```
                        * GMXBUG-09 V2.1

                        * 6809 ROM debugger/monitor
                        * Copyright (C) 1981 by Gimix, Inc. Chicago, Illinois
                        * All rights reserved


                        * Section origins

                DFC0    SBUGS   EQU     $DFC0       SBUG scratchpad
                E400    SCRATC  EQU     $E400       GMXBUG scratchpad
                F400    VIDPRM  EQU     $F400       Video ROM
                F800    GMXBUG  EQU     $F800       Start of GMXBUG
                FFF0    VECTOR  EQU     $FFF0       Interrupt vectors


                        * DP value for direct addressing of scratchpad

                00E4    DPR     EQU     $E4
                00E4            SETDP   $E4


                        * I/O addresses

                E004    TERM    EQU     $E004       control terminal port
                E000    SPRINT  EQU     $E000       serial printer port
                E042    PPRINT  EQU     $E042       parallel printer port

                FF7F    TSR     EQU     $FF7F       task select register


                        * SBUG compatible scratchpad

DFC0                            ORG      SBUGS
DFC0                            RMB      2

                        * pointers to interrupt handlers (SWTPc compatible)
DFC2                    SW3VEC  RMB     2           SWI3
DFC4                    SW2VEC  RMB     2           SWI2
DFC6                    FIRVEC  RMB     2           FIRQ
DFC8                    IRQVEC  RMB     2           IRQ
DFCA                    SWIVEC  RMB     2           SWI

                        * Supervisor call table start & end
DFCC                    SVCORG  RMB     2
DFCE                    SVCLIM  RMB     12
DFDA                    CTLADR  RMB     2           address of control port
```

```
                    * GMXBUG scratchpad usage

E400                        ORG     SCRATC

                    * I/O vectors

E400                INVEC   RMB     2       char input
E402                KEYVEC  RMB     2       quick char input
E404                TSTVEC  RMB     2       char input test
E406                OUTVEC  RMB     2       char output
E408                PRTVEC  RMB     2       printer vector
E40A                NMIVEC  RMB     2       NMI vector
E40C                LINBUF  RMB     2
E40E                ENDLIN  RMB     2       end of line in buffer
E410                BRKTBL  RMB     16      breakpoint table
E420                MMODE   RMB     1       memory mode flag
E421                UPCASE  RMB     1       force uppercase flag
E422                BEGIN1  RMB     2       general purpose pointers
E424                END1    RMB     2
E426                BEGIN2  RMB     2
E428                        RMB     2
E42A                SUBTOT  RMB     1       extra work byte
E42B                        RMB     4
E42F                WAIT    RMB     1       output suspended flag
E430                MEMPTR  RMB     2       memory mode pointer
E432                POINTR  RMB     2       extra pointer
E434                NULLS   RMB     1       # of nulls after CR for printer
E435                PRTFLG  RMB     1       printer type flag
E436                DEST    RMB     1       console output flag
E437                MATCH1  RMB     1       match bytes for hex locate
E438                MATCH2  RMB     1
E439                MATCH3  RMB     1


                    * ASCII equates

                0008    BS      EQU     $08     backspace
                000A    LF      EQU     $0A     line feed
                000D    CR      EQU     $0D     carriage return
                0013    CTLS    EQU     $13     control-S
                001B    ESC     EQU     $1B     escape
                0020    SPC     EQU     $20     space
```

```
                    *********************************
                    *                               *
                    *                               *
                    *       START OF MONITOR        *
                    *                               *
                    *                               *
                    *********************************


F800                        ORG     GMXBUG

                    * vectors for indirect JMPs and JSRs


                    * SBUG-E compatible vectors

F800 F82E           MONITR  FDB     COLD        cold start
F802 F894           NXTCMD  FDB     WARMS       warm start
F804 F8D9           INCH    FDB     INCHAR      char input
F806 F8DD           INCHE   FDB     INECHO      char input with echo
F808 F8F0           INCHEK  FDB     INTEST      test for char input
F80A F8E3           OUTCH   FDB     OUTCHR      char output
F80C F8FA           PDATA   FDB     PRTDAT      output string
F80E F906           PCRLF   FDB     CARRTN      output CR/LF
F810 F8F8           PSTRNG  FDB     PRTST       output string with CR/LF
F812 F912           LRA     FDB     LREAL       Load Real Address


F814 20   7E   F894         BRA     WARMS       for FLEX compatibility


                    * GMXBUG-09 V2.0 additional system calls

F816 F91B           PSPACE  FDB     PRTSPC      output a space
F818 F923           PREGS   FDB     PRTREG      output register values
F81A F95A           PBYTE   FDB     PRTBYT      output byte in hex
F81C F8F4           INKEY   FDB     GETKEY      quick char input
F81E F8E7           INCAPS  FDB     GETUPS      input uppercase char
F820 F967           HEXIN   FDB     GETHEX      input value in hexadecimal
F822 F9A7           BINHEX  FDB     BINCVT      convert binary to hex ASCII
F824 F980           HEXBIN  FDB     HEXCVT      convert hex ASCII to binary
F826 F9BD           LINEIN  FDB     GETLIN      input a line
F828 F9EC           TBSRCH  FDB     LOOKUP      table search
F82A FE23           MOVBLK  FDB     MOVER0      move block of memory
F82C F9F9           PRINT   FDB     PRTOUT      hardcopy output
```

```
                        * Start of executable code

                        * Cold start initialization
 F82E 10CE E7FF       COLD    LDS    #SCRATC+$3FF set stack at top of scratch
 F832 B6   F400               LDA    VIDPRM    test for video ROM
 F835 81   12                 CMPA   #$12      1st byte would be NOP
 F837 1027 FBC5 F400          LBEQ   VIDPRM

                        * default - initialize serial terminal
 F83B CE   E400               LDU    #SCRATC   set up vectors to serial I/0
 F83E 8E   FEEB               LDX    #SERIN    routines in the scratch pad
 F841 AF   C1                 STX    0,U++
 F843 8E   FEF0               LDX    #SERKEY
 F846 AF   C1                 STX    0,U++
 F848 8E   FF0B               LDX    #SERTST
 F84B AF   C1                 STX    0,U++
 F84D 8E   FF14               LDX    #SEROUT
 F850 AF   C1                 STX    0,U++
 F852 8E   E004               LDX    #SPRINT+4 set SBUG control port pointer
 F855 BF   DFE0               STX    SBUGS+$20
 F858 86   13                 LDA    #$13      reset printer & terminal ACIAs
 F85A A7   84                 STA    0,X
 F85C A7   1C                 STA    -4,X
                        * initial setting for ACIAS:
                        * 8 data bits, 2 stop bits, no parity, interrupt disabled
 F85E 86   11                 LDA    #$11
 F860 A7   84                 STA    0,X
 F862 A7   1C                 STA    -4,X

                        * Video initialization jumps here when done
 F864 8E   FF27       SETUP    LDX   #HARDC    set printer output vector
 F867 AF   C1                  STX   0,U++
 F869 8E   FF88                LDX   #NMITRP   set NMI vector to point
 F86C AF   C1                  STX   0,U++     to GMXBUG abort handler
 F86E CC   E700                LDD   #SCRATC+$300
 F871 ED   C1                  STD   0,U++     set line buffer pointer
 F873 33   C8 12               LEAU  18,U      skip over breakpoint table
 F876 C6   1A                  LDB   #26
 F878 6F   C0         COLD1    CLR   0,U+      zero the next 26 bytes
 F87A 5A                       DECB
 F87B 26   FB   F878           BNE   COLD1
 F87D 8E   FF9E                LDX   #DUMRTI
 F880 CE   DFC2                LDU   #SW3VEC   set SBUG SWI2,SWI3,IRQ & FIRQ
 F883 C6   04                  LDB   #4        vectors,to point to RTI
 F885 AF   C1         COLD2    STX   0,U++
 F887 5A                       DECB
 F888 26   FB   F885           BNE   COLD2
 F88A 8E   FEBC                LDX   #BRKPT    set SBUG SWI vector to point
 F88D AF   C1                  STX   0,U++     to GMXBUG breakpoint routine
 F88F 8E   FFFF                LDX   #$FFFF    set SBUG SVC table pointers
 F892 AF   C1                  STX   0,U++     to SBUG default
```

```
                    * Warm start - also mainline loop re-entry

 F894 8E   F9FD       WARMS    LDX   #GMXHED   output header message
 F897 8D   5F   F8F8           BSR   PRTST
 F899 8E   F894                 LDX   #WARMS    put warmstart addr on stack
 F89C 32   7E                   LEAS  -2,S
 F89E AF   E4                   STX   0,S
 F8A0 1A   80                   ORCC  #$80      set Entire flag in CCR
 F8A2 34   7F                   PSHS  #$7F      save all the registers but PC

                    * Entrance from NMI trap
 F8A4 7F   E420       WARMS0   CLR   >MMODE

 F8A7 8D   5D   F906  WARMS1   BSR   CARRTN
 F8A9 86   E4                   LDA   #DPR
 F8AB 1F   8B                   TFR   A,DP      set DP to scratchpad
 F8AD 0D   20                   TST   <MMODE
 F8AF 27   0E   F8BF            BEQ   WARMS2    if in memory mode
 F8B1 DC   30                   LDD   <MEMPTR   output pointer & byte contents
 F8B3 17   0226 FADC            LBSR  OUT4HS    instead of prompt
 F8B6 DE   30                   LDU   <MEMPTR
 F8B8 A6   C4                   LDA   0,U
 F8BA 17   0224 FAE1            LBSR  OUT2HS
 F8BD 20   05   F8C4            BRA   WARM25

 F8BF 8E   FA23       WARMS2   LDX   #PROMPT   prompt the user for a command
 F8C2 8D   36   F8FA            BSR   PRTDAT

 F8C4 8D   21   F8E7  WARM25   BSR   GETUPS    input command letter
 F8C6 8E   FA6E                 LDX   #CODTBL
 F8C9 0D   20                   TST   <MMODE
 F8CB 27   03   F8D0            BEQ   WARMS3
 F8CD 8E   FA56                 LDX   #MEMTBL
 F8D0 17   0119 F9EC  WARMS3   LBSR  LOOKUP    look up command in table
 F8D3 26   D2   F8A7            BNE   WARMS1
 F8D5 AD   94                   JSR   [0,X]     do the command
 F8D7 20   CE   F8A7            BRA   WARMS1    do another command
```

```
                        * SBUG-E compatible utility subroutines

                        * Input character from console
 F8D9 6E   9F   E400    INCHAR  JMP     [INVEC]

                        * Input character from console with echo
                        * This routine doesn't echo ESC, to avoid trouble
                        * with terminals that use escape sequences.
 F8DD 8D   FA   F8D9    INECHO  BSR     INCHAR
 F8DF 81   1B                   CMPA    #ESC
 F8E1 27   0C   F8EF            BEQ     INECX
 F8E3 6E   9F  E406     OUTCHR  JMP     [OUTVEC]

                        * Input upper case only
 F8E7 7C   E421         GETUPS  INC     >UPCASE   force upper case shift
 F8EA 8D   Fl   F8DD            BSR     INECHO    do input and echo
 F8EC 7F   E421                 CLR     >UPCASE
 F8EF 39                INECX   RTS

                        * Test for char waiting
 F8F0 6E   9F  E404     INTEST  JMP     [TSTVEC]

                        * Grab char from control device
 F8F4 6E   9F  E402     GETKEY  JMP     [KEYVEC]

                        * Output string with prefixed CR/LF
 F8F8 8D   0C   F906    PRTST   BSR     CARRTN
                        * Output string
 F8FA 34   02           PRTDAT  PSHS    A         save ACCA value
 F8FC A6   80           PRTDT1  LDA     0,X+      get a char
 F8FE 81   04                   CMPA    #4
 F900 27   0E   F910            BEQ     PRTDT2    terminate at a 4
 F902 8D   DF   F8E3            BSR     OUTCHR    else output it & loop
 F904 20   F6   F8FC            BRA     PRTDT1

                        * output CR/LF
 F906 34   02           CARRTN  PSHS    A
 F908 86   0D                   LDA     #CR
 F90A 8D   D7   F8E3            BSR     OUTCHR
 F90C 86   0A                   LDA     #LF
 F90E 8D   D3   F8E3            BSR     OUTCHR
 F910 35   82           PRTDT2  PULS    A,PC

                        * This routine returns extended address 00 for 0-DFFF, &
                        * returns 0F for E000-FFFF, in conformity with SBUG-E.
                        * Address is passed in X, extension bits in ACCA.
 F912 4F                LREAL   CLRA
 F913 8C   DFFF                 CMPX    #$DFFF
 F916 23   02   F91A            BLS     LRAOUT
 F918 8A   0F                   ORA     #$F
 F91A 39                LRAOUT  RTS
```

```
                        * GMXBUG 09 extended utility subroutines

                        * Output a space
 F91B 34   02           PRTSPC  PSHS   A
 F91D 86   20                   LDA    #SPC
 F91F 8D   C2   F8E3            BSR    OUTCHR
 F921 35   82                   PULS   A,PC

                        * Output registers
                        * This routine outputs the current contents
                        * of the machine registers. The PC value is the
                        * return address left by the call to this routine.
 F923 34   7F           PRTREG  PSHS   #$7F      push all but PC
 F925 1F   43                   TFR    S,U       copy stack pointer
 F927 8D   04   F92D            BSR    PRTR1     call output routine
 F929 35   FF                   PULS   #$FF      exit by pulling all regs

                        * Callable subroutine for use by debugger - outputs
                        * registers previously stacked by breakpoint or interrupt
 F92B 33   64           REGPRT  LEAU   4,S       point to stack
 F92D 34   40           PRTR1   PSHS   U         save SP value
 F92F 8E   FA28                 LDX    #REGID    output label string
 F932 8D   C4   F8F8            BSR    PRTST
 F934 E6   C0                   LDB    0,U+      output CCR as binary
 F936 17   03C9 FD02            LBSR   BINRY1
 F939 8D   E0   F91B            BSR    PRTSPC
 F93B C6   03                   LDB    #3        output ACCA, ACCB, DPR
 F93D A6   C0           PRTRG1  LDA    0,U+
 F93F 17   019F FAE1            LBSR   OUT2HS
 F942 5A                        DECB
 F943 26   F8   F93D            BNE    PRTRG1
 F945 C6   04                   LDB    #4        output IX, IX, US, PC
 F947 34   04                   PSHS   B         keep count on stack
 F949 EC   C1           PRTRG2  LDD    0,U++
 F94B 17   018E FADC            BSR    OUT4HS
 F94E 6A   E4                   DEC    0,S
 F950 26   F7   F949            BNE    PRTRG2
 F952 32   61                   LEAS   1,S       pop count off stack
 F954 35   06                   PULS   D         output SP value
 F956 8D   02   F95A            BSR    PRTBYT
 F958 1F   98                   TFR    B,A
                        * Fall thru to output LS byte of SP & exit

                        * Output ACCA contents as 2 hex digits
 F95A 34   06           PRTBYT  PSHS   D         save regs
 F95C 8D   49   F9A7    PRTB1   BSR    BINCVT    convert ACCA
 F95E 8D   83   F8E3            BSR    OUTCHR    output lst digit
 F960 1F   98                   TFR    B,A
 F962 17   FF7E F8E3            LBSR   OUTCHR    output 2nd digit
 F965 35   86                   PULS   D,PC      restore regs & exit
```

```
                        * Hexadecimal input from console
F967 7C   E421      GETHEX  INC    >UPCASE   force upper case input
F96A 34   30                PSHS   X,Y
F96C 8D   4C F9BA           BSR    GETLNL    input a line
F96E 7F   E421              CLR    >UPCASE   reset case flag
F971 81   1B                CMPA   #ESC      abort if terminated with ESC
F973 27   05 F97A           BEQ    GETHX3
F975 BE   E40C              LDX    >LINBUF   get buffer pointer & convert
F978 20   08 F982           BRA    HXCVT0
F97A 1A   02        GETHX3  ORCC   #2        abort & set V=1
F97C 1F   20        HXCVT9  TFR    Y,D       return with value in D
F97E 35   B0                PULS   X,Y,PC

                        * Convert string pointed to by IX to binary value in D
F980 34   30        HEXCVT  PSHS   X,Y
F982 108E 0000      HXCVT0  LDY    #0        clear IY as accumulator
F986 E6   80        HXCVT1  LDB    0,X+      get a char
F988 C0   30                SUBB   #$30      remove ASCII offset
F98A C1   09        HXCVT2  CMPB   #9
F98C 23   0A F998           BLS    HXCVT3    if 0-9 OK
F98E C1   10                CMPB   #$10
F990 23   EA F97C           BLS    HXCVT9    if :-? exit
F992 C0   07                SUBB   #$7       adjust for A-F
F994 C5   F0                BITB   #$F0
F996 26   E4 F97C           BNE    HXCVT9    terminate if not 0-$F
F998 4F             HXCVT3  CLRA              clear MS byte of D
F999 1E   02                EXG    D,Y       swap new nybble and total
F99B 58                     ASLB
F99C 49                     ROLA              shift total 4 bits to the left
F99D 58                     ASLB
F99E 49                     ROLA
F99F 58                     ASLB
F9A0 49                     ROLA
F9A1 58                     ASLB
F9A2 49                     ROLA
F9A3 31   AB                LEAY   D,Y       add back to next 4 bits
F9A5 20   DF F986           BRA    HXCVT1

                        * Convert binary value in ACCA to hex digits in A:B
F9A7 1F   89        BINCVT  TFR    A,B       copy number
F9A9 8D   06 F9B1           BSR    CONVRT    convert low digit
F9AB 1E   89                EXG    A,B       swap with orig value
F9AD 44                     LSRA
F9AE 44                     LSRA
F9AF 44                     LSRA              convert high digit
F9B0 44                     LSRA
F9B1 84   0F        CONVRT  ANDA   #$F
F9B3 8B   90                ADDA   #$90      trick convert 1 digit in ACCA
F9B5 19                     DAA
F9B6 89   40                ADCA   #$40
F9B8 19                     DAA
F9B9 39                     RTS
```

```
                        * Input line into line buffer
 F9BA 17   FF5E F91B  GETLNL  LBSR   PRTSPC    leading space if needed
 F9BD 34   14         GETLIN  PSHS   B,X
 F9BF 5F                      CLRB              init char pointer
 F9C0 BE   E40C               LDX    >LINBUF
 F9C3 17   FF13 F8D9  GETLN1  LBSR   INCHAR    input char no echo
 F9C6 81   1B                 CMPA   #ESC
 F9C8 27   1A   F9E4          BEQ    GETLN6    abort on ESC
 F9CA 81   0D                 CMPA   #CR
 F9CC 27   16   F9E4          BEQ    GETLN6    terminate on CR
 F9CE 81   08                 CMPA   #BS
 F9D0 26   06   F9D8          BNE    GETLN3    move pointer back on BS
 F9D2 5D                      TSTB
 F9D3 27   EE   F9C3          BEQ    GETLN1    except at beginning
 F9D5 5A                      DECB
 F9D6 20   07   F9DF          BRA    GETLN5
 F9D8 C1   4F         GETLN3  CMPB   #79        buffer full?
 F9DA 27   E7   F9C3          BEQ    GETLN1    ignore printing chars
 F9DC A7   85                 STA    B,X        store char
 F9DE 5C                      INCB              bump pointer
 F9DF 17   FF01 F8E3  GETLN5  LBSR   OUTCHR    echo char
 F9E2 20   DF   F9C3          BRA    GETLN1
 F9E4 A7   85         GETLN6  STA    B,X        store terminator
 F9E6 3A                      ABX
 F9E7 BF   E40E               STX    >ENDLIN    store term position
 F9EA 35   94                 PULS   B,X,PC     exit

                        *  Search table at [IX] for byte in ACCA
 F9EC A1   80         LOOKUP  CMPA   0,X+       match?
 F9EE 27   08   F9F8          BEQ    LOKOUT     yes - return Z=1
 F9F0 30   02                 LEAX   2.X        skip 2 data bytes
 F9F2 6D   84                 TST    0,X        end of table?
 F9F4 26   F6   F9EC          BNE    LOOKUP
 F9F6 1C   FB                 ANDCC  #$FB       not found: return Z=0
 F9F8 39              LOKOUT  RTS

                        * Output to hard copy device
 F9F9 6E   9F E408    PRTOUT  JMP    [PRTVEC]

                        * Header message
 F9FD 47 4D 58 42     GMXHED  FCC    /GMXBUG-09 V2.1/,13,10
 FA0E 28 43 29 20             FCC    /(C) 1981 Gimix Inc/,13,10,4

                        * Prompt string
 FA23 47 4D 58 3A     PROMPT  FCC    /GMX:/,4

                        * Label string for register dump
 FA28 20 45 46 48     REGID   FCC    / EFHI NZVC A  B  DP  X    Y/
 FA43 20 20 20 20             FCC    /    U    PC   SP/,13,10,4
```

```
                    * GMXBUG-09 commands

                    * Memory mode commands
FA56 20             MEMTBL  FCC     / /         change & next
FA57 FCF2                   FDB     CHGBMP
FA59 2B                     FCC     /+/         next
FA5A FCF4                   FDB     BUMP
FA5C 2D                     FCC     /-/         previous
FA5D FCF6                   FDB     DROP
FA5F 3D                     FCC     /=/         change
FA60 FCFB                   FDB     CHANGE
FA62 32                     FCC     /2/         binary
FA63 FD00                   FDB     BINARY
FA65 22                     FCC     /"/         ASCII entry
FA66 FD18                   FDB     ASCII
FA68 24                     FCC     /$/         hex entry
FA69 FD23                   FDB     HEX
FA6B 0D                     FCB     CR          exit memory mode
FA6C FCEF                   FDB     MEMOFF
FA6E 41             CODTBL  FCC     /A/         hex arithmetic
FA6F FAA8                   FDB     ARITHM
FA71 42                     FCC     /B/         set breakpoint
FA72 FE4F                   FDB     BRKSET
FA74 43                     FCC     /C/         checksum
FA75 FAB9                   FDB     CHKSUM
FA77 46                     FCC     /F/         disassembly dump
FA78 FB71                   FDB     FDUMP
FA7A 44                     FCC     /D/         formatted memory dump
FA7B FAE7                   FDB     DDUMP
FA7D 47                     FCC     /G/         exit from breakpoint
FA7E FEAF                   FDB     GOTO
FA80 48                     FCC     /H/         hex locate
FA81 FBFE                   FDB     HEXLOC
FA83 49                     FCC     /I/         initialize system values
FA84 FC3E                   FDB     INIT
FA86 4A                     FCC     /J/         jump to address
FA87 FCC3                   FDB     JUMP
FA89 4B                     FCC     /K/         kill breakpoints
FA8A FE6B                   FDB     KILBRK
FA8C 4D                     FCC     /M/         memory mode
FA8D FCE8                   FDB     MEMODE
FA8F 4F                     FCC     /0/         OS-9
FA90 FC99                   FDB     OS9CMD
FA92 50                     FCC     /P/         print breakpoints
FA93 FE2D                   FDB     PRTBRK
FA95 52                     FCC     /R/         register dump & change
FA96 FD2B                   FDB     REGDMP
FA98 54                     FCC     /T/         test memory
FA99 FD99                   FDB     TSTMEM
FA9B 55                     FCC     /U/         jump to $F000
FA9C FCCC                   FDB     USER
FA9E 58                     FCC     /X/         block move
```

```
FA9F FDE6                      FDB    BLKMOV
FAA1 5A                        FCC    /Z/       fill memory
FAA2 FCDA                      FDB    ZAPMEM
FAA4 57                        FCC    /W/       FLEX warm start
FAA5 FCD3                      FDB    FLXWRM
FAA7 00                        FCB    0


                       * 'A' - hex arithmetic

FAA8 17   0088 FB33  ARITHM   LBSR   GETADS    get 2 values
FAAB 8D   37   FAE4           BSR    OUTSPC
FAAD DC   22                   LDD    <BEGIN1
FAAF D3   24                   ADDD   <END1     add them together
FAB1 8D   29   FADC           BSR    OUT4HS    output result
FAB3 DC   22                   LDD    <BEGIN1
FAB5 93   24                   SUBD   <END1     subtract 2nd from 1st
FAB7 20   23   FADC           BRA    OUT4HS    print result & exit


                       * 'C' - checksum a block of memory

FAB9 8D   78   FB33  CHKSUM   BSR    GETADS    get limits
FABB 9E   22                   LDX    <BEGIN1   get start address
FABD 4F                        CLRA             zero accumulator
FABE 5F                        CLRB
FABF 0F   2A                   CLR    <SUBTOT   zero 3rd byte
FAC1 EB   80           CHKSMJ  ADDB   0,X+
FAC3 89   00                   ADCA   #0        add carry if any
FAC5 24   02   FAC9            BCC    CHKSM2
FAC7 0C   2A                   INC    <SUBTOT   carry to 3rd byte if any
FAC9 30   84           CHKSM2  LEAX   0,X
FACB 27   04   FAD1            BEQ    CHKSM3    terminate at 0000
FACD 9C   24                   CMPX   <END1
FACF 23   F0   FAC1            BLS    CHKSM1    end at END1 address
FAD1 34   06           CHKSM3  PSHS   D         save low bytes of total
FAD3 8D   0F   FAE4            BSR    OUTSPC
FAD5 96   2A                   LDA    <SUBTOT   output highest byte
FAD7 17   FE80 F95A            LBSR   PRTBYT

                       * Recover & output the lower 2 bytes
FADA 35   06                   PULS   D


                       * Output 16-bit value in hex followed by a space

FADC 17   FE7B F95A  OUT4HS    LBSR   PRTBYT
FADF 1F   98                   TFR    B,A

                       * Output 8-bit value followed by a space

FAE1 17   FE76 F95A  OUT2HS    LBSR   PRTBYT
FAE4 16   FE34 F91B  OUTSPC    LBRA   PRTSPC
```

```
                    * 'D' - dump memory as hexadecimal and ASCII

FAE7 8D   4A   FB33 DDUMP   BSR    GETADS    input start and end addresses
FAE9 8E   FB2D              LDX    #SPCMSG
FAEC 17   FE09 F8F8         LBSR   PRTST     new line + 5 spaces
FAEF C6   10                LDB    #16
FAF1 96   23                LDA    <BEGIN1+1 at the top of each column
FAF3 84   0F         DDUMP1 ANDA   #$F       output the 4th digit of
FAF5 8D   EA   FAE1         BSR    OUT2HS    the addresses
FAF7 4C                     INCA
FAF8 5A                     DECB
FAF9 26   F8   FAF3         BNE    DDUMP1
FAFB 9E   22                LDX    <BEGIN1   get start address

                    * beginning of line loop
FAFD 17   FE06 F906 DDUMP2  LBSR   CARRTN    new line
FB00 1F   10                TFR    X,D
FB02 8D   D8   FADC         BSR    OUT4HS    output current address
FB04 C6   10                LDB    #16
FB06 A6   80         DDUMP3 LDA    0,X+      output 16 bytes in hex
FB08 8D   D7   FAE1         BSR    OUT2HS
FB0A 5A                     DECB
FB0B 26   F9   FB06         BNE    DDUMP3
FB0D 8D   D5   FAE4         BSR    OUTSPC    output 2 spaces
FB0F 8D   D3   FAE4         BSR    OUTSPC
FB11 30   10                LEAX   -16,X     move pointer back
FB13 C6   10                LDB    #16
FB15 A6   80         DDUMP4 LDA    0,X+
FB17 84   7F                ANDA   #$7F      print 16 bytes as ASCII
FB19 81   20                CMPA   #$20
FB1B 22   02   FB1F         BHI    DDUMP7
FB1D 86   2E         DDUMP6 LDA    #'.       replace 0-1F & 7F with "."
FB1F 81   7F         DDUMP7 CMPA   #$7F
FB21 27   FA   FB1D         BEQ    DDUMP6
FB23 17   FDBD F8E3         LBSR   OUTCHR
FB26 5A                     DECB
FB27 26   EC   FB15         BNE    DDUMP4
FB29 8D   1D   FB48         BSR    EOLCHK    end check or pause at end of line
FB2B 20   D0   FAFD         BRA    DDUMP2


FB2D 20 20 20 20     SPCMSG FCC    /     /,4 string of 5 spaces for dump
```

```
                        * Input 2 16-bit values & store in scratchpad with abort of
                        * calling routine if either input is terminated with ESC

FB33 17   FE31 F967  GETADS  LBSR   GETHEX
FB36 29   22   FB5A         BVS    ESCOUT
FB38 DD   22                STD    <BEGIN1
FB3A 17   FE2A F967         LBSR   GETHEX
FB3D 29   1B   FB5A         BVS    ESCOUT
FB3F DD   24                STD    <END1
FB41 39                     RTS


                        * Input hex value with caller abort on ESC

FB42 17   FE22 F967  GETHX1  LBSR   GETHEX    input the value
FB45 29   13   FB5A         BVS    ESCOUT    if term was ESC return 2 levels
FB47 39                     RTS               else return normally


                        * End of line end check or pause for dump & disassembly

FB48 9C   24         EOLCHK  CMPX   <END1      compare current addr to bounds
FB4A 22   0A   FB56         BHI    EOLCH1
FB4C 9C   22                CMPX   <BEGIN1
FB4E 25   06   FB56         BLO    EOLCH1     if outside bounds cheek for done
FB50 17   FD9D F8F0         LBSR   INTEST     if in bounds, check for pause
FB53 26   0A   FB5F         BNE    EOLCH2
FB55 39                     RTS               continue if no input waiting

FB56 DC   24         EOLCH1  LDD    <END1      check end value
FB58 27   05   FB5F         BEQ    EOLCH2     if end>0 exit command
FB5A 32   62         ESCOUT  LEAS   2,S        exit calling routine (command)
FB5C 0F   2F                CLR    <WAIT      leave WAIT off
FB5E 39                     RTS

FB5F 17   FD77 F8D9  EOLCH2  LBSR   INCHAR     if end=0 then pause
FB62 81   1B                CMPA   #ESC
FB64 27   F4   FB5A         BEQ    ESCOUT     if ESC is typed exit command
FB66 81   13                CMPA   #CTLS
FB68 26   02   FB6C         BNE    EOLCH3     if ctl-S is typed toggle WAIT
FB6A 03   2F                COM    <WAIT
FB6C 0D   2F         EOLCH3  TST    <WAIT      continue after 2nd ctl-S
FB6E 26   EF   FB5F         BNE    EOLCH2
FB70 39                     RTS
```

```
                    * 'F' -  disassembly dump

FB71 8D   C0   FB33 FDUMP   BSR    GETADS
FB73 9E   22                LDX    <BEGIN1
FB75 9F   32                STX    <POINTR

FB77 17   FD8C F906 FDUMP1  LBSR   CARRTN
FB7A DC   32                LDD    <POINTR   output current address
FB7C 17   FF5D FADC         LBSR   OUT4HS
FB7F C6   01                LDB    #1        initialize byte count
FB81 A6   80                LDA    0,X+      get instruction byte
FB83 81   60         FDUMP2 CMPA   #$60
FB85 22   33   FBBA         BHI    FDUM50
FB87 81   3C                CMPA   #$3C      CWAI
FB89 27   60   FBEB         BEQ    FDUMP4
FB8B 81   38                CMPA   #$38      misc 1-bytes
FB8D 2C   5D   FBEC         BGE    FDUM90
FB8F 85   10                BITA   #$10
FB91 27   58   FBEB         BEQ    FDUMP4    branches direct one-ops
FB93 81   1E                CMPA   #$1E
FB95 2D   06   FB9D         BLT    FDUMP3
FB97 85   04                BITA   #4
FB99 27   3C   FBD7         BEQ    FDUM58    LEA
FB9B 20   4E   FBEB         BRA    FDUMP4    EXG, TFR, PSH, PUL

FB9D 43              FDUMP3 COMA
FB9E 85   09                BITA   #9
FBA0 27   4A   FBEC         BEQ    FDUM90    DAA, SEX
FBA2 43                     COMA
FBA3 84   0E                ANDA   #$0E
FBA5 26   0A   FBB1         BNE    FDUMP6
FBA7 5C                     INCB              extended page instructions
FBA8 A6   80                LDA    0,X+      bump count & decode 2nd byte
FBAA 81   2F                CMPA   #$2F
FBAC 22   D5   FB83         BHI    FDUMP2
FBAE 5C                     INCB              add 1 for long branches
FBAF 20   D2   FB83         BRA    FDUMP2

FBB1 81   06         FDUMP6 CMPA   #6
FBB3 2D   37   FBEC         BLT    FDUM90    SYNC, NOP
FBB5 2E   34   FBEB         BGT    FDUMP4    ORCC, ANDCC
FBB7 5C                     INCB              LBRA, LBSR
FBB8 20   31   FBEB         BRA    FDUMP4


                    * 2-operand, & 1-operand indexed & extended

FBBA 5C              FDUM50 INCB
FBBB 84   3F                ANDA   #$3F
FBBD 81   30                CMPA   #$30      extended
FBBF 2C   2A   FBEB         BGE    FDUMP4
FBC1 85   10         FDUM55 BITA   #$10      direct
```

```
FBC3 26   27   FBEC          BNE    FDUM90
FBC5 85   20                 BITA   #$20
FBC7 26   0F   FBD8          BNE    FDUM60
FBC9 81   03                 CMPA   #3        test for 3-byte immediates
FBCB 27   1E   FBEB          BEQ    FDUMP4
FBCD 81   0C                 CMPA   #$C
FBCF 27   1A   FBEB          BEQ    FDUMP4
FBD1 81   0E                 CMPA   #$E
FBD3 27   16   FBEB          BEQ    FDUMP4
FBD5 20   15   FBEC          BRA    FDUM90


                    * indexed

FBD7 5C              FDUM58  INCB
FBD8 A6   84         FDUM60  LDA    0,X       decode postbyte
FBDA 2A   10   FBEC          BPL    FDUM90    5-bit offset
FBDC 84   0F                 ANDA   #$0F
FBDE 81   08                 CMPA   #8
FBE0 2D   0A   FBEC          BLT    FDUM90    auto(+ & -), reg & no offset
FBE2 81   0B                 CMPA   #$0B
FBE4 27   06   FBEC          BEQ    FDUM90    D offset
FBE6 5C                      INCB
FBE7 85   01                 BITA   #1
FBE9 27   0l   FBEC          BEQ    FDUM90    8-bit & 8-bit PC offset
FBEB 5C              FDUMP4  INCB


                    * output the instruction bytes in hex

FBEC 9E   32         FDUM90  LDX    <POINTR
FBEE A6   80         FDUM91  LDA    0,X+
FBF0 17   FEEE FAE1          LBSR   OUT2HS
FBF3 5A                      DECB
FBF4 26   F8   FBEE          BNE    FDUM91
FBF6 9F   32                 STX    <POINTR
FBF8 17   FF4D FB48          LBSR   EOLCHK    check for completion
FBFB 16   FF79 FB77          LBRA   FDUMP1
```

```
                    * 'H' - hexadecimal locate for 1, 2, or 3 bytes

FBFE 17   FF32 FB33  HEXLOC  LBSR   GETADS    get bounds for search
FC01 8E   E437               LDX    #MATCH1
FC04 0F   2A                 CLR    <SUBTOT   get 1, 2, or 3 match byte values
FC06 C6   03                 LDB    #3
FC08 D7   26                 STB    <BEGIN2
FC0A 17   FF35 FB42  HXLOC1  LBSR   GETHX1    input a hex value
FC0D DE   0E                 LDU    <ENDLIN   check line length
FC0F 1193 0C                 CMPU   <LINBUF
FC12 27   08   FC1C          BEQ    HXLOC2    if null string, go search
FC14 E7   80                 STB    0,X+
FC16 0C   2A                 INC    <SUBTOT   store match byte
FC18 0A   26                 DEC    <BEGIN2
FC1A 26   EE   FC0A          BNE    HXLOC1

                    * beginning of search loop
FC1C 9E   22         HXLOC2  LDX    <BEGIN1
FC1E CE   E437               LDU    #MATCH1
FC21 5F              HXLOC3  CLRB
FC22 A6   85         HXLOC4  LDA    B,X       compare a byte
FC24 Al   C5                 CMPA   B,U
FC26 26   0D   FC35          BNE    HXLOC5    if no match move pointer
FC28 5C                      INCB             try the next byte
FC29 Dl   2A                 CMPB   <SUBTOT
FC2B 26   F5   FC22          BNE    HXLOC4    if match complete output address
FC2D 17   FCD6 F906  HXLOC6  LBSR   CARRTN
FC30 1F   10                 TFR    X,D
FC32 17   FEA7 FADC          LBSR   OUT4HS
FC35 30   01         HXLOC5  LEAX   1,X       bump pointer and check bounds
FC37 27   04   FC3D          BEQ    HXLOC7    terminate at 0000
FC39 9C   24                 CMPX   <END1
FC3B 23   E4   FC21          BLS    HXLOC3
FC3D 39              HXLOC7  RTS

                    * 'I' - initialize system values

FC3E 8D   7A   FCBA  INIT    BSR    SPCIL1
FC40 17   FCA4 F8E7          LBSR   GETUPS    get subcommand
FC43 8E   FC4F               LDX    #INITAB   scan table
FC46 17   FDA3 F9EC          LBSR   LOOKUP
FC49 26   20   FC6B          BNE    REPORT    do report if no command
FC4B 8D   6D   FCBA          BSR    SPCIL1
FC4D 6E   94                 JMP    [0,X]     if found execute

                           Initialization subcommand table
FC4F 44              INITAB  FCC    /D/       output device
FC50 FC5C                    FDB    SETDST
FC52 4E                      FCC    /N/       null count
FC53 FC66                    FDB    SETNUL
FC55 50                      FCC    /P/       printer type - in part III
FC56 FF66                    FDB    SETPRT
```

```
FC58 55                     FCC    /U/       soft reset
FC59 FFB4                   FDB    RESETI
FC5B 00                     FCB    0


                    * set console output device
                    * B=Both (screen & printer) otherwise screen only

FC5C 8D   5F   FCBD SETDST  BSR    UPSIL1
FC5E 81   42                CMPA   #,B
FC60 27   01   FC63         BEQ    STDST1
FC62 4F                     CLRA
FC63 97   36        STDST1  STA    <DEST
FC65 39                     RTS


                    * set null count - number of nulls after CR in printer outpu

FC66 8D   58   FCC0 SETNUL  BSR    GETHXX
FC68 D7   34                STB    <NULLS
FC6A 39                     RTS


                    * report system status

FC6B 8E   FC90      REPORT  LDX    #SYSTAT   output status header
FC6E 17   FC87 F8F8         LBSR   PRTST
FC71 8E   E434              LDX    #NULLS
FC74 A6   80                LDA    0,X+      output null count
FC76 17   FE68 FAE1         LBSR   OUT2HS
FC79 86   50                LDA    #'P
FC7B E6   80                LDB    0,X+      output printer type
FC7D 26   02   FC81         BNE    REPRT1
FC7F 86   53                LDA    #'S
FC81 8D   0A   FC8D REPRT1  BSR    REPRT2
FC83 8D   35   FCBA         BSR    SPCIL1
FC85 86   42                LDA    #'B
FC87 E6   80                LDB    0,X+      output console device
FC89 26   02   FC8D         BNE    REPRT2
FC8B 86   53                LDA    #'S
FC8D 16   FC53 F8E3 REPRT2  LBRA   OUTCHR


                    * status header

FC90 4E 4C 20 50    SYSTAT  FCC    /NL P D/,13,10,4
```

```
                        * 'O' - switch to 0S-9 or other alternate monitor
FC99 8D   25   FCC0  OS9CMD  BSR    GETHXX
FC9B 8E   FCAE       OS9     LDX    #SWITCH
FC9E 108E E500               LDY    #$E500
FCA2 C6   0C                 LDB    #12
FCA4 A6   80         OS91    LDA    0,X+      copy switch code to RAM
FCA6 A7   A0                 STA    0,Y+
FCA8 5A                      DECB
FCA9 26   F9   FCA4          BNE    OS91
FCAB 7E   E500               JMP    $E500     execute it

                        * this code is actually  executed at $E500
FCAE 4F              SWITCH  CLRA
FCAF 1F   8B                 TFR    A,DP      clear the DP register
FCB1 86   20                 LDA    #$20
FCB3 B7   FF7F               STA    TSR       select OS9 task
FCB6 6E   9F FFFE            JMP    [RESET]   jump on reset vector

FCBA 16   FC5E F91B  SPCILJ  LBRA   PRTSPC
FCBD 16   FC27 F8E7  UPSIL1  LBRA   GETUPS
FCC0 16   FE7F FB42  GETHXX  LBRA   GETHX1

                        * 'J' - JSR to user program
                        * RTS goes back to debugger mainline

FCC3 8D   FB   FCC0  JUMP    BSR    GETHXX    input jump address
FCC5 1F   01                 TFR    D,X
FCC7 4F              JUMP1   CLRA
FCC8 1F   8B                 TFR    A,DP      set DP to 0
FCCA 6E   84                 JMP    0,X       go to user routine

                        * 'U' - jump to PROM at $F000

FCCC 8D   F2   FCC0  USER    BSR    GETHXX
FCCE 8E   F000               LDX    #$F000
FCD1 20   F4   FCC7          BRA    JUMP1

                        * 'W' - jump to FLEX warm start

FCD3 8D   EB   FCC0  FLXWRM  BSR    GETHXX
FCD5 8E   CD03               LDX    #$CD03
FCD8 20   ED   FCC7          BRA    JUMP1

                        * 'Z' - fill memory

FCDA 17   FE56 FB33  ZAPMEM  LBSR   GETADS    input bounds
FCDD 8D   El   FCC0          BSR    GETHXX    input fill byte
FCDF 9E   22                 LDX    <BEGIN1
FCE1 E7   80         ZPMEM1  STB    0,X+      fill with specified value
FCE3 9C   24                 CMPX   <END1
FCE5 23   FA   FCE1          BLS    ZPMEM1
FCE7 39                      RTS
```

```
                        * Memory mode commands

                        * 'M' - turn on memory mode

FCE8 8D   D6  FCC0   MEMODE  BSR    GETHXX    input pointer value
FCEA DD   30                 STD    <MEMPTR
FCEC 0C   20                 INC    <MMODE    set flag
FCEE 39                      RTS

                        * CR - turn off memory mode

FCEF 0F   20         MEMOFF  CLR    <MMODE    exit memory mode on CR
FCF1 39                      RTS

                        * ' ' - change byte & bump pointer

FCF2 8D   07   FCFB  CHGBMP  BSR    CHANGE

                        * '+' - bump pointer

FCF4 33   42         BUMP    LEAU   2,U

                        * '-' - decrement address

FCF6 33   5F         DROP    LEAU   -1,U
FCF8 DF   30         DRPRTS  STU    <MEMPTR   store changed pointer
FCFA 39                      RTS

                        * '=' - change byte

FCFB 8D   C3   FCC0  CHANGE  BSR    GETHXX    input a value
FCFD E7   C4                 STB    0,U       store it in memory
FCFF 39                      RTS

                        * '2' - binary display

FD00 E6   C4         BINARY  LDB    0,U       get byte contents
FD02 8D   00   FD04  BINRY1  BSR    DSPLAY    display MS nybble

                        * fall thru to display LS nybble

FD04 8D   B4   FCBA  DSPLAY  BSR    SPCIL1    space between nybbles
FD06 86   04                 LDA    04        output 4 bits
FD08 34   02                 PSHS   A
FD0A 4F              DSPLY1  CLRA
```

```
FD0B 58                     ASLB            shift a bit to ACCA
FD0C 49                     ROLA    #'0     make ASCII
FD0D 8A   30                ORA
FD0F 17   FBD1 F8E3 DSPLY2  LBSR    OUTCHR  output it
FD12 6A   E4                DEC     0,S
FD14 26   F4   FD0A         BNE     DSPLY1  decrement & loop
FD16 35   82                PULS    A,PC    exit


                    * '"' - ASCII entry
                    * terminates on a CR, which is not stored

FD18 17   FBC2 F8DD ASCII   LBSR    INECHO  input & echo a byte
FD1B 81   0D                CMPA    #CR
FD1D 27   D9   FCF8         BEQ     DRPRTS  exit on CR
FD1F A7   C0                STA     0,U+    store the byte in memory
FD21 20   F5   FD18         BRA     ASCII


                    * '$' - hexadecimal entry

FD23 8D   9B   FCC0 HEX     BSR     GETHXX  input a value
FD25 E7   C0                STB     0,U+    store in memory
FD27 DF   30                STU     <MEMPTR
FD29 20   F8   FD23         BRA     HEX     exit on GETHX1 abort


                    * End of memory mode commands
```

```
                        * 'R' - register display & change

 FD2B 17 FBFD F92B    REGDMP  LBSR   REGPRT    output the register values
 FD2E 8E FD71                 LDX    #CHGMSG   output prompt
 FD31 17 FBC4 F8F8            LBSR   PRTST
 FD34 17 FBB0 F8E7            LBSR   GETUPS    input a char
 FD37 81 0D                   CMPA   #CR
 FD39 27 35   FD70            BEQ    RGD15     exit on CR
 FD3B 8E FD86                 LDX    #REGTBL   find char in register table
 FD3E Al 81           RGDMP1  CMPA   0,X++
 FD40 27 09   FD4B            BEQ    RGDMP2
 FD42 6D 84                   TST    0,X
 FD44 26 F8   FD3E            BNE    RGDMP1
 FD46 86 3F                   LDA    #'?'
 FD48 16 FB98 F8E3            LBRA   OUTCHR    invalid char - exit with ?

 FD4B A6 1F           RGDMP2  LDA    -1,X      get offset into stack from table
 FD4D 34 02                   PSHS   A
 FD4F 17 FDF0 FB42            LBSR   GETHX1    input new value
 FD52 1F 02                   TFR    D,Y
 FD54 35 02                   PULS   A
 FD56 4D                      TSTA              changing stack value?
 FD57 2A 09   FD62            BPL    RGDMP3
 FD59 1F 24                   TFR    Y,S       yes - set new SP
 FD5B 8E F894                 LDX    #WARMS    setup return to mainline
 FD5E AF E2                   STX    0,-S
 FD60 6E 84                   JMP    0,X

 FD62 33 62           RGDMP3  LEAU   2,S       adjust around return address
 FD64 33 C6                   LEAU   A,U       point to register
 FD66 81 04                   CMPA   #4        1 or 2 byte register?
 FD68 2D 04   FD6E            BLT    RGDMP5
 FD6A 10AF C4                 STY    0,U       store 2 bytes
 FD6D 39                      RTS
 FD6E E7 C4           RGDMP5  STB    0,U       store 1 byte
 FD70 39              RGD15   RTS


                        * Prompt for register change
 FD71 52 65 67 69     CHGMSG  FCC    /Register to change? /,4

                        * Offsets for register change
 FD86 43 00           REGTBL  FCC    /C/.0     CCR
 FD88 41 01                   FCC    /A/,1     ACCA
 FD8A 42 02                   FCC    /B/,2     ACCB
 FD8C 44 03                   FCC    /D/,3     DPR
 FD8E 58 04                   FCC    /X/,4     IX
 FD90 59 06                   FCC    /Y/,6     IY
 FD92 55 08                   FCC    /U/,8     US
 FD94 50 0A                   FCC    /P/,10    PC
 FD96 53 FF                   FCC    /S/,255   SP
 FD98 00                      FCB    0
```

```
                        * 'T' - memory test - does a convergence test
                        * that will detect most memory failures

 FD99 17   FD97 FB33  TSTMEM  LBSR   GETADS    input bounds of test area
 FD9C 5F                      CLRB             init pass counter
 FD9D 17   FB66 F906  TESTM1  LBSR   CARRTN

                        * fill memory with pattern

 FDA0 9E   22         TSTM1A  LDX    <BEGIN1
 FDA2 8D   28   FDCC  TESTM2  BSR    MAKBYT    make byte & store
 FDA4 A7   80                 STA    0,X+
 FDA6 9C   24                 CMPX   <END1
 FDA8 23   F8   FDA2          BLS    TESTM2

                        * compare memory with pattern

 FDAA 9E   22                 LDX    <BEGIN1
 FDAC 8D   1E   FDCC  TESTM3  BSR    MAKBYT
 FDAE A8   80                 EORA   0,X+      make byte & compare
 FDB0 26   23   FDD5          BNE    ERROR     exit on error
 FDB2 9C   24                 CMPX   <END1
 FDB4 23   F6   FDAC          BLS    TESTM3
 FDB6 86   23                 LDA    #'#       output '#' for each pass
 FDB8 17   FB28 F8E3          LBSR   OUTCHR
 FDBB 4F                      CLRA
 FDBC 17   FB35 F8F4          LBSR   GETKEY
 FDBF 81   1B                 CMPA   #ESC
 FDC1 27   11   FDD4          BEQ    MKBRTS    exit if ESC is typed on console
 FDC3 5C                      INCB             bump pass counter
 FDC4 27   0E   FDD4          BEQ    MKBRTS    exit after 256 passes
 FDC6 C5   3F                 BITB   #$3F
 FDC8 26   D6   FDA0          BNE    TSTM1A    output CRLF every 64th pass
 FDCA 20   D1   FD9D          BRA    TESTMJ

                        * make byte for test - MS byte of addr + LS byte + pass #

 FDCC 9F   26         MAKBYT  STX    <BEGIN2
 FDCE 1F   98                 TFR    B,A
 FDD0 9B   26                 ADDA   <BEGIN2
 FDD2 9B   27                 ADDA   <BEGIN2+1
 FDD4 39              MKBRTS  RTS

                        * output error message, display wrong bits & exit

 FDD5 17   FB2E F906  ERROR   LBSR   CARRTN
 FDD8 30   1F                 LEAX   -1,X
 FDDA 34   02                 PSHS   A
 FDDC 1F   10                 TFR    X,D
 FDDE 17   FCFB FADC          LBSR   OUT4HS    Output error address
 FDE1 35   04                 PULS   B
 FDE3 16   FF1C FD02          LBRA   BINRY1    output error mask in binary
```

```
                    * 'X' - block move

FDE6 17   FD4A FB33 BLKMOV  LBSR   GETADS    input bounds of area to move
FDE9 17   FD56 FB42         LBSR   GETHX1    input destination address
FDEC DD   26                STD    <BEGIN2
FDEE DC   26         MOVER1 LDD    <BEGIN2
FDF0 1093 22                CMPD   <BEGIN1
FDF3 23   17   FE0C         BLS    NORMAL    check for start<dest<=end
FDF5 1093 24                CMPD   <END1
FDF8 22   12   FE0C         BHI    NORMAL
FDFA DC   24                LDD    <END1
FDFC 93   22                SUBD   <BEGIN1   reverse move: refigure dest ptr
FDFE D3   26                ADDD   <BEGIN2
FE00 1F   03                TFR    D,U
FE02 9E   24                LDX    <END1
FE04 DC   22                LDD    <BEGIN1
FE06 DD   24                STD    <END1     invert end pointer
FE08 C6   FF                LDB    #-1
FE0A 20   06   FE12         BRA    NORML1    go to move loop

FE0C 9E   22         NORMAL LDX    <BEGIN1   load pointers
FE0E DE   26                LDU    <BEGIN2
FE10 C6   01                LDB    #1
FE12 A6   84         NORML1 LDA    0,X       move a byte
FE14 A7   C4                STA    0,U
FE16 30   85                LEAX   B,X
FE18 33   C5                LEAU   B,U
FE1A 9C   24                CMPX   <END1     continue till bound is reached
FE1C 26   F4   FE12         BNE    NORML1
FE1E A6   84                LDA    0,X
FE20 A7   C4                STA    0,U       move last byte
FE22 39                     RTS


                    * entry for MOVBLK utility subroutine

FE23 34   5E         MOVER0 PSHS   A,B,DP,X,U save working registers
FE25 86   E4                LDA    #DPR      set DP
FE27 1F   8B                TFR    A,DP
FE29 8D   C3   FDEE         BSR    MOVER1    do the move
FE2B 35   DE                PULS   A,B,DP,X,U.PC restore registers & exit
```

```
                    * 'P' - output breakpoint table contents
FE2D 5F             PRTBRK  CLRB
FE2E 8E   E410              LDX    #BRKTBL   set pointer
FE31 17   FAD2 F906 PRTBK1  LBSR   CARRTN
FE34 34   04                PSHS   B
FE36 1F   98                TFR    B,A
FE38 17   FCA6 FAE1         LBSR   OUT2HS    output breakpoint
FE3B EC   84                LDD    0,X
FE3D 17   FC9C FADC         LBSR   OUT4HS    output address
FE40 A6   08                LDA    8,X
FE42 17   FB15 F95A         LBSR   PRTBYT    output saved byte
FE45 30   02                LEAX   2,X
FE47 35   04                PULS   B
FE49 5C                     INCB
FE4A C1   04                CMPB   #4        loop for 4 breakpoints
FE4C 2D   E3   FE31         BLT    PRTBK1
FE4E 39                     RTS

                    * 'B' - set a breakpoint
FE4F 8D   46   FE97 BRKSET  BSR    PICKBR    input breakpoint number
FE51 17   FCEE FB42         LBSR   GETHX1    input address
FE54 1083 0000              CMPD   #$0000    exit if address is 0000
FE58 27   10   FE6A         BEQ    BREAKX
FE5A 34   06                PSHS   D         save address
FE5C 8D   17   FE75         BSR    KLBRK1    if break set already kill it
FE5E 35   40                PULS   U         recover address
FE60 A6   C4                LDA    0,U       put break address and contents
FE62 EF   84                STU    0,X       of byte in table
FE64 A7   08                STA    8,X
FE66 86   3F                LDA    #$3F      put SWI at address
FE68 A7   C4                STA    0,U
FE6A 39             BREAKX  RTS

                    * 'K' - kill a breakpoint
                    * kill all 4 breakpoints if 'X' is entered
FE6B 8D   2A   FE97 KILBRK  BSR    PICKBR    input brkpt number & set pointer
FE6D DE   0E                LDU    <ENDLIN
FE6F A6   5F                LDA    -1,U
FE71 81   58                CMPA   #'X       check for X
FE73 27   15   FE8A         BEQ    KILALL
FE75 EE   84        KLBRK1  LDU    0,X       get address from table
FE77 27   Fl   FE6A         BEQ    BREAKX    exit if addr=0
FE79 A6   C4                LDA    0,U       check for SWI at addr
FE7B 81   3F                CMPA   #$3F
FE7D 26   04   FE83         BNE    KLBRK2    if no SWI, don't replace byte
FE7F A6   08                LDA    8,X       get byte contents
FE81 A7   C4                STA    0,U       replace the byte
FE83 6F   84        KLBRK2  CLR    0,X       zero the address & data entries
FE85 6F   01                CLR    1,X
FE87 6F   08                CLR    8.X
FE89 39                     RTS
```

```
                        * kill all 4 breakpoints
 FE8A 8E   E410         KILALL  LDX    #BRKTBL
 FE8D C6   04                   LDB    #4          loop for 4 entries
 FE8F 8D   E4   FE75    KILAL1  BSR    KLBRK1      use part of main routine
 FE91 30   02                   LEAX   2,X
 FE93 5A                        DECB
 FE94 26   F9   FE8F            BNE    KILAL1
 FE96 39                        RTS

                        * Input a breakpoint number & point into the table
 FE97 17   FACD F967    PICKBR  LBSR   GETHEX      input a number
 FE9A 29   0C   FEA8            BVS    PICKBX
 FE9C 1083 0003                 CMPD   #3
 FEA0 22   06   FEA8            BHI    PICKBX      abort with '?' if >3
 FEA2 58                        ASLB
 FEA3 8E   E410                 LDX    #BRKTBL     make pointer into brk table
 FEA6 3A                        ABX
 FEA7 39                        RTS
 FEA8 86   3F           PICKBX  LDA    #'?         print '?' & exit to mainline
 FEAA 32   62                   LEAS   2,S
 FEAC 16   FA34 F8E3            LBRA   OUTCHR

                        * 'G' - continue from breakpoint executes an RTI

 FEAF 17   FA69 F91B    GOTO    LBSR   PRTSPC
 FEB2 17   FA28 F8DD            LBSR   INECHO      get a char
 FEB5 81   0D                   CMPA   #CR
 FEB7 26   B1   FE6A            BNE    BREAKX      abort if not CR
 FEB9 32   62           GOTO1   LEAS   2,S         pop return to GMXBUG mainline
 FEBB 3B                        RTI

                        * Breakpoint handler
 FEBC AE   6A           BRKPT   LDX    10,S
 FEBE 33   1F                   LEAU   -1,X        adjust return address
 FEC0 27   0E   FED0            BEQ    BRKSWI      if addr=0 can't be breakpoint
 FEC2 5F                        CLRB
 FEC3 8E   E410                 LDX    #BRKTBL
 FEC6 11A3 81           BRKP1   CMPU   0,X++       search breakpoint table for addr
 FEC9 27   07   FED2            BEQ    FDBKPT
 FECB 5C                        INCB
 FECC Cl   04                   CMPB   #4          check four entries
 FECE 26   F6   FEC6            BNE    BRKP1
 FED0 C6   FF           BRKSWI  LDB    #$FF        not in table - user SWI
 FED2 34   04           FDBKPT  PSHS   B
 FED4 17   FA2F F906            LBSR   CARRTN      found - output "#"
 FED7 86   23                   LDA    #'#
 FED9 17   FA07 F8E3            LBSR   OUTCHR
 FEDC 35   02                   PULS   A           output breakpoint #
 FEDE 17   FA79 F95A            LBSR   PRTBYT
 FEE1 EF   6A                   STU    10,S        put adjusted PC back in stack
 FEE3 1F   43                   TFR    S,U
 FEE5 17   FA45 F92D            LBSR   PRTR1       output register contents
 FEE8 16   F9BC F8A7            LBRA   WARMS1      jump to debugger mainline
```

```
                        * Console I/O routines for serial interface

                        * Wait for input then return with char in ACCA

FEEB 8D   03   FEF0  SERIN   BSR    SERKEY
FEED 27   FC   FEEB          BEQ    SERIN     wait till SERKEY returns Z=0
FEEF 39                      RTS

                        * Return any pending input in ACCA; Z=0 if input found

FEF0 8D   19   FF0B  SERKEY  BSR    SERTST
FEF2 27   16   FF0A          BEQ    SERKYX    exit if no input waiting
FEF4 B6   E005               LDA    TERM+1    get the char
FEF7 84   7F                 ANDA   #$7F      mask off parity bit
FEF9 7D   E421               TST    >UPCASE
FEFC 27   0A   FF08          BEQ    SERKY8    force upperease if flag is set
FEFE 81   61                 CMPA   #'a
FF00 2D   06   FF08          BLT    SERKY8
FF02 81   7A                 CMPA   #'z
FF04 2E   02   FF08          BGT    SERKY8
FF06 84   5F                 ANDA   #$5F
FF08 1C   FB         SERKY8  ANDCC  #$FB      set Z=0 to indicate data
FF0A 39              SERKYX  RTS

                        * Test for pending input; Z=0 if so, else Z=1

FF0B 34   02         SERTST  PSHS   A
FF0D B6   E004               LDA    TERM
FF10 85   01                 BITA   #$1       check RxDRF
FF12 35   82                 PULS   A,PC

                        * Serial terminal output - character in ACCA

FF14 34   06         SEROUT  PSHS   D
FF16 F6   E004       SER1    LDB    TERM
FF19 C5   02                 BITB   #2
FF1B 27   F9   FF16          BEQ    SER1      wait for TXDRE
FF1D B7   E005               STA    TERM+1    put char in ACIA
FF20 7D   E436               TST    >DEST
FF23 26   04   FF29          BNE    HRDCPY    do hardcopy if flag is set
FF25 35   86         SERX    PULS   D,PC      else exit
```

```
                    * Printer output subroutines


                    * Print the char in ACCA on the hard copy device

FF27 34   06        HARDC   PSHS   D
FF29 8D   11  FF3C  HRDCPY  BSR    HRDOUT    output the char
FF2B 81   0D                CMPA   #CR
FF2D 26   0B  FF3A          BNE    HARDCX    if char was CR, output nulls
FF2F F6   E434              LDB    >NULLS
FF32 27   06  FF3A          BEQ    HARDCX    if null count=0, exit
FF34 4F                     CLRA
FF35 8D   05  FF3C  HRDCP1  BSR    HRDOUT    output # of nulls in NULLS
FF37 5A                     DECB
FF38 26   FB  FF35          BNE    HRDCP1
FF3A 35   86        HARDCX  PULS   D,PC      exit


                    * Actual printer output routine

FF3C 34   04        HRDOUT  PSHS   B
FF3E 7D   E435              TST    >PRTFLG   check printer type flag
FF41 26   0C  FF4F          BNE    PARLEL


                    * Serial printer routine

FF43 F6   E000      SERIAL  LDB    SPRINT    wait for TXDRE
FF46 C5   02                BITB   #2
FF48 27   F9  FF43          BEQ    SERIAL
FF4A B7   E001              STA    SPRINT+1  put char in ACIA
FF4D 35   84                PULS   B,PC      exit


                    * Parallel printer routine

FF4F 7D   E043      PARLEL  TST    PPRINT+1  wait for printer ready
FF52 2A   FB  FF4F          BPL    PARLEL
FF54 7D   E042      PARLL1  TST    PPRINT    clear IRQ flag
FF57 B7   E042              STA    PPRINT    put data in PiA
FF5A C6   36                LDB    #$36
FF5C F7   E043              STB    PPRINT+1  toggle handshake output
FF5F C6   3E                LDB    #$3E
FF61 F7   E043              STB    PPRINT+1
FF64 35   84                PULS   B,PC      exit
```

```
                    * Subcommand P of I command
                    * Set printer type - S=serial P=parallel

FF66 17   F97E F8E7 SETPRT  LBSR   GETUPS    input a character
FF69 81   53                CMPA   #'S
FF6B 27   18   FF85         BEQ    SETSRL    compare to S & P
FF6D 81   50                CMPA   #'P
FF6F 26   16   FF87         BNE    STPTOT    exit if neither
FF71 97   35                STA    <PRTFLG   set flag
FF73 8E   E042              LDX    #PPRINT   initialize PIA
FF76 6F   01                CLR    1,X
FF78 86   FF                LDA    #$FF      set DDR for output
FF7A A7   84                STA    0,X
FF7C 86   3E                LDA    #$3E      set up strobe
FF7E A7   01                STA    1,X
FF80 4F                     CLRA
FF81 34   02                PSHS   A         adjust stack for output routine
FF83 20   CF   FF54         BRA    PARLL1    exit thru output routine


                    * initialize for serial printer

FF85 0F   35        SETSRL  CLR    <PRTFLG   set flag
FF87 39             STPTOT  RTS              ACIA is set up by default
```

```
                        * Handler for Non-Maskable Interrupt
 FF88 1F   43           NMITRP  TFR     S,U       copy stack pointer
 FF8A 17   F9A0 F92D            LBSR    PRTR1     print register contents
 FF8D 8E   FF9F                 LDX     #NMIMSG
 FF90 17   F965 F8F8            LBSR    PRTST     output NMI message
 FF93 17   F947 F8DD            LBSR    INECHO    get a char
 FF96 84   DF                   ANDA    #$DF      force upper case
 FF98 81   59                   CMPA    #'Y       if Y, return to program
 FF9A 1026 F906 F8A4            LBNE    WARMS0    else go to GMXBUG
 FF9E 3B                DUMRTI  RTI
 FF9F 4E 4D 49 3A       NMIMSG  FCC     ;NMI: restart (Y/N)? ;,4

                        * DAT initialization
                        * Initializes all 16 tasks to bank 0, physical, with all
                        * 4K segments at the same physical & logical addresses.
                        * SBUG-E DAT image is maintained.
 FFB4 8E   FFF0         RESETI  LDX     #$FFF0
 FFB7 108E DFD0                 LDY     #$DFD0
 FFBB C6   0F                   LDB     #15
 FFBD F7   FF7F         RESET1  STB     TSR       select a task
 FFC0 86   0F                   LDA     #$0F      assign each 4k its own
 FFC2 A7   86           RESET2  STA     A,X       top 4 bits
 FFC4 4A                        DECA
 FFC5 2A   FB   FFC2            BPL     RESET2
 FFC7 E7   A0                   STB     0,Y+      do same for SBUG image
 FFC9 5A                        DECB
 FFCA 2A   Fl   FFBD            BPL     RESET1
 FFCC CC   F1F0                 LDD     #$F1F0    set SBUG ext addr bits for
 FFCF ED   3E,                  STD     -2,Y      $E000-$F000 to $F
 FFD1 16   F85A F82E            LBRA    COLD

                        * Jumps to soft interrupt vector addresses
 FFD4 6E   9F   DFC2    SW3JMP  JMP     [SW3VEC]
 FFD8 6E   9F   DFC4    SW2JMP  JMP     [SW2VEC]
 FFDC 6E   9F   DFC6    FRQJMP  JMP     [FIRVEC]
 FFE0 6E   9F   DFC8    IRQJMP  JMP     [IRQVEC]
 FFE4 6E   9F   DFCA    SWIJMP  JMP     [SWIVEC]
 FFE8 6E   9F   E40A    NMIJMP  JMP     [NMIVEC]
                FFEC    ENDIT   EQU     *

                        * Absolute interrupt vectors
 FFF0                           ORG     VECTOR
 FFF0 0000              RESERV  FDB     0000      reserved for future use
 FFF2 FFD4              SWI3    FDB     SW3JMP
 FFF4 FFD8              SWI2    FDB     SW2JMP
 FFF6 FFDC              FIRQ    FDB     FRQJMP
 FFF8 FFE0              IRQ     FDB     IRQJMP
 FFFA FFE4              SWI     FDB     SWIJMP
 FFFC FFE8              NMI     FDB     NMIJMP
 FFFE FFB4              RESET   FDB     RESETI

0 ERROR(S) DETECTED
```

SYMBOL TABLE:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ARITHM | FAA8 | ASCII | FD18 | BEGIN1 | E422 | BEGIN2 | E426 |
| BINARY | FD00 | BINCVT | F9A7 | BINHEX | F822 | BINRY1 | FD02 |
| BLKMOV | FDE6 | BREAKX | FE6A | BRKPJ | FEC6 | BRKPT | FEBC |
| BRKSET | FE4F | BRKSWI | FED0 | BRKTBL | E410 | BS | 0008 |
| BUMP | FCF4 | CARRTN | F906 | CHANGE | FCFB | CHGBMP | FCF2 |
| CHGMSG | FD71 | CHKSM1 | FAC1 | CHKSM2 | FAC9 | CHKSM3 | FAD1 |
| CHKSUM | FAB9 | CODTBL | FA6E | COLD | F82E | COLDJ | F878 |
| COLD2 | F885 | CONVRT | F9B1 | CR | 000D | CTLADR | DFDA |
| CTLS | 0013 | DDUMP | FAE7 | DDUMP1 | FAF3 | DDUMP2 | FAFD |
| DDUMP3 | FB06 | DDUMP4 | FB15 | DDUMP6 | FB1D | DDUMP7 | FB1F |
| DEST | E436 | DPR | 00E4 | DROP | FCF6 | DRPRTS | FCF8 |
| DSPLAY | FD04 | DSPLY1 | FD0A | DSPLY2 | FD0F | DUMRTI | FF9E |
| END1 | E424 | ENDIT | FFEC | ENDLIN | E40E | EOLCH1 | FB56 |
| EOLCH2 | FB5F | EOLCH3 | FB6C | EOLCHK | FB48 | ERROR | FDD5 |
| ESC | 001B | ESCOUT | FB5A | FDBKPT | FED2 | FDUM50 | FBBA |
| FDUM55 | FBC1 | FDUM58 | FBD7 | FDUM60 | FBD8 | FDUM90 | FBEC |
| FDUM91 | FBEE | FDUMP | FB71 | FDUMP1 | FB77 | FDUMP2 | FB83 |
| FDUMP3 | FB9D | FDUMP4 | FBEB | FDUMP6 | FBB1 | FIRQ | FFF6 |
| FIRVEC | DFC6 | FLXWRM | FCD3 | FRQJMP | FFDC | GETADS | FB33 |
| GETHEX | F967 | GETHX1 | FB42 | GETHX3 | F97A | GETHXX | FCC0 |
| GETKEY | F8F4 | GETLIN | F9BD | GETLN1 | F9C3 | GETLN3 | F9D8 |
| GETLN5 | F9DF | GETLN6 | F9E4 | GETLNL | F9BA | GETUPS | F8E7 |
| GMXBUG | F800 | GMXHED | F9FD | GOTO | FEAF | GOTOJ | FEB9 |
| HARDC | FF27 | HARDCX | FF3A | HEX | FD23 | HEXBIN | F824 |
| HEXCVT | F980 | HEXIN | F820 | HEXLOC | FBFE | HRDCP1 | FF35 |
| HRDCPY | FF29 | HRDOUT | FF3C | HXCVT0 | F982 | HXCVT1 | F986 |
| HXCVT2 | F98A | HXCVT3 | F998 | HXCVT9 | F97C | HXL0C1 | FC0A |
| HXLGC2 | FC1C | HXL0C3 | FC21 | HXL0C4 | FC22 | HXL0C5 | FC35 |
| HXL0C6 | FC2D | HXL0C7 | FC3D | INCAPS | F81E | INCH | F804 |
| INCHAR | F8D9 | INCHE | F806 | INCHEK | F808 | INECHO | F8DD |
| INECX | F8EF | INIT | FC3E | INITAB | FC4F | INKEY | F81C |
| INTEST | F8F0 | INVEC | E400 | IRQ | FFF8 | IRQJMP | FFE0 |
| IRQVEC | DFC8 | JUMP | FCC3 | JUMP1 | FCC7 | KEYVEC | E402 |
| KILAL1 | FE8F | KILALL | FE8A | KILBRK | FE6B | KLBRK1 | FE75 |
| KLBRK2 | FE83 | LF | 000A | LINBUF | E40C | LINEIN | F826 |
| LOKOUT | F9F8 | LOOKUP | F9EC | LRA | F812 | LRAOUT | F91A |
| LREAL | F912 | MAKBYT | FDCC | MATCH1 | E437 | MATCH2 | E438 |
| MATCH3 | E439 | MEMODE | FCE8 | MEMOFF | FCEF | MEMPTR | E430 |
| MEMTBL | FA56 | MKBRTS | FDD4 | MMODE | E420 | MONITR | F800 |
| MOVBLK | F82A | MOVER0 | FE23 | MOVER1 | FDEE | NMI | FFFC |
| NMIJMP | FFE8 | NMIMSG | FF9F | NMITRP | FF88 | NMIVEC | E40A |
| NORMAL | FE0C | NORML1 | FE12 | NULLS | E434 | NXTCMD | F802 |
| OS9 | FC9B | OS91 | FCA4 | OS9CMD | FC99 | OUT2HS | FAE1 |
| OUT4HS | FADC | OUTCH | F80A | OUTCHR | F8E3 | OUTSPC | FAE4 |
| OUTVEC | E406 | PARLEL | FF4F | PARLL1 | FF54 | PBYTE | F81A |
| PCRLF | F80E | PDATA | F80C | PICKBR | FE97 | PICKBX | FEA8 |
| POINTR | E432 | PPRINT | E042 | PREGS | F818 | PRINT | F82C |
| PROMPT | FA23 | PRTB1 | F95C | PRTBK1 | FE31 | PRTBRK | FE2D |
| PRTBYT | F95A | PRTDAT | F8FA | PRTDT1 | F8FC | PRTDT2 | F910 |
| PRTFLG | E435 | PRTOUT | F9F9 | PRTR1 | F92D | PRTREG | F923 |
| PRTRG1 | F93D | PRTRG2 | F949 | PRTSPC | F91B | PRTST | F8F8 |
| PRTVEC | E408 | PSPACE | F816 | PSTRNG | F810 | REGDMP | FD2B |
| REGID | FA28 | REGPRT | F92B | REGTBL | FD86 | REPORT | FC6B |
| REPRT1 | FC81 | REPRT2 | FC8D | RESERV | FFF0 | RESET | FFFE |
| RESET1 | FFBD | RESET2 | FFC2 | RESETI | FFB4 | RGD15 | FD70 |
| RGDMP1 | FD3E | RGDMP2 | FD4B | RGDMP3 | FD62 | RGDMP5 | FD6E |
| SBUGS | DFC0 | SCRATC | E400 | SER1 | FF16 | SERIAL | FF43 |
| SERIN | FEEB | SERKEY | FEF0 | SERKY8 | FF08 | SERKYX | FF0A |
| SEROUT | FF14 | SERTST | FF0B | SERX | FF25 | SETDST | FC5C |
| SETNUL | FC66 | SETPRT | FF66 | SETSRL | FF85 | SETUP | F864 |
| SPC | 0020 | SPCIL1 | FCBA | SPCMSG | FB2D | SPRINT | E000 |
| STDST1 | FC63 | | | | | | |

```
STPTOT FF87   SUBTOT E42A   SVCLIM DFCE   SVCORG DFCC   SW2JMP FFD8
SW2VEC DFC4   SW3JMP FFD4   SW3VEC DFC2   SWI    FFFA   SWI2   FFF4
SWI3   FFF2   SWIJMP FFE4   SWITCH FCAE   SWIVEC DFCA   SYSTAT FC90
TBSRCH F828   TERM   E004   TESTM1 FD9D   TESTM2 FDA2   TESTM3 FDAC
TSR    FF7F   TSTM1A FDA0   TSTMEM FD99   TSTVEC E404   UPCASE E421
UPSIL1 FCBD   USER   FCCC   VECTOR FFF0   VIDPRM F400   WAIT   E42F
WARM25 F8C4   WARMS  F894   WARMS0 F8A4   WARMS1 F8A7   WARMS2 F8BF
WARMS3 F8D0   ZAPMEM FCDA   ZPMEM1 FCE1
```