HWTEST

```
 2  0000             BEGIN HWTEST
 3
 4                  ************************************************************
 5                  *
 6                  *    H A R D W A R E T E S T 2            5-DEC-79
 7                  *
 8                  *    INCLUDES IFU, DEVICES,MICROMEMORY
 9                  *
10                  *    AUTHORS:
11                  *       JIRKA HOPPE
12                  *       FARRELL OSTLER
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)
```

```
264                 RMM         EQU   0C1FE
265                 ***         READ MAIN MEMORY
266                 *           B:= MAIN(X)DIV 256
267                 *           A:= MAIN(X)MOD 256
268
269                 RMMS        EQU   0C211
270                 ****        SHORT (QUICK) VERSION OF RMM
271                 *           MAY ONLY BE USED AFTER A LONG VERSION CALL HAS BEEN MADE AND
272                 *           NO OTHER DMIR OPERATIONS HAVE OCCURED
273
274                 WMM         EQU   0C228
275                 ***         WRITE MAIN MEMORY
276                 *           MAIN(X):=256*B + A
277
278                 WMMS        EQU   0C23D
279                 ****        SHORT VERSION OF WMM. SEE RMMS COMMENTS
280
281                 LATCH       EQU   0C151
282                 *****       MICRO MEMORY ADDRESS REGISTER IS LATCHED
283
284                 UNLCH       EQU   0C14A
285                 *****       MICRO MEMORY ADDRESS REGISTER IS UNLATCHED
286
287                 D2911       EQU   0C13E
288                 *****       MICRO MEMORY ADDRESS OUTPUTS FROM PERSONAL COMPUTER
289                 *           ARE DISABLED AND ADDRESS REGISTER FROM DIAGNOSTIC
290                 *           PANEL IS ENABLED
291
292                 E2911       EQU   0C174
293                 *****       REVERSE OF D2911
294
295                 STOP        EQU   0C16D
296                 ****        DISABLE CPU CLOCK AND ENABLE SINGLE STEP MODE
297
298                 START       EQU   0C166
299                 *****       ENABLE CPU CLOCK
300
301                 RESET       EQU   0C290
302                 *****       RESET COMPUTER; INITIATE STAT0 VECTOR
303
304                 READ.REG    EQU   0C2DD
305                 ********    READ A 2901 REGISTER
306                 *           ARGUMENT:  B = NR OF THE REGISTER  ( 0 .. 0F OR Q.REG)
307                 *           RESULT:    X = THE VALUE
308
309                 READ.R      EQU   0C2A5
310                 ******      MIR USED BY THE READ.REG
311
312                 LOAD.REG    EQU   0C2AA
313                 *******     LOAD A 2901 REGISTER
314                 *           ARGUMENTS: B = NR OF THE REGISTER  ( 0..0F OF Q.REG)
315                 *                      X = VALUE TO BE LOADED
316
317                 WRITE.R     EQU   0C2A0
318                 *******     MIR USED BY LOAD.REG
319
```

```
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                 ************************************************************
```

```
405                     *                                                     *
406                     *                                                     *
407                     *         HARDWARE TEST SUBMONITOR                    *
408                     *                                                     *
409                     *         AUTHOR JIRKA HOPPE                          *
410                     *                                                     *
411                     *         INSTITUT OF INFORMATICS                     *
412                     *         ETH ZURICH                                  *
413                     *                                                     *
414                     *         VERSION   17.6.79                           *
415                     *                                                     *
416                     *                                                     *
417                     *                                                     *
418                     *                                                     *
419                     *******************************************************
420                             USE       HWTEST,DEVICES,MICROMEM
421                     CODE    SET       *
422  0000                       ORG       USER.DATA
423                     *       DEFINITION OF VARIABLES
424  002C  00 00        NEXT.IN.CH WORD     0         POINTS TO THE NEXT ENTRY IN THE CHAIN
425  002E  00 00        HELP1     WORD      0         IS USED LOCALLY BY TEST PROCEDURES
426  0030  00 00        HELP2     WORD      0             ''    NO GLOBAL PROCEDURE MAKES USE OF IT
427  0032  00 00        HELP3     WORD      0             ''
428
429                     USER.DATA SET       *
430
431                     *         ENTRY POINTS USED BY THE MAIN MONITOR
432  0034                       ORG       2003
433  2003  7E E0 00             JMP       MONITOR      RETURN TO MAIN MONITOR
434  2006  7E E0 00             JMP       MONITOR      RETURN TO MAIN MONITOR
435  2009  7E 26 DF             JMP       DEVICES
436  200C  7E 21 AC             JMP       HWTEST
437  200F  7E 29 13             JMP       MICROMEM
438  2012  7E 20 12  JUMP       JMP       JUMP      USED BY CHAIN FOR JUMP TO THE ROUTINE ENTRY
439                     *
440                     *******************************************************
```

```
442  2015                       BEGIN     HARDWARE TEST
443                             DEF       HWTEST,ENDTEST,CHAIN,WRITELN.A,SET.MAIN.ID
444                             DEF       X.L.ROTATE,X.R.ROTATE
445                             DEF       NEXTCHAIN,INSPECT
446                             USE       IFU.TEST,IFU.CHAIN,IFU.CH.LEN
447                             USE       MICROMEM
448                     ************************************************************
449                     *
450                     *         MAIN HARDWARE TEST MONITOR
451                     *
452                     ************************************************************
453                     *
454                     *    THIS PROCEDURE ALLOWS CHAINING OF A NUMBER OF HARDWARE TESTS
455                     *
456                     *    A CHAIN HAS FOLLOWING FORMAT:
457                     *      #ITEMS      ! ENTRY.ADR    X.VALUE "
458                     *
459                     *    #ITEMS    - HOW MANY PAIRS (ENTRYADR,VALUE) ARE CONTAINED IN THE CHAIN
460                     *    ENTRY.ADR - ENTRY POINT OF THE SUBROUTINE
461                     *    X.VALUE   - INITIAL VALUE FOR THE X REGISTER
462                     *
463                     *  PROCEDURE CHAIN(X : ↑CHAINS);
464                     *    VAR COUNT, SAVEX : SAVES;
465                     *    BEGIN  COUNT := X↑; INC(X);
466                     *      REPEAT  JUMP := X↑; INC(X,2); SAVEX := X;
467                     *        X := X↑; CALL(JUMP);
468                     *        X := SAVEX; INX(X,2);
469                     *        DEC(COUNT);
470                     *      UNTIL COUNT = 0
471                     *    END;
472                     *
473  2015  A6 00        CHAIN     LDAA      0,X
474  2017  36                    PSH       A         SAVE COUNT
475  2018  08                    INX
476  2019  A6 00        CHLOOP    LDAA      0,X       GET ENTRY POINT
477  201B  B7 20 13              STAA      JUMP+1
478  201E  A6 01                 LDAA      1,X
479  2020  B7 20 14              STAA      JUMP+2
480  2023  08                    INX
481  2024  08                    INX
482  2025  DF 2C                 STX       NEXT.IN.CH
483  2027  EE 00                 LDX       0,X       GET INITIAL X VALUE
484  2029  BD 20 12              JSR       JUMP      EXECUTE ROUTINE
485  202C  BD C0 46              JSR       BREAK     GET OUT BY 'ESC'
486  202F  DE 2C                 LDX       NEXT.IN.CH
487  2031  08                    INX
488  2032  08                    INX
489  2033  32                    PUL       A         GET COUNTER
490  2034  4A                    DEC       A
491  2035  36                    PSH       A         SAVE IT AGAIN
492  2036  26 E1                BNE       CHLOOP
493  2038  32                    PUL       A         JUST CLEAN UP STACK
494  2039  39                    RTS
495                     *
496                     ************************************************************
497                     *
```

```
498                     *  PROCEDURE NEXTCHAIN( X : ↑CHAIN);
499                     *    (* THIS PROCEDURE SWITCHES TO THE NEXT CHAIN POINTED BY X *)
500                     *
501  203A  DF 2C        NEXTCHAIN STX       NEXT.IN.CH
502  203C  39                     RTS
503                     *
504                     ************************************************************
505                     *
506                     *  PROCEDURE LOOP.CHAIN( X : ↑CHAINS);
507                     *    BEGIN LOOP CHAIN(X.WORD) END END;
508                     *
509  203D  CE 22 12     LOOPCHAIN LDX       =GLOB.CHAIN
510  2040  8D D3                BSR       CHAIN
511  2042  20 F9                BRA       LOOPCHAIN
512                     *
513                     ************************************************************
514                     *
515                     *  PROCEDURE LOOPROUTINE( X );
516                     *    BEGIN LOOP CALL(X↑) END END;
517                     *
518  2044  DE 0F        LOOP.RTNE LDX       X.WORD
519  2046  AD 00                 JSR       0,X
520  2048  20 FA                BRA       LOOP.RTNE
521                     *
522                     ************************************************************
523                     *
524                     *  PROCEDURE SET.MAIN.ID( X : ↑CHAINS);
525                     *    BEGIN MAIN.ID := X END
526  204A  DF 0B        SETMAINID STX       MAIN.ID
527  204C  39                     RTS
528                     *
529                     ************************************************************
530                     *
531                     *  PROCEDURE END.TEST;
532                     *    BEGIN WRITE('END OF ',SUB.ID) END;
533                     *
534  204D  36           END.TEST  PSH       A
535  204E  96 16                LDAA      TYPING    TYPE CTRL
536  2050  27 12                BEQ       END.END   NO TYPING AT ALL
537  2052  81 02                CMPA      =2
538  2054  27 0E                BEQ       END.END   ONLY ERROR MESSAGES
539  2056  CE 20 69             LDX       =END.TEXT
540  2059  BD C0 6A             JSR       W.STRING
541  205C  DE 0D                LDX       SUB.ID
542  205E  BD C0 6A             JSR       W.STRING
543  2061  BD C0 78             JSR       W.CRLF
544  2064  32           END.END   PUL       A
545  2065  BD C2 90             JSR       RESET
546  2068  39                   RTS
547                     *
548  2069  45 4E 44     END.TEXT  BYTE      'END OF ',0
     206C  20 4F 46
     206F  20 00
549                     *
550                     ************************************************************
551                     *
552                     *  PROCEDURE PRINTAB( A, B, X)
```

```
                        *    BEGIN WRITE(X↑, A, B) END;
                        *
2071  36        PRINT.AB  PSH       A
2072  BD C0 6A            JSR       W.STRING
2075  BD C0 62            JSR       W.1.BLANK
2078  BD C0 C9            JSR       OUT.BYTE
207B  BD C0 62            JSR       W.1.BLANK
207E  17                  TBA
207F  BD C0 C9            JSR       OUT.BYTE
2082  BD C0 78            JSR       W.CRLF
2085  32                  PUL       A
2086  39                  RTS
                        *
                        ************************************************************
                        *  PROCEDURE WRITELN(A);
                        *    BEGIN WRITE(A); CRLF END;
                        *
2087  BD C0 C9  WRITELN.A JSR    OUTBYTE
208A  7E C0 78            JMP       W.CRLF    MAKES RTS
                        *
                        ************************************************************
                        *
                        *  ROTATE X REGISTER ONE BIT TO LEFT
                        *
208D  36        X.LROTATE PSH       A
208E  37                  PSH       B
208F  DF 01               STX       X.SAVE
2091  96 01               LDAA      X.SAVE    HIGH BYTE
2093  48                  ASL       A         CARRY <- MSB; LSB <- 0
2094  D6 02               LDAB      XSAVE+1   LOWER BYTE; DOESN'T CHANGE CARRY
2096  2A 01               BPL       X.L.1
2098  4C                  INC       A         SET LSB OF 'A' IF MSB OF 'B' WAS SET
                        *                     CARRY IS NOT CHANGED
2099  59        X.L.1     ROL       B         CARRY FROM 'ASL A'
209A  97 01               STAA      X.SAVE
209C  D7 02               STAB      XSAVE+1
209E  DE 01               LDX       X.SAVE
20A0  33                  PUL       B
20A1  32                  PUL       A
20A2  39                  RTS
                        *
                        ************************************************************
                        *
                        *  ROTATE X REGISTER ONE BIT TO RIGHT
                        *
20A3  36        X.RROTATE PSH       A
20A4  37                  PSH       B
20A5  DF 01               STX       X.SAVE
20A7  96 02               LDAA      X.SAVE+1  LOWER
20A9  D6 01               LDAB      X.SAVE    HIGHER
20AB  44                  LSR       A
20AC  56                  ROR       B
20AD  24 02               BCC       X.R.1
20AF  8A 80               ORAA      =80       SET MSB
20B1  97 02     X.R.1     STAA      X.SAVE+1
20B3  D7 01               STAB      X.SAVE
20B5  DE 01               LDX       X.SAVE
```

```
610  20B7  33              PUL       B
611  20B8  32              PUL       A
612  20B9  39              RTS
613                *
614                ************************************************************
615                *
616                *  PROCEDURE TESTBUS;
617                *    (* TEST FOR SHORT CONNECTION ON THE BUS *)
618                *    BEGIN SUB.ID := 'CPU BUS';
619                *      MIR := (1 - OR ZA 0 0 0 - - - - 0 - PANEL);
620                *      X := 1;
621                *      FOR B := 2 TO 0 BY -1
622                *        FOR A := 16 TO 0 BY -1 DO
623                *          BUS := X;
624                *          IF BUS <> X THEN PRINTBUS(X) END;
625                *          X.L ROTATE;
626                *        END;
627                *        X := 0FFFEH;
628                *      END;
629                *    END;
630                *
631  20BA  CE 20 F2  TESTBUS   LDX       =BUSTEXT
632  20BD  DF 0D             STX       SUB.ID
633  20BF  CE C2 A5          LDX       =READ.R
634  20C2  BD C1 1D          JSR       LDMIR
635  20C5  86 FD             LDAA      =0FD      SOURCE := PANEL
636  20C7  B7 3C 0C          STAA      MIR0
637  20CA  BD C1 58          JSR       EDMIR
638  20CD  CE 00 01          LDX       =1        ROTATING 1
639  20D0  C6 02             LDAB      =2
640  20D2  86 10     FORBUS1 LDAA      =16Z
641  20D4  FF 3C 0E  FORBUS2 STX       CPU1
642  20D7  BC 3C 0E          CPX       CPU1
643  20DA  27 03             BEQ       TSTBS1
644  20DC  BD E1 AA          JSR       PRINTBUS
645  20DF  BD 20 8D  TSTBS1  JSR       X.L.ROTATE
646  20E2  4A                DEC       A
647  20E3  26 EF             BNE       FORBUS2
648  20E5  CE FF FE          LDX       =0FFFE    ROTATING 0
649  20E8  5A                DEC       B
650  20E9  26 E7             BNE       FORBUS1
651  20EB  BD C1 5F          JSR       EPMIR
652  20EE  BD 20 4D          JSR       END.TEST
653  20F1  39                RTS
654                *
655  20F2  43 50 55  BUSTEXT   BYTE      'CPU BUS',0
     20F5  20 42 55
     20F8  53 00
656                *
657                ************************************************
658                *
659                INSPECT     EQU   *
660  20FA                      BEGIN INSPECT
661                VALUE     EQU       HELP1
662                CRNT.ADR  EQU       HELP2
663                *
664                *           FORMAT
```

```
665                     *           ADDRESS 'I' VALUE [ ':' NEWVALUE ] [',']
666                     *           ADDRESS  = UP TO 4 HEX DIGIT
667                     *           NEWVALUE = UP TO 4 HEX DIGIT
668                     *
669                     *           PROCEDURE INSPECT;
670                     *             BEGIN X := X.WORD;
671                     *              LOOP
672                     *                CRNT.ADR := X; (A,B) := RMM(X);
673                     *                VALUE := (A,B); OUT.WORD(VALUE);
674                     *                R.CH(A);
675                     *                IF A =':' THEN
676                     *                  GET.WORD; (A,B) := X.WORD; X := CRNT.ADR; WMM(A,B,X);
677                     *                END;
678                     *                IF A = BACKSLASH
679                     *                THEN
680                     *                  DEC(X);
681                     *                ELSIF A = ','
682                     *                THEN
683                     *                  INC(X);
684                     *                ELSE
685                     *                  EXIT
686                     *                END;
687                     *                CRNT.ADR := X;
688                     *                CRLF; OUT.WORD(CRNT.ADR); WRITE(' ')
689                     *              END
690                     *             END
691                     *
692  20FA  DE 0F                    LDX   X.WORD
693  20FC  DF 30                    STX   CRNT.ADR
694  20FE  BD C1 58                 JSR   EDMIR
695  2101  BD C1 FE  LOOP          JSR   RMM
696  2104  D7 2E                    STAB  VALUE       HIGHER PART
697  2106  97 2F                    STAA  VALUE+1     LOWER PART
698  2108  CE 00 2E                 LDX   =VALUE
699  210B  BD C0 E3                 JSR   OUT.WORD
700  210E  BD C0 24                 JSR   R.CH
701  2111  81 3A                    CMPA  =':'
702  2113  26 0E                    BNE   ENDIF1
703  2115  BD C2 FE                 JSR   GET.WORD
704  2118  36                      PSH   A           SAVE DELIMITER
705  2119  D6 0F                    LDAB  X.WORD      HIGHER PART
706  211B  96 10                    LDAA  X.WORD+1    LOWER PART
707  211D  DE 30                    LDX   CRNT.ADR
708  211F  BD C2 28                 JSR   WMM
709  2122  32                      PUL   A           GET DELIMITER BACK
710  2123  81 2C     ENDIF1       CMPA  =COMMA
711  2125  27 08                    BEQ   ENDIF2
712  2127  81 5C                    CMPA  =BACK.SLASH
713  2129  27 09                    BEQ   ENDIF3
714  212B  BD C1 5F                 JSR   EPMIR
715  212E  39                      RTS
716
717  212F  DE 30     ENDIF2       LDX   CRNT.ADR
718  2131  08                      INX
719  2132  20 03                    BRA   ENDIF
720  2134  DE 30     ENDIF3       LDX   CRNT.ADR
721  2136  09                      DEX
```

```
722  2137  DF 30     ENDIF       STX   CRNT.ADR
723  2139  BD C0 78               JSR   W.CRLF
724  213C  BD C2 87               JSR   PRINT.X
725  213F  BD C0 62               JSR   W.1.BLANK
726  2142  20 BD                 BRA   LOOP
727  2144                         END   INSPECT
728                    *
729                    ***********************************************
730                    *
731                    *  PROCEDURE MANUAL.CHECK;
732                    *    (* ALLOWS MANUAL SETTING OF MICROINSTRUCTIONS,
733                    *       CPU-BUS AND CLOCK CONTROL
734                    *       => CPU-BUS IS ADDITIONALLY CONTROLLED BY THE
735                    *       BUS SOURCE FIELD OF A MICROINSTRUCTION *)
736                    *    BEGIN EDMIR;
737                    *      LOOP
738                    *        WRITELN(CPUBUS,' ',UAR);
739                    *        READ(A)
740                    *        CASE A OF
741                    *        'I': FOR B := 4 TO 0 BY -1 DO
742                    *                MIR[I] := IN.BYTE; EDMIR
743                    *        'B': CPU.BUS := HEX.IN
744                    *        'G': START RUNNING CLOCK
745                    *        'H': HALT RUNNIG CLOCK
746                    *        CR : RESET; EXIT
747                    *        ELSE ONE SINGLE STEP
748                    *        END;
749                    *        WRITE(' ')
750                    *      END
751                    *    END
752                    *
753  2144  BD C1 58  MANUALCHK JSR EDMIR
754  2147  FE 3C 0E  M.LOOP    LDX       CPU1
755  214A  BD C2 87            JSR       PRINTX
756  214D  BD C0 62            JSR       W.1.BLANK
757  2150  B6 3C 10            LDAA      UAR1
758  2153  84 0F                ANDA      =0F       MASK OFF  NOT USED BITS
759  2155  BD C0 C9            JSR       OUT.BYTE
760  2158  B6 3C 11            LDAA      UAR0
761  215B  BD C0 C9            JSR       OUT.BYTE
762  215E  BD C0 78            JSR       W.CRLF
763  2161  BD C0 24            JSR       R.CH
764  2164  81 49                CMPA      ='I'
765  2166  26 16                BNE       B.LABEL
766  2168  C6 05     I.LABEL    LDAB      =5
767  216A  CE 3C 08            LDX       =MIR4
768  216D  BD C0 BC  I.LOOP     JSR       IN.BYTE
769  2170  A7 00                STAA      0,X
770  2172  08                  INX
771  2173  BD C0 62            JSR       W.1.BLANK
772  2176  5A                  DEC       B
773  2177  26 F4               BNE       I.LOOP
774  2179  BD C1 58            JSR       EDMIR
775  217C  20 C9               BRA       M.LOOP
776  217E  81 42     B.LABEL    CMPA      ='B'
777  2180  26 08               BNE       G.LABEL
778  2182  CE 3C 0E            LDX       =CPU1
```

```
779  2185  BD C0 D8          JSR       IN.WORD
780  2188  20 BD             BRA       M.LOOP
781  218A  81 47    G.LABEL   CMPA      ='G'
782  218C  26 05             BNE       H.LABEL
783  218E  BD C1 66          JSR       START
784  2191  20 B4             BRA       M.LOOP
785  2193  81 48    H.LABEL   CMPA      ='H'
786  2195  26 05             BNE       CR.LABEL
787  2197  BD C1 6D          JSR       STOP
788  219A  20 AB             BRA       M.LOOP
789  219C  81 0D    CR.LABEL  CMPA      =CR
790  219E  26 04             BNE       ELSE.LAB
791  21A0  BD C2 90          JSR       RESET
792  21A3  39                RTS
793  21A4  B7 3C 15 ELSE.LAB  STAA      SSP       SINGLE STEP
794  21A7  BD C0 62          JSR       W.1.BLANK
795  21AA  20 9B             BRA       M.LOOP
796                *
797                *
798                ************************************************************
799                *
800                *         ENTRY TO THE HARDWARE MONITOR
801                *
802                ************************************************************
803                *
804  21AC  CE 21 BA HWTEST    LDX       =HW.TEXT
805  21AF  DF 0B             STX       MAIN.ID
806  21B1  BD C2 90          JSR       RESET
807  21B4  CE 21 C1          LDX       =HW.MENU
808  21B7  7E C3 91          JMP       COMMANDS
809                *
810                ************************************************************
811                *
812  21BA  48 57 54 HWTEXT    BYTE      'HWTEST',0
     21BD  45 53 54
     21C0  00
813  21C1  06       HW.MENU   BYTE      06
814  21C2  58 20 45          BYTE      'X EXECUTE CHAIN',0
     21C5  58 45 43
     21C8  55 54 45
     21CB  20 43 48
     21CE  41 49 4E
     21D1  00
815  21D2  20 3D             WORD      LOOPCHAIN
816  21D4  4C 20 4C          BYTE      'L LOOP ROUTINE',0
     21D7  4F 4F 50
     21DA  20 52 4F
     21DD  55 54 49
     21E0  4E 45 00
817  21E3  20 44             WORD      LOOP.RTNE
818  21E5  54 20 54          BYTE      'T TEST BUS',0
     21E8  45 53 54
     21EB  20 42 55
     21EE  53 00
819  21F0  20 BA             WORD      TESTBUS
820  21F2  48 20 44          BYTE      'H DO BY HAND',0
     21F5  4F 20 42
```

```
     21F8  59 20 48
     21FB  41 4E 44
     21FE  00
821  21FF  21 44              WORD      MANUALCHK
822  2201  55 20 49           BYTE      'U IFU',0
     2204  46 55 00
823  2207  24 67              WORD      IFU.TEST
824  2209  4D 20 55           BYTE      'M UMEM',0
     220C  4D 45 4D
     220F  00
825  2210  29 13              WORD      MICROMEM
826                    *
827  2212  09          GLOB.CHAIN BYTE   GL.CH.LEN+IFU.CH.LEN
828  2213  20 4A 21           WORD      SET.MAIN.ID,HWTEXT
     2216  BA
829  2217  20 BA 00           WORD      TESTBUS,0
     221A  00
830  221B  20 3A 24           WORD      NEXTCHAIN,IFU.CHAIN-1
     221E  D7
831                    GL.CH.LEN EQU     *-1-GLOB.CHAIN/4
832                    ************************************************
833                    *
834  221F              END       HARDWARE
```

```
836                     *
837                     ******************************************************
838                     *
839                     *                  IFU TESTS
840                     *
841                     *
842                     *
843                     *           10 OCT 1979
844                     *           F. OSTLER
845                     *           JIRKA HOPPE
846                     *
847                     ******************************************************
848  221F                         BEGIN     IFU
849                               USE       END.TEST,CHAIN,NEXTCHAIN,SET.MAIN.ID,INSPECT
850                               DEF       IFU.TEST,IFU.CHAIN,IFU.CH.LEN
851                     *
852                     *  LOAD AND READ F REGISTER
853                     *
854                     *  FORMAT: F
855                     *
856                     *  PROCEDURE TEST.FREG;
857                     *  BEGIN
858                     *    SUB.ID := 'LOAD.F.REG';
859                     *    EDMIR;
860                     *    MIR := SAFE.STATE;
861                     *    MIR0 := (DST=F, SRC=PANEL);
862                     *    FOR X := 0 TO FFFF DO
863                     *      CPU := X; SSP;
864                     *      MIR0 := (DST=-, SRC=F);
865                     *      IF X # CPU THEN PRINTBUS;  (* B,A=ACTUAL, X=EXPECTED*)
866                     *      MIR0 := (DST=F, SRC=PANEL);
867                     *    END;
868                     *  END;
869                     *
870  221F  CE 22 50  TEST.FREG LDX       =LD.F.TEXT
871  2222  DF 0D               STX       SUB.ID
872  2224  BD C1 58           JSR       EDMIR
873  2227  BD C1 FE           JSR       RMM       ONLY TO PUT MIR IN A SAFE STATE
874  222A  86 ED              LDAA      =0ED       DST=F, SRC=PANEL
875  222C  B7 3C 0C           STAA      MIR0
876  222F  CE 00 00           LDX       =00
877  2232  FF 3C 0E  FOR1     STX       CPU1
878  2235  B7 3C 15           STAA      SSP
879  2238  86 FE              LDAA      =0FE       DST=-, SRC=F
880  223A  B7 3C 0C           STAA      MIR0
881  223D  BC 3C 0E           CPX       CPU1
882  2240  27 03              BEQ       ENDIF1
883  2242  BD E1 AA           JSR       PRINTBUS  ** B,A=ACTUAL, X=EXPECTED *******
884  2245  86 ED     ENDIF1    LDAA      =0ED       DST=F, SRC=PANEL
885  2247  B7 3C 0C           STAA      MIR0
886  224A  08                 INX
887  224B  26 E5              BNE       FOR1      ZERO IF X ROLLED FROM FFFF TO 0
888  224D  7E 20 4D           JMP       END.TEST
889                     *
890  2250  4C 4F 41  LD.F.TEXT BYTE      'LOAD.F.REG',0
     2253  44 2E 46
```

```
                2256  2E 52 45
                2259  47 00
891                   *
892                   ************************************************************
893                   *
894                   *  LOAD AND READ OFFSET
895                   *
896                   *  PROCEDURE TEST.OFFSET;  (*LOAD ALL COMBINATIONS INTO OFFSET AND READ*)
897                   *  BEGIN
898                   *    SUB.ID := 'LOAD.OFFSET';
899                   *    EDMIR;
900                   *    MIR := SAFESTATE;
901                   *    MIR0 := (DST=OFFSET, SRC=PANEL);
902                   *    FOR X := 0 TO 0FFFF DO
903                   *      CPU:=X; SSP:
904                   *      MIR0 :=(DST=-, SRC=OFFSET);
905                   *      IF X # CPU THEN PRINTBUS;  (* B,A = ACTUAL, X = DESIRED*)
906                   *      MIR0:=(DST=OFFSET, SRC=PANEL);
907                   *    END;
908                   *  END;  (*TEST.OFFSET*)
909                   *
910   225B  CE 22 8C  TST.OFFS  LDX      =OFF.TXT
911   225E  DF 0D               STX       SUB.ID
912   2260  BD C1 58            JSR       EDMIR
913   2263  BD C1 FE            JSR       RMM       ONLY TO PUT MIR INTO DEFINED STATE
914   2266  86 2D               LDAA      =2D       DST=OFFSET, SRC=PANEL
915   2268  B7 3C 0C            STAA      MIR0
916   226B  CE 00 00            LDX       =0
917   226E  FF 3C 0E  FOR2      STX       CPU1
918   2271  B7 3C 15            STAA      SSP
919   2274  86 F2               LDAA      =0F2       DST=-, SRC=OFFSET
920   2276  B7 3C 0C            STAA      MIR0
921   2279  BC 3C 0E            CPX       CPU1
922   227C  27 03               BEQ       ENDIF2
923   227E  BD E1 AA            JSR       PRINTBUS  **B,A=ACTUAL; X=EXPECTED **
924   2281  86 2D     ENDIF2    LDAA      =2D       DST=OFFSET, SCR=PANEL
925   2283  B7 3C 0C            STAA      MIR0
926   2286  08                  INX
927   2287  26 E5               BNE       FOR2      END FORLOOP IF X ROLLS OVER
928   2289  7E 20 4D            JMP       END.TEST
929                   *
930   228C  4C 4F 41  OFF.TXT   BYTE      'LOAD OFFSET',0
      228F  44 20 4F
      2292  46 46 53
      2295  45 54 00
931                   *
932                   ************************************************************
933                   *
934                   *           INCREMENT OFFSET TEST
935                   *
936                   * PROCEDURE TEST.INC;
937                   * BEGIN
938                   *   EDMIR;
939                   *   MIR:=SAFESTATE;
940                   *   CLR.OFFSET;
941                   *   MIR0:=(DST=ALU, SRC=IR8+);
942                   *   SUB.ID:='INCPC.IR8+';
```

```
943                     *   CYCLEPC;
944                     *   MIR0:=(DST=ALU, SRC=IR8-);
945                     *   SUB.ID:='INCPC.IR8-';
946                     *   CYCLEPC;
947                     *   MIR0:=(DST=IRM, SRC=PANEL);
948                     *   CPU0:=FFH;   (*DISABLE INTERRUPTS TO ASSURE GENERATION OF MAP ENBL L*)
949                     *   SSP;
950                     *   MIR0:=(DST=ALU, SRC=IR4);
951                     *   SUB.ID:='INCPC.IR4';
952                     *   CYCLEPC;
953                     * END;
954                     *
955  2298  BD C1 58  TST.INC   JSR       EDMIR
956  229B  BD C1 FE           JSR       RMM       PUTS MIR INTO DEFINED STATE
957  229E  BD 23 34           JSR       CLR.OFFSET
958  22A1  86 04              LDAA      =04       DST=ALU, SRC=IR8+
959  22A3  B7 3C 0C           STAA      MIR0
960  22A6  CE 22 E6           LDX       =IRPL.TXT
961  22A9  DF 0D              STX       SUB.ID
962  22AB  BD 23 11           JSR       CYCLEPC
963  22AE  86 05              LDAA      =05       DST=ALU, SRC=IR8-
964  22B0  B7 3C 0C           STAA      MIR0
965  22B3  CE 22 F1           LDX       =IRMIN.TXT
966  22B6  DF 0D              STX       SUB.ID
967  22B8  BD 23 11           JSR       CYCLEPC
968  22BB  86 06              LDAA      =06       DST=ALU, SRC=IR8*
969  22BD  B7 3C 0C           STAA      MIR0
970  22C0  CE 22 FC           LDX       =IRSTR.TXT
971  22C3  DF 0D              STX       SUB.ID
972  22C5  BD 23 11           JSR       CYCLEPC
973  22C8  86 8D              LDAA      =8D       DST=IRM, SRC=PANEL
974  22CA  B7 3C 0C           STAA      MIR0
975  22CD  CE 00 FF           LDX       =0FF
976  22D0  FF 3C 0E           STX       CPU1
977  22D3  B7 3C 15           STAA      SSP
978  22D6  86 03              LDAA      =03       DST=ALU, SRC=IR4
979  22D8  B7 3C 0C           STAA      MIR0
980  22DB  CE 23 07           LDX       =MAPEN.TXT
981  22DE  DF 0D              STX       SUB.ID
982  22E0  BD 23 11           JSR       CYCLEPC
983  22E3  7E 20 4D           JMP       END.TEST
984                     *
985  22E6  49 4E 43  IRPL.TXT  BYTE      'INCPC.IR8+',0
     22E9  50 43 2E
     22EC  49 52 38
     22EF  2B 00
986  22F1  49 4E 43  IRMIN.TXT BYTE      'INCPC.IR8-',0
     22F4  50 43 2E
     22F7  49 52 38
     22FA  2D 00
987  22FC  49 4E 43  IRSTR.TXT BYTE      'INCPC.IR8*',0
     22FF  50 43 2E
     2302  49 52 38
     2305  2A 00
988  2307  49 4E 43  MAPEN.TXT BYTE      'INCPC.IR4',0
     230A  50 43 2E
     230D  49 52 34
```

```
     2310  00
 989                 *
 990                 *
 991                 ******************************************************
 992                 *
 993                 * PROCEDURE CYCLEPC; (*ASSUMES OFFSET=0 AT START AND LEAVES IT ZERO*)
 994                 * BEGIN
 995                 *   FOR X:=1 TO 10000H DO
 996                 *     SSP;
 997                 *     A:=MIR0;
 998                 *     MIR0:=(DST=-, SRC=OFFSET);
 999                 *     IF CPU#X THEN PRINTBUS;
1000                 *     MIR0 := A;
1001                 *   END;
1002                 *   CLROFFSET;
1003                 * END;
1004                 *
1005  2311  CE 00 00 CYCLEPC   LDX       =0
1006  2314  08       FOR3      INX
1007  2315  B7 3C 15           STAA      SSP
1008  2318  B6 3C 0C           LDAA      MIR0
1009  231B  C6 F2              LDAB      =0F2        DST=-, SRC=OFFSET
1010  231D  F7 3C 0C           STAB      MIR0
1011  2320  BC 3C 0E           CPX       CPU1
1012  2323  27 03              BEQ       ENDIF3
1013  2325  BD E1 AA           JSR       PRINTBUS  ** B,A=ACTUAL, X=EXPECTED **
1014  2328  B7 3C 0C ENDIF3    STAA      MIR0
1015  232B  8C 00 00           CPX       =0
1016  232E  26 E4              BNE       FOR3      IF OFFSET JUST ROLLED OVER THEN QUIT
1017  2330  BD 23 34           JSR       CLROFFSET
1018  2333  39                RTS
1019                 *
1020                 ************************************************************
1021                 *
1022                 * PROCEDURE CLROFFSET;  (*CHANGES MIR*)
1023                 * BEGIN
1024                 *  MIR:=SAFESTATE;
1025                 *  MIR0:=(DST=OFFSET, SRC=PANEL);
1026                 *  CPU:=0000H;
1027                 *  SSP;
1028                 * END;
1029                 *
1030  2334  BD C1 FE CLROFFSET JSR       RMM       PUT MIR INTO DEFINED STATE
1031  2337  36                PSH       A
1032  2338  86 2D              LDAA      =2D       DST=OFFSET, SRC=PANEL
1033  233A  B7 3C 0C           STAA      MIR0
1034  233D  7F 3C 0E           CLR       CPU1
1035  2340  7F 3C 0F           CLR       CPU0
1036  2343  B7 3C 15           STAA      SSP
1037  2346  32                PUL       A
1038  2347  39                RTS
1039                 *
1040                 ************************************************************
1041                 *
1042                 *       TEST IFU SEQUENCING
1043                 *
1044                 * PROCEDURE IFU.SEQ; (*READ ALL COMBINATIONS OF INSTRUCTION BYTES
```

```
1045                    *                       FROM MAIN MEMORY INTO EACH OF THE 8 IFU REGISTERS*)
1046                    * BEGIN
1047                    *   SUB.ID:='IFU.SEQUENCE';
1048                    *   EDMIR;
1049                    *   FOR HELP1:=0 TO 7 DO
1050                    *     B:=HELP1+1; A:=HELP1;
1051                    *     FOR X:=0 TO 7FH DO
1052                    *       MAINMEMORY[X]:=(B MOD 256)*256 + A MOD 256;
1053                    *       INC(B,2); INC(A,2);
1054                    *     END;
1055                    *     CLROFFSET;
1056                    *     MIR0:=(DST=F, SRC=PANEL);
1057                    *     CPU:=0;
1058                    *     SSP;
1059                    *     MIR0:=(DST=-,SRC=IR8+);
1060                    *     B:=HELP1;
1061                    *     FOR X:=0 TO FF DO
1062                    *       IF CPU0 # (B MOD 256)
1063                    *       THEN
1064                    *         A:=CPU0;
1065                    *         MIR0:=(DST=-, SRC=OFFSET);
1066                    *         PUSH(X);
1067                    *         X := CPU;
1068                    *         SWI;      (*B=DESIRED BYTE, A=ACTUAL BYTE, X=OFFSET*)
1069                    *         PULL(X);
1070                    *         MIR0:=(DST=-, SRC=IR8+);
1071                    *       END; (*IF*)
1072                    *       SSP;
1073                    *       INC(B);
1074                    *     END; (*FOR*)
1075                    *   END; (*FOR*)
1076                    * END
1077                    *
1078                    *
1079  2348  CE 23 BC  IFU.SEQ   LDX       =SEQ.TEXT
1080  234B  DF 0D               STX       SUB.ID
1081  234D  BD C1 58            JSR       EDMIR
1082  2350  7F 00 2E            CLR       HELP1
1083  2353  D6 2E      FORL1    LDAB      HELP1
1084  2355  17                 TBA
1085  2356  8B 01               ADDA      =1
1086  2358  CE 00 00            LDX       =0
1087  235B  BD C2 28            JSR       WMM
1088  235E  8C 00 7F  FORL2     CPX       =7F
1089  2361  27 0A               BEQ       ENDFORL2
1090  2363  08                 INX
1091  2364  8B 02               ADDA      =2
1092  2366  CB 02               ADDB      =2
1093  2368  BD C2 3D            JSR       WMMS
1094  236B  20 F1               BRA       FORL2
1095  236D  86 ED      ENDFORL2 LDAA      =0ED       DST=F, SRC=PANEL
1096  236F  B7 3C 0C            STAA      MIR0
1097  2372  7F 3C 0E            CLR       CPU1
1098  2375  7F 3C 0F            CLR       CPU0
1099  2378  B7 3C 15            STAA      SSP
1100  237B  BD 23 34            JSR       CLR.OFFSET IT LOADS THE FIRST BYTE FROM THE MEMORY
1101  237E  86 F4               LDAA      =0F4       DST=-, SRC=IR8+
```

```
1102  2380  B7 3C 0C            STAA      MIR0
1103  2383  D6 2E               LDAB      HELP1
1104  2385  CE 00 00            LDX       =0
1105  2388  F1 3C 0F  FORL3     CMPB      CPU0
1106  238B  27 19               BEQ       ENDIFA
1107  238D  B6 3C 0F            LDAA      CPU0
1108  2390  36                  PSH       A
1109  2391  86 F2               LDAA      =0F2       DST=-, SRC=OFFSET
1110  2393  B7 3C 0C            STAA      MIR0
1111  2396  32                  PUL       A
1112  2397  BD C2 59            JSR       PUSH.X
1113  239A  FE 3C 0E            LDX       CPU1
1114  239D  3F                  SWI                  **B=EXPECTED BYTE, A =ACTUAL BYTE, X=OFFSET **
1115  239E  BD C2 70            JSR       PULL.X
1116  23A1  86 F4               LDAA      =0F4       DST=-, SRC=IR8+
1117  23A3  B7 3C 0C            STAA      MIR0
1118  23A6  B7 3C 15  ENDIFA    STAA      SSP
1119  23A9  5C                  INC       B
1120  23AA  08                  INX
1121  23AB  8C 01 00            CPX       =0100
1122  23AE  26 D8               BNE       FORL3
1123  23B0  7C 00 2E            INC       HELP1
1124  23B3  86 07               LDAA      =7
1125  23B5  91 2E               CMPA      HELP1
1126  23B7  2C 9A               BGE       FORL1
1127  23B9  7E 20 4D            JMP       END.TEST
1128                  *
1129  23BC  49 46 55  SEQ.TEXT  BYTE      'IFUSEQ',0
      23BF  53 45 51
      23C2  00
1130                  *
1131                  ***********************************************
1132                  *
1133                  *  TEST THE IFU ADDER
1134                  *
1135                  *  PROCEDURE ADDER;
1136                  *    CONST INC.F= 152; INC.OFFS = 174;  (* BOTH MUST BE EVEN *)
1137                  *          (* THOSE CONSTANTS HAVE NO FIXED MEANING
1138                  *             THEY ARE USED TO SPEED UP THE TEST ONLY *)
1139                  *    VAR HELP1 : F.REGISTER;
1140                  *        HELP2 : OFFSET;
1141                  *        HELP3 : EXPECTED VALUE;
1142                  *    BEGIN SUB.ID := 'ADDER';
1143                  *      (*FILL MEMORY*)
1144                  *      FOR X := 0 TO 0FFFFH DO WMM(X,X) END;
1145                  *
1146                  *      HELP1 :=0;
1147                  *      LOOP  F.REG := HELP1;
1148                  *        HELP2 := 0;
1149                  *        LOOP  OFFSET := HELP2;
1150                  *          HELP3 := (HELP2 DIV 2) + (HELP1 *2);
1151                  *              (* THIS IS TO CORRECT THE SHIFTED POSITION OF F.REG AND OFFSET*)
1152                  *          IF HELP3>0FFFFH THEN EXIT END;
1153                  *          B := FETCH(NEXT INSTR);  A := FETCH(NEXT INSTR);
1154                  *          IF (B,A) <> HELP3 THEN S W I(B,A,HELP3) END;
1155                  *          HELP2 := HELP2 + INC.OFFS;
1156                  *        END;
```

```
1157                    *       HELP1 := HELP1 + INC.F;
1158                    *       IF HELP1 > 0FFFFH THEN EXIT END
1159                    *     END
1160                    *   END ADDER;
1161                    *
1162                    INC.F     EQU     152Z      MUST BE EVEN
1163                    INC.OFFS  EQU     174Z
1164                    *
1165  23C3  CE 24 59    ADDER     LDX     =ADDER.TXT
1166  23C6  DF 0D               STX     SUB.ID
1167  23C8  BD C1 58            JSR     EDMIR
1168  23CB  BD C2 28            JSR     WMM       JUST TO DO SLOW VERSION OF 'WMM' ONCE
1169  23CE  CE 00 00            LDX     =0
1170  23D1  DF 2E       FILL.LP STX     HELP1
1171  23D3  D6 2E               LDAB    HELP1
1172  23D5  96 2F               LDAA    HELP1+1
1173  23D7  BD C2 3D            JSR     WMMS      FAST VERSION OF WMM
1174  23DA  08                  INX
1175  23DB  26 F4               BNE     FILL.LP
1176                    *
1177  23DD  CE 00 00            LDX     =0
1178  23E0  DF 2E               STX     HELP1
1179  23E2  DE 2E       AD.LP.1 LDX     HELP1
1180  23E4  FF 3C 0E            STX     CPU1
1181  23E7  86 ED               LDAA    =0ED      SOURCE=PANEL; DEST=F.REG
1182  23E9  B7 3C 0C            STAA    MIR0
1183  23EC  B7 3C 15            STAA    SSP
1184  23EF  CE 00 00            LDX     =0
1185  23F2  DF 30               STX     HELP2
1186  23F4  DE 30       AD.LP.2 LDX     HELP2
1187  23F6  FF 3C 0E            STX     CPU1
1188  23F9  86 2D               LDAA    =2D       SOURCE=PANEL; DEST=OFFSET
1189  23FB  B7 3C 0C            STAA    MIR0
1190  23FE  B7 3C 15            STAA    SSP
1191                    *
1192  2401  96 31               LDAA    HELP2+1   X.SAVE := HELP2 DIV 2
1193  2403  D6 30               LDAB    HELP2
1194  2405  54                  LSR     B
1195  2406  46                  ROR     A
1196  2407  97 02               STAA    X.SAVE+1
1197  2409  D7 01               STAB    X.SAVE
1198                    *
1199  240B  96 2F               LDAA    HELP1+1   (B,A) := 2*HELP1
1200  240D  D6 2E               LDAB    HELP1
1201  240F  48                  ASL     A
1202  2410  59                  ROL     B
1203  2411  25 33               BCS     END.LP.2  IF OVERFLOW
1204  2413  9B 02               ADDA    X.SAVE+1
1205  2415  D9 01               ADCB    X.SAVE
1206  2417  25 2D               BCS     END.LP.2
1207  2419  97 33               STAA    HELP3+1
1208  241B  D7 32               STAB    HELP3     HELP3 := (HELP1 DIV 2) + (HELP2 * 2)
1209                    *
1210  241D  86 F4               LDAA    =0F4       SOURCE=IR8; DEST= NONE
1211  241F  B7 3C 0C            STAA    MIR0
1212  2422  F6 3C 0F            LDAB    CPU0      FETCH FIRST INSTRUCTION BYTE
1213  2425  B7 3C 15            STAA    SSP
```

```
1214  2428  B6 3C 0F          LDAA      CPU0      FETCH SECOND INSTRUCTION BYTE
1215                     *
1216  242B  91 33             CMPA      HELP3+1
1217  242D  26 04             BNE       ADD.ERR
1218  242F  D1 32             CMPB      HELP3
1219  2431  27 03             BEQ       ADD.NO.ERR
1220  2433  DE 32     ADD.ERR  LDX       HELP3
1221  2435  3F                 SWI                 (B,A)=ACTUAL; X=EXPECTED  **********
1222  2436  96 31     ADD.NO.ERR LDAA     HELP2+1
1223  2438  D6 30             LDAB      HELP2
1224  243A  8B AE             ADDA      =INC.OFFS
1225  243C  C9 00             ADCB      =0
1226  243E  25 06             BCS       END.LP.2
1227  2440  97 31             STAA      HELP2+1
1228  2442  D7 30             STAB      HELP2
1229  2444  20 AE     NO.CARRY1 BRA      AD.LP.2
1230                     *
1231  2446  96 2F     END.LP.2  LDAA      HELP1+1
1232  2448  D6 2E             LDAB      HELP1
1233  244A  8B 98             ADDA      =INC.F
1234  244C  C9 00             ADCB      =0        INC.F IS LESS THEN 255; WE ADD CARRY ONLY
1235  244E  25 06             BCS       ADD.END
1236  2450  97 2F             STAA      HELP1+1
1237  2452  D7 2E             STAB      HELP1
1238  2454  20 8C             BRA       AD.LP.1
1239                     *
1240  2456  7E 20 4D     ADD.END   JMP       END.TEST
1241                     *
1242  2459  41 44 44     ADDER.TXT BYTE      'ADDER',0
      245C  45 52 00
1243
1244                     ************************************************************
1245                     *
1246                     *    LOOP ALL IFU TESTS IN A CHAIN
1247                     *
1248  245F  CE 24 D8     LOOP.IFU  LDX       =IFU.CHAIN
1249  2462  BD 20 15             JSR       CHAIN
1250  2465  20 F8                BRA       LOOP.IFU
1251                     *
1252                     ****************************************************************
1253                     *
1254                     *  ENTRY POINT FOR THE IFU TEST
1255                     *
1256                     ****************************************************************
1257                     *
1258  2467  CE 24 F1     IFU.TEST  LDX       =IFU.ID
1259  246A  DF 0B                STX       MAIN.ID
1260  246C  CE 24 72             LDX       =IFU.MENU
1261  246F  7E C3 91             JMP       COMMANDS
1262                     *
1263  2472  07           IFU.MENU  BYTE      7
1264  2473  58 20 45             BYTE      'X EXEC CHAIN',0
      2476  58 45 43
      2479  20 43 48
      247C  41 49 4E
      247F  00
1265  2480  24 5F                WORD      LOOP.IFU
```

```
1266  2482  52 20 54          BYTE      'R TEST.F.REG',0
      2485  45 53 54
      2488  2E 46 2E
      248B  52 45 47
      248E  00
1267  248F  22 1F             WORD      TEST.FREG
1268  2491  4F 20 54          BYTE      'O TEST OFFSET',0
      2494  45 53 54
      2497  20 4F 46
      249A  46 53 45
      249D  54 00
1269  249F  22 5B             WORD      TST.OFFS
1270  24A1  4E 20 49          BYTE      'N INC OFFSET',0
      24A4  4E 43 20
      24A7  4F 46 46
      24AA  53 45 54
      24AD  00
1271  24AE  22 98             WORD      TST.INC
1272  24B0  53 20 53          BYTE      'S SEQUENCING',0
      24B3  45 51 55
      24B6  45 4E 43
      24B9  49 4E 47
      24BC  00
1273  24BD  23 48             WORD      IFU.SEQ
1274  24BF  50 20 50          BYTE      'P PC ADDER',0
      24C2  43 20 41
      24C5  44 44 45
      24C8  52 00
1275  24CA  23 C3             WORD      ADDER
1276  24CC  49 20 49          BYTE      'I INSPECT',0
      24CF  4E 53 50
      24D2  45 43 54
      24D5  00
1277  24D6  20 FA             WORD      INSPECT
1278                        *
1279                        ***********************************
1280                        *
1281  24D8  06          IFU.CHAIN BYTE    IFU.CH.LEN
1282  24D9  20 4A 24            WORD      SET.MAIN.ID,IFU.ID
      24DC  F1
1283  24DD  22 1F 00            WORD      TEST.FREG,0
      24E0  00
1284  24E1  22 5B 00            WORD      TST.OFFS,0
      24E4  00
1285  24E5  22 98 00            WORD      TST.INC,0
      24E8  00
1286  24E9  23 48 00            WORD      IFU.SEQ,0
      24EC  00
1287  24ED  23 C3 00            WORD      ADDER,0
      24F0  00
1288                        IFU.CH.LEN EQU    *-1-IFU.CHAIN/4
1289                        *
1290                        ***********************************
1291                        *
1292                        *
1293  24F1  49 46 55  IFU.ID    BYTE      'IFU',0
      24F4  00
```

```
1294
1295  24F5                    END       IFU
```

```
1298  24F5                      BEGIN DEVICES
1299                    ********************************************
1300                    *
1301                    *         DEVICES
1302                    *
1303                    *         AUTHOR JIRKA HOPPE       5-DEC-79
1304                    *
1305                    ********************************************
1306                              USE  INSPECT
1307                              DEF   DEVICES
1308                    ****************************************************
1309                    *
1310                    *         READ DEVICE
1311                    *         INPUT   A - DEVICE ADR
1312                    *         OUTPUT  X - VALUE
1313                    *
1314  24F5  B7 3C 0F    GET       STAA  CPU0        SET DEVICE ADR   BYTE ONLY
1315  24F8  CE 25 67              LDX   =SET.IO.ADR
1316  24FB  BD C1 17              JSR   EXX
1317                    *
1318  24FE  36                    PSH   A
1319  24FF  86 07                 LDAA  =07         BUS SOURCE = IODATA
1320  2501  B7 3C 0C              STAA  MIR0        THIS STARTS THE READ OPERATION
1321  2504  B7 3C 15              STAA  SSP      SINGLE STEP
1322  2507  B7 3C 15              STAA  SSP
1323  250A  FE 3C 0E              LDX   CPU1
1324  250D  B7 3C 15              STAA  SSP
1325  2510  B7 3C 15              STAA  SSP
1326  2513  86 FF                 LDAA  =0FF
1327  2515  B7 3C 0C              STAA  MIR0        DISABLE ANY BUS SOURCING IN ANY NEXT PROCEDURE
1328  2518  32                   PUL   A
1329  2519  39                   RTS
1330                    ****************************************************
1331                    *
1332                    *         WRITE DEVICE
1333                    *         INPUT   A - DEVICE ADR
1334                    *                 X - VALUE
1335                    *
1336  251A  B7 3C 0F    PUT       STAA  CPU0
1337  251D  BD C2 59              JSR   PUSHX
1338  2520  CE 25 67              LDX   =SET.IO.ADR
1339  2523  BD C1 17              JSR   EXX         SET DEVICE ADR
1340                    *
1341  2526  BD C2 70              JSR   PULLX
1342  2529  FF 3C 0E              STX   CPU1
1343  252C  36                    PSH   A
1344  252D  86 7D                 LDAA  =7D         SOURCE=PANEL; DEST=IOD
1345  252F  B7 3C 0C              STAA  MIR0
1346  2532  B7 3C 15              STAA  SSP         SINGLE STEP
1347  2535  B7 3C 15              STAA  SSP         SINGLE STEP
1348  2538  B7 3C 15              STAA  SSP         SINGLE STEP
1349  253B  B7 3C 15              STAA  SSP         SINGLE STEP
1350  253E  86 FF                 LDAA  =0FF        DISABLE BUS
1351  2540  B7 3C 0C              STAA  MIR0
1352  2543  32                   PUL   A
1353  2544  39                   RTS
```

```
1354                    ************************************************
1355                    *
1356                    *           CALL READ DEVICE
1357  2545  96 10       RD.DEV      LDAA  X.WORD+1
1358  2547  BD C1 58                JSR   EDMIR
1359  254A  BD 24 F5                JSR   GET
1360  254D  BD C1 5F                JSR   EPMIR
1361  2550  BD C2 87                JSR   PRINTX
1362  2553  39                     RTS
1363                    ************************************************
1364                    *           CALL WRITE DEVICE
1365  2554  96 10       WR.DEV      LDAA  X.WORD+1    ADR
1366  2556  36                     PSH   A
1367  2557  BD C2 FE                JSR   GETWORD     GET X VALUE
1368  255A  DE 0F                   LDX   X.WORD
1369  255C  32                     PUL   A
1370  255D  BD C1 58                JSR   EDMIR
1371  2560  BD 25 1A                JSR   PUT
1372  2563  BD C1 5F                JSR   EPMIR
1373  2566  39                     RTS
1374                    ************************************************
1375                    *
1376  2567  2E 00 07   SET.IO.ADR  BYTE  2E,0,7,60,6D
      256A  60 6D
1377                    ************************************************************
1378                    ******************************************
1379                    *
1380                    *  PROCEDURE KEYBOARD.TEST;
1381                    *    BEGIN W.CRLF;
1382                    *      LOOP
1383                    *       WRITE(GET(KB.DATA); WRITE(' '); WRITE(GET(KB.STATUS)); WRITE(CR)
1384                    *      END
1385                    *    END KEYBOARD.TEST;
1386                    *
1387  256C  BD C0 78   KEYBOARD  JSR       W.CRLF
1388  256F  BD C1 58             JSR       EDMIR
1389  2572  86 02     KB.LOOP   LDAA      =2        I/O ADR OF KEYBOARD DATA
1390  2574  BD 24 F5             JSR       GET
1391  2577  BD C2 87             JSR       PRINTX
1392  257A  BD C0 62             JSR       W.1.BLANK
1393  257D  CE F0 00             LDX       =0F000    A DELAY LOOP
1394  2580  09       KB.LP.1   DEX
1395  2581  26 FD              BNE       KB.LP.1
1396  2583  86 01              LDAA      =1        I/O ADR OF KEYBOARD STATUS
1397  2585  BD 24 F5             JSR       GET
1398  2588  BD C2 87             JSR       PRINTX
1399  258B  86 0D              LDAA      =CR       'CR' ONLY; STAY ON THE SAME LINE
1400  258D  BD C0 51             JSR       W.CH
1401  2590  BD C0 46             JSR       BREAK
1402  2593  4D                 TST       A
1403  2594  27 DC              BEQ       KB.LOOP
1404  2596  BD C1 5F             JSR       EPMIR
1405  2599  39                 RTS
1406                    *
1407                    ******************************************
1408                    *
1409                    *  PROCEDURE MOUSE.TEST
```

```
1410                    *    BEGIN W.CRLF
1411                    *      LOOP
1412                    *       WRITE(GET(M.X)); WRITE(GET(M.Y)); WRITE(GET(M.STAT)); WRITE(CR);
1413                    *      END
1414                    *    END MOUSE.TEST;
1415                    *
1416  259A  BD C0 78    MOUSE     JSR       W.CRLF
1417  259D  BD C1 58              JSR       EDMIR
1418  25A0  86 06       MS.LOOP   LDAA      =6        X COORDINATE
1419  25A2  BD 24 F5              JSR       GET
1420  25A5  BD C2 87              JSR       PRINTX
1421  25A8  BD C0 62              JSR       W.1.BLANK
1422  25AB  86 07                 LDAA      =7        Y COORDINATE
1423  25AD  BD 24 F5              JSR       GET
1424  25B0  BD C2 87              JSR       PRINTX
1425  25B3  BD C0 62              JSR       W.1.BLANK
1426  25B6  86 03                 LDAA      =3
1427  25B8  BD 24 F5              JSR       GET
1428  25BB  BD C2 87              JSR       PRINTX
1429  25BE  86 0D                 LDAA      =CR
1430  25C0  BD C0 51              JSR       W.CH
1431  25C3  BD C0 46              JSR       BREAK
1432  25C6  4D                    TST       A
1433  25C7  27 D7                 BEQ       MS.LOOP
1434  25C9  BD C1 5F              JSR       EPMIR
1435  25CC  39                    RTS
1436                    *
1437                    ***************************************************
1438                    *
1439                    *  PROCEDURE UART.TST;
1440                    *    (* AT THE BEGINNIG IS X.WORD SPECIFIED FROM COMMANDS *)
1441                    *    BEGIN
1442                    *      LOOP
1443                    *        REPEAT UNTIL TBRE IN GET(UART.ST); (* WAIT XMITER EMPTY *)
1444                    *        PUT(UART.DATA ,X.WORD);
1445                    *        IF DRDY IN GET(UART.ST) THEN X.WORD := GET(UART.DATA) END
1446                    *        IF BREAK THEN EXIT
1447                    *      END
1448                    *    END UART.TST;
1449                    *
1450  25CD  BD C1 58    UART      JSR       EDMIR
1451  25D0  86 05       UART.LP   LDAA      =5        UART.ST
1452  25D2  BD 24 F5              JSR       GET
1453  25D5  DF 2E                 STX       HELP1
1454  25D7  D6 2F                 LDAB      HELP1+1
1455  25D9  C4 01                 ANDB      =1        TBRE
1456  25DB  27 F3                 BEQ       UART.LP
1457  25DD  86 04                 LDAA      =4        UART.DATA
1458  25DF  DE 0F                 LDX       X.WORD
1459  25E1  BD 25 1A              JSR       PUT
1460                    *
1461  25E4  86 05                 LDAA      =5         UART.ST
1462  25E6  BD 24 F5              JSR       GET
1463  25E9  DF 2E                 STX       HELP1
1464  25EB  D6 2F                 LDAB      HELP1+1
1465  25ED  C4 02                 ANDB      =2         DBRDY
1466  25EF  27 07                 BEQ       NO.DATA
```

```
1467  25F1  86 04        LDAA    =4          UART.DATA
1468  25F3  BD 24 F5     JSR     GET
1469  25F6  DF 0F        STX     X.WORD
1470                    *
1471  25F8  BD C0 46  NO.DATA JSR BREAK
1472  25FB  4D          TST     A
1473  25FC  27 CF       BEQ     UART
1474  25FE  BD C1 5F    JSR     EPMIR
1475  2601  39          RTS
1476                    ****************************************
1477                    *
1478                    *         WRITE PATTERN INTO MEMORY
1479                    *
1480                    *         BEGIN (A,B) := X.WORD;
1481                    *           FOR X:=LOWDISP TO LOWDISP+DISPSIZE DO WMM(X,A,B)
1482                    *         END
1483                    *
1484                    LOW.DISP  EQU   1000            STARTING ADR OF DISPLAY MAP
1485                    DISP.SIZE EQU   768Z*596Z/16Z   SIZE OF DISPLAY MEMORY
1486                    *
1487  2602  96 10     D.PATTERN LDAA  X.WORD+1
1488  2604  D6 0F              LDAB  X.WORD
1489  2606  CE 10 00           LDX   =LOW.DISP
1490  2609  BD C2 28           JSR   WMM         DO THE SLOW VERSION OF WMM ONCE
1491  260C  BD C1 58           JSR   EDMIR
1492  260F  BD C2 3D  FOR      JSR   WMMS
1493  2612  08                INX
1494  2613  8C 7F C0          CPX   =LOW.DISP+DISP.SIZE
1495  2616  26 F7            BNE   FOR
1496  2618  BD C1 5F          JSR   EPMIR
1497  261B  39              RTS
1498                    *
1499                    ****************************************
1500                    *
1501                    *  PROCEDURE SET.MEMORY(X: ↑COMMAND STRING-4);
1502                    *    (* FORMAT OF THE COMMAND STRING:
1503                    *       VALUE, ADDRESS *)
1504                    *     BEGIN HELP1 := X;
1505                    *       LOOP INC(HELP1,4); (BA) := HELP1↑;
1506                    *         X := (HELP1+2)↑;
1507                    *         IF X=0 THEN EXIT END;
1508                    *         WMM(X,B,A)
1509                    *       END
1510                    *     END SET.MEMORY
1511                    *
1512  261C  DF 2E     SET.MEM   STX   HELP1
1513  261E  DE 2E     SET.M.LP  LDX   HELP1
1514  2620  08                INX
1515  2621  08                INX
1516  2622  08                INX
1517  2623  08                INX
1518  2624  DF 2E            STX   HELP1
1519  2626  E6 00            LDAB  0,X       HIGH OF VALUE
1520  2628  A6 01            LDAA  1,X       LOW OF VALUE
1521  262A  EE 02            LDX   2,X       ADDRESS
1522  262C  27 05            BEQ   S.M.END
1523  262E  BD C2 28         JSR   WMM
```

```
1524  2631  20 EB              BRA       SET.M.LP
1525  2633  39        S.M.END   RTS
1526                  *
1527                  ***********************************************
1528                  *
1529                  *  PROCEDURE TEST.PICTURE;
1530                  *   (* DRAW A SIMPLE TEST PICTURE ON THE SCREEN *)
1531                  *   BEGIN D.PATTERN(0); (* CLEAR THE WHOLE SCREEN *)
1532                  *     FOR X := LOW.DISP TO LOW.DISP+W DO WMM(X,FFFF) END;
1533                  *     SET.MEM(MEM.STRING);
1534                  *     PUT(0, LOW.DISP); (*START DISPLAY CONTROLLER *)
1535                  *   END
1536                  *
1537                  H.PICT    EQU       594Z      HEIGHT OF THE PICTURE
1538                  W.PICT    EQU       48Z       WIDTH OF THE PICTURE
1539                  BMD       EQU       0FFFC     ADDRESS OF THE BMD
1540                  *
1541  2634  BD C1 58 TEST.PICT JSR       EDMIR
1542  2637  86 00              LDAA      =0
1543  2639  CE FF FC           LDX       =BMD
1544  263C  BD 25 1A           JSR       PUT
1545  263F  CE 00 00           LDX       =0
1546  2642  DF 0F              STX       X.WORD
1547  2644  8D BC              BSR       D.PATTERN
1548
1549  2646  BD C1 58           JSR       EDMIR
1550                  *  LEFT + RIGHT MARGIN
1551  2649  CE 10 2F           LDX       =LOW.DISP+W.PICT-1
1552  264C  86 01     TST.PC.L3 LDAA      =1        RIGHT MARGIN
1553  264E  5F                 CLR       B
1554  264F  BD C2 28           JSR       WMM
1555  2652  4F                 CLR       A
1556  2653  C6 80              LDAB      =80
1557  2655  08                 INX
1558  2656  BD C2 3D           JSR       WMMS
1559  2659  DF 2E              STX       HELP1
1560  265B  96 2F              LDAA      HELP1+1   HELP1 := HELP1 + W.PICT - 1
1561  265D  8B 2F              ADDA      =W.PICT-1
1562  265F  97 2F              STAA      HELP1+1
1563  2661  24 03              BCC       T.P.NO.CARR1
1564  2663  7C 00 2E           INC       HELP1
1565  2666  DE 2E     T.P.NO.CARR1 LDX    HELP1
1566  2668  8C 7F 5F           CPX       =H.PICT*W.PICT+LOWDISP-1
1567  266B  26 DF              BNE       TST.PC.L3
1568                  *  UPPER LINE
1569  266D  CE 10 00           LDX       =LOW.DISP
1570  2670  86 AA              LDAA      =0AA
1571  2672  16                 TAB                 (B,A)=AAAA
1572  2673  BD C2 3D TST.PCT.L JSR       WMMS
1573  2676  08                 INX
1574  2677  8C 10 30           CPX       =LOW.DISP+W.PICT
1575  267A  26 F7              BNE       TST.PCT.L
1576                  *  LOWER LINE
1577  267C  CE 7F 00           LDX       =H.PICT-2*W.PICT+LOW.DISP
1578  267F  BD C2 3D TST.PC.L2 JSR       WMMS
1579  2682  08                 INX
1580  2683  8C 7F 30           CPX       =H.PICT-1*WPICT+LOW.DISP
```

```
1581  2686  26 F7              BNE       TST.PC.L2
1582                    *  DIAGONAL
1583  2688  CE 1C 00          LDX       =40*W.PICT+LOW.DISP    START AT 40TH LINE
1584  268B  86 FF     TSTPC.L4 LDAA      =0FF
1585  268D  16                TAB
1586  268E  BD C2 3D          JSR       WMMS
1587  2691  DF 2E              STX       HELP1
1588  2693  96 2F              LDAA      HELP1+1   HELP1 := HELP1 + W.PICT + 1
1589  2695  8B 31              ADDA      =W.PICT+1
1590  2697  97 2F              STAA      HELP1+1
1591  2699  24 03              BCC       T.P.NO.CARR2
1592  269B  7C 00 2E          INC       HELP1
1593  269E  DE 2E     T.P.NO.CARR2 LDX   HELP1
1594  26A0  8C 25 30          CPX       =30+40*W.PICT+30+LOW.DISP
1595  26A3  26 E6              BNE       TST.PC.L4
1596  26A5  CE 26 AB          LDX       =MEM.STRING-4
1597  26A8  BD 26 1C          JSR       SET.MEM
1598  26AB  BD C1 5F          JSR       EPMIR
1599  26AE  39                RTS
1600                    *
1601  26AF  10 00 FF  MEMSTRING WORD     LOW.DISP,BMD,W.PICT,BMD+1,H.PICT+1,BMD+2,0,BMD+3   BMD
      26B2  FC 00 30
      26B5  FF FD 02
      26B8  53 FF FE
      26BB  00 00 FF
      26BE  FF
1602                    *         TEST RESOLUTION
1603                    LINE5     EQU       5*W.PICT+LOW.DISP
1604  26BF  AA AA 10            WORD      0AAAA,LINE5,0AAAA,LINE5+W.PICT-1
      26C2  F0 AA AA
      26C5  11 1F
1605  26C7  AA AA 7E            WORD      0AAAA,H.PICT-4*W.PICT+LOW.DISP         LOWER LEFT EDGE
      26CA  A0
1606  26CB  AA AA 7E            WORD      0AAAA,H.PICT-4*W.PICT+LOW.DISP+W.PICT-1 LOWER RIGHT EDGE
      26CE  CF
1607                    *   MIDDLE OF THE PICTURE
1608                    W.PICT2   EQU       W.PICT/2
1609                    MIDDLE    EQU       H.PICT/2*W.PICT+W.PICT2+LOW.DISP
1610  26CF  AA AA 47            WORD      0AAAA,MIDDLE,9999,MIDDLE+W.PICT-1
      26D2  C8 99 99
      26D5  47 F7
1611  26D7  99 99 47            WORD      9999,MIDDLE+W.PICT+1
      26DA  F9
1612                    *   END
1613  26DB  00 00 00            WORD      0,0
      26DE  00
1614
1615                    *
1616                    ******************************************
1617                    *
1618  26DF  CE 26 EA  DEVICES     LDX   =DEV.TITLE
1619  26E2  DF 0B                 STX   MAIN.ID
1620  26E4  CE 26 F2              LDX   =DEV.MENU
1621  26E7  7E C3 91              JMP   COMMANDS
1622  26EA  44 45 56  DEV.TITLE   BYTE  'DEVICES',0
      26ED  49 43 45
      26F0  53 00
```

```
1623                        *
1624  26F2  08       DEV.MENU   BYTE  8
1625  26F3  50 20 50            BYTE      'P PUT',0
      26F6  55 54 00
1626  26F9  25 54                WORD      WR.DEV
1627  26FB  47 20 47            BYTE      'G GET',0
      26FE  45 54 00
1628  2701  25 45                WORD      RD.DEV
1629  2703  4B 20 4B            BYTE      'K KEYBOARD',0
      2706  45 59 42
      2709  4F 41 52
      270C  44 00
1630  270E  25 6C                WORD      KEYBOARD
1631  2710  4D 20 4D            BYTE      'M MOUSE',0
      2713  4F 55 53
      2716  45 00
1632  2718  25 9A                WORD      MOUSE
1633  271A  55 20 55            BYTE      'U UART',0
      271D  41 52 54
      2720  00
1634  2721  25 CD                WORD      UART
1635  2723  53 20 53            BYTE      'S SET MEMORY',0
      2726  45 54 20
      2729  4D 45 4D
      272C  4F 52 59
      272F  00
1636  2730  26 02                WORD      D.PATTERN
1637  2732  54 20 54            BYTE      'T TEST PICTURE',0
      2735  45 53 54
      2738  20 50 49
      273B  43 54 55
      273E  52 45 00
1638  2741  26 34                WORD      TEST.PICT
1639  2743  49 20 49            BYTE      'I INSPECT',0
      2746  4E 53 50
      2749  45 43 54
      274C  00
1640  274D  20 FA                WORD      INSPECT
1641  274F                       END       DEVICES
1642  274F                       BEGIN     MICROMEM
```

```
1644                    ************************************************************
1645                    *
1646                    *         MICRO MEMORY TEST PROGRAMS
1647                    *
1648                    *         AUTHOR JIRKA HOPPE
1649                    *         VERSION 28/5/80
1650                    *
1651                    ************************************************************
1652                             DEF       MICROMEM
1653                             USE       END.TEST,LOOPCHAIN,CHAIN,WRITELN.A
1654                             USE       U.INSPECT,REREAD
1655                    STARTADR EQU       0000
1656                    ENDADR   EQU       STARTADR+2048Z
1657                    *
1658                    ******************************************
1659                    *
1660                    *  READ LOOP FROM MICRO MEMORY
1661                    *
1662                    *  FORMAT    ADDR 'R' SUBADDR
1663                    *
1664  274F  DE 0F       RD.LOOP   LDX       X.WORD
1665  2751  BD C0 85              JSR       IN.HEX
1666  2754  17                    TBA
1667  2755  BD C1 90   RD.LOOP1  JSR       RUM
1668  2758  BD C0 46              JSR       BREAK
1669  275B  4D                    TST       A
1670  275C  27 F7                 BEQ       RDLOOP1
1671  275E  39                    RTS
1672                    *
1673                    ******************************************
1674                    *
1675                    *  WRITE LOOP TO MICRO MEMORY
1676                    *
1677                    *  FORMAT    ADDR 'W' SUBADDR '>' VALUE
1678                    *
1679  275F  DE 0F       WR.LOOP   LDX       X.WORD
1680  2761  BD C0 85              JSR       IN.HEX
1681  2764  17                    TBA
1682  2765  86 3E                LDAA      '>'
1683  2767  BD C0 51              JSR       W.CH
1684  276A  BD C0 BC              JSR       IN.BYTE
1685  276D  97 2E                 STAA      HELP1
1686  276F  96 2E       WR.LOOP1  LDAA      HELP1
1687  2771  BD C1 7B              JSR       WUM
1688  2774  BD C0 46              JSR       BREAK
1689  2777  4D                    TST       A
1690  2778  27 F5                 BEQ       WRLOOP1
1691  277A  39                    RTS
1692                    *
1693                    ******************************************
1694                    *
1695                    *
1696                    *  PROCEDURE PATTERN;
1697                    *    (* FILL FULL MICRO MEMORY WITH PATTERN
1698                    *       THEN COMPARE IT *)
1699                    *    BEGIN HELP1:=X; SUB.ID := 'PATTERN';  A := HELP1;
```

```
1700                    *       FOR X := STARTADR TO ENDADR DO
1701                    *         FOR B := 0 TO 4 DO  WUM(X, B, A) END;
1702                    *       END;
1703                    *       FOR X := 0 TO 1023 DO
1704                    *         FOR B := 0 TO DO RUM(X, B, A);
1705                    *           IF A<>HELP1 THEN WRITELN(HELP1); S W I(B, A, X) END;
1706                    *         END
1707                    *       FOR
1708                    *     END;
1709                    *
1710  277B  DE 0F       PATTERN1  LDX      X.WORD
1711  277D  DF 2E       PATTERN   STX      HELP1
1712  277F  CE 27 BD              LDX      =PATTEXT
1713  2782  DF 0D                 STX      SUB.ID
1714  2784  96 2F                 LDAA     HELP1+1
1715  2786  CE 00 00              LDX      =STARTADR
1716  2789  5F         FORPAT1   CLR      B
1717  278A  BD C1 7B   FORPAT2   JSR      WUM
1718  278D  5C                   INC      B
1719  278E  C1 05                CMPB     =5
1720  2790  26 F8                BNE      FORPAT2
1721  2792  08                   INX
1722  2793  8C 08 00              CPX      =ENDADR
1723  2796  26 F1                BNE      FORPAT1
1724
1725  2798  CE 00 00              LDX      =STARTADR
1726  279B  5F         FORPAT3   CLR      B
1727  279C  BD C1 90   FORPAT4   JSR      RUM
1728  279F  91 2F                CMPA     HELP1+1
1729  27A1  27 0B                BEQ      ENDIFP
1730  27A3  36                   PSH      A
1731  27A4  96 2F                LDAA     HELP1+1
1732  27A6  BD C0 C9             JSR      OUT.BYTE       ***  EXPECTED VALUE ****
1733  27A9  BD C0 78             JSR      W.CRLF
1734  27AC  32                   PUL      A
1735  27AD  3F                   SWI           ***  B = BYTEADR, A = ACT VALUE, X = ADR
1736  27AE  5C         ENDIFP    INC      B
1737  27AF  C1 05                CMPB     =5
1738  27B1  26 E9                BNE      FORPAT4
1739  27B3  08                   INX
1740  27B4  8C 08 00              CPX      =ENDADR
1741  27B7  26 E2                BNE      FORPAT3
1742  27B9  BD 20 4D             JSR      END.TEST
1743  27BC  39                   RTS
1744                    *
1745  27BD  50 41 54    PATTEXT   BYTE     'PATTERN',0
      27C0  54 45 52
      27C3  4E 00
1746                    *
1747                    ********************************************************
1748                    *
1749                    *  PROCEDURE ADRTEST;
1750                    *    (* WRITE ON LOCATION X := X*HELP1;
1751                    *       READ AND COMPARE IT AGAIN *)
1752                    *    BEGIN HELP1 := X; SUB.ID := 'ADR'; HELP2 :=0;
1753                    *      FOR X := STARTADR TO ENDADR DO
1754                    *        FOR B := 0 TO 4 DO
```

```
1755                    *           A := HELP2 := HELP2+HELP1;  WUM(X, B, A);
1756                    *         END
1757                    *       END;
1758                    *       HELP2 := 0
1759                    *       FOR X := STARTADR TO ENDADR DO
1760                    *         FOR B := 0 TO 4 DO HELP2 := HELP2+HELP1;  RUM(X, B, A);
1761                    *           IF A<>HELP2 THEN WRITELN(HELP2); S W I(B, A, X);
1762                    *         END
1763                    *       END
1764                    *   END;
1765                    *
1766  27C5  DE 0F       ADRTEST1  LDX      X.WORD
1767  27C7  DF 2E       ADRTEST   STX      HELP1
1768  27C9  CE 28 16              LDX      =ADRTEXT
1769  27CC  DF 0D                 STX      SUB.ID
1770  27CE  7F 00 30              CLR      HELP2
1771  27D1  CE 00 00              LDX      =STARTADR
1772  27D4  5F          FORADR1   CLR      B
1773  27D5  96 30       FORADR2   LDAA     HELP2
1774  27D7  9B 2F                 ADDA     HELP1+1
1775  27D9  97 30                 STAA     HELP2
1776  27DB  BD C1 7B              JSR      WUM
1777  27DE  5C                    INC      B
1778  27DF  C1 05                 CMPB     =5
1779  27E1  26 F2                 BNE      FORADR2
1780  27E3  08                    INX
1781  27E4  8C 08 00              CPX      =ENDADR
1782  27E7  26 EB                 BNE      FORADR1
1783
1784  27E9  7F 00 30              CLR      HELP2
1785  27EC  CE 00 00              LDX      =STARTADR
1786  27EF  5F          FORADR3   CLR      B
1787  27F0  96 30       FORADR4   LDAA     HELP2
1788  27F2  9B 2F                 ADDA     HELP1+1
1789  27F4  97 30                 STAA     HELP2
1790  27F6  BD C1 90              JSR      RUM
1791  27F9  91 30                 CMPA     HELP2
1792  27FB  27 0B                 BEQ      ENDIFADR
1793  27FD  36                    PSH      A
1794  27FE  96 30                 LDAA     HELP2
1795  2800  BD C0 C9              JSR      OUT.BYTE       ***  EXPECTED VALUE ****
1796  2803  BD C0 78              JSR      W.CRLF
1797  2806  32                    PUL      A
1798  2807  3F                    SWI           ***  B = BYTEADR, A = ACT VALUE, X = ADR
1799  2808  5C          ENDIFADR  INC      B
1800  2809  C1 05                 CMPB     =5
1801  280B  26 E3                 BNE      FORADR4
1802  280D  08                    INX
1803  280E  8C 08 00              CPX      =ENDADR
1804  2811  26 DC                 BNE      FORADR3
1805  2813  7E 20 4D              JMP      END.TEST
1806                    *
1807  2816  41 44 52    ADRTEXT   BYTE     'ADR',0
      2819  00
1808                    *
1809                    ************************************************************
1810                    *
```

```
1811                    *           INSPECT AND CHANGE MICRO MEMORY
1812  281A             *           BEGIN INSP
1813                    *           DEF   U.INSPECT
1814                    *
1815                    *           FORMAT:
1816                    *           ADDRESS 'I' SUBADR  VALUE  [ ':' NEWVALUE] [ ',' ]
1817                    *           ADDRESS  =  4 DIGIT HEX
1818                    *           SUBADR   =  0..4
1819                    *
1820                    *       PROCEDURE INSPECT
1821                    *          BEGIN
1822                    *             CRNT.ADR := X.WORD;
1823                    *             REPEAT R.CH(A) UNTIL (A>='0') AND (A<='4'); CRNT.SUBADR := A-'0';
1824                    *           LOOP
1825                    *             A := RUM(CRNT.ADR, CRNT.SUBADR);
1826                    *             W.CH(' '); OUT.BYTE(A);
1827                    *             R.CH(A);
1828                    *             IF A=':' THEN
1829                    *               IN.BYTE(A); WUM(CRNT.ADR, CRNT.SUBADR, A); R.CH(A)
1830                    *             END;
1831                    *           IF A#',' THEN EXIT  END;
1832                    *             IF CRNT.SUBADR = 4 THEN INC(CRNT.ADR); CRNT.SUBADR := -1; END;
1833                    *             INC(CRNT.SUBADR);
1834                    *             W.CRLF; OUT.HEX(CRNT.ADR); W.CH('-');
1835                    *             W.CH(CRNT.SUBADR + '0'); W.CH(' ')
1836                    *             END
1837                    *           END
1838                    *         END INSPECT;
1839                    *
1840                    *
1841                    CRNT.ADR    EQU   HELP1
1842                    CRNT.SBADR  EQU   HELP2
1843                    *
1844  281A  DE 0F       U.INSPECT   LDX   X.WORD
1845  281C  DF 2E                   STX   CRNT.ADR
1846  281E  BD C0 85    REP2        JSR   IN.HEX      GET SUBADR
1847  2821  81 04                   CMPA  =4
1848  2823  2E F9                   BGT   REP2
1849  2825  97 30                   STAA  CRNT.SBADR
1850  2827  BD C0 62    LOOP        JSR   W.1.BLANK
1851  282A  D6 30                   LDAB  CRNT.SBADR
1852  282C  BD C1 90                JSR   RUM         READ IT
1853  282F  BD C0 C9                JSR   OUT.BYTE    A CONTAINS ALREADY THE VALUE
1854  2832  BD C0 24                JSR   R.CH        GET DELIMITER
1855  2835  81 3A                   CMPA  =':'
1856  2837  26 09                   BNE   ENDIF1
1857  2839  BD C0 BC                JSR   IN.BYTE     GET NEW VALUE
1858  283C  BD C1 7B                JSR   WUM         WRITE MICRO MEMORY; B, X ARE ALREADY SET
1859  283F  BD C0 24                JSR   R.CH        GET DELIMITER
1860  2842  81 2C       ENDIF1      CMPA  =','
1861  2844  27 01                   BEQ   ENDIF2
1862  2846  39                      RTS               ANY OTHER DELIMITER THEN ',' => EXIT
1863  2847  C1 04       ENDIF2      CMPB  =4           COMP CRNT.SUBADR,#4
1864  2849  26 07                   BNE   ENDIF3
1865  284B  08                      INX               INC(CRNT.ADR)
1866  284C  DF 2E                   STX   CRNT.ADR
1867  284E  86 FF                   LDAA  =0FF
```

```
1868  2850  97 30              STAA  CRNT.SBADR CRNT.SUBADR := -1
1869  2852  7C 00 30  ENDIF3    INC   CRNT.SBADR
1870  2855  BD C0 78            JSR   W.CRLF
1871  2858  BD C2 87            JSR   PRINT.X     WRITE(ADR)
1872  285B  86 2D               LDAA  ='-'
1873  285D  BD C0 51            JSR   W.CH        WRITE('-')
1874  2860  96 30              LDAA  CRNT.SBADR
1875  2862  BD C0 90            JSR   OUT.HEX     WRITE(SUBADR)
1876  2865  20 C0              BRA   LOOP
1877  2867                     END   INSP
1878                  *
1879                  ************************************************************
1880                  *
1881                  * COMPARE MICRO CODE WITH HP-CASSETTE (STANDARD HEXADECIMAL CODED 6800-FORMAT)
1882                  * *************************
1883                  *
1884                  *           PROCEDURE REREAD
1885                  *           VAR EOT: BOOLEAN; (* A-REG, 0=TRUE *)
1886                  *
1887                  *             PROCEDURE SKIPLINE;
1888                  *               BEGIN REPEAT READ(CH) UNTIL CH=CR END;
1889                  *
1890                  *             PROCEDURE READBLOCK;
1891                  *             VAR CH : CHAR; (* A-REG *)
1892                  *                 CHECKSUM,  (* HELP1 *)
1893                  *                 BYTECOUNT : BYTE; (* HELP2 *)
1894                  *                 ADDR : ADDRESS;   (* X-REG *)
1895                  *             BEGIN   1: LINEREAD;
1896                  *               REPEAT
1897                  *                 REACH(CH);
1898                  *               UNTIL CH = 'S';
1899                  *               READ(CH);
1900                  *               EOT := CH = '9';
1901                  *               IF NOT EOT
1902                  *               THEN BEGIN
1903                  *                 INBYTE(BYTECOUNT);
1904                  *                 CHECKSUM := BYTECOUNT;
1905                  *                 INWORD(ADDR);
1906                  *                 CHECKSUM := CHECKSUM + ADDR MOD 40 + ADDR DIV 40;
1907                  *                 REPEAT
1908                  *                   INBYTE(A);
1909                  *                   IF A <> RUM(X,B) THEN
1910                  *                      WRITELN(ROMVALUE); SWI(B, TAPEVALUE,X);
1911                  *                      SKIP THE REST OF THE LINE;  GOTO 1;
1912                  *                   CHECKSUM := CHECKSUM + M[ADDR];
1913                  *                   ADDR := ADDR + 1;
1914                  *                   BYTECOUNT := BYTECOUNT - 1;
1915                  *                 UNTIL BYTECOUNT = 3;
1916                  *                 INBYTE(CH);  (* CHECKSUM *)
1917                  *                 IF CHECKSUM <> COMPL(CH)
1918                  *                 THEN BEGIN
1919                  *                   WSTRING('CHECKSUM ERROR');
1920                  *                 END;
1921                  *               END;
1922                  *             END READBLOCK;
1923                  *
1924                  *           BEGIN (* REREAD *)
```

```
1925                    *             FOR B := 0 TO 4 DO  (* READ 4 FILES *)
1926                    *               (*SKIP HEADER AND INTERFILE GAP*)
1927                    *               LINEREAD;
1928                    *               REPEAT    READ(CH);
1929                    *                 IF CH=CR THEN LINEREAD END;
1930                    *               UNTIL CH = 'S';
1931                    *               REPEAT READ(CH) UNTIL CH = CR;
1932                    *               REPEAT
1933                    *                 READBLOCK;
1934                    *               UNTIL EOT;
1935                    *             END
1936                    *           END REREAD;
1937
1938  2867                          BEGIN TAPE.CMP
1939                                DEF   REREAD
1940
1941  2867  BD E0 03  SKIP.LINE     JSR   READ.CH
1942  286A  81 0D                   CMPA  =CR
1943  286C  26 F9                   BNE   SKIP.LINE
1944  286E  39                      RTS
1945                  *
1946                  READ.BLOCK    EQU   *
1947  286F                          BEGIN READBLOCK
1948  286F  BD C0 FB  REPEAT0       JSR   LINE.READ
1949  2872  BD E0 03  REPEAT1       JSR   READ.CH       SKIP UNTIL 'S'
1950  2875  81 53                  CMPA  ='S'
1951  2877  26 F9      UNTIL1       BNE   REPEAT1
1952  2879  BD E0 03                JSR   READ.CH
1953
1954  287C  80 39                   SUBA  ='9'
1955  287E  36                      PSH   A             SAVE EOF INFORMATION
1956  287F  27 49     IF1           BEQ   ENDIF
1957  2881  BD E0 1A  THEN1         JSR   READ.BYTE     BYTECOUNT
1958  2884  36                      PSH   A
1959  2885  80 03                   SUBA  =3            SUBSTRACT BYTECNT, 2 BYTES ADDRESS
1960  2887  97 30                   STAA  HELP2
1961  2889  32                      PUL   A
1962  288A  CE 00 01                LDX   =X.SAVE
1963  288D  BD E0 27                JSR   READ.WORD     ADDRESS
1964  2890  AB 00                   ADDA  0,X
1965  2892  AB 01                   ADDA  1,X
1966  2894  97 2E                   STAA  HELP1         XSUM:= BYTECOUNT + ADDR(HI) + ADDR(LO)
1967  2896  DE 01                   LDX   X.SAVE
1968  2898  BD E0 1A  REPEAT2       JSR   READ.BYTE     DATA
1969  289B  97 32                   STAA  HELP3         HELP3 := TAPE DATA
1970  289D  BD C1 90                JSR   RUM           READ MICRO MEM; X,B ARE ALREADY SET
1971  28A0  91 32                   CMPA  HELP3
1972  28A2  27 0C                   BEQ   U.ROM.OK
1973  28A4  36                      PSH   A             SKIP THE REST OF THE LINE
1974  28A5  8D C0                   BSR   SKIP.LINE
1975  28A7  32                      PUL   A
1976  28A8  BD 20 87                JSR   WRITELN.A     ROM VALUE**********************
1977  28AB  96 32                   LDAA  HELP3
1978  28AD  3F                      SWI                 B=BYTE ADR; A=TAPE VALUE; X=ADDRESS  ****
1979  28AE  20 BF                   BRA   REPEAT0       TRY NEW LINE
1980
1981  28B0  96 32     U.ROM.OK      LDAA  HELP3
```

```
1982  28B2  9B 2E               ADDA  HELP1
1983  28B4  97 2E               STAA  HELP1       XSUM := XSUM + A
1984  28B6  08                  INX
1985  28B7  7A 00 30            DEC   HELP2
1986  28BA  26 DC     UNTIL2    BNE   REPEAT2
1987  28BC  BD E0 1A            JSR   READ.BYTE   CHECKSUM
1988  28BF  43                  COM   A
1989  28C0  91 2E               CMPA  HELP1
1990  28C2  27 06     IF2       BEQ   ENDIF
1991  28C4  CE 28 F9  THEN2     LDX   =MESS1
1992  28C7  BD C0 6A            JSR   W.STRING
1993                  ENDIF     EQU   *
1994  28CA  32                  PUL   A           RESAVE EOF INFORMATION
1995  28CB  39                  RTS
1996  28CC                      END   READBLOCK
1997
1998
1999  28CC  CE 29 02  REREAD    LDX   =CMP.TXT
2000  28CF  DF 0D               STX   SUB.ID
2001                  ****    FOLLOWING VALUE COULD BE PATCHED TO READ THE N'TH FILE ****
2002  28D1  C6 00               LDAB  =0          WE START READING THE FIRST FILE
2003  28D3  BD C0 FB  SKIP.HDR0 JSR   LINEREAD    SKIP THE HEADER LINE
2004  28D6  BD E0 03  SKIP.HDR1 JSR   READ.CH
2005  28D9  81 0D               CMPA  =CR
2006  28DB  26 03               BNE   SKIP.HDR2
2007  28DD  BD C0 FB            JSR   LINEREAD
2008  28E0  81 53     SKIP.HDR2 CMPA  ='S'
2009  28E2  26 F2               BNE   SKIP.HDR1
2010
2011  28E4  BD E0 03  SKIP.HDR3 JSR   READ.CH     SKIP THE FIRST LINE
2012  28E7  81 0D               CMPA  =CR
2013  28E9  26 F9               BNE   SKIP.HDR3
2014
2015  28EB  8D 82     REPEAT    BSR   READ.BLOCK
2016  28ED  4D                  TST   A           END OF TAPE
2017  28EE  26 FB     UNTIL     BNE   REPEAT
2018
2019  28F0  BD 28 67            JSR   SKIP.LINE
2020
2021  28F3  5C                  INC   B           INCREMENT THE FILE NUMBER
2022  28F4  C1 04               CMPB  =4
2023  28F6  2F DB               BLE   SKIP.HDR0
2024  28F8  39                  RTS
2025                  *
2026  28F9  58 53 55  MESS1     BYTE  'XSUM ERR',0
      28FC  4D 20 45
      28FF  52 52 00
2027  2902  55 52 4F  CMPTXT    BYTE  'UROM CMP',0
      2905  4D 20 43
      2908  4D 50 00
2028  290B                      END   TAPE.CMP
2029                  *
2030                  ************************************************************
2031                  *
2032                  *  LOOP ALL TEST USING CHAIN
2033  290B  CE 29 8A  LOOPTEST  LDX       =UCHAIN
2034  290E  BD 20 15            JSR       CHAIN
```

```
2035  2911  20 F8              BRA       LOOPTEST
2036                    *
2037                    ***********************************************************
2038                    *
2039                    *         ENTRY POINT TO THE MICRO MEMORY MONITOR
2040                    *
2041                    ***********************************************************
2042                    *
2043  2913  CE 29 83    MICROMEM  LDX       =MICROTEXT
2044  2916  DF 0B                STX       MAIN.ID
2045  2918  CE 29 1E              LDX       =MICRO.MENU
2046  291B  7E C3 91              JMP       COMMANDS
2047                    *
2048  291E  07          MICROMENU  BYTE      7
2049  291F  52 20 52              BYTE      'R READLOOP',0
      2922  45 41 44
      2925  4C 4F 4F
      2928  50 00
2050  292A  27 4F                 WORD      RD.LOOP
2051  292C  57 20 57              BYTE      'W WRITELOOP',0
      292F  52 49 54
      2932  45 4C 4F
      2935  4F 50 00
2052  2938  27 5F                 WORD      WR.LOOP
2053  293A  50 20 50              BYTE      'P PATTERN',0
      293D  41 54 54
      2940  45 52 4E
      2943  00
2054  2944  27 7B                 WORD      PATTERN1
2055  2946  56 20 41              BYTE      'V ADRESSING',0
      2949  44 52 45
      294C  53 53 49
      294F  4E 47 00
2056  2952  27 C5                 WORD      ADRTEST1
2057  2954  49 20 49              BYTE      'I INSPECT',0
      2957  4E 53 50
      295A  45 43 54
      295D  00
2058  295E  28 1A                 WORD      U.INSPECT
2059  2960  55 20 55              BYTE      'U UROM COMPARE',0
      2963  52 4F 4D
      2966  20 43 4F
      2969  4D 50 41
      296C  52 45 00
2060  296F  28 CC                 WORD      REREAD
2061  2971  58 20 45              BYTE      'X EXECUTE CHAIN',0
      2974  58 45 43
      2977  55 54 45
      297A  20 43 48
      297D  41 49 4E
      2980  00
2062  2981  29 0B                 WORD      LOOPTEST
2063                    *
2064  2983  4D 49 43    MICROTEXT BYTE      'MICROM',0
      2986  52 4F 4D
      2989  00
2065                    *
```



```
2066  298A  04          UCHAIN    BYTE      U.CH.LEN
2067  298B  27 7D                 WORD      PATTERN
2068  298D  00 00                WORD      0         PATTERN(0)
2069  298F  27 C7                WORD      ADRTEST
2070  2991  00 01                WORD      1         ADRTEST(1)
2071  2993  27 7D                WORD      PATTERN
2072  2995  FF FF                WORD      0FFFF     PATTERN(FF)
```

```
2073  2997  27 C7              WORD      ADRTEST
2074  2999  00 31              WORD      31        ADRTEST(31)
2075                 U.CH.LEN   EQU       *-1-UCHAIN/4
2076                 *
2077  299B                     END       MICROMEM
2078  299B                     END       HWTEST
```