Appendix F
Motorola Diagnostic Listings

The source for the diagnostic programs are written in Motorola 6800 assembly language. They are found on the standard software release disk under the following names

PANEL.*
ALU.*
IFU.*
EXECUTER.*

The object programs are stored in a special ASCII format known as the Motorola S-format. The filenames for these programs are:

alu.MOT
ifu.MOT
ex.MOT

The program named 'PANEL' is not on the disk in object form. It is stored in ROM on the DPU.

*** the listings for the Motorola software will not be included in the preliminary release of the hardware manual.

```
SUBMONIT

 2  0000                    BEGIN SUBMONIT
 3
 4                 ************************************************
 5                 *                                              *
 6                 *       SUBMONITORS FOR:          REV.     23.9.80 *
 7                 *                                              *
 8                 *       MAIN MEMORY                        23.9.80 *
 9                 *       DEVICES                            23.9.80 *
10                 *       EXECUTER                           17.8.79 *
11                 *       MICRO MEMORY                               *
12                 *                                              *
13                 ************************************************
```

```
16                  ********************************************************************************
17                  *                                                                              *
18                  *      DIAGNOSTIC PANEL M6802-SOFTWARE                    23.05.80             *
19                  *                                                                              *
20                  *                J.HOPPE                                                       *
21                  *                I.NOACK                                                       *
22                  *                R.OHRAN                                                       *
23                  *                W.WINIGER                                                     *
24                  *                                                                              *
25                  ********************************************************************************
26
27
28                  BIN         EQU   0C09C
29                  ***         HEXADECIMAL (ASCII) TO BINARY CONVERSION
30                  *
31                  *           ARGUMENT:    A = CHARACTER TO BE CONVERTED
32                  *           RESULTS:     CARRY = RESET, IF A CONTAINED A VALID HEX DIGIT
33                  *                             SET OTHERWISE
34                  *                        A = BINARY VALUE OF THE PARAMETER A IF CARRY RESET
35                  *                            UNDEFINED OTHERWISE
36
37                  HEX         EQU   0C0B3
38                  ***         BINARY TO HEXADECIMAL (ASCII) CONVERSION
39                  *
40                  *           ARGUMENT:    A = FOUR BIT VALUE TO BE CONVERTED
41                  *           RESULTS:     A = ASCII-CHARACTER REPRESENTING AS HEX DIGIT
42                  *                            THE FOUR BIT NUMBER CONTAINED IN PAR. A
43
44                  R.CH        EQU   0C024
45                  ****        READ A CHARACTER FROM THE HP AND ECHO IT
46                  *
47                  *           ARGUMENT:    -
48                  *           RESULT:      A = CHARACTER READ FROM THE UART
49                  *           SIDE EFFECT: R.CH POLLS UNTIL A CHARACTER IS SENT TO IT
50                  *                        A = ESC -> CONTROL IS TRANSFERED TO THE SUBMONITOR
51                  *                        A = ↑C -> CONTROL IS TRANSFERED TO THE MONITOR
52                  *                        A = QUESTIONMARK -> MENU IS DISPLAYED
53
54                  READ.CH     EQU   0E003  SAME AS R.CH BUT WITHOUT ECHO AND WITHOUT INTER-
55                  *******                  PRETATION OF A
56
57                  BREAK       EQU   0C046
58                  *****       READ A CHARACTER FROM THE HP (WITHOUT POLLING)
59                  *           RESULT:      A = CHARACTER READ FROM THE UART
60                  *                           = 0 IF NO KEY WAS PRESSED
61                  *           SIDE EFFECT: SEE R.CH
62
63                  W.CH        EQU   0C051
64                  ****        WRITE A CHARACTER TO THE HP
65                  *
66                  *           ARGUMENT:    A = CHARACTER TO BE WRITTEN
67                  *           RESULT:      -
68                  *           SIDE EFFECT: W.CH POLLS UNTIL IT MAY SEND THE CHARACTER
```

```
                W.1.BLANK   EQU   0C062
                W.2.BLANK   EQU   0C060
                W.3.BLANK   EQU   0C05E
                *********   WRITE N BLANKS TO THE HP

                W.STRING    EQU   0C06A
                ********    WRITE A STRING TO THE HP
                *
                *           ARGUMENT:    X = ADDRESS OF THE STRING TO BE WRITTEN
                *                            (TERMINATED BY A 0-BYTE)
                *           RESULT:      -
                *           SIDE EFFECT: POLLING MAY OCCUR, SINCE W.STRING CALLS W.CH
                *                        X RETURNS THE ADDRESS OF THE 0-BYTE TERMINATING
                *                        THE STRING +1

                W.CRLF      EQU   0C078
                ******      WRITE A CARRIAGE RETURN / LINE FEED TO THE HP
                *
                *           ARGUMENT:    -
                *           RESULT:      -

                NO          EQU   0C097
                **          WRITE AN EXCLAMATION POINT TO THE HP

                IN.HEX      EQU   0C085
                ******      INPUT A HEXADECIMAL DIGIT FROM THE HP (WITH ECHO)
                *
                *           ARGUMENT:    -
                *           RESULT:      A = FOUR BIT VALUE OF THE CHARACTER READ
                *           SIDE EFFECT: IN.HEX READS CHARACTERS FROM THE HP UNTIL A
                *                        VALID HEX DIGIT IS ENCOUNTERED; CHARACTERS > '9'
                *                        ARE CONVERTED TO CAPS

                READ.HEX    EQU   0E00D  IN.HEX WITHOUT ECHO
                ********

                OUT.HEX     EQU   0C090
                *******     OUTPUT A HEXADECIMAL DIGIT TO THE HP
                *
                *           ARGUMENT:    A = FOUR BIT BINARY VALUE OF THE HEX. DIGIT TO BE
                *                            WRITTEN
                *           RESULT:      -

                IN.BYTE     EQU   0C0BC
                *******     INPUT TWO HEX DIGITS FROM THE HP INTERPRETING THEM AS A BYTE
                *                                                       (WITH ECHO)
                *           ARGUMENT:    -
                *           RESULT:      A = EIGHT BIT VALUE FORMED FROM THE TWO CHARACTERS
                *                            READ
                *           SIDE EFFECT: IN.BYTE READS CHARACTERS FROM THE HP UNTIL TWO VALID
                *                        HEX DIGITS ARE ENCOUNTERED

                READ.BYTE   EQU   0E01A  IN.BYTE WITHOUT ECHO
                *********
```

```
125                OUT.BYTE    EQU   0C0C9
126                ********    OUTPUT AN EIGHT BIT NUMBER REPRESENTED BY TWO HEXADECIMAL DIGITS
127                *
128                *           ARGUMENT:     A = THE BYTE TO BE WRITTEN TO THE HP
129                *           RESULT:       -
130
131                IN.WORD     EQU   0C0D8
132                *******     INPUT FOUR HEX DIGITS FROM THE HP INTERPRETING THEM AS A SIXTEEN
133                *           BIT NUMBER (WITH ECHO)
134                *
135                *           ARGUMENT:     X = ADDRESS OF THE MEMORY LOCATION WHERE THE WORD
136                *                             READ HAS TO BE STORED
137                *           RESULTS:      M[X] AND M[X+1] CONTAIN THE DATA READ
138                *           SIDE EFFECT:  IN.WORD READS CHARACTERS FROM THE HP UNTIL FOUR
139                *                         VALID HEX DIGITS ARE ENCOUNTERED
140
141                READ.WORD   EQU   0E027   IN.WORD WITHOUT ECHO
142                *********
143
144                OUT.WORD    EQU   0C0E3
145                ********    OUTPUT A 16-BIT NUMBER REPRESENTED BY FOUR HEX DIGITS
146                *
147                *           ARGUMENTS:    X = ADDRESS OF THE MEMORY LOCATION CONTAINING THE
148                *                             WORD TO BE WRITTEN
149                *                         M[X] = HIGH BYTE,
150                *                         M[X+1] = LOW BYTE OF THE WORD
151                *           RESULT:       -
152
153                FAST.READ   EQU   0C0EE
154                *********   INITALIZE THE HP READING ITS CASSETTE AND TRANSFERING ONE FILE
155
156                LINE.READ   EQU   0C0FB
157                *********   INITIALIZE THE HP READING ITS CASSETTE AND SENDING ONE LINE
158
159
```

```
161                MENU        EQU   0C336
162                ****        WRITE THE MENU POINTED TO BY ACT.MENU ON THE SCREEN
163                *
164                *           ARGUMENT:    ACT.MENU = ADDRESS OF THE MENU (ASCII STRINGS
165                *                                   AND ENTRYPTS)
166
167                GET.KEY     EQU   0C352
168                *******     PROMPT WITH CR/LF AND A '*', CALL GETWORD AND REPEAT THEN
169                *           ACCEPTING A 'KEY' UNTIL (SKIPPING BLANKS) THIS IS THE FIRST CHA-
170                *           RACTER OF ONE OF THE LINES OF THE MENU. CONVERT ALL CHARACTERS
171                *           TO CAPS. JUMP THEN TO THE ROUTINE IDENTIFIED BY THIS 'KEY'.
172
173                *
174                *           ARGUMENT:    ACT.SUB.MON = ADDRESS OF THE MENU
175                *           RESULTS:     X.WORD (AT LOCATION 0F) = THE ADDRESS ENTERED BEFOR
176                *                                                  THE CHARACTER
177                *                        CARRY = FALSE IF NO ADDRESS AT ALL WAS ENTERED
178                *                              = TRUE IF X.WORD CONTAINES AN ADDRESS
179                *           SIDE EFFECT: X IS DESTROID BY GET.KEY
180
181
182                GET.WORD    EQU   0C2FE
183                ********    READ HEX DIGITS FROM THE HP SKIPPING BLANKS AT THE BEGINNIG INTER-
184                *           PRETING THEM AS A NUMBER UNTIL A NONNUMERICAL CHARACTER IS
185                *           ENCOUNTERED
186                *           RESULTS:     X.WORD (AT ADDRESS 0F) = BINARY VALUE OF THE LAST
187                *                                                 4 HEX DIGITS ENTERED
188                *                        A = THE NONHEX CHARACTER WHICH TERMINATED THE INPUT
189                *                        CARR (AT LOCATION 11) = 0 IF THE FIRST CHARACTER
190                *                                                ENTERED WAS NONHEXADECIMAL,
191                *                                              = FF IF A NUMBER WAS ENTERED
192                *                                                FIRST
193
194                COMMANDS    EQU   0C391  STORE THE CONTENT OF THE X-REGISTER INTO ACT.SUB.MON,
195                ********                 WRITE THE STRING POINTED TO BY MAIN.ID ON THE SCREEN
196                *                        AND REPEAT THEN CALLING GET.KEY.
197                *           ARGUMENT:    X = POINTER TO THE ACTUAL MENU.
198
199                BOOT        EQU   0E2C5  BOOTSTRAP THE PERSONAL COMPUTER
200                ****
201
202                SET.U.PC    EQU   0E2D5  SET THE MICRO PROGRAM COUNTER
203                ********
204
205                LOAD.U.RAM  EQU   0E3B3  LOAD THE MICRO-RAM FROM THE EPROM'S
206                **********
```

```
208                PUSH.X      EQU   0C259
209                ******      STORES X ON STACK
210
211                PULL.X      EQU   0C270
212                ******      RETRIEVES X FROM STACK
213
214                PRINT.X     EQU   0C287
215                *******     PRINTS THE CONTENT OF THE X-REGISTER
216
217                WUM         EQU   0C17B
218                ***         WRITE MICRO MEMORY ROUTINE
219                *           UMEM(X,B):=A
220
221                RUM         EQU   0C190
222                ***         MICRO MEMORY READ ROUTINE
223                *           A:=UMEM(X,B)
224
225                STUPC       EQU   0C1C2
226                *****       STORE MICRO PROGRAM COUNTER
227                *           PREPARES MICROCONTROLLER FOR EXECUTION AT ADDRESS IN X
228                *           UPC:=X
229
230                EXX         EQU   0C117
231                ***         EXECUTES MICRO INSTRUCTION POINTED TO BY X
232                *           X POINTS TO MOST SIGNIFICANT BYTE
233                *           MOST SIGNIFICANT BYTE HAS LOWEST ADDRESS
234                *           X IS INCREMENTED BY 5
235
236                RCPU        EQU   0C1F0
237                ****        READ CPU DATA BUS
238                *           B GETS TOP HALF
239                *           A GETS BOTTOM HALF
240
241                WCPU        EQU   0C1F7
242                ****        WRITE CPU DATA REGISTER
243                *           CPU BUS REGISTER := B*256 + A
244                *           DATA IS GATED TO BUS WHEN SOURCE 13 IS ADDRESSED
245
246                EPMIR       EQU   0C15F
247                *****       ENABLE PERSONAL COMPUTER MIR REGISTER
248
249                EDMIR       EQU   0C158
250                *****       DIAGNOSTIC PANEL MIR REGISTER IS ENABLED
251
252                LDMIR       EQU   0C11D
253                *****       DIAGNOSTIC PANEL MIR REGISTER IS LOADED WITH
254                *           DATA POINTED TO BY X
255                *           MOST SIGNIFICANT BYTE HAS THE LOWEST ADDRESS
256                *           X IS RETURNED INCREMENTED BY 5
257                *           RETURNS WITH DIAGNOSTIC MIR REGISTER ENABLED
258                *           NO CPU CLOCK PULSE IS GIVEN
259                *           USEFUL TO GATE DATE NON-DESTRUCTIVELY TO CPU BUS
260
261                MCLK        EQU   0C1AC
262                ****        SEND ONE MICRO CLOCK PULSE (DISABLE CPU CLOCK)
```

```
                RMM         EQU   0C1FE
                ***         READ MAIN MEMORY
                *           B:= MAIN(X)DIV 256
                *           A:= MAIN(X)MOD 256

                RMMS        EQU   0C211
                ****        SHORT (QUICK) VERSION OF RMM
                *           MAY ONLY BE USED AFTER A LONG VERSION CALL HAS BEEN MADE AND
                *           NO OTHER DMIR OPERATIONS HAVE OCCURED

                WMM         EQU   0C228
                ***         WRITE MAIN MEMORY
                *           MAIN(X):=256*B + A

                WMMS        EQU   0C23D
                ****        SHORT VERSION OF WMM. SEE RMMS COMMENTS

                LATCH       EQU   0C151
                *****       MICRO MEMORY ADDRESS REGISTER IS LATCHED

                UNLCH       EQU   0C14A
                *****       MICRO MEMORY ADDRESS REGISTER IS UNLATCHED

                D2911       EQU   0C13E
                *****       MICRO MEMORY ADDRESS OUTPUTS FROM PERSONAL COMPUTER
                *           ARE DISABLED AND ADDRESS REGISTER FROM DIAGNOSTIC
                *           PANEL IS ENABLED

                E2911       EQU   0C174
                *****       REVERSE OF D2911

                STOP        EQU   0C16D
                ****        DISABLE CPU CLOCK AND ENABLE SINGLE STEP MODE

                START       EQU   0C166
                *****       ENABLE CPU CLOCK

                RESET       EQU   0C290
                *****       RESET COMPUTER; INITIATE STAT0 VECTOR

                READ.REG    EQU   0C2DD
                ********    READ A 2901 REGISTER
                *           ARGUMENT:  B = NR OF THE REGISTER  ( 0 .. 0F OR Q.REG)
                *           RESULT:    X = THE VALUE

                READ.R      EQU   0C2A5
                ******      MIR USED BY THE READ.REG

                LOAD.REG    EQU   0C2AA
                *******     LOAD A 2901 REGISTER
                *           ARGUMENTS: B = NR OF THE REGISTER  ( 0..0F OF Q.REG)
                *                      X = VALUE TO BE LOADED

                WRITE.R     EQU   0C2A0
                *******     MIR USED BY LOAD.REG
```

```
321                 PRINT.BUS   EQU   0E1AA
322                 *********         PRINTS AN ERRORMESSAGE AND THE REGISTERS WITH THE VALUE OF
323                 *                 THE CPU-BUS INSTEAD OF THE REGISTERS B,A
324
325
326
327
328
329
330                 *           ASCII CHARACTERS
331                 *
332                 LF          EQU   0A
333                 CR          EQU   0D
334                 DC1         EQU   11
335                 ESC         EQU   1B
336                 RS          EQU   1E
337                 EXCLAM      EQU   21
338                 AMPERSAND   EQU   26
339                 LOWER.E     EQU   65
340                 LOWER.P     EQU   70
341                 RUBOUT      EQU   7F
342                 BLANK       EQU   20
343                 LOWER.U     EQU   75
344                 QUESTION.M  EQU   3F
345                 PROMPT      EQU   2A
346                 COMMA       EQU   2C
347                 BACK.SLASH  EQU   5C
348
349                 *           I/O ADDRESSES
350                 *
351                 HP.STATUS   EQU   4000
352                 HP.DATA     EQU   4001
353
354                 MIR4        EQU   3C08
355                 MIR3        EQU   3C09
356                 MIR2        EQU   3C0A
357                 MIR1        EQU   3C0B
358                 MIR0        EQU   3C0C
359                 CPU1        EQU   3C0E
360                 CPU0        EQU   3C0F
361                 CON0        EQU   3C13
362                 CON1        EQU   3C12
363                 MEMPU       EQU   3C14
364                 SSP         EQU   3C15
365                 UAR1        EQU   3C10
366                 UAR0        EQU   3C11
367
368                 Q.REG       EQU   10
369                 STACK       EQU   07F
370                 RET         EQU   2000 (JMP   0)          'RETURN' - ADDRESS FOR PUSHX/PULLX
371
372                 *           EXTERNAL ENTRY POINTS
373                 *
374                 MONITOR     EQU   0E000
```

```
376                     *           VARIABLES
377
378                     ACT.MENU    EQU    9 (WORD  0)          ADDRESS OF THE ACTUAL MENU
379                     MAIN.ID     EQU   0B (WORD  0)
380                     SUB.ID      EQU   0D (WORD  0)
381                     C           EQU    0 (BYTE  0)          DUMMY REGISTERS,
382                     X.SAVE      EQU    1 (WORD  0)          FOR STRICTLY LOCAL USE ONLY
383                     ADDR        EQU   18 (WORD  0)          USED BY DUMP AND DUMP.LINE
384                     X.WORD      EQU   0F (WORD  0)          SAVE WORD FOR X, USED BY GET.WORD,
385                     *                                       AND MENU
386                     CARR        EQU   11 (BYTE  0)          'CARRY'-BYTE (RESULT OF GET.WORD)
387                     CHECK.SUM   EQU   17 (BYTE  0)          USED BY READBLOCK (LOAD)
388                     STAT0       EQU    3 (BYTE  0)
389
390                     MTMP4       EQU    4 (BYTE  0)
391                     MTMP3       EQU    5 (BYTE  0)
392                     MTMP2       EQU    6 (BYTE  0)
393                     MTMP1       EQU    7 (BYTE  0)
394                     MTMP0       EQU    8 (BYTE  0)
395
396                     TYPING      EQU   16 (BYTE  1)          CONTROLS TYPING ERRORMESSAGES
397                     PSEUDO.IRQ  EQU   21
398                     XSUM        EQU   25 (BYTE  0)          CHECKSUM (MICRO)
399                     CRNT.ADR    EQU   26 (WORD  0)          CURRENT ADR (MICRO, MAIN MEMORY)
400
401                     USER.DATA   SET   2C                    NEXT FREE LOCATION FOR DATA IN THE
402                     *                                       ZERO-PAGE
403
404
405
406
407  0000                           ORG   USER.DATA          IS SPECIFIED ON ROM FILE
408                     *           VARIABLES
409                     *           VARIABLES USED IN MAIN MEMORY
410  002C  00 00        VALUE       WORD  0                  USED IN INSPECT, LOAD
411  002E  00           CL          BYTE  0                  CHECK LOAD FLAG
412  002F  00           COUNT       BYTE  0
413                     ADR         EQU   *
414  0030  00 00        HELP1       WORD  0
415  0032  00 00        HELP2       WORD  0
416  0034  00 00        HELP3       WORD  0
417                     *
418                     *
419                                 USE   EXEC,MAINMEM,DEVICES,MICROMEM
420  0036                           ORG   2003       BEGIN OF RAM
421  2003  7E 22 15                 JMP   MAINMEM
422  2006  7E 23 DF                 JMP   EXEC
423  2009  7E 25 5D                 JMP   DEVICES
424  200C  7E E0 00                 JMP   MONITOR
425  200F  7E 27 73                 JMP   MICROMEM
```

```
428  2012                        BEGIN MAINMEM
429                              DEF   MAINMEM,XINSPECT,BYTE.INSP
430                              USE   MMLOAD,LOAD,CHECK.LOAD
431                   *
432                   *
433                   *          FOLLOWING ROUTINES ARE SUPPLIED (SEE ALSO MENU)
434                   *          I INSPECT MAIN MEMORY AND CHANGE IT
435                   *          L LOAD MAIN MEMORY FROM TAPE
436                   *
437                   *
438                   ************************************************
439                   *
440                   *          INSPECT AND CHANGE MAIN MEMORY
441                   *
442                   INSPECT     EQU   *
443                   XINSPECT    EQU   *       WILL BE USED IN MONITOR EXECUTER
444  2012                         BEGIN INSPECT
445                   *
446                   *          FORMAT
447                   *          ADDRESS 'I' VALUE [ ':' NEWVALUE ] [',']
448                   *          ADDRESS  = UP TO 4 HEX DIGIT
449                   *          NEWVALUE = UP TO 4 HEX DIGIT
450                   *
451                   *          PROCEDURE INSPECT;
452                   *            BEGIN X := X.WORD;
453                   *             LOOP
454                   *               CRNT.ADR := X; (A,B) := RMM(X);
455                   *               VALUE := (A,B); OUT.WORD(VALUE);
456                   *               R.CH(A);
457                   *               IF A =':' THEN
458                   *                 GET.WORD; (A,B) := X.WORD; X := CRNT.ADR; WMM(A,B,X);
459                   *               END;
460                   *               IF A = BACKSLASH
461                   *               THEN
462                   *                 DEC(X);
463                   *               ELSIF A = ','
464                   *               THEN
465                   *                 INC(X);
466                   *               ELSE
467                   *                 EXIT
468                   *               END;
469                   *               CRNT.ADR := X;
470                   *               CRLF; OUT.WORD(CRNT.ADR); WRITE(' ')
471                   *             END
472                   *            END
473                   *
474  2012  FE 3C 10               LDX   UAR1      SAVE UADR
475  2015  DF 34                  STX   HELP3
476  2017  DE 0F                  LDX   X.WORD
477  2019  DF 26                  STX   CRNT.ADR
478  201B  BD C1 58               JSR   EDMIR
479  201E  BD C1 FE  LOOP         JSR   RMM
480  2021  D7 2C                  STAB  VALUE       HIGHER PART
481  2023  97 2D                  STAA  VALUE+1     LOWER PART
482  2025  CE 00 2C               LDX   =VALUE
483  2028  BD C0 E3               JSR   OUT.WORD
```

```
484  202B  BD C0 24            JSR   R.CH
485  202E  81 3A               CMPA  =':'
486  2030  26 0E               BNE   ENDIF1
487  2032  BD C2 FE            JSR   GET.WORD
488  2035  36                  PSH   A           SAVE DELIMITER
489  2036  D6 0F               LDAB  X.WORD      HIGHER PART
490  2038  96 10               LDAA  X.WORD+1    LOWER PART
491  203A  DE 26               LDX   CRNT.ADR
492  203C  BD C2 28            JSR   WMM
493  203F  32                  PUL   A           GET DELIMITER BACK
494  2040  81 2C     ENDIF1    CMPA  =COMMA
495  2042  27 0D               BEQ   ENDIF2
496  2044  81 5C               CMPA  =BACK.SLASH
497  2046  27 0E               BEQ   ENDIF3
498  2048  DE 34               LDX   HELP3     RELOAD UADR
499  204A  BD C1 C2            JSR   STUPC
500  204D  BD C1 5F            JSR   EPMIR
501  2050  39                  RTS
502
503  2051  DE 26     ENDIF2    LDX   CRNT.ADR
504  2053  08                  INX
505  2054  20 03               BRA   ENDIF
506  2056  DE 26     ENDIF3    LDX   CRNT.ADR
507  2058  09                  DEX
508  2059  DF 26     ENDIF     STX   CRNT.ADR
509  205B  BD C0 78            JSR   W.CRLF
510  205E  BD C2 87            JSR   PRINT.X
511  2061  BD C0 62            JSR   W.1.BLANK
512  2064  20 B8               BRA   LOOP
513  2066                      END   INSPECT
```

```
                        BYTEINSP    EQU  *
2066                                BEGIN  BYTE
                        *
                        *           INSPECT AND CHANGE MAIN MEMORY IN BYTES
                        *           FORMAT     SIMILAR TO NORMAL INSPECT
                        *
                        *           PROCEDURE BYTEINSPECT
                        *             VAR HELP1: BOOLEAN; (* 0=TRUE,FF=FALSE *)
                        *             BEGIN AB:=X.WORD
                        *               CRNT.ADR:= AB/2; HELP1 := AB/2 MOD 2 = 0;
                        *               LOOP
                        *                 A,B := RMM(CRNT.ADR); HELP2 := A,B;
                        *                 IF HELP1=0 THEN OUTBYTE(A) ELSE OUTBYTE(B) END;
                        *                 R.CH(A);
                        *                 IF CH = ':' THEN GETWORD;
                        *                   IF HELP1=0 THEN HELP2(LOW):=X.WORD(LOW)
                        *                              ELSE HELP2(HIGH):=X.WORD(LOW)
                        *                   END;
                        *                   WMM(HELP2,CRNT.ADR);
                        *                 END
                        *                 IF CH = BACK.SLASH
                        *                 THEN
                        *                   HELP1 := NOT HELP1;
                        *                   IF NOT HELP1 THEN DEC(CRNT.ADR) END;
                        *                 ELSIF CH = ','
                        *                 THEN
                        *                   HELP1 := NOT HELP1;
                        *                   IF HELP1 THEN INC(CRNT.ADR) END;
                        *                 ELSE
                        *                   EXIT
                        *                 END
                        *               END;
                        *             END BYTEINSPECT;
                        *
2066  FE 3C 10                      LDX   UAR1
2069  DF 34                        STX   HELP3
206B  7F 00 30                     CLR   HELP1
206E  96 10                        LDAA  X.WORD+1
2070  D6 0F                        LDAB  X.WORD      HIGHER BYTE
2072  54                           LSR    B
2073  46                           ROR    A
2074  24 03                        BCC   BYTINS1
2076  73 00 30                     COM   HELP1
2079  D7 26              BYTINS1   STAB  CRNT.ADR
207B  97 27                        STAA  CRNT.ADR+1
207D  BD C1 58                     JSR   EDMIR

2080  DE 26              BLOOP     LDX   CRNT.ADR
2082  BD C1 FE                     JSR   RMM
2085  36                           PSH   A
2086  17                           TBA
2087  33                           PUL   B
2088  D7 32                        STAB  HELP2
208A  97 33                        STAA  HELP2+1
208C  7D 00 30                     TST   HELP1
208F  27 01                        BEQ   BYTINS2
```

```
571  2091  17                    TBA
572  2092  BD C0 C9  BYTINS2     JSR   OUT.BYTE
573  2095  BD C0 24              JSR   R.CH
574  2098  81 3A                 CMPA  =':'
575  209A  26 1B                 BNE   IF
576  209C  BD C2 FE              JSR   GETWORD
577  209F  36                    PSH   A           SAVE DELIMITER
578  20A0  96 10                 LDAA  X.WORD+1    LOWER BYTE
579  20A2  7D 00 30              TST   HELP1
580  20A5  26 04                 BNE   BYTINS3
581  20A7  97 33                 STAA  HELP2+1     CHANGE LOWER BYTE
582  20A9  20 02                 BRA   BYTINS4
583  20AB  97 32     BYTINS3     STAA  HELP2       CHANGE UPPER BYTE
584  20AD  D6 33     BYTINS4     LDAB  HELP2+1
585  20AF  96 32                 LDAA  HELP2
586  20B1  DE 26                 LDX   CRNT.ADR
587  20B3  BD C2 28              JSR   WMM
588  20B6  32                    PUL   A           RESAVE DELIMITER
589  20B7  81 5C     IF          CMPA  =BACK.SLASH
590  20B9  26 0A                 BNE   ELSIF
591  20BB  73 00 30              COM   HELP1
592  20BE  27 C0                 BEQ   BLOOP
593  20C0  DE 26                 LDX   CRNT.ADR
594  20C2  09                    DEX
595  20C3  20 0C                 BRA   THEN
596  20C5  81 2C     ELSIF       CMPA  =COMMA
597  20C7  26 0C                 BNE   ELSE
598  20C9  73 00 30              COM   HELP1
599  20CC  26 B2                 BNE   BLOOP
600  20CE  DE 26                 LDX   CRNT.ADR
601  20D0  08                    INX
602  20D1  DF 26     THEN        STX   CRNT.ADR
603  20D3  20 AB                 BRA   BLOOP
604  20D5  DE 34     ELSE        LDX   HELP3
605  20D7  BD C1 C2              JSR   STUPC
606  20DA  BD C1 5F              JSR   EPMIR
607  20DD  39                    RTS
608  20DE                        END   BYTE
```

```
610  20DE                    BEGIN LOADER
611                          DEF   MMLOAD,LGETWORD
612                  *       ABSOLUTE LOADER
613                  *
614                  *       LOAD MAIN MEMORY FROM TAPE
615                  *       FORMAT ON TAPE:
616                  *       'NR OF WORDS' 'LOAD ADR' ! INFO " 'CHSUM'
617                  *       ALL IS LOADED IN WORDS
618                  *       EACH WORD IS CODED:
619                  *         WORD := (CH1-'A') + 16*(CH2-'A') + 256*(CH3-'A') * 4096*(CH4-'A')
620                  *
621                  *       PROCEDURE GETBYTE(VAR A:BYTE); EXTERN; (* DEFINED IN MICRO MEMORY *)
622                  *
623                  *       PROCEDURE LGETWORD;
624                  *         (* CONVERT 4 CHARACTER FROM TAPE INTO ONE 16BIT WORD *)
625                  *         BEGIN
626                  *           GETBYTE; VALUE1 := A;
627                  *           GETBYTE; VALUE0 := A;
628                  *           LINESUM := LINESUM + VALUE;
629                  *         END;
630                  *
631                  *       PROCEDURE MMLOAD;
632                  *         BEGIN
633                  *           LINE.READ;
634                  *           LOOP LINESUM := 0;
635                  *             LGETWORD;
636                  *             IF VALUE = 0 THEN EXIT END;
637                  *             B := VALUE; (* COUNTER*);
638                  *             LGETWORD; CRNT.ADR := X := VALUE;
639                  *             REPEAT PUSH(B);
640                  *               LGETWORD;
641                  *               WMM(VALUE,X);
642                  *               INC(X); POP(B); DEC(B);
643                  *             UNTIL B=0;
644                  *             LGETWORD; (*CHSUM*)
645                  *             IF LINESUM<>0 THEN W.CH(QUESTIONMARK); EXIT END;
646                  *           END
647                  *           G.REG := M[1];
648                  *           SUM(GLOBSUM);
649                  *         END
650                  *
```

```
652                     LINESUM EQU CHECK.SUM
653
654
655  20DE  BD E0 03  GETCH       JSR   READ.CH
656  20E1  81 2E                 CMPA  ='.'
657  20E3  26 19                 BNE   L3
658
659  20E5  BD E0 03  L1          JSR   READ.CH
660  20E8  81 0D                 CMPA  =CR
661  20EA  26 F9                 BNE   L1
662  20EC  BD C0 FB              JSR   LINEREAD
663
664  20EF  BD E0 03  L2          JSR   READ.CH
665  20F2  81 0D                 CMPA  =CR
666  20F4  26 F9                 BNE   L2
667  20F6  31                    INS
668  20F7  31                    INS
669  20F8  31                    INS
670  20F9  31                    INS
671  20FA  31                    INS
672  20FB  31                    INS
673  20FC  31                    INS
674  20FD  39                    RTS
675
676  20FE  81 0D     L3          CMPA  =CR
677  2100  26 05                 BNE   L4
678  2102  BD C0 FB              JSR   LINE.READ
679  2105  20 D7                 BRA   GETCH
680
681  2107  81 41     L4          CMPA  ='A'
682  2109  2D D3                 BLT   GETCH
683  210B  80 41                 SUBA  ='A'
684  210D  39                    RTS
685
686  210E  37        GET.BYTE    PSH   B
687  210F  8D CD                 BSR   GET.CH
688  2111  16                    TAB
689  2112  8D CA                 BSR   GET.CH
690  2114  48                    ASL   A
691  2115  48                    ASL   A
692  2116  48                    ASL   A
693  2117  48                    ASL   A
694  2118  1B                    ABA
695  2119  16                    TAB
696  211A  33                    PUL   B
697  211B  39                    RTS
```

```
699  211C  BD 21 0E  LGETWORD    JSR   GETBYTE
700  211F  97 2D                 STAA  VALUE+1     LOWER PART
701  2121  BD 21 0E              JSR   GETBYTE
702  2124  97 2C                 STAA  VALUE       UPPER PART
703  2126  96 2D                 LDAA  VALUE+1     ADD ; LOWER
704  2128  9B 18                 ADDA  LINESUM+1
705  212A  97 18                 STAA  LINESUM+1
706  212C  96 2C                 LDAA  VALUE       UPPER PART  WITH CARRY
707  212E  99 17                 ADCA  LINESUM
708  2130  97 17                 STAA  LINESUM
709  2132  39                    RTS
710                      *
711  2133  BD C0 FB  MMLOAD      JSR   LINE.READ
712  2136  4F         LOOP       CLR   A
713  2137  97 17                 STAA  LINESUM     LINESUM := 0
714  2139  97 18                 STAA  LINESUM+1
715  213B  BD 21 1C              JSR   LGETWORD
716  213E  D6 2D                 LDAB  VALUE+1
717  2140  5D                    TST    B          NR OF WORDS HAS BYTE SIZE
718  2141  27 2C                 BEQ   ENDLOOP
719  2143  BD 21 1C              JSR   LGETWORD   GET LOAD ADR
720  2146  DE 2C                 LDX   VALUE
721  2148  DF 26                 STX   CRNT.ADR
722  214A  BD C1 58              JSR   EDMIR
723  214D  37         REP        PSH   B
724  214E  BD 21 1C              JSR   LGETWORD
725  2151  96 2C                 LDAA  VALUE       LOWER PART
726  2153  D6 2D                 LDAB  VALUE+1     UPPER PART
727  2155  BD C2 28              JSR   WMM         WRITE MAIN MEMORY
728  2158  08                    INX
729  2159  33                    PUL   B
730  215A  5A                    DEC   B
731  215B  26 F0                 BNE   REP
732                      *      STX   HIGH.M.ADR
733  215D  BD 21 1C              JSR   LGETWORD     CHECKSUM
734  2160  7D 00 17              TST   LINESUM
735  2163  26 05                 BNE   XSUMERR     CHECKSUM ERROR
736  2165  7D 00 18              TST   LINESUM+1
737  2168  27 CC                 BEQ   LOOP        OK GO ON
738  216A  86 3F      XSUMERR    LDAA  =QUESTION.M
739  216C  BD C0 51              JSR   W.CH
740  216F  BD C1 5F  ENDLOOP     JSR  EPMIR
741  2172  39                    RTS
742  2173                        END   LOADER
```

```
744  2173                          BEGIN LOAD
745                                DEF   LOAD,CHECK.LOAD
746
747  2173  BD E0 03  SKIP          JSR   READ.CH      SKIP A LINE
748  2176  81 0D                   CMPA  =CR
749  2178  26 F9                   BNE   SKIP
750  217A  39                      RTS
751
752                   READ.LINE    EQU   *
753  217B  BD C0 FB                JSR   LINE.READ    START TAPE
754
755  217E  BD E0 03  REPEAT        JSR   READ.CH
756  2181  81 53                   CMPA  ='S'
757  2183  26 F9                   BNE   REPEAT       SKIP UNTIL 'S'
758
759  2185  BD E0 03                JSR   READ.CH
760  2188  81 30                   CMPA  ='0'
761  218A  26 04                   BNE   NOT.HEAD
762  218C  8D E5                   BSR   SKIP         HEADER
763  218E  20 EB                   BRA   READ.LINE
764
765  2190  81 39      NOT.HEAD     CMPA  ='9'
766  2192  26 04                   BNE   NOT.EOF
767  2194  8D DD                   BSR   SKIP         EOF
768  2196  20 6B                   BRA   EO.LAST.LN   (A=CR)
769
770  2198  81 32      NOT.EOF      CMPA  ='2'
771  219A  26 61                   BNE   ERR
772
773  219C  BD E0 1A                JSR   READ.BYTE
774  219F  97 2F                   STAA  COUNT        BYTE COUNT
775  21A1  97 17                   STAA  CHECK.SUM    !
776
777  21A3  7A 00 2F                DEC   COUNT
778  21A6  27 4D                   BEQ   XS.CHECK
779
780  21A8  CE 00 26                LDX   =CRNT.ADR
781  21AB  BD E0 27                JSR   READ.WORD    LOAD ADDRESS
782
783  21AE  AB 00                   ADDA  0,X
784  21B0  AB 01                   ADDA  1,X
785  21B2  97 17                   STAA  CHECK.SUM
```

```
787  21B4  DE 26              LDX   CRNT.ADR
788  21B6  BD C1 58           JSR   EDMIR
789  21B9  7A 00 2F  WHILE    DEC   COUNT
790  21BC  7A 00 2F           DEC   COUNT
791  21BF  27 34              BEQ   XS.CHECK
792
793  21C1  BD E0 1A           JSR   READ.BYTE
794  21C4  16                 TAB
795  21C5  BD E0 1A           JSR   READ.BYTE
796
797  21C8  7D 00 2E           TST   CL
798  21CB  27 1D              BEQ   LOADING
799  21CD  D7 2C     TESTING  STAB  VALUE
800  21CF  97 2D              STAA  VALUE+1
801  21D1  BD C1 FE           JSR   RMM
802  21D4  D1 2C              CMPB  VALUE
803  21D6  26 04              BNE   ERROR
804  21D8  91 2D              CMPA  VALUE+1
805  21DA  27 11              BEQ   OK
806  21DC  36        ERROR    PSH   A
807  21DD  8D 94              BSR   SKIP
808  21DF  32                 PUL   A
809  21E0  3F                 SWI
810  21E1  DE 2C              LDX   VALUE
811  21E3  BD C0 60           JSR   W.2.BLANK
812  21E6  BD C2 87           JSR   PRINT.X
813  21E9  39                 RTS
814
815  21EA  BD C2 28  LOADING  JSR   WMM
816
817  21ED  9B 17     OK       ADDA  CHECK.SUM
818  21EF  1B                 ABA
819  21F0  97 17              STAA  CHECK.SUM
820  21F2  08                 INX
821  21F3  20 C4              BRA   WHILE
822
823  21F5  BD E0 1A  XS.CHECK JSR   READ.BYTE
824  21F8  43                 COM   A
825  21F9  91 17              CMPA  CHECK.SUM
826  21FB  27 05              BEQ   EOL
827
828  21FD  96 3F     ERR      LDAA  QUESTION.M
829  21FF  BD C0 51           JSR   W.CH
830
831  2202  4F        EOL      CLR   A           A#0 <=> EOF
832  2203  BD C1 5F  EO.LAST.LN JSR  EPMIR
833  2206  39                 RTS
834
835
836  2207  97 2E     CHECK.LOAD    STAA  CL
837  2209  20 03              BRA   LOOP
838
839  220B  7F 00 2E  LOAD     CLR   CL
840
841  220E  BD 21 7B  LOOP     JSR   READ.LINE
842  2211  4D                 TST   A
```

SUBMONIT MAINMEM LOAD

```
843  2212  27 FA              BEQ   LOOP
844  2214  39                  RTS
845
846  2215                      END   LOAD
```

SUBMONIT MAINMEM

```
848  2215  CE 22 35  MAINMEM     LDX   =M.TITLE
849  2218  DF 0B                 STX   MAIN.ID
850  221A  96 03                 LDAA  STAT0
851  221C  84 BF                 ANDA  =0BF            STOP RUNNING CLOCK   %%  =0FD
852  221E  97 03                 STAA  STAT0
853  2220  B7 3C 13              STAA  CON0
854  2223  BD C1 58              JSR   EDMIR           EXECUTE ONE INSTRUCTION TO PUT
855  2226  CE C2 A5              LDX   =READ.R              MEMORY IN A DEFINED STATE
856  2229  BD C1 17              JSR   EXX
857  222C  BD C1 5F              JSR EPMIR
858  222F  CE 22 41              LDX   =MAINMMENU
859  2232  7E C3 91              JMP   COMMANDS
860
861  2235  4D 41 49  M.TITLE     BYTE  'MAIN MEMORY',0
     2238  4E 20 4D
     223B  45 4D 4F
     223E  52 59 00
862
863  2241  05        MAINMMENU   BYTE  5
864  2242  49 20 49              BYTE  'I INSPECT',0
     2245  4E 53 50
     2248  45 43 54
     224B  00
865  224C  20 12                 WORD  INSPECT
866  224E  4C 20 4C              BYTE  'L LOAD',0
     2251  4F 41 44
     2254  00
867  2255  22 0B                 WORD  LOAD
868  2257  4F 20 4F              BYTE  'O OLD FORMAT LOAD',0
     225A  4C 44 20
     225D  46 4F 52
     2260  4D 41 54
     2263  20 4C 4F
     2266  41 44 00
869  2269  21 33                 WORD  MMLOAD
870  226B  52 20 52              BYTE  'R REREAD',0
     226E  45 52 45
     2271  41 44 00
871  2274  22 07                 WORD  CHECK.LOAD
872  2276  59 20 42              BYTE  'Y BYTE INSPECT',0
     2279  59 54 45
     227C  20 49 4E
     227F  53 50 45
     2282  43 54 00
873  2285  20 66                 WORD  BYTE.INSP
874
875  2287                        END   MAINMEM
```

```
878  2287                        BEGIN  EXECUTER
879                              USE    XINSPECT,BYTE.INSP,PCTOBUS     FROM MAIN MEMORY MONITOR
880                              DEF    EXEC
881                   *
882                   *          EXECUTE ROUTINES FOR SOFTWARE TESTING
883                   *
884                   *          FOLLOWING ROUTINES ARE SUPPLIED
885                   *          R  RESET THE COMPUTER
886                   *          N  EXECUTE ONE MICROINSTRUCTION
887                   *             AND DISPLAY MICROADDRESS AND CPU BUS
888                   *          G  GO - START FREE RUNNING CLOCK
889                   *          H  HALT FREE RUNNING CLOCK
890                   *          U  DISPLAY CURRENT MICRO INSTRUCTION
891                   *          P  SET MICRO PC
892                   *          X  EXECUTE SINGLE MACRO STEP
893                   *
894                   ************************************************************
895                   *
896                   NEXTI       EQU   *
897  2287                         BEGIN  NEXTINSTR
898                   *          PROCEDURE NEXTI
899                   *            BEGIN
900                   *              IF CARRY THEN X := X.WORD
901                   *                LOOP  DEC(X);
902                   *                  IF X=0 THEN EXIT END;
903                   *                  SS;
904                   *                 END;
905                   *              END;
906                   *              WRITE(MICROADR);
907                   *              SS;  (* SINGLE STEP *)
908                   *              WRITE(' ');
909                   *              WRITE(CPU BUS);
910                   *            END;
911                   *
912  2287  24 0A                    BCC   END1
913  2289  DE 0F                    LDX   X.WORD
914  228B  09              LOOP     DEX
915  228C  27 05                    BEQ   END1
916  228E  B7 3C 15                 STAA  SSP              SINGLE STEP
917  2291  20 F8                    BRA   LOOP
918  2293  FE 3C 10  END1            LDX   UAR1              GET MICRO ADR
919  2296  BD C2 87                 JSR   PRINTX
920  2299  B7 3C 15                 STAA  SSP               MAKES A CPU CLOCK
921  229C  BD C0 62                 JSR   W.1.BLANK
922  229F  FE 3C 0E                 LDX   CPU1
923  22A2  BD C2 87                 JSR   PRINTX
924  22A5  39                       RTS
925  22A6                           END   NEXTINSTR
926                   *
927                   ************************************************
928                   DMICROINS   EQU   *
929                   *          DISPLAY CURRENT MICRO INSTRUCTION
930                   *
931                   *+         PROCEDURE DISPLAYMICROINSTRUCTION
932                   *          BEGIN
933                   *            WRITE(UAR,' ');
```

```
934                     *             FOR I:= 5 DOWNTO 1 DO WRITE(MIR[I])
935                     *           END
936                     *
937  22A6  BD C0 62             JSR   W.1.BLANK
938  22A9  FE 3C 10             LDX   UAR1
939  22AC  BD C2 87             JSR   PRINTX
940  22AF  BD C0 62             JSR   W.1.BLANK
941  22B2  C6 05                LDAB  =5
942  22B4  CE 3C 08             LDX   =MIR4
943  22B7  A6 00      DMILOOP   LDAA  0,X
944  22B9  BD C0 C9             JSR   OUT.BYTE
945  22BC  86 20                LDAA  =' '
946  22BE  BD C0 51             JSR   W.CH
947  22C1  08                   INX
948  22C2  5A                   DEC   B
949  22C3  26 F2                BNE   DMILOOP
950  22C5  39                   RTS
951                     ****************************************
952
953  22C6  BD C2 90  RESET1     JSR   RESET
954  22C9  CE 00 01             LDX   =1
955  22CC  DF 0F                STX   X.WORD
956  22CE  7E E2 D5             JMP   SET.U.PC
957
958                     ****************************************
959                     *
960                     *  PROCEDURE REAL.PC;
961                     *    (* GET THE REAL PC := 4*F+OFFSET *)
962                     *    BEGIN MIR := (F.REG -> BUS);
963                     *      HELP1 := CPU1;
964                     *      MIR := (OFFSET -> BUS);
965                     *      HELP1 := 4*HELP1 + BUS; X := HELP1
966                     *    END REAL.PC;
967                     *
968  22D1  CE 22 F6  REAL.PC   LDX       =F.TO.BUS
969  22D4  BD C1 1D            JSR       LDMIR
970  22D7  FE 3C 0E            LDX       CPU1
971  22DA  DF 30               STX       HELP1
972  22DC  86 F2               LDAA      =0F2      SRC := OFFSET; DST := NONE
973  22DE  B7 3C 0C            STAA      MIR0
974  22E1  96 31               LDAA      HELP1+1
975  22E3  D6 30               LDAB      HELP1
976  22E5  48                  ASL       A
977  22E6  59                  ROL       B
978  22E7  48                  ASL       A
979  22E8  59                  ROL       B
980  22E9  BB 3C 0F            ADDA      CPU0
981  22EC  F9 3C 0E            ADCB      CPU1      (B,A) := 4*F.REG + OFFSET
982  22EF  97 31               STAA      HELP1+1
983  22F1  D7 30               STAB      HELP1
984  22F3  DE 30               LDX       HELP1
985  22F5  39                  RTS
986                     *
987  22F6  2E 60 0C  F.TO.BUS  BYTE      2E,60,0C,60,0FE
     22F9  60 FE
988                     *
989                     ************************************
```

```
                                MSS         EQU   *
                                *           EXECUTE SINGLE MACRO STEP
                                *           DSIPLAY MACRO PC, MICRO PC
 22FB                           *           BEGIN MSS
                                *           DEF   PCTOBUS
                                *
                                *           REPEAT
                                *             WHILE MIR0 <> 3 (* BUS SOURCE IR4 *) DO SS END;
                                *             DEC(X.WORD);
                                *           UNTIL X.WORD = 0;
                                *           OUTWORD(REAL.PC); WRITE(' ');
                                *           LDMIR(BUS <- PC);
                                *           OUT.WORD(BUS)
                                *           WRITE(' ');
                                *           EPMIR;
                                *           OUTWORD(UAR1);
                                *           SS;
                                *           W.CRLF
                                *
 22FB  25 05                    MSS1        BCS   MSS1
 22FD  CE 00 01                             LDX   =1            DEFAULT -> 1 MACRO STEP
 2300  DF 0F                                STX   X.WORD

 2302  B6 3C 0C                 MSS1        LDAA  MIR0
 2305  81 03                                CMPA  =3
 2307  27 05                                BEQ   ENDWHILE
 2309  B7 3C 15                             STAA  SSP         MAKE MICRO CLOCK
 230C  20 F4                                BRA   MSS1
 230E  DE 0F                    ENDWHILE    LDX   X.WORD
 2310  09                                   DEX
 2311  DF 0F                                STX   X.WORD
 2313  27 05                                BEQ   CONT
 2315  B7 3C 15                             STAA  SSP
 2318  20 E8                                BRA   MSS1
 231A  BD C1 58                 CONT        JSR   EDMIR
 231D  8D B2                                BSR   REAL.PC
 231F  BD C2 87                             JSR   PRINTX
 2322  BD C0 62                             JSR   W.1.BLANK
 2325  FE 3C 0E                             LDX   CPU1        THERE IS STILL THE OFFSET FROM REAL.PC
 2328  BD C2 87                             JSR   PRINTX
 232B  BD C0 62                             JSR   W.1.BLANK
 232E  BD C1 5F                             JSR   EPMIR
 2331  FE 3C 10                             LDX   UAR1
 2334  BD C2 87                             JSR   PRINTX
 2337  B7 3C 15                             STAA  SSP         MAKE MICRO CLOCK
 233A  39                                   RTS
 233B  2E 00 07                 PCTOBUS     BYTE  2E,0,7,60,02    NO ALU OPERATION, GATE PC TO BUS
 233E  60 02
 2340                                       END   MSS
                                ****************************************
                                *
                                *           PROCEDURE EXECUTE UNTIL GIVEN ADDRESS ENCOUNTERED
                                *             BEGIN
                                *               WHILE PC<>X.WORD DO
                                *                 REPEAT SINGLE STEP UNTIL MIR0=3 (* IR4 *)
                                *               END
                                *             END ;
```

```
1046                    BRPT1       EQU   *
1047  2340  BD C1 58   BRKPOINT    JSR   EDMIR
1048  2343  8D 8C                 BSR   REAL.PC
1049  2345  BD C1 5F               JSR   EPMIR
1050  2348  9C 0F                 CPX   X.WORD
1051  234A  26 01                 BNE   BRPT2
1052  234C  39                    RTS
1053  234D  B7 3C 15   BRPT2      STAA  SSP         SINGLE STEP
1054  2350  B6 3C 0C               LDAA  MIR0
1055  2353  81 03                 CMPA  =3
1056  2355  26 F6                 BNE   BRPT2
1057  2357  20 E7                 BRA   BRPT1
1058                    ********************************************
1059                    *
1060                    *  PROCEDURE TRACE.REG
1061                    *    BEGIN
1062                    *      FOR A := 0 TO 8 DO WRITE('  ',A:1,'  ') END;
1063                    *      WRITE(' M P H S G L T ');
1064                    *      FOR B := 0 TO 16 DO
1065                    *        READREG(B,X); WRITE(X,' ');
1066                    *      END
1067                    *      WRITE(READREG(Q.REG);
1068                    *      WRITE(F.REG);
1069                    *      WRITE(OFFS);
1070                    *      WRITELN(REAL.PC);
1071                    *      WRITE('Q  F  OFFS  PC');
1072                    *    END TRACE.REG
1073                    *
1074  2359  BD C0 78   TRACE.REG JSR       W.CRLF
1075
1076  235C  CE 23 BC             LDX       =TRTABLE
1077  235F  BD C0 60   TRLOOP2   JSR       W.2.BLANK
1078  2362  A6 00                LDAA      0,X
1079  2364  BD C0 51             JSR       W.CH
1080  2367  BD C0 60             JSR       W.2.BLANK
1081  236A  08                   INX
1082  236B  8C 23 CC             CPX       =TRTABLE+16Z
1083  236E  26 EF                BNE       TRLOOP2
1084
1085  2370  BD C1 58             JSR       EDMIR
1086  2373  5F                   CLR       B
1087  2374  BD C2 DD   FOR.TR     JSR       READREG
1088  2377  BD C2 87             JSR       PRINTX
1089  237A  BD C0 62             JSR       W.1.BLANK
1090  237D  5C                   INC       B
1091  237E  C1 10                CMPB      =10
1092  2380  26 F2                BNE       FOR.TR
1093
1094  2382  C6 10                LDAB      =Q.REG
1095  2384  BD C2 DD             JSR       READREG
1096  2387  BD C2 87             JSR       PRINTX
1097  238A  BD C0 62             JSR       W.1.BLANK
1098
1099  238D  86 FE                LDAA      =0FE      SRC=F.REG
1100  238F  B7 3C 0C             STAA      MIR0
1101  2392  CE 3C 0E             LDX       =CPU1
1102  2395  BD C0 E3             JSR       OUT.WORD
```

```
1103  2398  BD C0 62           JSR       W.1.BLANK
1104
1105  239B  86 F2              LDAA      =0F2      SRC=OFFSET
1106  239D  B7 3C 0C           STAA      MIR0
1107  23A0  CE 3C 0E           LDX       =CPU1
1108  23A3  BD C0 E3           JSR       OUT.WORD
1109  23A6  BD C0 62           JSR       W.1.BLANK
1110
1111  23A9  BD 22 D1           JSR       REAL.PC
1112  23AC  BD C2 87           JSR       PRINT.X
1113
1114  23AF  BD C0 78           JSR       W.CRLF
1115  23B2  CE 23 CC           LDX       =SEC.LINE
1116  23B5  BD C0 6A           JSR       W.STRING
1117  23B8  BD C1 5F           JSR       EPMIR
1118  23BB  39                 RTS
1119
1120  23BC  30 31 32  TRTABLE  BYTE      '012345678MPHSGLT'
      23BF  33 34 35
      23C2  36 37 38
      23C5  4D 50 48
      23C8  53 47 4C
      23CB  54
1121  23CC  20 20 51  SEC.LINE BYTE      '  Q    F  OFFS  PC',0
      23CF  20 20 20
      23D2  20 46 20
      23D5  20 4F 46
      23D8  46 53 20
      23DB  20 50 43
      23DE  00
1122                  ************************************************
1123                  *
1124  23DF  CE 23 EA  EXEC        LDX   =EXEC.TITLE
1125  23E2  DF 0B               STX   MAIN.ID
1126  23E4  CE 23 F3            LDX   =EXEC.MENU
1127  23E7  7E C3 91            JMP   COMMANDS
1128
1129  23EA  45 58 45  EXEC.TITLE BYTE  'EXECUTER',0
      23ED  43 55 54
      23F0  45 52 00
1130
1131  23F3  0C        EXEC.MENU BYTE  0C
1132  23F4  52 20 52            BYTE  'R RESET1',0
      23F7  45 53 45
      23FA  54 31 00
1133  23FD  22 C6               WORD  RESET1
1134  23FF  4F 20 42            BYTE  'O BOOT + GO',0
      2402  4F 4F 54
      2405  20 2B 20
      2408  47 4F 00
1135  240B  E2 C5               WORD  BOOT
1136  240D  50 20 53            BYTE  'P SET ',LOWER.U,'PC',0
      2410  45 54 20
      2413  75 50 43
      2416  00
1137  2417  E2 D5               WORD  SETUPC
1138  2419  55 20 44            BYTE  'U DISPLAY ',LOWER.U,'ADR+',LOWER.U',INSTR',0
```

```
      241C  49 53 50
      241F  4C 41 59
      2422  20 75 41
      2425  44 52 2B
      2428  75 49 4E
      242B  53 54 52
      242E  00
1139  242F  22 A6              WORD  DMICROINS
1140  2431  4E 20 4E           BYTE  'N NEXT ',LOWER.U,'INSTR',0
      2434  45 58 54
      2437  20 75 49
      243A  4E 53 54
      243D  52 00
1141  243F  22 87              WORD  NEXTI
1142  2441  54 20 54           BYTE  'T TRACE REG',0
      2444  52 41 43
      2447  45 20 52
      244A  45 47 00
1143  244D  23 59              WORD  TRACE.REG
1144  244F  58 20 53           BYTE  'X SINGLE STEP',0
      2452  49 4E 47
      2455  4C 45 20
      2458  53 54 45
      245B  50 00
1145  245D  22 FB              WORD  MSS
1146  245F  53 20 45           BYTE  'S EXEC UNTIL STOP ADR',0
      2462  58 45 43
      2465  20 55 4E
      2468  54 49 4C
      246B  20 53 54
      246E  4F 50 20
      2471  41 44 52
      2474  00
1147  2475  23 40              WORD  BRKPOINT
1148  2477  47 20 47           BYTE  'G GO',0
      247A  4F 00
1149  247C  C1 66              WORD  START
1150  247E  48 20 48           BYTE  'H HALT',0
      2481  41 4C 54
      2484  00
1151  2485  C1 6D              WORD  STOP
1152  2487  49 20 49           BYTE  'I INSPECT',0
      248A  4E 53 50
      248D  45 43 54
      2490  00
1153  2491  20 12              WORD  XINSPECT
1154  2493  59 20 42           BYTE  'Y BYTE INSPECT',0
      2496  59 54 45
      2499  20 49 4E
      249C  53 50 45
      249F  43 54 00
1155  24A2  20 66              WORD  BYTE.INSP
1156  24A4                     END   EXECUTER
```

```
1159  24A4                        BEGIN DEVICES
1160                              DEF   DEVICES
1161                              USE   XINSPECT,DISK.DRIVE
1162                    *
1163                    *         ROUTINES FOR DISPLAY PROCESSOR TESTING
1164                    *
1165                    *         FOLLWING ROUTINES ARE SUPPLIED
1166                    *         R - READ DEVICE
1167                    *         W - WRITE DEVICE
1168                    *         P - SET PATTERN IN MEMORY
1169                    *         Q - SET QUADRUPLE PATTERN
1170                    *         I - INSPECT (SEE MEMORY SUBMONITOR)
1171                    *         M - MAKE A BITMAP DESCRIPTOR
1172                    *
1173                    *
1174                    ****************************************************
1175                    *
1176                    *         READ DEVICE
1177                    *         INPUT   A - DEVICE ADR
1178                    *         OUTPUT  X - VALUE
1179                    *
1180  24A4  B7 3C 0F    GET       STAA  CPU0        SET DEVICE ADR   BYTE ONLY
1181  24A7  CE 25 16              LDX   =SET.IO.ADR
1182  24AA  BD C1 17              JSR   EXX
1183                    *
1184  24AD  36                    PSH   A
1185  24AE  86 07                 LDAA  =07         BUS SOURCE = IODATA
1186  24B0  B7 3C 0C              STAA  MIR0        THIS STARTS THE READ OPERATION
1187  24B3  B7 3C 15            STAA  SSP       SINGLE STEP
1188  24B6  B7 3C 15              STAA  SSP
1189  24B9  FE 3C 0E              LDX   CPU1
1190  24BC  B7 3C 15              STAA  SSP
1191  24BF  B7 3C 15              STAA  SSP
1192  24C2  86 FF                 LDAA  =0FF
1193  24C4  B7 3C 0C              STAA  MIR0        DISABLE ANY BUS SOURCING IN ANY NEXT PROCEDURE
1194  24C7  32                    PUL   A
1195  24C8  39                    RTS
1196                    ****************************************************
1197                    *
1198                    *         WRITE DEVICE
1199                    *         INPUT   A - DEVICE ADR
1200                    *                 X - VALUE
1201                    *
1202  24C9  B7 3C 0F    PUT       STAA  CPU0
1203  24CC  BD C2 59              JSR   PUSHX
1204  24CF  CE 25 16              LDX   =SET.IO.ADR
1205  24D2  BD C1 17              JSR   EXX         SET DEVICE ADR
1206                    *
1207  24D5  BD C2 70              JSR   PULLX
1208  24D8  FF 3C 0E              STX   CPU1
1209  24DB  36                    PSH   A
1210  24DC  86 7D                 LDAA  =7D         SOURCE=PANEL; DEST=IOD
1211  24DE  B7 3C 0C              STAA  MIR0
1212  24E1  B7 3C 15              STAA  SSP         SINGLE STEP
1213  24E4  B7 3C 15              STAA  SSP         SINGLE STEP
1214  24E7  B7 3C 15              STAA  SSP         SINGLE STEP
```

```
1215  24EA  B7 3C 15            STAA  SSP         SINGLE STEP
1216  24ED  86 FF               LDAA  =0FF        DISABLE BUS
1217  24EF  B7 3C 0C            STAA  MIR0
1218  24F2  32                  PUL   A
1219  24F3  39                  RTS
1220                  ************************************************
1221                  *
1222                  *          CALL READ DEVICE
1223  24F4  96 10     RD.DEV     LDAA  X.WORD+1
1224  24F6  BD C1 58             JSR   EDMIR
1225  24F9  BD 24 A4             JSR   GET
1226  24FC  BD C1 5F             JSR   EPMIR
1227  24FF  BD C2 87             JSR   PRINTX
1228  2502  39                  RTS
1229                  ************************************************
1230                  *          CALL WRITE DEVICE
1231  2503  96 10     WR.DEV     LDAA  X.WORD+1    ADR
1232  2505  36                  PSH   A
1233  2506  BD C2 FE             JSR   GETWORD     GET X VALUE
1234  2509  DE 0F                LDX   X.WORD
1235  250B  32                  PUL   A
1236  250C  BD C1 58             JSR   EDMIR
1237  250F  BD 24 C9             JSR   PUT
1238  2512  BD C1 5F             JSR   EPMIR
1239  2515  39                  RTS
1240                  ************************************************
1241                  *
1242  2516  2E 00 07  SET.IO.ADR  BYTE  2E,0,7,60,6D
      2519  60 6D
1243                  ************************************************************
1244                  *
1245                  *          WRITE PATTERN INTO MEMORY
1246                  *
1247                  *          BEGIN (A,B) := X.WORD;
1248                  *            FOR X:=LOWDISP TO LOWDISP+DISPSIZE DO WMM(X,A,B)
1249                  *          END
1250                  *
1251                  LOW.DISP    EQU   0000              STARTING ADR OF DISPLAY MAP
1252                  DISP.SIZE   EQU   0FFF0             SIZE OF DISPLAY MEMORY
1253                  *
1254  251B  96 10     D.PATTERN  LDAA  X.WORD+1
1255  251D  D6 0F                LDAB  X.WORD
1256  251F  CE 00 00             LDX   =LOW.DISP
1257  2522  BD C1 58             JSR   EDMIR
1258  2525  BD C2 28  FOR        JSR   WMM
1259  2528  08                  INX
1260  2529  8C FF F0             CPX   =LOW.DISP+DISP.SIZE
1261  252C  26 F7                BNE   FOR
1262  252E  BD C1 5F             JSR   EPMIR
1263  2531  39                  RTS
1264
1265
1266  2532  CE 00 00  MAKE.BMD    LDX   =0
1267  2535  09                  DEX
1268  2536  4F                  CLR   A
1269  2537  5F                  CLR   B
1270  2538  BD C1 58             JSR   EDMIR
```

```
1271  253B  BD C2 28            JSR   WMM
1272
1273  253E  09                 DEX
1274  253F  86 4F              LDAA  =4F
1275  2541  C6 02              LDAB  =2
1276  2543  BD C2 28            JSR   WMM
1277
1278  2546  09                 DEX
1279  2547  86 30              LDAA  =30
1280  2549  5F                 CLR   B
1281  254A  BD C2 28            JSR   WMM
1282
1283  254D  09                 DEX
1284  254E  86 00              LDAA  =LO(LOW.DISP)
1285  2550  C6 00              LDAB  =HI(LOW.DISP)
1286  2552  BD C2 28            JSR   WMM
1287
1288  2555  4F                 CLR   A
1289  2556  BD 24 C9            JSR   PUT
1290  2559  BD C1 5F            JSR   EPMIR
1291  255C  39                 RTS
1292
1293                  ************************************************
1294
1295  255D  CE 25 68  DEVICES    LDX   =DEV.TITLE
1296  2560  DF 0B               STX   MAIN.ID
1297  2562  CE 25 70            LDX   =DEV.MENU
1298  2565  7E C3 91            JMP   COMMANDS
1299  2568  44 45 56  DEV.TITLE  BYTE  'DEVICES',0
      256B  49 43 45
      256E  53 00
1300                  *
1301  2570  05        DEV.MENU   BYTE  5
1302  2571  47 20 47            BYTE  'G GET',0
      2574  45 54 00
1303  2577  24 F4               WORD  RD.DEV
1304  2579  50 20 50            BYTE  'P PUT',0
      257C  55 54 00
1305  257F  25 03               WORD  WR.DEV
1306  2581  53 20 53            BYTE  'S SET PATTERN',0
      2584  45 54 20
      2587  50 41 54
      258A  54 45 52
      258D  4E 00
1307  258F  25 1B               WORD  D.PATTERN
1308  2591  49 20 49            BYTE  'I INSPECT',0
      2594  4E 53 50
      2597  45 43 54
      259A  00
1309  259B  20 12               WORD  XINSPECT
1310  259D  4D 20 4D            BYTE  'M MAKE BMD',0
      25A0  41 4B 45
      25A3  20 42 4D
      25A6  44 00
1311  25A8  25 32               WORD  MAKE.BMD
1312
1313  25AA                     END   DEVICES
```

```
1315  25AA                        BEGIN MICRO.MEM
1316                              DEF   MICRO.MEM
1317                              USE   U.LOAD,REREAD,U.INSPECT
1318
1319                    ****************************************
1320                    *
1321                    *  PROCEDURE CHECK.LOAD;
1322                    *    (* CHECK IF THE RAM IS PROPERLY LOADED *)
1323                    *    BEGIN
1324                    *      HELP1 := 0; HELP2 := 800H;
1325                    *      REPEAT
1326                    *        FOR B := 4 TO 0 STEP -1 DO
1327                    *          HELP3 := RUM(HELP2,B); A := RUM(HELP1,B)
1328                    *          IF A <> HELP3 THEN
1329                    *            PRINT(HELP1); PRINT(B); PRINT(A); PRINT(HELP3);
1330                    *          END
1331                    *        END
1332                    *        INC(HELP1); INC(HELP2);
1333                    *      UNTIL HELP2 = 1000H;
1334                    *    END CHECK.LOAD;
1335                    *
1336  25AA  CE 00 00    CHECK.LOAD LDX      =0
1337  25AD  DF 30                  STX      HELP1     RAM POIINTER
1338  25AF  CE 08 00               LDX      =800
1339  25B2  DF 32                  STX      HELP2
1340  25B4  C6 04       CH.L.LP1   LDAB     =4
1341  25B6  DE 32       CH.L.LP2   LDX      HELP2
1342  25B8  BD C1 90               JSR      RUM
1343  25BB  97 34                  STAA     HELP3
1344  25BD  DE 30                  LDX      HELP1
1345  25BF  BD C1 90               JSR      RUM
1346  25C2  91 34                  CMPA     HELP3
1347  25C4  27 1D                  BEQ      CH.L.OK
1348  25C6  BD C2 87               JSR      PRINTX    PRINT ADR
1349  25C9  36                     PSH      A
1350  25CA  BD C0 62               JSR      W.1.BLANK
1351  25CD  17                     TBA
1352  25CE  BD C0 C9               JSR      OUT.BYTE  PRINT BYTE ADR
1353  25D1  BD C0 62               JSR      W.1.BLANK
1354  25D4  32                     PUL      A
1355  25D5  BD C0 C9               JSR      OUT.BYTE  PRINT RAM BYTE
1356  25D8  BD C0 62               JSR      W.1.BLANK
1357  25DB  96 34                  LDAA     HELP3
1358  25DD  BD C0 C9               JSR      OUT.BYTE  PRINT ROM BYTE
1359  25E0  BD C0 78               JSR      W.CRLF
1360  25E3  5A          CH.L.OK    DEC      B
1361  25E4  2A D0                  BPL      CH.L.LP2
1362  25E6  08                     INX                INC(HELP1)
1363  25E7  DF 30                  STX      HELP1
1364  25E9  DE 32                  LDX      HELP2
1365  25EB  08                     INX
1366  25EC  DF 32                  STX      HELP2
1367  25EE  8C 10 00               CPX      =1000
1368  25F1  26 C1                  BNE      CH.L.LP1
1369  25F3  39                     RTS
1370                    *
```

```
1371                    ************************************************************
1372                    *
1373                    *           INSPECT AND CHANGE MICRO MEMORY
1374  25F4              *           BEGIN INSP
1375                    *           DEF   U.INSPECT
1376                    *
1377                    *           FORMAT:
1378                    *           ADDRESS 'I' SUBADR  VALUE  [ ':' NEWVALUE] [ ',' ]
1379                    *           ADDRESS  =  4 DIGIT HEX
1380                    *           SUBADR   =  0..4
1381                    *
1382                    *       PROCEDURE INSPECT
1383                    *          BEGIN
1384                    *             CRNT.ADR := X.WORD;
1385                    *             REPEAT R.CH(A) UNTIL (A>='0') AND (A<='4'); CRNT.SUBADR := A-'0';
1386                    *           LOOP
1387                    *             A := RUM(CRNT.ADR, CRNT.SUBADR);
1388                    *             W.CH(' '); OUT.BYTE(A);
1389                    *             R.CH(A);
1390                    *             IF A=':' THEN
1391                    *               IN.BYTE(A); WUM(CRNT.ADR, CRNT.SUBADR, A); R.CH(A)
1392                    *             END;
1393                    *           IF A#',' THEN EXIT  END;
1394                    *             IF CRNT.SUBADR = 4 THEN INC(CRNT.ADR); CRNT.SUBADR := -1; END;
1395                    *             INC(CRNT.SUBADR);
1396                    *             W.CRLF; OUT.HEX(CRNT.ADR); W.CH('-');
1397                    *             W.CH(CRNT.SUBADR + '0'); W.CH(' ')
1398                    *             END
1399                    *           END
1400                    *         END INSPECT;
1401                    *
1402                    *
1403                    CRNT.ADR    EQU   HELP1
1404                    CRNT.SBADR  EQU   HELP2
1405                    *
1406  25F4  DE 0F       U.INSPECT   LDX   X.WORD
1407  25F6  DF 30                   STX   CRNT.ADR
1408  25F8  BD C0 85    REP2        JSR   IN.HEX      GET SUBADR
1409  25FB  81 04                   CMPA  =4
1410  25FD  2E F9                   BGT   REP2
1411  25FF  97 32                   STAA  CRNT.SBADR
1412  2601  BD C0 62    LOOP        JSR   W.1.BLANK
1413  2604  D6 32                   LDAB  CRNT.SBADR
1414  2606  BD C1 90                JSR   RUM         READ IT
1415  2609  BD C0 C9                JSR   OUT.BYTE    A CONTAINS ALREADY THE VALUE
1416  260C  BD C0 24                JSR   R.CH        GET DELIMITER
1417  260F  81 3A                   CMPA  =':'
1418  2611  26 09                   BNE   ENDIF1
1419  2613  BD C0 BC                JSR   IN.BYTE     GET NEW VALUE
1420  2616  BD C1 7B                JSR   WUM         WRITE MICRO MEMORY; B, X ARE ALREADY SET
1421  2619  BD C0 24                JSR   R.CH        GET DELIMITER
1422  261C  81 2C       ENDIF1      CMPA  =','
1423  261E  27 01                   BEQ   ENDIF2
1424  2620  39                      RTS               ANY OTHER DELIMITER THEN ',' => EXIT
1425  2621  C1 04       ENDIF2      CMPB  =4           COMP CRNT.SUBADR,#4
1426  2623  26 07                   BNE   ENDIF3
1427  2625  08                      INX               INC(CRNT.ADR)
```

```
1428  2626  DF 30               STX   CRNT.ADR
1429  2628  86 FF               LDAA  =0FF
1430  262A  97 32               STAA  CRNT.SBADR CRNT.SUBADR := -1
1431  262C  7C 00 32  ENDIF3    INC   CRNT.SBADR
1432  262F  BD C0 78             JSR   W.CRLF
1433  2632  BD C2 87             JSR   PRINT.X     WRITE(ADR)
1434  2635  86 2D               LDAA  ='-'
1435  2637  BD C0 51             JSR   W.CH        WRITE('-')
1436  263A  96 32               LDAA  CRNT.SBADR
1437  263C  BD C0 90             JSR   OUT.HEX     WRITE(SUBADR)
1438  263F  20 C0               BRA   LOOP
1439  2641                      END   INSP
1440                  *
1441                  ************************************************************
1442                  *
1443                  * LOAD MICRO RAM FROM HP-CASSETTE (STANDARD HEXADECIMAL CODED 6800-FORMAT)
1444                  * *************************
1445                  *
1446                  *           PROCEDURE LOAD
1447                  *           VAR EOT: BOOLEAN; (* A-REG, 0=TRUE *)
1448                  *
1449                  *             PROCEDURE SKIPLINE;
1450                  *               BEGIN REPEAT READ(CH) UNTIL CH=CR END;
1451                  *
1452                  *             PROCEDURE READBLOCK;
1453                  *             VAR CH : CHAR; (* A-REG *)
1454                  *                 CHECKSUM,  (* HELP1 *)
1455                  *                 BYTECOUNT : BYTE; (* HELP2 *)
1456                  *                 ADDR : ADDRESS;   (* X-REG *)
1457                  *             BEGIN   1: LINEREAD;
1458                  *               REPEAT
1459                  *                 REACH(CH);
1460                  *               UNTIL CH = 'S';
1461                  *               READ(CH);
1462                  *               EOT := CH = '9';
1463                  *               IF NOT EOT
1464                  *               THEN BEGIN
1465                  *                 INBYTE(BYTECOUNT);
1466                  *                 CHECKSUM := BYTECOUNT;
1467                  *                 INWORD(ADDR);
1468                  *                 CHECKSUM := CHECKSUM + ADDR MOD 40 + ADDR DIV 40;
1469                  *                 REPEAT
1470                  *                   INBYTE(A);
1471                  *                   WUM(X, B);
1472                  *                   CHECKSUM := CHECKSUM + M[ADDR];
1473                  *                   ADDR := ADDR + 1;
1474                  *                   BYTECOUNT := BYTECOUNT - 1;
1475                  *                 UNTIL BYTECOUNT = 3;
1476                  *                 INBYTE(CH);  (* CHECKSUM *)
1477                  *                 IF CHECKSUM <> COMPL(CH)
1478                  *                 THEN BEGIN
1479                  *                   WSTRING('CHECKSUM ERROR');
1480                  *                 END;
1481                  *               END;
1482                  *             END READBLOCK;
1483                  *
1484                  *           BEGIN (* LOAD *)
```

```
1485                    *             FOR B := 0 TO 4 DO  (* READ 4 FILES *)
1486                    *               (*SKIP HEADER AND INTERFILE GAP*)
1487                    *               LINEREAD;
1488                    *               REPEAT    READ(CH);
1489                    *                 IF CH=CR THEN LINEREAD END;
1490                    *               UNTIL CH = 'S';
1491                    *               REPEAT READ(CH) UNTIL CH = CR;
1492                    *               REPEAT
1493                    *                 READBLOCK;
1494                    *               UNTIL EOT;
1495                    *             END
1496                    *           END LOAD;
1497
1498  2641                          BEGIN TAPE.LOAD
1499                                DEF   U.LOAD
1500
1501                  CRNT.SBADR    EQU   HELP3
1502
1503  2641  BD E0 03  SKIP.LINE    JSR   READ.CH
1504  2644  81 0D                  CMPA  =CR
1505  2646  26 F9                  BNE   SKIP.LINE
1506  2648  39                    RTS
1507                  *
1508                  READ.BLOCK    EQU   *
1509  2649                          BEGIN READBLOCK
1510  2649  BD C0 FB  REPEAT0      JSR   LINE.READ
1511  264C  BD E0 03  REPEAT1      JSR   READ.CH     SKIP UNTIL 'S'
1512  264F  81 53                  CMPA  ='S'
1513  2651  26 F9     UNTIL1       BNE   REPEAT1
1514  2653  BD E0 03               JSR   READ.CH
1515
1516  2656  80 39                  SUBA  ='9'
1517  2658  36                    PSH   A            SAVE THE EOF INFORMATION
1518  2659  27 37     IF1          BEQ   ENDIF
1519  265B  BD E0 1A  THEN1        JSR   READ.BYTE   BYTECOUNT
1520  265E  36                    PSH   A
1521  265F  80 03                  SUBA  =3          SUBSTRACT BYTECNT, 2 BYTES ADDRESS
1522  2661  97 32                  STAA  HELP2
1523  2663  32                    PUL   A
1524  2664  CE 00 01               LDX   =X.SAVE
1525  2667  BD E0 27               JSR   READ.WORD   ADDRESS
1526  266A  AB 00                  ADDA  0,X
1527  266C  AB 01                  ADDA  1,X
1528  266E  97 30                  STAA  HELP1       XSUM:= BYTECOUNT + ADDR(HI) + ADDR(LO)
1529  2670  DE 01                  LDX   X.SAVE
1530  2672  BD E0 1A  REPEAT2      JSR   READ.BYTE   DATA
1531  2675  D6 34                  LDAB  CRNT.SBADR
1532  2677  BD C1 7B               JSR   WUM         X IS ALREADY SET
1533
1534  267A  9B 30                  ADDA  HELP1
1535  267C  97 30                  STAA  HELP1       XSUM := XSUM + A
1536  267E  08                    INX
1537  267F  7A 00 32               DEC   HELP2
1538  2682  26 EE     UNTIL2       BNE   REPEAT2
1539  2684  BD E0 1A               JSR   READ.BYTE   CHECKSUM
1540  2687  43                    COM   A
1541  2688  91 30                  CMPA  HELP1
```

```
1542  268A  27 06     IF2         BEQ   ENDIF
1543  268C  CE 26 C2  THEN2       LDX   =MESS1
1544  268F  BD C0 6A              JSR   W.STRING
1545                  ENDIF       EQU   *
1546  2692  32                    PUL   A            RESAVE THE EOF INFORMATION
1547  2693  39                    RTS
1548  2694                        END   READBLOCK
1549
1550
1551  2694  7F 00 34  U.LOAD      CLR   CRNT.SBADR  WE START READING THE FIRST FILE
1552  2697  BD C0 FB  SKIP.HDR0   JSR   LINEREAD    SKIP THE HEADER LINE
1553  269A  BD E0 03  SKIP.HDR1   JSR   READ.CH
1554  269D  81 0D                 CMPA  =CR
1555  269F  26 03                 BNE   SKIP.HDR2
1556  26A1  BD C0 FB              JSR   LINEREAD
1557  26A4  81 53     SKIP.HDR2   CMPA  ='S'
1558  26A6  26 F2                 BNE   SKIP.HDR1
1559
1560  26A8  BD E0 03  SKIP.HDR3   JSR   READ.CH     SKIP THE FIRST LINE
1561  26AB  81 0D                 CMPA  =CR
1562  26AD  26 F9                 BNE   SKIP.HDR3
1563
1564  26AF  BD 26 49  REPEAT      JSR   READ.BLOCK
1565  26B2  4D                    TST   A            END OF TAPE
1566  26B3  26 FA     UNTIL       BNE   REPEAT
1567
1568  26B5  BD 26 41              JSR   SKIP.LINE
1569
1570  26B8  7C 00 34              INC   CRNT.SBADR  INCREMENT THE FILE NUMBER
1571  26BB  D6 34                 LDAB  CRNT.SB.ADR
1572  26BD  C1 04                 CMPB  =4
1573  26BF  2F D6                 BLE   SKIP.HDR0
1574  26C1  39                    RTS
1575                  *
1576  26C2  58 53 55  MESS1       BYTE  'XSUM ERR',0
      26C5  4D 20 45
      26C8  52 52 00
1577  26CB                        END   TAPE.LOAD
1578                  *
1579                  ************************************************************
1580                  *
1581                  * COMPARE MICRO CODE WITH HP-CASSETTE (STANDARD HEXADECIMAL CODED 6800-FORMAT)
1582                  * *************************
1583                  *
1584                  *           PROCEDURE REREAD
1585                  *           VAR EOT: BOOLEAN; (* A-REG, 0=TRUE *)
1586                  *
1587                  *             PROCEDURE SKIPLINE;
1588                  *               BEGIN REPEAT READ(CH) UNTIL CH=CR END;
1589                  *
1590                  *             PROCEDURE READBLOCK;
1591                  *             VAR CH : CHAR; (* A-REG *)
1592                  *                 CHECKSUM,  (* HELP1 *)
1593                  *                 BYTECOUNT : BYTE; (* HELP2 *)
1594                  *                 ADDR : ADDRESS;   (* X-REG *)
1595                  *             BEGIN   1: LINEREAD;
1596                  *               REPEAT
```

```
1597                    *                 REACH(CH);
1598                    *               UNTIL CH = 'S';
1599                    *               READ(CH);
1600                    *               EOT := CH = '9';
1601                    *               IF NOT EOT
1602                    *               THEN BEGIN
1603                    *                 INBYTE(BYTECOUNT);
1604                    *                 CHECKSUM := BYTECOUNT;
1605                    *                 INWORD(ADDR);
1606                    *                 CHECKSUM := CHECKSUM + ADDR MOD 40 + ADDR DIV 40;
1607                    *                 REPEAT
1608                    *                   INBYTE(A);
1609                    *                   IF A <> RUM(X,B) THEN
1610                    *                      WRITELN(ROMVALUE); SWI(B, TAPEVALUE,X);
1611                    *                      SKIP THE REST OF THE LINE;  GOTO 1;
1612                    *                   CHECKSUM := CHECKSUM + M[ADDR];
1613                    *                   ADDR := ADDR + 1;
1614                    *                   BYTECOUNT := BYTECOUNT - 1;
1615                    *                 UNTIL BYTECOUNT = 3;
1616                    *                 INBYTE(CH);  (* CHECKSUM *)
1617                    *                 IF CHECKSUM <> COMPL(CH)
1618                    *                 THEN BEGIN
1619                    *                   WSTRING('CHECKSUM ERROR');
1620                    *                 END;
1621                    *               END;
1622                    *             END READBLOCK;
1623                    *
1624                    *           BEGIN (* REREAD *)
1625                    *             FOR B := 0 TO 4 DO  (* READ 4 FILES *)
1626                    *               (*SKIP HEADER AND INTERFILE GAP*)
1627                    *               LINEREAD;
1628                    *               REPEAT    READ(CH);
1629                    *                 IF CH=CR THEN LINEREAD END;
1630                    *               UNTIL CH = 'S';
1631                    *               REPEAT READ(CH) UNTIL CH = CR;
1632                    *               REPEAT
1633                    *                 READBLOCK;
1634                    *               UNTIL EOT;
1635                    *             END
1636                    *           END REREAD;
1637
1638  26CB                          BEGIN TAPE.CMP
1639                                DEF   REREAD
1640
1641  26CB  BD E0 03  SKIP.LINE     JSR   READ.CH
1642  26CE  81 0D                   CMPA  =CR
1643  26D0  26 F9                   BNE   SKIP.LINE
1644  26D2  39                      RTS
1645                  *
1646                  READ.BLOCK    EQU   *
1647  26D3                          BEGIN READBLOCK
1648  26D3  BD C0 FB  REPEAT0       JSR   LINE.READ
1649  26D6  BD E0 03  REPEAT1       JSR   READ.CH     SKIP UNTIL 'S'
1650  26D9  81 53                   CMPA  ='S'
1651  26DB  26 F9     UNTIL1        BNE   REPEAT1
1652  26DD  BD E0 03                JSR   READ.CH
1653
```

```
1654  26E0  80 39              SUBA  ='9'
1655  26E2  36                 PSH   A            SAVE THE EOF INFORMATION
1656  26E3  27 4C     IF1      BEQ   ENDIF
1657  26E5  BD E0 1A  THEN1    JSR   READ.BYTE    BYTECOUNT
1658  26E8  36                 PSH   A
1659  26E9  80 03              SUBA  =3           SUBSTRACT BYTECNT, 2 BYTES ADDRESS
1660  26EB  97 32              STAA  HELP2
1661  26ED  32                 PUL   A
1662  26EE  CE 00 01           LDX   =X.SAVE
1663  26F1  BD E0 27           JSR   READ.WORD    ADDRESS
1664  26F4  AB 00              ADDA  0,X
1665  26F6  AB 01              ADDA  1,X
1666  26F8  97 30              STAA  HELP1        XSUM:= BYTECOUNT + ADDR(HI) + ADDR(LO)
1667  26FA  DE 01              LDX   X.SAVE
1668  26FC  BD E0 1A  REPEAT2  JSR   READ.BYTE    DATA
1669  26FF  97 34              STAA  HELP3        HELP3 := TAPE DATA
1670  2701  BD C1 90           JSR   RUM          READ MICRO MEM; X,B ARE ALREADY SET
1671  2704  91 34              CMPA  HELP3
1672  2706  27 0F              BEQ   U.ROM.OK
1673  2708  36                 PSH   A            SKIP THE REST OF THE LINE
1674  2709  8D C0              BSR   SKIP.LINE
1675  270B  32                 PUL   A
1676  270C  BD C0 C9           JSR   OUT.BYTE     ROM VALUE ********************
1677  270F  BD C0 78           JSR   W.CRLF
1678  2712  96 34              LDAA  HELP3
1679  2714  3F                 SWI                B=BYTE ADR; A=TAPE VALUE; X=ADDRESS  ****
1680  2715  20 BC              BRA   REPEAT0      TRY NEW LINE
1681
1682  2717  96 34     U.ROM.OK LDAA  HELP3
1683  2719  9B 30              ADDA  HELP1
1684  271B  97 30              STAA  HELP1        XSUM := XSUM + A
1685  271D  08                 INX
1686  271E  7A 00 32           DEC   HELP2
1687  2721  26 D9     UNTIL2   BNE   REPEAT2
1688  2723  BD E0 1A           JSR   READ.BYTE    CHECKSUM
1689  2726  43                 COM   A
1690  2727  91 30              CMPA  HELP1
1691  2729  27 06     IF2      BEQ   ENDIF
1692  272B  CE 27 61 THEN2     LDX   =MESS1
1693  272E  BD C0 6A           JSR   W.STRING
1694                  ENDIF    EQU   *
1695  2731  32                 PUL   A            RESAVE THE EOF INFORMATION
1696  2732  39                 RTS
1697  2733                     END   READBLOCK
1698
1699
1700  2733  CE 27 6A  REREAD   LDX   =CMP.TXT
1701  2736  DF 0D              STX   SUB.ID
1702  2738  C6 00              LDAB  =0           WE START READING THE FIRST FILE
1703  273A  BD C0 FB SKIP.HDR0 JSR   LINEREAD    SKIP THE HEADER LINE
1704  273D  BD E0 03 SKIP.HDR1 JSR   READ.CH
1705  2740  81 0D              CMPA  =CR
1706  2742  26 03              BNE   SKIP.HDR2
1707  2744  BD C0 FB           JSR   LINEREAD
1708  2747  81 53     SKIP.HDR2 CMPA ='S'
1709  2749  26 F2              BNE   SKIP.HDR1
1710
```

```
1711  274B  BD E0 03  SKIP.HDR3   JSR   READ.CH     SKIP THE FIRST LINE
1712  274E  81 0D                 CMPA  =CR
1713  2750  26 F9                 BNE   SKIP.HDR3
1714
1715  2752  BD 26 D3  REPEAT      JSR   READ.BLOCK
1716  2755  4D                   TST   A           END OF TAPE
1717  2756  26 FA     UNTIL       BNE   REPEAT
1718
1719  2758  BD 26 CB              JSR   SKIP.LINE
1720
1721  275B  5C                   INC   B           INCREMENT THE FILE NUMBER
1722  275C  C1 04                CMPB  =4
1723  275E  2F DA                BLE   SKIP.HDR0
1724  2760  39                   RTS
1725                      *
1726  2761  58 53 55  MESS1       BYTE  'XSUM ERR',0
      2764  4D 20 45
      2767  52 52 00
1727  276A  55 52 4F  CMPTXT      BYTE  'UROM CMP',0
      276D  4D 20 43
      2770  4D 50 00
1728  2773                       END   TAPE.CMP
1729
1730
1731                      ************************************************************
1732                      *
1733  2773  CE 27 7E  MICRO.MEM   LDX   =MIC.TITLE
1734  2776  DF 0B                STX   MAIN.ID
1735  2778  CE 27 8B              LDX   =MIC.MENU
1736  277B  7E C3 91             JMP   COMMANDS
1737
1738  277E  4D 49 43  MIC.TITLE   BYTE  'MICRO-MEMORY',0
      2781  52 4F 2D
      2784  4D 45 4D
      2787  4F 52 59
      278A  00
1739
1740  278B  05        MIC.MENU    BYTE  5
1741  278C  49 20 49              BYTE  'I INSPECT',0
      278F  4E 53 50
      2792  45 43 54
      2795  00
1742  2796  25 F4                 WORD  U.INSPECT
1743  2798  5A 20 43              BYTE  'Z CHECK RAM',0
      279B  48 45 43
      279E  4B 20 52
      27A1  41 4D 00
1744  27A4  25 AA                 WORD  CHECK.LOAD
1745  27A6  4C 20 4C              BYTE  'L LOAD RAM FROM ROM',0
      27A9  4F 41 44
      27AC  20 52 41
      27AF  4D 20 46
      27B2  52 4F 4D
      27B5  20 52 4F
      27B8  4D 00
1746  27BA  E3 B3                 WORD  LOAD.U.RAM
1747  27BC  54 20 4C              BYTE  'T LOAD RAM FROM TAPE',0
```

```
     SUBMONIT  MICROMEM

      27BF  4F 41 44
      27C2  20 52 41
      27C5  4D 20 46
      27C8  52 4F 4D
      27CB  20 54 41
      27CE  50 45 00
1748  27D1  26 94              WORD  U.LOAD
1749  27D3  52 20 52           BYTE  'R REREAD TAPE',0
      27D6  45 52 45
      27D9  41 44 20
      27DC  54 41 50
      27DF  45 00
1750  27E1  27 33              WORD  REREAD
1751
1752  27E3                     END   MICRO.MEM
1753
1754  27E3                     END   SUBMONIT
```