTN1   ;NOT DONE YET
113 23F3 4C C0 23       JMP     MSTO    ;NOW MARK THE STACK AS NORMAL
114                     ;
115                     ; LOD1 (5X YY) LOAD 1 BYTE DATA ITEM ONTO STACK
116                     ;
117 23F6 20 89 21 LOD1  JSR     SADGET  ;GET ITEM ADDRESS IN RESERVED STACK A
118 23F9 A2 00          LDX     #$00
119 23FB A1 16          LDA     (TMP2L2,X)      ;LOAD ITEM
120 23FD A8             TAY             ;IT IS THE LOW PART
121 23FE 8A             TXA             ;X WAS STILL ZERO SO PUSH DUMMY
122 23FF 20 87 20       JSR     PUSH2   ;ALL ITEMS ON STACK ARE 16 BIT REMEMI
123 2402 4C 29 23       JMP     TSTSTK
124                     ;
125                     ; LOD2 (6X YY) LOAD 2 BYTE DATA ITEM
126                     ;
127 2405 20 89 21 LOD2  JSR     SADGET  ;GET ADDRESS IN TMP2L2 OF ITEM
128 2408 A2 00          LDX     #$00    ;INDEX = 0
129 240A A1 16          LDA     (TMP2L2,X)      ;LOW PART OF ITEM
130 240C A8             TAY             ;Y NOT USED IN NEXT ROUTINES SO SAVE
131 240D A2 16          LDX     #TMP2L2 ;NOW POINT TO HIGH ADDRESS OF ITEM
```

```
                        ;
                        ; STR1 (7X YY) STORE 1 BYTE DATA ITEM
                        ;
241A 20 89 21 STR1      JSR     SADGET
241D 20 7B 20           JSR     PULL2   ;GET 16 BIT WITH 8 VALID BIT ITEM
2420 98                 TYA             ;LOW PART TO A ( X IS STILL ZERO )
2421 81 16              STA     (TMP2L2,X)      ;STORE IN MEMORY
2423 4C 45 23           JMP     LOOP
                        ;
                        ; STR2 (8X YY) STORE 2 BYTE DATA ITEM IN MEMORY
                        ;
2426 20 89 21 STR2      JSR     SADGET  ;STORE 16 BITS IN RESERVED STACK AREA
2429 20 7B 20           JSR     PULL2   ;HIGH PART OF ITEM IN A REG
242C 48                 PHA             ;SAVE A ON STACK
242D 98                 TYA             ;STORE LOW PART FIRST
242E 81 16              STA     (TMP2L2,X)
2430 A2 16              LDX     #TMP2L2 ;DECREMENT ADDRESS IN STACK
2432 20 3A 20           JSR     INCR
2435 68                 PLA             ;NOW SAVE HIGH PART OF ITEM
2436 81 16              STA     (TMP2L2,X)      ;STORE HIGH PART OF 16 BIT IT
2438 4C 45 23           JMP     LOOP
                        ;
                        .FILE   FILE2


                        ;
                        ; NEXT ARE THE INDIRECT ROUTINES
                        ; 90 LEQ2
243B 20 00 25 LEQ2      JSR     CMP2    ;COMPARE NEXT TO TOP WITH TOP
243E D0 04              BNE     LEQ2X1  ;FALSE
2440 A0 01    LEQ2X0    LDY     #$01    ;TRUE
2442 D0 02              BNE     LEQ2X2  ;ALWAYS BRANCH
2444 A0 00    LEQ2X1    LDY     #$00    ;FALSE
2446 A9 00    LEQ2X2    LDA     #$00    ;DUMMY
2448 20 87 20           JSR     PUSH2   ;PUSH BOOLEAN ONTO STACK
244B 4C 45 23           JMP     LOOP    ;NEXT ONE PLEASE
                        ; 91 FOR
244E A0 01    FOR       LDY     #$01    ;POINT TO FIRST CELL ON STACK
2450 B1 20              LDA     (STACK),Y       ;TEST FOR INIT LOOP CELL
2452 30 0A              BMI     FOR1    ;WE HAVE BEEN HERE BEFORE
2454 09 80              ORA     #$80    ;SET 'BEEN HERE' MEANING LOOP INITIAL
2456 91 20              STA     (STACK),Y
2458 20 D5 2C           JSR     FORIN   ;INIT LOOP CELL
245B 4C A0 24           JMP     FOR5    ;NOW PROCESS BECAUSE SETUP IS DONE AL
245E A0 07    FOR1      LDY     #$07    ;GET ADDRESS OF CELL IN TMP2
2460 B1 20              LDA     (STACK),Y
2462 85 16              STA     TMP2L2
2464 C8                 INY
2465 B1 20              LDA     (STACK),Y
2467 85 17              STA     TMP2L2+1
2469 A0 00              LDY     #$00    ;NOW GET THE LOOP COUNT IN TMP3
246B B1 16              LDA     (TMP2L2),Y
246D 85 18              STA     TMP3L2
246F C8                 INY
2470 B1 20              LDA     (STACK),Y       ;TEST FOR 1 OR 2 BYTES
2472 29 01              AND     #$01
```

```
172 2474 85 19          STA     TMP3L2+1        ;INIT TMP3L2+1 TO ZERO FOR
193 2476 F0 04          BEQ     FOR2
194 2478 B1 16          LDA     (TMP2L2),Y      ;GET SECOND BYTE
195 247A 85 19          STA     TMP3L2+1
196 247C B1 20    FOR2  LDA     (STACK),Y       ;Y IS STILL 1
197 247E 29 04          AND     #$04    ;TO OR DOWNTO TEST
198 2480 D0 08          BNE     FOR3    ;FOR DOWN LOOP
199 2482 A2 18          LDX     #TMP3L2
200 2484 20 3A 20       JSR     INCR    ;16 BIT INCREMENT BY 1
201 2487 4C 8F 24       JMP     FOR4
202 248A A2 18    FOR3  LDX     #TMP3L2
203 248C 20 4C 20       JSR     DECR    ;16 BIT DECREMENT BY 1
204 248F A0 00    FOR4  LDY     #$00    ;SAVE NEW LOOP COUNT NUMBER
205 2491 A5 18          LDA     TMP3L2
206 2493 91 16          STA     (TMP2L2),Y
207 2495 C8             INY
208 2496 B1 20          LDA     (STACK),Y
209 2498 29 01          AND     #$01    ;1 OR 2 BYTES TO SAVE
2   249A F0 04          BEQ     FOR5    ;IF 1 BYTE GO PROCESS
211 249C A5 19          LDA     TMP3L2+1
212 249E 91 16          STA     (TMP2L2),Y      ;SAVE SECOND BYTE
213 24A0 A0 01    FOR5  LDY     #$01
214 24A2 B1 20          LDA     (STACK),Y
215 24A4 29 04          AND     #$04    ;TEST TO OR DOWNTO
216 24A6 D0 1C          BNE     FOR6    ;IF DOWNTO LOOP
217 24A8 38             SEC             ;TO LOOP
218 24A9 A0 03          LDY     #$03    ;POINT TO END VALUE
219 24AB B1 20          LDA     (STACK),Y
220 24AD E5 18          SBC     TMP3L2
221 24AF C8             INY
222 24B0 B1 20          LDA     (STACK),Y       ;DO ALWAYS 16 BIT TEST TO BE
223 24B2 E5 19          SBC     TMP3L2+1
224 24B4 30 1C          BMI     FOREND
225 24B6 A0 00    FORLP LDY     #$00
226 24B8 B1 00          LDA     (PC),Y  ;GET JUMP ADDRESS IN TMP2
227 24BA 85 17          STA     TMP2L2+1
228 24BC C8             INY
2   24BD B1 00          LDA     (PC),Y
230 24BF 85 16          STA     TMP2L2
231 24C1 4C CD 28       JMP     CAS2    ;USE COMMON JUMP END
232 24C4 38       FOR6  SEC             ;DOWNTO
233 24C5 A0 03          LDY     #$03
234 24C7 A5 18          LDA     TMP3L2
235 24C9 F1 20          SBC     (STACK),Y
236 24CB C8             INY
237 24CC A5 19          LDA     TMP3L2+1        ;IT IS ALL THE SAME AS TO LC
238 24CE F1 20          SBC     (STACK),Y
239 24D0 10 E4          BPL     FORLP
240 24D2 18       FOREND CLC
241 24D3 A5 20          LDA     STACK
  1 24D5 69 08          ADC     #$08
  2 24D7 85 20          STA     STACK
  3 24D9 A5 21          LDA     STACK+1
  4 24DB 69 00          ADC     #$00
  5 24DD 85 21          STA     STACK+1 ;REMOVE FOR TABLE
  ' 24DF 20 5E 20       JSR     INCPC2  ;INCREMENT PC BY 2
    24E2 4C 45 23       JMP     LOOP
  8               ; 92 LEQM
  9 24E5 20 9F 2D LEQM  JSR     COMPAR
 10 24E8 A9 00          LDA     #$00
```

```
11 24EA A4 0E          LDY   TMP1L8
12 24EC 88             DEY
13 24ED 10 08          BPL   LEQM2
14 24EF A0 01          LDY   #$01
 1 24F1 20 87 20 LEQM1 JSR   PUSH2
 2 24F4 4C 29 23       JMP   TSTSTK
 3 24F7 A8       LEQM2 TAY
 4 24F8 F0 F7          BEQ   LEQM1
 5                ; 93 LES2
 6 24FA 20 00 25 LES2  JSR   CMP2    ;SAME AS LEQ2 WITHOUT EQUAL
 7 24FD 4C 44 24       JMP   LEQ2X1  ;MINUS TEST SHOULD HAVE CHATCHED
 8                ;
 9 2500 20 8D 23 CMP2  JSR   SETUP2  ;FIRST OP IN TMP3 SECOND IN TMP2
10 2503 38             SEC          ;SET CARRY FOR SUBTRACT
11 2504 A5 16          LDA   TMP2L2
12 2506 E5 18          SBC   TMP3L2
13 2508 85 16          STA   TMP2L2  ;SAVE FOR EQUAL TEST
14 250A A5 17          LDA   TMP2L2+1
15 250C E5 19          SBC   TMP3L2+1
16 250E 30 05          BMI   CMP2A   ;IF LESS THAN (15 BITS... )
17 2510 D0 08          BNE   CMP2B   ;NOT EQUAL HIGH PART
18 2512 A5 16          LDA   TMP2L2  ;POSIBLE HIGH PART EQUAL
19 2514 60             RTS
20 2515 68       CMP2A PLA           ;FULL RETURN ADDRESS
21 2516 68             PLA
22 2517 4C 40 24       JMP   LEQ2X0
23 251A 68       CMP2B PLA
24 251B 68             PLA
25 251C 4C 44 24       JMP   LEQ2X1
26                ; 94 LEQ8
27 251F A0 08    LEQ8  LDY   #$08    ;NUMBER OF SET WORDS AND OFFSET IN S
28 2521 A2 00          LDX   #$00
29 2523 84 16          STY   TMP2L2  ;TMP2 IS THE COUNT
30 2525 86 18          STX   TMP3L2  ;TEST FOR END RESULT
31 2527 20 7F 20 LEQ8A JSR   PULL    ;GET A MEMBER
32 252A 31 20          AND   (STACK),Y      ;DELETE NOT INTERESTED BITS
33 252C 51 20          EOR   (STACK),Y      ;FLIP REMAINING BITS
34 252E F0 02          BEQ   LEQ8B   ;IF ALL BITS FLIPPED
35 2530 E6 18          INC   TMP3L2  ;NOT ALL BITS FLIPPED NOT SAME,
36 2532 C6 16    LEQ8B DEC   TMP2L2  ;WE DID THE LOOP (AGAIN)
37 2534 D0 F1          BNE   LEQ8A   ;IF NOT 8 TIMES REPEAT
38 2536 20 7B 20       JSR   PULL2   ;GET SECOND SET FROM STACK
39 2539 20 7B 20       JSR   PULL2
40 253C 20 7B 20       JSR   PULL2
41 253F 20 7B 20       JSR   PULL2
42 2542 A5 18          LDA   TMP3L2
43 2544 D0 03          BNE   LEQ8C
44 2546 4C 40 24       JMP   LEQ2X0  ;SET WAS CONTAINED IN
45 2549 4C 44 24 LEQ8C JMP   LEQ2X1  ;NOT CONTAINED IN
46                ; 95 LESM
47 254C 20 9F 2D LESM  JSR   COMPAR
48 254F A9 00          LDA   #$00
49 2551 A4 0E          LDY   TMP1L8
50 2553 10 08          BPL   LESM2
51 2555 A0 01          LDY   #$01
52 2557 20 87 20 LESM1 JSR   PUSH2
53 255A 4C 29 23       JMP   TSTSTK
54 255D A8       LESM2 TAY
55 255E F0 F7          BEQ   LESM1
56                ; 96 EQU2
```

```
2560 20 8D 23 EQU2   JSR     SETUP2  ;COMPARE TMP2 AND TMP3
2563 A5 18           LDA     TMP3L2  ;TEST LOWER PART
2565 C5 16           CMP     TMP2L2
2567 D0 09           BNE     EQU2X1
2569 A5 19           LDA     TMP3L2+1
256B C5 17           CMP     TMP2L2+1
256D D0 03           BNE     EQU2X1  ;NOT EQUAL
256F 4C 40 24        JMP     LEQ2X0  ;PUSH 1
2572 4C 44 24 EQU2X1 JMP     LEQ2X1
                     ; 97 GEQ8
2575 A0 08    GEQ8   LDY     #$08    ;OFFSET IN STACK AND COUNT
2577 A2 00           LDX     #$00
2579 84 16           STY     TMP2L2  ;READ FOR FURTER COMMENT ISTR 94
257B 86 18           STX     TMP3L2
257D 20 7F 20 GEQ8A  JSR     PULL
2580 85 17           STA     TMP2L2+1
2582 B1 20           LDA     (STACK),Y       ;GET MEMBER TO TEST
2584 25 17           AND     TMP2L2+1
2586 45 17           EOR     TMP2L2+1
2588 F0 02           BEQ     GEQ8B
258A E6 18           INC     TMP3L2
258C C6 16    GEQ8B  DEC     TMP2L2
258E D0 ED           BNE     GEQ8A
2590 20 7B 20        JSR     PULL2   ;CLEAN UP STACK
2593 20 7B 20        JSR     PULL2
2596 20 7B 20        JSR     PULL2
2599 20 7B 20        JSR     PULL2
259C A5 18           LDA     TMP3L2
259E D0 03           BNE     GEQ8C
25A0 4C 44 24        JMP     LEQ2X1  ;WARNING DUE TO NOT INSTR FOLLOWING
25A3 4C 40 24 GEQ8C  JMP     LEQ2X0  ;SEND FALSE IF TRUE .SORRY.
                     ; 98 EQUM
25A6 20 9F 2D EQUM   JSR     COMPAR
25A9 A9 00           LDA     #$00
25AB A4 0E           LDY     TMP1L8
25AD D0 07           BNE     EQUM2
25AF C8              INY
25B0 20 87 20 EQUM1  JSR     PUSH2
25B3 4C 29 23        JMP     TSTSTK
25B6 A8       EQUM2  TAY
25B7 F0 F7           BEQ     EQUM1
                     ; 99 EQU8
25B9 A0 08    EQU8   LDY     #$08    ;8 BYTES TO COMPARE
25BB 84 16           STY     TMP2L2  ;USE AS COUNT ALSO
25BD A2 00           LDX     #$00
25BF 86 17           STX     TMP2L2+1        ;USE AS SWITCH ZERO ON END IS
25C1 20 7F 20 EQU8X1 JSR     PULL    ;GET A CHAR IN A TO COMPARE
25C4 D1 20           CMP     (STACK),Y       ;COMPARE IN STACK
25C6 F0 02           BEQ     EQU8X2  ;IF EQUAL
25C8 E6 17           INC     TMP2L2+1        ;NOT EQUAL SET SWITCH
25CA C6 16    EQU8X2 DEC     TMP2L2
25CC D0 F3           BNE     EQU8X1
25CE 20 7B 20        JSR     PULL2   ;CLEAN UP STACK
25D1 20 7B 20        JSR     PULL2
25D4 20 7B 20        JSR     PULL2
25D7 20 7B 20        JSR     PULL2
25DA A5 17           LDA     TMP2L2+1
25DC D0 03           BNE     EQU8X3  ;IF NOT EQUAL
25DE 4C 40 24        JMP     LEQ2X0  ;SEND 1
25E1 4C 44 24 EQU8X3 JMP     LEQ2X1  ;SEND 0
```

```
                        ; 9A IND1
25E4 20 94 23 IND1      JSR     SETUP1  ;GET INDIRECT ADDRESS FROM STACK
25E7 A0 00              LDY     #$00    ;INDEX
25E9 B1 16              LDA     (TMP2L2),Y      ;GET 1 BYTE OPERAND
25EB A8                 TAY
25EC A9 00              LDA     #$00    DUMMY
25EE 20 87 20           JSR     PUSH2
25F1 4C 45 23           JMP     LOOP
                        ; 9B IND2
25F4 20 94 23 IND2      JSR     SETUP1  ;GET ADDRESS TO 16 BYTE INDERECT OP
25F7 A2 00              LDX     #$00
25F9 A1 16              LDA     (TMP2L2,X)
25FB A8                 TAY             ;SETUP FOR PUSH
25FC A2 16              LDX     #TMP2L2 ;INCREMENT POINTER
25FE 20 3A 20           JSR     INCR    ;POINT TO UPPER PART
2601 A1 16              LDA     (TMP2L2,X)
2603 20 87 20           JSR     PUSH2   ;PUSH ITEM ONTO STACK
2606 4C 45 23           JMP     LOOP
                        ; 9C IND8
2609 20 94 23 IND8      JSR     SETUP1  ;GET INDERECT ADDRESS
260C A0 07    IND8X1    LDY     #$07
260E B1 16    IND8X2    LDA     (TMP2L2),Y      ;LINEAR MOVE
2610 20 8D 20           JSR     PUSH
2613 88                 DEY
2614 10 F8              BPL     IND8X2  ;REPEAT UNTILL 8 WORDS DONE
2616 4C 29 23           JMP     TSTSTK
                        ; 9D STO1
2619 20 8D 23 STO1      JSR     SETUP2
261C A0 00              LDY     #$00
261E A5 18              LDA     TMP3L2  ;OP TO STORE
2620 91 16              STA     (TMP2L2),Y
2622 4C 45 23           JMP     LOOP
                        ; 9E STO2
2625 20 8D 23 STO2      JSR     SETUP2  ;GET 16 BIT VALUE IN TMP3 AND ADDRESS
2628 A2 00              LDX     #$00
262A A5 18              LDA     TMP3L2
262C 81 16              STA     (TMP2L2,X)
262E A2 16              LDX     #TMP2L2
2630 20 3A 20           JSR     INCR
2633 A5 19              LDA     TMP3L2+1        ;UPPER HALF
2635 81 16              STA     (TMP2L2,X)
2637 4C 45 23           JMP     LOOP
                        ; 9F STO8
263A A0 09    STO8      LDY     #$09    ;ADD 10 TO STACK POINTER TO REACH ADD
263C B1 20              LDA     (STACK),Y       ;GET LOWER PART OF ADDRESS
263E 85 16              STA     TMP2L2
2640 C8                 INY
2641 B1 20              LDA     (STACK),Y       ;GET UPPER PART OF ADDRESS
2643 85 17              STA     TMP2L2+1
2645 A0 00              LDY     #$00
2647 20 7F 20 STO8X1    JSR     PULL    ;GET ITEM FROM STACK
264A 91 16              STA     (TMP2L2),Y      ;LINEAR MOVE
264C C8                 INY
264D C0 08              CPY     #$08    ;DO 8 WORDS
264F D0 F6              BNE     STO8X1
2651 20 7B 20           JSR     PULL2   ;AND FOR FINAL GET ADDRESS FROM STACK
2654 4C 45 23           JMP     LOOP
                        ; A0 LDC
2657 A1 00    LDC       LDA     (PC,X)  ;LOAD 16 BIT CONSTANT X WAS ZERO FROM
2659 48                 PHA             ;THIS WAS THE HIGH PART
```

```
177 265A E6 00          INC     PC      ;SPEED REASON DO IT YOURSELF
178 265C D0 02          BNE     LDC1
179 265E E6 01          INC     PC+1
180 2660 A1 00    LDC1  LDA     (PC,X)  ;GET LOW PART
181 2662 A8             TAY             ;READY FOR PUSH
182 2663 E6 00          INC     PC
183 2665 D0 02          BNE     LDC2
184 2667 E6 01          INC     PC+1
185 2669 68       LDC2  PLA             ;GET HIGH PART BACK
186 266A 20 87 20       JSR     PUSH2   ;AND FOR FINAL PUSH ONTO STACK
187 266D 4C 29 23       JMP     TSTSTK
188                 ; A1 RETP
189 2670 A5 04    RETP  LDA     MP      ;RESET STACK POINTER
190 2672 38             SEC
191 2673 E9 06          SBC     #$06
192 2675 85 20          STA     STACK
193 2677 A5 05          LDA     MP+1
194 2679 E9 00          SBC     #$00
195                 ; SETUP STACK POINTER 6 BYTES AWAY FROM MP
196 267B 85 21          STA     STACK+1
197 267D 20 7B 20       JSR     PULL2   ;GET PC FROM STACK
198 2680 84 00          STY     PC
199 2682 85 01          STA     PC+1
200 2684 20 7B 20       JSR     PULL2   ;NOW GET MARK POINTER
201 2687 84 04          STY     MP
202 2689 85 05          STA     MP+1                    base address
203 268B 20 7B 20       JSR     PULL2   ;NOW GET DUMMY MP FROM STACK
204 268E 4C 45 23       JMP     LOOP    ;RETURN IS DONE
205                 ; A2 ADI
206 2691 20 8D 23 ADI   JSR     SETUP2  ;ADD TWO 16 BIT NUMBERS
207 2694 18             CLC
208 2695 A5 16          LDA     TMP2L2
209 2697 65 18          ADC     TMP3L2  ;ADD LOW PART
210 2699 A8             TAY             ;RESULT TO Y REG
211 269A A5 17          LDA     TMP2L2+1
212 269C 65 19          ADC     TMP3L2+1
213 269E 20 87 20       JSR     PUSH2
214 26A1 4C 45 23       JMP     LOOP
215                 ; A3 ANDB
216 26A4 20 8D 23 ANDB  JSR     SETUP2  ;AND FOR BOOLEAN
217 26A7 A5 18          LDA     TMP3L2
218 26A9 25 16          AND     TMP2L2
219 26AB A8             TAY             ;RESULT TO Y REG
220 26AC A9 00          LDA     #$00    ;DUMMY
221 26AE 20 87 20       JSR     PUSH2
222 26B1 4C 45 23       JMP     LOOP
223                 ; A4 DIF
224 26B4 A0 08    DIF   LDY     #$08
225 26B6 84 16          STY     TMP2L2  ;FOR COUNTER
226 26B8 20 7F 20 DIF1  JSR     PULL
227 26BB 31 20          AND     (STACK),Y
228 26BD 51 20          EOR     (STACK),Y
229 26BF 91 20          STA     (STACK),Y
230 26C1 C6 16          DEC     TMP2L2
231 26C3 D0 F3          BNE     DIF1
232 26C5 4C 45 23       JMP     LOOP
233                 ; A5 DVI
234 26C8 20 19 2D DVI   JSR     MDSET   ;SETUP FOR DIVIDE
235 26CB 20 5C 2D       JSR     DIVIDE
236 26CE A5 14          LDA     TMP1L8+6        ;GET SIGN OF RESULT
```

```
237 26D0 29 01           AND     #$01
238 26D2 F0 11           BEQ     DVI1    ;NOT TO BE CORRECTED
239 26D4 A5 12           LDA     TMP1L8+4
240 26D6 49 FF           EOR     #$FF
241 26D8 85 12           STA     TMP1L8+4
242 26DA A5 13           LDA     TMP1L8+5
243 26DC 49 FF           EOR     #$FF
244 26DE 85 13           STA     TMP1L8+5
245 26E0 A2 12           LDX     #TMP1L8+4
246 26E2 20 3A 20        JSR     INCR
247 26E5 A4 12    DVI1   LDY     TMP1L8+4
248 26E7 A5 13           LDA     TMP1L8+5
249 26E9 20 87 20        JSR     PUSH2
250 26EC 4C 45 23        JMP     LOOP
251                ; A6 INN
252 26EF A0 00    INN    LDY     #$00    ;GET THE SET FROM THE STACK
253 26F1 20 7F 20 INN1   JSR     PULL
254 26F4 99 0E 00        STA     TMP1L8,Y
255 26F7 C8              INY
256 26F8 C0 08           CPY     #$08    ;SETS ARE 8 BYTES LONG
257 26FA D0 F5           BNE     INN1
258 26FC 20 7B 20        JSR     PULL2   ;NOW GET THE BIT NO=testbit
259 26FF 98              TYA
260 2700 29 07           AND     #$07    ;BIT IN WORD  _bitnr := testbit mod 8
261 2702 AA              TAX
  1 2703 98              TYA                     bytenr := testbit div 8
  2 2704 4A              LSR             A
  3 2705 4A              LSR             A
  4 2706 4A              LSR             A
  5 2707 A8              TAY             ;WORD NO (OFFSET IN TMP1L8)
  6 2708 A9 00           LDA     #$00    } repeat
  7 270A 38              SEC             }
  8 270B 6A        INN2  ROR             A } shift bit in A
  9 270C CA              DEX             } bitnr := bitnr - 1
 10 270D 10 FC           BPL     INN2    } until bitnr < 0
 11 270F 39 0E 00        AND     TMP1L8,Y        ;TEST THE BIT
 12 2712 F0 04           BEQ     INN3    ;IF NOT SET
 13 2714 A0 01           LDY     #$01    ;SEND TRUE
 14 2716 D0 02           BNE     INN4
 15 2718 A0 00    INN3   LDY     #$00    ;FALSE
 16 271A A9 00    INN4   LDA     #$00    ;DUMMY FOR 16 BITS
 17 271C 20 87 20        JSR     PUSH2
 18 271F 4C 45 23        JMP     LOOP
 19                ; A7 INT
 20 2722 A0 08    INT    LDY     #$08
 21 2724 84 16           STY     TMP2L2
 22 2726 20 7F 20 INT1   JSR     PULL
 23 2729 31 20           AND     (STACK),Y
 24 272B 91 20           STA     (STACK),Y
 25 272D C6 16           DEC     TMP2L2
 26 272F D0 F5           BNE     INT1
 27 2731 4C 45 23        JMP     LOOP
 28                ; A8 IOR
 29 2734 20 94 23 IOR    JSR     SETUP1
 30 2737 A0 01           LDY     #$01
 31 2739 B1 20           LDA     (STACK),Y
 32 273B 05 16           ORA     TMP2L2
 33 273D 91 20           STA     (STACK),Y
 34 273F 4C 45 23        JMP     LOOP
 35                ; A9 MOD
```

```
36 2742 20 19 2D MOD     JSR     MDSET
37 2745 20 5C 2D         JSR     DIVIDE
38 2748 A5 14            LDA     TMP1L8+6
39 274A 29 01            AND     #$01
40 274C F0 11            BEQ     MOD1
41 274E A5 10            LDA     TMP1L8+2
42 2750 49 FF            EOR     #$FF
43 2752 85 10            STA     TMP1L8+2
44 2754 A5 11            LDA     TMP1L8+3
45 2756 49 FF            EOR     #$FF
46 2758 85 11            STA     TMP1L8+3
47 275A A2 10            LDX     #TMP1L8+2
48 275C 20 3A 20         JSR     INCR
49 275F A4 10     MOD1   LDY     TMP1L8+2
50 2761 A5 11            LDA     TMP1L8+3
51 2763 20 87 20         JSR     PUSH2
52 2766 4C 45 23         JMP     LOOP
53                    ; AA MPI
5  2769 20 19 2D MPI     JSR     MDSET
55 276C 20 FC 2C         JSR     MPLY
56 276F A5 14            LDA     TMP1L8+6
57 2771 29 01            AND     #$01
58 2773 F0 11            BEQ     MPI1
59 2775 A5 12            LDA     TMP1L8+4
60 2777 49 FF            EOR     #$FF
61 2779 85 12            STA     TMP1L8+4
62 277B A5 13            LDA     TMP1L8+5
63 277D 49 FF            EOR     #$FF
64 277F 85 13            STA     TMP1L8+5
65 2781 A2 12            LDX     #TMP1L8+4
66 2783 20 3A 20         JSR     INCR
67 2786 A4 12     MPI1   LDY     TMP1L8+4
68 2788 A5 13            LDA     TMP1L8+5
69 278A 20 87 20         JSR     PUSH2
70 278D 4C 45 23         JMP     LOOP
71                    ; AB NGI
72 2790 20 94 23 NGI     JSR     SETUP1  ;GET NUMBER TO COMPLEMENT
7  2793 A5 16            LDA     TMP2L2
74 2795 49 FF            EOR     #$FF
75 2797 85 16            STA     TMP2L2
76 2799 A5 17            LDA     TMP2L2+1
77 279B 49 FF            EOR     #$FF
78 279D 85 17            STA     TMP2L2+1
79 279F A2 16            LDX     #TMP2L2
80 27A1 20 3A 20         JSR     INCR    ;FORM ONES COMPLEMENT
81 27A4 A4 16            LDY     TMP2L2
82 27A6 A5 17            LDA     TMP2L2+1
83 27A8 20 87 20         JSR     PUSH2
84 27AB 4C 45 23         JMP     LOOP
85                    ; AC NOT
86 27AE A0 01     NOT    LDY     #$01    ;COMPLEMENT BOOLEAN ON STACK
87 27B0 B1 20            LDA     (STACK),Y
88 27B2 49 01            EOR     #$01    ;ONLY BIT 0 TO DO
89 27B4 91 20            STA     (STACK),Y
90 27B6 4C 45 23         JMP     LOOP
91                    ; AD SBI
9  27B9 20 8D 23 SBI     JSR     SETUP2  ;SUBRTRACT TWO 16 BIT NUMBERS ON STAC
93 27BC 38               SEC             ;SET CARRY FOR SUBTRACT WORK
94 27BD A5 16            LDA     TMP2L2
95 27BF E5 18            SBC     TMP3L2
```

```
 96 27C1 A8                TAY             ;RESULT LOW PART TO Y
 97 27C2 A5 17             LDA     TMP2L2+1
 98 27C4 E5 19             SBC     TMP3L2+1
 99 27C6 20 87 20          JSR     PUSH2   ;RESULT OK
100 27C9 4C 45 23          JMP     LOOP
101                   ; AE SGS
102 27CC A2 00        SGS   LDX     #$00    ;GENERATE SINGLE BIT SET
103 27CE A0 07              LDY     #$07    ;CLEAR TMP1L8
104 27D0 96 0E        SGS1  STX     TMP1L8,Y
105 27D2 88                 DEY
106 27D3 10 FB              BPL     SGS1
107 27D5 20 7B 20           JSR     PULL2   ;GET BIT NUMBER OF SET
108 27D8 98                 TYA
109 27D9 29 07              AND     #$07    ;GET BIT NO
110 27DB AA                 TAX
111 27DC 98                 TYA
112 27DD 4A                 LSR             A
113 27DE 4A                 LSR             A
114 27DF 4A                 LSR             A
115 27E0 A8                 TAY             ;BYTE NO IN Y REG
116 27E1 A9 00              LDA     #$00
117 27E3 38                 SEC
118 27E4 6A           SGS2  ROR             A ;SHIFT THE BIT
119 27E5 CA                 DEX
120 27E6 10 FC              BPL     SGS2
121 27E8 99 0E 00           STA     TMP1L8,Y        ;NOW BIT IN WORD IS FORMED
122 27EB A0 07              LDY     #$07    ;NOW PUSH SET ONTO STACK
123 27ED A2 00              LDX     #$00    ;PUSH WANTS X = 00
124 27EF B9 0E 00 SGS3  LDA     TMP1L8,Y        ;HIGH BITS FIRST
125 27F2 20 8D 20           JSR     PUSH
126 27F5 88                 DEY
127 27F6 10 F7              BPL     SGS3
128 27F8 4C 29 23           JMP     TSTSTK
129                   ; AF UNI
130 27FB A0 08        UNI   LDY     #$08
131 27FD 84 16              STY     TMP2L2
132 27FF 20 7F 20 UNI1  JSR     PULL
133 2802 11 20              ORA     (STACK),Y
134 2804 91 20              STA     (STACK),Y
135 2806 C6 16              DEC     TMP2L2
136 2808 D0 F5              BNE     UNI1
137 280A 4C 45 23           JMP     LOOP
138                   ; B0 LNC
139 280D A0 01        LNC   LDY     #$01
140 280F B1 00              LDA     (PC),Y
141 2811 49 FF              EOR     #$FF    ;COMPLEMENT
142 2813 85 16              STA     TMP2L2
143 2815 88                 DEY
144 2816 B1 00              LDA     (PC),Y
145 2818 49 FF              EOR     #$FF
146 281A 85 17              STA     TMP2L2+1
147 281C A2 16              LDX     #TMP2L2
148 281E 20 3A 20           JSR     INCR
149 2821 A5 17              LDA     TMP2L2+1        ;GET A FOR CORRECT SETUP
150 2823 4C 52 28           JMP     DECX1   ;USE COMMON END
151                   ; B1 FJP
152 2826 20 94 23 FJP   JSR     SETUP1  ;GET BOOLEAN FROM STACK
153 2829 A5 16              LDA     TMP2L2
154 282B F0 06              BEQ     UJP     ;FALSE JUMP
155 282D 20 5E 20           JSR     INCPC2
```

```
156 2830 4C 45 23          JMP     LOOP
157                ; B2 UJP
158 2833 A0 01     UJP     LDY     #$01
159 2835 B1 00             LDA     (PC),Y
160 2837 85 16             STA     TMP2L2
161 2839 88                DEY
162 283A B1 00             LDA     (PC),Y
163 283C 85 17             STA     TMP2L2+1
164 283E 4C CD 28          JMP     CAS2
165                ; B3 DECB
166 2841 20 94 23  DECB    JSR     SETUP1
167 2844 A0 01             LDY     #$01
168 2846 38                SEC
169 2847 A5 16             LDA     TMP2L2
170 2849 F1 00             SBC     (PC),Y
171 284B 85 16             STA     TMP2L2
172 284D 88                DEY
173 284E A5 17             LDA     TMP2L2+1
174 2850 F1 00             SBC     (PC),Y
175 2852 A4 16     DECX1   LDY     TMP2L2
176 2854 20 87 20          JSR     PUSH2
177 2857 20 5E 20          JSR     INCPC2
178 285A 4C 45 23          JMP     LOOP
179                ; B4 INCB
180 285D 20 94 23  INCB    JSR     SETUP1
181 2860 A0 01             LDY     #$01
182 2862 18                CLC
183 2863 A5 16             LDA     TMP2L2
184 2865 71 00             ADC     (PC),Y
185 2867 85 16             STA     TMP2L2
186 2869 88                DEY
187 286A A5 17             LDA     TMP2L2+1
188 286C 71 00             ADC     (PC),Y
189 286E 4C 52 28          JMP     DECX1
190                ; B5 ENT
191 2871 A0 01     ENT     LDY     #$01    ;INDEX TO LOW PART OF NO OF BYTES
192 2873 A5 20             LDA     STACK   ;NO OF BYTES TO RESERVE ON STACK
193 2875 38                SEC             ;SUBTRACT SET DAMEDCARRY
194 2876 F1 00             SBC     (PC),Y  ;UPDATE (PC),Y POINTER
195 2878 85 20             STA     STACK
196 287A 88                DEY             ;POINT TO HIGH PART
197 287B A5 21             LDA     STACK+1
198 287D F1 00             SBC     (PC),Y  ;ADJUST UPPER BITS
199 287F 85 21             STA     STACK+1 ;THE WORDS ARE RESERVED NOW
200 2881 20 5E 20          JSR     INCPC2
201 2884 4C 29 23          JMP     TSTSTK  ;CHECK IF STILL ROOM ON STACK
202                ; B6 CAS
203 2887 20 94 23  CAS     JSR     SETUP1  ;GET CASE NO INTO TMP2L2
204 288A A0 01             LDY     #$01
205 288C 38                SEC
206 288D A5 16             LDA     TMP2L2  ;TEST FOR LESS THAN MINIMUM
207 288F F1 00             SBC     (PC),Y
208 2891 85 18             STA     TMP3L2  ;SAVE FOR INDEX
209 2893 88                DEY
210 2894 A5 17             LDA     TMP2L2+1
211 2896 F1 00             SBC     (PC),Y
212 2898 85 19             STA     TMP3L2+1
213 289A 30 41             BMI     CASEX   ;LESS THAN MINIMUM TEST FOR ELSE CL
214 289C A0 03             LDY     #$03    ;ELSE IS STANDARDIZED TO OTHERWISE
215 289E 38                SEC
```

```
216 289F B1 00          LDA     (PC),Y  ;TEST FOR LARGER THAN MAXIMUM
217 28A1 E5 16          SBC     TMP2L2
218 28A3 88             DEY
219 28A4 B1 00          LDA     (PC),Y
220 28A6 E5 17          SBC     TMP2L2+1
221 28A8 30 33          BMI     CASEX   ;IF LARGER THAN MAX
222 28AA 06 18          ASL     TMP3L2  ;MULT BY 2 FOR DUAL WORD INDEX
223 28AC 26 19          ROL     TMP3L2+1
224 28AE 18             CLC;IN          CASE OF
225 28AF A5 18          LDA     TMP3L2
226 28B1 65 00          ADC     PC
227 28B3 85 18          STA     TMP3L2  ;ADD INDEX TO PC TO LOOK INTO JUMP T
228 28B5 A5 19          LDA     TMP3L2+1
229 28B7 65 01          ADC     PC+1
230 28B9 85 19          STA     TMP3L2+1
231 28BB A0 07          LDY     #$07    ;OFFSET IN TABLE
232 28BD B1 18          LDA     (TMP3L2),Y
233 28BF 85 16          STA     TMP2L2  ;LOWER PART OF NEW PC
234 28C1 88             DEY
235 28C2 B1 18          LDA     (TMP3L2),Y
236 28C4 85 17          STA     TMP2L2+1
237 28C6 18             CLC
238 28C7 65 16          ADC     TMP2L2
239 28C9 D0 02          BNE     CAS2    ;IF NO NILL POINTER
240 28CB 90 10          BCC     CASEX   ;TEST IF CARY
241 28CD 18     CAS2    CLC
242 28CE A5 16          LDA     TMP2L2  ;ADJUST ADDRESS IN TMP2 WITH BASE AD
243 28D0 65 2C          ADC     BASE
244 28D2 85 00          STA     PC
245 28D4 A5 17          LDA     TMP2L2+1
246 28D6 65 2D          ADC     BASE+1
247 28D8 85 01          STA     PC+1
248 28DA 4C 45 23       JMP     LOOP    ;HELLO NEW PART
249 28DD A0 05  CASEX   LDY     #$05    ;TEST FOR OTHERWISE
250 28DF B1 00          LDA     (PC),Y
251 28E1 85 16          STA     TMP2L2
252 28E3 88             DEY
253 28E4 B1 00          LDA     (PC),Y
254 28E6 85 17          STA     TMP2L2+1
255 28E8 18             CLC
256 28E9 65 16          ADC     TMP2L2  ;TEST FOR NILL
257 28EB D0 E0          BNE     CAS2
258 28ED B0 DE          BCS     CAS2
259 28EF A9 20          LDA     #>MSG006
260 28F1 A2 FD          LDX     #<MSG006
261 28F3 4C 95 20       JMP     STRING
262             ; B7 MOV
263 28F6 20 8D 23 MOV   JSR     SETUP2  ;TMP2L2 IS TO AND TMP3L2 IS FROM
264 28F9 A0 00          LDY     #$00
265 28FB B1 00          LDA     (PC),Y  ;NUMBER OF BYTES TO MOVE
266 28FD 85 0F          STA     TMP1L8+1
267 28FF A2 00          LDX     #PC
268 2901 20 3A 20       JSR     INCR
269 2904 B1 00          LDA     (PC),Y
270 2906 85 0E          STA     TMP1L8
271 2908 A2 00          LDX     #PC
272 290A 20 3A 20       JSR     INCR    ;PC IS NOW TO NEXT INSTR.
273 290D B1 18  MOV1    LDA     (TMP3L2),Y
274 290F 91 16          STA     (TMP2L2),Y
275 2911 A2 16          LDX     #TMP2L2
```

```
276 2913 20 3A 20          JSR     INCR
277 2916 A2 18             LDX     #TMP3L2
278 2918 20 3A 20          JSR     INCR
279 291B A2 0E             LDX     #TMP1L8
280 291D 20 4C 20          JSR     DECR
281 2920 A5 0E             LDA     TMP1L8
282 2922 D0 E9             BNE     MOV1    ;NOT DONE YET
283 2924 A5 0F             LDA     TMP1L8+1
284 2926 D0 E5             BNE     MOV1
285 2928 4C 45 23          JMP     LOOP
286                   ; B8 DEC1
287 292B 20 94 23 DEC1     JSR     SETUP1  ;GET OPERAND IN TMP2L2
288 292E A2 16             LDX     #TMP2L2
289 2930 20 4C 20          JSR     DECR    ;SUBTRACT 1
290 2933 A4 16    DEC1A    LDY     TMP2L2  ;SETUP FOR PUSH
291 2935 A5 17             LDA     TMP2L2+1
292 2937 20 87 20          JSR     PUSH2   ;ON STACK
293 293A 4C 45 23          JMP     LOOP
2                 ; B9 INC1
295 293D 20 94 23 INC1     JSR     SETUP1
296 2940 A2 16             LDX     #TMP2L2
297 2942 20 3A 20          JSR     INCR    ;INCREMENT 16 BIT SCALAR BY 1
298 2945 4C 33 29          JMP     DEC1A   ;USE COMMON END
299                   ; BA LDCS
300 2948 A0 07    LDCS     LDY     #$07
301 294A B1 00    LDCS1    LDA     (PC),Y  ;GET CHAR OF CHAIN
302 294C 20 8D 20          JSR     PUSH
303 294F 88                DEY
304 2950 10 F8             BPL     LDCS1   ;IF NOT 8 DONE
305 2952 18                CLC
306 2953 A9 08             LDA     #$08    ;INCREMENT PC BY 8
307 2955 65 00             ADC     PC
308 2957 85 00             STA     PC
  1 2959 A5 01             LDA     PC+1
  2 295B 69 00             ADC     #$00
  3 295D 85 01             STA     PC+1
  4 295F 4C 29 23          JMP     TSTSTK
                    ; BB CAP
  6 2962 A2 00    CAP      LDX     #$00    ;REMEMBER THAT AT LEAST 2 BYTES ARE
  7 2964 A1 00             LDA     (PC,X)  ; FOR YOU (GIVING NUMBER OF BYTES EX
  8 2966 85 17             STA     TMP2L2+1
  9 2968 A2 00             LDX     #PC
 10 296A 20 3A 20          JSR     INCR    ;INCREMENT PROGRAM COUNTER
 11 296D A1 00             LDA     (PC,X)
 12 296F 85 16             STA     TMP2L2
 13 2971 A2 00             LDX     #PC
 14 2973 20 3A 20          JSR     INCR
 15 2976 6C 16 00          JMP     (TMP2L2)        ;HOPEFULLY THE ADDRESS IS CO
 16                   ; BC LCA
 17 2979 A0 00    LCA      LDY     #$00
 18 297B B1 00             LDA     (PC),Y  ;GET NUMBER OF BYTES (STRING LENGTH)
 19 297D 48                PHA
 20 297E A2 00             LDX     #PC
 21 2980 20 3A 20          JSR     INCR
 22 2983 A5 01             LDA     PC+1
 2  2985 A4 00             LDY     PC      ;PUSH ADDRESS OF STRING ONTO STACK
 24 2987 20 87 20          JSR     PUSH2
 25 298A 18                CLC
 26 298B 68                PLA             ;GET NO OF BYTES BACK
 27 298C 65 00             ADC     PC      ;INCREMENT PC WITH LENGTH
```

```
'MACRO INTEPRETER FOR PASCAL-M'              PAL65-V1 PAGE  7-  1

28 298E 85 00           STA     PC
29 2990 A9 00           LDA     #$00
30 2992 65 01           ADC     PC+1
31 2994 85 01           STA     PC+1
32 2996 4C 29 23        JMP     TSTSTK
33                      .FILE   FILE3
34              W
35
36              ; DO CSP
37              ; CALL STANDARD PROCEDURE
38              ;
39 2999 A0 00   CSP     LDY     #$00    ;INDEX
40 299B B1 00           LDA     (PC),Y  GET CSP NUMBER
41 299D 0A              ASL             A NOT AS LA WE HAVE CV
42 299E A8              TAY             MULT BY TWO FOR JUMPTABLE INDEX
43 299F A2 00           LDX     #PC
44 29A1 20 3A 20        JSR     INCR    SET PC FOR NEXT INSTR.
45 29A4 B9 B2 29        LDA     CSPTAB,Y
   29A7 85 08           STA     JTABP   SETUP INDERECT JUMP
47 29A9 C8              INY
48 29AA B9 B2 29        LDA     CSPTAB,Y        PAGE NO
49 29AD 85 09           STA     JTABP+1
50 29AF 6C 08 00        JMP     (JTABP) JUMP TO ROUTINE
51 29B2 44 2A   CSPTAB  .WOR    WRI     WRITE INTEGER
52 29B4 15 2B           .WOR    WRC     WRITE CHARACTER
53 29B6 2F 2B           .WOR    WRS     WRITE STRING
54 29B8 5E 2B           .WOR    RDI     READ INTEGER
55 29BA BE 2B           .WOR    RLN     READ END OF LINE
56 29BC CF 2B           .WOR    RDC     READ CHARACTER
57 29BE DC 2B           .WOR    WLN     WRITE END OF LINE
58 29C0 E4 2B           .WOR    NEW     AQUIRE NEW POINTER
59 29C2 FE 2B           .WOR    EOF     CHECK FOR END OF FILE
60 29C4 08 2C           .WOR    RST     RESET HEAP POINTER
61 29C6 12 2C           .WOR    ELN     TEST IF END OF LINE
62 29C8 1C 2C           .WOR    STP     STOP PASCAL PROGRAM
63 29CA 49 2C           .WOR    ODD     CHECK IF NUMBER ON STACK IS ODD
64 29CC 58 2C           .WOR    RSET    RESET EOF BOOLEAN
                ; BE CUP1
66 29CE A5 06   CUP1    LDA     MPSAVE  MOVE MPSAVE TO MP
67 29D0 85 04           STA     MP
68 29D2 A5 07           LDA     MPSAVE+1
69 29D4 85 05           STA     MP+1
70 29D6 4C F8 29        JMP     CUPGO
71                      ; BF CUP2
72 29D9 20 7B 20 CUP2   JSR     PULL2   COMPLEX CALL WITH FUNKT ROOM
73 29DC 84 16           STY     TMP2L2  NO OF RESERVE STACK ROOM
74 29DE 18              CLC
75 29DF A5 20           LDA     STACK   SUBTRACKT ROOM FROM STACK
76 29E1 65 16           ADC     TMP2L2
77 29E3 85 16           STA     TMP2L2
78 29E5 A5 21           LDA     STACK+1
79 29E7 69 00           ADC     #$00
80 29E9 85 17           STA     TMP2L2+1
81 29EB A5 16           LDA     TMP2L2
82 29ED 18              CLC
83 29EE 69 06           ADC     #$06    OFSET FOR MARKSTACK
84 29F0 85 04           STA     MP
85 29F2 A5 17           LDA     TMP2L2+1
86 29F4 69 00           ADC     #$00
87 29F6 85 05           STA     MP+1
```

```
 88 29F8 A0 00     CUPGO  LDY     #$00
 89 29FA B1 00            LDA     (PC),Y  GET PROCEDURE NUMBER
 90 29FC 48               PHA             SAVE ON STACK
 91 29FD A2 00            LDX     #PC     INCREMENT PC FOR RETURN
 92 29FF 20 3A 20         JSR     INCR
 93 2A02 38               SEC
 94 2A03 A5 04            LDA     MP      GET A CORRECT PLACE TO STORE PC
 95 2A05 E9 05            SBC     #$05
 96 2A07 85 16            STA     TMP2L2
 97 2A09 A5 05            LDA     MP+1
 98 2A0B E9 00            SBC     #$00
 99 2A0D 85 17            STA     TMP2L2+1
100 2A0F A0 01            LDY     #$01    INDEX INTO MARK PACKAGE
101 2A11 A5 01            LDA     PC+1    HIGH PART OF PC
102 2A13 91 16            STA     (TMP2L2),Y      STORE IN PACKAGE
103 2A15 88               DEY             TO NEXT CELL
104 2A16 A5 00            LDA     PC
105 2A18 91 16            STA     (TMP2L2),Y      LOW PART OF PC
106 2A1A 68               PLA             GET INDEX BACK
107 2A1B 0A               ASL             A MULT BY 2 (2 WORD ENTRYS IN JUMP TA
108 2A1C AA               TAX             USE INDEX AS INDEX
109 2A1D BD 00 02         LDA     PRCBUF,X        ABS INDEX NICE FEATURE
110 2A20 85 00            STA     PC      LOW PART OF NEW PC
111 2A22 E8               INX
112 2A23 BD 00 02         LDA     PRCBUF,X        GET HIGH PART OF PC
113 2A26 85 01            STA     PC+1    NEW PC FINISHED SO START IT
114 2A28 4C 45 23         JMP     LOOP    HELLO NEW PROC !!
115                 ; C0 FIX21
116 2A2B 20 7F 20 FIX21   JSR     PULL    GET OPERAND TO SHIFT TO LOWER POSITIO
117 2A2E A8               TAY             PUT ON CORRECT POS.
118 2A2F A9 00            LDA     #$00    PUSH2DUMMY ON STACK
119 2A31 20 87 20         JSR     PUSH2   ONTO STACK FOR TRUNCATED FUNCTIONS
120 2A34 4C 45 23         JMP     LOOP
121                 ; C1 LNS
122 2A37 A0 08      LNS   LDY     #$08
123 2A39 A9 00      LNS1X1 LDA    #$00
124 2A3B 20 8D 20         JSR     PUSH
125 2A3E 88               DEY             SORRY IT TAKES THAT LONG
126 2A3F D0 F8            BNE     LNS1X1
127 2A41 4C 29 23         JMP     TSTSTK
128                 ; C2 NOT IMPLEMENTED FURTHER
129                 ;
130                 ; NEXT ARE THE STANDARD PROCEDURE HANDLERS
131                 ;
132                 ; PROC 0 WRI
133 2A44 20 8D 23 WRI     JSR     SETUP2  TMP3 IS LENGTH TO WRITE
134 2A47 A5 17            LDA     TMP2L2+1        AND TMP2 COMTAIMS THE NUMBER
135 2A49 85 0F            STA     TMP1L8+1        SAVE SIGN
136 2A4B 10 0F            BPL     WRI1    NOT TO BE COMPLEMENTED
137 2A4D 49 FF            EOR     #$FF    WE STILL HAVE IT
138 2A4F 85 17            STA     TMP2L2+1
139 2A51 A5 16            LDA     TMP2L2
140 2A53 49 FF            EOR     #$FF    NOW LOWER BITS
141 2A55 85 16            STA     TMP2L2
142 2A57 A2 16            LDX     #TMP2L2
143 2A59 20 3A 20         JSR     INCR    FORM 1NS COMPLEMENT
144 2A5C A0 05      WRI1  LDY     #$05    INIT TMP1L8
145 2A5E A2 00      WRI1A LDX     #$00
146 2A60 96 10            STX     TMP1L8+2,Y
147 2A62 88               DEY
```

```
148 2A63 10 F9          BPL     WRI1A   CLEAR 6 CELLS
149 2A65 20 81 2C       JSR     CVDEC   GET NUMBERS IN TMP1L8+3..7
150 2A68 A5 0F          LDA     TMP1L8+1
151 2A6A 30 21          BMI     WRI2    SETUP FOR OUTPUT NUMBER
152 2A6C A0 20          LDY     #$20
153 2A6E 84 0F          STY     TMP1L8+1        LENGTH IS 6 SIGN IS SPACE
154 2A70 84 10          STY     TMP1L8+2        NO PLUS TO DO
155 2A72 A5 11          LDA     TMP1L8+3        ADJUST LEADING SPACES
156 2A74 D0 3F          BNE     WRI3
157 2A76 84 11          STY     TMP1L8+3        STUF A SPACE
158 2A78 A5 12          LDA     TMP1L8+4
159 2A7A D0 39          BNE     WRI3
160 2A7C 84 12          STY     TMP1L8+4
161 2A7E A5 13          LDA     TMP1L8+5
162 2A80 D0 33          BNE     WRI3
163 2A82 84 13          STY     TMP1L8+5
164 2A84 A5 14          LDA     TMP1L8+6
165 2A86 D0 2D          BNE     WRI3
166 2A88 84 14          STY     TMP1L8+6        FOR TMP1L8+7 IS TAKEN CARE OFF
167 2A8A 4C B5 2A       JMP     WRI3    SO FAR FOR POSITIVE NUMBERS
168 2A8D A2 20  WRI2    LDX     #$20    GET A SPACE IN X
169 2A8F 86 0F          STX     TMP1L8+1        SIGN IS NOT IMPORTAND ANYMORE
170 2A91 A0 2D          LDY     #$2D    GET MINUS SIGN IN Y
171 2A93 84 10          STY     TMP1L8+2        -XXXXX LOOK LIKE THIS NOW
172 2A95 A5 11          LDA     TMP1L8+3        TEST IF THERE ARE THENTHOUSANI
173 2A97 D0 1C          BNE     WRI3    GO DOWN THE LADDRER
174 2A99 86 10          STX     TMP1L8+2        -XXXX
175 2A9B 84 11          STY     TMP1L8+3
176 2A9D A5 12          LDA     TMP1L8+4        TEST FOR THOUSANDS
177 2A9F D0 14          BNE     WRI3
178 2AA1 86 11          STX     TMP1L8+3
179 2AA3 84 12          STY     TMP1L8+4
180 2AA5 A5 13          LDA     TMP1L8+5
181 2AA7 D0 0C          BNE     WRI3
182 2AA9 86 12          STX     TMP1L8+4
183 2AAB 84 13          STY     TMP1L8+5
184 2AAD A5 14          LDA     TMP1L8+6
185 2AAF D0 04          BNE     WRI3
186 2AB1 86 13          STX     TMP1L8+5
187 2AB3 84 14          STY     TMP1L8+6        WE ARE NOW ON THE LAST DIGIT
188 2AB5 A5 15  WRI3    LDA     TMP1L8+7
189 2AB7 09 30          ORA     #$30    ADJUST TO ASCII
190 2AB9 85 15          STA     TMP1L8+7
191 2ABB A5 14          LDA     TMP1L8+6
192 2ABD C9 0A          CMP     #$0A    TEST IF ALREADY HIGHER BITS
193 2ABF B0 22          BCS     WRI4    IF YES THEN END OF TREE
194 2AC1 09 30          ORA     #$30
195 2AC3 85 14          STA     TMP1L8+6
196 2AC5 A5 13          LDA     TMP1L8+5
197 2AC7 C9 0A          CMP     #$0A
198 2AC9 B0 18          BCS     WRI4
199 2ACB 09 30          ORA     #$30
200 2ACD 85 13          STA     TMP1L8+5
201 2ACF A5 12          LDA     TMP1L8+4
202 2AD1 C9 0A          CMP     #$0A
203 2AD3 B0 0E          BCS     WRI4
204 2AD5 09 30          ORA     #$30
205 2AD7 85 12          STA     TMP1L8+4
206 2AD9 A5 11          LDA     TMP1L8+3
207 2ADB C9 0A          CMP     #$0A
```

```
208 2ADD B0 04          BCS     WRI4
209 2ADF 09 30          ORA     #$30
210 2AE1 85 11          STA     TMP1L8+3        NOW ALL OF TMP1L8+2.,7 IS ASC
211 2AE3 38     WRI4    SEC                     NOW WE SUBTRACT LENGTHS
212 2AE4 A5 18          LDA     TMP3L2  THIS IS WAT IS SPECIFIED
213 2AE6 E9 06          SBC     #$06    AND THIS HOW LONG DEFOULT
214 2AE8 30 0D          BMI     WRI5    IF LESS THAN DEFOULT
215 2AEA F0 0B          BEQ     WRI5    OR IF DEFOULT
216 2AEC A8             TAY             Y NOW CONTAINS NUMBER OF PRECEDING SP
217 2AED A9 20  WRI4A   LDA     #$20    SPACE
218 2AEF 20 10 20       JSR     WRT     OUTPUT IT
219 2AF2 C6 18          DEC     TMP3L2  ADJUST NUMBER TO LEAVE
220 2AF4 88             DEY
221 2AF5 D0 F6          BNE     WRI4A   IF NOT DONE YET
222 2AF7 A4 18  WRI5    LDY     TMP3L2  GET NO OF BYTES TO WRITE
223 2AF9 F0 17          BEQ     WRI6    ZERO CHAR TO DO /0/
224 2AFB 38             SEC
225 2AFC A9 16          LDA     #TMP1L8+8       GET ADDRESS WERE STRING START
226 2AFE E5 18          SBC     TMP3L2  SUBTRACKT LENGTH OF STRING
227 2B00 85 16          STA     TMP2L2  INIT INDERECT ADDRESS
228 2B02 A2 00          LDX     #$00
229 2B04 86 17          STX     TMP2L2+1        PAGE ZERO YOU KNEW
230 2B06 A1 16  WRI5A   LDA     (TMP2L2,X)
231 2B08 20 10 20       JSR     WRT
232 2B0B E6 16          INC     TMP2L2  POINT TO LOWER
233 2B0D A2 00          LDX     #$00    FOR SAVETY BECAUSE WRT DOESNT SAVE X
234 2B0F 88             DEY
235 2B10 D0 F4          BNE     WRI5A   NUMBER NOT COMPLETE JET
236 2B12 4C 45 23 WRI6  JMP     LOOP    DO YOU HAVE A NICE NUMBER NOW.
237                     ;
238                     ;PROC 1 WRC
239 2B15 20 7B 20 WRC   JSR     PULL2   GET NO OF SPACES
240 2B18 88     WRC1    DEY
241 2B19 F0 0A          BEQ     WRC2    ;Y WAS ONE IF SPECIEFIED LENGTH = 1
242 2B1B 30 08          BMI     WRC2    IN CASE OF LENGTH WAS 0
  1 2B1D A9 20          LDA     #$20    LOAD A SPACE
  2 2B1F 20 10 20       JSR     WRT     WRITE IT OUT
  3 2B22 4C 18 2B       JMP     WRC1    CHECK IF MORE SPACES TO DO
  4 2B25 20 7B 20 WRC2  JSR     PULL2   GET CHAR AND DUMMY
  5 2B28 98             TYA
  6 2B29 20 10 20       JSR     WRT     WRITE THE CHAR
  7 2B2C 4C 45 23       JMP     LOOP
  8                     ;
  9                     ; PROC 2 WRS
 10 2B2F 20 8D 23 WRS   JSR     SETUP2  TMP3L2 CONTAINS ACTUAL LENGTH
 11 2B32 38             SEC             AND TMP2L2 THE SPECIFIED LENGTH
 12 2B33 A5 16          LDA     TMP2L2
 13 2B35 E5 18          SBC     TMP3L2  ACTUAL - SPECIFIED = NUMBER OF SPACES
 14 2B37 A8             TAY
 15 2B38 10 08          BPL     WRS1    NO SPACES
 16 2B3A A5 16          LDA     TMP2L2  SPECIFIED LESS THAN ACTUAL
 17 2B3C F0 1D          BEQ     WRS4    IF SPECIFIED IS ZERO
 18 2B3E 85 18          STA     TMP3L2  OVETWRITE ACTUAL
 19 2B40 D0 0A          BNE     WRS2    START PRINTING
 20 2B42 F0 08  WRS1    BEQ     WRS2    IF NO OF SPACES IS ZERO
 21 2B44 A9 20          LDA     #$20
 22 2B46 20 10 20       JSR     WRT     OUTPUT SPACE
 23 2B49 88             DEY             CHECK FOR MORE SPACES
 24 2B4A D0 F6          BNE     WRS1    YES
 25 2B4C 20 94 23 WRS2  JSR     SETUP1  OVERWRITE SPECIFIED LENGTH WITH ADDRES
```

```
26 2B4F A0 00          LDY   #$00     Y IS INDEX TO STRING
27 2B51 B1 16    WRS3  LDA   (TMP2L2),Y    GET THE CHAR
28 2B53 20 10 20       JSR   WRT      SEND AWAY
29 2B56 C8             INY           POINT TO NEXT CHAR
30 2B57 C6 18          DEC   TMP3L2   DECREMENT ACTUAL TO GO
31 2B59 D0 F6          BNE   WRS3     IF MORE TO GO
32 2B5B 4C 45 23 WRS4  JMP   LOOP
33                     ; PROC 3 RDI
34 2B5E A9 00    RDI   LDA   #$00
35 2B60 85 0E          STA   TMP1L8   INITIAL RESULT IS ZERO
36 2B62 85 0F          STA   TMP1L8+1         UPPER OF RESULT
37 2B64 85 10          STA   TMP1L8+2         INIT SIGN FLAG = +
38 2B66 20 23 2C RDI1  JSR   GETCH    GET A CHAR
39 2B69 C9 20          CMP   #$20     CHECK FOR SPACE
40 2B6B F0 F9          BEQ   RDI1     SKIP THEM
41 2B6D C9 2B          CMP   #$2B     +
42 2B6F F0 06          BEQ   RDI2
43 2B71 C9 2D          CMP   #$2D     -
44 2B73 D0 05          BNE   RDI3     IT WAS NOT + OR -
45 2B75 E6 10          INC   TMP1L8+2         SET SIGN = 1 IS COMPLEMENT ON
46 2B77 20 23 2C RDI2  JSR   GETCH    GET NEXT CHAR
47 2B7A 38       RDI3  SEC             FOR SUBTRACT
48 2B7B E9 30          SBC   #$30     ADJUST TO ASCII ZERO
49 2B7D 30 19          BMI   RDI4     IF LESS THAN 0 END OF NUMBER
50 2B7F C9 0A          CMP   #$0A     IF LARGER THAN 9
51 2B81 B0 15          BCS   RDI4     ALSO END OF INTEGER
52 2B83 85 11          STA   TMP1L8+3
53 2B85 20 5F 2C       JSR   MUL10    MULTIPLY OLD NUMBER BY 10
54 2B88 18             CLC
55 2B89 A5 0E          LDA   TMP1L8
56 2B8B 65 11          ADC   TMP1L8+3         ADD INTO OLD NUMBER
57 2B8D 85 0E          STA   TMP1L8
58 2B8F A5 0F          LDA   TMP1L8+1
59 2B91 69 00          ADC   #$00     ADJUST CARRY
60 2B93 85 0F          STA   TMP1L8+1
61 2B95 4C 77 2B       JMP   RDI2     GET NEXT ITEM OR END
62 2B98 20 94 23 RDI4  JSR   SETUP1   GET ADDRESS TO STORE NUMBER INTO TMP2
63 2B9B A0 00          LDY   #$00
64 2B9D A5 10          LDA   TMP1L8+2         CHECK SIGN FLAG
65 2B9F F0 11          BEQ   RDI5     NOT TO COMPLEMENT
66 2BA1 A5 0E          LDA   TMP1L8
67 2BA3 49 FF          EOR   #$FF     COMPLEMENT
68 2BA5 85 0E          STA   TMP1L8
69 2BA7 A5 0F          LDA   TMP1L8+1
70 2BA9 49 FF          EOR   #$FF     COMPLEMENT UPPER
71 2BAB 85 0F          STA   TMP1L8+1
72 2BAD A2 0E          LDX   #TMP1L8 BIG TRICK FOR INCREMENT
73 2BAF 20 3A 20       JSR   INCR
74 2BB2 A5 0E    RDI5  LDA   TMP1L8   NOW STORE NUMBER
75 2BB4 91 16          STA   (TMP2L2),Y       LOW PART
76 2BB6 C8             INY
77 2BB7 A5 0F          LDA   TMP1L8+1
78 2BB9 91 16          STA   (TMP2L2),Y       HIGH PART
79 2BBB 4C 45 23       JMP   LOOP     HERE IS YOU NUMBER
80                     ; PROC 4 RLN
81 2BBE A5 1A    RLN   LDA   EOL      IS END OF LINE SET
82 2BC0 D0 06          BNE   RLN1     YES SO CLEAR
83 2BC2 20 23 2C       JSR   GETCH    NO SKIP INPUT TILL EOLN
84 2BC5 4C BE 2B       JMP   RLN
85 2BC8 A9 00    RLN1  LDA   #$00
```

```
 86 2BCA 85 1A          STA     EOL     ENDOFLINE IS CLEARED NOW
 87 2BCC 4C 45 23       JMP     LOOP
 88                     ; PROC 5 RDC
 89 2BCF 20 94 23 RDC   JSR     SETUP1  GET ADDRESS TO STORE CHAR IN INTO TM
 90 2BD2 20 23 2C       JSR     GETCH
 91 2BD5 A0 00          LDY     #$00
 92 2BD7 91 16          STA     (TMP2L2),Y      SAVE THE CHARACTER
 93 2BD9 4C 45 23       JMP     LOOP
 94                     ; PROC 6 WLN
 95 2BDC A9 80    WLN   LDA     #$80
 96 2BDE 20 1A 20       JSR     CRLF
 97 2BE1 4C 45 23       JMP     LOOP
 98                     ; PROC 7 NEW
 99 2BE4 20 8D 23 NEW   JSR     SETUP2  TMP3 = LENGTH OF ROOM WANTED
100 2BE7 A0 00          LDY     #$00    TMP2 = ADDRESS TO PUT OLD HP
101 2BE9 A5 02          LDA     HP
102 2BEB 91 16          STA     (TMP2L2),Y      PUT OLD HP IN CELL
103 2BED 18             CLC
104 2BEE 65 18          ADC     TMP3L2  ADD NUMBER OF CELS TO OLD HP
105 2BF0 85 02          STA     HP
106 2BF2 A5 03          LDA     HP+1    NOW DO UPPER PART
107 2BF4 C8             INY             POINT TO UPPER STORE CELL
108 2BF5 91 16          STA     (TMP2L2),Y
109 2BF7 65 19          ADC     TMP3L2+1
110 2BF9 85 03          STA     HP+1    ROOM IS RESERVED NOW
111 2BFB 4C 29 23       JMP     TSTSTK  CHECK IF WE GAVE TOO MUCH AWAY
112                     ; PROC 8 EOF
113 2BFE A4 22    EOF   LDY     EOFB    GET END OF FILE BOOLEAN
114 2C00 A9 00          LDA     #$00    DUMMY
115 2C02 20 87 20       JSR     PUSH2
116 2C05 4C 29 23       JMP     TSTSTK
117                     ; PROC 9 RST
118 2C08 20 7B 20 RST   JSR     PULL2   GET OLD HEAP POINTER FROM STACK
119 2C0B 85 03          STA     HP+1
120 2C0D 84 02          STY     HP
121 2C0F 4C 45 23       JMP     LOOP
122                     ; PROC A ELN
123 2C12 A4 1A    ELN   LDY     EOL     GET END OF LINE BOOLEAN
124 2C14 A9 00          LDA     #$00    DUMMY
125 2C16 20 87 20       JSR     PUSH2   PUT ON STACK
126 2C19 4C 29 23       JMP     TSTSTK
127                     ; PROC B STP
128 2C1C A9 20    STP   LDA     #>MSG002
129 2C1E A2 C2          LDX     #<MSG002
130 2C20 4C 95 20       JMP     STRING  ;PRINT -END PASCAL- AND STOP
131 2C23 20 06 20 GETCH JSR     RDT     ;READ CHAR FROM INPUT DEVICE
132 2C26 C9 20          CMP     #$20    ;TEST FOR SMALLER THAN SPACE
133 2C28 90 09          BCC     GETCH2
134 2C2A C9 7F          CMP     #$7F    ;TEST FOR RUBOUT
135 2C2C F0 F5          BEQ     GETCH   ;IGNORE RUBOUT
136 2C2E A0 00          LDY     #$00    ;NOT EOL BOOLEAN
137 2C30 84 1A    GETCH1 STY    EOL
138 2C32 60             RTS             ;RETURN WITH CHAR IN A
139 2C33 C9 0D    GETCH2 CMP    #$0D    ;TEST FOR CR
140 2C35 D0 06          BNE     GETCH3  ;IGNORE ALL CONTROL CHAR BUT CR AND
141 2C37 A9 20          LDA     #$20    ;FAKE A SPACE
142 2C39 A0 01          LDY     #$01    ;END OF LINE IS TRUE
143 2C3B D0 F3          BNE     GETCH1
144 2C3D C9 17    GETCH3 CMP    #$17
145 2C3F D0 E2          BNE     GETCH
```

```
2C41 A0 01          LDY     #$01
2C43 84 22          STY     EOFB    ;END OF FILE
2C45 A9 20          LDA     #$20
2C47 D0 E7          BNE     GETCH1  ;AND WITH EOF EOL IS TRUE
                    ; PROC C ODD
2C49 20 78 20 ODD   JSR     PULL2   ;GET INTEGER FROM STACK
2C4C 98             TYA             ;WE ARE INTERESTED ONLY IN BIT 0
2C4D 29 01          AND     #$01    ;IT IS OK FOR POS AND NEG NUMBERS
2C4F A8             TAY             ;ODD IS A BOOLEAN ONE IS BIT 0
2C50 A9 00          LDA     #$00    ;DUMMY
2C52 20 87 20       JSR     PUSH2   ;PUSH BOOLEAN ON STACK
2C55 4C 45 23       JMP     LOOP
                    ; PROC D RSET
2C58 A9 00    RSET  LDA     #$00    ;BOOLEAN FALSE
2C5A 85 22          STA     EOFB    ;END OF FILE IS FALSE NOW
2C5C 4C 45 23       JMP     LOOP    ;THAT WAS ALL TO DO
2C5F 06 0E    MUL10 ASL     TMP1L8  ;MULTIPLY TMP1L8 BY 10
2C61 26 0F          ROL     TMP1L8+1
2C63 A5 0E          LDA     TMP1L8
2C65 85 12          STA     TMP1L8+4
2C67 A5 0F          LDA     TMP1L8+1
2C69 85 13          STA     TMP1L8+5
2C6B 06 12          ASL     TMP1L8+4
2C6D 26 13          ROL     TMP1L8+5
2C6F 06 12          ASL     TMP1L8+4
2C71 26 13          ROL     TMP1L8+5        ;X 8
2C73 18             CLC
2C74 A5 0E          LDA     TMP1L8  ;ADD TIMES 2 AND 8
2C76 65 12          ADC     TMP1L8+4
2C78 85 0E          STA     TMP1L8
2C7A A5 0F          LDA     TMP1L8+1
2C7C 65 13          ADC     TMP1L8+5
2C7E 85 0F          STA     TMP1L8+1
2C80 60             RTS
2C81 38       CVDEC SEC             ;CONVERT HEX TO DECIMAL
2C82 A5 16          LDA     TMP2L2
2C84 E9 10          SBC     #$10    SUBTRACKT 10000
2C86 48             PHA             SAVE ON STACK
2C87 A5 17          LDA     TMP2L2+1
2C89 E9 27          SBC     #$27
2C8B 90 09          BCC     CVDEC2  NOT LARGER ANYMORE
2C8D 85 17          STA     TMP2L2+1        STILL LARGER SO TRY AGAIN
2C8F 68             PLA
2C90 85 16          STA     TMP2L2
2C92 E6 11          INC     TMP1L8+3        INCREMENT 10000
2C94 D0 EB          BNE     CVDEC   MORE TO DO
2C96 68       CVDEC2 PLA            CLEAN UP STACK
2C97 38       CVDEC3 SEC
2C98 A5 16          LDA     TMP2L2
2C9A E9 E8          SBC     #$E8
2C9C 48             PHA
2C9D A5 17          LDA     TMP2L2+1
2C9F E9 03          SBC     #$03    -1000
2CA1 90 09          BCC     CVDEC4
2CA3 85 17          STA     TMP2L2+1
2CA5 68             PLA
2CA6 85 16          STA     TMP2L2
2CA8 E6 12          INC     TMP1L8+4
2CAA D0 EB          BNE     CVDEC3
2CAC 68       CVDEC4 PLA            CLEAN UP
```

```
206 2CAD 38       CVDEC5 SEC             TMP2L2
207 2CAE A5 16           LDA     TMP2L2
208 2CB0 E9 64           SBC     #$64    100
209 2CB2 48              PHA
210 2CB3 A5 17           LDA     TMP2L2+1
211 2CB5 E9 00           SBC     #$00
212 2CB7 90 09           BCC     CVDEC6
213 2CB9 85 17           STA     TMP2L2+1
214 2CBB 68              PLA
215 2CBC 85 16           STA     TMP2L2
216 2CBE E6 13           INC     TMP1L8+5
217 2CC0 D0 EB           BNE     CVDEC5
218 2CC2 68       CVDEC6 PLA
219 2CC3 38       CVDEC7 SEC
220 2CC4 A5 16           LDA     TMP2L2
221 2CC6 E9 0A           SBC     #$0A
222 2CC8 90 06           BCC     CVDEC8
223 2CCA 85 16           STA     TMP2L2
22  2CCC E6 14           INC     TMP1L8+6
225 2CCE D0 F3           BNE     CVDEC7
226 2CD0 A5 16    CVDEC8 LDA     TMP2L2
227 2CD2 85 15           STA     TMP1L8+7
228 2CD4 60              RTS             END OF DECODE
229 2CD5 A0 05    FORIN  LDY     #$05
230 2CD7 B1 20           LDA     (STACK),Y
231 2CD9 85 18           STA     TMP3L2  STORE INITIAL LOOP COUNT
232 2CDB C8              INY
233 2CDC B1 20           LDA     (STACK),Y
234 2CDE 85 19           STA     TMP3L2+1
235 2CE0 C8              INY
236 2CE1 B1 20           LDA     (STACK),Y       GET ADDRESS OF LOOP CELL
237 2CE3 85 16           STA     TMP2L2
238 2CE5 C8              INY
239 2CE6 B1 20           LDA     (STACK),Y
240 2CE8 85 17           STA     TMP2L2+1
241 2CEA A0 00           LDY     #$00    NOW STORE INITIAL COUNT INTO CELL
242 2CEC A5 18           LDA     TMP3L2
24  2CEE 91 16           STA     (TMP2L2),Y
244 2CF0 C8              INY
245 2CF1 B1 20           LDA     (STACK),Y       TEST FOR 1 OR 2 BYTES
246 2CF3 29 01           AND     #$01
247 2CF5 F0 04           BEQ     FORIN1  1 BYTE
248 2CF7 A5 19           LDA     TMP3L2+1
  1 2CF9 91 16           STA     (TMP2L2),Y      SAVE UPPER 8 BITS
  2 2CFB 60       FORIN1 RTS             END OF INITIALIZATION OF FOR LOOP
  3 2CFC A0 0F    MPLY   LDY     #$0F
  4 2CFE 46 11    MPLY1  LSR     TMP1L8+3
  5 2D00 66 10           ROR     TMP1L8+2        GET LEAST BIT
  6 2D02 90 0D           BCC     MPLY2   IF NOT TO ADD
  7 2D04 18              CLC             CLEAR CARRY AGAIN
  8 2D05 A5 0E           LDA     TMP1L8
  9 2D07 65 12           ADC     TMP1L8+4
 10 2D09 85 12           STA     TMP1L8+4
 11 2D0B A5 0F           LDA     TMP1L8+1
 12 2D0D 65 13           ADC     TMP1L8+5
 13 2D0F 85 13           STA     TMP1L8+5
 14 2D11 06 0E    MPLY2  ASL     TMP1L8
 15 2D13 26 0F           ROL     TMP1L8+1        SHIFT MULIPICANT
 16 2D15 88              DEY
 17 2D16 D0 E6           BNE     MPLY1
```

```
18 2D18 60              RTS
19 2D19 20 7B 20 MDSET  JSR     PULL2   GET DIVIDOR/MULTIPLICANT
20 2D1C 84 0E           STY     TMP1L8
21 2D1E 85 0F           STA     TMP1L8+1
22 2D20 20 7B 20        JSR     PULL2   GET;DIVIDENT/MULTIPLIER
23 2D23 84 10           STY     TMP1L8+2
24 2D25 85 11           STA     TMP1L8+3
25 2D27 A9 00           LDA     #$00    CLEAR VARIUS CELLS
26 2D29 85 15           STA     TMP1L8+7        FOR NO OF SHIFTS(MULT)
27 2D2B 85 14           STA     TMP1L8+6        END SIGN 1=NEG 0 OR 2 IS POS
28 2D2D 85 12           STA     TMP1L8+4        CLEAR END RESULT
29 2D2F 85 13           STA     TMP1L8+5
30 2D31 A5 0F           LDA     TMP1L8+1        TEST SIGN
31 2D33 10 11           BPL     MDSET1
32 2D35 49 FF           EOR     #$FF
33 2D37 85 0F           STA     TMP1L8+1
34 2D39 A5 0E           LDA     TMP1L8
35 2D3B 49 FF           EOR     #$FF
36 2D3D 85 0E           STA     TMP1L8
37 2D3F A2 0E           LDX     #TMP1L8
38 2D41 20 3A 20        JSR     INCR
39 2D44 E6 14           INC     TMP1L8+6
40 2D46 A5 11     MDSET1 LDA    TMP1L8+3
41 2D48 10 11           BPL     MDSET2
42 2D4A 49 FF           EOR     #$FF
43 2D4C 85 11           STA     TMP1L8+3
44 2D4E A5 10           LDA     TMP1L8+2
45 2D50 49 FF           EOR     #$FF
46 2D52 85 10           STA     TMP1L8+2
47 2D54 A2 10           LDX     #TMP1L8+2
48 2D56 20 3A 20        JSR     INCR
49 2D59 E6 14           INC     TMP1L8+6        ADJUST END SIGN
50 2D5B 60        MDSET2 RTS
51 2D5C A0 01     DIVIDE LDY    #$01    WE DO IT AT LEAST ONCE
52 2D5E 38        DIV1   SEC
53 2D5F A5 10           LDA     TMP1L8+2
54 2D61 E5 0E           SBC     TMP1L8
55 2D63 A5 11           LDA     TMP1L8+3
56 2D65 E5 0F           SBC     TMP1L8+1
57 2D67 90 10           BCC     DIV2    IF DIVISOR IS LARGER THAN DIVIDENT
58 2D69 C8              INY
59 2D6A 06 0E           ASL     TMP1L8
60 2D6C 26 0F           ROL     TMP1L8+1
61 2D6E C0 10           CPY     #$10    IF MORE THAN 16 TIMES
62 2D70 D0 EC           BNE     DIV1
63 2D72 A2 27           LDX     #<MSG008        DIVISION BY 0
64 2D74 A9 21           LDA     #>MSG008
65 2D76 4C 95 20        JMP     STRING
66 2D79 84 15     DIV2   STY    TMP1L8+7
67 2D7B 38        DIV3   SEC
68 2D7C A5 10           LDA     TMP1L8+2
69 2D7E E5 0E           SBC     TMP1L8
70 2D80 48              PHA             SAVE RESULT ON STACK
71 2D81 A5 11           LDA     TMP1L8+3
72 2D83 E5 0F           SBC     TMP1L8+1
73 2D85 08              PHP             SAVE STATUS
74 2D86 26 12           ROL     TMP1L8+4
75 2D88 26 13           ROL     TMP1L8+5
76 2D8A 28              PLP
77 2D8B 90 08           BCC     DIV4
```

```
 78 2D8D 85 11           STA     TMP1L8+3
 79 2D8F 68              PLA
 80 2D90 85 10           STA     TMP1L8+2
 81 2D92 4C 96 2D        JMP     DIV5
 82 2D95 68      DIV4    PLA
 83 2D96 46 0F   DIV5    LSR     TMP1L8+1
 84 2D98 66 0E           ROR     TMP1L8
 85 2D9A C6 15           DEC     TMP1L8+7
 86 2D9C D0 DD           BNE     DIV3
 87 2D9E 60              RTS
 88 2D9F 20 8D 23 COMPAR JSR     SETUP2  TMP3 IS RIGHT SIDE OF COMPARE
 89 2DA2 A0 00           LDY     #$00
 90 2DA4 B1 00           LDA     (PC),Y  GET NO TO COMPARE
 91 2DA6 85 0F           STA     TMP1L8+1
 92 2DA8 A2 00           LDX     #PC
 93 2DAA 20 3A 20        JSR     INCR
 94 2DAD B1 00           LDA     (PC),Y
 95 2DAF 85 0E           STA     TMP1L8
    2DB1 A2 00           LDX     #PC
    2DB3 20 3A 20        JSR     INCR
 98 2DB6 38      COMPA1  SEC
 99 2DB7 B1 16           LDA     (TMP2L2),Y
100 2DB9 F1 18           SBC     (TMP3L2),Y
101 2DBB F0 09           BEQ     COMPA4  IF STILL THE SAME CHECK END
102 2DBD 10 03           BPL     COMPA2  IF (TMP2) SMALLER
103 2DBF 88              DEY             Y=-1
104 2DC0 D0 01           BNE     COMPA3
105 2DC2 C8     COMPA2  INY
106 2DC3 84 0E  COMPA3  STY     TMP1L8  LEAVE RESULT
107 2DC5 60             RTS
108 2DC6 A2 16  COMPA4  LDX     #TMP2L2
109 2DC8 20 3A 20       JSR     INCR
110 2DCB A2 18          LDX     #TMP3L2
111 2DCD 20 3A 20       JSR     INCR
112 2DD0 A2 0E          LDX     #TMP1L8
113 2DD2 20 4C 20       JSR     DECR
114 2DD5 A5 0E          LDA     TMP1L8
    2DD7 D0 D0          BNE     COMPA1  IF NOT DONE YET
116 2DD9 A5 0F          LDA     TMP1L8+1
117 2DDB D0 D9          BNE     COMPA1
118 2DDD 84 0E          STY     TMP1L8  DONE SO LEAVE RESULT
119 2DDF 60             RTS
120 2DE0
121 2DE0
  1                     ;
  2                     ; LEES EN DUMP ROUTINE VOOR DE
  3                     ; PASCAL INTERPRETER.
  4                     ;
  5                     ; PROGRAMMEUR: W.V. GELDEREN
  6                     ; DATUM:       05-11-1979
  7                     ; PLAATS:      KROMMENIE
  8                     ;
  9       2500  TIMTMP  =       $25
 10       2600  TAMTMP  =       TIMTMP+1
 11       F000  GANG    =       $F0     CWRITE POINTER
 12       F100  TIC     =       GANG+1  CWRITE TIMER
    F200  COUNT   =       TIC+1   CWRITE COUNTER
 14       F300  TMP     =       COUNT+1 TEMP STORAGE
 15       F400  YTMP    =       TMP+1   IDEM
 16       F500  XTEMP   =       YTMP+1  IDEM
```

```
                FE00    TRIB    =       $FE     CYCLE COUNTER
                0003    BUFFER  =       $300    INPUT BUFFER
                0803    BUFFR1  =       $380    OUTPUT BUFFER
                        ;
                        ; KIM ROM AND PIA ADRESSES
                        ;
                4217    SBD     =       $1742   PIA LOCATION
                E717    CHKL    =       $17E7   CHKSUM
                E817    CHKH    =       $17E8
                EC17    VEB     =       $17EC   VOLATILE EXECUTION BLOCK
                F517    SAL     =       $17F5   TAPE START ADRESS
                F617    SAH     =       $17F6
                F717    EAL     =       $17F7   TAPE END ADDRESS
                F817    EAH     =       $17F8
                3219    INTVEB  =       $1932   INIT VEB SUBROUTINE
                4C19    CHKT    =       $194C   CHECK SUM SUBROUTINE
                F319    RDBYTE  =       $19F3   READ BYTE SUBROUTINE
                EA19    INCVEB  =       $19EA   INCREMENT VEB SUBROUTINE
                241A    RDCHT   =       $1A24   READ CHAR SUBROUTINE
                411A    RDBIT   =       $1A41   READ BIT SUBROUTINE
                8C1E    INIT    =       $1E8C   RESET ALL PIA'S
2DF6                    ORG     $2DF6
                        ;
                        ; READ ONE CHARACTER FROM TTY
2DF6 84 F4      READ    STY     YTMP
2DF8 20 5A 1E           JSR     TTYIN
2DFB A4 F4              LDY     YTMP
2DFD 60                 RTS
                        ;
                        ; WRITE ONE CHARACTER TO TELETYPE
                        ;
2DFE 84 F4      OUTPUT  STY     YTMP
2E00 20 A0 1E           JSR     TTYOUT
2E03 A4 F4              LDY     YTMP
2E05 60                 RTS
                        ;
                        ; ROUTINE GEEFT EEN CHARACTER AAN UIT
                        ; EEN BUFFER EN VULT DEZE BUFFER
                        ; ALS DEZE LEEG IS.
                        ; CHARACTERS WORDEN VAN CASSETTE TAPE GELEZEN
                        ;
2E06 84 25      INALL   STY     TIMTMP
2E08 A5 F5      INALLA  LDA     XTEMP
2E0A 30 14              BMI     RED
2E0C A4 F4              LDY     YTMP
2E0E B9 00 03           LDA     BUFFER,Y
2E11 C9 0D              CMP     #$D
2E13 D0 06              BNE     PAKM
2E15 48                 PHA
2E16 A9 FF              LDA     #$FF
2E18 85 F5              STA     XTEMP
2E1A 68                 PLA
2E1B E6 F4      PAKM    INC     YTMP
2E1D A4 25              LDY     TIMTMP
2E1F 60                 RTS
2E20 20 70 2E   RED     JSR     GETLIN
2E23 20 4A 2E           JSR     INTPAN
2E26 A9 00              LDA     #0
2E28 85 F5              STA     XTEMP
2E2A 4C 08 2E           JMP     INALLA
```

```
 77                    ;
 78                    ; INITIALISERING VAN LEES/DUMP BUFFER
 79                    ;
 80 2E2D 20 4A 2E INOT    JSR     INTPAN
 81 2E30 A9 0C            LDA     #$C
 82 2E32 8D 03 17         STA     $1703
 83 2E35 A9 BF            LDA     #$BF
 84 2E37 8D 02 17         STA     $1702
 85 2E3A A2 FF            LDX     #$FF
 86 2E3C 68               PLA
 87 2E3D A8               TAY
 88 2E3E 68               PLA
 89 2E3F 9A               TXS
 90 2E40 48               PHA
 91 2E41 98               TYA
 92 2E42 48               PHA
 93 2E43 86 F5            STX     XTEMP
 94 2E45 A9 00            LDA     #0
 95 2E47 85 26            STA     TAMTMP
 96 2E49 60               RTS
 97                    ;
 98                    ; INITIALISERING V D BUFFER POINTER
 99                    ;
100 2E4A A0 00    INTPAN LDY     #0
101 2E4C 84 F4           STY     YTMP
102 2E4E 60              RTS
103                   ;
104                   ; PLAATS DE AANGEBODEN CHARACTER IN HET
105                   ; BUFFER, ALS HET BUFFER VOL IS , DE INHOUD
106                   ; OP DE CASSETTE DUMPEN
107                   ;
108 2E4F 84 25    OUTALL STY     TIMTMP
109 2E51 A4 26           LDY     TAMTMP
110 2E53 C9 0D           CMP     #$D
111 2E55 D0 11           BNE     NODUP
112 2E57 99 80 03        STA     BUFFR1,Y
113 2E5A C8              INY
114 2E5B A9 0A           LDA     #$A
115 2E5D 99 80 03        STA     BUFFR1,Y
116 2E60 20 C5 2E        JSR     DUMP
117 2E63 A9 00           LDA     #0
118 2E65 85 26           STA     TAMTMP
119 2E67 60              RTS
120 2E68 99 80 03 NODUP  STA     BUFFR1,Y
121 2E6B E6 26           INC     TAMTMP
122 2E6D A4 25           LDY     TIMTMP
123 2E6F 60              RTS
124                   ;
125                   ; LEES EEN REGEL VAN DE TAPE
126                   ;
127 2E70 AD 02 17 GETLIN LDA     $1702
128 2E73 29 FB           AND     #$FB
129 2E75 8D 02 17        STA     $1702
130 2E78 A9 7F           LDA     #$7F
131 2E7A 8D 41 17        STA     $1741
132 2E7D A9 13           LDA     #$13
133 2E7F 8D 42 17        STA     SBD
134 2E82 D8              CLD
135 2E83 20 70 2E NEWLIX JSR     GETLIN
136 2E86 AD 02 17 ENDAD  LDA     $1702
```

```
137 2E89 09 04          ORA     #$4
138 2E8B 8D 02 17       STA     $1702
139 2E8E 4C 8C 1E       JMP     INIT
140                     ;
141 2E91 A9 00    GETLYN LDA    #0
142 2E93 85 F4          STA     YTMP
143 2E95 20 41 1A SYNC  JSR     RDBIT
144 2E98 46 F3          LSR     TMP
145 2E9A 05 F3          ORA     TMP
146 2E9C 85 F3          STA     TMP
147 2E9E 8D 40 17       STA     $1740
148 2EA1 C9 16    TST   CMP     #$16
149 2EA3 D0 F0          BNE     SYNC
150                     ;
151 2EA5 20 24 1A       JSR     RDCHT
152 2EA8 8D 40 17       STA     $1740
153 2EAB C9 13          CMP     #$13
154 2EAD D0 F2          BNE     TST
155                     ;
156 2EAF 20 4A 2E GETCHR JSR    INTPAN
157 2EB2 20 24 1A GET   JSR     RDCHT
158 2EB5 A4 F4          LDY     YTMP
159 2EB7 C9 81          CMP     #$81
160 2EB9 F0 05          BEQ     DOIR
161 2EBB E6 F4          INC     YTMP
162 2EBD 99 00 03       STA     BUFFER,Y
163 2EC0 C9 0D    DOIR  CMP     #$D
164 2EC2 D0 EE          BNE     GET
165 2EC4 60             RTS
166                     ;
167 2EC5 AD 02 17 DUMP  LDA     $1702
168 2EC8 29 F7          AND     #$F7
169 2ECA 8D 02 17       STA     $1702
170                     ;
171                     ; OUTPUT SOURCE DATA
172 2ECD A9 27    SORCOT LDA    #$27
173 2ECF 85 F0          STA     GANG
174 2ED1 A9 AD          LDA     #$AD
175 2ED3 8D EC 17       STA     VEB
176 2ED6 A9 00          LDA     #0
177 2ED8 8D E7 17       STA     CHKL
178 2EDB 8D E8 17       STA     CHKH
179 2EDE A9 80          LDA     #<BUFFR1
180 2EE0 8D ED 17       STA     VEB+1
181 2EE3 A9 03          LDA     #>BUFFR1
182 2EE5 8D EE 17       STA     VEB+2
183 2EE8 A9 60          LDA     #$60
184 2EEA 8D EF 17       STA     VEB+3
185 2EED A9 BF          LDA     #$BF    TURN ON OUTPUT
186 2EEF 8D 43 17       STA     $1743   TO CASSETTE
187 2EF2 A2 64          LDX     #100    # SYNCPULSES
188 2EF4 A9 16    LEADER LDA    #$16
189 2EF6 20 20 2F       JSR     HIC
190 2EF9 A9 13          LDA     #$13    SEND START OF DATA CHARACTER
191 2EFB 20 2C 2F       JSR     OUTCHT
192 2EFE 20 EA 19 NEXT  JSR     INCVEB
193 2F01 20 EC 17       JSR     VEB
194 2F04 C9 0A          CMP     #$A
195 2F06 D0 08          BNE     NOCRA   BRANCH IF NOT END OF LINE
196 2F08 20 2C 2F       JSR     OUTCHT  END OF LINE
```

```
197 2F0B A2 02           LDX     #2      WAIT A MOMENT
198 2F0D 4C 66 2F ONEMIN JMP     TAIL
199 2F10 C9 40    NOCRA  CMP     #$40    END OF FILE
200 2F12 F0 06           BEQ     EOFC    BRANCH IF YES
201 2F14 20 2C 2F        JSR     OUTCHT
202 2F17 4C FE 2E        JMP     NEXT
203 2F1A 20 2C 2F EOFC   JSR     OUTCHT
204 2F1D 4C 66 2F        JMP     TAIL
205                      ;
206                      ; DUMP LEADER
207 2F20 86 F1    HIC    STX     TIC
208 2F22 48       HICA   PHA
209 2F23 20 2C 2F        JSR     OUTCHT
210 2F26 68              PLA
211 2F27 C6 F1           DEC     TIC
212 2F29 D0 F7           BNE     HICA
213 2F2B 60              RTS
214                      ;
215                      ; SUB TO SEND ONE 8 BIT BYTE
216 2F2C A0 08    OUTCHT LDY     #8
217 2F2E 84 F2           STY     COUNT   8 BIT COUNT
218 2F30 A0 02    TRY    LDY     #2      START AT
219 2F32 84 FE           STY     TRIB    3600 HZ
220 2F34 BE 62 2F ZON    LDX     NPUL,Y  NUMBER OF HALF CYCLES
221 2F37 48              PHA             SAVE THE CHARACTER
222 2F38 2C 47 17 ZONA   BIT     $1747   WAIT FOR END OF CYCLE
223 2F3B 10 FB           BPL     ZONA
224 2F3D B9 63 2F        LDA     TIMG,Y  SETUP TIMER
225 2F40 8D 44 17        STA     $1744   FOR THIS PULSE
226 2F43 A5 F0           LDA     GANG    CHANGE STATE
227 2F45 49 80           EOR     #$80    OF OUTPUT
228 2F47 8D 42 17        STA     $1742   PORT
229 2F4A 85 F0           STA     GANG    AND SAVE STATE
230 2F4C CA              DEX             DONE ALL CYCLES ?
231 2F4D D0 E9           BNE     ZONA    BRANCH IF NO
232 2F4F 68              PLA
233 2F50 C6 FE           DEC     TRIB    ONE MORE GONE
234 2F52 F0 05           BEQ     SETZ    THE LAST ONE TOO
235 2F54 30 07           BMI     ROUT    EVEN THE LAST ONE WENT
236 2F56 4A              LSRA            ANOTHER BIT TO THE CARRY
237 2F57 90 DB           BCC     ZON     IF IT IS NOT SET
238 2F59 A0 00    SETZ   LDY     #0      SWITCH TO 2400 HERTZ
239 2F5B F0 D7           BEQ     ZON     ALWAYS
240 2F5D C6 F2    ROUT   DEC     COUNT   ONE BIT SENT
241 2F5F D0 CF           BNE     TRY
242 2F61 60              RTS             ALL OVER GO HOME
243                      ;
244                      ; TIMING TABLE
245                      ;
246 2F62 02       NPUL   BYT     2       TWO PULSES
247 2F63 C3 03 7E TIMG   BYT     $C3,3,$7E       THE RIGHT TIME
248                      ;
249 2F66 A9 2F    TAIL   LDA     #$2F
250 2F68 20 2C 2F        JSR     OUTCHT  AS CHAR
251 2F6B AD E7 17        LDA     CHKL    SEND
252 2F6E 20 8C 2F        JSR     OUTBT   CHECKSUM
253 2F71 AD E8 17        LDA     CHKH    LO AND
254 2F74 20 8C 2F        JSR     OUTBT   HI
255 2F77 A2 02           LDX     #2      AND SEND 2 CHARS
256 2F79 A9 04           LDA     #4      EOT
```

```
257 2F7B 20 20 2F          JSR     HIC
258 2F7E AD 02 17          LDA     $1702    TURN CASSETTE OFF
259 2F81 09 08             ORA     #8
260 2F83 8D 02 17          STA     $1702
261 2F86 4C 8C 1E          JMP     INIT
262                        ;
263                        ; SUB TO SEND CHAR WITH CHECKSUM CALCULATION
264 2F89 20 4C 19 OUTBTC   JSR      CHKT     ADD CHAR TO CHECKSUM
265                        ;
266                        ; SUB TO SEND BYTE AS TWO ASCII CHARACTERS
267 2F8C 48               OUTBT    PHA              SAVE BYTE
268 2F8D 4A                        LSRA             GET UPPER NIBBLE
269 2F8E 4A                        LSRA
270 2F8F 4A                        LSRA
271 2F90 4A                        LSRA
272 2F91 20 95 2F                  JSR      HEXT     AND SEND IT
273 2F94 68                        PLA              RETURN BYTE
27                         ;
275                        ; SUB TO SEND ONE HEX CHARACTER AS ASCII
276 2F95 29 0F            HEXT     AND      #$F
277 2F97 C9 0A                     CMP      #$A      CHANGE TO ASCII
278 2F99 18                        CLC              BY ADDING
279 2F9A 30 02                     BMI      HEXAT
280 2F9C 69 07                     ADC      #7       37 TO A..F
281 2F9E 69 30            HEXAT    ADC      #'0
282 2FA0 4C 2C 2F                  JMP      OUTCHT
283       A32F            PATCH    =        *        SOME SPACE FOR HACKERS
284 3007                  *        =        *+100
285       0730            BASBAS   =        *        BEGIN OF PROGRAM SPACE
286                                END
```

```
ADCHKS 2321  ADDGET 2166  ADDG1  217E  ADDR   001C  ADI    2691
ANDB   26A4  ASSEM  001E  BASBAS 3007  BASE   002C  BASEAD 2139
BAS1   2148  BAS2   2165  BUFFER 0300  BUFFR1 0380  CAP    2962
CAS    2887  CASEX  28DD  CAS2   28CD  CHKH   17E8  CHKL   17E7
CHKSUM 001F  CHKT   194C  CMP2   2500  CMP2A  2515  CMP2B  251A
COMMAN 1C4F  COMPAR 2D9F  COMPA1 2DB6  COMPA2 2DC2  COMPA3 2DC3
COMPA4 2DC6  CONTIN 2003  COUNT  00F2  CRLF   201A  CRLF1  202C
CRLF2  2036  CSP    2999  CSPTAB 29B2  CUPGO  29F8  CUP1   29CE
CUP2   29D9  CVDEC  2C81  CVDEC2 2C96  CVDEC3 2C97  CVDEC4 2CAC
CVDEC5 2CAD  CVDEC6 2CC2  CVDEC7 2CC3  CVDEC8 2CD0  C2     2202
C3     2204  C4     2206  DECB   2841  DECR   204C  DECR1  205B
DECX1  2852  DEC1   292B  DEC1A  2933  DIF    26B4  DIF1   26B8
DIVIDE 2D5C  DIV1   2D5E  DIV2   2D79  DIV3   2D7B  DIV4   2D95
DIV5   2D96  DOIR   2EC0  DUMP   2EC5  DVI    26C8  DVI1   26E5
EAH    17F8  EAL    17F7  ELN    2C12  ENDAD  2E86  ENDCOR 002E
ENT    2871  ENTRY  2000  EOF    2BFE  EOFB   0022  EOFC   2F1A
EOL    001A  EQUM   25A6  EQUM1  25B0  EQUM2  25B6  EQU2   2560
EQU2X1 2572  EQU8   25B9  EQU8X1 25C1  EQU8X2 25CA  EQU8X3 25E1
EXIT   2037  FIX21  2A2B  FJP    2826  FOR    244E  FOREND 24D2
FORIN  2CD5  FORIN1 2CFB  FORLP  24B6  FOR1   245E  FOR2   247C
FOR3   248A  FOR4   248F  FOR5   24A0  FOR6   24C4  GANG   00F0
GEQ8   2575  GEQ8A  257D  GEQ8B  258C  GEQ8C  25A3  GET    2EB2
GETCH  2C23  GETCHR 2EAF  GETCH1 2C30  GETCH2 2C33  GETCH3 2C3D
GETHEX 230B  GETLIN 2E70  GETLYN 2E91  HEXAT  2F9E  HEXIT  2070
HEXIT1 207A  HEXT   2F95  HIC    2F20  HICA   2F22  HP     0002
INALL  2E06  INALLA 2E08  INCB   285D  INCPCA 206D  INCPC2 205E
INCR   203A  INCR1  2049  INCVEB 19EA  INC1   293D  IND1   25E4
IND2   25F4  IND8   2609  IND8X1 260C  IND8X2 260E  INIT   1E8C
INN    26EF  INN1   26F1  INN2   270B  INN3   2718  INN4   271A
INOT   2E2D  INT    2722  INTPAN 2E4A  INTVEB 1932  INT1   2726
IOR    2734  JTAB   219E  JTABP  0008  JTAB2  2208  JTAB2P 000A
LCA    2979  LDAD   23B3  LDAS   23A6  LDC    2657  LDCIS  239C
LDCS   2948  LDCS1  294A  LDC1   2660  LDC2   2669  LEADER 2EF4
LEQM   24E5  LEQM1  24F1  LEQM2  24F7  LEQ2   243B  LEQ2X0 2440
LEQ2X1 2444  LEQ2X2 2446  LEQ8   251F  LEQ8A  2527  LEQ8B  2532
LEQ8C  2549  LESM   254C  LESM1  2557  LESM2  255D  LES2   24FA
LNF    280D  LNS    2A37  LNS1X1 2A39  LOADA  224B  LOADB  2256
LODER  222E  LOADIN 221F  LODERR 22AA  LOD1   23F6  LOD2   2405
LOOP   2345  LOOP1  2351  LOOP2  235C  MDSET  2D19  MDSET1 2D46
MDSET2 2D5B  MOD    2742  MOD1   275F  MOV    28F6  MOV1   290D
MP     0004  MPI    2769  MPI1   2786  MPLY   2CFC  MPLY1  2CFE
MPLY2  2D11  MPSAVE 0006  MSG001 20B2  MSG002 20C2  MSG003 20CE
MSG004 20DF  MSG005 20ED  MSG006 20FD  MSG007 2116  MSG008 2127
MSTART 221A  MSTN   23E2  MSTN1  23EC  MST0   23C0  MUL10  2C5F
NAME   0030  NEW    2BE4  NEWLIX 2E83  NEXT   2EFE  NGI    2790
NOCRA  2F10  NODUP  2E68  NOT    27AE  NOTIMP 2386  NPUL   2F62
ODD    2C49  ONEMIN 2F0D  OP     001B  OUTALL 2E4F  OUTBT  2F8C
OUTBTC 2F89  OUTCHT 2F2C  OUTPUT 2DFE  OVFL   2336  PACOP  2374
PAKM   2E1B  PATCH  2FA3  PC     0000  PGO    225D  PRCBUF 0200
PROCEX 00DC  PROC1  00C0  PROC2  00C8  PROC3  00D0  PROC4  00D8
PULL   207F  PULL2  207B  PUSH   208D  PUSH2  2087  RDBIT  1A41
RDBYTE 19F3  RDC    2BCF  RDCHT  1A24  RDHSR  200D  RDI    2B5E
RDI1   2B66  RDI2   2B77  RDI3   2B7A  RDI4   2B98  RDI5   2BB2
RDT    2006  RDT1   200A  READ   2DF6  REC1   228C  REC1X1 2291
REC1X2 22A3  REC2   22B1  REC4   22E1  REC4X1 22F3  RED    2E20
RETP   2670  RLN    2BBE  RLN1   2BC8  ROUT   2F5D  RSET   2C58
RSH    2C08  SAVGET 2189  SAH    17F6  SAL    17F5  SBD    1742
SBI    27B9  SETIN  0023  SETOUT 0024  SETUP1 2394  SETUP2 238D
SETZ   2F59  SGS    27CC  SGS1   27D0  SGS2   27E4  SGS3   27EF
SORCOT 2ECD  STACK  0020  STINB  20AC  STO1   2619  STO2   2625
```

STO8	263A	STO8X1	2647	STP	2C1C	STRINA	20A0	STRING	2095
STRPTR	000C	STR1	241A	STR2	2426	SYNC	2E95	TAIL	2F66
TAMTMP	0026	TIC	00F1	TIMG	2F63	TIMTMP	0025	TMP	00F3
TMP1L8	000E	TMP2L2	0016	TMP3L2	0018	TRIB	00FE	TRY	2F30
TST	2EA1	TSTSTK	2329	TST1X1	233D	TTYIN	1E5A	TTYOUT	1EA0
UJP	2833	UNI	27FB	UNI1	27FF	VEB	17EC	W	2999
WLN	2BDC	WRC	2B15	WRC1	2B18	WRC2	2B25	WRHSP	2017
WRI	2A44	WRI1	2A5C	WRI1A	2A5E	WRI2	2A8D	WRI3	2AB5
WRI4	2AE3	WRI4A	2AED	WRI5	2AF7	WRI5A	2B06	WRI6	2B12
WRS	2B2F	WRS1	2B42	WRS2	2B4C	WRS3	2B51	WRS4	2B5B
WRT	2010	WRT1	2014	XTEMP	00F5	YTMP	00F4	ZON	2F34
ZONA	2F38								