```
                ********************************************************************************
                *                                                                              *
                *                          *****                                               *
                *      DIAGNOSTIC PANEL M6802-SOFTWARE       23.09.80                          *
                *                          *****                                               *
                *                                                                              *
                *                 J.HOPPE                                                      *
                *                 I.NOACK                                                      *
                *                 R.OHRAN                                                      *
                *                 W.WINIGER                                                    *
                *                                                                              *
                ********************************************************************************

0000                            BEGIN PANEL

                *               CONSTANT DEFINITIONS
                *
                *               ASCII CHARACTERS
                *
                CTRL.C          EQU   3
                LF              EQU   0A
                CR              EQU   0D
                DC1             EQU   11
                ESC             EQU   1B
                BLANK           EQU   20
                EXCLAM          EQU   21
                AMPERSAND       EQU   26
                LOWER.E         EQU   65
                LOWER.P         EQU   70
                LOWER.U         EQU   75
                RUBOUT          EQU   1B
                QUESTION.M      EQU   3F
                PROMPT          EQU   2A
                COMMA           EQU   2C
                BACK.SLASH      EQU   5C
                SHIFT.MASK      EQU   0DF

                *               I/O ADDRESSES

                HP.STATUS       EQU   4000
                HP.DATA         EQU   4001

                *               LOCAL ENTRYPOINTS OF NESTED BLOCKS

                                USE   MONITOR,TYPE.CTRL,U.RAM.LOAD,X.WORD,MAINID,SUBID
                                USE   DUMP,CALCULATE,LOAD,MENU,GETWORD,GETKEY,READ.REG

                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
```

```
59                      SSP         EQU   3C15
60                      UAR1        EQU   3C10
61                      UAR0        EQU   3C11
62
63                      STACK       EQU   7F
64                      RET         EQU   2000        'RETURN'-JUMP FOR PUSHX/PULLX
65
66
67                      *           VARIABLES
68
69  0000  00            C           BYTE  0           DUMMY REGISTERS,
70  0001  00 00         X.SAVE      WORD  0           FOR STRICTLY LOCAL USE ONLY
71                      DATA        SET   *
72
```

```
74                      * * * * * * * * * * * * * * * * * * * * * * * * *
75                      *          P R O M   C 0 0 0                    *
76                      * * * * * * * * * * * * * * * * * * * * * * * * *
77
78  0003                            ORG   0C000
79
80  C000  1B 26 70  READ.CTRL      BYTE  ESC,AMPERSAND,LOWER.P,'0','R',DC1,0
    C003  30 52 11
    C006  00
81  C007  20 45 4F  MESS3           BYTE  ' EOT',0
    C00A  54 00
82  C00C  43 4B 53  MESS1           BYTE  'CKSUM',0
    C00F  55 4D 00
83  C012  41 44 44  MESS2           BYTE  'ADDR: ',0
    C015  52 3A 20
    C018  00
84
85
86                      INIT        EQU   *           INITIALIZE THE ACIA
87  C019                            BEGIN INIT
88  C019  86 03                     LDAA  =3
89  C01B  B7 40 00                  STAA  HP.STATUS
90  C01E  86 12                     LDAA  =12
91  C020  B7 40 00                  STAA  HP.STATUS
92  C023  39                        RTS
93  C024                            END   INIT
```

```
 95                     *          PROCEDURE RCH (VAR A:CHAR);
 96                     *          VAR S : BITSET; (* A-REG *)
 97                     *          BEGIN
 98                     *            REPEAT
 99                     *              S := HP.STATUS
100                     *            UNTIL 0 IN S;
101                     *            A := HP.DATA;
102                     *            IF A = ESC THEN GOTO MONITOR;
103                     *            IF A = QUESTION.MARK
104                     *              MENU; R.CH(A);
105                     *            END;
106                     *          END; (* RCH *)
107
108                     R.CH       EQU   *
109  C024                          BEGIN RCH
110                               DEF   GOT
111                               USE   ESCAPED,COM.ESC
112
113  C024  B6 40 00    REPEAT     LDAA  HP.STATUS
114  C027  47                     ASR   A
115  C028  24 FA       UNTIL      BCC   REPEAT
116  C02A  B6 40 01    GOT        LDAA  HP.DATA
117
118  C02D  81 03       IF1        CMPA  =CTRL.C
119  C02F  26 03                  BNE   IF2
120  C031  7E E2 B3    THEN1      JMP   ESCAPED
121
122  C034  81 3F       IF2        CMPA  =QUESTION.M
123  C036  26 05                  BNE   IF3
124  C038  BD C3 36    THEN2      JSR   MENU
125  C03B  20 E7                  BRA   REPEAT
126
127  C03D  81 1B       IF3        CMPA  =ESC
128  C03F  26 10                  BNE   W.CH          FULL DUPLEX
129
130  C041  8D 1D                  BSR   W.2.BLANK
131  C043  7E C3 95               JMP   COM.ESC
132  C046                         END   RCH
```

```
                        *           PROCEDURE BREAK (VAR A:CHAR);
                        *             VAR S: BITSET; (* A-REG *)
                        *           BEGIN
                        *             S := HP.STATUS;
                        *             IF 0 IN S
                        *             THEN
                        *               A := HP.DATA;
                        *               IF A = ESC THEN GOTO MONITOR END;
                        *               IF A = QUESTION.MARK
                        *               THEN
                        *                 MENU; RCH(A);
                        *               END;
                        *             ELSE
                        *               A := NULL;
                        *             END;
                        *           END BREAK;

                                    DEF   GOT
                        BREAK       EQU   *
C046                                BEGIN BREAK
C046  B6 40 00                      LDAA  HP.STATUS
C049  47                            ASR   A
C04A  24 03                         BCC   RET
C04C  7E C0 2A                      JMP   GOT
C04F  4F              RET           CLR   A
C050  39                            RTS
C051                                END   BREAK


                        *           PROCEDURE WCH (A:CHAR);
                        *           VAR S : BITSET;  (* A-REG *)
                        *           BEGIN
                        *             REPEAT
                        *               S := HP.STATUS
                        *             UNTIL 1 IN S;
                        *             HP.DATA := A;
                        *           END; (* WCH *)

                        W.CH        EQU   *
C051                                BEGIN WCH
C051  36                            PSH   A
C052  B6 40 00        REPEAT       LDAA  HP.STATUS
C055  47                            ASR   A
C056  47                            ASR   A
C057  24 F9           UNTIL         BCC   REPEAT

C059  32                            PUL   A
C05A  B7 40 01                      STAA  HP.DATA
C05D  39                            RTS
C05E                                END   WCH
```

```
188  C05E  8D 02     W.3.BLANK   BSR   W.1.BLANK
189  C060  8D 00     W.2.BLANK   BSR   W.1.BLANK
190
191  C062  36        W.1.BLANK   PSH   A
192  C063  86 20                 LDAA  =BLANK
193  C065  BD C0 51              JSR   W.CH
194  C068  32                    PUL   A
195  C069  39                    RTS
196
197                  *           PROCEDURE WSTRING (VAR X:ADDRESS);
198                  *           VAR A : CHAR;
199                  *           BEGIN
200                  *             A := M[X]; X := X + 1;
201                  *             WHILE A <> NULL DO BEGIN
202                  *               WCH(A);
203                  *               A := M[X];
204                  *               X := X + 1;
205                  *             END;
206                  *           END; (* WSTRING *)
207
208
209                  W.STRING    EQU   *
210  C06A                        BEGIN WSTRING
211  C06A  36                    PSH   A
212  C06B  A6 00     WHILE       LDAA  0,X
213  C06D  27 06                 BEQ   ENDWHILE
214  C06F  BD C0 51              JSR   W.CH
215  C072  08                    INX
216  C073  20 F6                 BRA   WHILE
217  C075  32        ENDWHILE    PUL   A
218  C076  08                    INX
219  C077  39                    RTS
220  C078                        END   WSTRING
221
222
223
224                  *           PROCEDURE CRLF;
225                  *           BEGIN
226                  *             WCH(CR);
227                  *             WCH(LF);
228                  *           END; (* CRLF *)
229
230
231                  W.CRLF      EQU   *
232  C078                        BEGIN CRLF
233  C078  36                    PSH   A
234  C079  86 0D                 LDAA  =CR
235  C07B  BD C0 51              JSR   W.CH
236  C07E  86 0A                 LDAA  =LF
237  C080  BD C0 51              JSR   W.CH
238  C083  32                    PUL   A
239  C084  39                    RTS
240  C085                        END   CRLF
241
```

```
                        *           PROCEDURE INHEX (VAR A:BYTE);
                        *           VAR C : CHAR;  (* A-REG *)
                        *           BEGIN
                        *             REPEAT
                        *               RCH(C);
                        *               A := BIN(C);
                        *               IF CARRY THEN WCH('!');
                        *             UNTIL NOT CARRY;
                        *           END;  (* INHEX *)


                        IN.HEX      EQU   *
C085                                BEGIN INHEX
C085  8D 9D             REPEAT      BSR   R.CH
C087  8D 13                         BSR   BIN
C089  24 04                         BCC   ENDREPEAT
C08B  8D 0A                         BSR   NO
C08D  20 F6                         BRA   REPEAT
                        ENDREPEAT   EQU   *
C08F  39                            RTS
C090                                END   INHEX




                        *           PROCEDURE OUTHEX (A:BYTE);
                        *           VAR C : CHAR;  (* A-REG *)
                        *           BEGIN
                        *             C := HEX(A);
                        *             WCH(C);
                        *           END;  (* OUTHEX *)


                        OUT.HEX     EQU   *
C090                                BEGIN OUTHEX
C090  36                            PSH   A
C091  8D 20                         BSR   HEX
C093  8D BC                         BSR   W.CH
C095  32                            PUL   A
C096  39                            RTS
C097                                END   OUTHEX




C097  36            NO              PSH   A
C098  7E C3 A3                      JMP   NONO
C09B  02                            NOP
```

```
                        BIN     EQU   *          FUNCTION BIN (A:CHAR; CARRY:BOOLEAN) : BYTE;
                        *                        VAR B : BYTE; (* A-REG *)
C09C                    BEGIN BIN                BEGIN
C09C  80 30             SUBA  ='0'                 B := BYTE(A) - BYTE('0');
                        *                          CARRY := B < 0;
C09E  25 12     IF1     BCS   ENDIF1               IF NOT CARRY
                        *                          THEN
C0A0  81 0A     IF2     CMPA  =0A                    IF B >= 10Z
C0A2  25 0D             BCS   ELSE2                  THEN BEGIN
C0A4  84 DF             ANDA  =SHIFT.MASK              B := CAP(B);
C0A6  80 07     THEN2   SUBA  ='A'-':'                 B := B - (BYTE('A')-BYTE(':'));
C0A8  81 10     IF3     CMPA  =10                      IF B >= 16Z
C0AA  25 02             BCS   ELSE3
C0AC  0D        THEN3   SEC                            TEHN CARRY := TRUE
C0AD  39                RTS
C0AE  81 0A     ELSE3   CMPA  =0A                      ELSE CARRY := B < 10Z;
C0B0  39                RTS                          END
C0B1  0C        ELSE2   CLC                          ELSE CARRY := FALSE;
                        ENDIF1  EQU   *            BIN := B;
C0B2  39                RTS                      END; (* BIN *)
C0B3                    END   BIN
```

```
                        *       FUNCTION HEX (A:BYTE) : CHAR;
                        *       BEGIN
                        *         IF A >= 10Z
                        *         THEN HEX := CHAR(A + BYTE('0') + BYTE('0') - BYTE(':'));
                        *         ELSE HEX := CHAR(A + BYTE('0'));
                        *       END; (* HEX *)
```

```
                        HEX     EQU   *
C0B3                    BEGIN HEX
C0B3  81 0A     IF      CMPA  =0A
C0B5  2D 02             BLT   ELSE
C0B7  8B 07     THEN    ADDA  ='A'-':'
C0B9  8B 30     ELSE    ADDA  ='0'
C0BB  39                RTS
C0BC                    END   HEX
```

```
337                     *           PROCEDURE INBYTE (VAR A:BYTE);
338                     *           VAR C : BYTE;
339                     *           BEGIN
340                     *             INHEX(A);
341                     *             C := 16 * A;
342                     *             INHEX(A);
343                     *             A := A + C;
344                     *           END;  (* INBYTE *)
345
346
347                     IN.BYTE     EQU   *
348  C0BC               C0BC       BEGIN INBYTE
349  C0BC  8D C7        C0BC       BSR   IN.HEX
350  C0BE  48           C0BE       ASL   A
351  C0BF  48           C0BF       ASL   A
352  C0C0  48           C0C0       ASL   A
353  C0C1  48           C0C1       ASL   A
354  C0C2  97 00        C0C2       STAA  C
355  C0C4  8D BF        C0C4       BSR   IN.HEX
356  C0C6  9A 00        C0C6       ORAA  C
357  C0C8  39           C0C8       RTS
358  C0C9               C0C9       END   INBYTE
359
360
361
362
363                     *           PROCEDURE OUTBYTE (A:BYTE);
364                     *           VAR B : BYTE;  (* A-REG *)
365                     *           BEGIN
366                     *             B := A MOD 16Z;
367                     *             A := A DIV 16Z;
368                     *             OUTHEX(A);
369                     *             OUTHEX(B);
370                     *           END;  (* OUTBYTE *)
371
372
373                     OUT.BYTE    EQU   *
374  C0C9               C0C9       BEGIN OUTBYTE
375  C0C9  36           C0C9       PSH   A
376  C0CA  36           C0CA       PSH   A
377  C0CB  44           C0CB       LSR   A
378  C0CC  44           C0CC       LSR   A
379  C0CD  44           C0CD       LSR   A
380  C0CE  44           C0CE       LSR   A
381  C0CF  8D BF        C0CF       BSR   OUT.HEX
382  C0D1  32           C0D1       PUL   A  (B)
383  C0D2  84 0F        C0D2       ANDA  =0F
384  C0D4  8D BA        C0D4       BSR   OUT.HEX
385  C0D6  32           C0D6       PUL   A
386  C0D7  39           C0D7       RTS
387  C0D8               C0D8       END   OUTBYTE
388
```

```
390                     *           PROCEDURE INWORD (X:ADDRESS);
391                     *           BEGIN
392                     *             INBYTE(M[X]);
393                     *             INBYTE(M[X+1]);
394                     *           END;  (* INWORD *)
395
396
397                     IN.WORD     EQU   *
398  C0D8               BEGIN INWORD
399  C0D8  36                      PSH   A
400  C0D9  8D E1                   BSR   IN.BYTE
401  C0DB  A7 00                   STAA  0,X
402  C0DD  8D DD                   BSR   IN.BYTE
403  C0DF  A7 01                   STAA  1,X
404  C0E1  32                      PUL   A
405  C0E2  39                      RTS
406  C0E3                          END   INWORD
407
408
409
410
411                     *           PROCEDURE OUTWORD (X:ADDRESS);
412                     *           BEGIN
413                     *             OUTBYTE(M[X]);
414                     *             OUTBYTE(M[X+1]);
415                     *           END;  (* OUTWORD *)
416
417
418                     OUT.WORD    EQU   *
419  C0E3               BEGIN OUTWORD
420  C0E3  36                      PSH   A
421  C0E4  A6 00                   LDAA  0,X
422  C0E6  8D E1                   BSR   OUT.BYTE
423  C0E8  A6 01                   LDAA  1,X
424  C0EA  8D DD                   BSR   OUT.BYTE
425  C0EC  32                      PUL   A
426  C0ED  39                      RTS
427  C0EE                          END   OUTWORD
428
```

```
                        *           PROCEDURE FASTREAD;
                        *           BEGIN
                        *             WCH(ESC);
                        *             WCH(LOWER.E);
                        *           END;  (* FASTREAD *)


                        FAST.READ   EQU   *
C0EE                                BEGIN FASTREAD
C0EE  36                            PSH   A
C0EF  86 1B                         LDAA  =ESC
C0F1  BD C0 51                      JSR   W.CH
C0F4  86 65                         LDAA  =LOWER.E
C0F6  BD C0 51                      JSR   W.CH
C0F9  32                            PUL   A
C0FA  39                            RTS
C0FB                                END   FASTREAD


                        *           PROCEDURE LINEREAD;
                        *           BEGIN
                        *             WSTRING(READCTRL);
                        *           END;

                        LINE.READ   EQU   *
C0FB                                BEGIN LINEREAD
C0FB  BD C2 59                      JSR   PUSHX
C0FE  CE C0 00                      LDX   =READ.CTRL
C101  BD C0 6A                      JSR   W.STRING
C104  BD C2 70                      JSR   PULLX
C107  39                            RTS
C108                                END   LINEREAD
```

```
467  C108  2E 00 07  DGTOM     BYTE  2E,0,7,60,3D,2E,0,7,60,1
     C10B  60 3D 2E
     C10E  00 07 60
     C111  01
468  C112  2E 00 07  DGTOD     BYTE  2E,0,7,60,1D
     C115  60 1D
469
470                  CODE      SET   *
471  C117                      ORG   DATA
472
473  0003  00        STAT0     BYTE  0
474
475                  DATA      SET   *
476  0004                      ORG   CODE
477
478  C117  8D 04     EXX       BSR   LDMIR
479  C119  B7 3C 15            STAA  SSP
480  C11C  39                  RTS
481
482  C11D  36        LDMIR     PSH   A
483  C11E  A6 00               LDAA  0,X
484  C120  B7 3C 08            STAA  MIR4
485  C123  A6 01               LDAA  1,X
486  C125  B7 3C 09            STAA  MIR3
487  C128  A6 02               LDAA  2,X
488  C12A  B7 3C 0A            STAA  MIR2
489  C12D  A6 03               LDAA  3,X
490  C12F  B7 3C 0B            STAA  MIR1
491  C132  A6 04               LDAA  4,X
492  C134  B7 3C 0C            STAA  MIR0
493  C137  08                  INX
494  C138  08                  INX
495  C139  08                  INX
496  C13A  08                  INX
497  C13B  08                  INX
498  C13C  32                  PUL   A
499  C13D  39                  RTS
500
501  C13E  36        D2911     PSH   A
502  C13F  96 03               LDAA  STAT0
503  C141  84 EF               ANDA  =0EF
504  C143  97 03     D29A      STAA  STAT0
505  C145  B7 3C 13            STAA  CON0
506  C148  32                  PUL   A
507  C149  39                  RTS
508
509  C14A  36        UNLCH     PSH   A          UNLATCH MICRO MEMORY ADDRESS REGISTER
510  C14B  96 03               LDAA  STAT0
511  C14D  8A 08               ORAA  =8         BIT3 := 1
512  C14F  20 F2               BRA   D29A
513
514  C151  36        LATCH     PSH   A          LATCH MICRO MEMORY ADDRESS REGISTER
515  C152  96 03               LDAA  STAT0
516  C154  84 F7               ANDA  =0F7       BIT3 := 0
517  C156  20 EB               BRA   D29A
518
```

```
519  C158  36        EDMIR     PSH   A          ENABLE DIAGNOSTIC MIR
520  C159  96 03                LDAA  STAT0
521  C15B  84 7F                ANDA  =07F       BIT7 := 0
522  C15D  20 E4                BRA   D29A
523
524  C15F  36        EPMIR     PSH   A          ENABLE PERSONAL COMPUTER'S MIR
525  C160  96 03                LDAA  STAT0
526  C162  8A 80                ORAA  =80        BIT7 := 1
527  C164  20 DD                BRA   D29A
528
529  C166  36        START     PSH   A          START CPU
530  C167  96 03                LDAA  STAT0
531  C169  8A 40                ORAA  =40        BIT6 := 1
532  C16B  20 D6                BRA   D29A
533
534  C16D  36        STOP      PSH   A          STOP CPU
535  C16E  96 03                LDAA  STAT0
536  C170  84 BF                ANDA  =0BF       BIT6 := 0
537  C172  20 CF                BRA   D29A
538
539  C174  36        E2911     PSH   A
540  C175  96 03                LDAA  STAT0
541  C177  8A 10                ORAA  =10
542  C179  20 C8                BRA   D29A
543
544  C17B  8D C1     WUM       BSR   D2911
545  C17D  FF 3C 10             STX   UAR1
546  C180  8D CF                BSR   LATCH
547  C182  B7 3C 11             STAA  UAR0
548  C185  F7 3C 10             STAB  UAR1
549  C188  B7 3C 14             STAA  MEMPU
550  C18B  8D E7                BSR   E2911
551  C18D  8D BB                BSR   UNLCH
552  C18F  39                   RTS
553
554  C190  8D AC     RUM       BSR   D2911
555  C192  FF 3C 10             STX   UAR1
556  C195  8D 15                BSR   MCLK
557  C197  BD C2 59             JSR   PUSHX
558  C19A  37                   PSH   B
559  C19B  CE 3C 0D             LDX   =MIR0+1
560  C19E  5C                   INC   B
561  C19F  09        R1        DEX
562  C1A0  5A                   DEC   B
563  C1A1  26 FC                BNE   R1
564  C1A3  A6 00                LDAA  0,X
565  C1A5  33                   PUL   B
566  C1A6  BD C2 70             JSR   PULLX
567  C1A9  8D C9                BSR   E2911
568  C1AB  39                   RTS
569
570  C1AC  36        MCLK      PSH   A
571  C1AD  96 03                LDAA  STAT0
572  C1AF  84 DF                ANDA  =0DF
573  C1B1  97 03                STAA  STAT0
574  C1B3  B7 3C 13             STAA  CON0
575  C1B6  B7 3C 15             STAA  SSP
```

```
576  C1B9  8A 20              ORAA  =20
577  C1BB  97 03              STAA  STAT0
578  C1BD  B7 3C 13           STAA  CON0
579  C1C0  32                 PUL   A
580  C1C1  39                 RTS
581
582
583                 CODE      SET   *
584  C1C2                     ORG   DATA
585
586  0004  00       MTMP4     BYTE  0
587  0005  00       MTMP3     BYTE  0
588  0006  00       MTMP2     BYTE  0
589  0007  00       MTMP1     BYTE  0
590  0008  00       MTMP0     BYTE  0
591
592                 DATA      SET   *
593  0009                     ORG   CODE
594
595  C1C2  36       STUPC     PSH   A
596  C1C3  37                 PSH   B
597  C1C4  BD C2 59           JSR   PUSHX
598  C1C7  33                 PUL   B          (B,A) := X
599  C1C8  32                 PUL   A
600  C1C9  36                 PSH   A          KEEP X ON STACK
601  C1CA  37                 PSH   B
602  C1CB  48                 ASL   A
603  C1CC  59                 ROL   B
604  C1CD  48                 ASL   A
605  C1CE  59                 ROL   B
606  C1CF  48                 ASL   A
607  C1D0  59                 ROL   B
608  C1D1  48                 ASL   A
609  C1D2  59                 ROL   B
610  C1D3  8A 08              ORAA  =8
611  C1D5  D7 04              STAB  MTMP4
612  C1D7  97 05              STAA  MTMP3
613  C1D9  CE 07 20           LDX   =0720
614  C1DC  DF 06              STX   MTMP2
615  C1DE  86 FF              LDAA  =0FF
616  C1E0  97 08              STAA  MTMP0
617  C1E2  CE 00 04           LDX   =MTMP4
618  C1E5  BD C1 1D           JSR   LDMIR
619  C1E8  8D C2              BSR   MCLK
620  C1EA  BD C2 70           JSR   PULLX
621  C1ED  33                 PUL   B
622  C1EE  32                 PUL   A
623  C1EF  39                 RTS
624
625  C1F0  B6 3C 0F  RCPU     LDAA  CPU0
626  C1F3  F6 3C 0E           LDAB  CPU1
627  C1F6  39                 RTS
628
629  C1F7  B7 3C 0F  WCPU     STAA  CPU0
630  C1FA  F7 3C 0E           STAB  CPU1
631  C1FD  39                 RTS
```

PANEL

```
633  C1FE  86 00     RMM         LDAA  =00
634  C200  B7 3C 08              STAA  MIR4
635  C203  4F                    CLR   A
636  C204  B7 3C 09              STAA  MIR3
637  C207  86 07                 LDAA  =7
638  C209  B7 3C 0A              STAA  MIR2
639  C20C  86 60                 LDAA  =60
640  C20E  B7 3C 0B              STAA  MIR1
641  C211  86 3D     RMMS        LDAA  =3D
642  C213  B7 3C 0C              STAA  MIR0
643  C216  FF 3C 0E              STX   CPU1
644  C219  B7 3C 15              STAA  SSP
645  C21C  86 01                 LDAA  =1
646  C21E  B7 3C 0C              STAA  MIR0
647  C221  B6 3C 0F              LDAA  CPU0
648  C224  F6 3C 0E              LDAB  CPU1
649  C227  39                    RTS
650  C228  36        WMM         PSH   A
651  C229  86 00                 LDAA  =00
652  C22B  B7 3C 08              STAA  MIR4
653  C22E  4F                    CLR   A
654  C22F  B7 3C 09              STAA  MIR3
655  C232  86 07                 LDAA  =7
656  C234  B7 3C 0A              STAA  MIR2
657  C237  86 60                 LDAA  =60
658  C239  B7 3C 0B              STAA  MIR1
659  C23C  32                    PUL   A
660
661  C23D  B7 3C 0F  WMMS        STAA  CPU0
662  C240  F7 3C 0E              STAB  CPU1
663  C243  36                    PSH   A
664  C244  86 1D                 LDAA  =1D
665  C246  B7 3C 0C              STAA  MIR0
666  C249  B7 3C 15              STAA  SSP
667  C24C  86 3D                 LDAA  =3D
668  C24E  B7 3C 0C              STAA  MIR0
669  C251  FF 3C 0E              STX   CPU1
670  C254  B7 3C 15              STAA  SSP
671  C257  32                    PUL   A
672  C258  39                    RTS
```

```
674  C259  97 00     PUSH.X      STAA  C
675  C25B  DF 01                 STX   X.SAVE
676  C25D  32                    PUL   A           HI(RA)
677  C25E  B7 20 01              STAA  RET+1
678  C261  32                    PUL   A           LO(RA)
679  C262  B7 20 02              STAA  RET+2
680  C265  96 02                 LDAA  X.SAVE+1    LO(X)
681  C267  36                    PSH   A
682  C268  96 01                 LDAA  X.SAVE      HI(X)
683  C26A  36                    PSH   A
684  C26B  96 00                 LDAA  C
685  C26D  7E 20 00              JMP   RET         'RTS'
686
687  C270  97 00     PULL.X      STAA  C
688  C272  32                    PUL   A           HI(RA)
689  C273  B7 20 01              STAA  RET+1
690  C276  32                    PUL   A           LO(RA)
691  C277  B7 20 02              STAA  RET+2
692  C27A  32                    PUL   A           HI(X)
693  C27B  97 01                 STAA  X.SAVE
694  C27D  32                    PUL   A           LO(X)
695  C27E  97 02                 STAA  X.SAVE+1
696  C280  DE 01                 LDX   X.SAVE
697  C282  96 00                 LDAA  C
698  C284  7E 20 00              JMP   RET         'RTS'
699
700  C287  8D D0     PRINT.X     BSR   PUSH.X
701  C289  30                    TSX
702  C28A  BD C0 E3              JSR   OUT.WORD
703  C28D  8D E1                 BSR   PULL.X
704  C28F  39                    RTS
705
706
707
708                 RESET       EQU   *
709  C290                       BEGIN RESET
710  C290  36                    PSH   A
711  C291  86 BE                 LDAA  =0BE
712  C293  97 03                 STAA  STAT0
713  C295  B7 3C 13              STAA  CON0
714  C298  B7 3C 15              STAA  SSP                 MAKES A CPU CLOCK PULSE
715  C29B  86 BF                 LDAA  =0BF
716  C29D  7E C1 43              JMP   D29A
717  C2A0                       END   RESET
```

```
719  C2A0                          BEGIN REGISTERS
720                                DEF   READ.REG
721
722                    *         MICROINSTRUCTION USED BY READ/WRITE REGISTER
723                    *
724                    *           DST FCT SRC C  A  B  SH  PC S  E  SC DST  SRC
725                    *WRITE.R  1  B  OR  DZ  0  0  0  -   -  -  -  0  ALU  PANEL
726  C2A0  6F 80 07    WRITE.R   BYTE      6F,80,07,60,0D
     C2A3  60 0D
727                    *
728                    *READ.R   1  -  OR  ZA  0  0  0  -  -  -  -  0  -    ALU
729  C2A5  2E 00 07    READ.R    BYTE      2E,00,07,60,0F0
     C2A8  60 F0
730                    *
731                    Q.REG     EQU       10        DEFINITION OF Q.REGISTER
732                    *
733                    ************************************************************
734                    *
735                    *  PROCEDURE LOADREG(REGNR, VALUE);
736                    *    (* LOAD 2901 REGISTER 'REGNR' WITH 'VALUE' *)
737                    *    BEGIN CPU := VALUE;
738                    *      MIR := (1 B OR DZ 0 0 0 SC - - - 0 ALU PANEL);
739                    *      IF REGNR <> Q.REG THEN MIR.REG = REGNR
740                    *      ELSE MIR.DST := Q
741                    *      END;
742                    *      SINGLE STEP
743                    *    END;
744                    *
745                    *  PARAMETERS
746                    *     REGNR - B REGISTER
747                    *     VALUE - X REGISTER
748                    *
749  C2AA  BD C2 59    LOADREG   JSR       PUSHX
750  C2AD  37                    PSH       B
751  C2AE  FF 3C 0E              STX       CPU1
752  C2B1  CE C2 A0              LDX       =WRITE.R
753  C2B4  BD C1 1D              JSR       LDMIR
754  C2B7  C1 10                 CMPB      =Q.REG
755  C2B9  27 15                 BEQ       LOAD.Q
756  C2BB  36                    PSH       A
757  C2BC  4F                    CLR       A
758  C2BD  58                    ASL       B
759  C2BE  58                    ASL       B
760  C2BF  58                    ASL       B
761  C2C0  58                    ASL       B
762  C2C1  58                    ASL       B
763  C2C2  49                    ROL       A         MOST SIGNIFICANT INTO A
764  C2C3  CA 07                 ORAB      =07
765  C2C5  8A 80                 ORAA      =80
766  C2C7  F7 3C 0A              STAB      MIR2
767  C2CA  B7 3C 09              STAA      MIR3
768  C2CD  32                    PUL       A
769  C2CE  20 05                 BRA       LDR
```

```
PANEL     REGISTERS

771  C2D0  C6 0F     LOAD.Q    LDAB      =0F
772  C2D2  F7 3C 08            STAB      MIR4
773  C2D5  B7 3C 15  LDR       STAA      SSP       MAKE SINGLE STEP
774  C2D8  33                  PUL       B
775  C2D9  BD C2 70            JSR       PULLX
776  C2DC  39                  RTS
777
778
779
780                    ************************************************************
781                    *
782                    *  PROCEDURE READREG(REGNR, VAR VALUE);
783                    *    (* GET CONTENT OF 2901 REGISTER 'REGNR'*);
784                    *    BEGIN MIR := (1 - OR ZA 0 0 0 SC - - - 0 - ALU);
785                    *      IF REGNR <> Q.REG THEN MIR.REG := REGNR
786                    *      ELSE MIR.RS := ZQ;
787                    *      END;
788                    *      EDMIR; VALUE := CPU ; EPMIR
789                    *    END;
790                    *
791                    *    PARAMETERS
792                    *       REGNR  B REG
793                    *       VALUE  X REG
794                    *
795  C2DD  37          READREG   PSH       B
796  C2DE  CE C2 A5              LDX       =READ.R
797  C2E1  BD C1 1D              JSR       LDMIR
798  C2E4  C1 10                 CMPB      =Q.REG
799  C2E6  27 06                 BEQ       READ.Q
800  C2E8  58                    ASL       B
801  C2E9  F7 3C 09              STAB      MIR3
802  C2EC  20 05                 BRA       RDR
803  C2EE  C6 2D     READ.Q      LDAB      =2D
804  C2F0  F7 3C 08              STAB      MIR4
805  C2F3  FE 3C 0E  RDR         LDX       CPU1
806  C2F6  02                    NOP
807  C2F7  02                    NOP
808  C2F8  02                    NOP
809  C2F9  02                    NOP
810  C2FA  02                    NOP
811  C2FB  02                    NOP
812  C2FC  33                    PUL       B
813  C2FD  39                    RTS
814
815  C2FE                        END   REGISTERS
```

```
817  C2FE                        BEGIN MENU
818                              DEF   GETWORD,GETKEY,MENU,COMMANDS,COM.ESC
819                              DEF   XWORD,CARR,MAINID,SUBID
820
821                  *           PROCEDURE GETWORD (VAR X.WORD: WORD; VAR A: CHAR;
822                  *                              VAR CARR: BYTE);
823                  *             VAR B: BYTE;
824                  *                 C: BYTE; (* A-REG *)
825                  *           BEGIN
826                  *             X.WORD := 0;  CARR := 0;
827                  *             R.CH(A);
828                  *             WHILE A = BLANK DO R.CH(A) END;
829                  *             LOOP
830                  *               C := BIN(A);
831                  *               IF CARRY (* SIDE EFFECT OF BIN *)
832                  *               THEN EXIT END;
833                  *               X.WORD := 16 * X.WORD + C;
834                  *               CARR := -1;
835                  *               R.CH(A);
836                  *             END;
837                  *           END GETWORD
838
839                  CODE        SET   *
840  C2FE                        ORG   DATA
841  0009  00 00     ACT.MENU    WORD  0           MENU OF THE ACTUAL COMMAND-INTERPRETER
842  000B  00 00     MAIN.ID     WORD  0
843  000D  00 00     SUB.ID      WORD  0
844  000F  00 00     X.WORD      WORD  0
845  0011  00        CARR        BYTE  0
846  0012  00 00     SUB.M.STACK WORD  0
847
848                  DATA        SET   *
849  0014                        ORG   CODE
850
851                  GET.WORD    EQU   *
852  C2FE                        BEGIN GETWORD
853  C2FE  BD C2 59              JSR   PUSH.X
854  C301  37                    PSH   B
855  C302  CE 00 0F              LDX   =X.WORD
856  C305  6F 00                 CLR   0,X
857  C307  6F 01                 CLR   1,X         X.WORD := 0
858  C309  7F 00 11              CLR   CARR
859
860  C30C  BD C0 24  FIRST.CHAR  JSR   R.CH
861  C30F  81 20                 CMPA  =BLANK
862  C311  27 F9                 BEQ   FIRST.CHAR
863  C313  36        LOOP        PSH   A
864  C314  BD C0 9C              JSR   BIN
865  C317  25 17                 BCS   EXIT.LOOP
866
867  C319  C6 04                 LDAB  =4
868  C31B  68 01     FOR         ASL   1,X
869  C31D  69 00                 ROL   0,X
870  C31F  5A                    DEC   B
871  C320  26 F9     END.FOR     BNE   FOR
```

```
873  C322  AA 01              ORAA  1,X         X.WORD := 16 * X.WORD + A
874  C324  A7 01              STAA  1,X
875  C326  32                 PUL   A          DUMMY
876  C327  CA FF               ORAB  =0FF
877  C329  D7 11               STAB  CARR
878  C32B  BD C0 24            JSR   R.CH
879  C32E  20 E3               BRA   LOOP
880
881  C330  32        EXIT.LOOP PUL   A
882  C331  33                  PUL   B
883  C332  BD C2 70            JSR   PULL.X
884  C335  39                  RTS
885  C336                      END   GETWORD
886
887
888
889                    *        PROCEDURE MENU(ACT.MENU: ADDRESS);
890                    *            X : CARDINAL;
891                    *            B : BYTE;
892                    *        BEGIN
893                    *          X := ACT.MENU;
894                    *          B := M[X]; INC(X);
895                    *          FOR B := B TO 1 BY -1 DO
896                    *            WSTRING(X);
897                    *            X := X + 2;
898                    *          END;
899                    *          W.CRLF;
900                    *        END MENU;
901
902                    MENU      EQU   *
903  C336  BD C2 59            JSR   PUSH.X
904  C339  37                  PSH   B
905  C33A  DE 09               LDX   ACT.MENU
906  C33C  E6 00               LDAB  0,X
907  C33E  08                  INX
908
909  C33F  BD C0 78  FOR       JSR   W.CRLF
910  C342  BD C0 6A            JSR   W.STRING
911  C345  08                  INX
912  C346  08                  INX
913  C347  5A                  DEC   B
914  C348  26 F5     END.FOR   BNE   FOR
915  C34A  BD C0 78            JSR   W.CRLF
916  C34D  33                  PUL   B
917  C34E  BD C2 70            JSR   PULL.X
918  C351  39                  RTS
```

```
920                 *           PROCEDURE GETKEY(X.WORD: WORD; VAR CARRY: BOOLEAN);
921                 *             VAR A: CHAR;
922                 *               CARR: BYTE;
923                 *               X: CARDINAL;
924                 *           BEGIN
925                 *             W.CRLF; W.CH(PROMPT);
926                 *             GETWORD(X.WORD,A,CARR);
927                 *             X := ACT.MENU;
928                 *             LOOP
929                 *               WHILE A=BLANK DO R.CH(A) END;
930                 *               B := M[X];
931                 *               INC(X);
932                 *               REPEAT
933                 *                 IF A = M[X]
934                 *                 THEN
935                 *                   SKIPTEXT;
936                 *                   CARRY := CARR < 0;
937                 *                   GOTO M[X];
938                 *                 ELSE
939                 *                   SKIPTEXT;
940                 *                   INC(X,2);
941                 *                 END;
942                 *                 DEC(B);
943                 *               UNTIL B=0;
944                 *               NO;
945                 *             END;
946                 *           END GETKEY;
```

```
948                     GET.KEY     EQU   *
949  C352                           BEGIN GETKEY
950  C352  37                      PSH   B
951  C353  36                      PSH   A
952  C354  BD C0 78                JSR   W.CRLF
953  C357  86 2A                   LDAA  =PROMPT
954  C359  BD C0 51                JSR   W.CH
955  C35C  BD C2 FE  LOOP          JSR   GET.WORD
956
957  C35F  81 20     WHILE         CMPA  =BLANK
958  C361  26 05                   BNE   END.WHILE
959  C363  BD C0 24                JSR   R.CH
960  C366  20 F7                   BRA   WHILE
961                     END.WHILE   EQU   *
962  C368  84 DF                   ANDA  =SHIFT.MASK
963  C36A  DE 09                   LDX   ACT.MENU
964  C36C  E6 00                   LDAB  0,X
965  C36E  08                      INX
966
967  C36F  A1 00     REPEAT        CMPA  0,X
968  C371  26 0B                   BNE   ELSE
969  C373  8D 15                   BSR   SKIP.TEXT
970  C375  32                      PUL   A
971  C376  33                      PUL   B
972  C377  EE 00                   LDX   0,X
973  C379  79 00 11                ROL   CARR          CARRY := BIT7(CARR)
974  C37C  6E 00                   JMP   0,X
975
976  C37E  8D 0A     ELSE          BSR   SKIP.TEXT
977  C380  08                      INX
978  C381  08                      INX
979  C382  5A                      DEC   B
980  C383  26 EA     UNTIL         BNE   REPEAT
981
982  C385  BD C0 97                JSR   NO
983  C388  20 D2     END.LOOP      BRA   LOOP
984
985  C38A  08        SKIP.TEXT     INX
986  C38B  6D 00                   TST   0,X
987  C38D  26 FB                   BNE   SKIP.TEXT
988  C38F  08                      INX
989  C390  39                      RTS
990
991  C391                          END   GETKEY
```

PANEL MENU

```
 993                    *           PROCEDURE COMMANDS(MAIN.ID,X: ADDRESS);
 994                    *             VAR C: BOOLEAN; Y: WORD; (* DUMMIES *)
 995                    *           BEGIN
 996                    *             ACT.MENU := X;
 997                    *             W.STRING(MAIN.ID);
 998                    *             LOOP
 999                    *               GET.KEY(Y,C);
1000                    *             END;
1001                    *           END COMMANDS;
1002
1003  C391  DF 09      COMMANDS    STX   ACT.MENU
1004  C393  9F 12                  STS   SUB.M.STACK
1005  C395  9E 12      COM.ESC     LDS   SUB.M.STACK
1006  C397  BD C0 78                JSR   W.CRLF
1007  C39A  DE 0B                   LDX   MAIN.ID
1008  C39C  BD C0 6A                JSR   W.STRING
1009
1010  C39F  8D B1      REP         BSR   GET.KEY
1011  C3A1  20 FC                   BRA   REP
1012
1013  C3A3                          END   MENU
1014
1015
1016
1017  C3A3  86 21      NONO        LDAA  =EXCLAM
1018  C3A5  BD C0 51                JSR   W.CH
1019  C3A8  32                      PUL   A
1020  C3A9  39                      RTS
```

PANEL

```
1022                    * * * * * * * * * * * * * * * * * * * * * * * *
1023                    *          P R O M   E 0 0 0                    *
1024                    * * * * * * * * * * * * * * * * * * * * * * * *
1025
1026  C3AA                          ORG   0E000
1027
1028                                USE   SOFT.INT,IRQ,NMI    (AS INTERRUPT VECTORS)
1029
1030  E000                          BEGIN MOTOROLA
1031                                USE   COMMANDS
1032                                USE   LOAD,DUMP,INSPECT,BOOT.GO,GO
1033                                USE   READ.CH,READ.HEX,READ.BYTE,READ.WORD
1034
1035                                DEF   U.RAM.LOAD
1036
1037                    CODE        SET   *
1038  E000                          ORG   DATA
1039
1040  0014  00          U.RAM.LOAD  BYTE  0       MICRO RAM LOADED
1041  0015  00          M.RAM.LOAD  BYTE  0       MOTOROLA RAM LOADED
1042  0016  01          TYPING      BYTE  1
1043
1044                    DATA        SET   *
1045  0017                          ORG   CODE
1046
1047
1048
1049
1050  E000  7E E2 73                JMP   MONITOR
```

```
1052                    * CHARACTER-, HEXDIGIT-, BYTE- AND WORDINPUT WITHOUT ECHO
1053                    * (TO BE USED BY TAPEDRIVING LOADERS)
1054
1055  E003                          BEGIN NO.ECHO
1056                                DEF   READ.CH,READ.HEX,READ.BYTE,READ.WORD
1057
1058                    READ.CH     EQU   *
1059  E003  B6 40 00    REPEAT      LDAA  HP.STATUS
1060  E006  47                      ASR   A
1061  E007  24 FA       UNTIL       BCC   REPEAT
1062  E009  B6 40 01                LDAA  HP.DATA
1063  E00C  39                      RTS
1064
1065
1066                    READ.HEX    EQU   *
1067  E00D  8D F4       REP         BSR   READ.CH
1068  E00F  BD C0 9C                JSR   BIN
1069  E012  24 05                   BCC   END.REP
1070  E014  BD C0 97                JSR   NO
1071  E017  20 F4                   BRA   REP
1072  E019  39          END.REP     RTS
1073
1074
1075                    READ.BYTE   EQU   *
1076  E01A  8D F1                   BSR   READ.HEX
1077  E01C  48                      ASL   A
1078  E01D  48                      ASL   A
1079  E01E  48                      ASL   A
1080  E01F  48                      ASL   A
1081  E020  97 00                   STAA  C
1082  E022  8D E9                   BSR   READ.HEX
1083  E024  9A 00                   ORAA  C
1084  E026  39                      RTS
1085
1086
1087                    READ.WORD   EQU   *
1088  E027  36                      PSH   A
1089  E028  8D F0                   BSR   READ.BYTE
1090  E02A  A7 00                   STAA  0,X
1091  E02C  8D EC                   BSR   READ.BYTE
1092  E02E  A7 01                   STAA  1,X
1093  E030  32                      PUL   A
1094  E031  39                      RTS
1095
1096  E032                          END   NO.ECHO
```

```
PANEL     MOTOROLA

1098                * LOAD CODE FROM HP-CASSETTE (STANDARD HEXADECIMAL CODED 6800-FORMAT)
1099                * ************************
1100
1101                *           PROCEDURE LOAD;
1102                *           VAR EOT: BOOLEAN; (* B-REG, 0=TRUE *)
1103                *
1104                *           PROCEDURE FASTREAD;
1105                *           BEGIN
1106                *             WCH(ESC);
1107                *             WCH(LOWER.E);
1108                *           END FASTREAD;
1109                *
1110                *             PROCEDURE READBLOCK;
1111                *             VAR CH : CHAR; (* A-REG *)
1112                *                 CHECKSUM,  (* RAM *)
1113                *                 BYTECOUNT : BYTE; (* B-REG *)
1114                *                 ADDR : ADDRESS;   (* X-REG *)
1115                *             BEGIN
1116                *               REPEAT
1117                *                 REPEAT
1118                *                   RCH(CH);
1119                *                 UNTIL CH = 'S';
1120                *                 READ(CH);
1121                *               UNTIL CH # '0';  (* SKIP HEADER *)
1122                *               EOT := CH = '9';
1123                *               IF NOT EOT
1124                *               THEN BEGIN
1125                *                 INBYTE(BYTECOUNT);
1126                *                 CHECKSUM := BYTECOUNT;
1127                *                 INWORD(ADDR);
1128                *                 CHECKSUM := CHECKSUM + ADDR MOD 40 + ADDR DIV 40;
1129                *                 REPEAT
1130                *                   INBYTE(M[ADDR]);
1131                *                   CHECKSUM := CHECKSUM + M[ADDR];
1132                *                   ADDR := ADDR + 1;
1133                *                   BYTECOUNT := BYTECOUNT - 1;
1134                *                 UNTIL BYTECOUNT = 3;
1135                *                 INBYTE(CH);  (* CHECKSUM *)
1136                *                 IF CHECKSUM <> COMPL(CH)
1137                *                 THEN BEGIN
1138                *                   WSTRING('CHECKSUM ERROR');
1139                *                 END;
1140                *               END;
1141                *             END READBLOCK;
1142                *
1143                *           BEGIN (* LOAD *)
1144                *             FASTREAD;
1145                *             REPEAT
1146                *               READBLOCK;
1147                *             UNTIL EOT;
1148                *             REPEAT READ(CH) UNTIL CH=NULL; (* FILE MARK *)
1149                *           END LOAD;
```

PANEL MOTOROLA

```
1151  E032                          BEGIN TAPE.LOAD
1152                                DEF   LOAD
1153
1154                  CODE          SET   *
1155  E032                          ORG   DATA
1156
1157  0017  00        CHECK.SUM     BYTE  0           USED BY READBLOCK (LOAD)
1158
1159                  DATA          SET   *
1160  0018                          ORG   CODE
1161
1162                  READ.BLOCK    EQU   *
1163  E032                          BEGIN READBLOCK
1164                  REPEAT0       EQU   *
1165  E032  8D CF     REPEAT1       BSR   READ.CH     SKIP UNTIL 'S'
1166  E034  81 53                   CMPA  ='S'
1167  E036  26 FA     UNTIL1        BNE   REPEAT1
1168  E038  8D C9                   BSR   READ.CH
1169  E03A  81 30                   CMPA  ='0'
1170  E03C  27 F4     UNTIL0        BEQ   REPEAT0     SKIP HEADER
1171
1172  E03E  16                      TAB
1173  E03F  C0 39                   SUBB  ='9'
1174  E041  27 2B     IF1           BEQ   ENDIF
1175  E043  8D D5     THEN1         BSR   READ.BYTE   BYTECOUNT
1176  E045  16                      TAB
1177  E046  CE 00 01                LDX   =X.SAVE
1178  E049  8D DC                   BSR   READ.WORD   ADDRESS
1179  E04B  AB 00                   ADDA  0,X
1180  E04D  AB 01                   ADDA  1,X
1181  E04F  97 17                   STAA  CHECKSUM    := BYTECOUNT + ADDR(HI) + ADDR(LO)
1182  E051  DE 01                   LDX   X.SAVE
1183  E053  8D C5     REPEAT2       BSR   READ.BYTE   DATA
1184  E055  A7 00                   STAA  0,X
1185  E057  9B 17                   ADDA  CHECKSUM
1186  E059  97 17                   STAA  CHECKSUM    := CHECKSUM + A
1187  E05B  08                      INX
1188  E05C  5A                      DEC   B
1189  E05D  C1 03                   CMPB  =3
1190  E05F  26 F2     UNTIL2        BNE   REPEAT2
1191  E061  8D B7                   BSR   READ.BYTE   CHECKSUM
1192  E063  43                      COM   A
1193  E064  91 17                   CMPA  CHECKSUM
1194  E066  27 06     IF2           BEQ   ENDIF
1195  E068  CE C0 0C  THEN2         LDX   =MESS1
1196  E06B  BD C0 6A                JSR   W.STRING
1197                  ENDIF         EQU   *
1198  E06E  39                      RTS
1199  E06F                          END   READBLOCK
1200
1201
1202                  LOAD          EQU   *
1203  E06F  BD C0 EE                JSR   FAST.READ
1204  E072  8D BE     REPEAT        BSR   READ.BLOCK
1205  E074  5D                      TST   B           END OF TAPE
1206  E075  26 FB     UNTIL         BNE   REPEAT
```

```
1207  E077  CE C0 07            LDX   =MESS3
1208  E07A  BD C0 6A            JSR   W.STRING
1209  E07D  8D 84      REP      BSR   READ.CH     TRACE UNTIL FILE MARK
1210  E07F  4D                  TST   A
1211  E080  26 FB      UNT      BNE   REP
1212  E082  7C 00 15            INC   M.RAM.LOAD
1213                    *        BRA   SHLURF
1214  E085                      END   TAPELOAD
1215
1216
1217  E085  5F          SHLURF  CLR   B
1218  E086  BD C0 46    LISTEN  JSR   BREAK       LISTEN TO FURTHER CHARACTERS FROM THE HP
1219  E089  5A                  DEC   B           AND SHLURF THEM
1220  E08A  26 FA               BNE   LISTEN
1221  E08C  39                  RTS
```

```
1223                    *           PROCEDURE DUMP(X.MENU: ADDRESS);
1224                    *           VAR ADDR : ADDRESS;
1225                    *               A : CHAR;
1226                    *
1227                    *           PROCEDURE DUMPLINE (VAR ADDR : ADDRESS);
1228                    *           VAR I : INTEGER; (* B-REG *)
1229                    *               J : INTEGER; (* LOWER 2 BITS OF ADDR *)
1230                    *           BEGIN
1231                    *             WCRLF;
1232                    *             PRINT(ADDR);
1233                    *             WCH(':');
1234                    *             FOR I := 1 TO 4 DO BEGIN
1235                    *               FOR J := 1 TO 4 DO BEGIN
1236                    *                 W1BLANK;
1237                    *                 OUTBYTE(M[ADDR]);
1238                    *                 ADDR := ADDR + 1;
1239                    *               END;
1240                    *               W1BLANK;
1241                    *             END;
1242                    *           END DUMPLINE;
1243                    *
1244                    *           BEGIN
1245                    *             ADDR := X.WORD;
1246                    *             GOTO ENTRY;
1247                    *             REPEAT
1248                    *               WRITE('ENTER ADDRESS:');
1249                    *               INWORD(ADDR);
1250                    *             ENTRY:
1251                    *               ADDR := 10 * (ADDR DIV 10);
1252                    *               REPEAT
1253                    *                 FOR I := 1 TO 10 DO DUMPLINE(ADDR);
1254                    *                 REPEAT
1255                    *                   RCH(A);
1256                    *                   IF A = 'L' THEN DUMPLINE(ADDR);
1257                    *                 UNTIL A <> 'L';
1258                    *               UNTIL A <> 'P';
1259                    *             UNTIL A <> 'A';
1260                    *           END DUMP;
1261
1262  E08D                      BEGIN DUMP
1263                            DEF   DUMP
1264
1265                    CODE    SET   *
1266  E08D                      ORG   DATA
1267  0018  00 00       ADDR    WORD  0
1268                    DATA    SET   *
1269  001A                      ORG   CODE
1270
1271                    DUMP.LINE EQU *
1272  E08D                      BEGIN DUMP.LINE
1273  E08D  36                  PSH   A
1274  E08E  37                  PSH   B
1275  E08F  BD C0 78            JSR   W.CRLF
1276  E092  DE 18               LDX   ADDR
1277  E094  BD C2 87            JSR   PRINT.X
1278  E097  86 3A               LDAA  =':'
```

PANEL MOTOROLA DUMP DUMPLINE

```
1279  E099  BD C0 51            JSR   W.CH
1280  E09C  C6 04               LDAB  =4
1281
1282  E09E  BD C0 62  FOR       JSR   W.1.BLANK
1283  E0A1  A6 00               LDAA  0,X
1284  E0A3  BD C0 C9            JSR   OUT.BYTE    M[ADDR]
1285  E0A6  08                  INX
1286  E0A7  DF 18               STX   ADDR
1287  E0A9  86 03               LDAA  =3
1288  E0AB  94 19               ANDA  ADDR+1
1289  E0AD  26 EF               BNE   FOR
1290
1291  E0AF  BD C0 62            JSR   W.1.BLANK
1292  E0B2  5A                  DEC   B
1293  E0B3  26 E9               BNE   FOR
1294
1295  E0B5  33                  PUL   B
1296  E0B6  32                  PUL   A
1297  E0B7  39                  RTS
1298  E0B8                      END   DUMP.LINE
1299
1300
1301  E0B8  FE 00 0F  DUMP      LDX   X.WORD
1302  E0BB  DF 18               STX   ADDR
1303  E0BD  CE 00 18            LDX   =ADDR
1304  E0C0  20 0F               BRA   ENTRY
1305
1306  E0C2  BD C0 78  REPEAT1   JSR   W.CRLF
1307  E0C5  CE C0 12            LDX   =MESS2
1308  E0C8  BD C0 6A            JSR   W.STRING
1309  E0CB  CE 00 18            LDX   =ADDR
1310  E0CE  BD C0 D8            JSR   IN.WORD
1311  E0D1  86 F0     ENTRY     LDAA  =0F0
1312  E0D3  A4 01               ANDA  1,X
1313  E0D5  A7 01               STAA  1,X
1314
1315  E0D7  86 10     REPEAT2   LDAA  =10
1316  E0D9  8D B2     FOR       BSR   DUMP.LINE
1317  E0DB  4A                  DEC   A
1318  E0DC  26 FB               BNE   FOR
1319
1320  E0DE  BD C0 24  REPEAT3   JSR   R.CH
1321  E0E1  84 DF               ANDA  =0DF
1322  E0E3  81 4C               CMPA  ='L'
1323  E0E5  26 04               BNE   NO.L
1324  E0E7  8D A4               BSR   DUMP.LINE
1325  E0E9  20 F3     UNTIL3    BRA   REPEAT3
1326
1327  E0EB  81 50     NO.L      CMPA  ='P'
1328  E0ED  27 E8     UNTIL2    BEQ   REPEAT2
1329
1330  E0EF  81 41               CMPA  ='A'
1331  E0F1  27 CF     UNTIL1    BEQ   REPEAT1
1332
1333  E0F3  39                  RTS
1334  E0F4                      END   DUMP
```

```
1336                    *           PROCEDURE INSPECT(X.WORD: ADDRESS);
1337                    *             VAR ADDR: ADDRESS; (* X-REG *)
1338                    *                 A: CHAR;
1339                    *           BEGIN
1340                    *             X := X.WORD;
1341                    *             LOOP
1342                    *               OUT.BYTE(M[X]);
1343                    *               R.CH(A);
1344                    *               WHILE A # ':' DO
1345                    *                 IN.BYTE(M[ADDR]);
1346                    *                 R.CH(A);
1347                    *               END;
1348                    *               IF NOT A IN [',','\'] THEN EXIT END;
1349                    *               IF A = ','
1350                    *               THEN INC(ADDR)
1351                    *               ELSE DEC(ADDR)
1352                    *               END;
1353                    *               W.CRLF;
1354                    *               PRINT.X;
1355                    *               W.1.BLANK;
1356                    *             END;
1357                    *           END INSPECT;
1358
1359  E0F4                      BEGIN INSPECT
1360                            DEF   INSPECT
1361
1362                    INSPECT EQU   *
1363  E0F4  FE 00 0F            LDX   X.WORD
1364
1365  E0F7  A6 00     LOOP      LDAA  0,X
1366  E0F9  BD C0 C9            JSR   OUT.BYTE
1367
1368  E0FC  BD C0 24  WHILE     JSR   R.CH
1369  E0FF  81 3A               CMPA  =':'
1370  E101  26 07               BNE   END.WHILE
1371  E103  BD C0 BC            JSR   IN.BYTE
1372  E106  A7 00               STAA  0,X
1373  E108  20 F2               BRA   WHILE
1374
1375  E10A  81 2C     END.WHILE CMPA  =COMMA
1376  E10C  27 05     IF        BEQ   THEN
1377  E10E  81 5C               CMPA  =BACK.SLASH
1378  E110  27 04               BEQ   ELSE
1379  E112  39                 RTS             EXIT
1380  E113  08        THEN      INX
1381  E114  20 01               BRA   END.IF
1382  E116  09        ELSE      DEX
1383  E117  BD C0 78  END.IF    JSR   W.CRLF
1384  E11A  BD C2 87            JSR   PRINT.X
1385  E11D  BD C0 62            JSR   W.1.BLANK
1386  E120  20 D5               BRA   LOOP
1387
1388  E122                      END   INSPECT
```

```
1390                    ************************************************************
1391                    *
1392                    *
1393                    *  PROCEDURE TYPECTRL;
1394                    *    VAR TYPING: BYTE;
1395                    *                0 - NO TYPING AT ALL
1396                    *                1 - TYPE END MESSAGE AND ERRORS
1397                    *                2 - TYPE ERRORS, DON'T TYPE END MESSAGE
1398                    *                3 - LIKE 1 + WAIT IN CASE OF ERROR
1399                    *                4 - SEND A PULSE IN CASE OF ERROR
1400                    *    BEGIN TYPING := X.WORD END;
1401
1402  E122                          BEGIN     TYPE.CTRL
1403                                DEF       TYPE.CTRL
1404
1405  E122  B6 00 10  TYPECTRL      LDAA      X.WORD+1
1406  E125  97 16                   STAA      TYPING
1407  E127  39                     RTS
1408
1409  E128                          END       TYPE.CTRL
1410                    *
1411                    ************************************************************
1412
1413
1414
1415
1416
1417                    *           MODULE INTERRUPTS;
1418                    *             EXPORT SOFT.INT,IRQ,NMI,PRINT.BUS,GO,TYPING
1419                    *             CONST SW.INT.INST = 3F;
1420                    *             VAR PSEUDO.IRQ,STOP.ADDR: ADDRESS;
1421                    *                 SAVE.AREA: ARRAY 0..2 OF BYTE;
1422
1423  E128                          BEGIN INTERRUPT
1424                                DEF   IRQ,NMI,GO
1425                                USE   FROM.NMI
1426
1427                    SW.INT.INST EQU   3F
1428
1429                    CODE        SET   *
1430  E128                          ORG   DATA
1431
1432  001A  00 00       STOP.ADDR   WORD  0
1433  001C              SAVE.AREA   BSS   3
1434  001F  00 00       PC.ON.STACK WORD  0
1435  0021  00 00       PSEUDO.IRQ  WORD  0
1436
1437                    DATA        SET   *
1438  0023                          ORG   CODE
```

```
1440                    *           PROCEDURE GO (X.WORD: ADDRESS);
1441                    *           CONST CC = 10H; (* INTERRUPT DISABLE BIT SET *)
1442                    *           VAR CARRY: BOOLEAN; (* COND.CODE *)
1443                    *             CARR: BOOLEAN; (* 0=FALSE *)
1444                    *             SP: ADDRESS; (* STACK POINTER *)
1445                    *           BEGIN
1446                    *             IF CARRY
1447                    *             THEN
1448                    *               SP := STACK;
1449                    *               PUSH(X.WORD);
1450                    *               DEC(SP,4);
1451                    *               PUSH(CC);
1452                    *             ELSE
1453                    *               INC(SP,2); (* SKIP THE MONITORS RETURN ADDRESS FROM *)
1454                    *             END;         (* CALLING GETKEY (IN COMMANDS) *)
1455                    *             GETWORD(STOP.ADDR,CARR);
1456                    *             W.CRLF;  (* ASSUME NOT AUTO LF *)
1457                    *             IF CARR
1458                    *             THEN
1459                    *                 SAV.AREA[0] := M[STOP.ADDR];
1460                    *               M[STOP.ADDR] := 'SW.INT.INST';
1461                    *             END;
1462                    *             TRANSFER;
1463                    *           END GO;
1464
1465
1466
1467                        GO      EQU   *
1468  E128                          BEGIN GO
1469                                USE   CARR
1470                        CC      EQU   10
1471                        JSR     EQU   0BD
1472
1473  E128  24 12           IF      BCC   ELSE
1474  E12A  8E 00 7F        THEN    LDS   =STACK
1475  E12D  FE 00 0F                LDX   X.WORD
1476  E130  BD C2 59                JSR   PUSH.X
1477  E133  34                      DES                 X-LOW
1478  E134  34                      DES                 X-HIGH
1479  E135  34                      DES                 A-REG
1480  E136  34                      DES                 B-REG
1481  E137  86 10                   LDAA  =CC
1482  E139  36                      PSH   A
1483  E13A  20 02                   BRA   END.IF
1484  E13C  31              ELSE    INS
1485  E13D  31                      INS
1486  E13E  BD C2 FE        END.IF  JSR   GETWORD
1487
1488  E141  BD C0 78                JSR   W.CRLF
1489
1490  E144  7D 00 11        IF1     TST   CARR
1491  E147  27 0D                   BEQ   END.IF1
1492  E149  FE 00 0F        THEN1   LDX   X.WORD
1493  E14C  DF 1A                   STX   STOP.ADDR
1494  E14E  A6 00                   LDAA  0,X
1495  E150  97 1C                   STAA  SAVE.AREA
```

```
1496  E152  86 3F              LDAA  =SW.INT.INST
1497  E154  A7 00              STAA  0,X
1498  E156  3B        END.IF1  RTI
1499
1500  E157                     END   GO
```

```
1502                    *           PROCEDURE IRQ;
1503                    *           BEGIN GOTO M[PSEUDO.IRQ] END IRQ;
1504
1505                    IRQ         EQU   *
1506  E157  DE 21                   LDX   PSEUDO.IRQ
1507  E159  6E 00                   JMP   0,X
1508
1509
1510                    *           PROCEDURE  NMI;
1511                    *           BEGIN
1512                    *             PRINTREGS;
1513                    *             GOTO MON.RESUME;
1514                    *           END NMI;
1515
1516                    NMI         EQU   *
1517  E15B  30                      TSX
1518  E15C  08                      INX
1519  E15D  08                      INX
1520  E15E  08                      INX
1521  E15F  08                      INX
1522  E160  08                      INX
1523  E161  DF 1F                   STX   PC.ON.STACK
1524  E163  20 40                   BRA   FROM.NMI
1525
1526
1527
1528
1529                    *           PROCEDURE SOFTWARE.INTERRUPT;
1530                    *             VAR I,CH: GESCHMAEUS;
1531                    *           BEGIN
1532                    *             IF HALT  (* M[SP+6]-1 = STOP.ADDR *)
1533                    *             THEN
1534                    *                 M[STOP.ADDR] := SAV.AREA[0];
1535                    *               DEC(M[SP+6],1);
1536                    *               PRINT.REGS;
1537                    *               GOTO MON.RESUME; (* MONITOR ENTRY WITHOUT STACK INIT *)
1538                    *             ELSE
1539                    *               IF TYPING # 0
1540                    *               THEN
1541                    *                 WRITE('ERROR IN ',MAIN.ID,' ',SUB.ID);
1542                    *                 PRINT.REGS;
1543                    *                 IF TYPING = 3 THEN READ(CH) END;
1544                    *                 IF TYPING = 4 THEN SEND(SIGNAL) END;
1545                    *                 W.CRLF;
1546                    *               END;
1547                    *               TRANSFER;
1548                    *             END;
1549                    *           END SOFTWARE.INTERRUPT;
1550
```

```
1552  E165                        BEGIN SOFT.INT
1553                              DEF   FROM.NMI,SOFT.INT
1554                              USE   MON.RESUME
1555
1556  E165  36        PRINT.REGS  PSH   A
1557  E166  17                    TBA
1558  E167  BD C0 C9              JSR   OUT.BYTE  WRITE(B)
1559  E16A  BD C0 62              JSR   W.1.BLANK
1560  E16D  32                    PUL   A
1561  E16E  BD C0 C9              JSR   OUT.BYTE  WRITE(A)
1562  E171  BD C0 62              JSR   W.1.BLANK
1563  E174  DE 1F                 LDX   PC.ON.STACK
1564  E176  09                    DEX
1565  E177  09                    DEX             X POINTS TO THE X REG ON THE STACK
1566  E178  BD C0 E3              JSR   OUTWORD   WRITE(X)
1567  E17B  BD C0 62              JSR   W.1.BLANK
1568  E17E  EE 02                 LDX   2,X       (PC)
1569  E180  BD C2 87              JSR   PRINT.X
1570  E183  39                    RTS
1571
1572                    SOFT.INT  EQU   *
1573  E184  30                    TSX             X -> CC
1574  E185  08                    INX
1575  E186  08                    INX
1576  E187  08                    INX             X -> X
1577  E188  08                    INX
1578  E189  08                    INX             X -> RA
1579  E18A  DF 1F                 STX   PC.ON.STACK
1580  E18C  EE 00                 LDX   0,X
1581  E18E  09                    DEX
1582  E18F  9C 1A       IF        CPX   STOP.ADDR
1583  E191  26 29                 BNE   ELSE
1584  E193  36          THEN      PSH   A         HALT
1585  E194  96 1C                 LDAA  SAVE.AREA
1586  E196  A7 00                 STAA  0,X
1587  E198  DE 1F                 LDX   PC.ON.STACK
1588  E19A  86 FF                 LDAA  =0FF      -1
1589  E19C  AB 01                 ADDA  1,X
1590  E19E  A7 01                 STAA  1,X
1591  E1A0  25 02                 BCS   PR
1592  E1A2  6A 00                 DEC   0,X
1593  E1A4  32          PR        PUL   A
1594
1595                    FROM.NMI  EQU   *
1596  E1A5  8D BE                 BSR   PRINT.REGS
1597  E1A7  7E E2 B9              JMP   MON.RESUME  LET THE REGISTERS ON THE STACK, FOR A FUTURE
1598                    *                           RESUME OF THE PROGRAM AT THE SAME ADDRESS
```

```
1600                    *           SIMULATION OF THE SOFTWARE INTERRUPT INSTRUCTION
1601                    *           REPLACING A,B BYE THE VALUE OF THE CPU-BUS THE SAME TIME
1602
1603  E1AA  BD C2 59  PRINT.BUS     JSR   PUSHX
1604  E1AD  30                      TSX
1605  E1AE  08                      INX
1606  E1AF  08                      INX
1607  E1B0  DF 1F                   STX   PC.ON.STACK
1608  E1B2  36                      PSH   A
1609  E1B3  37                      PSH   B
1610  E1B4  07                      TPA
1611  E1B5  36                      PSH   A
1612  E1B6  F6 3C 0E                LDAB  CPU1        LOAD A,B WITH CPU BUS
1613  E1B9  B6 3C 0F                LDAA  CPU1+1
1614
1615
1616                    ELSE        EQU   *
1617  E1BC  7D 00 16    IF1         TST   TYPING
1618  E1BF  27 42                   BEQ   END.IF1
1619  E1C1  CE E2 04                LDX   =ERR.IN     WRITE('ERROR IN')
1620  E1C4  BD C0 6A                JSR   W.STRING
1621  E1C7  FE 00 0B                LDX   MAIN.ID     WRITE(MAIN.ID)
1622  E1CA  BD C0 6A                JSR   W.STRING
1623  E1CD  BD C0 62                JSR   W.1.BLANK
1624  E1D0  FE 00 0D                LDX   SUB.ID      WRITE(SUB.ID)
1625  E1D3  BD C0 6A                JSR   W.STRING
1626  E1D6  BD C0 62                JSR   W.1.BLANK
1627  E1D9  8D 8A                   BSR   PRINT.REGS
1628  E1DB  96 16                   LDAA  TYPING
1629  E1DD  81 03                   CMPA  =3          WAIT ON ERROR
1630  E1DF  26 03                   BNE   PRNT2
1631  E1E1  BD C0 24                JSR   R.CH
1632  E1E4  81 04       PRNT2       CMPA  =4          SEND A SIGNAL
1633  E1E6  26 0F                   BNE   PRNT3
1634  E1E8  96 03                   LDAA  STAT0
1635  E1EA  84 FD                   ANDA  =0FD        RESET BIT 1
1636  E1EC  B7 3C 13                STAA  CON0
1637  E1EF  BD C1 AC                JSR   MCLK
1638  E1F2  8A 02                   ORAA  =2          SET AGAIN
1639  E1F4  B7 3C 13                STAA  CON0
1640  E1F7  BD C0 78    PRNT3       JSR   W.CRLF
1641  E1FA  BD C0 46                JSR   BREAK       HANDLE BREAK
1642  E1FD  4D                      TST   A
1643  E1FE  27 03                   BEQ   PRNT4       NO BREAK
1644  E200  BD C0 24                JSR   R.CH        DUMMY READ
1645                    PRNT4       EQU   *
1646  E203  3B          END.IF1     RTI
1647
1648  E204  45 52 52    ERR.IN      BYTE  'ERROR IN ',0
      E207  4F 52 20
      E20A  49 4E 20
      E20D  00
1649
1650  E20E                          END   SOFT.INT
1651
1652  E20E                          END   INTERRUPT
```

```
1654                    *           PROCEDURE MONITOR;
1655                    *             CONST STACK = 7F; GARBAGE;
1656                    *             VAR SP : ADDRESS;
1657                    *                 MEMORY;
1658                    *                 B, (* TRUE <=> U-RAM *)
1659                    *                 URAM.LOAD, MRAM.LOAD: BOOLEAN;
1660                    *           BEGIN
1661                    *             SP := STACK;
1662                    *             INIT;
1663                    *             RESET;
1664                    *             IF MEMORY = GARBAGE
1665                    *             THEN
1666                    *               MEMORY := NO GARBAGE;
1667                    *               URAM.LOAD := FALSE;
1668                    *               MRAM.LOAD := FALSE;
1669                    *             END;
1670                    *             CHECK.URAM(B);
1671                    *             IF HP.ONLINE
1672                    *             THEN
1673                    *               IF NOT MRAM.LOAD THEN WRITE('LOAD MOTOROLA RAM'); END;
1674                    *               IF B AND NOT URAM.LOAD THEN WRITE('LOAD U-RAM'); END;
1675                    *               COMMANDS(MONITOR.TITLE,MAIN.MENU);
1676                    *             ELSE
1677                    *               IF B AND NOT URAM.LOAD THEN LOAD.URAM; END;
1678                    *               BOOT.AND.GO;
1679                    *               LOOP END;
1680                    *             END;
1681                    *           END MONITOR;
1682
1683  E20E                          BEGIN MONITOR
1684
1685                    CODE        SET   *
1686  E20E                          ORG   DATA
1687
1688                    NO.GARBAGE  EQU   0A596
1689  0023  00 00       MEMORY      WORD  0
1690
1691                    DATA        SET   *
1692  0025                          ORG   CODE
1693
1694                    CHECK.URAM  EQU   *
1695  E20E                          BEGIN CHECK.URAM
1696
1697  E20E  CE 00 00                LDX   =0
1698  E211  5F                      CLR   B
1699  E212  BD C1 90                JSR   RUM
1700  E215  36                      PSH   A
1701
1702  E216  4C                      INC   A
1703  E217  BD C1 7B                JSR   WUM
1704  E21A  BD C1 90                JSR   RUM
1705
1706  E21D  33                      PUL   B
1707  E21E  11                      CBA
1708  E21F  27 08                   BEQ   NO.RAM
1709
```

```
1710  E221  17                    TBA
1711  E222  5F                    CLR   B
1712  E223  BD C1 7B              JSR   WUM
1713
1714  E226  C6 FF                 LDAB  #0FF
1715  E228  39                    RTS
1716
1717  E229  5F          NO.RAM    CLR   B
1718  E22A  39                    RTS
1719  E22B                        END   CHECK.URAM
```

```
1721                    HP.ONLINE   EQU   *
1722  E22B                          BEGIN CHECK.HP
1723
1724  E22B  37                      PSH   B
1725  E22C  CE E2 57                LDX   =STATUS.REQ
1726  E22F  BD C0 6A                JSR   W.STRING
1727
1728  E232  CE 00 00                LDX   =0
1729  E235  B6 40 00  REPEAT        LDAA  HP.STATUS
1730  E238  47                      ASR   A
1731  E239  25 06                   BCS   ANSWER
1732
1733  E23B  09                      DEX
1734  E23C  26 F7                   BNE   REPEAT
1735
1736  E23E  33        OFF.LINE      PUL   B
1737  E23F  4F                      CLR   A
1738  E240  39                      RTS                 Z <=> HP OFFLINE
1739
1740  E241  B6 40 01  ANSWER        LDAA  HP.DATA
1741  E244  81 1B                   CMPA  =ESC
1742  E246  26 F6                   BNE   OFF.LINE
1743
1744  E248  CE 00 06                LDX   =6
1745  E24B  BD C0 24  FOR           JSR   R.CH
1746  E24E  09                      DEX
1747  E24F  26 FA                   BNE   FOR
1748
1749  E251  BD E0 85                JSR   SHLURF
1750  E254  08                      INX                 NOT Z <=> HP ONLINE
1751  E255  33                      PUL   B
1752  E256  39                      RTS
1753
1754  E257  1B 5E 11  STATUS.REQ    BYTE  ESC,'↑',DC1,0
      E25A  00
1755
1756  E25B                          END   CHECK.HP
1757
1758
1759                    SAVE.L.REG  EQU   *
1760  E25B                          BEGIN SAVE.LREG
1761
1762                    L           EQU   0E
1763                    ERR.LOC     EQU   6
1764
1765  E25B  BD C1 58                JSR   EDMIR
1766  E25E  C6 0E                   LDAB  =L
1767  E260  BD C2 DD                JSR   READ.REG
1768  E263  DF 01                   STX   X.SAVE
1769  E265  D6 01                   LDAB  X.SAVE
1770  E267  96 02                   LDAA  X.SAVE+1
1771  E269  CE 00 06                LDX   =ERR.LOC
1772  E26C  BD C2 28                JSR   WMM
1773  E26F  BD C1 5F                JSR   EPMIR
1774  E272  39                      RTS
1775  E273                          END   SAVE.LREG
```

```
1777  E273                        BEGIN RESET
1778                              DEF   MONITOR,ESCAPED,MON.RESUME,BOOT.GO
1779
1780                        MONITOR EQU   *
1781  E273  8E 00 7F               LDS   =STACK
1782  E276  86 7E                  LDAA  =7E             RETURN FOR PULLX/PUSHX
1783  E278  B7 20 00               STAA  RET
1784  E27B  BD C0 19               JSR   INIT
1785  E27E  BD C2 90               JSR   RESET
1786  E281  BD E2 5B               JSR   SAVE.L.REG
1787
1788  E284  CE A5 96               LDX   =NO.GARBAGE
1789  E287  9C 23                  CPX   MEMORY
1790  E289  27 08                  BEQ   NOT.FIRST
1791
1792  E28B  DF 23                  STX   MEMORY
1793  E28D  7F 00 14               CLR   URAM.LOAD
1794  E290  7F 00 15               CLR   MRAM.LOAD
1795
1796  E293  BD E2 0E  NOT.FIRST    JSR   CHECK.URAM
1797  E296  8D 93                  BSR   HP.ONLINE
1798  E298  27 48                  BEQ   NO.HP
1799
1800  E29A  7D 00 15               TST   MRAM.LOAD
1801  E29D  26 06                  BNE   ELSE
1802  E29F  CE E2 F2               LDX   =WARN1
1803  E2A2  BD C0 6A               JSR   W.STRING
1804
1805  E2A5  5D            ELSE     TST   B
1806  E2A6  27 0B                  BEQ   ESCAPED
1807  E2A8  7D 00 14               TST   URAM.LOAD
1808  E2AB  26 06                  BNE   ESCAPED
1809  E2AD  CE E3 05               LDX   =WARN2
1810  E2B0  BD C0 6A               JSR   W.STRING
1811
1812  E2B3  8E 00 7F  ESCAPED      LDS   =STACK          ENTRY WITHOUT ACIA INITIALIZATION
1813  E2B6  BD C2 90               JSR   RESET
1814
1815  E2B9  CE E3 0F  MON.RESUME   LDX   =MONITOR.TI     ENTRY WITHOUT STACK INITIALIZATION
1816  E2BC  FF 00 0B               STX   MAIN.ID
1817  E2BF  CE E3 17               LDX   =MAIN.MENU
1818  E2C2  7E C3 91               JMP   COMMANDS
```

```
1820                    BOOT.GO     EQU   *
1821  E2C5  BD C2 90                JSR   RESET
1822  E2C8  CE 00 00                LDX   =0
1823  E2CB  FF 00 0F                STX   X.WORD
1824  E2CE  BD E2 D5                JSR   SET.U.PC
1825  E2D1  BD C1 66                JSR   START
1826  E2D4  39                      RTS
1827
1828
1829  E2D5  FE 00 0F  SET.UPC       LDX   X.WORD
1830  E2D8  BD C1 58                JSR   EDMIR
1831  E2DB  BD C1 C2                JSR   STUPC
1832  E2DE  BD C1 5F                JSR   EPMIR
1833  E2E1  39                      RTS
1834
1835
1836  E2E2  5D        NO.HP         TST   B
1837  E2E3  27 08                   BEQ   BOOT
1838  E2E5  7D 00 14                TST   URAM.LOAD
1839  E2E8  26 03                   BNE   BOOT
1840
1841  E2EA  BD E3 B3                JSR   LOAD.URAM
1842
1843  E2ED  8D D6     BOOT          BSR   BOOT.GO
1844  E2EF  3E        WAIT          WAI
1845  E2F0  20 FD                   BRA   WAIT
1846
1847  E2F2  4C 4F 41  WARN1         BYTE  'LOAD MOTOROLA RAM ',0
      E2F5  44 20 4D
      E2F8  4F 54 4F
      E2FB  52 4F 4C
      E2FE  41 20 52
      E301  41 4D 20
      E304  00
1848  E305  4C 4F 41  WARN2         BYTE  'LOAD ',LOWER.U,'RAM',0
      E308  44 20 75
      E30B  52 41 4D
      E30E  00
1849
1850  E30F                          END   RESET
1851
1852  E30F                          END   MONITOR
1853
1854
1855                  ****************************************************
1856
1857
1858  E30F  4D 4F 4E  MONITOR.TI    BYTE  'MONITOR',0
      E312  49 54 4F
      E315  52 00
1859  E317  0C        MAIN.MENU     BYTE  0C
1860  E318  55 20 75                BYTE  'U ',LOWER.U,'RAM LADEN',0
      E31B  52 41 4D
      E31E  20 4C 41
      E321  44 45 4E
      E324  00
```

PANEL MOTOROLA

```
1861  E325  E3 B3          WORD  LOAD.U.RAM
1862  E327  4F 20 42       BYTE  'O BOOT & GO',0
      E32A  4F 4F 54
      E32D  20 08 20
      E330  47 4F 00
1863  E333  E2 C5          WORD  BOOT.GO
1864  E335  4C 20 4C       BYTE  'L LOAD MOTOROLA',0
      E338  4F 41 44
      E33B  20 4D 4F
      E33E  54 4F 52
      E341  4F 4C 41
      E344  00
1865  E345  E0 6F          WORD  LOAD
1866  E347  50 20 4D       BYTE  'P MEM DUMP',0
      E34A  45 4D 20
      E34D  44 55 4D
      E350  50 00
1867  E352  E0 B8          WORD  DUMP
1868  E354  49 20 49       BYTE  'I INSPECT',0
      E357  4E 53 50
      E35A  45 43 54
      E35D  00
1869  E35E  E0 F4          WORD  INSPECT
1870  E360  47 20 47       BYTE  'G GO',0
      E363  4F 00
1871  E365  E1 28          WORD  GO
1872  E367  59 20 54       BYTE  'Y TYPE CTRL',0
      E36A  59 50 45
      E36D  20 43 54
      E370  52 4C 00
1873  E373  E1 22          WORD  TYPE.CTRL
1874  E375  53 20 4D       BYTE  'S MAIN STORE',0
      E378  41 49 4E
      E37B  20 53 54
      E37E  4F 52 45
      E381  00
1875  E382  20 03          WORD  2003
1876  E384  58 20 45       BYTE  'X EXEC',0
      E387  58 45 43
      E38A  00
1877  E38B  20 06          WORD  2006
1878  E38D  56 20 44       BYTE  'V DEVICES',0
      E390  45 56 49
      E393  43 45 53
      E396  00
1879  E397  20 09          WORD  2009
1880  E399  54 20 54       BYTE  'T TEST',0
      E39C  45 53 54
      E39F  00
1881  E3A0  20 0C          WORD  200C
1882  E3A2  4D 20 4D       BYTE  'M MICRO MEMORY',0
      E3A5  49 43 52
      E3A8  4F 20 4D
      E3AB  45 4D 4F
      E3AE  52 59 00
1883  E3B1  20 0F          WORD  200F
1884
```

```
      PANEL     MOTOROLA

1885  E3B3                      END   MOTOROLA
```

```
1887                    ************************************************************
1888                    LOAD.URAM   EQU   *
1889  E3B3                          BEGIN MICROM
1890                    *
1891                    *  PROCEDURE LOADRAM;
1892                    *    (* COPY ROM INTO RAM *)
1893                    *    BEGIN
1894                    *      FOR HELP1 := 800H TO 0FFFH DO
1895                    *        FOR B := 4 TO 0 STEP -1 DO
1896                    *          A := RUM(X,B); WUM(X,B,A); INC(XSUM,A)
1897                    *        END;
1898                    *      END;
1899                    *      HIGH.U.ADR := 800H;
1900                    *    END LOADRAM;
1901
1902                    CODE        SET   *
1903  E3B3                          ORG   DATA
1904
1905  0025  00          XSUM        BYTE  0           XSUM OF MICRO MEMORY
1906  0026  00 00       HIGH.U.ADR  WORD  0           HIGHEST LOADED ADR OF MICRO MEMORY
1907  0028  00 00       HELP1       WORD  0
1908  002A  00 00       HELP2       WORD  0
1909
1910                    DATA        SET   *
1911  002C                          ORG   CODE
1912
1913
1914  E3B3  CE 00 00              LDX       =0        START OF RAM
1915  E3B6  DF 2A                 STX       HELP2
1916  E3B8  CE 08 00              LDX       =800
1917  E3BB  DF 28                 STX       HELP1     START OF ROM
1918  E3BD  7F 00 25              CLR       XSUM
1919  E3C0  C6 04       LDR.LOP1  LDAB      =4
1920  E3C2  DE 28       LDR.LOP2  LDX       HELP1
1921  E3C4  BD C1 90              JSR       RUM
1922  E3C7  DE 2A                 LDX       HELP2
1923  E3C9  BD C1 7B              JSR       WUM
1924  E3CC  9B 25                 ADDA      XSUM
1925  E3CE  97 25                 STAA      XSUM
1926  E3D0  5A                    DEC       B
1927  E3D1  2A EF                 BPL       LDR.LOP2
1928  E3D3  08                    INX                 INC(HELP2)
1929  E3D4  DF 2A                 STX       HELP2
1930  E3D6  DE 28                 LDX       HELP1
1931  E3D8  08                    INX
1932  E3D9  DF 28                 STX       HELP1
1933  E3DB  8C 10 00              CPX       =1000     HIGHEST OF RAM + 1
1934  E3DE  26 E0                 BNE       LDR.LOP1
1935
1936  E3E0  CE 08 00              LDX       =800
1937  E3E3  DF 26                 STX       HIGH.U.ADR
1938  E3E5  7C 00 14              INC       U.RAM.LOAD
1939  E3E8  39                    RTS
1940                    *
1941  E3E9                        END   MICROM
```

```
1943                    .           INTERRUPT VECTOR'S
1944
1945  E3E9                          ORG   0E3F8
1946
1947  E3F8  E1 57                   WORD  IRQ
1948  E3FA  E1 84                   WORD  SOFT.INT
1949  E3FC  E1 5B                   WORD  NMI
1950  E3FE  E2 73                   WORD  MONITOR
1951
1952                    USER.DATA   EQU   DATA
1953
1954  E400                          END   PANEL
```