HWTEST

```
 2  0000             BEGIN HWTEST
 3
 4                  ************************************************************
 5                  *
 6                  *    H A R D W A R E T E S T 2            24-SEP-80
 7                  *
 8                  *    INCLUDES:
 9                  *
10                  *    MAIN HARDWARE SUBMONITOR
11                  *    SELFTEST MONITOR
12                  *    MICROCONTROLLER SUBMONITOR
13                  *    ALU-TEST SUBMONITOR
14                  *    MAIN MEMORY SUBMONITOR
15                  *
16                  ************************************************************
```

```
19                  ********************************************************************************
20                  *                                                                              *
21                  *      DIAGNOSTIC PANEL M6802-SOFTWARE                    23.05.80       *
22                  *                                                                              *
23                  *                J.HOPPE                                                       *
24                  *                I.NOACK                                                       *
25                  *                R.OHRAN                                                       *
26                  *                W.WINIGER                                                     *
27                  *                                                                              *
28                  ********************************************************************************
29
30
31                  BIN         EQU   0C09C
32                  ***         HEXADECIMAL (ASCII) TO BINARY CONVERSION
33                  *
34                  *           ARGUMENT:    A = CHARACTER TO BE CONVERTED
35                  *           RESULTS:     CARRY = RESET, IF A CONTAINED A VALID HEX DIGIT
36                  *                             SET OTHERWISE
37                  *                        A = BINARY VALUE OF THE PARAMETER A IF CARRY RESET
38                  *                            UNDEFINED OTHERWISE
39
40                  HEX         EQU   0C0B3
41                  ***         BINARY TO HEXADECIMAL (ASCII) CONVERSION
42                  *
43                  *           ARGUMENT:    A = FOUR BIT VALUE TO BE CONVERTED
44                  *           RESULTS:     A = ASCII-CHARACTER REPRESENTING AS HEX DIGIT
45                  *                            THE FOUR BIT NUMBER CONTAINED IN PAR. A
46
47                  R.CH        EQU   0C024
48                  ****        READ A CHARACTER FROM THE HP AND ECHO IT
49                  *
50                  *           ARGUMENT:    -
51                  *           RESULT:      A = CHARACTER READ FROM THE UART
52                  *           SIDE EFFECT: R.CH POLLS UNTIL A CHARACTER IS SENT TO IT
53                  *                        A = ESC -> CONTROL IS TRANSFERED TO THE SUBMONITOR
54                  *                        A = ↑C -> CONTROL IS TRANSFERED TO THE MONITOR
55                  *                        A = QUESTIONMARK -> MENU IS DISPLAYED
56
57                  READ.CH     EQU   0E003  SAME AS R.CH BUT WITHOUT ECHO AND WITHOUT INTER-
58                  *******                  PRETATION OF A
59
60                  BREAK       EQU   0C046
61                  *****       READ A CHARACTER FROM THE HP (WITHOUT POLLING)
62                  *           RESULT:      A = CHARACTER READ FROM THE UART
63                  *                           = 0 IF NO KEY WAS PRESSED
64                  *           SIDE EFFECT: SEE R.CH
65
66                  W.CH        EQU   0C051
67                  ****        WRITE A CHARACTER TO THE HP
68                  *
69                  *           ARGUMENT:    A = CHARACTER TO BE WRITTEN
70                  *           RESULT:      -
71                  *           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
                *********
```

```
128                 OUT.BYTE    EQU   0C0C9
129                 ********    OUTPUT AN EIGHT BIT NUMBER REPRESENTED BY TWO HEXADECIMAL DIGITS
130                 *
131                 *           ARGUMENT:     A = THE BYTE TO BE WRITTEN TO THE HP
132                 *           RESULT:       -
133
134                 IN.WORD     EQU   0C0D8
135                 *******     INPUT FOUR HEX DIGITS FROM THE HP INTERPRETING THEM AS A SIXTEEN
136                 *           BIT NUMBER (WITH ECHO)
137                 *
138                 *           ARGUMENT:     X = ADDRESS OF THE MEMORY LOCATION WHERE THE WORD
139                 *                             READ HAS TO BE STORED
140                 *           RESULTS:      M[X] AND M[X+1] CONTAIN THE DATA READ
141                 *           SIDE EFFECT:  IN.WORD READS CHARACTERS FROM THE HP UNTIL FOUR
142                 *                         VALID HEX DIGITS ARE ENCOUNTERED
143
144                 READ.WORD   EQU   0E027   IN.WORD WITHOUT ECHO
145                 *********
146
147                 OUT.WORD    EQU   0C0E3
148                 ********    OUTPUT A 16-BIT NUMBER REPRESENTED BY FOUR HEX DIGITS
149                 *
150                 *           ARGUMENTS:    X = ADDRESS OF THE MEMORY LOCATION CONTAINING THE
151                 *                             WORD TO BE WRITTEN
152                 *                         M[X] = HIGH BYTE,
153                 *                         M[X+1] = LOW BYTE OF THE WORD
154                 *           RESULT:       -
155
156                 FAST.READ   EQU   0C0EE
157                 *********   INITALIZE THE HP READING ITS CASSETTE AND TRANSFERING ONE FILE
158
159                 LINE.READ   EQU   0C0FB
160                 *********   INITIALIZE THE HP READING ITS CASSETTE AND SENDING ONE LINE
161
162
```

```
164                MENU        EQU   0C336
165                ****        WRITE THE MENU POINTED TO BY ACT.MENU ON THE SCREEN
166                *
167                *           ARGUMENT:    ACT.MENU = ADDRESS OF THE MENU (ASCII STRINGS
168                *                                   AND ENTRYPTS)
169
170                GET.KEY     EQU   0C352
171                *******     PROMPT WITH CR/LF AND A '*', CALL GETWORD AND REPEAT THEN
172                *           ACCEPTING A 'KEY' UNTIL (SKIPPING BLANKS) THIS IS THE FIRST CHA-
173                *           RACTER OF ONE OF THE LINES OF THE MENU. CONVERT ALL CHARACTERS
174                *           TO CAPS. JUMP THEN TO THE ROUTINE IDENTIFIED BY THIS 'KEY'.
175
176                *
177                *           ARGUMENT:    ACT.SUB.MON = ADDRESS OF THE MENU
178                *           RESULTS:     X.WORD (AT LOCATION 0F) = THE ADDRESS ENTERED BEFOR
179                *                                                  THE CHARACTER
180                *                        CARRY = FALSE IF NO ADDRESS AT ALL WAS ENTERED
181                *                              = TRUE IF X.WORD CONTAINES AN ADDRESS
182                *           SIDE EFFECT: X IS DESTROID BY GET.KEY
183
184
185                GET.WORD    EQU   0C2FE
186                ********    READ HEX DIGITS FROM THE HP SKIPPING BLANKS AT THE BEGINNIG INTER-
187                *           PRETING THEM AS A NUMBER UNTIL A NONNUMERICAL CHARACTER IS
188                *           ENCOUNTERED
189                *           RESULTS:     X.WORD (AT ADDRESS 0F) = BINARY VALUE OF THE LAST
190                *                                                  4 HEX DIGITS ENTERED
191                *                        A = THE NONHEX CHARACTER WHICH TERMINATED THE INPUT
192                *                        CARR (AT LOCATION 11) = 0 IF THE FIRST CHARACTER
193                *                                                ENTERED WAS NONHEXADECIMAL,
194                *                                              = FF IF A NUMBER WAS ENTERED
195                *                                                FIRST
196
197                COMMANDS    EQU   0C391  STORE THE CONTENT OF THE X-REGISTER INTO ACT.SUB.MON,
198                ********                 WRITE THE STRING POINTED TO BY MAIN.ID ON THE SCREEN
199                *                        AND REPEAT THEN CALLING GET.KEY.
200                *           ARGUMENT:    X = POINTER TO THE ACTUAL MENU.
201
202                BOOT        EQU   0E2C5  BOOTSTRAP THE PERSONAL COMPUTER
203                ****
204
205                SET.U.PC    EQU   0E2D5  SET THE MICRO PROGRAM COUNTER
206                ********
207
208                LOAD.U.RAM  EQU   0E3B3  LOAD THE MICRO-RAM FROM THE EPROM'S
209                **********
```

```
211                 PUSH.X      EQU   0C259
212                 ******      STORES X ON STACK
213
214                 PULL.X      EQU   0C270
215                 ******      RETRIEVES X FROM STACK
216
217                 PRINT.X     EQU   0C287
218                 *******     PRINTS THE CONTENT OF THE X-REGISTER
219
220                 WUM         EQU   0C17B
221                 ***         WRITE MICRO MEMORY ROUTINE
222                 *           UMEM(X,B):=A
223
224                 RUM         EQU   0C190
225                 ***         MICRO MEMORY READ ROUTINE
226                 *           A:=UMEM(X,B)
227
228                 STUPC       EQU   0C1C2
229                 *****       STORE MICRO PROGRAM COUNTER
230                 *           PREPARES MICROCONTROLLER FOR EXECUTION AT ADDRESS IN X
231                 *           UPC:=X
232
233                 EXX         EQU   0C117
234                 ***         EXECUTES MICRO INSTRUCTION POINTED TO BY X
235                 *           X POINTS TO MOST SIGNIFICANT BYTE
236                 *           MOST SIGNIFICANT BYTE HAS LOWEST ADDRESS
237                 *           X IS INCREMENTED BY 5
238
239                 RCPU        EQU   0C1F0
240                 ****        READ CPU DATA BUS
241                 *           B GETS TOP HALF
242                 *           A GETS BOTTOM HALF
243
244                 WCPU        EQU   0C1F7
245                 ****        WRITE CPU DATA REGISTER
246                 *           CPU BUS REGISTER := B*256 + A
247                 *           DATA IS GATED TO BUS WHEN SOURCE 13 IS ADDRESSED
248
249                 EPMIR       EQU   0C15F
250                 *****       ENABLE PERSONAL COMPUTER MIR REGISTER
251
252                 EDMIR       EQU   0C158
253                 *****       DIAGNOSTIC PANEL MIR REGISTER IS ENABLED
254
255                 LDMIR       EQU   0C11D
256                 *****       DIAGNOSTIC PANEL MIR REGISTER IS LOADED WITH
257                 *           DATA POINTED TO BY X
258                 *           MOST SIGNIFICANT BYTE HAS THE LOWEST ADDRESS
259                 *           X IS RETURNED INCREMENTED BY 5
260                 *           RETURNS WITH DIAGNOSTIC MIR REGISTER ENABLED
261                 *           NO CPU CLOCK PULSE IS GIVEN
262                 *           USEFUL TO GATE DATE NON-DESTRUCTIVELY TO CPU BUS
263
264                 MCLK        EQU   0C1AC
265                 ****        SEND ONE MICRO CLOCK PULSE (DISABLE CPU CLOCK)
```

```
267                 RMM         EQU   0C1FE
268                 ***         READ MAIN MEMORY
269                 *           B:= MAIN(X)DIV 256
270                 *           A:= MAIN(X)MOD 256
271
272                 RMMS        EQU   0C211
273                 ****        SHORT (QUICK) VERSION OF RMM
274                 *           MAY ONLY BE USED AFTER A LONG VERSION CALL HAS BEEN MADE AND
275                 *           NO OTHER DMIR OPERATIONS HAVE OCCURED
276
277                 WMM         EQU   0C228
278                 ***         WRITE MAIN MEMORY
279                 *           MAIN(X):=256*B + A
280
281                 WMMS        EQU   0C23D
282                 ****        SHORT VERSION OF WMM. SEE RMMS COMMENTS
283
284                 LATCH       EQU   0C151
285                 *****       MICRO MEMORY ADDRESS REGISTER IS LATCHED
286
287                 UNLCH       EQU   0C14A
288                 *****       MICRO MEMORY ADDRESS REGISTER IS UNLATCHED
289
290                 D2911       EQU   0C13E
291                 *****       MICRO MEMORY ADDRESS OUTPUTS FROM PERSONAL COMPUTER
292                 *           ARE DISABLED AND ADDRESS REGISTER FROM DIAGNOSTIC
293                 *           PANEL IS ENABLED
294
295                 E2911       EQU   0C174
296                 *****       REVERSE OF D2911
297
298                 STOP        EQU   0C16D
299                 ****        DISABLE CPU CLOCK AND ENABLE SINGLE STEP MODE
300
301                 START       EQU   0C166
302                 *****       ENABLE CPU CLOCK
303
304                 RESET       EQU   0C290
305                 *****       RESET COMPUTER; INITIATE STAT0 VECTOR
306
307                 READ.REG    EQU   0C2DD
308                 ********    READ A 2901 REGISTER
309                 *           ARGUMENT:  B = NR OF THE REGISTER  ( 0 .. 0F OR Q.REG)
310                 *           RESULT:    X = THE VALUE
311
312                 READ.R      EQU   0C2A5
313                 ******      MIR USED BY THE READ.REG
314
315                 LOAD.REG    EQU   0C2AA
316                 *******     LOAD A 2901 REGISTER
317                 *           ARGUMENTS: B = NR OF THE REGISTER  ( 0..0F OF Q.REG)
318                 *                      X = VALUE TO BE LOADED
319
320                 WRITE.R     EQU   0C2A0
321                 *******     MIR USED BY LOAD.REG
322
```

```
                PRINT.BUS   EQU   0E1AA
                *********         PRINTS AN ERRORMESSAGE AND THE REGISTERS WITH THE VALUE OF
                *                 THE CPU-BUS INSTEAD OF THE REGISTERS B,A
                *           ASCII CHARACTERS
                *
                LF          EQU   0A
                CR          EQU   0D
                DC1         EQU   11
                ESC         EQU   1B
                RS          EQU   1E
                EXCLAM      EQU   21
                AMPERSAND   EQU   26
                LOWER.E     EQU   65
                LOWER.P     EQU   70
                RUBOUT      EQU   7F
                BLANK       EQU   20
                LOWER.U     EQU   75
                QUESTION.M  EQU   3F
                PROMPT      EQU   2A
                COMMA       EQU   2C
                BACK.SLASH  EQU   5C
                *           I/O ADDRESSES
                *
                HP.STATUS   EQU   4000
                HP.DATA     EQU   4001
                MIR4        EQU   3C08
                MIR3        EQU   3C09
                MIR2        EQU   3C0A
                MIR1        EQU   3C0B
                MIR0        EQU   3C0C
                CPU1        EQU   3C0E
                CPU0        EQU   3C0F
                CON0        EQU   3C13
                CON1        EQU   3C12
                MEMPU       EQU   3C14
                SSP         EQU   3C15
                UAR1        EQU   3C10
                UAR0        EQU   3C11
                Q.REG       EQU   10
                STACK       EQU   07F
                RET         EQU   2000 (JMP   0)          'RETURN' - ADDRESS FOR PUSHX/PULLX
                *           EXTERNAL ENTRY POINTS
                *
                MONITOR     EQU   0E000
```

```
379                 *           VARIABLES
380
381                 ACT.MENU    EQU    9 (WORD  0)          ADDRESS OF THE ACTUAL MENU
382                 MAIN.ID     EQU   0B (WORD  0)
383                 SUB.ID      EQU   0D (WORD  0)
384                 C           EQU    0 (BYTE  0)          DUMMY REGISTERS,
385                 X.SAVE      EQU    1 (WORD  0)          FOR STRICTLY LOCAL USE ONLY
386                 ADDR        EQU   18 (WORD  0)          USED BY DUMP AND DUMP.LINE
387                 X.WORD      EQU   0F (WORD  0)          SAVE WORD FOR X, USED BY GET.WORD,
388                 *                                       AND MENU
389                 CARR        EQU   11 (BYTE  0)          'CARRY'-BYTE (RESULT OF GET.WORD)
390                 CHECK.SUM   EQU   17 (BYTE  0)          USED BY READBLOCK (LOAD)
391                 STAT0       EQU    3 (BYTE  0)
392
393                 MTMP4       EQU    4 (BYTE  0)
394                 MTMP3       EQU    5 (BYTE  0)
395                 MTMP2       EQU    6 (BYTE  0)
396                 MTMP1       EQU    7 (BYTE  0)
397                 MTMP0       EQU    8 (BYTE  0)
398
399                 TYPING      EQU   16 (BYTE  1)          CONTROLS TYPING ERRORMESSAGES
400                 PSEUDO.IRQ  EQU   21
401                 XSUM        EQU   25 (BYTE  0)          CHECKSUM (MICRO)
402                 CRNT.ADR    EQU   26 (WORD  0)          CURRENT ADR (MICRO, MAIN MEMORY)
403
404                 USER.DATA   SET   2C                    NEXT FREE LOCATION FOR DATA IN THE
405                 *                                       ZERO-PAGE
406                 ************************************************************
```

```
408                 *                                                     *
409                 *                                                     *
410                 *         HARDWARE TEST SUBMONITOR                    *
411                 *                                                     *
412                 *         AUTHOR JIRKA HOPPE                          *
413                 *                                                     *
414                 *         INSTITUT OF INFORMATICS                     *
415                 *         ETH ZURICH                                  *
416                 *                                                     *
417                 *         VERSION   24.9.80                           *
418                 *                                                     *
419                 *                                                     *
420                 *                                                     *
421                 *                                                     *
422                 *******************************************************
423                           USE       HWTEST
424                 CODE      SET       *
425  0000                     ORG       USER.DATA
426                 *         DEFINITION OF VARIABLES
427  002C  00 00    NEXT.IN.CH WORD     0         POINTS TO THE NEXT ENTRY IN THE CHAIN
428  002E  00 00    HELP1     WORD      0         IS USED LOCALLY BY TEST PROCEDURES
429  0030  00 00    HELP2     WORD      0             ''    NO GLOBAL PROCEDURE MAKES USE OF IT
430  0032  00 00    HELP3     WORD      0             ''
431  0034  00 00    HELP4     WORD      0             ''
432
433                 USER.DATA SET       *
434
435                 *         ENTRY POINTS USED BY THE MAIN MONITOR
436  0036                     ORG       2003
437  2003  7E E0 00           JMP       MONITOR      RETURN TO MAIN MONITOR
438  2006  7E E0 00           JMP       MONITOR      RETURN TO MAIN MONITOR
439  2009  7E E0 00           JMP       MONITOR
440  200C  7E 21 92           JMP       HWTEST
441  200F  7E E0 00           JMP       MONITOR   DUMMY FOR MICRO MEMORY
442  2012  7E 20 12 JUMP      JMP       JUMP      USED BY CHAIN FOR JUMP TO THE ROUTINE ENTRY
443                 *
444                 *******************************************************
```

```
2015                    BEGIN     HARDWARE TEST
                        DEF       HWTEST,ENDTEST,CHAIN,WRITELN.A,SET.MAIN.ID
                        DEF       X.L.ROTATE,X.R.ROTATE
                        DEF       NEXTCHAIN
                        USE       SELFTEST,STOR.PORT,ALU,MICROCTRL
                        USE       UC.CHAIN,MC.CH.LEN,ALU.CHAIN,ALU.CH0.LEN
                ************************************************************
                *
                *         MAIN HARDWARE TEST MONITOR
                *
                ************************************************************
                *
                *    THIS PROCEDURE ALLOWS CHAINING OF A NUMBER OF HARDWARE TESTS
                *
                *    A CHAIN HAS FOLLOWING FORMAT:
                *      #ITEMS      ! ENTRY.ADR    X.VALUE "
                *
                *    #ITEMS    - HOW MANY PAIRS (ENTRYADR,VALUE) ARE CONTAINED IN THE CHAIN
                *    ENTRY.ADR - ENTRY POINT OF THE SUBROUTINE
                *    X.VALUE   - INITIAL VALUE FOR THE X REGISTER
                *
                *  PROCEDURE CHAIN(X : ↑CHAINS);
                *    VAR COUNT, SAVEX : SAVES;
                *    BEGIN  COUNT := X↑; INC(X);
                *      REPEAT  JUMP := X↑; INC(X,2); SAVEX := X;
                *        X := X↑; CALL(JUMP);
                *        X := SAVEX; INX(X,2);
                *        DEC(COUNT);
                *      UNTIL COUNT = 0
                *    END;
                *
2015  A6 00     CHAIN     LDAA      0,X
2017  36                 PSH       A         SAVE COUNT
2018  08                 INX
2019  A6 00     CHLOOP    LDAA      0,X       GET ENTRY POINT
201B  B7 20 13           STAA      JUMP+1
201E  A6 01              LDAA      1,X
2020  B7 20 14           STAA      JUMP+2
2023  08                 INX
2024  08                 INX
2025  DF 2C              STX       NEXT.IN.CH
2027  EE 00              LDX       0,X       GET INITIAL X VALUE
2029  BD 20 12           JSR       JUMP      EXECUTE ROUTINE
202C  BD C0 46           JSR       BREAK     GET OUT BY 'ESC'
202F  DE 2C              LDX       NEXT.IN.CH
2031  08                 INX
2032  08                 INX
2033  32                 PUL       A         GET COUNTER
2034  4A                 DEC       A
2035  36                 PSH       A         SAVE IT AGAIN
2036  26 E1              BNE       CHLOOP
2038  32                 PUL       A         JUST CLEAN UP STACK
2039  39                 RTS
                *
                ************************************************************
                *
```

```
502                     *  PROCEDURE NEXTCHAIN( X : ↑CHAIN);
503                     *    (* THIS PROCEDURE SWITCHES TO THE NEXT CHAIN POINTED BY X *)
504                     *
505  203A  DF 2C        NEXTCHAIN STX       NEXT.IN.CH
506  203C  39                     RTS
507                     *
508                     ************************************************************
509                     *
510                     *  PROCEDURE LOOP.CHAIN( X : ↑CHAINS);
511                     *    BEGIN LOOP CHAIN(X.WORD) END END;
512                     *
513  203D  CE 22 62     LOOPCHAIN LDX       =GLOB.CHAIN
514  2040  8D D3                BSR       CHAIN
515  2042  20 F9                BRA       LOOPCHAIN
516                     *
517                     ************************************************************
518                     *
519                     *  PROCEDURE LOOPROUTINE( X );
520                     *    BEGIN LOOP CALL(X↑) END END;
521                     *
522  2044  DE 0F        LOOP.RTNE LDX       X.WORD
523  2046  AD 00                JSR       0,X
524  2048  20 FA                BRA       LOOP.RTNE
525                     *
526                     ************************************************************
527                     *
528                     *  PROCEDURE SET.MAIN.ID( X : ↑CHAINS);
529                     *    BEGIN MAIN.ID := X END
530  204A  DF 0B        SETMAINID STX       MAIN.ID
531  204C  39                     RTS
532                     *
533                     ************************************************************
534                     *
535                     *  PROCEDURE END.TEST;
536                     *    BEGIN WRITE('END OF ',SUB.ID) END;
537                     *
538  204D  36           END.TEST  PSH       A
539  204E  96 16                LDAA      TYPING    TYPE CTRL
540  2050  27 12                BEQ       END.END   NO TYPING AT ALL
541  2052  81 02                CMPA      =2
542  2054  27 0E                BEQ       END.END   ONLY ERROR MESSAGES
543  2056  CE 20 69             LDX       =END.TEXT
544  2059  BD C0 6A             JSR       W.STRING
545  205C  DE 0D                LDX       SUB.ID
546  205E  BD C0 6A             JSR       W.STRING
547  2061  BD C0 78             JSR       W.CRLF
548  2064  32           END.END   PUL       A
549  2065  BD C2 90                JSR       RESET
550  2068  39                     RTS
551                     *
552  2069  45 4E 44     END.TEXT  BYTE      'END OF ',0
     206C  20 4F 46
     206F  20 00
553                     *
554                     ************************************************************
555                     *
556                     *  PROCEDURE PRINTAB( A, B, X)
```

```
557                     *    BEGIN WRITE(X↑, A, B) END;
558                     *
559  2071  36           PRINT.AB  PSH       A
560  2072  BD C0 6A              JSR       W.STRING
561  2075  BD C0 62              JSR       W.1.BLANK
562  2078  BD C0 C9              JSR       OUT.BYTE
563  207B  BD C0 62              JSR       W.1.BLANK
564  207E  17                    TBA
565  207F  BD C0 C9              JSR       OUT.BYTE
566  2082  BD C0 78              JSR       W.CRLF
567  2085  32                    PUL       A
568  2086  39                    RTS
569                     *
570                     ************************************************************
571                     *  PROCEDURE WRITELN(A);
572                     *    BEGIN WRITE(A); CRLF END;
573                     *
574  2087  BD C0 C9     WRITELN.A JSR    OUTBYTE
575  208A  7E C0 78              JMP       W.CRLF    MAKES RTS
576                     *
577                     ************************************************************
578                     *
579                     *  ROTATE X REGISTER ONE BIT TO LEFT
580                     *
581  208D  36           X.LROTATE PSH       A
582  208E  37                    PSH       B
583  208F  DF 01                 STX       X.SAVE
584  2091  96 01                 LDAA      X.SAVE    HIGH BYTE
585  2093  48                    ASL       A         CARRY <- MSB; LSB <- 0
586  2094  D6 02                 LDAB      XSAVE+1   LOWER BYTE; DOESN'T CHANGE CARRY
587  2096  2A 01                 BPL       X.L.1
588  2098  4C                    INC       A         SET LSB OF 'A' IF MSB OF 'B' WAS SET
589                     *                           CARRY IS NOT CHANGED
590  2099  59           X.L.1     ROL       B         CARRY FROM 'ASL A'
591  209A  97 01                 STAA      X.SAVE
592  209C  D7 02                 STAB      XSAVE+1
593  209E  DE 01                 LDX       X.SAVE
594  20A0  33                    PUL       B
595  20A1  32                    PUL       A
596  20A2  39                    RTS
597                     *
598                     ************************************************************
599                     *
600                     *  ROTATE X REGISTER ONE BIT TO RIGHT
601                     *
602  20A3  36           X.RROTATE PSH       A
603  20A4  37                    PSH       B
604  20A5  DF 01                 STX       X.SAVE
605  20A7  96 02                 LDAA      X.SAVE+1  LOWER
606  20A9  D6 01                 LDAB      X.SAVE    HIGHER
607  20AB  44                    LSR       A
608  20AC  56                    ROR       B
609  20AD  24 02                 BCC       X.R.1
610  20AF  8A 80                 ORAA      =80       SET MSB
611  20B1  97 02     X.R.1       STAA      X.SAVE+1
612  20B3  D7 01                 STAB      X.SAVE
613  20B5  DE 01                 LDX       X.SAVE
```

```
614  20B7  33                PUL       B
615  20B8  32                PUL       A
616  20B9  39                RTS
617                  *
618                  ************************************************************
619                  *
620                  *  PROCEDURE TESTBUS;
621                  *    (* TEST FOR SHORT CONNECTION ON THE BUS *)
622                  *    BEGIN SUB.ID := 'CPU BUS';
623                  *      MIR := (1 - OR ZA 0 0 0 - - - - 0 - PANEL);
624                  *      X := 1;
625                  *      FOR B := 2 TO 0 BY -1
626                  *        FOR A := 16 TO 0 BY -1 DO
627                  *          BUS := X;
628                  *          IF BUS <> X THEN PRINTBUS(X) END;
629                  *          X.L ROTATE;
630                  *        END;
631                  *        X := 0FFFEH;
632                  *      END;
633                  *    END;
634                  *
635  20BA  CE 20 F2  TESTBUS  LDX       =BUSTEXT
636  20BD  DF 0D             STX       SUB.ID
637  20BF  CE C2 A5          LDX       =READ.R      JUST DEFINE ANY MIR
638  20C2  BD C1 1D          JSR       LDMIR
639  20C5  86 FD             LDAA      =0FD      SOURCE := PANEL
640  20C7  B7 3C 0C          STAA      MIR0
641  20CA  BD C1 58          JSR       EDMIR
642  20CD  CE 00 01          LDX       =1        ROTATING 1
643  20D0  C6 02             LDAB      =2
644  20D2  86 10     FORBUS1  LDAA     =16Z
645  20D4  FF 3C 0E  FORBUS2  STX      CPU1
646  20D7  BC 3C 0E          CPX       CPU1
647  20DA  27 03             BEQ       TSTBS1
648  20DC  BD E1 AA          JSR       PRINTBUS
649  20DF  BD 20 8D  TSTBS1   JSR       X.L.ROTATE
650  20E2  4A                DEC       A
651  20E3  26 EF             BNE       FORBUS2
652  20E5  CE FF FE          LDX       =0FFFE    ROTATING 0
653  20E8  5A                DEC       B
654  20E9  26 E7             BNE       FORBUS1
655  20EB  BD C1 5F          JSR       EPMIR
656  20EE  BD 20 4D          JSR       END.TEST
657  20F1  39                RTS
658                  *
659  20F2  43 50 55  BUSTEXT  BYTE      'CPU BUS',0
     20F5  20 42 55
     20F8  53 00
660                  *
661                  ************************************************
662                  *
663                  *  PROCEDURE CLOCK;
664                  *    (* TEST VARIOUS CONTROL OF CPU CLOCK *)
665                  *    BEGIN RESET;  R.CH(A);
666                  *      CASE A OF
667                  *      '0':                %
668                  *      '1': MIR := READ.REG;
```

```
669                     *                 REPEAT SINGLE STEP;
670                     *                 UNTIL BREAK
671                     *      ELSE MIR := READ.REG; START RUNNIG CLOCK; R.CH
672                     *      END;
673                     *      RESET;
674                     *    END CLOCK
675                     *
676  20FA  BD C2 90  CLOCK     JSR       RESET
677  20FD  96 10               LDAA      X.WORD+1
678  20FF  4D                  TST       A
679  2100  27 1D               BEQ       CLEND     NO CLOCK - DO NOTHING
680  2102  CE C2 A5            LDX       =READ.R   JUST DEFINE ANY MIR
681  2105  BD C1 1D            JSR       LDMIR
682  2108  BD C1 58            JSR       EDMIR
683  210B  81 01               CMPA      =1
684  210D  26 0A               BNE       CL1
685  210F  B7 3C 15  CLLOOP    STAA      SSP       SINGLE STEP
686  2112  BD C0 46            JSR       BREAK
687  2115  27 F8               BEQ       CLLOOP
688  2117  20 06               BRA       CLEND
689  2119  BD C1 66  CL1       JSR       START     START RUNNING CLOCK
690  211C  BD C0 24            JSR       R.CH      WAIT ANY KEY
691  211F  BD C2 90  CLEND     JSR       RESET
692  2122  39                  RTS
693                     *
694                     ************************************************
695                     *
696                     *  PROCEDURE MANUAL.CHECK;
697                     *    (* ALLOWS MANUAL SETTING OF MICROINSTRUCTIONS,
698                     *       CPU-BUS AND CLOCK CONTROL
699                     *       => CPU-BUS IS ADDITIONALLY CONTROLLED BY THE
700                     *       BUS SOURCE FIELD OF A MICROINSTRUCTION *)
701                     *    BEGIN EDMIR;
702                     *      LOOP
703                     *        WRITELN(CPUBUS,' ',UAR);
704                     *        READ(A)
705                     *        CASE A OF
706                     *        'I': FOR B := 4 TO 0 BY -1 DO
707                     *               MIR[I] := IN.BYTE; EDMIR
708                     *        'B': CPU.BUS := HEX.IN
709                     *        'G': START RUNNING CLOCK
710                     *        'H': HALT RUNNIG CLOCK
711                     *       CR : RESET; EXIT
712                     *        ELSE ONE SINGLE STEP
713                     *        END;
714                     *        WRITE(' ')
715                     *      END
716                     *    END
717                     *
718  2123  BD C1 58  MANUALCHK JSR       EDMIR
719  2126  FE 3C 0E  M.LOOP    LDX       CPU1
720  2129  BD C2 87            JSR       PRINTX
721  212C  BD C0 62            JSR       W.1.BLANK
722  212F  B6 3C 10            LDAA      UAR1
723  2132  84 0F               ANDA      =0F       MASK OFF  NOT USED BITS
724  2134  BD C0 C9            JSR       OUT.BYTE
725  2137  B6 3C 11            LDAA      UAR0
```

```
726  213A  BD C0 C9          JSR       OUT.BYTE
727  213D  BD C0 78          JSR       W.CRLF
728  2140  BD C0 24          JSR       R.CH
729  2143  81 49             CMPA      ='I'       SPECIFY MICROINSTRUCTION
730  2145  26 16             BNE       B.LABEL
731  2147  C6 05     I.LABEL  LDAB      =5
732  2149  CE 3C 08          LDX       =MIR4
733  214C  BD C0 BC  I.LOOP   JSR       IN.BYTE
734  214F  A7 00             STAA      0,X
735  2151  08                INX
736  2152  BD C0 62          JSR       W.1.BLANK
737  2155  5A                DEC       B
738  2156  26 F4             BNE       I.LOOP
739  2158  BD C1 58          JSR       EDMIR
740  215B  20 C9             BRA       M.LOOP
741  215D  81 42     B.LABEL  CMPA      ='B'       SPECIFY THE BUS CONTENT
742  215F  26 08             BNE       G.LABEL
743  2161  CE 3C 0E          LDX       =CPU1
744  2164  BD C0 D8          JSR       IN.WORD
745  2167  20 BD             BRA       M.LOOP
746  2169  81 47     G.LABEL  CMPA      ='G'       START RUNNING CLOCK
747  216B  26 05             BNE       H.LABEL
748  216D  BD C1 66          JSR       START
749  2170  20 B4             BRA       M.LOOP
750  2172  81 48     H.LABEL  CMPA      ='H'       HALT RUNNING CLOCK
751  2174  26 05             BNE       CR.LABEL
752  2176  BD C1 6D          JSR       STOP
753  2179  20 AB             BRA       M.LOOP
754  217B  81 0D     CR.LABEL CMPA      =CR
755  217D  26 04             BNE       ELSE.LAB
756  217F  BD C2 90          JSR       RESET
757  2182  39                RTS
758  2183  B7 3C 15  ELSE.LAB STAA      SSP        SINGLE STEP
759  2186  BD C0 62          JSR       W.1.BLANK
760  2189  20 9B             BRA       M.LOOP
761                 *
762                 *******************************************
763                 *
764                 *  PROCEDURE GO;
765                 *    BEGIN SETUPC(X.WORD); START END GO;
766                 *
767  218B  BD E2 D5  GO       JSR       SETUPC     TAKES THE PARAMETER FROM X.WORD
768  218E  BD C1 66          JSR       START
769  2191  39                RTS
770                 *
771                 *
772                 ************************************************************
773                 *
774                 *         ENTRY TO THE HARDWARE MONITOR
775                 *
776                 ************************************************************
777                 *
778  2192  CE 21 A0  HWTEST   LDX       =HW.TEXT
779  2195  DF 0B             STX       MAIN.ID
780  2197  BD C2 90          JSR       RESET
781  219A  CE 21 A7          LDX       =HW.MENU
782  219D  7E C3 91          JMP       COMMANDS
```

```
783                     *
784                     ************************************************************
785                     *
786  21A0  48 57 54  HWTEXT    BYTE      'HWTEST',0
     21A3  45 53 54
     21A6  00
787  21A7  0C        HW.MENU   BYTE      0C
788  21A8  58 20 45            BYTE      'X EXECUTE CHAIN',0
     21AB  58 45 43
     21AE  55 54 45
     21B1  20 43 48
     21B4  41 49 4E
     21B7  00
789  21B8  20 3D               WORD      LOOPCHAIN
790  21BA  4C 20 4C            BYTE      'L LOOP ROUTINE',0
     21BD  4F 4F 50
     21C0  20 52 4F
     21C3  55 54 49
     21C6  4E 45 00
791  21C9  20 44               WORD      LOOP.RTNE
792  21CB  54 20 54            BYTE      'T TEST BUS',0
     21CE  45 53 54
     21D1  20 42 55
     21D4  53 00
793  21D6  20 BA               WORD      TESTBUS
794  21D8  4E 20 50            BYTE      'N PANEL SELF',0
     21DB  41 4E 45
     21DE  4C 20 53
     21E1  45 4C 46
     21E4  00
795  21E5  22 87               WORD      SELFTEST.
796  21E7  4F 20 43            BYTE      'O CLOCK  NO%SINGLE%RUNNING',0
     21EA  4C 4F 43
     21ED  4B 20 20
     21F0  4E 4F 07
     21F3  53 49 4E
     21F6  47 4C 45
     21F9  07 52 55
     21FC  4E 4E 49
     21FF  4E 47 00
797  2202  20 FA               WORD      CLOCK
798  2204  48 20 44            BYTE      'H DO BY HAND',0
     2207  4F 20 42
     220A  59 20 48
     220D  41 4E 44
     2210  00
799  2211  21 23               WORD      MANUALCHK
800  2213  47 20 47            BYTE      'G GO',0
     2216  4F 00
801  2218  21 8B               WORD      GO
802  221A  55 20 75            BYTE      'U ',75,'CTRL',0
     221D  43 54 52
     2220  4C 00
803  2222  25 40               WORD      MICROCTRL
804  2224  50 20 43            BYTE      'P CPU',0
     2227  50 55 00
805  222A  2B FA               WORD      ALU
```

```
806  222C  53 20 53          BYTE    'S STORE & PORT',0
     222F  54 4F 52
     2232  45 20 08
     2235  20 50 4F
     2238  52 54 00
807  223B  2E 58             WORD    STOR.PORT
808  223D  52 20 52          BYTE    'R READ REGISTER',0
     2240  45 41 44
     2243  20 52 45
     2246  47 49 53
     2249  54 45 52
     224C  00
809  224D  22 73             WORD    RDREG
810  224F  57 20 57          BYTE    'W WRITE REGISTER',0
     2252  52 49 54
     2255  45 20 52
     2258  45 47 49
     225B  53 54 45
     225E  52 00
811  2260  22 7C             WORD    WRREG
812                  *
813  2262  1B        GLOB.CHAIN BYTE  GL.CH.LEN+MC.CH.LEN+1+ALU.CH0.LEN
814  2263  20 4A 21          WORD    SET.MAIN.ID,HWTEXT
     2266  A0
815  2267  22 87 00          WORD    SELFTEST,0
     226A  00
816  226B  20 BA 00          WORD    TESTBUS,0
     226E  00
817  226F  20 3A 25          WORD    NEXTCHAIN,UC.CHAIN-1
     2272  C6
818                  GL.CH.LEN EQU   *-1-GLOB.CHAIN/4
819                  ************************************************
820  2273  D6 10     RDREG   LDAB    X.WORD+1
821  2275  BD C2 DD          JSR     READREG
822  2278  BD C2 87          JSR     PRINTX
823  227B  39                RTS
824                  ****************************************
825  227C  D6 10     WRREG   LDAB    X.WORD+1
826  227E  BD C2 FE          JSR     GETWORD
827  2281  DE 0F             LDX     X.WORD
828  2283  BD C2 AA          JSR     LOAD.REG
829  2286  39                RTS
830                  ************************************************
831                  *
832  2287                    END     HARDWARE
```

```
                        ***********************************************
                        *
                        *         PANEL SELFTEST PROGRAM
                        *
                        *         AUTHOR JIRKA HOPPE
                        *         VERSION 1. 10. 1979
                        *
                        ***********************************************
2287                    BEGIN SELFTEST
                        USE       END.TEST
                        DEF       SELFTEST
                        ***********************************************
                        * ENABLE DIAGNOSTIC MIR AND TEST ALL POSIBLE PATTERNS
                        *
                        *  PROCEDURE TEST.MIR
                        *    BEGIN SUB.ID := 'MIR.TST'; EDMIR;
                        *      FOR I := 4 TO 0 STEP -1 DO
                        *        FOR A := 0 TO 255 DO
                        *        MIR[I] := A;
                        *        IF MIR[I] <> A THEN S.W.I(MIR[I],A) END;
                        *      END
                        *    END TEST.MIR
                        *
2287  CE 22 C8  SELFTEST  LDX       =MIR.TEXT
228A  DF 0D               STX       SUB.ID
228C  BD C1 58            JSR       EDMIR
228F  CE 3C 08            LDX       =MIR4
2292  4F          L1      CLR       A
2293  A7 00       L2      STAA      0,X       LOOP ALL COMBINATIONS
2295  A1 00               CMPA      0,X
2297  27 03               BEQ       L4
2299  E6 00               LDAB      0,X
229B  3F                  SWI                 B=ACTUAL; A=EXPECTED; X=ADR OF MIR ****
                        
229C  4C          L4      INC       A
229D  26 F4               BNE       L2
229F  08                  INX
22A0  8C 3C 0D            CPX       =MIR0+1   LAST MIR + 1
22A3  26 ED               BNE       L1
                        *
                        *****************************************
                        *
                        *  PROCEDURE TEST.U.ADR
                        *    BEGIN SUB.ID := 'UADR.TST'; D2911;
                        *      FOR X := 0 TO 0FFFH DO
                        *        UAR := X;
                        *        IF UAR <> X THEN S.W.I(UAR,X) END
                        *      END
                        *    END TEST.U.ADR;
                        *
22A5  CE 22 D0            LDX       =UAR.TXT
22A8  DF 0D               STX       SUB.ID
22AA  BD C1 3E            JSR       D2911     DISABLE 2911 ADR OUTPUT; ENABLE PANEL
22AD  CE 00 00            LDX       =0
22B0  FF 3C 10  L5        STX       UAR1
22B3  BC 3C 10            CPX       UAR1
```

```
890  22B6  27 07              BEQ       L6
891  22B8  F6 3C 10           LDAB      UAR1
892  22BB  B6 3C 11           LDAA      UAR0
893  22BE  3F                 SWI                 (B,A)=ACTUAL; X=EXPECTED *****
894  22BF  08          L6     INX
895  22C0  8C 10 00           CPX       =1000
896  22C3  26 EB              BNE       L5
897  22C5  7E 20 4D           JMP       END.TEST
898                    *
899  22C8  4D 49 52  MIR.TEXT BYTE      'MIR.TST',0
     22CB  2E 54 53
     22CE  54 00
900  22D0  55 41 52  UAR.TXT  BYTE      'UAR.TST',0
     22D3  2E 54 53
     22D6  54 00
901  22D8                     END       SELFTEST
```

```
903  22D8                    BEGIN     MICROCTRL
904                     ***************************************************
905                     *
906                     *         MICRO CONTROLLER TEST PROGRAMS
907                     *
908                     *         AUTHOR    IMMO NOACK / RICHARD OHRAN
909                     *         MODIFICATION JIRKA HOPPE
910                     *
911                     *         VERSION     23.9.80
912                     *
913                     ***************************************************
914                               DEF       MICROCTRL,UCCHAIN,MC.CH.LEN
915                               USE       END.TEST,CHAIN,SET.MAIN.ID,WRITELN.A
916                               USE       NEXTCHAIN,ALU.CHAIN,X.L.ROTATE,X.R.ROTATE
917
918                     ***************************************************
919                     *           DST FCT SRC C  A  B  SH  PC S  E  SC DST  SRC
920                     *JMP.MIR   0 0          T           JMP
921  22D8  00 08 07     JMP.MIR   BYTE      00,08,07,20,0F
     22DB  20 0F
922                     *CONTINUE  1 -  OR  DZ  0  0  0  -    - -  0  0   -    -
923  22DD  2F 80 04     CONTINUE  BYTE      2F,80,04,60,0F
     22E0  60 0F
924                     *JMPMAP    1 Q  OR  DZ  0  -  -  -  JMP -  -  0  ALU  IR4
925  22E2  0F 80 07     JMPMAP    BYTE      0F,80,07,60,03
     22E5  60 03
926                     *JSR.MIR   0 0          T           JSR
927  22E7  00 08 03     JSR.MIR   BYTE      00,08,03,20,0F
     22EA  20 0F
928                     *RTN.MIR   0 0          T           RTN
929  22EC  00 08 01     RTN.MIR   BYTE      00,08,01,20,0F    MUST IMMEDIATELLY FOLLOW JSR.MIR
     22EF  20 0F
930                     ***********************************************************
931                     *
932                     *  PROCEDURE JMPOP;
933                     *   (* TESTING JUMP OPERATIONS *)
934                     *   BEGIN SUB.ID := 'JUMP';
935                     *     LDMIR (JMP.MIR);
936                     *     FOR I := 0 TO 4K DO
937                     *       MIR.ADR := I; SINGLE STEP;
938                     *       IF UAR # I THEN S W I END
939                     *     END
940                     *  END JMPOP;
941                     *
942  22F1  CE 23 2D     JMPOP     LDX       =JUMPS
943  22F4  DF 0D                STX       SUB.ID
944  22F6  CE 22 D8             LDX       =JMP.MIR
945  22F9  BD C1 1D             JSR       LDMIR
946  22FC  CE 00 00             LDX       =0
947  22FF  BD C1 58             JSR       EDMIR
948  2302  B7 3C 15     JMP.FOR   STAA      SSP  SINGLE STEP
949  2305  BC 3C 10             CPX       UAR1
950  2308  27 07                BEQ       JMP.OK
951
952  230A  F6 3C 10             LDAB      UAR1
953  230D  B6 3C 11             LDAA      UAR0
```

```
954  2310  3F                SWI           (B,A)=ACTUAL; X=DESIRED***********
955
956  2311  B6 3C 09  JMP.OK   LDAA    MIR3      MIR.ADR := MIR.ADR + 1
957  2314  8B 10             ADDA    =10       POSITION OF '1' ON THE MIR.ADR
958  2316  B7 3C 09          STAA    MIR3
959  2319  24 03             BCC     JMP.N.C   CARRY IS SET FROM THE ADDITION
960  231B  7C 3C 08          INC     MIR4
961  231E  08        JMP.N.C  INX
962  231F  8C 10 00           CPX     =1000
963  2322  26 DE              BNE     JMP.FOR
964  2324  CE 00 00           LDX     =0
965  2327  BD C1 C2           JSR     STUPC     LEAVE UAR IN SAFE STATE
966  232A  7E 20 4D           JMP     END.TEST
967
968  232D  4A 55 4D  JUMPS    BYTE    'JUMP',0
     2330  50 00
969                  *
970                  ***********************************************************
971                  *
972                  *  PROCEDURE JSR.RTN;
973                  *    (*TESTING STACK OPERATION*)
974                  *    (* PUSH & POP ALL POSSIBLE ADDRESSES *)
975                  *    BEGIN
976                  *      FOR I = 0 TO 4K DO
977                  *        STUPC(I);
978                  *        MIR := (0 0 T JSR); SINGLE STEP;
979                  *        MIR := (0 0 T RTN);
980                  *    IF UAR # I THEN ERROR
981                  *    END
982
983
984
985  2332  CE 23 6B  PUSH.POP  LDX     =PU.PO
986  2335  DF 0D              STX     SUB.ID
987  2337  CE 00 00           LDX     =0
988  233A  BD C1 58           JSR     EDMIR
989  233D  BD C1 C2  MC5      JSR     STUPC
990  2340  86 03              LDAA    =03       MIR := JSR
991  2342  B7 3C 0A           STAA    MIR2
992  2345  B7 3C 15           STAA    SSP       SINGLE STEP
993  2348  86 01              LDAA    =01       MIR := RTN
994  234A  B7 3C 0A           STAA    MIR2
995  234D  08                 INX
996  234E  BC 3C 10           CPX     UAR1
997  2351  27 07              BEQ     MC4
998  2353  F6 3C 10           LDAB    UAR1
999  2356  B6 3C 11           LDAA    UAR0
1000 2359  3F                 SWI           (B,A)=ACTUAL; X DESIRED *************
1001
1002 235A  B7 3C 15  MC4      STAA    SSP       SINGLE STEP
1003 235D  8C 0F FF           CPX     =0FFF
1004 2360  26 DB              BNE     MC5
1005 2362  CE 00 00           LDX     =0
1006 2365  BD C1 C2           JSR     STUPC     LEAVE UAR IN SAFE STATE
1007 2368  7E 20 4D           JMP     END.TEST
1008 236B  50 53 48  PU.PO    BYTE    'PSH/POP',0
     236E  2F 50 4F
```

```
     2371  50 00
1009                 *
1010                 ******************************************************
1011                 *  PROCEDURE STACKDPTH;
1012                 *    (*TESTING STACK OPERATIONS
1013                 *      TEST DEPTH OF STACK BY
1014                 *      PUSHING OPERANDS*)
1015                 *    BEGIN
1016                 *      HELP2:=111H;
1017                 *      FOR I:= 1 TO STACKDEPTH DO
1018                 *        SETUPC(HELP2); LEFTROTATE(HELP2);
1019                 *        MIR := (0 0 T JSR); SINGLE STEP
1020                 *      END;
1021                 *      HELP2 := 888H;
1022                 *      FOR I:= STACKDEPTH TO 1 BY -1 DO
1023                 *        MIR := (0 0 T RTN);
1024                 *        IF UAR # HELP2 THEN S W I END;
1025                 *        SINGLE STEP;
1026                 *        RIGHTROTATE(HELP2)
1027                 *      END;
1028                 *    END STACKDPTH;
1029                 *
1030                 STACKDEP  EQU       4
1031                 *
1032  2373  CE 23 B7 STACKDPTH LDX       =STACK
1033  2376  DF 0D              STX       SUB.ID
1034  2378  CE 01 11           LDX       =111
1035  237B  C6 04              LDAB      =STACKDEP.
1036  237D  BD C1 58           JSR       EDMIR
1037  2380  DF 30     MC11     STX       HELP2
1038  2382  09                DEX                     JSR STORES UADR+1; COMPENSATE IT
1039  2383  BD C1 C2           JSR       STUPC
1040  2386  CE 22 E7           LDX       =JSR.MIR
1041  2389  BD C1 17           JSR       EXX
1042  238C  DE 30              LDX       HELP2
1043  238E  BD 20 8D           JSR       X.L.ROTATE
1044  2391  5A                DEC       B
1045  2392  26 EC              BNE       MC11
1046
1047  2394  CE 22 EC           LDX       =RTN.MIR
1048  2397  BD C1 1D           JSR       LDMIR
1049  239A  CE 08 88           LDX       =888
1050  239D  BC 3C 10  MC7      CPX       UAR1
1051  23A0  27 07              BEQ       MC6
1052  23A2  F6 3C 10           LDAB      UAR1
1053  23A5  B6 3C 11           LDAA      UAR0
1054  23A8  3F                 SWI               (B,A)=ACTUAL; X=EXPECTED*******
1055
1056  23A9  B7 3C 15  MC6      STAA      SSP
1057  23AC  BD 20 A3           JSR       X.R.ROTATE
1058  23AF  8C 80 88           CPX       =8088
1059  23B2  26 E9              BNE       MC7
1060  23B4  7E 20 4D           JMP       END.TEST
1061
1062  23B7  44 45 50  STACK    BYTE      'DEPTH',0
      23BA  54 48 00
1063                 *
```

```
1064                    ***************************************************
1065                    *
1066                    *  PROCEDURE INCPC;
1067                    *    (*TEST INCREMENTING OF PC *)
1068                    *    BEGIN
1069                    *     PC := 0
1070                    *     FOR I:= 0 TO 4095 DO
1071                    *       IF I # UAR THEN S W I END;
1072                    *       SINGLESTEP;
1073                    *     END
1074                    *    END INCPC;
1075                    *
1076  23BD  CE 23 F1    INCPC     LDX       =INC
1077  23C0  DF 0D               STX       SUB.ID
1078  23C2  CE 00 00            LDX       =0
1079  23C5  BD C1 58            JSR       EDMIR
1080  23C8  BD C1 C2            JSR       STUPC
1081  23CB  DF 30               STX       HELP2
1082  23CD  CE 22 DD            LDX       =CONTINUE
1083  23D0  BD C1 1D            JSR       LDMIR
1084  23D3  DE 30               LDX       HELP2
1085  23D5  08                  INX
1086
1087  23D6  BC 3C 10    MC9       CPX       UAR1
1088  23D9  27 07               BEQ       MC8
1089  23DB  F6 3C 10            LDAB      UAR1
1090  23DE  B6 3C 11            LDAA      UAR0
1091  23E1  3F                  SWI               (B,A)=ACTUAL; X=DESIRED ***********
1092  23E2  B7 3C 15    MC8       STAA      SSP       (* SINGLE STEP *)
1093  23E5  08                  INX
1094  23E6  8C 10 00            CPX       =01000
1095  23E9  26 EB               BNE       MC9
1096  23EB  B7 3C 15            STAA      SSP       ROLL UPC TO LEAVE IN SAVE STATE
1097  23EE  7E 20 4D            JMP       END.TEST
1098
1099  23F1  50 43 2D    INC       BYTE      'PC-INC',0
      23F4  49 4E 43
      23F7  00
1100                    *
1101                    *
1102                    ***********************************************
1103                    *
1104                    *  TEST CONSTANT
1105                    *  PROCEDURE CONST.TST;
1106                    *  BEGIN SUB.ID:='CONST';
1107                    *    MIR := (1 - OR DZ 0 0 0 - - - 1 0 0 );
1108                    *    FOR A:=0 TO 255 DO
1109                    *        MIR0:= A;
1110                    *        IF CPU <> BUS THEN PRINTBUS  END;
1111                    *    END;
1112                    *  END;
1113                    *
1114  23F8  CE 24 1D    CONST.TST LDX       =CONST
1115  23FB  DF 0D               STX       SUB.ID
1116  23FD  86 70               LDAA      =70       MIR1 := CONST DEFINITION
1117  23FF  B7 3C 0B            STAA      MIR1
1118  2402  BD C1 58            JSR       EDMIR
```

```
1119  2405  CE 00 00           LDX       =0
1120  2408  4F                 CLR       A
1121  2409  B7 3C 0C  CON1     STAA      MIR0
1122  240C  BC 3C 0E           CPX       CPU1
1123  240F  27 03              BEQ       CON2
1124  2411  BD E1 AA           JSR       PRINTBUS  (B,A)=ACTUAL; X=DESIRED *********
1125  2414  08        CON2     INX
1126  2415  4C                 INC       A
1127  2416  81 FF              CMPA      =255Z
1128  2418  26 EF              BNE       CON1
1129  241A  7E 20 4D           JMP       END.TEST
1130                    *
1131  241D  43 4E 53  CONST    BYTE      'CNST.TEST',0
      2420  54 2E 54
      2423  45 53 54
      2426  00
1132
1133                    *
1134                    ************************************************************
1135                    *  PROCEDURE BUS SRC.DEST
1136                    *     BEGIN
1137                    *       MIR1 :=60H;
1138                    *       MIR0 :=X.WORD;
1139                    *            LOOP
1140                    *               R.CH (A);
1141                    *               IF A:='/' THEN INC (MIR0,10H)(INCREMENT DST);
1142                    *               ELSIF A:=',' THEN INC (MIR0,1  )(INCREMENT SRC);
1143                    *               ELSE EXIT;
1144                    *               OUT.BYTE (A);
1145                    *            END;
1146                    *     END;
1147                    *
1148  2427  96 10     B.SRC.DST LDAA     X.WORD+1
1149  2429  B7 3C 0C           STAA      MIR0
1150  242C  86 60              LDAA      =60H
1151  242E  B7 3C 0B           STAA      MIR1
1152  2431  BD C1 58           JSR       EDMIR
1153  2434  BD C0 24  READ     JSR       R.CH
1154  2437  81 2F              CMPA      '/'
1155  2439  26 0A              BNE       SRC
1156  243B  F6 3C 0C           LDAB      MIR0
1157  243E  CB 10              ADDB      =10
1158  2440  F7 3C 0C           STAB      MIR0
1159  2443  20 07              BRA       NEXT
1160  2445  81 2C     SRC      CMPA      ','
1161  2447  26 0B              BNE       EXIT
1162  2449  7C 3C 0C           INC       MIR0
1163  244C  B6 3C 0C  NEXT     LDAA      MIR0
1164  244F  BD 20 87           JSR       WRITELN.A
1165  2452  20 E0              BRA       READ
1166  2454  39        EXIT     RTS
1167                    *
1168                    ************************************************
1169                    *
1170                    * BE SURE THAT: STACK.EMPTY AND PC0.L LINES ARE HIGH
1171                    * GROUND ONE REQUEST LINE AND RUN THE PROGRAM
1172                    * THE PROGRAM DISPLAYS THE VALUES OF THE MICRO ADDRESS
```

```
1173                    *   FOR VARIOUS MASK VALUES
1174                    *
1175                    *  THE CORRECT VALUES ARE
1176                    *  REQUEST  MASKS     UAR
1177                    *    0      00 TO FF  00F
1178                    *    1      00    FC  00E
1179                    *    2      00    F8  00D
1180                    *    3      00    F0  00C  FOR ALL OTHER MASKS IS UAR>10
1181                    *    4      00    E0  00B
1182                    *    5      00    C0  00A
1183                    *    6      00    80  009
1184                    *    7      00    00  008
1185                    *
1186                    *  PROCEDURE INTERRUPT.TEST
1187                    *    BEGIN B :=0; (* B CONTAINS THE CURRENT MASK VALUE *)
1188                    *      REPEAT CPU.BUS := B;
1189                    *        MIR := (PANEL -> IRM); SSP;
1190                    *        SETUPC(0);
1191                    *        PRINT(UAR); PRINT(B);
1192                    *        MIR := (IRM -> BUS); PRINT(BUS); CRLF;
1193                    *        SHIFT 'B' RO RIGHT WITH MSB SET
1194                    *      UNTIL B = 0FFH
1195                    *    END INTERRUPT.TEST;
1196                    *
1197  2455  BD C1 58    INTERPT   JSR       EDMIR
1198  2458  5F                    CLR       B
1199  2459  F7 3C 0E              STAB      CPU1       THE HIGHER BYTE OF BUS WILL NEVER BE USED
1200  245C  CE C2 A5    INT.REP   LDX       =READ.R    THIS IS TO SET MIR1 TO MIR4 TO A DEFINED STATE ONLY
1201  245F  BD C1 1D              JSR       LDMIR      NO OTHER FUNCTION
1202  2462  F7 3C 0F              STAB      CPU0
1203  2465  86 8D                LDAA      =8D        SRC=PANEL; DST=IRM
1204  2467  B7 3C 0C              STAA      MIR0
1205  246A  B7 3C 15              STAA      SSP        SINGLE STEP
1206  246D  CE 22 E2              LDX       =JMPMAP
1207  2470  BD C1 1D              JSR       LDMIR
1208  2473  CE 3C 10              LDX       =UAR1
1209  2476  BD C0 E3              JSR       OUT.WORD   PRINT(UAR)****************
1210  2479  BD C0 62              JSR       W.1.BLANK
1211  247C  17                    TBA
1212  247D  BD C0 C9              JSR       OUT.BYTE   PRINT(MASK)**************
1213  2480  BD C0 62              JSR       W.1.BLANK
1214  2483  86 F8                LDAA      =0F8       SRC=IRM
1215  2485  B7 3C 0C              STAA      MIR0
1216  2488  B6 3C 0F              LDAA      CPU0
1217  248B  BD C0 C9              JSR       OUT.BYTE   READ BACK MASK AND PRINT IT ******
1218  248E  BD C0 78              JSR       W.CRLF
1219
1220  2491  0D                    SEC
1221  2492  56                    ROR       B          ROTATE B TO RIGHT WITH MSB SET
1222  2493  24 C7                BCC       INT.REP    LSB ROLLS OUT TO CARRY
1223  2495  39                    RTS
1224                    *
1225                    ******************************************************
1226                    *
1227                    *  PROCEDURE MAPPROM;
1228                    *    (*COMPARE MAP-PROM TAPE  WITH MAP PROM *)
1229                    *    (* FORMAT OF THE TAPE
```

```
1230                    *         GARBAGE 'S105' INSTR UADR GARBAGE
1231                    *          INSTR - 4 HEX DIGITS  = INSTRCODE*2 + 4000H
1232                    *          UADR  - 4 HEX DIGIST *)
1233                    *     BEGIN SUB.ID := 'MAP';
1234                    *       INTERRUPT.MASK := 0FFFF; (* TURN OFF INTERRUPT *)
1235                    *       SKIP ONE LINE;
1236                    *       FOR HELP1 := 0 TO 255 DO
1237                    *         WMM(HELP1,0);
1238                    *         F.REG := 0; PC.REG := 0;
1239                    *         LINEREAD; (*STARTS TAPE MOTION*)
1240                    *         REPEAT R.CH(A) UNTIL CH = 'S';  (*OVERREAD GARBAGE*)
1241                    *         FOR B := 0 TO 7 DO R.CH(A) END; (*OVERREAD*)
1242                    *         HELP2 := IN.BYTE + 100H*IN.BYTE; (*MICRO ADR*)
1243                    *         REPEAT R.CH(A) UNTIL CH=CR; (*OVEREAD GARBAGE*)
1244                    *         MIR := (1 Q OR DZ 0 - - - JMP - - 0 ALU IR4); (* JMP MAP *)
1245                    *         IF UADR <> HELP3 THEN S W I(UADR,HELP3) END
1246                    *       END
1247                    *     END MAPPROM
1248                    *
1249  2496  CE 25 34  MAPPROM   LDX       =MAP
1250  2499  DF 0D               STX       SUB.ID
1251                    *         TURN OFF INTERRUPT
1252  249B  CE C2 A5            LDX       =READ.R   DEFINE ANY MIR
1253  249E  BD C1 1D            JSR       LDMIR
1254  24A1  86 8D               LDAA      =8D       SRC=PANEL; DST=INTERRUPT MASK
1255  24A3  B7 3C 0C            STAA      MIR0
1256  24A6  CE FF FF            LDX       =0FFFF
1257  24A9  FF 3C 0E            STX       CPU1     BUS := 0FFFF
1258  24AC  BD C1 58            JSR       EDMIR
1259  24AF  B7 3C 15            STAA      SSP       SINGLE STEP
1260
1261  24B2  CE 00 00            LDX       =0
1262  24B5  DF 0F               STX       X.WORD       USED AS PARAMETER TO LOAD.PC
1263  24B7  BD C0 FB            JSR       LINE.READ SKIP FIRST LINE
1264  24BA  BD E0 03  REP.MP0   JSR       READ.CH
1265  24BD  81 0D               CMPA      =CR
1266  24BF  26 F9               BNE       REP.MP0
1267  24C1  CE 00 00            LDX       =0
1268  24C4  DF 2E               STX       HELP1
1269  24C6  7F 00 30            CLR       HELP2     CLR HIGHER BYTE OF HELP2
1270  24C9  BD C1 58  FORMPROM  JSR       EDMIR
1271  24CC  D6 2F               LDAB      HELP1+1
1272  24CE  96 2E               LDAA      HELP1
1273  24D0  CE 00 00            LDX       =0
1274  24D3  BD C2 28            JSR       WMM
1275
1276  24D6  CE 00 00            LDX       =0
1277  24D9  86 ED               LDAA      =0ED       SRC=PANEL; DST = F.REG
1278  24DB  B7 3C 0C            STAA      MIR0
1279  24DE  B7 3C 15            STAA      SSP       F.REG :=0
1280  24E1  86 2D               LDAA      =2D       SRC=PANEL; DST = PC
1281  24E3  B7 3C 0C            STAA      MIR0
1282  24E6  B7 3C 15            STAA      SSP       PC := 0
1283  24E9  BD C0 FB            JSR       LINE.READ
1284  24EC  BD E0 03  REPMP1    JSR       READ.CH
1285  24EF  81 53               CMPA      ='S'
1286  24F1  26 F9               BNE       REPMP1    FIND BEGIN
```

```
1287  24F3  C6 07          LDAB    =7
1288  24F5  BD E0 03 FOR2MP JSR     READ.CH     OVERREAD
1289  24F8  5A             DEC     B
1290  24F9  26 FA          BNE     FOR2MP
1291  24FB  BD E0 1A       JSR     READ.BYTE
1292  24FE  97 31          STAA    HELP2+1  LOWER BYTE OF MICRO ADR
1293  2500  BD E0 1A       JSR     READ.BYTE
1294  2503  97 30          STAA    HELP2    HIGH BYTE
1295  2505  BD E0 03 REPMP2 JSR     READ.CH     OVERREAD GARBAGE
1296  2508  81 0D          CMPA    =CR
1297  250A  26 F9          BNE     REPMP2
1298  250C  CE 22 E2       LDX     =JMPMAP
1299  250F  BD C1 1D       JSR     LDMIR
1300  2512  DE 30          LDX     HELP2    GET DESIRED ADR
1301  2514  BC 3C 10       CPX     UAR1
1302  2517  27 0C          BEQ     ENDIFMP
1303  2519  96 2F          LDAA    HELP1+1
1304  251B  BD 20 87       JSR     WRITELN.A PRINT(INSTRUCTION)**********
1305  251E  F6 3C 10       LDAB    UAR1
1306  2521  B6 3C 11       LDAA    UAR0
1307  2524  3F             SWI           (B,A)=ACTUAL; X=DESIRED******
1308  2525  DE 2E    ENDIFMP LDX     HELP1
1309  2527  8C 00 FF       CPX     =0FF
1310  252A  27 05          BEQ     ENDMP
1311  252C  08             INX
1312  252D  DF 2E          STX     HELP1
1313  252F  20 98          BRA     FORMPROM
1314
1315  2531  7E 20 4D ENDMP  JMP     END.TEST
1316                *
1317  2534  4D 41 50 MAP    BYTE    'MAP',0
      2537  00
1318                *
1319                ********************************************************
1320                *
1321                *       LOOP ALL TESTS USING CHAIN
1322  2538  CE 25 C7 LOOPTEST LDX     =UCCHAIN
1323  253B  BD 20 15       JSR     CHAIN
1324  253E  20 F8          BRA     LOOPTEST
1325                *
1326                ********************************************************
1327                *
1328                *       ENTRY POINT TO THE MICRO MONITOR
1329                *
1330                ********************************************************
1331                *
1332  2540  CE 25 C0 MICROCTRL LDX     =MICROCNTR
1333  2543  DF 0B          STX     MAIN.ID
1334  2545  CE 25 4B       LDX     =CONTMENU
1335  2548  7E C3 91       JMP     COMMANDS
1336
1337  254B  09       CONTMENU BYTE    09
1338  254C  4A 20 4A       BYTE    'J JUMPS',0
      254F  55 4D 50
      2552  53 00
1339  2554  22 F1          WORD    JMPOP
1340  2556  50 20 50       BYTE    'P PUSH.POP',0
```

```
      2559  55 53 48
      255C  2E 50 4F
      255F  50 00
1341  2561  23 32              WORD     PUSH.POP
1342  2563  54 20 44           BYTE     'T DEPTH OF STACK',0
      2566  45 50 54
      2569  48 20 4F
      256C  46 20 53
      256F  54 41 43
      2572  4B 00
1343  2574  23 73              WORD     STACKDPTH
1344  2576  49 20 49           BYTE     'I INC OF PC',0
      2579  4E 43 20
      257C  4F 46 20
      257F  50 43 00
1345  2582  23 BD              WORD     INCPC
1346  2584  4B 20 43           BYTE     'K CONST',0
      2587  4F 4E 53
      258A  54 00
1347  258C  23 F8              WORD     CONST.TST
1348  258E  53 20 53           BYTE     'S SRC/DST',0
      2591  52 43 2F
      2594  44 53 54
      2597  00
1349  2598  24 27              WORD     B.SRC.DST
1350  259A  52 20 49           BYTE     'R INTERPT',0
      259D  4E 54 45
      25A0  52 50 54
      25A3  00
1351  25A4  24 55              WORD     INTERPT
1352  25A6  4D 20 4D           BYTE     'M MAPROM',0
      25A9  41 50 52
      25AC  4F 4D 00
1353  25AF  24 96              WORD     MAPPROM
1354  25B1  58 20 45           BYTE     'X EXEC CHAIN',0
      25B4  58 45 43
      25B7  20 43 48
      25BA  41 49 4E
      25BD  00
1355  25BE  25 38              WORD     LOOPTEST
1356                    *
1357  25C0  75 43 4F  MICROCNTR BYTE     75,'CONTR',0
      25C3  4E 54 52
      25C6  00
1358                    *
1359  25C7  06        UC.CHAIN  BYTE     MC.CH.LEN
1360  25C8  20 4A 25            WORD     SET.MAIN.ID,MICROCNTR
      25CB  C0
1361  25CC  22 F1 00            WORD     JMPOP,0
      25CF  00
1362  25D0  23 32 00            WORD     PUSH.POP,0
      25D3  00
1363  25D4  23 73 00            WORD     STACK.DPTH,0
      25D7  00
1364  25D8  23 BD 00            WORD     INCPC,0
      25DB  00
1365  25DC  23 F8 00            WORD     CONST.TST,0
```

```
      25DF  00
1366                  MC.CH.LEN EQU       *-1-UC.CHAIN/4
1367  25E0  20 3A 2C             WORD      NEXTCHAIN,ALU.CHAIN-1
      25E3  A3
1368                  *
1369  25E4                      END       MICROCTRL
```

```
1371  25E4                      BEGIN     ALU
1372                    ************************************************************
1373                    *
1374                    *         ALU TEST PROGRAMS
1375                    *
1376                    *         AUTHOR  JIRKA HOPPE
1377                    *         VERSION  7.6.79
1378                    *
1379                    ************************************************************
1380                              DEF       ALU,ALU.CHAIN,ALU.CH0.LEN
1381                              USE       END.TEST,CHAIN,SET.MAIN.ID
1382                              USE       X.R.ROTATE,X.L.ROTATE
1383                              USE       WRITELN.A
1384                    *
1385                    ************************************************************
1386                    *
1387                    *         MICROINSTRUCTION USED BY VARIOUS ROUTINES
1388                    *
1389                    *           DST  FCT  SRC  C A B  SH  PC  S  E  SC  DST  SRC
1390                    *CLR.R0    1  B    &   DZ   0 0 0   -   -  -  0  0    -   ALU
1391  25E4  73 E0 04    CLRR0.MIR BYTE       73,0E0,04,60,0F0
      25E7  60 F0
1392                    *INC.R0    1  B    +   ZB   1 0 0   -   -  -  0  0    -   ALU
1393  25E9  61 C0 04    INCR0.MIR BYTE       61,0C0,04,60,0F0
      25EC  60 F0
1394                    *SHIFT     1  B   OR   DZ   0 0 0   RC  -  -  0  0    ALU PANEL
1395  25EE  6F E0 04    SHIFT.MIR BYTE       6F,0E0,04,60,0D
      25F1  60 0D
1396                    *MASK      1  -   OR   DZ   0 0 0   MC  -  -  0  1    0
1397  25F3  6F 80 09    MASK.MIR  BYTE       6F,80,09,70,00
      25F6  70 00
1398                    *SOURCE    1  B   OR   ZA   0 1 0   -   -  -  0  0    ALU PANEL
1399  25F8  6E 02 04    SOURCE.MIR BYTE      6E,02,04,60,0D
      25FB  60 0D
1400                    *FCT       1  -    +   AB   0 0 1   -   -  -  0  0    -   ALU
1401  25FD  20 E0 24    FCT.MIR   BYTE       20,0E0,24,60,0F0
      2600  60 F0
1402                    *BQL.MIR   1  BQL  +   ZA   0 0 0   -   -  -  0  0    -   ALU
1403  2602  C2 60 04    BQL.MIR   BYTE       0C2,60,04,60,0F0
      2605  60 F0
1404                    *BQR.MIR   1  BQR  +   ZA   0 0 0   -   -  -  0  0    -   ALU
1405  2607  82 60 04    BQR.MIR   BYTE       82,60,04,60,0F0
      260A  60 F0
1406                    *MULT.MIR  1  BQR  +   AB   0 1 0   -   -  -  0  0    MDS ALU
1407  260C  80 E2 04    MULT.MIR  BYTE       80,0E2,04,60,50
      260F  60 50
1408                    *DIV1.MIR  1  B    -+  AB   1 1 0   -   -  -  0  0    -   ALU
1409  2611  64 C2 04    DIV1.MIR  BYTE       64,0C2,04,60,0F0
      2614  60 F0
1410                    *DIV2.MIR  1  BQL  +   AB   0 1 0   -   -  -  0  0    MDS ALU
1411  2616  C0 E2 04    DIV2.MIR  BYTE       0C0,0E2,04,60,50
      2619  60 50
1412                    *PUSH      1  -    OR   DZ   0 0 0   -   -  S  0  0    ALU PANEL
1413  261B  2F 80 04    PUSH.MIR  BYTE       2F,80,04,40,0D
      261E  40 0D
1414                    *POP       1  -    OR   DZ   0 0 0   -   -  S  0  0    -   ALU
```

```
1415  2620  2F 80 04  POP.MIR   BYTE       2F,80,04,40,0F0
      2623  40 F0
1416                   *U.JMP.MIR 0 10            E              JMP
1417  2625  01 00 0F  U.JMP.MIR BYTE       01,00,0F,20,0F
      2628  20 0F
1418                   *
1419                   ************************************************************
1420                   *
1421                   *  TEST OUTPUT FROM 2901
1422                   *    TEST INSTRUCTION LINES, REGISTER,....   (WE ASSUME THAT THESE ARE OK.)
1423                   *
1424                   *  PROCEDURE OUT2901;
1425                   *    BEGIN SUB.ID= 'OUT1';
1426                   *      MIR := (1 B & DZ 0 0 0 - - - 0 0 - ALU); (*CLR R0*); SINGLE STEP;
1427                   *      IF CPU<>0 THEN PRINTBUS END;
1428                   *      SUB.ID:= 'OUT2';
1429                   *      MIR := (1 B + ZB 1 0 0 - - - 0 0 - ALU); (*INC R0*);
1430                   *      FOR X := 1 TO 0FFFFH DO
1431                   *        IF CPUBUS <> X THEN PRINTBUS END;
1432                   *        SINGLE STEP;
1433                   *      END;
1434                   *    END;
1435                   *
1436  262A  CE 26 68  OUT2901   LDX        =OUT1TEXT
1437  262D  DF 0D               STX        SUB.ID
1438  262F  CE 25 E4            LDX        =CLR.R0.MIR
1439  2632  BD C1 1D            JSR        LDMIR      MIR := CLR.R0
1440  2635  BD C1 58            JSR        EDMIR
1441  2638  B7 3C 15            STAA       SSP        SINGLE STEP
1442  263B  CE 00 00            LDX        =0
1443  263E  BC 3C 0E            CPX        CPU1
1444  2641  27 03               BEQ        ENDIF1OUT
1445  2643  BD E1 AA            JSR        PRINTBUS   (B,A)=ACTUAL; X=ACTUAL**********
1446  2646  CE 26 6D  ENDIF1OUT LDX        =OUT2TEXT
1447  2649  DF 0D               STX        SUB.ID
1448  264B  CE 25 E9            LDX        =INC.R0.MIR
1449  264E  BD C1 1D            JSR        LDMIR      MIR := INC.R0
1450  2651  CE 00 01            LDX        =1
1451  2654  BC 3C 0E  FOROUT    CPX        CPU1
1452  2657  27 03               BEQ        ENDIF2OUT
1453  2659  BD E1 AA            JSR        PRINTBUS   (B,A)=ACTUAL; X=ACTUAL**********
1454  265C  B7 3C 15  ENDIF2OUT STAA       SSP        SINGLE STEP
1455  265F  08                  INX
1456  2660  26 F2               BNE        FOROUT     OVERFLOWS TO ZERO
1457
1458  2662  BD C1 5F            JSR        EPMIR
1459  2665  7E 20 4D            JMP        ENDTEST    MAKES RTS
1460                   *
1461  2668  4F 55 54  OUT1TEXT  BYTE       'OUT1',0
      266B  31 00
1462  266D  4F 55 54  OUT2TEXT  BYTE       'OUT2',0
      2670  32 00
1463                   *
1464                   ************************************************************
1465                   *
1466                   *  TEST INPUT LINES TO 2901
1467                   *
```

```
1468                    *  PROCEDURE INP2901;
1469                    *    BEGIN SUB.ID := 'INP';
1470                    *      FOR X := 0 TO 0FFFFH DO HELP1 := X;
1471                    *        MIR := LOADREGISTER; SINGLE STEP;
1472                    *        MIR := READREGISTER;
1473                    *        IF CPUBUS <> HELP1 THEN PRINTBUS END;
1474                    *      END;
1475                    *    END;
1476                    *
1477  2672  CE 26 A6    INP2901   LDX       =INPTEXT
1478  2675  DF 0D                STX       SUB.ID
1479  2677  BD C1 58              JSR       EDMIR
1480  267A  CE 00 00              LDX       =0
1481  267D  DF 2E      FORINP    STX       HELP1
1482  267F  FF 3C 0E              STX       CPU1
1483  2682  CE C2 A0              LDX       =WRITE.R
1484  2685  BD C1 1D              JSR       LDMIR
1485  2688  B7 3C 15              STAA      SSP         SINGLE STEP
1486  268B  CE C2 A5              LDX       =READ.R
1487  268E  BD C1 1D              JSR       LDMIR
1488  2691  DE 2E                 LDX       HELP1
1489  2693  BC 3C 0E              CPX       CPU1
1490  2696  27 03                 BEQ       INP1
1491  2698  BD E1 AA              JSR       PRINTBUS    (B,A)=ACTUAL; X=DESIRED**********
1492  269B  DE 2E      INP1      LDX       HELP1
1493  269D  08                   INX
1494  269E  26 DD                BNE       FORINP      OVERFLOWS TO ZERO
1495
1496  26A0  BD C1 5F              JSR       EPMIR
1497  26A3  7E 20 4D              JMP       END.TEST    MAKES RTS
1498                    *
1499  26A6  49 4E 50    INPTEXT   BYTE      'INP',0
      26A9  00
1500                    *
1501                    ******************************************************
1502                    *
1503                    *  TEST THE 25S10 SHIFTER UNIT
1504                    *
1505                    *  PROCEDURE SHIFTER(X);
1506                    *    VAR HELP1, HELP2;
1507                    *    BEGIN  HELP2 := X; SUB.ID:='SHIFTER';
1508                    *      FOR B := 0 TO 15 DO
1509                    *        CPUBUS := HELP1 := HELP2;
1510                    *        FOR A := 0 TO 15 DO
1511                    *          MIR := (1 B OR DZ 0 0 0 RC - - 0 0 ALU PANEL);
1512                    *          MIR.SHIFT.COUNT := A;
1513                    *          SINGLE STEP;   (* THE SHIFTED VALUE IS READ INTO R0 *)
1514                    *          MIR := (1 - OR ZA 0 0 0 - - - 0 0 - ALU);
1515                    *          IF HELP1 <> CPUBUS THEN WRITELN(A); PRINTBUS; END;
1516                    *          R.ROTATE(HELP1);
1517                    *        END;
1518                    *        L.ROTATE(HELP2);
1519                    *      END
1520                    *    END;
1521                    *
1522  26AA  CE FF FE    SHIFTR0   LDX       =0FFFE      ENTRY FROM MENU
1523  26AD  24 02                 BCC       SHIFTER     IF NO VALUE SPECIFIED TAKE DEFAULT 0FFFE
```

```
1524  26AF  DE 0F              LDX       X.WORD
1525
1526  26B1  DF 30     SHIFTER  STX       HELP2
1527  26B3  CE 27 06           LDX       =SHIFTTEXT
1528  26B6  DF 0D              STX       SUB.ID
1529  26B8  BD C1 58           JSR       EDMIR
1530  26BB  5F                 CLR       B
1531
1532  26BC  DE 30     FOR1SH   LDX       HELP2
1533  26BE  DF 2E              STX       HELP1
1534  26C0  FF 3C 0E           STX       CPU1
1535  26C3  86 60              LDAA      =01100000L MIR1 := (SC=0)
1536  26C5  CE 25 EE  FOR2SH   LDX       =SHIFT.MIR
1537  26C8  BD C1 1D           JSR       LDMIR
1538  26CB  B7 3C 0B           STAA      MIR1       SPECIFY SHIFT COUNT
1539  26CE  B7 3C 15           STAA      SSP        SINGLE STEP
1540  26D1  CE C2 A5           LDX       =READ.R
1541  26D4  BD C1 1D           JSR       LDMIR
1542  26D7  DE 2E              LDX       HELP1
1543  26D9  BC 3C 0E           CPX       CPU1
1544  26DC  27 0A              BEQ       ENDIFSH
1545  26DE  36                 PSH       A
1546  26DF  84 0F              ANDA      =0F        MASK LOWER BITS
1547  26E1  BD 20 87           JSR       WRITELN.A  PRINT(COUNT) ********************
1548  26E4  32                 PUL       A
1549  26E5  BD E1 AA           JSR       PRINTBUS   (B,A) ACTUAL; X DESIRED ********
1550  26E8  DE 2E     ENDIFSH  LDX       HELP1
1551  26EA  BD 20 A3           JSR       X.R.ROTATE
1552  26ED  DF 2E              STX       HELP1
1553  26EF  4C                 INC       A
1554  26F0  81 70              CMPA      =01110000L SHIFT COUNT = 16Z
1555  26F2  26 D1              BNE       FOR2SH
1556
1557  26F4  DE 30              LDX       HELP2
1558  26F6  BD 20 8D           JSR       X.L.ROTATE
1559  26F9  DF 30              STX       HELP2
1560  26FB  5C                 INC       B
1561  26FC  C1 10              CMPB      =10
1562  26FE  2D BC              BLT       FOR1SH
1563
1564  2700  BD C1 5F           JSR       EPMIR
1565  2703  7E 20 4D           JMP       END.TEST   MAKES RTS
1566                  *
1567  2706  53 48 49  SHIFTTEXT BYTE      'SHIFTER',0
      2709  46 54 45
      270C  52 00
1568                  *
1569                  ***************************************************************
1570                  *
1571                  *  TEST 25S10 SHIFTER FOR MASKING OPERATION
1572                  *
1573                  *  PROCEDURE MASK;
1574                  *     BEGIN SUB.ID := 'MASK'; HELP1:=0;
1575                  *       FOR A := 0 TO 15 DO
1576                  *         MIR := (1 - OR DZ 0 0 0 MC - - 0 1 0);
1577                  *         MIR.SHIFTCOUNT := A; SINGLE STEP;
1578                  *         MIR := READREGISTER;
```

```
1579                    *         IF CPUBUS<>HELP1 THEN WRITELN(A); PRINTBUS END;
1580                    *         ARITHMETIC.RIGHT.SHIFT(HELP1);
1581                    *       END;
1582                    *     END;
1583                    *
1584  270E  CE 27 53    MASK      LDX       =MASK.TEXT
1585  2711  DF 0D                STX       SUB.ID
1586  2713  CE 00 00              LDX       =0
1587  2716  DF 2E                STX       HELP1
1588  2718  BD C1 58              JSR       EDMIR
1589  271B  86 70                LDAA      =01110000L   MIR1 := (COUNT = 0)
1590
1591  271D  CE 25 F3    FORMSK    LDX       =MASK.MIR
1592  2720  BD C1 1D              JSR       LDMIR
1593  2723  B7 3C 0B              STAA      MIR1         SET MASK COUNTER
1594  2726  B7 3C 15              STAA      SSP          NOW READ IT INTO R0
1595  2729  CE C2 A5              LDX       =READ.R
1596  272C  BD C1 1D              JSR       LDMIR        READ R0 ONTO CPU BUS
1597  272F  DE 2E                LDX       HELP1
1598  2731  BC 3C 0E              CPX       CPU1
1599  2734  27 0A                BEQ       ENDIFMSK
1600  2736  36                  PSH       A
1601  2737  84 0F                ANDA      =0F          MASK LOWER BITS
1602  2739  BD 20 87              JSR       WRITELN.A    WRITE(SHIFT COUNT) **************
1603  273C  32                  PUL       A
1604  273D  BD E1 AA              JSR       PRINTBUS     (B,A) ACTUAL; X DESIRED *********
1605  2740  DE 2E     ENDIFMSK  LDX       HELP1
1606
1607  2742  08                  INX                    THESE 2 INSTRUCTIONS MAKE ASR(X)
1608  2743  BD 20 A3              JSR       X.R.ROTATE
1609  2746  DF 2E                STX       HELP1
1610  2748  4C                  INC       A
1611  2749  81 80                CMPA      =10000000L SHIFT COUNT = 16
1612  274B  26 D0                BNE       FORMSK
1613
1614  274D  BD C1 5F              JSR       EPMIR
1615  2750  7E 20 4D              JMP       END.TEST     MAKES RTS
1616                    *
1617  2753  4D 41 53    MASK.TEXT BYTE      'MASK',0
      2756  4B 00
1618                    *
1619                    ***************************************************************
1620                    *
1621                    *  TEST 2901 REGISTER ADDRESSING
1622                    *    USE PROCEDURES
1623                    *      - LOADREG => USES B ADDRESSING MODE
1624                    *      - READREG => USES A ADDRESSING MODE
1625                    *
1626                    *  PROCEDURE REGISTERADDRESSING;
1627                    *    BEGIN SUB.ID := 'REGADR';   X := 0
1628                    *      FOR B := 0 TO 15 DO
1629                    *        LOADREG(B,X);  X := X + 1111H; (*INCREMENT ALL NIBBLES *)
1630                    *      END;
1631                    *      HELP1:=0;
1632                    *      FOR B := 0 TO 15 DO
1633                    *        IF X<>HELP1 THEN WRITELN(B); PRINTBUS END;
1634                    *        X := X + 1111H;
```

```
1635                    *       END;
1636                    *    END;
1637                    *
1638  2758  CE 27 9E  REG.ADR   LDX       =REGADRTXT
1639  275B  DF 0D               STX       SUB.ID
1640  275D  CE 00 00            LDX       =0
1641  2760  DF 2E               STX       HELP1
1642  2762  4F                  CLR       A
1643  2763  5F                  CLR       B
1644  2764  BD C1 58            JSR       EDMIR
1645  2767  DE 2E     FOR1RA    LDX       HELP1
1646  2769  BD C2 AA            JSR       LOADREG
1647  276C  8B 11               ADDA      =11       INCREMENT ALL NIBBLES BY ONE
1648  276E  97 2E               STAA      HELP1
1649  2770  97 2F               STAA      HELP1+1
1650  2772  5C                  INC       B
1651  2773  C1 10               CMPB       =10
1652  2775  2D F0               BLT       FOR1RA
1653
1654  2777  CE 00 00            LDX       =0
1655  277A  DF 2E               STX       HELP1
1656  277C  4F                  CLR       A
1657  277D  5F                  CLR       B
1658  277E  BD C2 DD  FOR2RA    JSR       READREG
1659  2781  9C 2E               CPX       HELP1
1660  2783  27 09               BEQ       ENDIFRA
1661  2785  36                  PSH       A
1662  2786  17                  TBA
1663  2787  BD 20 87            JSR       WRITELN.A  WRITE(REG.ADR)  ****************
1664  278A  BD E1 AA            JSR       PRINTBUS   (B,A) := ACTUAL; X-DESIRED ******
1665  278D  36                  PSH       A
1666  278E  DE 2E     ENDIFRA   LDX       HELP1
1667  2790  8B 11               ADDA      =11
1668  2792  97 2E               STAA      HELP1
1669  2794  97 2F               STAA      HELP1+1
1670  2796  5C                  INC       B
1671  2797  C1 10               CMPB      =10
1672  2799  2D E3               BLT       FOR2RA
1673
1674  279B  7E 20 4D            JMP        END.TEST   MAKES RTS
1675                    *
1676  279E  52 45 47  REGADRTXT BYTE       'REGADR',0
      27A1  41 44 52
      27A4  00
1677                    *
1678                    ********************************************************************
1679                    *
1680                    *  TEST 2901 SOURCE ADDRESSING
1681                    *
1682                    *  PROCEDURE SOURCE;
1683                    *    BEGIN SUB.ID := 'SOURCE';
1684                    *      LOADREG(Q,4444H);
1685                    *      FOR A := 0 TO 7 DO
1686                    *        LOADREG(0,1111H);
1687                    *        LOADREG(1,2222H);
1688                    *        CPUBUS:=8888;
1689                    *        MIR := (1 B OR ZA 0 1 0 - - - 0 0 ALU PANEL);
```

```
1690                    *       MIR.RS := A; SINGLE STEP
1691                    *       IF READREG(0) <> TABLE[A] THEN WRITELN(A); PRINTCPU END;
1692                    *     END;
1693                    *   END;
1694                    *
1695  27A5  CE 28 09    SOURCE  LDX     =SRC.TEXT
1696  27A8  DF 0D               STX     SUB.ID
1697  27AA  BD C1 58            JSR     EDMIR
1698  27AD  C6 10               LDAB    =Q.REG
1699  27AF  CE 44 44            LDX     =4444
1700  27B2  BD C2 AA            JSR     LOADREG   Q :=4444
1701  27B5  CE 6C 02            LDX     =0110110000000010L  MIR3,4:=(1 B OR AQ 0 0 1...)
1702  27B8  DF 30               STX     HELP2     SOURCE COUNTER
1703  27BA  4F                  CLR     A
1704  27BB  CE 28 0B            LDX     =SRC.TABLE-2  POINTER TO THE RESULT TABLE
1705  27BE  DF 2E               STX     HELP1
1706  27C0  5F          FORSRC  CLR     B
1707  27C1  CE 11 11            LDX     =1111
1708  27C4  BD C2 AA            JSR     LOADREG   R0:=1111
1709  27C7  5C                  INC     B
1710  27C8  CE 22 22            LDX     =2222
1711  27CB  BD C2 AA            JSR     LOADREG   R1:=2222
1712  27CE  CE 88 88            LDX     =8888
1713  27D1  FF 3C 0E            STX     CPU1      CPUBUS := 8888
1714  27D4  CE 25 F8            LDX     =SOURCE.MIR
1715  27D7  BD C1 1D            JSR     LDMIR
1716  27DA  DE 30               LDX     HELP2
1717  27DC  FF 3C 08            STX     MIR4
1718  27DF  B7 3C 15            STAA    SSP       SINGLE STEP
1719  27E2  CE C2 A5            LDX     =READ.R
1720  27E5  BD C1 1D            JSR     LDMIR
1721  27E8  BD 28 8B            JSR     INC.LOAD
1722  27EB  BC 3C 0E            CPX     CPU1
1723  27EE  27 06               BEQ     ENDIFSRC
1724  27F0  BD 20 87            JSR     WRITELN.A WRITE(RS)*****************************
1725  27F3  BD E1 AA            JSR     PRINTBUS  (B,A)=ACTUAL;  X=DESIRED *************
1726  27F6  D6 31       ENDIFSRC LDAB   HELP2+1   INCREMENT SOURCE COUNTER
1727  27F8  CB 80               ADDB    =80       HELP1 := HELP1 + 80   (* 80 IS POSITION
1728  27FA  D7 31               STAB    HELP2+1                          OF SRC COUNTER *)
1729  27FC  24 03               BCC     NOCARYSRC
1730  27FE  7C 00 30            INC     HELP2
1731  2801  4C          NOCARYSRC INC   A
1732  2802  81 08               CMPA    =8
1733  2804  2D BA               BLT     FORSRC
1734
1735  2806  7E 20 4D            JMP     END.TEST  MAKES RTS
1736                    *
1737  2809  53 52 43    SRC.TEXT BYTE   'SRC',0
      280C  00
1738                    *   A=2222,  B=1111,  Q=4444,  CPU=8888
1739                    *   OR FUNCTION     AQ   AB   ZQ   ZB   ZA    DA   DQ    DZ
1740  280D  66 66 33    SRC.TABLE WORD  6666,3333,4444,1111,2222,0AAAA,0CCCC,8888
      2810  33 44 44
      2813  11 11 22
      2816  22 AA AA
      2819  CC CC 88
      281C  88
```

```
1741                    *
1742                    ************************************************************************
1743                    *
1744                    *  BASIC ARITHMETICAL AND LOGICAL FUNCTION TEST
1745                    *
1746                    *  PROCEDURE FUNCTIONS;
1747                    *    BEGIN SUB.ID := 'FCT';
1748                    *      LOADREG(0,1111H);
1749                    *      LOADREG(1,2222H);
1750                    *      MIR := (1 - + AB 0 0 1 - - - 0 0 - ALU);
1751                    *      FOR A := 0 TO 7 DO   MIR.FCT := A;
1752                    *        IF CPUBUS<>X THEN WRITELN(A); PRINTBUS END;
1753                    *      END;
1754                    *    END;
1755                    *
1756  281D  CE 28 5B    FUNCTIONS LDX       =FCT.TEXT
1757  2820  DF 0D                 STX       SUB.ID
1758  2822  BD C1 58              JSR       EDMIR
1759  2825  5F                    CLR       B
1760  2826  CE 11 11              LDX       =1111
1761  2829  BD C2 AA              JSR       LOADREG    R0 := 1111
1762  282C  5C                    INC       B
1763  282D  CE 22 22              LDX       =2222
1764  2830  BD C2 AA              JSR       LOADREG    R1 := 2222
1765  2833  CE 25 FD              LDX       =FCT.MIR
1766  2836  BD C1 1D              JSR       LDMIR
1767  2839  CE 28 5D              LDX       =FCT.TABLE-2
1768  283C  DF 2E                 STX       HELP1      HELP1 := ↑TABLE
1769  283E  C6 20                 LDAB      =00100000L   MIR4 := (- + AB ...)
1770  2840  4F                    CLR       A          A = COUNTER FOR PRINT OUT
1771  2841  F7 3C 08  FOR.FCT     STAB      MIR4
1772  2844  8D 45                 BSR       INC.LOAD
1773  2846  BC 3C 0E              CPX       CPU1
1774  2849  27 06                 BEQ       ENDIFFCT
1775  284B  BD 20 87              JSR       WRITELN.A  WRITELN(FCT)******************
1776  284E  BD E1 AA              JSR       PRINTBUS   (B,A)=ACTUAL;  X=DESIRED*******
1777  2851  CB 04     ENDIFFCT   ADDB      =4         INCREMENT FCT
1778  2853  4C                    INC       A
1779  2854  81 08                 CMPA      =8
1780  2856  2D E9                 BLT       FOR.FCT
1781
1782  2858  7E 20 4D              JMP       END.TEST   MAKES RTS
1783                    *
1784  285B  46 43 54    FCT.TEXT   BYTE      'FCT',0
      285E  00
1785                    *  A=1111,   B=2222
1786                    *    FCT             +    -+   -     OR   &    -&   XOR  -XOR
1787  285F  33 33 11    FCT.TABLE  WORD      3333,1110,0EEEE,3333,0000,2222,3333,0CCCC
      2862  10 EE EE
      2865  33 33 00
      2868  00 22 22
      286B  33 33 CC
      286E  CC
1788                    ****************************************
1789                    *
1790                    *  PROCEDURE PRINT.HELP;
1791                    *    (* A PRINT HELP FOR PROCEDURES TEST.R0.Q.CHAIN AND DIVIDE *)
```

```
1792                    *    BEGIN HEXOUT(HELP1);
1793                    *      (B,A) := HELP2; S.W.I(B,A,HELP1↑)
1794                    *    END PRINT.HELP
1795                    *
1796  286F  BD C2 59    PRINTHELP JSR      PUSHX      .
1797  2872  36                    PSH      A
1798  2873  37                    PSH      B
1799  2874  DE 2E                LDX      HELP1
1800  2876  BD C2 87              JSR      PRINTX
1801  2879  BD C0 78              JSR      W.CRLF
1802  287C  D6 30                LDAB     HELP2
1803  287E  96 31                LDAA     HELP2+1
1804  2880  DE 2E                LDX      HELP1
1805  2882  EE 00                LDX      0,X
1806  2884  3F                   SWI                 **********
1807  2885  33                   PUL      B
1808  2886  32                   PUL      A
1809  2887  BD C2 70             JSR      PULLX
1810  288A  39                   RTS
1811                    ************************************************
1812                    *  PROCEDURE INC.LOAD
1813                    *    (* INCREMENT A POINTER TO A LIST AND LOAD THE POINTED VALUE *)
1814
1815  288B  DE 2E       INC.LOAD LDX      HELP1
1816  288D  08                   INX
1817  288E  08                   INX
1818  288F  DF 2E                STX      HELP1
1819  2891  EE 00                LDX      0,X
1820  2893  39                    RTS
1821                    ********************************************
1822                    *
1823                    *  PROCEDURE TEST.R0.Q.CHAIN(X : ↑COMMAND STRING);
1824                    *    (* THIS PROCEDURE EXECUTES A COMMAND STRING
1825                    *       AND CHECKS IF THE ACTUAL VALUES OF R0.REG AND Q.REG
1826                    *       CORRESPOND TO THE EXPECTED VALUES
1827                    *       FORMAT OF THE STRING:
1828                    *         SET(R0); SET(Q); ↑MICROINTRUCTION;
1829                    *         EXPECTED(R0); EXPECTED(Q); EXPECTED(R0); EXPECTED(Q) *)
1830                    *    BEGIN HELP1 := X;
1831                    *      LOADREG(0, X↑);
1832                    *      INC(HELP1, 2); LOADREG(Q, HELP1↑);
1833                    *        INC(HELP1, 2); HELP4:=HELP1;
1834                    *      FOR HELP3 := 2 TO 0 STEP -1 DO
1835                    *        EXX(HELP4↑);
1836                    *        HELP2 := READREG(0);
1837                    *        INC(HELP1, 2);
1838                    *        IF HELP2 <> HELP1 THEN PRINTHELP END
1839                    *        INC(HELP1, 2); HELP2 := READREG(Q.REG);
1840                    *        IF HELP2 <> HELP1 THEN PRINTHELP END
1841                    *      END TEST.R0.Q.CHAIN;
1842                    *
1843  2894  DF 2E       TEST.R0.Q STX      HELP1
1844  2896  BD C1 58              JSR      EDMIR
1845  2899  5F                   CLR      B
1846  289A  EE 00                LDX      0,X
1847  289C  BD C2 AA             JSR      LOADREG   R0 := X↑
1848  289F  C6 10                LDAB     =Q.REG
```

```
1849  28A1  8D E8              BSR       INC.LOAD
1850  28A3  BD C2 AA           JSR       LOADREG   Q := HELP1
1851  28A6  86 02              LDAA      =2
1852  28A8  97 32              STAA      HELP3
1853  28AA  8D DF              BSR       INC.LOAD
1854  28AC  DF 34              STX       HELP4
1855  28AE  DE 34     TST.RQ.FOR LDX     HELP4
1856  28B0  BD C1 17           JSR       EXX
1857  28B3  5F                 CLR       B
1858  28B4  BD C2 DD           JSR       READREG
1859  28B7  DF 30              STX       HELP2     HELP2 := R0
1860  28B9  8D D0              BSR       INC.LOAD
1861  28BB  9C 30              CPX       HELP2
1862  28BD  27 02              BEQ       TST.R.OK
1863  28BF  8D AE              BSR       PRINT.HELP
1864  28C1  C6 10     TST.R.OK LDAB      =Q.REG
1865  28C3  BD C2 DD           JSR       READREG
1866  28C6  DF 30              STX       HELP2     HELP2 := Q.REG
1867  28C8  8D C1              BSR       INC.LOAD
1868  28CA  9C 30              CPX       HELP2
1869  28CC  27 02              BEQ       TST.Q.OK
1870  28CE  8D 9F              BSR       PRINT.HELP   ******************
1871  28D0  7A 00 32  TST.Q.OK DEC       HELP3
1872  28D3  26 D9              BNE       TST.RQ.FOR
1873  28D5  39                 RTS
1874                  *
1875                  ***********************************************
1876                  *
1877                  *  PROCEDURE DIVIDE( X : ↑ COMMANDSTRING);
1878                  *    (* TEST A DIVIDE STEP *)
1879                  *     FORMAT : LOAD(R0); LOAD(Q.REG);
1880                  *              TEST   R0,Q,  R0,Q,  R0,Q,  R0,Q *)
1881                  *
1882                  *    BEGIN HELP1 := X;
1883                  *      LOADREG(R0, HELP1↑);
1884                  *      INC(HELP1, 2); LOADREG(Q.REG, HELP1↑);
1885                  *        HELP31 := 0;
1886                  *      FOR HELP3 := 4 TO 0 STEP -1 DO
1887                  *        IF EVEN(HELP31) THEN EXX(DIV1.MIR) ELSE EXX(DIV2.MIR) END;
1888                  *        HELP2 := READREG(R0);  INC(HELP1, 2);
1889                  *        IF HELP2 <> HELP1 THEN PRINTHELP END;
1890                  *        HELP2 := READREG(Q.REG); INC(HELP1, 2);
1891                  *        IF HELP2 <> HELP1 THEN PRINTHELP END;
1892                  *        INC(HELP31)
1893                  *      END
1894                  *    END;
1895                  *
1896  28D6  DF 2E     DIVIDE   STX       HELP1
1897  28D8  BD C1 58           JSR       EDMIR
1898  28DB  5F                 CLR       B
1899  28DC  EE 00              LDX       0,X
1900  28DE  BD C2 AA           JSR       LOADREG   R0:= X↑
1901  28E1  C6 10              LDAB      =Q.REG
1902  28E3  8D A6              BSR       INC.LOAD
1903  28E5  BD C2 AA           JSR       LOADREG   Q := HELP1
1904  28E8  86 04              LDAA      =4
1905  28EA  97 32              STAA      HELP3     COUNTER FOR THE FOR STATEMENT
```

```
1906  28EC  7F 00 33          CLR       HELP3+1
1907  28EF  CE 26 11  DIV.FOR  LDX       =DIV1.MIR
1908  28F2  96 33              LDAA      HELP3+1
1909  28F4  84 01              ANDA      =1
1910  28F6  27 03              BEQ       DO.EXX
1911  28F8  CE 26 16           LDX       =DIV2.MIR
1912  28FB  BD C1 17  DO.EXX   JSR       EXX
1913  28FE  5F                 CLR       B
1914  28FF  BD C2 DD           JSR       READREG
1915  2902  DF 30              STX       HELP2     HELP2 := READREG(0)
1916  2904  8D 85              BSR       INC.LOAD
1917  2906  9C 30              CPX       HELP2
1918  2908  27 03              BEQ       DIV.R0.OK
1919  290A  BD 28 6F           JSR       PRINTHELP  ***************
1920  290D  C6 10     DIV.R0.OK LDAB     =Q.REG
1921  290F  BD C2 DD           JSR       READREG
1922  2912  DF 30              STX       HELP2     HELP2 := READREG(Q)
1923  2914  BD 28 8B           JSR       INC.LOAD
1924  2917  9C 30              CPX       HELP2
1925  2919  27 03              BEQ       DIV.Q.OK
1926  291B  BD 28 6F           JSR       PRINTHELP ******************
1927  291E  7C 00 33  DIV.Q.OK INC       HELP3+1
1928  2921  7A 00 32           DEC       HELP3
1929  2924  26 C9              BNE       DIV.FOR
1930  2926  39                 RTS
1931                    *
1932                    ***********************************************
1933                    *
1934                    *  PROCEDURE SHIFTED.DESTINATIONS
1935                    *    BEGIN
1936                    *      SUB.ID := 'BQL'; TEST.R0.Q.CHAIN(BQL.SEQUENCE);
1937                    *      SUB.ID := 'BQR'; TEST.R0.Q.CHAIN(BQR.SEQUENCE);
1938                    *    END SHIFTED.DESTINATIONS;
1939                    *
1940  2927  CE 29 40  SH.DEST  LDX       =BQL.TEXT
1941  292A  DF 0D              STX       SUB.ID
1942  292C  CE 29 48           LDX       =BQL.SEQ
1943  292F  BD 28 94           JSR       TEST.R0.Q
1944  2932  CE 29 44           LDX       =BQR.TEXT
1945  2935  DF 0D              STX       SUB.ID
1946  2937  CE 29 56           LDX       =BQR.SEQ
1947  293A  BD 28 94           JSR       TEST.R0.Q
1948  293D  7E 20 4D           JMP       END.TEST
1949                    *
1950  2940  42 51 4C  BQL.TEXT BYTE      'BQL',0
      2943  00
1951  2944  42 51 52  BQR.TEXT BYTE      'BQR',0
      2947  00
1952                    *                 R0   Q    MIR     R0   Q    R0   Q
1953  2948  88 88 88  BQL.SEQ  WORD      8888,8888,BQL.MIR,1111,1111,2222,2222
      294B  88 26 02
      294E  11 11 11
      2951  11 22 22
      2954  22 22
1954  2956  11 11 22  BQR.SEQ  WORD      1111,2222,BQR.MIR,0888,9111,0444,4888
      2959  22 26 07
      295C  08 88 91
```

```
      295F  11 04 44
      2962  48 88
1955                  *
1956                  ******************************************************
1957                  *
1958                  *  PROCEDURE MULTIPLICATION;
1959                  *   BEGIN SUB.ID := MULT; LOADREG(1,1)
1960                  *     TEST.R0.Q.CHAIN(MULT1.SEQ);
1961                  *     TEST.R0.Q.CHAIN(MULT2.SEQ);
1962                  *   END MULTIPLICATION;
1963                  *
1964  2964  CE 29 83  MULT      LDX       =MULT.TEXT
1965  2967  DF 0D              STX       SUB.ID
1966  2969  BD C1 58           JSR       EDMIR
1967  296C  C6 01              LDAB      =1
1968  296E  CE 00 01           LDX       =1
1969  2971  BD C2 AA           JSR       LOADREG   R1 := 1
1970  2974  CE 29 88           LDX       =MULT1.SEQ
1971  2977  BD 28 94           JSR       TEST.R0.Q
1972  297A  CE 29 96           LDX       =MULT2.SEQ
1973  297D  BD 28 94           JSR       TEST.R0.Q
1974  2980  7E 20 4D           JMP       END.TEST
1975                  *
1976  2983  4D 55 4C  MULT.TEXT BYTE      'MULT',0
      2986  54 00
1977                  *                   R0 Q          R0 Q   R0  Q
1978  2988  00 05 00  MULT1.SEQ WORD      0005,0002,MULT.MIR,0002,8001,0001,0C000
      298B  02 26 0C
      298E  00 02 80
      2991  01 00 01
      2994  C0 00
1979  2996  FF FB 00  MULT2.SEQ WORD      0FFFB,002,MULT.MIR,07FFD,8001,03FFF,4000
      2999  02 26 0C
      299C  7F FD 80
      299F  01 3F FF
      29A2  40 00
1980                  *
1981                  ************************************************
1982                  *
1983                  *  PROCEDURE DIVISION;
1984                  *    BEGIN SUB.ID := 'DIV'; LOADREG(1, 2)
1985                  *     DIVIDE(DIV1.SEQ);    DIVIDE(DIV2.SEQ);
1986                  *    END DIVISION
1987                  *
1988  29A4  CE 29 CB  DIVISION LDX       =DIV.TEXT
1989  29A7  DF 0D              STX       SUB.ID
1990  29A9  BD C1 58           JSR       EDMIR
1991  29AC  C6 01              LDAB      =1
1992  29AE  CE 00 02           LDX       =2
1993  29B1  BD C2 AA           JSR       LOADREG     R1:=2
1994  29B4  CE 29 CF           LDX       =DIV1.SEQ
1995  29B7  BD 28 D6           JSR       DIVIDE
1996  29BA  C6 01              LDAB      =1
1997  29BC  CE FF FE           LDX       =0FFFE
1998  29BF  BD C2 AA           JSR       LOADREG     R1:=FFFE
1999  29C2  CE 29 E3           LDX       =DIV2.SEQ
2000  29C5  BD 28 D6           JSR       DIVIDE
```

```
2001  29C8  7E 20 4D          JMP       END.TEST
2002                  *
2003  29CB  44 49 56  DIV.TEXT BYTE      'DIV',0
      29CE  00
2004                  *                 R0  Q    R0  Q  R0  Q  R0  Q  R0  Q
2005  29CF  00 01 80  DIV1.SEQ WORD      1,8000,0FFFF,8000,3,0,1,0,2,1
      29D2  00 FF FF
      29D5  80 00 00
      29D8  03 00 00
      29DB  00 01 00
      29DE  00 00 02
      29E1  00 01
2006  29E3  FF FF 80  DIV2.SEQ WORD      0FFFF,8000,1,8000,3,1,5,1,06,2
      29E6  00 00 01
      29E9  80 00 00
      29EC  03 00 01
      29EF  00 05 00
      29F2  01 00 06
      29F5  00 02
2007                  *
2008
2009                  *
2010                  ************************************************************************
2011                  *
2012                  *  TEST CARRY INPUT;
2013                  *
2014                  *  PROCEDURE CARRY.INP;
2015                  *    BEGIN SUB.ID := 'CARRY';
2016                  *      LOADREG(0,FFFFH); LOADREQ(1,0);
2017                  *      MIR := (1 - + AB 0 0 1 - - - 0 0 - ALU);
2018                  *      IF CPUBUS <> FFFFH THEN PRINTBUS END;
2019                  *      MIR.C := 1
2020                  *      IF CPUBUS <> 0 THEN PRINTBUS END;
2021                  *      SINGLE STEP;   (* TO GET CARRY BIT LOADED *)
2022                  *      MIR.C := C;
2023                  *      IF CPUBUS <> 0 THEN PRINTBUS END
2024                  *      SINGLE STEP;
2025                  *      MIR.C := -C;
2026                  *      IF CPUBUS <> FFFFH THEN PRINTBUS END;
2027                  *      SINGLE STEP;
2028                  *      IF CPUBUS <> 0 THEN PRINTBUS END;
2029                  *    END;
2030                  *
2031  29F7  CE 2A 68  CARRY.INP LDX       =CARR.IN.TXT
2032  29FA  DF 0D             STX       SUB.ID
2033  29FC  BD C1 58          JSR       EDMIR
2034  29FF  5F                CLR       B
2035  2A00  CE FF FF          LDX       =0FFFF
2036  2A03  BD C2 AA          JSR       LOADREG    R0 := FFFF;
2037  2A06  5C                INC       B
2038  2A07  CE 00 00          LDX       =0
2039  2A0A  BD C2 AA          JSR       LOADREG    R1 := 0
2040  2A0D  CE 25 FD          LDX       =FCT.MIR
2041  2A10  BD C1 1D          JSR       LDMIR
2042  2A13  CE FF FF          LDX       =0FFFF
2043  2A16  BC 3C 0E          CPX       CPU1
2044  2A19  27 03             BEQ       ENDIF1CI
```

```
2045  2A1B  BD E1 AA           JSR       PRINTBUS   (B,A)=ACTUAL; X=DESIRED**********
2046  2A1E  86 80     ENDIF1CI  LDAA      =10000000L MIR.C := 1
2047  2A20  B7 3C 09           STAA      MIR3
2048  2A23  CE 00 00           LDX       =0
2049  2A26  BC 3C 0E           CPX       CPU1
2050  2A29  27 03              BEQ       ENDIF2CI
2051  2A2B  BD E1 AA           JSR       PRINTBUS   (B,A)=ACTUAL; X=DESIRED  ********
2052  2A2E  B7 3C 15  ENDIF2CI  STAA      SSP        SINGLE STEP
2053  2A31  86 80              LDAA      =10000000L MIR.C := C
2054  2A33  B7 3C 09           STAA      MIR3
2055  2A36  CE 00 00           LDX       =0
2056  2A39  BC 3C 0E           CPX       CPU1
2057  2A3C  27 03              BEQ       ENDIF3CI
2058  2A3E  BD E1 AA           JSR       PRINTBUS   (B,A)=ACTUAL; X=DESIRED *********
2059  2A41  B7 3C 15  ENDIF3CI  STAA      SSP        SINGLE STEP
2060  2A44  86 A0              LDAA      =10100000L MIR.C := -C
2061  2A46  B7 3C 09           STAA      MIR3
2062  2A49  CE FF FF           LDX       =0FFFF
2063  2A4C  BC 3C 0E           CPX       CPU1
2064  2A4F  27 03              BEQ       ENDIF4CI
2065  2A51  BD E1 AA           JSR       PRINTBUS   (B,A)=ACTUAL; X=DESIRED *********
2066  2A54  B7 3C 15  ENDIF4CI  STAA      SSP        SINGLE STEP
2067  2A57  CE 00 00           LDX       =0
2068  2A5A  BC 3C 0E           CPX       CPU1
2069  2A5D  27 03              BEQ       ENDIF5CI
2070  2A5F  BD E1 AA           JSR       PRINTBUS   (B,A)=ACTUAL; X=DESIRED *********
2071
2072  2A62  BD C1 5F  ENDIF5CI  JSR       EPMIR
2073  2A65  7E 20 4D           JMP       ENDTEST    MAKES RTS
2074                  *
2075  2A68  43 41 52  CARR.IN.TXT BYTE    'CARRY.IN',0
      2A6B  52 59 2E
      2A6E  49 4E 00
2076                  *
2077                  ***************************************************************
2078                  *
2079                  *  STACK TEST
2080                  *
2081                  *  PROCEDURE STACK(X);
2082                  *    VAR HELP1;
2083                  *    BEGIN SUB.ID := 'STACK';  HELP1 := X;
2084                  *      FOR B := 0 TO 15 DO
2085                  *        MIR := (1 - OR DZ 0 0 0 - - S 0 ALU PANEL); (*PUSH*)
2086                  *        X := HELP1;
2087                  *        FOR A := 0 TO 15 DO
2088                  *          CPUBUS := X; X.L.ROTATE; SINGLE STEP;
2089                  *        END;
2090                  *        MIR := (1 - OR DZ 0 0 0 - - S 0 -  ALU); (*POP*)
2091                  *        FOR A := 0 TO 15 DO
2092                  *          X.R.ROTATE;
2093                  *          IF CPUBUS<>X THEN WRITELN(A); PRINTBUS END;
2094                  *          SINGLE STEP;
2095                  *        END
2096                  *        R.ROTATE(HELP1);
2097                  *      END
2098                  *    END;
2099                  *
```

```
2100  2A71  CE FF FE  STACK0    LDX       =0FFFE     DEFAULT VALUE
2101  2A74  24 02               BCC       STACK      IS SET BY MENU IF X.WORD WAS SPECIFIED
2102  2A76  DE 0F               LDX       X.WORD
2103
2104  2A78  DF 2E     STACK     STX       HELP1
2105  2A7A  CE 2A CF             LDX       =STACK.TEXT
2106  2A7D  DF 0D               STX       SUB.ID
2107  2A7F  5F                  CLR       B
2108  2A80  BD C2 90  FOR1STK   JSR       RESET
2109  2A83  CE 26 1B             LDX      =PUSH.MIR
2110  2A86  BD C1 1D             JSR       LDMIR
2111  2A89  BD C1 58             JSR       EDMIR
2112  2A8C  DE 2E               LDX       HELP1
2113  2A8E  4F                  CLR       A
2114  2A8F  FF 3C 0E  FOR2STK   STX       CPU1
2115  2A92  B7 3C 15             STAA      SSP        SINGLE STEP
2116  2A95  BD 20 8D             JSR       X.L.ROTATE
2117  2A98  4C                  INC       A
2118  2A99  81 10               CMPA      =10
2119  2A9B  2D F2               BLT       FOR2STK
2120
2121  2A9D  BD C2 59             JSR       PUSHX
2122  2AA0  CE 26 20             LDX       =POP.MIR
2123  2AA3  BD C1 1D             JSR       LDMIR
2124  2AA6  BD C2 70             JSR       PULLX
2125  2AA9  4F                  CLR       A
2126  2AAA  BD 20 A3  FOR3STK   JSR       X.R.ROTATE
2127  2AAD  BC 3C 0E             CPX       CPU1
2128  2AB0  27 06               BEQ       ENDIF1STK
2129  2AB2  BD 20 87             JSR       WRITELN.A  WRITE(STK COUNTER)**********
2130  2AB5  BD E1 AA             JSR       PRINTBUS   (B,A)=ACTUAL; X=DESIRED
2131  2AB8  B7 3C 15  ENDIF1STK STAA       SSP        SINGLE STEP
2132  2ABB  4C                  INC       A
2133  2ABC  81 10               CMPA      =10
2134  2ABE  2D EA               BLT       FOR3STK
2135
2136  2AC0  DE 2E               LDX       HELP1
2137  2AC2  BD 20 A3             JSR       X.R.ROTATE
2138  2AC5  DF 2E               STX       HELP1
2139  2AC7  5C                  INC       B
2140  2AC8  C1 10               CMPB      =10
2141  2ACA  2D B4               BLT       FOR1STK
2142
2143  2ACC  7E 20 4D             JMP       END.TEST   MAKES RTN
2144                    *
2145  2ACF  53 54 41  STACK.TEXT BYTE      'STACK',0
      2AD2  43 4B 00
2146                    ******************************************
2147                    *
2148                    *  PROCEDURE STATUS.BITS;
2149                    *   FORMAT OF THE COMMAND STRING:
2150                    *     WORD->R0;  WORD->R1; BYTE->MASK; BYTE->JUMP/NOJUMP
2151                    *    VAR HELP1 : POINTER TO COMMAND.STRING;
2152                    *    BEGIN SUB.ID := 'STAT';
2153                    *      HELP1 := ↑COMMAND.STRING;
2154                    *      REPEAT LOADREG(R0, HELP1↑);
2155                    *        INC(HELP1,2); LOADREG(R1, HELP1↑);
```

```
2156                    *       SETUPC(0); (* SET MICRO.PC TO A DEFINED VALUE *)
2157                    *       MIR := (0 10 ...  JMP); (* JUMP TO U.ADR 10*)
2158                    *       INC(HELP1,2); MIR.CONDCODE := HELP1↑;
2159                    *       SINGLE STEP;
2160                    *       (* THE ADDRESS SHOULD BE EITHER 2 ( NO JUMP )
2161                    *          OR 10 IF THE JUMP WAS EXECUTED *)
2162                    *       IF (U.ADR=10) XOR (HELP1↑) THEN S.W.I(HELP1↑, HELP1) END;
2163                    *     UNTIL HELP1=END.OF.STREAM;
2164                    *     EXX(PUSH.MIR);
2165                    *     MIR := (0 10 -E JMP); (* JUMP ON NONEMPTY STACK *); SSP;
2166                    *     IF U.ADR<>10 THEN S.W.I(.., U.ADR) END;
2167                    *     RESET; (* CLEARS THE STACK *);
2168                    *     EDMIR; SSP; (* THERE'S STILL THE LAST JUMP ON NONEMPTY STACK *)
2169                    *     IF U.ADR<>1 THEN S.W.I(..,U.ADR) END;
2170                    *   END STATUS.BITS;
2171                    *
2172  2AD5  CE 2B 8D    STATUS    LDX       =STAT.TXT
2173  2AD8  DF 0D               STX       SUB.ID
2174  2ADA  CE 2B 96            LDX       =STAT.STRNG-2
2175  2ADD  DF 2E               STX       HELP1
2176  2ADF  BD C1 58            JSR       EDMIR
2177  2AE2  BD 28 8B    STAT.RPT  JSR       INC.LOAD
2178  2AE5  5F                  CLR       B
2179  2AE6  BD C2 AA            JSR       LOAD.REG  R0 := HELP1↑;
2180  2AE9  5C                  INC       B
2181  2AEA  BD 28 8B            JSR       INC.LOAD
2182  2AED  BD C2 AA            JSR       LOAD.REG  R1 := HELP1↑;
2183  2AF0  CE 00 00            LDX       =0
2184  2AF3  BD C1 C2            JSR       STUPC    U.PC := 0
2185  2AF6  CE 25 FD            LDX       =FCT.MIR  BUS := R0 + R1
2186  2AF9  BD C1 17            JSR       EXX
2187  2AFC  CE 26 25            LDX       =U.JMP.MIR
2188  2AFF  BD C1 1D            JSR       LDMIR
2189                    *         SET THE CONDITION BITS ON THE RIGHT POSITION
2190  2B02  86 04               LDAA      =4
2191  2B04  97 30               STAA      HELP2     COUNTER FOR SHIFT
2192  2B06  DE 2E               LDX       HELP1      GET THE MASK FROM THE STRING
2193  2B08  08                  INX
2194  2B09  08                  INX
2195  2B0A  DF 2E               STX       HELP1      HELP1 POINTS TO THE BYTE WITH THE MASK INFO
2196  2B0C  A6 00               LDAA      0,X        LOWER PART OF THE MASK
2197  2B0E  5F                  CLR       B         UPPER PART
2198  2B0F  48          ST.SH.LP  ASL       A
2199  2B10  59                  ROL       B
2200  2B11  7A 00 30            DEC       HELP2     DOESN'T CHANGE CARRY
2201  2B14  26 F9               BNE       ST.SH.LP
2202  2B16  8A 07               ORAA      =7        MIR.PC := JMP
2203  2B18  B7 3C 0A            STAA      MIR2      SET THE BITS INTO THE MIR
2204  2B1B  F7 3C 09            STAB      MIR3
2205  2B1E  B7 3C 15            STAA      SSP       SINGLE STEP
2206                    * DEAR READER OF THIS PROGRAM
2207                    * I KNOW THAT THE FOLLOWING PART OF THE PROGRAM
2208                    * IS VERY HARD TO READ, BUT BELIEVE IT WORKS
2209                    * IT COMPUTES    IF (U.ADR<>10) XOR HELP1↑ THEN
2210  2B21  5F                  CLR       B         B := FALSE - ASSUME JUMP CONDITION
2211  2B22  B6 3C 11            LDAA      UAR0      ONLY LOWER BYTE IS OF INTEREST
2212  2B25  8B FC               ADDA      =0FC      IF U.ADR=1 THEN NOCARRY; IF U.ADR=10 THEN CARRY
```

```
2213  2B27  C9 00            ADCB    =0        IF U.ADR=1 THEN B=0 ELSE B=1
2214  2B29  DE 2E            LDX     HELP1
2215  2B2B  E8 01            EORB    1,X       X POINTS TO THE MASK
2216  2B2D  27 08            BEQ     STAT.OK
2217  2B2F  DE 2E            LDX     HELP1     PREPARE FOR SWI
2218  2B31  E6 00            LDAB    0,X       GET THE LAST MASK
2219  2B33  B6 3C 11         LDAA    UAR0
2220  2B36  3F               SWI               B=MASK, A=UAR0; X=ADR IN THE STRING; *************
2221  2B37  DE 2E     STAT.OK LDX     HELP1
2222  2B39  8C 2B F0         CPX     =ST.END.STR
2223  2B3C  26 A4            BNE     STAT.RPT
2224                *
2225                *         NOW TEST THE STACK EMPTY BIT
2226  2B3E  CE 00 00          LDX     =0
2227  2B41  BD C1 C2          JSR     STUPC
2228  2B44  CE 2B 92          LDX     =STAT.TXT.1
2229  2B47  DF 0D             STX     SUB.ID
2230  2B49  CE 26 1B          LDX     =PUSH.MIR
2231  2B4C  BD C1 17          JSR     EXX       EXECUTE PUSH STACK
2232  2B4F  86 60             LDAA    =60       TO LOAD THE STACK EMPTY BIT INTO THE REGISTER
2233  2B51  B7 3C 0B          STAA    MIR1        WE HAVE TO EXECUTE ANY NON STACK INSTRUCTION
2234  2B54  B7 3C 15          STAA    SSP
2235  2B57  CE 26 25          LDX     =U.JMP.MIR  THERE IS ALREADY THE STACK BIT SET
2236  2B5A  BD C1 1D          JSR     LDMIR
2237  2B5D  FE 3C 10          LDX     UAR1
2238  2B60  8C 00 03          CPX     =3
2239  2B63  27 01             BEQ     STAT.OK.1
2240  2B65  3F                SWI               X=UADR **********************************
2241  2B66  BD C2 90  STAT.OK.1 JSR    RESET     IT CLEARS THE STACK
2242  2B69  BD C1 58          JSR     EDMIR     RESET CLEARED THE EDMIR
2243  2B6C  CE 00 00          LDX     =0
2244  2B6F  BD C1 C2          JSR     STUPC     JUMP TO ZERO
2245  2B72  CE 25 E4          LDX     =CLRR0.MIR   EXECUTE ANY INSTRUCTION TO LOAD THE COND REG
2246  2B75  BD C1 17          JSR     EXX
2247  2B78  CE 26 25          LDX     =U.JMP.MIR
2248  2B7B  BD C1 1D          JSR     LDMIR
2249  2B7E  B7 3C 15          STAA    SSP
2250  2B81  FE 3C 10          LDX     UAR1
2251  2B84  8C 00 10          CPX     =10
2252  2B87  27 01             BEQ     STAT.OK.2
2253  2B89  3F                SWI               X=U.ADR**********************************
2254  2B8A  7E 20 4D  STAT.OK.2 JMP    END.TEST
2255                *
2256  2B8D  43 4F 4E  STAT.TXT  BYTE    'COND',0
      2B90  44 00
2257  2B92  43 4F 4E  STAT.TXT1 BYTE    'CONDS',0
      2B95  44 53 00
2258                ST.JUMP   EQU     1
2259                ST.NOJUMP EQU     0
2260                *
2261  2B98  00 80 00  STAT.STRNG WORD   0080,0000 HALFBIT
      2B9B  00
2262  2B9C  01 01             BYTE    01,ST.JUMP
2263  2B9E  00 00 00          WORD    0000,0000
      2BA1  00
2264  2BA2  01 00             BYTE    01,ST.NOJUMP
2265                * SIGN
```

```
2266  2BA4  00 00 00          WORD      0000,0000
      2BA7  00
2267  2BA8  02 00             BYTE      02,ST.NO.JUMP
2268  2BAA  80 00 00          WORD      8000,0000
      2BAD  00
2269  2BAE  02 01             BYTE      02,ST.JUMP
2270  2BB0  80 00 80          WORD      8000,8000 SIGN WITH OVERFLOW
      2BB3  00
2271  2BB4  02 01             BYTE      02,ST.JUMP
2272                        * FIRST
2273  2BB6  00 00 00          WORD      0000,0000
      2BB9  00
2274  2BBA  04 00             BYTE      04,ST.NOJUMP
2275  2BBC  80 00 00          WORD      8000,0000
      2BBF  00
2276  2BC0  04 01             BYTE      04,ST.JUMP
2277                        * OVERFLOW
2278  2BC2  00 00 00          WORD      0000,0000
      2BC5  00
2279  2BC6  08 00             BYTE      08,ST.NOJUMP
2280  2BC8  88 88 88          WORD      8888,8888
      2BCB  88
2281  2BCC  08 01             BYTE      08,ST.JUMP
2282                        * ZERO
2283  2BCE  00 00 00          WORD      0000,0000
      2BD1  00
2284  2BD2  10 01             BYTE      10,ST.JUMP
2285  2BD4  00 01 00          WORD      0001,0000
      2BD7  00
2286  2BD8  10 00             BYTE      10,ST.NOJUMP
2287                        * CARRY
2288  2BDA  00 00 00          WORD      0000,0000
      2BDD  00
2289  2BDE  20 00             BYTE      20,ST.NOJUMP
2290  2BE0  FF FF 00          WORD      0FFFF,1
      2BE3  01
2291  2BE4  20 01             BYTE      20,ST.JUMP
2292                        * INVERSE
2293  2BE6  00 00 00          WORD      0000,0000
      2BE9  00
2294  2BEA  00 00             BYTE      00,ST.NOJUMP
2295  2BEC  00 00 00          WORD      0000,0000
      2BEF  00
2296  2BF0  80 01             BYTE      80,ST.JUMP
2297                        ST.END.STR EQU      *-2
2298                        *
2299                        ************************************************************
2300                        *
2301                        *  PROCEDURE LOOPCHAIN;
2302                        *    BEGIN LOOP CHAIN(X.WORD) END END;
2303                        *
2304  2BF2  CE 2C A4        LOOPCHAIN LDX        =ALU.CHAIN
2305  2BF5  BD 20 15                  JSR        CHAIN
2306  2BF8  20 F8                     BRA        LOOPCHAIN
2307                        *
2308                        ************************************************************
2309                        *
```

```
2310                    *         ENTRY FOR ALU SUBMONITOR
2311                    *
2312                    ************************************************************
2313                    *
2314  2BFA  CE 2C 05    ALU       LDX       =ALU.TEXT
2315  2BFD  DF 0B               STX       MAIN.ID
2316  2BFF  CE 2C 09            LDX       =ALU.MENU
2317  2C02  7E C3 91            JMP       COMMANDS
2318                    *
2319                    ************************************************************
2320                    *
2321  2C05  41 4C 55    ALU.TEXT  BYTE      'ALU',0
      2C08  00
2322                    *
2323  2C09  0E          ALU.MENU  BYTE      14Z
2324  2C0A  4F 20 4F            BYTE      'O OUTP',0
      2C0D  55 54 50
      2C10  00
2325  2C11  26 2A               WORD      OUT2901
2326  2C13  49 20 49            BYTE      'I INP',0
      2C16  4E 50 00
2327  2C19  26 72               WORD      INP2901
2328  2C1B  48 20 53            BYTE      'H SHIFTER',0
      2C1E  48 49 46
      2C21  54 45 52
      2C24  00
2329  2C25  26 AA               WORD      SHIFTR0
2330  2C27  4D 20 4D            BYTE      'M MASK',0
      2C2A  41 53 4B
      2C2D  00
2331  2C2E  27 0E               WORD      MASK
2332  2C30  52 20 52            BYTE      'R REG ADR',0
      2C33  45 47 20
      2C36  41 44 52
      2C39  00
2333  2C3A  27 58               WORD      REG.ADR
2334  2C3C  55 20 53            BYTE      'U SOURCE',0
      2C3F  4F 55 52
      2C42  43 45 00
2335  2C45  27 A5               WORD      SOURCE
2336  2C47  50 20 4F            BYTE      'P OPERATIONS',0
      2C4A  50 45 52
      2C4D  41 54 49
      2C50  4F 4E 53
      2C53  00
2337  2C54  28 1D               WORD      FUNCTIONS
2338  2C56  54 20 53            BYTE      'T SH.DEST',0
      2C59  48 2E 44
      2C5C  45 53 54
      2C5F  00
2339  2C60  29 27               WORD      SH.DEST
2340  2C62  4C 20 4D            BYTE      'L MULT',0
      2C65  55 4C 54
      2C68  00
2341  2C69  29 64               WORD      MULT
2342  2C6B  56 20 44            BYTE      'V DIV',0
      2C6E  49 56 00
```

```
2343  2C71  29 A4              WORD      DIVISION
2344  2C73  59 20 43           BYTE      'Y CARRY',0
      2C76  41 52 52
      2C79  59 00
2345  2C7B  29 F7              WORD      CARRY.INP
2346  2C7D  53 20 53           BYTE      'S STACK',0
      2C80  54 41 43
      2C83  4B 00
2347  2C85  2A 71              WORD      STACK0
2348  2C87  4E 20 43           BYTE      'N CONDITION',0
      2C8A  4F 4E 44
      2C8D  49 54 49
      2C90  4F 4E 00
2349  2C93  2A D5              WORD     STATUS
2350  2C95  58 20 45           BYTE     'X EXEC CHAIN',0
      2C98  58 45 43
      2C9B  20 43 48
      2C9E  41 49 4E
      2CA1  00
2351  2CA2  2B F2              WORD      LOOPCHAIN
2352                    *
2353                    ******************************************************
2354                    *
2355  2CA4  10          ALU.CHAIN BYTE      ALU.CH0.LEN
2356  2CA5  20 4A 2C            WORD      SET.MAIN.ID,ALU.TEXT
      2CA8  05
2357  2CA9  26 2A 00            WORD      OUT2901,0
      2CAC  00
2358  2CAD  26 72 00            WORD      INP2901,0
      2CB0  00
2359  2CB1  26 B1 00            WORD      SHIFTER,1
      2CB4  01
2360  2CB5  26 B1 FF            WORD      SHIFTER,0FFFE
      2CB8  FE
2361  2CB9  27 0E 00            WORD      MASK,0
      2CBC  00
2362  2CBD  27 58 00            WORD      REG.ADR,0
      2CC0  00
2363  2CC1  27 A5 00            WORD      SOURCE,0
      2CC4  00
2364  2CC5  28 1D 00            WORD      FUNCTIONS,0
      2CC8  00
2365  2CC9  29 27 00            WORD      SH.DEST,0
      2CCC  00
2366  2CCD  29 64 00            WORD      MULT,0
      2CD0  00
2367  2CD1  29 A4 00            WORD      DIVISION,0
      2CD4  00
2368  2CD5  29 F7 00            WORD      CARRY.INP,0
      2CD8  00
2369  2CD9  2A 78 00            WORD      STACK,1
      2CDC  01
2370  2CDD  2A 78 FF            WORD      STACK,0FFFE
      2CE0  FE
2371  2CE1  2A D5 00            WORD      STATUS,0
      2CE4  00
2372                    ALU.CH0.LEN EQU     *-1-ALU.CHAIN/4
```

```
2373  2CE5                      END         ALU
```

```
2375                    ********************************************************
2376  2CE5                      BEGIN     STORE.PORT
2377                            DEF       STOR.PORT,STOR.CHAIN,STOR.CH.LN
2378                            USE       END.TEST,CHAIN,SET.MAIN.ID,WRITELN.A
2379                            USE       XMR5,XMW3
2380                    *
2381                    *       DATA PORT AND MAIN MEMORY TEST PROGRAMS
2382                    *
2383                    *       JIRKA HOPPE
2384                    *       FARRELL OSTLER
2385                    *       23 SEPT 80
2386                    *
2387                    ********************************************************
2388                    *
2389                    *  EXTENSIONS TO HANDLE EXTENDED MEMORY
2390                    *
2391                    *  FAST READ IN THE MEMORY BANK 1
2392  2CE5  86 01       RMMS1     LDAA      =1
2393  2CE7  B7 3C 0F             STAA      CPU0
2394  2CEA  86 DD                LDAA      =0DD      SRC=PANEL; DST=MAR1
2395  2CEC  B7 3C 0C             STAA      MIR0
2396  2CEF  B7 3C 15             STAA      SSP
2397  2CF2  7E C2 11             JMP       RMMS      USE THE OLD GOOD ROUTINE
2398                    *
2399                    *  FAST WRITE IN THE MEMORY BANK 1
2400  2CF5  36          WMMS1     PSH       A
2401  2CF6  86 01                LDAA      =1
2402  2CF8  B7 3C 0F             STAA      CPU0
2403  2CFB  86 DD                LDAA      =0DD      SRC=PANEL; DST=MAR1
2404  2CFD  B7 3C 0C             STAA      MIR0
2405  2D00  B7 3C 15             STAA      SSP
2406  2D03  32                   PUL       A
2407  2D04  7E C2 3D             JMP       WMMS      USE THE OLD GOOD ROUTINE
2408                    *
2409                    ********************************************
2410                    *
2411                    *
2412                    *       READ LOOP TEST
2413                    *
2414                    *       COMMAND FORM:   ADR R
2415                    *
2416                    *           READS DATA FROM ADR CONTINUOUSLY.
2417                    *           ADR IS FOUR HEX DIGITS
2418                    *
2419                    *
2420                    *       PROCEDURE READ LOOP(X.WORD);
2421                    *       BEGIN
2422                    *         X:=X.WORD; (*X GETS ADR TO BE READ*)
2423                    *         EDMIR
2424                    *         RMM;       (*READ MAIN MEMORY LOCATION AT X*)
2425                    *         LOOP
2426                    *           RMMS;    (*READ MAIN MEMORY SHORT FORM*)
2427                    *           BREAK(A);
2428                    *           IF A<>0 THEN EXIT;  (*EXIT IF ANY KEY WAS TYPED*)
2429                    *         END (*LOOP*)
2430                    *       END;
```

```
2431  2D07  DE 0F     READLOOP  LDX       X.WORD
2432  2D09  BD C1 58             JSR       EDMIR
2433  2D0C  BD C1 FE             JSR       RMM
2434                  LOOP      EQU       *
2435  2D0F  BD C2 11  XMR1      JSR       RMMS
2436  2D12  BD C0 46             JSR       BREAK
2437  2D15  4D                   TST       A
2438  2D16  27 F7                BEQ       LOOP
2439  2D18  39                   RTS
2440                  *
2441                  ************************************************************************
2442                  *
2443                  *         WRITE LOOP
2444                  *
2445                  *         FORM OF COMMAND:  ADR W VALUE
2446                  *         ADR AND VALUE ARE EACH FOUR HEX DIGITS
2447                  *         VALUE IS CONTINUOUSLY WRITTEN INTO ADR
2448                  *
2449                  *         PROCEDURE WRLOOP(X.WORD);
2450                  *         BEGIN
2451                  *         X:=X.WORD;GETWORD(X.WORD;
2452                  *         B:=X.WORD DIV 256; A:=X.WORD MOD 256;   (*B,A := VALUE*)
2453                  *         WMM  (*WRITE B,A INTO ADDRESS X*)
2454                  *         LOOP
2455                  *           WMMS;  (*WRITE, SHORT FORM*)
2456                  *           PUSH(A);
2457                  *           BREAK(A);
2458                  *           IF A<>0 THEN PULL(A); EXIT; ENDIF;
2459                  *           PULL(A);
2460                  *         END (*LOOP*);
2461                  *
2462  2D19  DE 0F     WRLOOP    LDX       X.WORD
2463  2D1B  BD C2 FE            JSR       GET.WORD
2464  2D1E  D6 0F               LDAB      X.WORD
2465  2D20  96 10               LDAA      X.WORD+1
2466  2D22  BD C1 58            JSR       EDMIR
2467  2D25  BD C2 28            JSR       WMM
2468                  LABELW    EQU       *
2469  2D28  BD C2 3D  XMW1      JSR       WMMS
2470  2D2B  36                  PSH       A
2471  2D2C  BD C0 46            JSR       BREAK
2472  2D2F  4D                  TST       A
2473  2D30  32                  PUL       A
2474  2D31  27 F5               BEQ       LABELW
2475  2D33  39                  RTS
2476                  *
2477                  *
2478                  ************************************************************************
2479                  *
2480                  *         BOTH READ AND WRITE LOOP
2481                  *
2482                  *         FORMAT:     ADR B VALUE
2483                  *         RESULT:     CONTINUOUSLY WRITES VALUE INTO ADR THEN READS IT
2484                  *
2485                  *         PROCEDURE READ.WRITE(X.WORD);
2486                  *         BEGIN
2487                  *         X:=X.WORD;
```

```
2488                    *         GETWORD;  EDMIR;
2489                    *         LOOP
2490                    *           B:=X.WORD DIV 256; A:=X.WORD MOD 256;
2491                    *           WMM;     (*MEMORY(X) := B,A *)
2492                    *           RMM;     (* B,A := MEMORY(X) *)
2493                    *           BREAK(A);
2494                    *           IF AS<>0 THEN EXIT;
2495                    *         END;  (*LOOP*)
2496                    *         END; (*READ.WRITE*)
2497                    *
2498  2D34  DE 0F       RD.WRITE  LDX       X.WORD
2499  2D36  BD C2 FE              JSR       GETWORD
2500  2D39  BD C1 58              JSR       EDMIR
2501  2D3C  BD C1 FE              JSR       RMM       JUST TO SET A PROPER MIR1-MIR4
2502  2D3F  D6 0F       RWLOOP    LDAB      X.WORD
2503  2D41  96 10                 LDAA      X.WORD+1
2504  2D43  BD C2 28    XMW2      JSR       WMM
2505  2D46  BD C1 FE    XMR2      JSR       RMM
2506  2D49  BD C0 46              JSR       BREAK
2507  2D4C  4D                    TST       A
2508  2D4D  27 F0                 BEQ       RWLOOP
2509  2D4F  39                    RTS
2510                    *
2511                    *
2512                    *
2513                    ************************************************************
2514                    *
2515                    *         ADDRESS CHANGE OF ALL BITS
2516                    *
2517                    *         FORMAT:   C
2518                    *
2519                    *         PROCEDURE ADR.CHNG;
2520                    *           BEGIN
2521                    *           EDMIR;
2522                    *           X:=0; RMM;   (*INITIALIZE LOOP*)
2523                    *           LOOP
2524                    *             X:=FFFF; RMMS;
2525                    *             X:=0;    RMMS;
2526                    *             BREAK(A);
2527                    *             IF A<>0 THEN EXIT;
2528                    *           END;  (*LOOP*)
2529                    *           END;
2530                    *
2531  2D50  BD C1 58    ADR.CHNG  JSR       EDMIR
2532  2D53  CE 00 00              LDX       =0
2533  2D56  BD C1 FE              JSR       RMM
2534  2D59  CE FF FF    L1        LDX       =0FFFF
2535  2D5C  BD C2 11    XMR3      JSR       RMMS
2536  2D5F  CE 00 00              LDX       =0
2537  2D62  BD C2 11    XMR4      JSR       RMMS
2538  2D65  BD C0 46              JSR       BREAK
2539  2D68  4D                    TST       A
2540  2D69  27 EE                 BEQ       L1
2541  2D6B  39                    RTS
2542                    *
2543                    ************************************************************
2544                    INSPECT     EQU   *
```

```
2545  2D6C                        BEGIN INSPECT
2546                              DEF   XMR5,XMW3
2547                  VALUE       EQU   HELP2
2548                  *
2549                  *           FORMAT
2550                  *           ADDRESS 'I' VALUE [ ':' NEWVALUE ] [',']
2551                  *           ADDRESS  = UP TO 4 HEX DIGIT
2552                  *           NEWVALUE = UP TO 4 HEX DIGIT
2553                  *
2554                  *           PROCEDURE INSPECT;
2555                  *             BEGIN X := X.WORD;
2556                  *               LOOP
2557                  *                 HELP1 := X; (A,B) := RMM(X);
2558                  *                 VALUE := (A,B); OUT.WORD(VALUE);
2559                  *                 R.CH(A);
2560                  *                 IF A =':' THEN
2561                  *                   GET.WORD; (A,B) := X.WORD; X := HELP1; WMM(A,B,X);
2562                  *                 END;
2563                  *                 IF A = BACKSLASH
2564                  *                 THEN
2565                  *                   DEC(X);
2566                  *                 ELSIF A = ','
2567                  *                 THEN
2568                  *                   INC(X);
2569                  *                 ELSE
2570                  *                   EXIT
2571                  *                 END;
2572                  *                 HELP1 := X;
2573                  *                 CRLF; OUT.WORD(HELP1); WRITE(' ')
2574                  *               END
2575                  *             END
2576                  *
2577  2D6C  DE 0F                     LDX   X.WORD
2578  2D6E  DF 2E                     STX   HELP1
2579  2D70  BD C1 58                  JSR   EDMIR
2580                  LOOP        EQU   *
2581  2D73  BD C1 FE  XMR5        JSR   RMM
2582  2D76  D7 30                     STAB  VALUE       HIGHER PART
2583  2D78  97 31                     STAA  VALUE+1     LOWER PART
2584  2D7A  CE 00 30                  LDX   =VALUE
2585  2D7D  BD C0 E3                  JSR   OUT.WORD
2586  2D80  BD C0 24                  JSR   R.CH
2587  2D83  81 3A                     CMPA  =':'
2588  2D85  26 0E                     BNE   ENDIF1
2589  2D87  BD C2 FE                  JSR   GET.WORD
2590  2D8A  36                        PSH   A           SAVE DELIMITER
2591  2D8B  D6 0F                     LDAB  X.WORD      HIGHER PART
2592  2D8D  96 10                     LDAA  X.WORD+1    LOWER PART
2593  2D8F  DE 2E                     LDX   HELP1
2594  2D91  BD C2 28  XMW3        JSR   WMM
2595  2D94  32                        PUL   A           GET DELIMITER BACK
2596  2D95  81 2C     ENDIF1      CMPA  =COMMA
2597  2D97  27 08                     BEQ   ENDIF2
2598  2D99  81 5C                     CMPA  =BACK.SLASH
2599  2D9B  27 09                     BEQ   ENDIF3
2600  2D9D  BD C1 5F                  JSR   EPMIR
2601  2DA0  39                        RTS
```

```
2602
2603  2DA1  DE 2E     ENDIF2      LDX   HELP1
2604  2DA3  08                    INX
2605  2DA4  20 03                 BRA   ENDIF
2606  2DA6  DE 2E     ENDIF3      LDX   HELP1
2607  2DA8  09                    DEX
2608  2DA9  DF 2E     ENDIF       STX   HELP1
2609  2DAB  BD C0 78              JSR   W.CRLF
2610  2DAE  BD C2 87              JSR   PRINT.X
2611  2DB1  BD C0 62              JSR   W.1.BLANK
2612  2DB4  20 BD                 BRA   LOOP
2613  2DB6                        END   INSPECT
2614                    ************************************************************
2615                    *
2616                    *  PROCEDURE MAIN.MEM.TEST;
2617                    *    VAR HELP1 = CURRENT MEMORY PATTERN;
2618                    *        HELP2 = INCREMENT;
2619                    *    BEGIN
2620                    *    ENTRY PATTERN: HELP2 :=0     ; HELP1 := X.WORD;
2621                    *    ENTRY RANDOM : HELP2 :=X.WORD; HELP1 := 0;
2622                    *
2623                    *      SUB.[0 := 'MEM'; (B,A):=HELP1;
2624                    *      FOR < := 0 TO 0FFFFH DO WMMS(X, B,A);
2625                    *        (B,A) := (B,A) + HELP2;
2626                    *
2627                    *      POP(HELP1);
2628                    *      FOR < := 0 TO 0FFFFH DO
2629                    *        IF READ MEMORY(X) <> HELP1 THEN
2630                    *         PRINT(ADR); PRINTERR(ACTUAL,EXPECTED)
2631                    *        HELP1 := HELP1+ HELP2;
2632                    *     END MAIN.MEM.TEST;
2633                    *
2634  2DB6  DF 0F     PATTERN0  STX       X.WORD
2635  2DB8  CE 00 00  PATTERN   LDX       =0
2636  2DBB  DF 30               STX       HELP2
2637  2DBD  DE 0F               LDX       X.WORD
2638  2DBF  DF 2E               STX       HELP1
2639  2DC1  20 0B               BRA       MEM.TST
2640  2DC3  DF 0F     RANDOM0   STX       X.WORD
2641  2DC5  DE 0F     RANDOM    LDX       X.WORD
2642  2DC7  DF 30               STX       HELP2
2643  2DC9  CE 00 00            LDX       =0
2644  2DCC  DF 2E               STX       HELP1
2645  2DCE  CE 2E 17  MEM.TST   LDX       =MEM.TXT
2646  2DD1  DF 0D               STX       SUB.ID
2647  2DD3  BD C1 58            JSR       EDMIR
2648  2DD6  BD C1 FE            JSR       RMM       JUST TO SET A NICE MIR1-MIR4
2649  2DD9  CE 00 00            LDX       =0
2650  2DDC  96 2F               LDAA      HELP1+1
2651  2DDE  D6 2E               LDAB      HELP1
2652                    FOR1      EQU       *
2653  2DE0  BD C2 3D  XMW4      JSR       WMMS      FAST WMM
2654  2DE3  9B 31               ADDA      HELP2+1
2655  2DE5  D9 30               ADCB      HELP2
2656  2DE7  08                  INX
2657  2DE8  26 F6   release.TEXT

    84604y
edhj2.BAK            Clil.dsk4.SIL          edfig51.SIL          9release.TEXT
Afirst.LST               **This software is distributedGGG
QSS
***/ and***)...5....

                                                                  ***/****
y
```

```
                    (B,A)=ACTUAL; X=EXPECTED
2670  2E03  DE 32               LDX       HELP3
2671  2E05  96 2F      NOERR    LDAA      HELP1+1
2672  2E07  D6 2E               LDAB      HELP1
2673
2674  2E09  9B 31               ADDA      HELP2+1
2675  2E0B  D9 30               ADCB      HELP2
2676  2E0D  97 2F               STAA      HELP1+1
2677  2E0F  D7 2E               STAB      HELP1
2678  2E11  08                  INX
2679  2E12  26 D9               BNE       FOR2
2680  2E14  7E 20 4D            JMP       END.TEST
2681                    *
2682  2E17  4D 45 4D    MEM.TXT  BYTE      'MEM',0
      2E1A  00
2683                    *
2684                    ****************************************
2685                    *
2686                    *  SET PROPER PROCEDURES FOR A GIVEN MEMORY BANK
2687                    *
2688  2E1B  CE C2 11    SETBNK   LDX       =RMMS     BANK0
2689  2E1E  7D 00 10             TST       X.WORD+1
2690  2E21  27 03                BEQ       SB1
2691  2E23  CE 2C E5             LDX       =RMMS1    BANK1
2692  2E26  FF 2D 10    SB1      STX       XMR1+1
2693  2E29  FF 2D 47             STX       XMR2+1
2694  2E2C  FF 2D 5D             STX       XMR3+1
2695  2E2F  FF 2D 63             STX       XMR4+1
2696  2E32  FF 2D 74             STX       XMR5+1
2697  2E35  FF 2D EE             STX       XMR6+1
2698  2E38  CE C2 3D             LDX       =WMMS     BANK0
2699  2E3B  7D 00 10             TST       X.WORD+1
2700  2E3E  27 03                BEQ       SB2
2701  2E40  CE 2C F5             LDX       =WMMS1    BANK1
2702  2E43  FF 2D 29    SB2      STX       XMW1+1
2703  2E46  FF 2D 44             STX       XMW2+1
2704  2E49  FF 2D 92             STX       XMW3+1
2705  2E4C  FF 2D E1             STX       XMW4+1
2706  2E4F  39                   RTS
2707                    *
2708                    ********************************************************************
2709  2E50  CE 2E EE    LOOPCHAIN LDX      =STOR.CHAIN
2710  2E53  BD 20 15             JSR       CHAIN
2711  2E56  20 F8                BRA       LOOPCHAIN
2712                    ********************************************************************
2713                    *
2714                    *
```

```
2715                        *         ENTRY FOR THE PORT AND STORE SUBMONITOR
2716                        *
2717  2E58  CE 2E 63  STOR.PORT LDX      =S.P.TEXT
2718  2E5B  DF 0B              STX      MAIN.ID
2719  2E5D  CE 2E 6E           LDX      =ST.PT.MENU
2720  2E60  7E C3 91           JMP      COMMANDS
2721                        *
2722                        ****************************************************************
2723                        *
2724  2E63  50 4F 52  S.P.TEXT  BYTE     'PORT.STORE',0
      2E66  54 2E 53
      2E69  54 4F 52
      2E6C  45 00
2725  2E6E  09        ST.PT.MENU BYTE   9
2726  2E6F  52 20 52           BYTE     'R READ LOOP',0
      2E72  45 41 44
      2E75  20 4C 4F
      2E78  4F 50 00
2727  2E7B  2D 07              WORD     READLOOP
2728  2E7D  57 20 57           BYTE     'W WRITE LOOP',0
      2E80  52 49 54
      2E83  45 20 4C
      2E86  4F 4F 50
      2E89  00
2729  2E8A  2D 19              WORD     WRLOOP
2730  2E8C  51 20 52           BYTE     'Q RD.WR LOOP',0
      2E8F  44 2E 57
      2E92  52 20 4C
      2E95  4F 4F 50
      2E98  00
2731  2E99  2D 34              WORD     RD.WRITE
2732  2E9B  4F 20 4F           BYTE     'O OSCILLATING ADR',0
      2E9E  53 43 49
      2EA1  4C 4C 41
      2EA4  54 49 4E
      2EA7  47 20 41
      2EAA  44 52 00
2733  2EAD  2D 50              WORD     ADR.CHNG
2734  2EAF  50 20 50           BYTE     'P PATTERN',0
      2EB2  41 54 54
      2EB5  45 52 4E
      2EB8  00
2735  2EB9  2D B8              WORD     PATTERN
2736  2EBB  4E 20 52           BYTE     'N RANDOM',0
      2EBE  41 4E 44
      2EC1  4F 4D 00
2737  2EC4  2D C5              WORD     RANDOM
2738  2EC6  49 20 49           BYTE     'I INSPECT',0
      2EC9  4E 53 50
      2ECC  45 43 54
      2ECF  00
2739  2ED0  2D 6C              WORD     INSPECT
2740  2ED2  53 20 53           BYTE     'S SET BANK',0
      2ED5  45 54 20
      2ED8  42 41 4E
      2EDB  4B 00
2741  2EDD  2E 1B              WORD     SETBNK
```

```
2742  2EDF  58 20 45           BYTE      'X EXEC CHAIN',0
      2EE2  58 45 43
      2EE5  20 43 48
      2EE8  41 49 4E
      2EEB  00
2743  2EEC  2E 50              WORD      LOOPCHAIN
2744                  *
2745                  ************************************************************
2746                  *
2747  2EEE  04        STOR.CHAIN BYTE     STOR.CH.LN
2748  2EEF  2D B6 00           WORD      PATTERN0,0
      2EF2  00
2749  2EF3  2D C3 00           WORD      RANDOM0,1
      2EF6  01
2750  2EF7  2D B6 FF           WORD      PATTERN0,0FFFF
      2EFA  FF
2751  2EFB  2D C3 55           WORD      RANDOM0,5531
      2EFE  31
2752                  STOR.CH.LN EQU      *-1-STOR.CHAIN/4
2753                  *
2754  2EFF                     END       STORE.PORT
2755                  *
2756  2EFF                     END       HWTEST
2757
```