MOTOROLA

M68MODOS(D)

M68MODOS010 DISK OPERATING SYSTEM

User's Guide

SYSTEMS

MICROSYSTEMS

M68MODOS(D)
NOVEMBER 1978

M68MODOS010

DISK OPERATING SYSTEM USER'S GUIDE VERSION 3.0

First Edition
Copyright 1978 by Motorola Inc.

TABLE OF CONTENTS

I. INTRODUCTION

MODOS is essentially compatible with MDOS. The major changes have been to eliminate all of the EXbug dependencies from it and to provide an automatic start up capability. These changes allow MODOS to run without the Debug Module in the EXORciser. Other changes to MDOS have been made so that the OEM Manufacturer can customize the system to his hardware configuration. The address of the console ACIA can easily be changed. If, for example, the Manufacturer's console terminal runs off a PIA instead of an ACIA, he may substitute his own I/O routines for the ones provided with the system. His routines may reside in ROM or be loaded into RAM by MODOS during initialization. The user may provide interrupt handlers for NMI, SWI, and IRQ interrupts. Like the I/O routines, these handlers may reside in ROM or be loaded in by MODOS. In addition to providing his own driver for the console (keyboard/printer, punch, and reader), the user may substitute his own line printer driver.

MODOS can be initialized when power is applied to the system. The OEM disk PROM initiates the MODOS bootload process. If an error occurs during the load, the error status is printed on the console and the system hangs up. If the output character and console initialization routines in the PROM are not compatible with the hardware configuration, the OEM Manufacturer must modify the routines in PROM to satisfy his needs. These routines are in addition to the resident console I/O routines. The user can cause his I/O routines and/or interrupt handlers to be loaded in by MODOS by putting them in a file called MDOSUDRV.SY. If he does not do this, these routines must be in ROM.

When MODOS has completed initialization, it does one of two things depending on how the user has configured the system. Either a user selected sign-on message is printed, followed by a prompt, or a user selected command is loaded and control passed to it. This user initialization routine may be used to perform any special processing required for his system.

Although MODOS is not an end-customer development system, the need for the OEM Manufacturer to test his programs is recognized. This is accomplished by placing the MODOS software in an EXORciser/EXORdisk system. The user must change the jump vectors for all of the PROM entry points (disk and line printer drivers) and insure that his OEM memory map is compatible with the map for EXORdisk II or III MDOS. After this has been done, the Manufacturer can use EXbug to set breakpoints, display and change memory, etc.

The MODOS system as configured and sent to the customer can be run in an EXORciser/EXORdisk environment using EXbug.

II. USER CHANGEABLE VARIABLES

This section contains a description of each MODOS variable that the OEM Manufacturer can modify to tailor the system to his hardware requirements. Also included is the absolute address of each variable so that the user can patch the file MDOS.SY. For jump vectors, the address given is to the jump address, not the jump instruction. All locations and values are in hexadecimal unless otherwise stated.

NAME	LOCATION(S)	DESCRIPTION
SWI$SV	116-117	If the system has an SWI handler that should be envoked after MODOS have been overlaid by a command (using the LOAD;VG command), SWI$SV must contain the address of this routine. If SWI$SV has not been initialized (i.e., SWI$SV = 0), MODOS initializes the value to cause a jump to self in case an SWI occurs after MODOS has been overlaid. If EXbug is present (see XBUGF$ description), MODOS stores MAID's SWI handler address at SWI$SV, regardless of its initialized value. This means that the user's SWI handler cannot get control if MODOS is overlaid and EXbug is present. The SWI handler must reside in ROM or be included in the file MDOSUDRV.SY.
SWI$UV	118-119	If the system has user function calls (reference MDOS manual, system functions), SWI$UV must contain the address of the user function call handler. If SWI$UV has not been initialized (i.e. SWI$UV = 0), MODOS sets up SWI$UV to cause an RTI to be executed if a user function call is encountered. The user function call handler must reside in ROM or be included in the file MDOSUDRV.SY.
IRQ$UV	11A-11B	If the user wishes to have an IRQ handler in the system while MODOS is resident, IRQ$UV must contain the address of the user's IRQ handler. If IRQ$UV has not been initialized, MODOS sets up IRQ$UV to cause an RTI to be executed if an IRQ is generated while MODOS is resident. The IRQ handler must reside in ROM or be included in the file MDOSUDRV.SY.
IRQ$SV	11C-11D	Similar to SWI$SV except for IRQ handler.

NMI$SV	1A0-1A1	Similar to SWI$SV except for NMI handler. The user must beware that a disk timeout causes an NMI. Thus, if the user is allowing NMI's to occur after MODOS has been overlaid, none can occur while a disk transfer is in progress as it will be treated as a disk timeout. For this reason, the system does not support user NMI's while MODOS is resident.
CNACI$	1A2-1A3	CNACI$ contains the address of the console ACIA. It is initialized to $FCF4 to run with EXbug but can be changed by the OEM Manufacturer to suit his hardware requirements.
SBIT$	1A4	SBIT$ contains the stop bits required for the console, as well as information about parity, clock divide ratio, and control of interrupts. The fields are:

1. Clock divide ratio bits (2 right most bits)

1	0	FUNCTION
0	0	/1
0	1	/16
1	0	/64
1	1	Master Reset

2. Word length, parity, and number of stop bits

4	3	2	FUNCTION
0	0	0	7 bits+even parity+2 stop bits
0	0	1	7 bits+odd parity+2 stop bits
0	1	0	7 bits+even parity+1 stop bit
0	1	1	7 bits+odd parity+1 stop bit
1	0	0	8 bits+2 stop bits
1	0	1	8 bits+1 stop bit
1	1	0	8 bits+even parity+1 stop bit
1	1	1	8 bits+odd parity+1 stop bit

3. Transmitter control bits

6	5	FUNCTION
0	0	~RTS=low, transmitting interrupt disabled
0	1	~RTS=low, transmitter interrupt enabled
1	0	~RTS=high, transmitting

	interrupt disabled
1 1	~RTS=low, transmits a break level on the transmit data output; transmitting interrupt disabled

4. Receive interrupt enable bit--bit 7

The following interrupts will be enabled by having bit 7 set: receive data register full, overrun, or a low to high transition on the ~DCD signal line.

SBIT$ is initialized to $55 which sets the proper number of stop bits for a console speed of 30 characters per second or more.

XPEED$ 1A5

XPEED$ indicates console speed and is initially set for 30 cps. Valid values are:

CPS	XPEED$
10	0
30	1
120	FF
240	FF

CHARN$ 1A6

CHARN$ indicates the number of null pads to follow every character but a carriage return. It is initially set for 30 cps. Valid values are:

CPS	CHARN$
10	0
30	0
120	3
240	7

CRNL$ 1A7

CRNL$ indicates the number of null pads to follow a carriage return and is initialized for 30 cps. Valid values are:

CPS	CRNL$
10	0
30	4
120	17
240	2F

XBUGF$ 1A8

XBUGF$ indicates whether or not EXbug is in the system. It is initialized to one

		to indicate that EXbug is present. This allows the OEM Manufacturer to test his software using an EXORciser/EXORdisk system. (Reference Section III.) When running without an EXORciser/EXORdisk system, this field should be set to zero.
AUTOI$	1A9	AUTOI$ is a flag that indicates whether or not MODOS should automatically load and pass control to a user selected program upon completion of MODOS initialization. AUTOI$ is initialized to zero which indicates that no program will be loaded. However, with AUTOI$ = 0, MODOS prints a user specified sign-on message at completion of initialization (reference AUTOL$). If AUTOI$ is non-zero, MODOS will load and pass control to a user specified program (reference AUTOL$).
AUTOL$	1AA-1BD	This is a 20 byte buffer used in conjuction with AUTOI$. If AUTOI$ is set to zero, AUTOL$ contains a sign-on message assigned by the user. The ASCII message must be terminated by an EOT (04) and cannot exceed 20 characters, including the EOT. If AUTOI$ is non-zero, AUTOL$ contains the name of a file that should be loaded and executed automatically. The file name and any parameters to the program should appear in AUTOL$ exactly as they would be entered on the command line by the user. The length of the command cannot exceed 20 characters. Unused trailing characters should be spaces. MODOS will insert a carriage return in the buffer when the message is moved to MODOS's internal command buffer. AUTOL$ is initially set to an EOT followed by 19 spaces. Thus, no sign-on message is printed.
LPPIA$	1BE-1BF	LPPIA$ contains the address of the line printer PIA. It is initialized to $EC10 to run with EXbug but can be changed by the OEM manufacturer to suit his hardware requirements.

The following describes the jump vectors for the console I/O routines. Most likely, the OEM Manufacturer would have to substitute his own routines for all of the resident routines if his console is not interfaced through an ACIA. Also included is the jump vector to the routine to output a character to the the line printer. It can be modified if a different type of printer is being used. The routines must be in ROM or in the file MDOSUDRV.SY. Each of his routines must remain call compatible with the existing routines, both in terms of entry and exit conditions. A listing of the resident I/O routines is included in Appendix B so as to document the calling sequences and functions.

NAME	LOCATIONS	DESCRIPTION
INCHNP	1C1-1C2	Inputs one character with no parity.
OUTCH	1C4-1C5	Outputs one character with required speed fill.
PCRLF	1C7-1C8	Prints carriage return, line feed, and null on console.
PDATA	1CA-1CB	Prints carriage return, line feed, and data string terminated by EOT on console.
OCHAR	1CD-1CE	Outputs one character with no null pading to console.
INTCN	1D0-1D1	Initializes console.
BRKCK	1D3-1D4	Checks to see if break key was pressed.
LIST	241-242	Print contents of A-accumulator on line printer.

The following are jump vectors for device drivers used by the unified I/O package in MODOS. For each driver, there are five jump vectors-one-for each of the entry points to the driver. Calling sequence requirements are described in the MDOS manual under Device Drivers.

LOCATIONS	ENTRY POINT
1D6-1D7	Console keyboard/printer on.
1D9-1DA	Console keyboard/printer off.
1DC-1DD	Console keyboard/printer initialization.
1DF-1E0	Console keyboard/printer termination.
1E2-1E3	Console keyboard/printer input/output.
1E5-1E6	Console reader on.

1E8-1E9	Console reader off.
1EB-1EC	Console reader initialization.
1EE-1EF	Console reader termination.
1F1-1F2	Console reader input/output.
1F4-1F5	Console punch on.
1F7-1F8	Console punch off.
1FA-1FB	Console punch initializtion.
1FD-1FE	Console punch termination.
200-201	Console punch input/output.
203-204	Line printer on.
206-207	Line printer off.
209-20A	Line printer initialization.
20C-20D	Line printer termination.
20F-210	Line printer output.

If a user-device driver is substituted for a resident device driver, it may also necessary for the OEM Manufacturer to modify the Controller Descriptor Block (CDB) for that device. The CDB format is described in the MDOS manual. The address of the start of each CDB (i.e. the CDBIOC address) is contained in the table below:

DEVICE	CDB ADDRESS
Console keyboard/printer	712
Console reader	6F8
Console punch	705
Line printer	6DE

The following are jump vectors for the entry points in the disk and line printer drivers that reside in ROM. Normally, these jump vectors are altered by running the chain files OEMEX.CF and OEMCVT.CF. However, if the OEM Manufacturer modifies the relative starting addresses of the ROM routines, he must also change the jump vectors. (Reference Section III.)

NAME	LOCATIONS	DESCRIPTION
DRVTY$	211-212	Not an entry point. Drive type.
OSLOAD	214-215	Initialize disk's PIA and SSDA.
FDINIT	217-218	Initialize disk's PIA and SSDA.
CHKERR	21A-21B	Check and print error.
PRNTER	21D-21E	Print error from FDSTAT.
READSC	220-221	Read sector(s).
READPS	223-224	Read partial sector.
RDCRC	226-227	Read and check for CRC.
RWTEST	229-22A	Read/write test.
RESTOR	22C-22D	Move head to track 0.
SEEK	22F-230	Position head to track of 'STRSCT'
WRTEST	232-233	Write test.
WRDDAM	235-236	Write deleted data mar.
WRVERF	238-239	Write and verify CRC.
WRITSC	23B-23C	Write sector(s).
LPINIT	23E-23F	Initialize printer PIA.

Two examples are included here to demonstrate how to use PATCH to change the variables. The first example shows how to change the console ACIA address and change the console speed to 120 characters per second. The second shows how to specify a program to be loaded and executed immediately following MODOS initialization. In both cases, the MODOS diskette is in drive 1 of an EXORciser/EXORdisk MDOS system.

```
PATCH MDOS.SY:1
0100 30
>00                          BE SURE TO SET OFFSET TO 0
>1A2/E4,08                   CHANGE ACIA ADDRESS
>1A5/FF,3,17                 CHANGE SPEED TO 120
>Q
=
```

```
PATCH MDOS.SY:1              PATCH TO EXECUTE 'DIR'
0100 30
>00                          CHANGE OFFSET
>1A9/1                       SET AUTOI$=1
>1AA/"DIR"                   INSERT COMMAND IN AUTOL$
>Q
=
```

The patch below can be used to elimate the ":9" that appears on the console during initialization. The ":9" is caused by the INTCN routine processing during initialization which causes a TI type printer to be turned on. If the console is a CRT, the ":9" can be omitted by eliminating the call to INTCN.

```
PATCH MDOS.SY:1
0100 30
>00                          CHANGE OFFSET
>1CF(RTS)                    ALTER INTCN JMP VECTOR
>Q
=
```

III. DEVELOPING PROGRAMS FOR MODOS

Programs intended for use with MODOS may be developed on the standard EXORciser/EXORdisk system using MDOS. The MDOS editor and assembler are used to develop MODOS compatiable programs. The programs may be debugged using the EXbug version of MODOS.

The EXbug version of MODOS is a configuration of MODOS designed to permit the user to debug with the aid of EXbug. As shown as Appendix A-1, this configuration has a memory map identical to that of the EXORciser/EXORdisk development system. However, the MODOS software has been modified to use the standard MDOS PROM. MODOS as supplied to the user on diskette is configured to this EXbug version.

In order to debug a program using the EXbug version, the following steps must be followed:

1) Assemble the MODOS program using MDOS. If the MODOS program references the system equate file, EQU.SA, the MODOS version as supplied on the MODOS diskette must be used when assembling the program. The MDOS and MODOS versions of EQU.SA are not identical.

   If RLOAD is used, the commands shown below must be used:

   ```
   BASE=$nnnn
   STRB=$bb
   ```

   where nnnn is a hexadecimal number larger than $21FF and bb if a hexadecimal number between $40 and $FF.

2) Copy the program object file to the MODOS diskette.

3) Hit the 'RESTART' button on the EXORciser.

4) Place the MODOS diskette in drive 0 and reboot the system using the EXbug command 'E800;G'.

5) Load the MODOS program to be debugged using the MODOS LOAD command:

   ```
   LOAD program or LOAD program;V
   ```

6) Debug using EXbug.

IV. HARDWARE CONFIGURATIONS

Once the MODOS program has been debugged, the user is ready to execute the program in the user's hardware configuration. A typical configuration for MODOS using the micromodule 1A-D is shown in Appendix A-2. The source for the MODOS PROM for this configuration is provided on the MODOS diskette. (Reference Section VI.) This software must be modified for any other user configuration.

In order to use MODOS with the final hardware configuration, some changes must be made. The disk ROM must be generated and placed in the final system. The jump vectors to the ROM entry points must be modified. Also, the console ACIA and line printer PIA addresses must be changed and XBUGF$ cleared. In addition to this, the bootblock directly accesses the ROM and must be changed. A chain has been provided to perform these changes. With the MODOS diskette in drive 1 of an EXORciser/EXORdisk MDOS system, enter the following:

```
CHAIN OEMCVT:1;ACIA%XX,XX%,ROMS%YY%,LPS%ZZ%,PIA%WW,WW%
```

The parameters XXXX, YY, ZZ, and WWWW are the console ACIA address, ROM starting address (most significant byte), line printer driver starting address (most significant byte), and line printer PIA address, respectively. (Reference Appendix C for more detail.) Assuming the disk ROM is to be used as provided on the diskette, the parameters would be entered as follows:

```
CHAIN OEMCVT:1;ACIA%E4,08%,ROMS%FC%,LPS%FF%,PIA%EC,10%
```

The chain OEMCVT can only be used if the relative start of each entry point in the ROM remains unchanged. If any changes are made to the ROM that affect the relative start of the routines, the OEM Manufacturer must modify the chain file before running it. A commented listing of OEMCVT is provided in Appendix C to aid the user.

In order to reconfigure MODOS to run with EXbug, again insert the MODOS diskette in drive 1 of an EXORciser/EXORdisk MDOS system and enter:

```
CHAIN OEMEX:1
```

On completion of the chain, the diskette can be used in an EXbug environment only.

There are a few restrictions as to the type of hardware that can be used while running with MODOS.

1. A Calcomp drive is required but it can be an EXORdisk II

(2 drive, single sided, single density) or EXORdisk III (4 drive, double sided, single density).

2. If running MODOS in an EXORciser II system, programs cannot be loaded into the user map.

3. If using a micromodule 1A board, a maximum of 32K of RAM is allowed.

Two hardware configurations that allow MODOS to run with a micromodule 1A board are described below. To run MODOS using a 1A board in an EXORrciser, do the following:

1. Configure the disk ROM at $CC00 or $FC00, console ACIA at $8408, and line printer PIA at $EC10.

2. Use the OEMCVT chain file to configure jump vectors according to hardware requirements above.

3. Put ROM on 1A board.

4. Remove debug module and MPU board from EXORciser.

5. IMPORTANT: Jumper K1 pins 1 & 2 and 1 & 3 on 1A board.

6. Set baud rate on 1A board the same as software is configured.

7. Place 1A board in EXORciser.

8. Power on sequence will initialize the system.

WARNING: With some older versions of the Calcomp drives, the data on the diskette may be destroyed if power is turned off on the EXORciser while the diskette is in a drive.

It is possible to configure the MODOS system with the 1A board in the EXORciser and still have EXbug in the system. To do this, perform the following:

1. Configure disk ROM at $CC00, console ACIA at $8408, and line printer PIA at $EC10.

2. Use the OEMCVT chain file to configure the jump vectors according to the hardware requirements above.

3. Put ROM on 1A board.

4. Keep debug module in EXORciser but remove MPU board.

5. IMPORTANT: Jumper K1 pins 1 & 3 and 5 & 6.

6. Two terminals must be interfaced to system--the one for

EXbug with ACIA at $FCF4 and the other for MODOS at $8408 interfaced to the 1A board.

7. Set baud rate on 1A board the same as software is configured.

8. Place 1A board in EXORciser.

9. Power on sequence will cause EXbug to get control.

10. Set up interrupt vectors as follows:

```
FFF8/0020
FFFA/0023
FFFC/0026
```

or

If you want to be able to use MAID to set break points:

```
1A8/1 (XBUGF$=1)
```

9. Enter CC00;G to initialize MODOS.

V. COMMANDS

MODOS does not provide all of the commands available on MDOS. However, it is functionally compatible with MDOS 3.00 and the MDOS 3.00 User's Guide should be consulted for details. What follows is a list of commands normally included with MODOS:

BACKUP
COPY
DEL
DIR
FORMAT
FREE
LOAD
MERGE
NAME
REPAIR

The family attribute feature available on MDOS for the commands BACKUP, DEL, DIR, and NAME is not available on the MODOS version of the commands.

VI. NEW ERROR MESSAGES

During MODOS initialization, the message "ED" may appear on the console. This error message indicates that the file containing the user's I/O routines and/or interrupt vectors (MDOSUDRV.SY) is not in a proper format to be loaded. This can occur for several reasons:

1. MDOSUDRV.SY is not a memory image type file.

2. MDOSUDRV.SY loads below the resident portion of MODOS. It should reside at the high end of memory so as to allow commands enough memory in which to run.

MODOS does not check that memory exists where MDOSUDRV.SY is being loaded. It can be loaded anywhere above the command interpreter. This allows MDOSUDRV.SY to be loaded as low as $2400 or it may even be loaded into discontiguous memory. Once MDOSUDRV.SY has been loaded, it cannot be overlaid by any commands. MDOSUDRV.SY is not a required file and if it does not exist, all of memory is available for commands.

If, during MODOS initialization, the message "WHAT?" appears prior to any user input, this indicates that MODOS was configured to automatically load and execute a user specified file during initialization and the file does not exist or is not in memory image format or would overlay resident MODOS. (Reference AUTOI$ and AUTOL$ descriptions.)

VII. DISK PROM

The disk and line printer controllers must reside in ROM. The source for the ROM is included on the MODOS diskette in the file ROM.SA. The source has the ROM origined at $FC00, the console ACIA address set at $E408, and the line printer PIA set at $EC10. ROM.LO file is also included on the disk and includes the memory image file to be used when generating a PROM with the PROM Programmer. The OEM Manufacturer may have to modify the ROM in order to remain compatible with his hardware configuration. The console ACIA address (CNACI$), if different from the initialized value, should be changed. The stop bit control byte (SBIT$) may have to be changed. The line printer PIA address (LPPIA$) may have to be changed. In addition, two console routines, an output character routine and console initialization, are also included in the PROM. If these are changed, the calling sequences must remain compatible with the existing routines.

To assemble the ROM, insert the MODOS diskette in drive 1 of an EXORciser/EXORdisk MDOS system and enter the following:

```
RASM ROM:1;L
```

With the ROM residing at $FC00, the interrupt vectors are set up. The restart vector is initialized so that MODOS is automatically loaded and control passed to it whenever a restart interrupt occurs. However, the ROM does not have to reside at $FC00. If it does not, it is the OEM Manufacturer's responsibility to initialize the interrupt vectors prior to loading MODOS. The interrupt vectors must be initialized as follows:

INTERRUPT VECTOR	ADDRESS	VALUE
IRQ	FFF8-FFF9	0020
SWI	FFFA-FFFB	0023
NMI	FFFC-FFFD	0026
RESTART	FFFE-FFFF	**

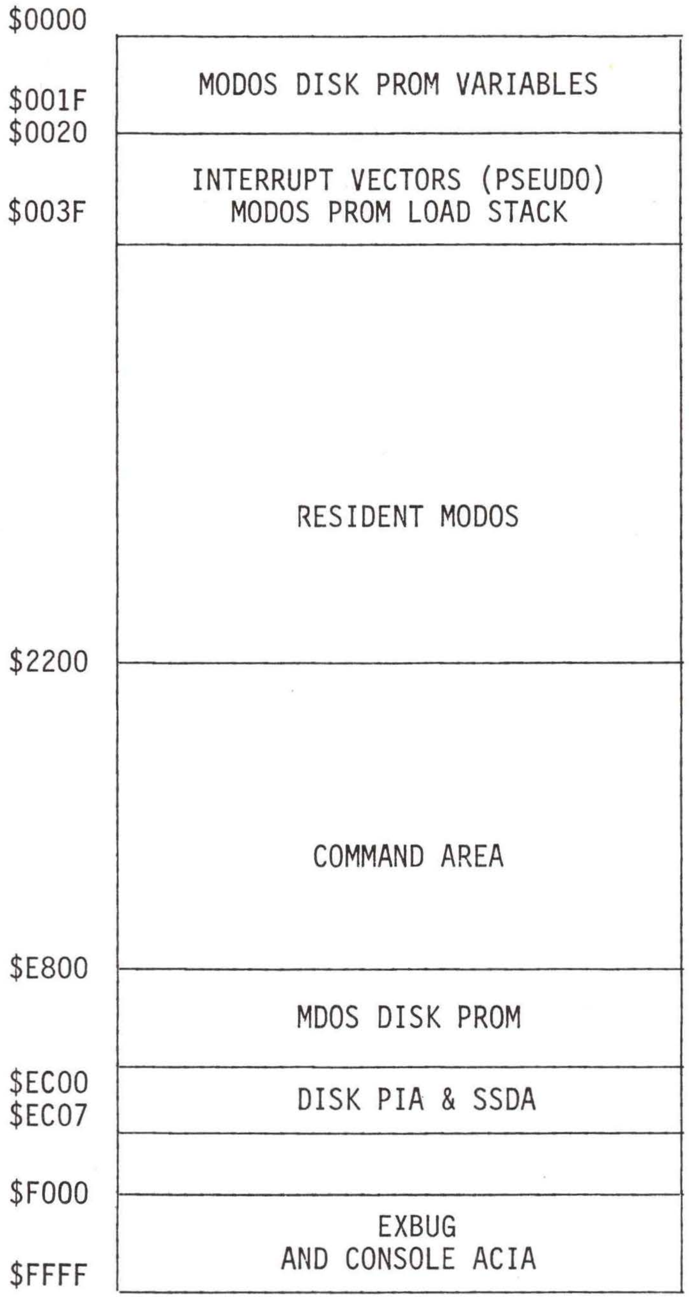
** The RESTART vector should contain the address of the start of ROM if MODOS is to be automatically loaded on RESTART. Otherwise, control must be passed to the beginning of the ROM from another program.

APPENDIX A

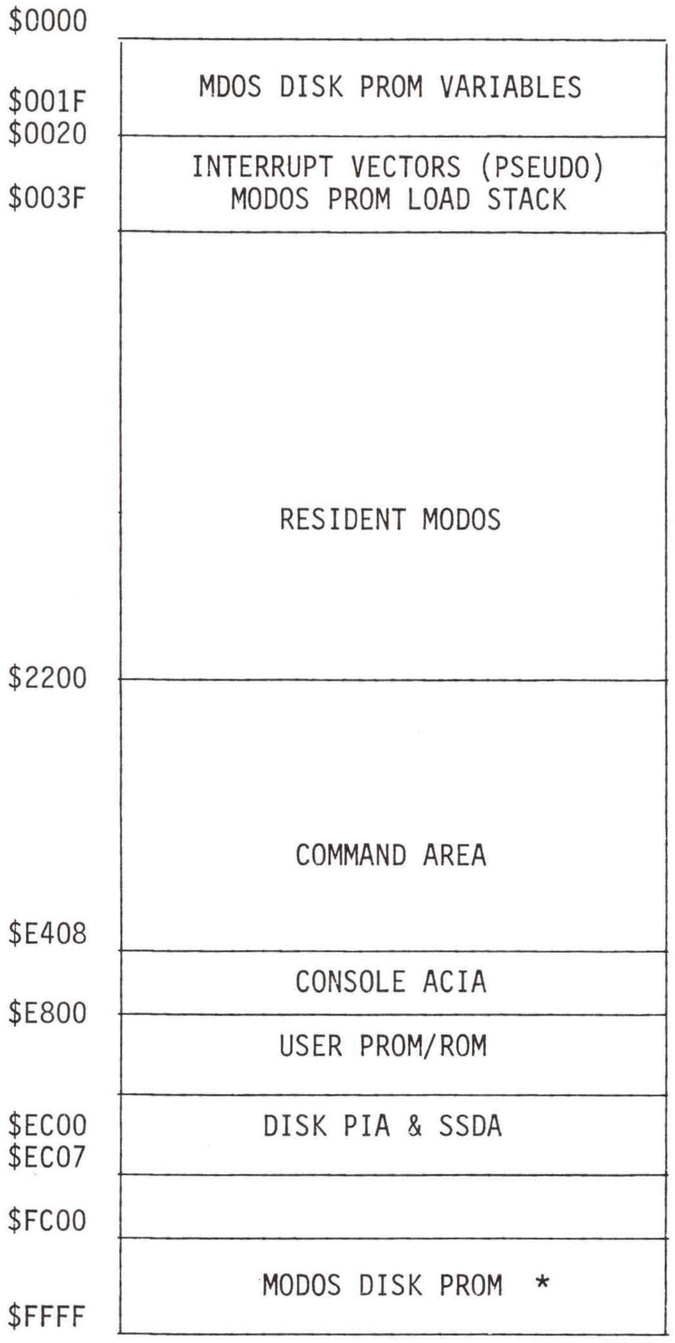
MEMORY MAPS

APPENDIX A

A MEMORY MAP SHOWING THE MINIMUM MEMORY REQUIREMENTS FOR MODOS WITH EXBUG



A MEMORY MAP SHOWING THE MINIMUM
MEMORY REQUIREMENTS FOR MODOS
WITHOUT EXBUG



*MODOS DISK PROM need not reside at $FC00. (Reference Section VII).

APPENDIX B

CONSOLE I/O ROUTINES

```
00536                        NAM    EXSUB
00537                        IDNT   EXBUG SUBSTITUTE I/O -- OCT. 27, 19
00538                     *
00539                     * CONSOLE I/O ROUTINES ARE RESIDENT TO
00540                     *     ELIMINATE MDOS EXBUG DEPENDENCY
00541                     *
00542                     *PART OF LINE PRINTER DRIVER FROM THE DISK
00543                     *     CONTROLLER ROM IS ALSO INCLUDED HERE.
00544                     *     OTHERWISE, THERE WOULD BE INSUFFICIENT ROOM
00545                     *     IN THE ROM TO INCORPORATE THE DOUBLE-SIDED,
00546                     *     FOUR DRIVE CHANGES.
00547                     *
00548                     *
00549                        XDEF   INCHN$,OUTCH$,PCRLF$,PDATA$,OCHAR$,
00550                     *
00551                     * TEMPORARY STORAGE FOR X REGISTER (USED IN PLACE
00552                     *
00553P 0000    0000    A EXSTMP FDB    0         .
00554                     *
00555                     * LIST$ -- PRINTS CHARACTER AND CHECKS FOR ERROR
00556                     *
00557                     * CALLING SEQUENCE:
00558                     *       A = CHARACTER TO BE PRINTED ON LINE PRINT
00559                     *       JSR LIST
00560                     *
00561                     *       ALL REGISTERS ARE PRESERVED UPON EXIT.
00562                     *       C = 0 IF NO ERROR.
00563                     *       C = 1 IF ERROR.
00564                     *
00565            0002    P LIST$  EQU    *         .
00566P 0002 FF 0000    P        STX    EXSTMP    . SAVE INDEX REGISTER
00567P 0005 FE 01BE    A        LDX    LPPIA$    . PICK UP PIA ADDRESS
00568P 0008 A7 00      A        STAA   PIADTA,X  . SEND DATA
00569P 000A 86 3E      A        LDAA   #$3E
00570P 000C A7 01      A        STAA   PIACTA,X
00571P 000E A6 00      A        LDAA   PIADTA,X
00572P 0010 86 36      A        LDAA   #$36      . STROBE PRINTER
00573P 0012 A7 01      A        STAA   PIACTA,X  .
00574P 0014 86 3E      A        LDAA   #$3E      .
00575P 0016 A7 01      A        STAA   PIACTA,X  .
00576P 0018 A6 02      A LIST3  LDAA   PIADTB,X  . CHECK STATUS
00577                     *BIT 0 => SELECT, BIT 1 => PAPER OUT
00578P 001A 84 03      A        ANDA   #3
00579P 001C 4A                  DECA             . A SHOULD NOT HAVE BEEN 0
00580P 001D 26 05 0024          BNE    ERROR     . NE=>NO PAPER OR NOT SELE
00581P 001F 6D 01      A        TST    PIACTA,X  . AKNOWLEDGE ?
00582P 0021 2A F5 0018          BPL    LIST3     . NO
00583P 0023                     SKIP1  .
00584P 0024 0D            ERROR SEC              .
00585P 0025 A6 00      A        LDAA   PIADTA,X  . RESTORE A REGISTER
00586P 0027 FE 0000    P        LDX    EXSTMP    . RESTORE X
00587P 002A 39                  RTS              .
```

```
                           *
                           *INTCN$--DOES SOME CONSOLE INITIALIZATION-
                           *     TURNS CONSOLE PRINTER ON
                           *     TURNS PUNCH AND READER OFF
                           *
                           *     THIS INITIALIZATION IS IN ADDITION TO
                           *     WHAT IS DONE BY THE DISK PROM
                           *
                           *CALLING SEQUENCE:
                           *     JSR INTCN
                           *     ALL REGISTERS DESTROYED ON RETURN
                           *
                002B  P INTCN$ EQU    *
                           *ACTIVATE TI RDC CARD
P 002B 8D 0F 003C                 BSR    DLE$      RDC ATTENTION
P 002D 86 3A       A              LDAA   #':
P 002F 8D 0D 003E                 BSR    XOCHA
                           *TURN ON TI PRINTER
P 0031 8D 09 003C                 BSR    DLE$      RDC ATTENTION
P 0033 86 39       A              LDAA   #'9       TI PRINTER ON
P 0035 8D 07 003E                 BSR    XOCHA
P 0037 CE 0041     P              LDX    #MST1     PUNCH AND READER OFF
P 003A 20 2C 0068                 BRA    PDATA1
P 003C 86 10       A DLE$         LDAA   #$10
P 003E 7E 01CC     A XOCHA        JMP    OCHAR


P 0041      14     A MST1         FCB    $14       DC4-PUNCH OFF
P 0042      13     A MST          FCB    $13       DC3-READER OFF
P 0043      00     A              FCB    0,0,0,0,4 NULLS FOR TTY TIMING
```

```
                         *
                         *INCHNP--INPUTS ONE CHAR AND REMOVES PARITY
                         *
                         *CALLING SEQUENCE:
                         *      AECHO = 0 IF CHAR. SHOULD BE ECHOED
                         *      AECHO .NE. 0 IF NO ECHO
                         *      JSR INCHNP
                         *      A = CHAR. INPUT
                         *      B,X ARE PRESERVED
                         *
             0048      P INCHN$ EQU     *
 0048 FF 0000          P        STX     EXSTMP   . SAVE X-REG.
 004B FE 01A2          A        LDX     CNACI$
 004E A6 00            A INCH1  LDAA    ACIACS,X
 0050 47                        ASRA
 0051 24 FB 004E                BCC     INCH1    RECEIVE NOT READY
 0053 A6 01            A        LDAA    ACIADT,X INPUT CHAR.
 0055 FE 0000          P        LDX     EXSTMP   RESTORE X
 0058 7D 018F          A        TST     AECHO
 005B 26 02 005F                BNE     INCH2    DON'T ECHO
 005D 8D 21 0080                BSR     XOUTCH   ECHO CHAR.
             005F      P INCH2  EQU     *
 005F 7F 018F          A        CLR     AECHO    SET ECHO FLAG
 0062 84 7F            A        ANDA    #$7F     REMOVE PARITY
 0064 39                        RTS
```

```
00645                     *
00646                     *PDATA--PRINTS CR/LF FOLLOWED BY DATA POINTED
00647                     *    TO BY X-REG
00648                     *
00649                     *CALLING SEQUENCE:
00650                     *     X = ADDR. OF BUFFER TO PRINT (EOT
00651                     *     IN BUFFER TERMINATES)
00652                     *    JSR PDATA
00653                     *     X AND A ARE ALTERED ON RETURN
00654                     *     B IS PRESERVED
00655                     *
00656           0065    P PDATA$ EQU    *
00657P 0065 BD 01C6   A        JSR    PCRLF    PRINT CR/LF
00658P 0068 A6 00     A PDATA1 LDAA   X        UNDOCUMENTED ENTRY-NO CR/L
00659P 006A 81 04     A        CMPA   #EOT
00660P 006C 27 06 0074         BEQ    PDATA2   STOP ON EOT
00661P 006E BD 01C3   A        JSR    OUTCH    OUTPUT CHAR
00662P 0071 08                 INX
00663P 0072 20 F4 0068         BRA    PDATA1
00664P 0074 39          PDATA2 RTS
```

```
00666                    *
00667                    *PCRLF--OUTPUTS LF, CR, NULL
00668                    *
00669                    *CALLING SEQUENCE:
00670                    *      JSR PCRLF
00671                    *      A ALTERED ON RETURN
00672                    *      B,X ARE PRESERVED
00673                    *
00674          0075   P PCRLF$ EQU    *
00675P 0075 86 0A     A        LDAA   #LF       PRINT LINE FEED
00676P 0077 BD 01C3   A        JSR    OUTCH
00677P 007A 86 0D     A        LDAA   #CR       PRINT CARRIAGE RET.
00678P 007C BD 01C3   A        JSR    OUTCH
00679P 007F 4F                 CLRA             PRINT NULL
00680P 0080 7E 01C3   A XOUTCH JMP    OUTCH
```

```
                 *
                 *OCHAR--OUTPUTS CHAR. IN A-REG.
                 *
                 *CALLING SEQUENCE:
                 *      A = CHAR. TO OUTPUT
                 *      JSR OCHAR
                 *      ALL REGISTERS ARE PRESERVED
                 *
       0083    P OCHAR$ EQU    *
P 0083 37               PSHB            SAVE B AND X
P 0084 FF 0000 P        STX    EXSTMP
P 0087 FE 01A2 A        LDX    CNACI$   GET ACIA ADDR.
P 008A E6 00   A OCHAR1 LDAB   ACIACS,X
P 008C C5 02   A        BITB   #2
P 008E 27 FA 008A       BEQ    OCHAR1   NOT READY
P 0090 A7 01   A        STAA   ACIADT,X OUTPUT CHAR.
P 0092 FE 0000 P        LDX    EXSTMP   RESTORE REGS.
P 0095 33               PULB
P 0096 39               RTS
```

```
                          *
                          *OUTCH--OUTPUTS CHAR. IN A-REG. AND PAD CHARS.
                          *
                          *IF PUNCHING AND CR, PAD 4 NULLS
                          *IF PUNCHING AND NOT CR, NO NULLS
                          *IF NOT PUNCHING AND CR, PAD CRNL$ NULLS
                          *IF NOT PUNCHING AND NOT CR, PAD CHARN$ NULLS
                          *
                          *CALLING SEQUENCE:
                          *      A = CHAR. TO OUTPUT
                          *      CAS$ET = 0 IF NOT PUNCHING
                          *      CAS$ET .NE. 0 IF PUNCHING
                          *      CRNL$ = NO. OF NULLS AFTER CARRIAGE RETURN
                          *      CHARN$ = NO. OF NULLS AFTER OTHER CHARS.
                          *      JSR OUTCH
                          *      ALL REGISTERS PRESERVED
                          *
                 0097   P OUTCH$ EQU    *
P 0097 BD 01CC   A               JSR    OCHAR       OUTPUT CHAR.
P 009A 37                        PSHB
P 009B 81 0D     A               CMPA   #CR         CARRIAGE RTN?
P 009D 26 15 00B4                BNE    OUTCH5      NO
P 009F C6 04     A               LDAB   #4          4 NULLS IF CR AND PUNCHIIN
P 00A1 7D 0190   A               TST    CAS$ET
P 00A4 26 03 00A9                BNE    OUTCH7      PUNCHING
P 00A6 F6 01A7   A               LDAB   CRNL$       CR AND NOT PUNCHING
                 00A9   P OUTCH7 EQU    *
P 00A9 5A                        DECB
P 00AA 2B 10 00BC                BMI    OUTCH9
P 00AC 36                        PSHA
P 00AD 4F                        CLRA               NULL
P 00AE BD 01CC   A               JSR    OCHAR
P 00B1 32                        PULA
P 00B2 20 F5 00A9                BRA    OUTCH7
P 00B4 F6 01A6   A OUTCH5        LDAB   CHARN$
P 00B7 7D 0190   A               TST    CAS$ET      NOT CR AND NOT PUNCHING
P 00BA 27 ED 00A9                BEQ    OUTCH7
P 00BC 33                 OUTCH9 PULB               PUNCHING AND NOT CR
P 00BD 39                        RTS
```

```
                   *
                   *BRKCK--CHECKS FOR BREAK KEY AND CONTROL-W
                   *
                   *CALLING SEQUENCE:
                   *      JSR BRKCKK
                   *      CARRY = 0 IF NO BREAK
                   *      CARRY = 1 IF BREAK DETECTED
                   *      ALL REGISTERS ARE PRESERVED
                   *
          00BE   P BRKCK$ EQU    *
P 00BE 36                 PSHA
P 00BF FF 00D0   P        STX    BREAK2+1  . SAVE X
P 00C2 FE 01A2   A        LDX    CNACI$    GET ACIA ADDR.
P 00C5 0C          BREAK1 CLC
P 00C6 A6 00     A        LDAA   ACIACS,X  READ STATUS REG.
P 00C8 85 10     A        BITA   #%10000   LOOK FOR FRAMING ERROR BIT
P 00CA 27 08 00D4         BEQ    BREAK3    NO FRAMING ERROR
P 00CC 8D 14 00E2         BSR    BREAK6    RESET ACIA FROM FRAMING ER
P 00CE 0D                 SEC
                   *
                   * THE NEXT INSTRUCTION IS MODIFIED AT ENTRY POINT
                   *
P 00CF CE 0000   A BREAK2 LDX    #0        . RESTORE REGS.
P 00D2 32                 PULA
P 00D3 39                 RTS
                   *
                   *NO FRAMING ERROR--BUT LOOK FOR CONTROL-W
                   *      (WAIT) TO PAUSE FOR A WHILE--WAIT
                   *      UNTIL ANOTHER KEY IS PRESSED
                   *
P 00D4 85 01     A BREAK3 BITA   #1        LOOK FOR RECEIVED DATA REA
P 00D6 27 F7 00CF         BEQ    BREAK2    NO CHAR. TO READ
P 00D8 A6 01     A BREAK4 LDAA   ACIADT,X  READ CHAR.
P 00DA 84 7F     A        ANDA   #$7F      STRIP PARITY
P 00DC 81 17     A        CMPA   #ETB      CTL-W?
P 00DE 27 F8 00D8         BEQ    BREAK4    YES, SO WAIT FOR OTHER KEY
P 00E0 20 E3 00C5         BRA    BREAK1    ELSE CHECK FOR FRAMING ERO
                   *SESET THE AACIA
P 00E2 8D 0B 00EF  BREAK6 BSR    BREAK9    WAIT FOR TRANSMIT READY
P 00E4 8D 05 00EB         BSR    BREAK8    SEND A NULL AND WAIT FOR R
P 00E6 A6 01     A        LDAA   ACIADT,X  READ A CHAR.
P 00E8 A6 01     A        LDAA   ACIADT,X  ANOTHER
P 00EA 39                 RTS
P 00EB 4F          BREAK8 CLRA
P 00EC BD 01C3   A        JSR    OUTCH     SEND A NULL
P 00EF A6 00     A BREAK9 LDAA   ACIACS,X  SET STATUS BYTE
P 00F1 85 02     A        BITA   #2        CHECK TRANSIT READY
P 00F3 27 FA 00EF         BEQ    BREAK9    NOT READY
P 00F5 39                 RTS
P 00F6                    TITLE  (SYMBOL   TABLE)
```

APPENDIX C

OEMCVT CHAIN FILE

```
/*GOING TO PATCH MODOS FOR NON-EXBUG USER CONFIGURATION
/*ENTER PARAMETER 'ACIA\%XX,XX\%' WHERE XX,XX IS CONSOLE ACIA
 ADDRESS
/*     EXPRESSED AS TWO BYTES
/*ENTER PARAMETER 'ROMS\%YY\%' WHERE YY IS MOST SIGN. BYTE
/*     OF DISK ROM STARTING ADDRESS
/*ENTER PARAMETER 'LPS\%ZZ\%' WHERE ZZ MUST BE THE VALUE 'ROM
S+3'
/*ENTER PARAMETER 'PIA\%WW,WW\%' WHERE WW,WW IS LINE PRINTER
PIA ADDRESS
/*     EXPRESSED AS TWO BYTES
/IFC ACIA
/*CONSOLE ACIA ADDRESS (ACIA) MUST BE SPECIFIED
/ABORT
/ELSE
/IFC ROMS
/*ROM STARTING ADDRESS (ROMS) MUST BE SPECIFIED
/ABORT
/ELSE
/IFC LPS
/*LINE PRINTER DRIVER STARTING ADDRESS (LPS) MUST BE SPECIFIE
D
/ABORT
/ELSE
/IFC PIA
/*LINE PRINTER PIA ADDRESS (PIA) MUST BE SPECIFIED
/ABORT
/XIF
/XIF
/XIF
/XIF
@. INSERT MODOS 3.0 DISKETTE IN DRIVE 1
PATCH MDOS.SY:1
OO
1A2/%ACIA%      THESE LOCATIONS ARE DESCRIBED IN SECTION II
1A8/O
1BE/%PIA%
211/%LPS%,F7
214/%ROMS%,00
217/%ROMS%,39
21A/%ROMS%,6A
21D/%ROMS%,71
220/%ROMS%,80
223/%ROMS%,84
226/%ROMS%,86
229/%ROMS%,89
22C/%ROMS%,8C
22F/%ROMS%,8F
232/%ROMS%,92
235/%ROMS%,95
238/%ROMS%,98
23B/%ROMS%,9B
23E/%LPS%,E8
```

```
Q
DUMP
U 1
R 17
4/%ROMS%,8C/        RESTOR
7/%LPS%,E8/       LPINIT
64/6A/      LSB OF CHKERR
67/%ROMS%/       MSB OF CHKERR
6A/%ROMS%,84/       READPS
72/%ROMS%,6A/       CHKERR
B/CO/      DON'T CHANGE
W
Q
PATCH REPAIR.CM:1
22000
1903/%ROMS%,8C        SEE DUMP ABOVE
1906/%LPS%,E8
1963/6A
1966/%ROMS%
1969/%ROMS%,84
1971/%ROMS%,6A
190A/CO
Q
@SET,M 8
DIR DOSGEN.CM:1
@TST,T NE,80
@JMP END
@SET,M 0
PATCH DOSGEN.CM:1
22000
5B4/%ROMS%,8C
5B7/%LPS%,E8
614/6A
617/%ROMS%
61A/%ROMS%,84
622/%ROMS%,6A
5BB/CO
Q
@LBL END
```