Using the OS-9 SPPT/GMXBUG V3.x Monitor PROM

The OS-9 SPPT/GMXBUG V3.x monitor PROM provides a means of running both OS-9 GMX IIIs (Support ROM version of OS-9 GMX III) and GMXBUG/FLEX on systems using the GIMIX 6809 CPU III. Switching between the two operating systems can be accomplished under software control or by hardware jumper.

The SPPT/GMXBUG3.x monitor PROM must be configured for either the I/O processor or standard ACIA by selection of a jumper option on the CPU III. If an I/O processor is used, a jumper block must be installed at JA-9 on the CPU III. If operation with a standard ACIA is required, remove the jumper block at JA-9. JA-9 controls a status bit on the CPU that is read by the monitors to select the I/O mode. The OS-9 operating system itself is supplied factory configured for either a 3-Port serial I/O processor or a 2-port ACIA interface (in I/O slot 0 on the motherboard) as its console I/O port. The FLEX operating system uses GMXBUG for its console I/O and does not need special configuration for console I/O type. However, the appropriate printer driver (PRINT.SYS) must be used if FLEX is to be used with a printer (see the FLEX manual and the file "READ-ME.TXT" on the FLEX disk.

An additional jumper on the CPU III is used to select the power-up operating mode of the system. There are four possible modes that may be selected.

1: Enter GMXBUG on power-up, select SPPT under software control
2: Enter SPPT on power-up, select GMXBUG under software control
3: Enter GMXBUG on power-up, SPPT locked out
4: Enter SPPT on power-up, GMXBUG locked out

Jumper area JA-8 on the CPU III selects the desired operating mode. The following figures show the jumper configuration for the each mode.

```
                JA-8                                JA-8
            ----------------                    ----------------
Mode 1:     [ O  O  *==*  O ]       Mode 2:     [ O  *==*  O  O ]
            ----------------                    ----------------

                JA-8                                JA-8
            ----------------                    ----------------
Mode 3:     [ O  O  O  *==* ]       Mode 4:     [ *==*  O  O  O ]
            ----------------                    ----------------
```

== : indicates jumper location

Booting the Disk Operating System

When mode 1 or mode 3 is selected, the system enters the GMXBUG monitor on power-up or after a system Reset. The appropriate GMXBUG command can then be used to boot FLEX (U) or enter the OS-9 monitor (O) if mode 1 is selected. See the GMXBUG manuals for information on other available commands. Before entering the FLEX boot command (U) the appropriate operating system disk should be inserted in drive #0 (the left hand drive in systems with more than one drive). To switch from FLEX to OS-9 (mode 1 only), return to GMXBUG using the FLEX "MON" command and use the "O" command. To switch from OS-9 to GMXBUG/FLEX (mode 1 only) the system must be reset. OS-9 can not be selected when the system is configured for mode 3.

In modes 2 and 4, the OS9 monitor (SPPT) is entered on power-up or after a Reset. The appropriate Support ROM commands can then be used to boot OS-9 (F or W) or enter the GMXBUG monitor (G) if mode 2 is selected. See the Support ROM manual for information on boot disk requirements and other available commands. When booting from a floppy disk (F command) the OS-9 system disk should be inserted in drive #0 as above. Note: In order to boot OS-9 with the Support ROM, the special GMX IIIs version of OS-9 must be used. To switch from OS-9 to GMXBUG/FLEX (mode 2 only) the system must be reset and the Support ROM "G" command used. GMXBUG/FLEX can not be selected when when the system is configured for mode 4.

GMXBUG/FLEX and Extended Memory

GMXBUG and FLEX do not directly support extended addressing or the use of memory beyond 64K (56K of RAM). However, the GMX VDISK program (supplied with some systems or available separately from GIMIX) allows the use of extended memory as a "virtual disk" with FLEX.

GMXBUG commands that access memory (see the GMXBUG manual) only operate on memory in the first address bank (extended address $0). Memory on other banks ($1-$F) is not accessible to GMXBUG. Note: The OS-9 Support ROM includes memory tests that can access and test the entire extended address space.

GMXBUG-09 System Monitor
Version 3

User's Guide

GIMIX Inc.
1337 W. 37th Place
Chicago, IL 60609
312-927-5510 * TWX 910-221-4055

GMXBUG Manual Addenda for GMXBUG Version 3.2

This version of GMXBUG is modified to support software switching between OS-9 GMX III and GMXBUG/FLEX on the GMX CPU III. The listing GMXBUG32 which is included shows the code which is overlaid on standard GMXBUG 2.2F to create GMXBUG 3.2. This code provides the setup of the DAT. It also establishes a handler and soft vector ($E460) for the Trap interrupt. This vector points to the GMXBUG warmstart address after Reset. The "O" command for switching to OS-9 is also slightly modified, and the "header" message displayed at GMXBUG warmstart is different.

GMXBUG support for the 3-Port I/O Processor

The IOPGMX extension to GMXBUG allows GMXBUG to use a 3-Port IOP for console I/O. A jumper on the CPU III is used to inform GMXBUG 3.2 of the type of I/O device (Intelligent or Standard ACIA) being used.

IOPGMX provides single character input and output to a serial device connected to port 1 of a 3-Port IOP in slot 0 of the SS-30 bus. This device will function as the GMXBUG console. All monitor calls including INCH, INCHE, INCHEK, INKEY, INCAPS, and OUTCH will function normally.

In addition, IOPGMX can drive a serial printer connected to port 2 of the same IOP. All output through the GMXBUG call PRINT will be sent to this device, as will any console output when the DEST flag is set for both console and printer. IOPGMX will also drive a parallel printer in slot 0, side B ($E042) if the printer type flag (PTRFLG) is set for a parallel-type printer.

FLEX output to the printer is provided by the program PRINTIOP.SYS, which drives a serial printer. PRINTPLL.SYS will drive a parallel printer and is independent. Source listings of these programs are included on the system disk.

IOPGMX also modifies the NMI handling routine for compatibility with IOP operation. The hardware NMI and IRQ inhibit function of the Gimix CPU III is used during some parts of the console and printer I/O operations, and after an NMI has occurred until the "Resume Y/N?" query has been answered. A flag variable at $E462 is used for this.

This NMI inhibition is done by manipulating the CPU III Task Select Register. However, this has a side effect: the attribute enable bits in the TSR are left set to 0 after any call to the IOPGMX console or printer I/O routines. GMXBUG/FLEX programs which use the memory attributes will not work with IOPGMX unless a copy of the TSR value is preserved elsewhere and the TSR is rewritten after any I/O call. This is needed only with GMXBUG/IOPGMX; OS-9 applications and operations are not affected in any way.

```
                        * GMXBG32

                        * GMXBUG interrupt handlers and DAT initialization for
                        * CPU-III board with vector for UAM and WPR trapping

                        * version to set up DAT for switching system
                        * also includes patch for new monitor switching circuit
                        * and new header string

              DFC2      SW3VEC  EQU     $DFC2
              DFC4      SW2VEC  EQU     $DFC4       soft interrupt vectors
              DFC6      FIRVEC  EQU     $DFC6
              DFC8      IRQVEC  EQU     $DFC8
              DFCA      SWIVEC  EQU     $DFCA
              E40A      NMIVEC  EQU     $E40A
              E460      TRPVEC  EQU     $E460       vector for UAM/WPR/WDT/SST trap
              E280      TSR     EQU     $E280       task select/control register
              F800      DATRAM  EQU     $F800       start of DAT memory
              F82E      COLD    EQU     $F82E       GMXBUG cold start


F9FD                            ORG     $F9FD

F9FD 47 4D 58 42      HEADER    FCC     /GMXBUG-09  V3.2/,13,10
FA0E 28 43 29 20              2 FCC     /(C) 1983 Gimix Inc/,13,10,4
              FA23      HDREND  EQU     *


                        * patch to monitor switching routine

FCB3                            ORG     $FCB3
FCB3 B7   E281                  STA     TSR+1       write to TSR+1

                        * Note: DAT is automatically available at power-up at
                        * $F800.  However this goes away on the first write to
                        * an odd address in the DAT. Therefore the first write
                        * to the DAT must be to $F83E-$F83F to set up the monitor
                        * in task 0. Then the monitor reset routine can finish
                        * setting up the rest of task 0.

FFB4                            ORG     $FFB4

FFB4 8E   F840        RESETI    LDX     #DATRAM+64 reset DAT - map task 0 to
FFB7 CC   01FF                  LDD     #$1FF       bank 15, segment to block
FFBA ED   83          RES1      STD     0,--X       from the top down
FFBC 5A                         DECB                for the top 8K
FFBD 8C   F838                  CMPX    #DATRAM+56 ($E000-$FFFF)
FFC0 26   F8   FFBA             BNE     RES1
FFC2 83   01E0                  SUBD    #$1E0       map the rest of task 0
FFC5 ED   83          RES2      STD     0,--X       ($E000-$FFFF) to bank 15
FFC7 5A                         DECB
FFC8 2A   FB   FFC5             BPL     RES2
FFCA 8E   F82E                  LDX     #COLD
```

```
FFCD BF   E460              STX    TRPVEC    set default ERRTRP value
FFD0 6E   84                JMP    0,X       go to main GMXBUG coldstart

FFD2 6E   9F E460   TRPHND  JMP    [TRPVEC]
FFD6 6E   9F DFC2   SW3HND  JMP    [SW3VEC]
FFDA 6E   9F DFC4   SW2HND  JMP    [SW2VEC]
FFDE 6E   9F DFC6   FIRHND  JMP    [FIRVEC]  handlers to use soft vectors
FFE2 6E   9F DFC8   IRQHND  JMP    [IRQVEC]
FFE6 6E   9F DFCA   SWIHND  JMP    [SWIVEC]
FFEA 6E   9F E40A   NMIHND  JMP    [NMIVEC]

             FFEE   ENDCOD  EQU    *

FFF0                        ORG    $FFF0

FFF0 FFD2                   FDB    TRPHND    system trap vector (ex-reserved)
FFF2 FFD6                   FDB    SW3HND
FFF4 FFDA                   FDB    SW2HND
FFF6 FFDE                   FDB    FIRHND    hard interrupt vectors
FFF8 FFE2                   FDB    IRQHND
FFFA FFE6                   FDB    SWIHND
FFFC FFEA                   FDB    NMIHND
FFFE FFB4                   FDB    RESETI

                            END
```

0 ERROR(S) DETECTED

SYMBOL TABLE:

```
COLD   F82E   DATRAM F800   ENDCOD FFEE   FIRHND FFDE   FIRVEC DFC6
HDREND FA23   HEADER F9FD   IRQHND FFE2   IRQVEC DFC8   NMIHND FFEA
NMIVEC E40A   RES1   FFBA   RES2   FFC5   RESETI FFB4   SW2HND FFDA
SW2VEC DFC4   SW3HND FFD6   SW3VEC DFC2   SWIHND FFE6   SWIVEC DFCA
TRPHND FFD2   TRPVEC E460   TSR    E280
```

```
                    * IOPGMX

                    * This program resides at $F400 in the GMXBUG 3.2 ROM,
                    * It supports a serial terminal as console on port 1,
                    * and a serial device as printer on port 2. IOPGMX
                    * includes subroutines to support the GMXBUG calls
                    * INCH, INCHEK, OUTCH, and PRINT, and also handles
                    * the "force upper case" and "output to both screen
                    * and printer" features.  The special general IRQ
                    * mask of the CPU III is used to avoid the conflicts
                    * that would result if an NMI, IRQ, or FIRQ were to
                    * occur during console or printer I/O.  However, this
                    * can cause spurious NMIs.  To avoid this a modified
                    * NMI handler is provided.

                    * IOPGMX also includes a patch to the "O" command for
                    * switching to OS-9; this patch makes sure that the IOP
                    * is in a state of reset when OS-9 comes up.

                    * Because of this patch code, the binary image of this
                    * program MUST overlay GMXBUG 3.2, rather than vice versa.

F806    INCHE   EQU     $F806
F810    PSTRNG  EQU     $F810
F86E    RENTER  EQU     $F86E       reentry address for GMXBUG 3.2
F83B    RENTR1  EQU     $F83B       reentry address for normal init
F8A4    WARMSO  EQU     $F8A4       addresses for NMI handler
F92D    PRTR1   EQU     $F92D
E280    TSR     EQU     $E280       Task Status Register of CPU III
E400    INVEC   EQU     $E400       console input vector
E421    UPCASE  EQU     $E421       force uppercase input flag
E435    PTRFLG  EQU     $E435       printer type flag
E436    DEST    EQU     $E436       printer output flag
E462    NMICTL  EQU     $E462       NMI inhibit control flag

2000    DRVADR  EQU     $2000       load address for driver code
2000    DRVTFR  EQU     $2000       transfer address for driver
0077    DRVSIZ  EQU     $77         size of driver code

E000    FIODAT  EQU     $E000       address of FIO
E001    FIOCTL  EQU     $E001
E002    FIOIRQ  EQU     $E002

E042    PPRINT  EQU     $E042       address of printer PIA

                    * offsets for indirect addressing of FIO registers
0000    CTLRGO  EQU     0           reset/mode control
0001    CTLRG1  EQU     1           mailbox/REQ control
0002    ISTATO  EQU     2           message IRQ ctl
0009    CTLRG2  EQU     9           port 2 control
000B    MSGOUT  EQU     11          outgoing message byte
000C    MSGIN   EQU     12          incoming message byte
```

```
F400                            ORG     $F400       address for alternate I/O ROM

F400 12                 START   NOP                 NOP at $F400 is flag to GMXBUG
F401 20   06   F409            BRA     BEGIN

                        * These two vectors are set up for use by PRINT.SYS
F403 16   0175 F57B     PRTTST  LBRA    IOPCHP      vector to IOP printer check
F406 16   013E F547     PRTOUT  LBRA    IOPPRT      vector to IOP printer output


F409 B6   E280          BEGIN   LDA     TSR         check CPU III sense bit (JA-9)
F40C 85   10                    BITA    #$10        if sense bit is 1, (jumper open)
F40E 1026 0429 F83B             LBNE    RENTR1      resume normal initialization

F412 B7   E002                  STA     FIOIRQ
F415 B6   E001                  LDA     FIOCTL      reset sequence for FIO
F418 B6   E001                  LDA     FIOCTL
F41B 86   00                    LDA     #CTLRG0
F41D C6   01                    LDB     #1          toggle FIO reset bit
F41F 17   00A4 F4C6             LBSR    STUFF
F422 5F                         CLRB
F423 17   00A0 F4C6             LBSR    STUFF
F426 B6   E001                  LDA     FIOCTL
F429 86   00                    LDA     #CTLRG0
F42B C6   14                    LDB     #$14        set non-Z-bus, vector w/status
F42D 17   0096 F4C6             LBSR    STUFF
F430 86   09                    LDA     #CTLRG2
F432 C6   01                    LDB     #1
F434 17   008F F4C6             LBSR    STUFF       enable IOP side

F437 86   05            WAIT    LDA     #5
F439 8E   F424          WAIT1   LDX     #62500      delay 1.25 seconds for
F43C 30   1F            WAIT2   LEAX    -1,X        IOPROM to complete its setup
F43E 26   FC   F43C             BNE     WAIT2       and enter the wait loop
F440 4A                         DECA
F441 26   F6   F439             BNE     WAIT1

F443 C6   F0                    LDB     #$F0        tell IOP to jump to GMX loader
F445 8D   61   F4A8             BSR     SEND

F447 86   14                    LDA     #20         delay 100 cycles for
F449 4A                 WAIT3   DECA                loader internal setup
F44A 26   FD   F449             BNE     WAIT3

F44C C6   55            HEADBT  LDB     #$55        send start-of-record byte
F44E 8D   58   F4A8             BSR     SEND
F450 8D   63   F4B5             BSR     RECEIV
F452 C1   55                    CMPB    #$55        get $55 back from IOP
F454 26   F6   F44C             BNE     HEADBT

F456 C6   20                    LDB     #DRVADR/256 send load address
F458 8D   4E   F4A8             BSR     SEND
F45A CC   2000                  LDD     #DRVADR
```

```
F45D 8D   49   F4A8          BSR    SEND

F45F C6   00                 LDB    #DRVSIZ/256 send byte count
F461 8D   45   F4A8          BSR    SEND
F463 CC   0077               LDD    #DRVSIZ
F466 8D   40   F4A8          BSR    SEND

F468 C6   20                 LDB    #DRVTFR/256 send transfer address
F46A 8D   3C   F4A8          BSR    SEND
F46C CC   2000               LDD    #DRVTFR
F46F 8D   37   F4A8          BSR    SEND

F471 30   8D 0180            LEAX   DRIVER,PCR send the block of code
F475 108E 0077               LDY    #DRVSIZ
F479 E6   80          SNDBLK LDB    0,X+        get a byte
F47B 8D   2B   F4A8          BSR    SEND        send it
F47D 31   3F                 LEAY   -1,Y        end of block?
F47F 26   F8   F479          BNE    SNDBLK      continue if not

                      * IOP will now be running in GMXBUG driver mode
F481 CE   E400               LDU    #INVEC      point to GMXBUG vector area
F484 CC   F4CD               LDD    #IOPINP
F487 ED   C1                 STD    0,U++
F489 CC   F4E9               LDD    #IOPKEY     set up console I/O vectors
F48C ED   C1                 STD    0,U++
F48E CC   F4D7               LDD    #IOPTST
F491 ED   C1                 STD    0,U++
F493 CC   F516               LDD    #IOPOUT
F496 ED   C1                 STD    0,U++
F498 CC   F547               LDD    #IOPPRT     set up printer vector
F49B ED   C1                 STD    0,U++
F49D CC   F5B2               LDD    #IOPNMI     set up NMI handler
F4A0 ED   C1                 STD    0,U++
F4A2 7F   E462               CLR    NMICTL      enable NMI reenable
F4A5 7E   F86E               JMP    RENTER      reenter GMXBUG

                      * send a byte (in ACCB) to the slave
F4A8 86   0B          SEND   LDA    #MSGOUT
F4AA 8D   1A   F4C6          BSR    STUFF       send the byte
F4AC 86   01          SEND1  LDA    #CTLRG1     wait till last byte is accepted
F4AE 8D   0F   F4BF          BSR    GRAB
F4B0 C5   20                 BITB   #$20
F4B2 26   F8   F4AC          BNE    SEND1
F4B4 39                      RTS

                      * receive a byte through the 8038 mailbox from the slave
F4B5 86   02          RECEIV LDA    #ISTAT0     get Interrupt Status 0
F4B7 8D   06   F4BF   RECV1  BSR    GRAB
F4B9 C5   20                 BITB   #$20        check Message IP
F4BB 27   FA   F4B7          BEQ    RECV1       wait till it's set
F4BD 86   0C                 LDA    #MSGIN      get Message byte

                      * read the 8038 register pointed to by ACCA
F4BF B7   E001        GRAB   STA    FIOCTL      point to the register
```

```
F4C2 F6   E001          LDB    FIOCTL    read it
F4C5 39                 RTS

                * put the value in ACCB in the register pointed to by ACCA
F4C6 B7   E001  STUFF   STA    FIOCTL    point to the register
F4C9 F7   E001          STB    FIOCTL    write to it
F4CC 39                 RTS

                * INCH equivalent - get input from console
F4CD 34   04    IOPINP  PSHS   B
F4CF 8D   06    F4D7    BSR    IOPTST    check console 6551 status
F4D1 27   FC    F4CF    BEQ    IOPIN1    if no data, continue to wait
F4D3 8D   20    F4F5    BSR    IOPGET    else get character
F4D5 35   84            PULS   B,PC      and exit

                * INCHEK equivalent - test for console input waiting
F4D7 34   06    IOPTST  PSHS   D
F4D9 17   00BD  F599    LBSR   MASK      mask interrupts
F4DC C6   01            LDB    #1        get status of console 6551
F4DE 8D   C8    F4A8    BSR    SEND
F4E0 8D   D3    F4B5    BSR    RECEIV
F4E2 17   00BF  F5A4    LBSR   UNMASK    unmask interrupts
F4E5 C5   08            BITB   #8        test RxDRF bit
F4E7 35   86            PULS   D,PC      return Z=1 or Z=0

                * INKEY equivalent - get input from console without waiting
F4E9 34   04    IOPKEY  PSHS   B
F4EB 8D   EA    F4D7    BSR    IOPTST    test for input waiting
F4ED 27   04    F4F3    BEQ    IOPK1     if none, exit with Z=1
F4EF 8D   04    F4F5    BSR    IOPGET    else get input and exit
F4F1 1C   FB            ANDCC  #$FB      with Z=0 (input found)
F4F3 35   84    IOPK1   PULS   B,PC

                * get a character from console through IOP
F4F5 34   04    IOPGET  PSHS   B
F4F7 17   009F  F599    LBSR   MASK      mask interrupts
F4FA C6   05            LDB    #5        get data
F4FC 8D   AA    F4A8    BSR    SEND
F4FE 8D   B5    F4B5    BSR    RECEIV
F500 17   00A1  F5A4    LBSR   UNMASK    unmask interrupts
F503 1F   98            TFR    B,A       move data to ACCA
F505 7D   E421          TST    UPCASE    force upper case flag set?
F508 27   0A    F514    BEQ    IOPG1
F50A 81   61            CMPA   #'a       if so, is input a-z?
F50C 2D   06    F514    BLT    IOPG1
F50E 81   7A            CMPA   #'z
F510 2E   02    F514    BGT    IOPG1
F512 84   DF            ANDA   #$DF      if so, convert to upper case
F514 35   84    IOPG1   PULS   B,PC      restore ACCB and exit

                * OUTCH equivalent - send character to console through IOP
F516 34   06    IOPOUT  PSHS   D
F518 17   007E  F599    IOPO1  LBSR   MASK      mask interrupts
F51B C6   01            LDB    #1        get console 6551 status
```

```
F51D 8D   89   F4A8          BSR    SEND
F51F 8D   94   F4B5          BSR    RECEIV
F521 17   0080 F5A4          LBSR   UNMASK     unmask interrupts
F524 C5   10                 BITB   #$10       test TxDRE
F526 27   F0   F518          BEQ    IOPO1      if 0 continue to wait
F528 C5   60                 BITB   #$60       test DCD and DSR bits also
F52A 26   EC   F518          BNE    IOPO1      if either not 0, wait
F52C 17   006A F599          LBSR   MASK       mask interrupts
F52F C6   03                 LDB    #3
F531 17   FF74 F4A8          LBSR   SEND
F534 E6   E4                 LDB    0,S        output byte to console
F536 17   FF6F F4A8          LBSR   SEND
F539 17   0068 F5A4          LBSR   UNMASK     unmask interrupts
F53C 7D   E436               TST    DEST       also to printer?
F53F 27   04   F545          BEQ    IOPO2
F541 A6   E4                 LDA    0,S
F543 8D   02   F547          BSR    IOPPRT     if so do it
F545 35   86          IOPO2  PULS   D,PC        restore ACCs and exit

                      * PRINT equivalent - send character to printer through IOP
F547 34   06          IOPPRT PSHS   D
F549 7D   E435               TST    PTRFLG     check for paraller printer
F54C 26   16   F564          BNE    PARLEL     if so branch to parallel driver
F54E 8D   2B   F57B  IOPPR1  BSR    IOPCHP     check printer 6551 status
F550 27   FC   F54E          BEQ    IOPPR1     wait till printer is ready
F552 17   0044 F599          LBSR   MASK       mask interrupts
F555 C6   04                 LDB    #4
F557 17   FF4E F4A8          LBSR   SEND
F55A E6   E4                 LDB    0,S        output data byte to printer
F55C 17   FF49 F4A8          LBSR   SEND
F55F 17   0042 F5A4          LBSR   UNMASK     unmask interrupts
F562 35   86                 PULS   D,PC       restore regs and exit

                      * Parallel printer routine
F564 7D   E043        PARLEL TST    PPRINT+1   wait for printer ready
F567 2A   FB   F564          BPL    PARLEL
F569 7D   E042        PARLL1 TST    PPRINT     clear IRQ flag
F56C B7   E042               STA    PPRINT     put data in PiA
F56F C6   36                 LDB    #$36
F571 F7   E043               STB    PPRINT+1   toggle handshake output
F574 C6   3E                 LDB    #$3E
F576 F7   E043               STB    PPRINT+1
F579 35   86                 PULS   D,PC       exit

                      * subroutine for FLEX: sample printer status
                      * and return Z=0 if printer is ready
F57B 34   06          IOPCHP PSHS   D
F57D 17   0019 F599          LBSR   MASK       mask interrupts
F580 C6   02                 LDB    #2
F582 17   FF23 F4A8          LBSR   SEND       get status of printer 6551
F585 17   FF2D F4B5          LBSR   RECEIV
F588 17   0019 F5A4          LBSR   UNMASK     unmask interrupts
F58B C5   10                 BITB   #$10       test for TxDRE=1
F58D 27   08   F597          BEQ    IOPCH1     if 0 exit immediately with Z=1
```

```
F58F C5   60              BITB   #$60      else test for DCD=1 or DSR=1
F591 1F   A8              TFR    CC,A      and invert resulting Z-bit;
F593 88   04              EORA   #4        thus at exit
F595 1F   8A              TFR    A,CC      Z=0 IFF TxDRE=1 & DCD=0 & DSR=0
F597 35   86    IOPCH1    PULS   D,PC      restore ACCs and exit .

                * Mask all interrupts during FIO handshake operations
F599 B6   E280  MASK      LDA    TSR       get TSR value
F59C 84   08              ANDA   #8        preserve clock write enable bit
F59E 8A   01              ORA    #1        set TSR task bits to non-zero
F5A0 B7   E280            STA    TSR       value, masking all interrupts
F5A3 39                   RTS

                * Restore interrupts
F5A4 7D   E462  UNMASK    TST    NMICTL    is NMI active?
F5A7 26   08    F5B1      BNE    UNMSK2    if so leave it masked
F5A9 B6   E280            LDA    TSR       get TSR value
F5AC 84   08              ANDA   #8        preserve clock write enable bit
F5AE B7   E280            STA    TSR       clear task bits
F5B1 39         UNMSK2    RTS

                * revised handler for Non-Maskable Interrupt
F5B2 7C   E462  IOPNMI    INC    NMICTL    "inhibit" bogus extra NMIs
F5B5 1F   43              TFR    S,U       copy stack pointer
F5B7 BD   F92D            JSR    PRTR1     print register contents
F5BA 30   8D 001A         LEAX   NMIMSG,PCR
F5BE AD   9F F810         JSR    [PSTRNG]  output NMI message
F5C2 AD   9F F806         JSR    [INCHE]   get a char
F5C6 34   02              PSHS   A
F5C8 7F   E462            CLR    NMICTL    release NMI unmask control
F5CB 8D   D7    F5A4      BSR    UNMASK    unmask NMI
F5CD 35   02              PULS   A
F5CF 84   DF              ANDA   #$DF       force upper case
F5D1 81   59              CMPA   #'Y       Y??
F5D3 1026 02CD F8A4       LBNE   WARMSO    if not, go to GMXBUG
F5D7 3B                   RTI              else return to program
F5D8 4E 4D 49 3A  NMIMSG  FCC    ;NMI: restart (Y/N)? ;,4

                * Extension to "O" command in GMXBUG
F5ED B7   E002  OS9ADD    STA    FIOIRQ    reset the IOP if any
                * 1 line of code was replaced by JSR to here
                * must be performed before returning
F5F0 108E E500            LDY    #$E500    point to RAM for switch code
F5F4 39                   RTS

F5F5            DRIVER    RMB    DRVSIZ    driver code storage area

      026C      CODSIZ    EQU    *-START   total size of IOP support code

                * "O" command patch
FC9E                      ORG    $FC9E
FC9E BD   F5ED            JSR    OS9ADD    call to extension
FCA1 12                   NOP              replaced instruction was 4 bytes
```

END

0 ERROR(S) DETECTED

SYMBOL TABLE:

BEGIN	F409	CODSIZ	026C	CTLRGO	0000	CTLRG1	0001	CTLRG2	0009
DEST	E436	DRIVER	F5F5	DRVADR	2000	DRVSIZ	0077	DRVTFR	2000
FIOCTL	E001	FIODAT	E000	FIOIRQ	E002	GRAB	F4BF	HEADBT	F44C
INCHE	F806	INVEC	E400	IOPCH1	F597	IOPCHP	F57B	IOPG1	F514
IOPGET	F4F5	IOPIN1	F4CF	IOPINP	F4CD	IOPK1	F4F3	IOPKEY	F4E9
IOPNMI	F5B2	IOPO1	F518	IOPO2	F545	IOPOUT	F516	IOPPR1	F54E
IOPPRT	F547	IOPTST	F4D7	ISTATO	0002	MASK	F599	MSGIN	000C
MSGOUT	000B	NMICTL	E462	NMIMSG	F5D8	OS9ADD	F5ED	PARLEL	F564
PARLL1	F569	PPRINT	E042	PRTOUT	F406	PRTR1	F92D	PRTTST	F403
PSTRNG	F810	PTRFLG	E435	RECEIV	F4B5	RECV1	F4B7	RENTER	F86E
RENTR1	F83B	SEND	F4A8	SEND1	F4AC	SNDBLK	F479	START	F400
STUFF	F4C6	TSR	E280	UNMASK	F5A4	UNMSK2	F5B1	UPCASE	E421
WAIT	F437	WAIT1	F439	WAIT2	F43C	WAIT3	F449	WARMSO	F8A4

```
                        * IOPDRV

                        * This program resides in RAM on the IOP.  It supports
                        * a terminal as console on port 1 and a printer on port 2.

                0000    RAM1    EQU     0           2K RAM
                2000    RAM2    EQU     $2000
                C000    RAMROM  EQU     $C000       2K RAM or 2K/4K/8K ROM
                E000    ROM1    EQU     $E000       2K/4K/8K ROM

                        * addresses of DIP switch registers
                6000    SWTCH1  EQU     $6000
                4000    SWTCH2  EQU     $4000

                        * addresses of 6551 serial ports
                A004    PORT1   EQU     $A004
                A008    PORT2   EQU     $A008
                A010    PORT3   EQU     $A010

                        * offsets for 6551s
                0000    DATA51  EQU     0
                0001    STAT51  EQU     1
                0002    CMND51  EQU     2
                0003    CTRL51  EQU     3

                        * address of FIO
                8000    FIODAT  EQU     $8000
                8001    FIOCTL  EQU     $8001

                        * offsets for indirect addressing of FIO registers
                0000    CTLRG0  EQU     0           reset/mode control
                0001    CTLRG1  EQU     1           mailbox/REQ control
                0002    ISTAT0  EQU     2           message IRQ ctl
                0009    CTLRG2  EQU     9           port 2 control
                000B    MSGOUT  EQU     11          outgoing message byte
                000C    MSGIN   EQU     12          incoming message byte


2000                            ORG     $2000       place in same RAM as stack

2000 10CE 27FF          START   LDS     #$27FF      init stack pointer

2004 86   0B                    LDA     #$0B        no parity or IRQs, RTS & DTR low

2006 F6   6000                  LDB     SWTCH1      get baud rate for port 1
2009 C4   0F                    ANDB    #$0F
200B CA   10                    ORB     #$10        1 stop bit, 8 data bits, no echo
200D FD   A006                  STD     PORT1+CMND51 set up port 1 (console)

2010 F6   6000                  LDB     SWTCH1      get baud rate for port 2
2013 54                         LSRB
2014 54                         LSRB
2015 54                         LSRB
```

```
2016 54                         LSRB
2017 CA   10                    ORB     #$10        1 stop bit, 8 data bits, no echo
2019 FD   A00A                  STD     PORT2+CMND51 set up port 2 (printer)

                        * main execution loop
201C 8D   41   205F     DOFIFO  BSR     RECEIV      get a command byte from host
201E C1   05                    CMPB    #CMDLMT
2020 22   FA   201C            BHI     DOFIFO      do not accept undefined commands

2022 30   8D 0005               LEAX    CMDTBL,PCR
2026 58                         ASLB
2027 AD   95                    JSR     [B,X]       call selected routine
2029 20   F1   201C            BRA     DOFIFO      return and do next command

202B 203C               CMDTBL  FDB     NULL        null command
202D 2037                       FDB     CNSTAT      sample status of console ACIA
202F 203D                       FDB     PRSTAT      sample status printer ACIA
2031 2043                       FDB     CNSEND      send data to console
2033 2049                       FDB     PRSEND      send data to printer
2035 204F                       FDB     CNRECV      receive data from console

             000C       TBLSIZ  EQU     *-CMDTBL
             0005       CMDLMT  EQU     TBLSIZ/2-1

                        * get status of console
2037 F6   A005          CNSTAT  LDB     PORT1+STAT51 get status byte from 6551
203A 8D   16   2052            BSR     SEND        return it to host
203C 39                 NULL    RTS

                        * get status of printer
203D F6   A009          PRSTAT  LDB     PORT2+STAT51 get status byte from 6551
2040 8D   10   2052            BSR     SEND        return it to host
2042 39                         RTS

                        * output a character to console
2043 8D   1A   205F     CNSEND  BSR     RECEIV      get data byte from host
2045 F7   A004                  STB     PORT1+DATA51 put it in console 6551
2048 39                         RTS

                        * output a character to printer
2049 8D   14   205F     PRSEND  BSR     RECEIV      get data byte from host
204B F7   A008                  STB     PORT2+DATA51 put it in printer 6551
204E 39                         RTS

                        * receive character from console
204F F6   A004          CNRECV  LDB     PORT1+DATA51 get data byte from 6551

                        * send a byte (in ACCB) to the host
2052 86   0B            SEND    LDA     #MSGOUT
2054 8D   1A   2070            BSR     STUFF       send the byte
2056 86   01            SEND1   LDA     #CTLRG1     wait till last byte is accepted
2058 8D   0F   2069            BSR     GRAB
205A C5   20                    BITB    #$20
205C 26   F8   2056            BNE     SEND1
```

```
205E 39                         RTS

                        * receive a byte through the 8038 mailbox from the host
205F 86   02            RECEIV  LDA     #ISTAT0   get Interrupt Status 0
2061 8D   06   2069     RECV1   BSR     GRAB
2063 C5   20                    BITB    #$20      check Message IP
2065 27   FA   2061            BEQ     RECV1     wait till it's set
2067 86   0C                    LDA     #MSGIN    get Message byte

                        * read the 8038 register pointed to by ACCA
2069 B7   8001          GRAB    STA     FIOCTL    point to the register
206C F6   8001                  LDB     FIOCTL    read it
206F 39                         RTS

                        * put the value in ACCB in the register pointed to by ACCA
2070 B7   8001          STUFF   STA     FIOCTL    point to the register
2073 F7   8001                  STB     FIOCTL    write to it
2076 39                         RTS

             0077       CODSIZ  EQU     *-START

                                END

0 ERROR(S) DETECTED
```

SYMBOL TABLE:

CMDLMT	0005	CMDTBL	202B	CMND51	0002	CNRECV	204F	CNSEND	2043
CNSTAT	2037	CODSIZ	0077	CTLRG0	0000	CTLRG1	0001	CTLRG2	0009
CTRL51	0003	DATA51	0000	DOFIFO	201C	FIOCTL	8001	FIODAT	8000
GRAB	2069	ISTAT0	0002	MSGIN	000C	MSGOUT	000B	NULL	203C
PORT1	A004	PORT2	A008	PORT3	A010	PRSEND	2049	PRSTAT	203D
RAM1	0000	RAM2	2000	RAMROM	C000	RECEIV	205F	RECV1	2061
ROM1	E000	SEND	2052	SEND1	2056	START	2000	STAT51	0001
STUFF	2070	SWTCH1	6000	SWTCH2	4000	TBLSIZ	000C		