SOFTWARE SECTION

The third section of this manual contains information about software that is helpful in configuring COMPUPRO systems to meet the varied needs of end users. This includes, making changes in the BIOS to work with different peripherals, choosing and installing the correct CP/M system, as well as updates and patches for the software.

Technical manuals for MP/M 8-16, CP/M 86 and CP/M 8-16 are included as well as handouts on software updates and patches.

The purpose of this section is to give the technician an easy reference to documentation on the software that is necessary to run COMPUPRO equipment in a variety of situations.

CONTENTS

This is a list of the I/O ports used by all COMPUPRO products.

PORT	PRODUCT
00 through 03	INTERFACER 1 and 2
10 through 17	INTERFACER 3 and 4
50 through 5f	SYSTEM SUPPORT
90	DISK 3
C0 through C3	DISK 1
C4 and C5	RESERVED
C6 and C7	MDRIVE/H
C8	DISK 2
F0	SELECTOR CHANNEL
F1	MPX
F2 through F5	RESERVED
FD	MEMORY MANAGER and SWAP PORT ON DUAL PROCESSOR
FFF0 through FFFF	80130 on CPU 8086/87

COMPUPRO
SOFTWARE ALTERATION GUIDE FOR CP/M 2.2 VERSION N
5/1/84

OPENING

This document is designed to aid the user of CP/M 2.2 in its reconfiguration and alteration. It includes the following main sections:

I REQUIREMENTS AND DEFINITIONS
II A GUIDE TO DESIRED CHANGES WITH STEP-BY-STEP EXAMPLES
III A GUIDE TO THE RECREATION OF THE SYSTEM
IV CLOSING AND REFERENCE SECTION

I REQUIREMENTS AND DEFINITIONS

This section of the manual will describe the features of this version of the operating system and the changes since the last revision, the required files for the modification and recreation of the operating system, a short description of the files involved in creating the CP/M 2.2 BIOS, and some simple file designator definitions

FEATURES

The 2.2N version of this operating system incorporates the following special features and capabilities:

A) DISK 2 and DISK 3 hard disk support
B) Support for MDRIVE using RAM with CPU 8085/88
C) Support for MDRIVE/H
D) Support for minifloppies using DISK 1 - 5
E) Support for interrupts

The new CP/M BIOS supports all of the same physical devices that the previous BIOS did.

CHANGES SINCE LAST REVISION

Since the last revision, 2.2M, The following changes have been made:

1) 2.2N now supports X/ON X/OFF and DTR protocols.
2) The I/O Byte is supported more extensively.
3) The DISK 3 controller is supported.

REQUIREMENTS:

Before you begin to use the instructions given here, you should become familiar with certain other documentation that is included with CP/M 2.2. You should be familiar with the material covered in Digital Research's CP/M 2.2 Manual and the CompuPro CP/M 2.2 Technical Manual & Installation Proceedures. If you do not have a word processor, you should carefully study and practice with the line editor, ED.COM. This document also assumes that you understand the function of the BIOS and the LOADER in the CP/M operating system.

The most important item that is required for the alteration of the 2.2 BIOS is Digital Research's RMAC assembler and linker. If you do not have it, you cannot reassemble the BIOS. The next item of importance is a text editor of some kind. Almost anything is better and easier to use than ED.COM. The following list of files and their short descriptions are required to reassemble the system and are included on your master diskette.

NOTE: Do not modify your master diskette! Make a copy of it and modify the copy! You will receive no sympathy if you destroy your master.

FILES REQUIRED FOR BIOS MODIFICATION

ACTIVE.LIB: Contains control variables which set the conditions, according to the particular hardware configuration for BIOS customization.

COMPUPRO.LIB: Contains data constants common to all the CP/M Operating System's components. For example, Input/output Port Assignments, UART register bit definitions for the I/O boards, etc.

BOOTSCPM.LIB: Contains a set of routines which perform cold and warm boot functions.

CPMDISK.LIB: Contains CP/M disk constants, definition of base page as related to file operations, definition of vectors to call CBIOS Routines as well as BDOS call function numbers.

ASCII.LIB: Contains definitions for all ASCII control characters.

HMXFBOOT.ASM: Contains a portion of the cold boot initialization as well as USART initialization.

HMX1BIOS.ASM: This is the main body of the BIOS. Consults all .LIB files for the conditions set within them.

HMX2BIOS.ASM: This is the BIOS to be used when extra I/O driver and /or interupt routines are used. This BIOS has minimal I/O drivers ie. system console and a printer. Other drivers and interupt routine are in HMX2IO (see HMX2IO.ASM).

HMX2IO.ASM: Contains extended I/O driver routines with software handshaking protocols for all I/O boards and initialization for UARTs 0-11. All changes made in HMXFBOOT.ASM should also be made in HMX2IO.ASM when using HMX2BIOS.ASM.

HMXFPROM.ASM: Contains floppy disk (8 & 5 1/4) initializing program loader, similar to the routine found in Boot Rom on the Disk 1 Controller Board. It loads the loader for the Operating System.

DSBLINTR.ASM: A command file which will disable the System Support I interupts, except the system console and printer.

CPMHMX2.COM: This is the assembled HMX2BIOS.ASM file ready for SYSGENing onto the boot diskette. This is used with HMX2IO.COM.

CPMPLAIN.COM: Contains floppy only Operating System with 8 ms. step rate. This is the system found on the system tracks.

CPM220.COM: Contains Hard Disk Operating System for the Pragmatic 20 Megabyte Hard Disk and Disk 2 Controller.

CPM210.COM: Contains Hard Disk Operating System for the Pragmatic 10 Megabyte Hard Disk and Disk 2 Controller.

CPMQ540.COM: Contains Hard Disk Operating System for the Quantum 40 Megabyte Hard Disk and Disk 3 Controller.

STARTUP.SUB: This file loads HMX2IO.COM when used with HMX2BIOS.COM. It also loads MFORM.COM.

Here are some short descriptions of file name extensions that you will find in the directory.

.LIB = library file.
.COM = 8 bit executable command file.
.CMD = 16 bit executable command file.
.ASM = assembly language file.
.HEX = hexadecimal file.
.PRN = print file produced by the assembler.
.SYM = symbol table file produced by the assembler.
.REL = relocatable binary file.
.ASC = ASCII text file.
.LST = ASCII list file.
.DOC = documentation file.
.TXT = text file.
.RSP = resident system process file.
.SUB = submit file used with SUBMIT command.
.SYS = operating system file.

II GUIDE TO DESIRED CHANGES WITH STEP-BY-STEP EXAMPLES

This section of the manual is a guide to which files need to be modified to accomplish specific changes in the operating system. It is divided into four parts. Part 1) describes changes that can be made without reassembling the BIOS but just using a different system file already provided. Part 2) describes simple changes to the ACTIVE.LIB file for modifications like floppy drive characteristics, turning M-DRIVES on and off, turning on and off both DISK 2 and 3, handshaking for INTERFACER 3 and 4, etc., and Part 3) describes major changes involving alteration of the HMX1BIOS.ASM file like baud rates, handshaking, I/O protocols, etc. Part 4) describes a simple method of modifying the baud and step rates on existing CPMxxxx.COM files using DDT.

PART 1) There are several CPMxxxxx.COM files already configured to run many standard hardware configurations. If one of these configurations fulfills your requirements, you do not need to reassemble your BIOS. To use these configurations, you simply SYSGEN your diskette with the proper file. An example is given at the end of this section.

CPMPLAIN.COM
- Four 8" floppy drives @ 8 ms. step rate
- No hard disk
- Console @ 9600 baud - no handshaking
- List device @ 9600 baud - no handshaking
- Supports M-DRIVE/H
- No "soft" M-DRIVE

CPM210.COM
- CPU 8085/88
- DISK2 and 10 Mb hard disk
- Four 8" floppy drives @ 3 ms. step rate
- Console @ 9600 baud - no handshaking
- List device @ 9600 baud - no handshaking
- Supports M-DRIVE/H
- No "soft" M-DRIVE

CPM220.COM
- CPU 8085/88
- DISK2 and 20 Mb hard disk
- Four 8" floppy drives @ 3 ms. step rate
- Console @ 9600 baud - no handshaking
- List device @ 9600 baud - no handshaking
- Supports M-DRIVE/H
- No "soft" M-DRIVE

CPMQ540.COM
- CPU 8085/88
- DISK3 and 40 Mb Quantum Q540 hard disk
- Four 8" floppy drives @ 3 ms. step rate
- Console @ 9600 baud - no handshaking
- List device @ 9600 baud - no handshaking
- Supports M-DRIVE/H
- No "soft" M-DRIVE

CPMSYS.COM — CPU 8085/88
Four 8" floppy drives @ 3 ms. step rate
Console @ 9600 baud - 7 data bits 1 stop
List device @ 9600 baud - DTR handshaking
Supports M-DRIVE/H
No "soft" M-DRIVE

CPMHMX2.COM — Loads HMX2IO.COM which has:
Capability for interrupts and considerable other features. This file should be reviewed and altered by only experienced users. We do not reccomend that this file be used without customization.

EXAMPLE FOR USING THE ABOVE FILES:

To use one of the above stock system files, you must SYSGEN them onto the system tracks of the diskette. The proceedure for doing this is as follows:

Type the following:

SYSGEN CPMxxxxx.COM<CR>

It will respond asking for the destination drive name or hit return to terminate. You should respond with the drive name that corresponds to the floppy that you wish to have the new system placed on. You should now be able to use the new diskette.

PART 2) Many changes to the system may be made by altering the ACTIVE.LIB file and then simply reassembing the BIOS using RMAC. The easiest way to do this is first print out the ACTIVE.LIB on your printer for reference. After reviewing the printout and deciding which changes you would like to make, then load the ACTIVE.LIB file using your text editor. Most changes will be made by changing equates from true to false or false to true, or changing the value of a constant. Examples are shown below.

EXAMPLE: To turn the SYSTEM SUPPORT 1 BOARD off (if it is not present) and hence the 8088 off (since it cannot be used without RAM up at 0FFFF0H), you would turn the underlined "true" to "false".

```
;---------------------------------------
Z80     EQU     FALSE           ;Z80 CPU present
I8085   EQU     TRUE            ;8085 processor present
CPUSPD  EQU     6               ;CPU speed factor delay for "x"
;
I8088   EQU     TRUE            ;8088 present (dual processor board)
I8087   EQU     FALSE           ;8087 coprocessor present
;---------------------------------------
;
```

And the following:

```
;---------------------------------------
;
; System Support I setup characteristics:
SYSUP1  EQU     TRUE            ;System Support board is present
;
; System Support Uart Active Status Masks:
SS1TMSK EQU  SS1TBE ;or SS1DSR  ;Transmit Ready Mask
SS1FMSK EQU  SS1TBE ;or SS1DSR  ;Transmit Ready Bit Flip Mask
;
; Timer/Counter Commands and Values:
TIMEBASE EQU    10000           ;Divide 2MHz clock for other clock/
;                               (as a more reasonable time base --
TIME1   EQU     20              ;Real Time Clock interval (not acti
TIME2   EQU     200             ; 1 sec interrupt to read clock
;---------------------------------------
```

EXAMPLE: To turn the DISK 3 BOARD off (if it is not present) you would turn the underlined "true" to "false".

```
;---------------------------------------
;
; CompuPro DISK 3 Equates:
;
D3Q540  EQU     TRUE
D3M20   EQU     FALSE
D3ST506 EQU     FALSE
DISK3   EQU     D3Q540 or D3ST506 or D3M20      ;DISK 3 code active contr
NDISK3  EQU     1               ;Number of drives on all DISK3s
DISK3X  EQU     NDISK3 > 1      ;Second drive on line
DISK3Y  EQU     NDISK3 > 2      ;Third drive on line
DISK3Z  EQU     NDISK3 > 3      ;Fourth drive on line
;---------------------------------------
```

EXAMPLE: To turn the XMDRIVE "ON" (M-DRIVE using system memory beyond 64K) you would turn the underlined "false" to "true".

```
;---------------------------------------
;
;       MEMORY DRIVE and BOOT activity indicators:
;
; (I8088 must also be true if using dual processor memory drive).
XMDRIVE EQU  (FALSE and I8088)  ;Memory drive using dual processor
XMBOOT  EQU  (TRUE   and I8088)         ;Warm boot from extra memory
;
HMDRIVE EQU     TRUE            ;M-DRIVE/H memory drive
HM$NUM  EQU     8               ;Total number of boards (may be
;           RAM if fewer boards than this maximum are ever to be
HMSIZER EQU (TRUE  and HMDRIVE) ;Must be turned off if HM$NUM not
;---------------------------------------
```

EXAMPLE: To change the Floppy drive step rate from 3 ms. to 8 ms. you would change the underlined "3" to "8" shown below.

```
;       Floppy Disk (DISK 1) controller and drive constants:
;
; 8 inch drive characteristics:
FLOPPY8 EQU     TRUE            ;8 inch floppy disk system (Disk 1) prese
FPY8X4  EQU     FALSE           ;4 drives present (2 is default)
STEPR8  EQU     3               ;8 inch drive step rate in milliseconds
ULOAD8  EQU     240             ;Head unload time delay in milliseconds
HDLT8   EQU     35              ;Head load settling time in milliseconds
```

EXAMPLE: To turn on DTR handshaking on the SYSTEM SUPPORT 1 and INTERFACER 3 or 4 boards, you would remove the underlined ";" that turns the OR statement into a comment as shown.

```
;---------------------------------------
;
; Interfacer 3,4 composit status masks and activity indicators:
;
INTERFACER3 EQU TRUE            ;Interfacer 3 board is present
INTERFACER4 EQU TRUE            ;Interfacer 4 board is present
;
IF3TMSK EQU  IF3TBE ;or IF3DSR  ;Xmit ready mask (for either board)
IF3FMSK EQU  IF3TBE ;or IF3DSR  ;Xmit buffer empty flip bit mask
;
;---------------------------------------
; System Support I setup characteristics:
SYSUP1  EQU     TRUE            ;System Support board is present
;
; System Support Uart Active Status Masks:
SS1TMSK EQU  SS1TBE ;or SS1DSR  ;Transmit Ready Mask
SS1FMSK EQU  SS1TBE ;or SS1DSR  ;Transmit Ready Bit Flip Mask
;
; Timer/Counter Commands and Values:
TIMEBASE EQU    10000           ;Divide 2MHz clock for other clock/
;                               (as a more reasonable time base --
TIME1   EQU     20              ;Real Time Clock interval (not acti
TIME2   EQU     200             ; 1 sec interrupt to read clock
;---------------------------------------
```

The key to altering this file is to simply look and see what the comment says, and change the appropriate label. Save the file after altering it and you are ready to recreate the system.

The first thing that must be done is to create a CCP and BDOS that have been relocated to the right memory address for this version of the BIOS. BIOS HMX1BIOS wants to start at E800H, therefore, the BDOS must start at DD00H and the CCP must start at D200H (CCP always starts 1600h bytes below the BIOS). Step one is to tell MOVCPM the size of the BIOS, then MOVCPM will relocate the CCP and BDOS to the right place. Once the file ACTIVE.LIB as been set for your system, the system is built as follows (enter the underlined words):

```
A>STAT MOVCPM.COM

  Recs  Bytes  Ext Acc
   100    14K    1 R/W A:MOVCPM.COM
  Bytes Remaining On A: xK

A>;MOVCPM must be 14K bytes long.
A>DDT MOVCPM.COM
DDT VERSION 2.2
NEXT  PC
3300 0100
-S806
806 14 18
807 00 .
-^C
A>SAVE 50 MOVCPM.COM
A>MOVCPM 64 *

CONSTRUCTING 64K CP/M Vers 2.2
READY FOR "SYSGEN" OR
"SAVE 43 CPM64.COM"
A>SAVE 43 CPM64.COM
A>RMAC HMX1BIOS (or RMAC HMX2BIOS)
CP/M RMAC ASSEM 1.1
F3AF       ( These numbers may change )
04FH USE FACTOR
END OF ASSEMBLY

A>LINK HMX1BIOS [LE800] (or LINK HMX2BIOS [LE800])
LINK 1.3

ABSOLUTE       0BAF (E800-F3AE)  ( These numbers may change )
CODE SIZE      0000
DATA SIZE      0000
COMMON SIZE    0000
USE FACTOR       00

A>RMAC HMXFBOOT
CP/M RMAC ASSEM 1.1
0200       ( These numbers may change )
016H USE FACTOR
END OF ASSEMBLY

A>LINK HMXFBOOT
LINK 1.3

ABSOLUTE       0100 (0100-01FF)  ( These numbers may change )
CODE SIZE      0000
DATA SIZE      0000
COMMON SIZE    0000
USE FACTOR       00
```

```
A>DDT CPM64.COM  (CPMHMX2.COM for HMX2BIOS)
NEXT  PC
2C00 0100
-IHMXFBOOT.COM
-R800
NEXT  PC
2C00 0100
-IHMX1BIOS.COM (or IHMX2BIOS.COM)
-R900
NEXT  PC
2C00 0100
-^C
A>SAVE 43 CPM.COM
A>SYSGEN CPM.COM
SYSGEN Version 2.2D
Destination drive name (or RETURN to terminate).B
Function complete.
Destination drive name (or RETURN to terminate).cr

A>;Now your "B" disk should boot up
```

PART 3) The HMX1BIOS.ASM file is the main body of the BIOS, and HMXFBOOT.ASM file that performs the loader and initialization functions. These are the files that must be changed to alter features that cannot be altered by just changing the ACTIVE.LIB file. After making changes to HMX1BIOS.ASM or HMXFBOOT.ASM, the system must be reassembled using RMAC as it was in PART 2.

To alter these files, load the HMX1BIOS.ASM or HMXFBOOT.ASM file using your text editor after deciding which changes you would like to make. Most changes will be made by changing equates from true to false or false to true, or changing the value of a constant. Examples are shown below.

EXAMPLE: To alter the USART initialization parameters, you would alter the INPUT/OUTPUT DEVICE INITIALIZATION SEQUENCE TABLE in the HMXFBOOT.ASM file below. The parameter values may be found in the INTERFACER 3 or 4 manuals, and the appropriate values may be changed.

```
;*****************************************************************
;*      INPUT/OUTPUT DEVICE INITIALIZATION SEQUENCE TABLE        *
;*****************************************************************
INISEQ: ;Port,  Value to transmit sequence until Port = 0FFh.
;
; Interfacer 3,4 UART initialization.
   DB    IF3UX,  4        ;Select Uart 4
   DB    IF3UM,01011010b ;Async, 16x, 7 bits, odd parity, 1 stop
   DB    IF3UM,01111110b ; 9600 baud
   DB    IF3UC,00100111b ;Trans. on, DTR low, rec. on, no break/
;
   DB    IF3UX,  5        ;Select Uart 5
   DB    IF3UM,01011010b ;Async, 16x, 7 bits, odd parity, 1 stop
   DB    IF3UM,01111110b ; 9600 baud
   DB    IF3UC,00100111b ;Trans. on, DTR low, rec. on, no break/
```

```
;
   DB    IF3UX,  6        ;Select Uart 6
   DB    IF3UM,01011110b ;Async, 16x, 8 bits, odd parity, 1 stop
   DB    IF3UM,01111110b ; 9600 baud
   DB    IF3UC,00100111b ;Trans. on, DTR low, rec. on, no break/
;
   DB    IF3UX,  7        ;Select Uart 7
   DB    IF3UM,01011010b ;Async, 16x, 7 bits, odd parity, 1 stop
   DB    IF3UM,01111110b ; 9600 baud
   DB    IF3UC,00100111b ;Trans. on, DTR low, rec. on, no break/
;
; System Support I UART initialization.
   DB    SS1UM,01011010b ;Async, 16x, 7 bits, odd parity, 1 stop
   DB    SS1UM,01111111b ;19200 baud
   DB    SS1UC,00100111b ;Xmit on, DTR low, rec. on, no break,
;
   DB    0FFh             ;End of I/O port initialization string
;
```

EXAMPLE: To alter the USART handshaking and device configuration, you would alter the INPUT/OUTPUT DEVICE INITIAL SELECT TABLE in the HMXFBOOT.ASM file below.

This is a sample switch that you will find in the table below. This table corresponds directly with S2 positions 1 and 2 on the DISK 1.

SWITCH 2 POSITIONS 1 AND 2

```
                              BOTH OFF    = SWITCH = 3
                              1 ON 2 OFF  = SWITCH = 2
                              1 OFF 2 ON  = SWITCH = 1
                              BOTH ON     = SWITCH = 0
```

EXAMPLE:

```
; Switch = 2
   DB    10000001b        ;LST:=LPT:, PUN:=TTY:, RDR:=TTY:, CON:=CRT:
   DB    00010101b        ;LPT:=Interfacer 3,4 USER 4, CRT:=System Support
```

The two "DB's" above can be decoded as shown below.

```
TOP DB:
   DB    AABBCCDD (line one) AA=LST: 00= TTY: BB= PUN: 00= TTY:
                                     01= CRT:          01= PTP:
                                     10= LPT:          10= UP1:
                                     11= UL1:          11= UP2:

                             CC=RDR: 00= TTY: DD= CON: 00= TTY:
                                     01= PTP:          01= CRT:
                                     10= UP1:          10= BAT:
                                     11= UP2:          11= UC1:
```

```
BOTTOM DB:
   DB  AABBCCDD(line two) AA=LPT: 00= I/O 3&4USER 4   BB= XON/XOFF
                                  01= I/O 1&2 UART 1   00= NO PROTOCOL
                                  10= I/O 1&2 UART 2   01= XON/XOFF
                                  11= SAME AS ABOVE    10= EXT/ACT

                              CC= XON/XOFF      DD=CRT: 00= I/O 3&4 USER 0
                              00= NO PROTOCOL           01= SYSTEM SUPPORT
                              01= XON/XOFF              10= I/O 1&2 UART 1
                              10= EXT/ACT               11= I/O 1&2 UART 2

 And (always):  UC1:= Interfacer 3,4 USER 7
                TTY:= Interfacer 3,4 USER 6
                UL1:= Interfacer 3,4 USER 5 at all times.

The actual listing is shown next:

;*********************************************************
;*      INPUT/OUTPUT DEVICE INITIAL SELECT TABLE         *
;*********************************************************
;
BIOTBL:          ;I/O byte (IOBYTE) value, Aux I/O control byte (IOCNTL) v
; Switch = 0
   DB    10$00$00$01b     ;LST:=LPT:, PUN:=TTY:, RDR:=TTY:, CON:=CRT:
   DB    01$00$00$10b     ;LPT:=Interfacer I UART 1, CRT:=Interfacer I UART
; Switch = 1
   DB    10$00$00$01b     ;LST:=LPT:, PUN:=TTY:, RDR:=TTY:, CON:=CRT:
   DB    00$00$01$00b     ;LPT:=Interfacer 3 USER 4 xon/xoff, CRT:=USER 0
; Switch = 2
   DB    10$00$00$01b     ;LST:=LPT:, PUN:=TTY:, RDR:=TTY:, CON:=CRT:
   DB    00$00$01$01b     ;LPT:=Interfacer 3,4 USER 4, CRT:=System Support
; Switch = 3
   DB    00$00$00$11b     ;LST:=TTY:, PUN:=TTY:, RDR:=TTY:, CON:=UC1:
   DB    00$00$01$01b     ;LPT:=Interfacer 3,4 USER 4, CRT:=System Support
;
; IOBYTE value is the first entry for each switch selection, and --
; IOCNTL =       ww$xx$yy$zzb     selects the following:
;(second byte    ww xx yy 00      CRT:=Interfacer 3 USER 0.
;of each entry   ww xx yy 01      CRT:=System Support I.
;in BIOTBL).     ww xx yy 10      CRT:=Interfacer 1,2 UART 0.
;                   xx yy 11      CRT:=Interfacer 1,2 UART 1 (Custom Routin
;                00 xx yy --      LPT:=Interfacer 3,4 USER 4.
;                01 xx yy --      LPT:=Interfacer 1,2 UART 1.
;                10 xx yy --      LPT:=Interfacer 1,2 UART 2 (Custom Routin
;                11 xx yy --      LPT:=   "         "       "        "     "
;                -- xx -- --      Interfacer 3,4 USER 5 list routine select
;                -- -- yy --      Interfacer 3,4 USER 4 list routine select
;Where xx and/or yy = 00          Straight output, no software protocol.
;                     01          XON/XOFF software protocol active.
;                     10          ETX/ACK software protocol active.
; And (always):
;                UC1:= Interfacer 3,4 USER 7
;                TTY:= Interfacer 3,4 USER 6
;                UL1:= Interfacer 3,4 USER 5 at all times.
;
```

```
;<=========  If CON:=BAT: then --
;/==|    BAT:=   RDR:= Interfacer 3,4 USER 3 when RDR:=UR2: on input.
;\==|__  BAT:=   PUN:=   "                 " 3 when PUN:=UP2: on output.
;/==|    BAT:=   RDR:= Interfacer 3,4 USER 2 when RDR:=UR1: on input.
;\==|__  BAT:=   PUN:=   "                 " 2 when PUN:=UP2: on output.
;/==|    BAT:=   RDR:= Interfacer 3,4 USER 1 when RDR:=PTR: on input.
;\==|__  BAT:=   PUN:=   "                 " 1 when PUN:=PTP: on output.
;x /--|  BAT:=   ----- Interfacer 3,4 USER 0 when RDR:=TTY: on input.
;x \--|  BAT:=   -----   "                 " 0 when PUN:=TTY: on output.
; and for reader/punch vectors only --
;x --    ------  RDR:= Interfacer 3,4 USER 6 when RDR:=TTY: on input.
;x --    ------  PUN:=   "                 " 6 when PUN:=TTY: on output.
;
        END
```

PART 4) This section describes a method of altering the baud rates and floppy step rates in existing CPMxxxx.COM files using DDT. An example for each is shown below.

FOR DISK DRIVE STEP RATES:

```
A>DDT CPMXXXX.COM
DDT VERS 2.2
NEXT  PC
2C00  0100
-S9A0
09A0
09A1 8F {8F for 8ms.,AF for 6ms., DF for 3ms.}
09A2 .
-^C
A>SAVE 43 CPMNEW.COM
```

After saving the file, you must SYSGEN it onto your diskette.

FOR SERIAL BAUD RATES:

```
A>DDT CPMXXXX.COM
DDT VERS 2.2
NEXT  PC
2C00  0100
-S9BF
09BF 04   {RELATIVE USER 4}
09C0 12                          Consult Interfacer 3 or 4 manual for values
09C1 EE   {MODE REGISTER 1}      of the mode and command registers.
09C2 12
09C3 7E   {MODE REGISTER 2}      (79 for 1200 baud, 7E for 9600 baud)
09C4 13
09C5 27   {COMMAND REGISTER}
09C6 17
09C7 05   {RELATIVE USER 5}
09C8 12
09C9 EE   {MODE REGISTER 1}
```

```
09CA 12
09CB 7E    {MODE REGISTER 2}
09CC 13
09CD 27    {COMMAND REGISTER}
09CE 17
09CF 06    {RELATIVE USER 6}
09D0 12
09D1 EE    {MODE REGISTER 1}
09D2 12
09D3 7E    {MODE REGISTER 2}
09D4 13
09D5 27    {COMMAND REGISTER}
09D6 17
09D7 07    {RELATIVE USER 7}
09D8 12
09D9 EE    {MODE REGISTER 1}
09DA 12
09DB 7E    {MODE REGISTER 2}   (7E for 9600 baud, 7F for 19200 baud)
09DC 13
09DD 27    {COMMAND REGISTER}
09DE 5E
09DF EE    {MODE REGISTER 1}
09E0 5E
09E1 7E    {MODE REGISTER 2}   (7E for 9600 baud, 7F for 19200 baud)
09E2 5F
09E3 27    {COMMAND REGISTER}
09E4 FF .
-^C
A>SAVE 43 CPMNEW.COM
```

You must now SYSGEN the new file onto your diskette. This is described below.

III GUIDE TO THE RECREATION OF THE SYSTEM

This section describes how to reassemble your BIOS using RMAC, and how to place your special system onto your diskette. The majority of this section you have seen before in this and other manuals, but it is reproduced here for quick reference.

Before getting into the process of reassembling your BIOS, below is a quick note on how files interact when assembling.

WHICH .ASM FILE USES WHICH .LIB FILE FOR ASSEMBLING:

HMXFBOOT.ASM uses COMPUPRO.LIB and ACTIVE.LIB during assembly.
HMX1BIOS.ASM uses all .LIB files during assembly.
HMXFPROM.ASM uses ACTIVE.LIB and .LIB during assembly.
HMX2IO.ASM uses ACTIVE.LIB, COMPUPRO.LIB,CPMDISK.LIB and ASCII.LIB during assembly.
DSBLINTR.ASM uses all .LIB except BOOTSCPM.LIB.

The first thing that must be done is to create a CCP and

BDOS that have been relocated to the right memory address for this version of the BIOS. BIOS HMX1BIOS wants to start at E800H, therefore, the BDOS must start at DD00H and the CCP must start at D200H (CCP always starts 1600h bytes below the BIOS). Step one is to tell MOVCPM the size of the BIOS, then MOVCPM will relocate the CCP and BDOS to the right place. Once the file ACTIVE.LIB as been set for your system, the system is built as follows (enter the underlined words):

This section deals with the creation of a MOVCPM.COM that know the size of the BIOS.

```
A>STAT MOVCPM.COM    (check the size of MOVCPM)

  Recs  Bytes  Ext Acc
   100    14K    1 R/W A:MOVCPM.COM
  Bytes Remaining On A: xK

A>;MOVCPM must be 14K bytes long.
A>DDT MOVCPM.COM     (load MOVCPM.COM under DDT)
DDT VERSION 2.2
NEXT  PC
3300 0100
-S806
806 14 18            (change the BIOS size)
807 00 .
-^C
A>SAVE 50 MOVCPM.COM      (save the corrected copy)
A>MOVCPM 64 *             (create a new image of CP/M)

CONSTRUCTING 64K CP/M Vers 2.2
READY FOR "SYSGEN" OR
"SAVE 43 CPM64.COM"
A>SAVE 43 CPM64.COM       (save the new image for later use)
```

Now that a new image of CP/M has been made for your size BIOS, we must reassemble and link the BIOS and BOOT LOADER.

```
A>RMAC HMX1BIOS (or RMAC HMX2BIOS)
CP/M RMAC ASSEM 1.1
F3AF       ( These numbers may change )
04FH USE FACTOR
END OF ASSEMBLY

A>LINK HMX1BIOS [LE800] (or LINK HMX2BIOS [LE800])
LINK 1.3

ABSOLUTE        0BAF (E800-F3AE)  ( These numbers may change )
CODE SIZE       0000
DATA SIZE       0000
COMMON SIZE     0000
USE FACTOR        00

A>RMAC HMXFBOOT
CP/M RMAC ASSEM 1.1
```

```
0200       ( These numbers may change )
016H USE FACTOR
END OF ASSEMBLY

A>LINK HMXFBOOT
LINK 1.3

ABSOLUTE        0100 (0100-01FF)  ( These numbers may change )
CODE SIZE       0000
DATA SIZE       0000
COMMON SIZE     0000
USE FACTOR        00
```

Now that we have assembled and linked the BIOS and BOOT LOADER, we must overlay them onto the newly created image of CP/M (CCP and BDOS).

```
A>DDT CPM64.COM  (CPMHMX2.COM for HMX2BIOS) (load the image)
NEXT  PC
2C00 0100
-IHMXFBOOT.COM
-R800                (overlay the BOOT LOADER)
NEXT  PC
2C00 0100
-IHMX1BIOS.COM (or IHMX2BIOS.COM)
-R900                (overlay the BIOS)
NEXT  PC
2C00 0100
-^C
A>SAVE 43 CPM.COM    (save the new image for SYSGENing)
```

Now that the new image is overlayed and complete, it is ready to SYSGEN onto the system tracks of the floppy.

```
A>SYSGEN CPM.COM
SYSGEN Version 2.2D
Destination drive name (or RETURN to terminate).B
Function complete.
Destination drive name (or RETURN to terminate).cr
```

Now the diskette in your "B" drive is ready to boot up. You have just completely recreated the system!

IV CLOSING AND REFERENCE SECTION

CLOSING

In short, we hope that this document helps you in the alteration and recreation of your operating system. CompuPro is open to any suggestions you might have concerning this information. If you find any inaccuracies, errors, or just have some comments or ideas for inclusion, please feel free to write a tell us. Address all correspondence to:

Kevin Fischer
Vice President of Customer Assurance
CompuPro
3506 Breakwater Court
Hayward, CA 94545

REFERENCE SECTION

For further information about CP/M 2.2 you may refer to the following manuals:

CompuPro Digital Research CP/M 2.2 Manual

CompuPro CP/M 2.2 Operating System Technical Manual

You may wish to consult any of the CompuPro Hardware technical manuals for information specific to an individual board.

COMPUPRO
SOFTWARE ALTERATION GUIDE FOR CP/M 86 OR 816 VERSION R
5/3/84

OPENING

This document is a guide to aid the user of CP/M 86 or 816 in its reconfiguration and alteration. Since the BIOS for CP/M 816 is the same as CP/M 86, we will only treat latter. This document includes the following sections:

I REQUIREMENTS AND DEFINITIONS
II A GUIDE TO DESIRED CHANGES WITH STEP-BY-STEP EXAMPLES
III A GUIDE TO THE RECREATION OF THE SYSTEM
IV CLOSING AND REFERENCE SECTION

I REQUIREMENTS AND DEFINITIONS

This section of the manual will describe the features of this version of the operating system and the changes since the last revision, the required files for the modification and recreation of the operating system, a short description of the files involved in creating the CP/M 86 BIOS, and some simple file designator definitions.

FEATURES

The 86 R version of this operating system incorporates the following special features and capabilities:

A) DISK 2 and DISK 3 hard disk support
B) Support for MDRIVE/H
C) Support for minifloppies using DISK 1 - 5
D) Support for interrupts

The new CP/M 86 BIOS supports all of the same physical devices that the previous BIOS did.

CHANGES SINCE LAST REVISION

Sine the last revision, PD, the following changes have been made:

1) ASM86 assembler has been added. Linking is done with GENCMD.CMD.
2) DISK 2 sector allocations are revised requiring reformatting from any previous revision.
3) A Skeleton for 5 1/4" floppy support has been added.
4) Interrupt capability has been introduced.

REQUIREMENTS

Before you begin to use the instructions given here, you should become familiar with certain other documentation that is included with CP/M 86. You should understand the material covered in the Digital Research's CP/M 86 Manuals and the CompuPro Technical Manual & Installation Procedures. If you do not have a word processor, you should carefully study and practice with the line editor, ED.CMD. This document also assumes that you understand the function of the BIOS and the LOADER in the CP/M 86 operating system.

The following list of files and their short descriptions are required to reassemble the system and are included on your master diskette.

Note: Do not modify your master diskette! Make a copy of it and modify the copy! You will receive no sympathy if you destroy your master.

FILES REQUIRED FOR BIOS MODIFICATION

ACTIVE.EQU Contains control variables which set the conditions, according to the particular hardware configuration for BIOS customization.

SINGLES.LIB, DEBLOCK.LIB and 8087.LIB are library files refered to in DRI's manuals. These do not get used by CompuPro's BIOS.

ASCII.EQU Contains definitions for all ASCII control character.

COMPUPRO.EQU Contains data constants common to all the CP/M Operating System's components. For example, Input/Output Port Assignments, USART register bit definitions for I/O boards, etc.

CPMDISK.EQU Contains CP/M disk constants, definition of base page as it relates to file operations, definition of vectors to call CBIOS routines as well as BDOS call function numbers.

TMXDEVIO.INI Contains all I/O drivers for all I/O boards. This file is used for changing the initialization of USART parameters, as well as the type of software protocol for the I/O boards.

TMXDISK 1, 2 AND 3.INI All of these files contain the initialization code specific to the disk controller.

TMXINTIO.DVR Contains the interrupt driven console, list and auxilary device I/O drivers.

TMXDISK 1, 2 AND 3.DVR All of these files contain the driver code for specific to the disk controller.

TMXDISK 1, 2 AND 3.TBL All of these files contain DPB and DPH tables for their repective disk controllers.

TMXCONIO.DVR This file contains Console device I/O drivers for the loader.

TMXXLATE.TBL This file contains sector translation and format tables.

TMXALLOC.TBL This file contains uninitialized storage allocation tables.

MAKLDR86/88.SUB Both files contain a group of sequential operations which generate the proper LOADER.CMD file which is then run with SYSGEN.CMD to place the loader onto the system tracks.

Here are some short descriptions of file name extensions that you will find in the directory.

.CMD = 16 bit executable command file
.COM = 8 bit executable command file
.EQU = equate table file
.DVR = driver routines file
.H86 = hex format file
.A86 = assembly files
.TBL = table of equates
.88 = system file (used with CPU 8085/8088)
.86 = system file (used with CPU 8086)
.INI = initialization routine file
.DEF = Digital Research file
.SYS = operating system file
.SUB = submit file used with the SUBMIT command
.DOC = document file
.TXT = text file
.ASC = ASCII text file
.LST = ASCII list file

II GUIDE TO DESIRED CHANGES WITH STEP-BY-STEP EXAMPLES

This section of the manual is a guide to which files need to be modified to accomplish specific changes in the operating system. It is divided into three parts. Part 1) describes changes that can be made without reassembling the BIOS but just using a different system file already provided. Part 2) describes simple changes to the ACTIVE.EQU file for modifications like floppy drive characteristics, turning M-DRIVES on and off, turning on and off both DISK 2 and 3, handshaking for INTERFACER 3 and 4, etc., and Part 3) describes major changes involving alteration of the TMXFBOOT.A86 and TMXBIOS.A86 files like baud rates, handshaking, I/O protocols, etc.

PART 1) There are several CPMxxxxx.SYS files already configured to run many standard hardware configurations. If one of these configurations fulfills your requirements, you do not need to reassemble your BIOS. To use these configurations, you simply rename the appropriate file to CPM.SYS on your diskette. An example is given at the end of this section.

CPM.SYS	Four 8" floppy drives @ 3 ms. step rate No hard disk Console @ 9600 baud - no handshaking List device @ 9600 baud - no handshaking Supports M-DRIVE/H
CPM210.SYS	DISK2 and 10 Mb hard disk Four 8" floppy drives @ 3 ms. step rate Console @ 9600 baud - no handshaking List device @ 9600 baud - no handshaking Supports M-DRIVE/H
CPM220.SYS	DISK2 and 20 Mb hard disk Four 8" floppy drives @ 3 ms. step rate Console @ 9600 baud - no handshaking List device @ 9600 baud - no handshaking Supports M-DRIVE/H
CPM340.SYS	DISK3 and 40 Mb Quantum Q540 hard disk Four 8" floppy drives @ 3 ms. step rate Console @ 9600 baud - no handshaking List device @ 9600 baud - no handshaking Supports M-DRIVE/H
CPM35.SYS	DISK3 and 5 Mb ST506 type hard disk Four 8" floppy drives @ 3 ms. step rate Console @ 9600 baud - no handshaking List device @ 9600 baud - no handshaking Supports M-DRIVE/H

EXAMPLE FOR USING THE ABOVE FILES:

To use one of the above stock system files, you must first SYSGEN the LOADER onto the system tracks, then rename the .SYS files. Do this only on a copy of the master diskette! Never do this on the master itself! The proceedure for doing this is as follows:

Type the following:

```
SYSGEN LDRxx.CMD<CR>      (where "xx" is 88 when using
                          CPU 85/88, and 86 when using
                          CPU 86/87)
```

It will respond asking for the destination drive name or hit return to terminate. You should respond with the drive name that corresponds to the floppy that you wish to have the new system placed on.

Now type the following:

```
REN CPMOLD.SYS=CPM.SYS        (save the old .SYS file)
```

Then type:

```
PIP CPM.SYS=CPMxxxx.SYS[V]    (create a copy of the desired
                              .SYS file and name it
                              CPM.SYS)
```

You should now be able to BOOT your new diskette.

PART 2) Many changes to the system may be made by altering the ACTIVE.EQU file and then simply reassembling the BIOS and the LOADER using ASM86.CMD. The easiest way to do this is to first print out the ACTIVE.EQU and TMXDEVIO.INI files on your printer for reference. After reviewing the printout and deciding which changes you would like to make, then load the ACTIVE.EQU or TMXDEVIO.INI file using your text editor. Most changes will be made by changing equates from true to false or false to true, or changing the value of a constant. Examples are shown below.

The following tables are a composite of TMXDEVIO.INI and ACTIVE.EQU. You should refer to the actual file for additional patches not covered in this section.

The first section deals with USART initialization. These values may be changed in TMXDEVIO.INI.

The following example will show you the basics of changing USART initialization. For INTERFACER 3 or 4, four bytes are defined:

```
   DB    IF3UX,  0    ; This is the relative user number.
   DB    IF3UM,01011010b ; This is mode register 1.
   DB    IF3UM,01111110b ; This is mode register 2.
   DB    IF3UC,00100111b ; This is the command register.
```

Mode register 1 = ABCDEFGH

AB= STOP BIT LENGTH	C=PARITY	D=PARITY CONTROL	EF=CHAR.LENGTH
00 for none	0= odd	0=disable	00 = 5 bits
01 for 1 stop	1=even	1=enable	01 = 6 bits
10 for 1 1/2 stop			10 = 7 bits
11 for 2 stop			11 = 8 bits

GH=MODE BAUD FACTOR
00=synchronous 1x rate
01=asynchronous 1x rate
10=asynchronous 16x rate
11=asynchronous 64x rate

Mode register 2 = ABCDEFGH

AB=NOT USED	C=TRANSMITTER CLOCK	D=RECIVER CLOCK	EFGH =BAUD SELECTION	
	0=external	0=external	0000=50	1000=1800
	1=internal	1=internal	0001=75	1001=2001
			0010=110	1010=2400
			0011=134	1011=3600
			0100=150	1100=4800
			0101=300	1101=7200
			0110=600	1110=9600
			0111=1200	1111=19.2k

COMMAND REGISTER = ABCDEFGH

AB=OPERATING MODE	C=RTS	D=RESET	E=ASYNC OR SYNC	F=REC.CNTRL.
00=normal	0=high	0=normal	0=normal	0=disable
01=async auto echo	1=low	1=reset flag	1=force break or send DLE	1=enable
10=local loop				
11=remote loop				

G=DTR	H=TRANSMIT CNTRL.
0=high	0=disable
1=low	1=enable

An example from the file:

```
      if INTERFACER3 or INTERFACER4
   DB    IF3UX,  4        ;Select Uart 4
   DB    IF3UM,11101110b ;Async, 16x, 8 bits, No parity, 2 stop
   DB    IF3UM,01111110b ; 9600 baud
   DB    IF3UC,00100111b ;Trans. on, DTR low, rec. on, no break/
;
```

```
   DB    IF3UX,  5        ;Select Uart 5
   DB    IF3UM,11101110b ;Async, 16x, 8 bits, No parity, 2 stop
   DB    IF3UM,01111110b ; 9600 baud
   DB    IF3UC,00100111b ;Trans. on, DTR low, rec. on, no break/
;
   DB    IF3UX,  6        ;Select Uart 6
   DB    IF3UM,01101110b ;Async, 16x, 8 bits,  no parity, 1 stop
   DB    IF3UM,01110111b ; 1200 baud
   DB    IF3UC,00100111b ;Trans. on, DTR low, rec. on, no break/
;
   DB    IF3UX,  7        ;Select Uart 7
   DB    IF3UM,01011010b ;Async, 16x, 7 bits, odd parity, 1 stop
   DB    IF3UM,01111110b ; 9600 baud
   DB    IF3UC,00100111b ;Trans. on, DTR low, rec. on, no break/
```

For the SYSTEM SUPPORT 1 console USART, there are three defined bytes:

```
; System Support I UART initialization.
      if SYSUP1
   DB    SS1UM,11101110b ;Async, 16x, 8 bits, No parity, 2 stop
   DB    SS1UM,01111110b ;9600 baud
   DB    SS1UC,00100111b ;Xmit on, DTR low, rec. on, no break, run,
```

By substituting the appropriate values into the above statements in the file, you will change the configuration of the port. You must reassemble the BIOS and LOADER of course...

This is a sample switch that you will find in the table below. By modifying these values, you can redefine your I/O configuration. This table corresponds directly with S2 positions 1 and 2 on the DISK 1.

SWITCH 2 POSITIONS 1 AND 2

BOTH OFF	= SWITCH 3
1 ON 2 OFF	= SWITCH 2
1 OFF 2 ON	= SWITCH 1
BOTH ON	= SWITCH 0

From the two defined bytes below, you can determine the exact I/O configuration.

```
; Switch = 2
   DB    10000001b        ;LST:=LPT:, PUN:=TTY:, RDR:=TTY:, CON:=CRT:
   DB    00010101b        ;LPT:=Interfacer 3,4 USER 4, CRT:=S. S. 1
```

The defined bytes may be decoded as follows.
TOP BYTE:

```
   DB    AABBCCDD (line one) AA=LST: 00= TTY: BB= PUN: 00= TTY:
                                     01= CRT:          01= PTP:
                                     10= LPT:          10= UP1:
                                     11= UL1:          11= UP2:
```

```
                                CC=RDR: 00= TTY: DD= CON: 00= TTY:
                                        01= PTP:          01= CRT:
                                        10= UP1:          10= BAT:
                                        11= UP2:          11= UC1:

LOWER BYTE:

 DB AABBCCDD(line two) AA=LPT:00= I/O 3&4USER 4   BB= XON/XOFF
                              01= I/O 1&2 UART 1  00= NO PROTOCOL
                              10= I/O 1&2 UART 2  01= XON/XOFF
                              11= SAME AS ABOVE   10= EXT/ACT

                         CC= XON/XOFF     DD=CRT: 00= I/O 3&4 USER 0
                         00= NO PROTOCOL          01= SYSTEM SUPPORT 1
                         01= XON/XOFF             10= I/O 1&2 UART 1
                         10= EXT/ACT              11= I/O 1&2 UART 2

 And (always):  UC1:= Interfacer 3,4 USER 7
                TTY:= Interfacer 3,4 USER 6
                UL1:= Interfacer 3,4 USER 5 at all times.


THIS IS THE TABLE AS PRESENTED WITHIN TMXDEVIO.INI:

;*******************************************************
;*      INPUT/OUTPUT DEVICE INITIAL SELECT TABLE       *
;*******************************************************
BIOTBL  RS 0    ;I/O byte (IOBYTE) value, Aux I/O control byte (IOCNTL)
; Switch = 0
   DB   10000001b       ;LST:=LPT:, PUN:=TTY:, RDR:=TTY:, CON:=CRT:
   DB   01000010b       ;LPT:=Interfacer I UART 1, CRT:=Interfacer I 0
; Switch = 1
   DB   10000001b       ;LST:=LPT:, PUN:=TTY:, RDR:=TTY:, CON:=CRT:
   DB   00010100b       ;LPT:=Interfacer 3 USER 4 xon/xoff, CRT:=USER 0
; Switch = 2
   DB   10000001b       ;LST:=LPT:, PUN:=TTY:, RDR:=TTY:, CON:=CRT:
   DB   00010101b       ;LPT:=Interfacer 3,4 USER 4, CRT:=System
; Switch = 3
   DB   00000011b       ;LST:=TTY:, PUN:=TTY:, RDR:=TTY:, CON:=UC1:
   DB   00010101b       ;LPT:=Interfacer 3,4 USER 4, CRT:=System
;
; IOBYTE value is the first entry for each switch selection, and --
; IOCNTL =      ww$xx$yy$zzb    selects the following:
;(second byte   ww xx yy 00     CRT:=Interfacer 3 USER 0.
;of each entry  ww xx yy 01     CRT:=System Support I.
;in BIOTBL).    ww xx yy 10     CRT:=Interfacer 1,2 UART 0.
;                  xx yy 11     CRT:=Interfacer 1,2 UART 1 (Cust Rout).
;               00 xx yy --     LPT:=Interfacer 3,4 USER 4.
;               01 xx yy --     LPT:=Interfacer 1,2 UART 1.
;               10 xx yy --     LPT:=Interfacer 1,2 UART 2 (Cust Rout).
;               11 xx yy --     LPT:=    "        "       "       "
;               -- xx -- --     Interfacer 3,4 USER 5 list routine sel,
;               -- -- yy --     Interfacer 3,4 USER 4 list routine sel,
;Where xx and/or yy = 00        Straight output, no software protocol.
;                     01        XON/XOFF software protocol active.
```

```
;                       10        ETX/ACK software protocol active.
; And (always): UC1:= Interfacer 3,4 USER 7
;               TTY:= Interfacer 3,4 USER 6
;               UL1:= Interfacer 3,4 USER 5 at all times.
;<=========  If CON:=BAT: then --
;/==|    BAT:=   RDR:= Interfacer 3,4 USER 3 when RDR:=UR2: on input.
;\==|__  BAT:=   PUN:=   "                 " 3 when PUN:=UP2: on output.
;/==|    BAT:=   RDR:= Interfacer 3,4 USER 2 when RDR:=UR1: on input.
;\==|__  BAT:=   PUN:=   "                 " 2 when PUN:=UP2: on output.
;/==|    BAT:=   RDR:= Interfacer 3,4 USER 1 when RDR:=PTR: on input.
;\==|__  BAT:=   PUN:=   "                 " 1 when PUN:=PTP: on output.
;x /--|  BAT:=   ----- Interfacer 3,4 USER 0 when RDR:=TTY: on input.
;x \--|  BAT:=   -----   "                 " 0 when PUN:=TTY: on output.
; and for reader/punch vectors only --
;x --    ------  RDR:= Interfacer 3,4 USER 6 when RDR:=TTY: on input.
;x --    ------  PUN:=   "                 " 6 when PUN:=TTY: on output.
```

By changing the value in the above tables, you can change the device configuration in your system.

Below is an example of where to change the floppy step rate and the hardware serial port handshaking. The follwing table is contained in ACTIVE.EQU. Any of these values may be changed to meet special hardware needs.

The two underlined semicolons in each grouping may be removed to enable DTR protocol in both sections either I/O 3 and 4 or SYSTEM SUPPORT 1.

THIS IS THE TABLE AS SHOWN IN ACTIVE.EQU:

```
;Interfacer 3,4 composite status masks and activity indicators:
;
INTERFACER3     EQU     TRUE    ;Interfacer 3 board is present
INTERFACER4     EQU     TRUE    ;Interfacer 4 board is present
;
IF3TMSK EQU  IF3TBE; + IF3DSR   ;Xmit ready mask (for either board)
IF3FMSK EQU  IF3TBE; + IF3DSR   ;Xmit buffer empty flip bit mask


; System Support I setup characteristics:
SYSUP1  EQU     TRUE            ;System Support board is present
;
; System Support Uart Active Status Masks:
SS1TMSK EQU  SS1TBE; + SS1DSR   ;Transmit Ready Mask
SS1FMSK EQU  SS1TBE; + SS1DSR   ;Transmit Ready Bit Flip Mask
```

To alter the configuration parameters of the floppy disk drives, change the underlined constants.

```
;CompuPro Floppy Disk (DISK 1 or DISK 1A) controller and drive constants:
;
; 8 inch drive characteristics:
FLOPPY8 EQU     TRUE            ;8 inch floppy disk system (Disk 1)
FPY8X4  EQU     TRUE            ;4 drives present (2 is default)
STEPR8  EQU     (3-8ms.)        ;8 inch drive step rate in millisec
ULOAD8  EQU     240             ;Head unload time delay in millisec
HDLT8   EQU     35              ;Head load settling time in millisec
```

PART 3) This section deals with the alteration and reassembly of the actual TMXBIOS.A86 and TMXFBOOT.A86 files. When changes are made to the operating system, they can affect the BIOS, the LOADER, or both.

MODIFICATION REQUIREMENT TABLE

The following table explains which files must be modified in order to accomplish desired changes in the operating system.

DESIRED CHANGE	MODIFY BIOS	MODIFY LOADER
SYSTEM CONSOLE I/O	XXXXXXXXX	XXXXXXXXXX
I/O BAUD RATE		XXXXXXXXXX
FLOPPY DISK STEP RATE		XXXXXXXXXX
HARD DISK SELECT	XXXXXXXXX	
FLOPPY/HARD DISK ORDER	XXXXXXXXX	
EXTRA FLOPPY DISK SELECT	XXXXXXXXX	

The BIOS and the LOADER are assembled from many files. Below is a description of which of the above uses which .EQU, .TBL, .DVR OR .INI file for assembling:

For the LOADER, TMXFBOOT.A86 uses:

COMPUPRO.EQU	TMXDISK1.INI
ACTIVE.EQU	TMXDISK3.INI
TMXDISK1.TBL	TMXDEVIO.INI
TMXDISK2.TBL	TMXDISK1.DVR
TMXDISK3.TBL	TMXDISK2.DVR
TMXXLATE.TBL	TMXDISK3.DVR
TMXALLOC.TBL	TMXCONIO.DVR

For the BIOS, TMXBIOS.A86 uses:

```
COMPUPRO.EQU
ACTIVE  .EQU              TMXDISK1.INI
TMXDISK1.TBL              TMXDISK2.INI
TMXDISK2.TBL              TMXDISK3.INI
TMXDISK3.TBL              TMXDEVIO.INI
TMXALLOC.TBL              TMXDISK1.DVR
TMXXLATE.TBL              TMXDISK2.DVR
TMXINTIO.INI              TMXDISK3.DVR
TMXINTIO.DVR              TMXDEVIO.DVR
```

III A GUIDE TO THE RECREATION OF THE SYSTEM

After you have completed the changes you wish to make within the appropriate files, then you will need to reassemble the BIOS and LOADER.

There are .SUB routines prepared to execute each step in assembling and linking for you. The files are called MAKLDR??.SUB for the LOADER, and MAKSYS.SUB for the BIOS. To evoke these files you will type in the following:

```
     SUBMIT MAKSYS.SUB <CR>
or   SUBMIT MAKLDR88.SUB <CR>  (for CPU 85/88)
or   SUBMIT MAKLDR86.SUB <CR>  (for CPU 86/87)
```

MAKSYS.SUB consists of the following commands:

```
; Make CPM86.SYS file for System 8-16
ASM86 TMXBIOS $$$1
PIP CPMX.H86=CPM86.H86,TMXBIOS.H86
GENCMD CPMX 8080 code[A41]
PIP CPM.SYS=CPMX.CMD
```

This command file assembles the BIOS and creates a file called CPMX.CMD and a new CPM.SYS file.

After this submit routine is complete, you will need to submit the MAKLDR??.SUB routine. Be sure to choose the proper routine (88 or 86) depending on your CPU type.

MAKLDR88.SUB consists of the following commands:

```
;Make LOADER.CMD file for CP/M 86
ASM86 TMXLOAD $$$1
PIP LOADER.H86=LDCPM.H86,LDBDOS.H86,TMXLOAD.H86
GENCMD LOADER 8080 code[A800]
PIP LDR88.CMD=TMXFBOOT.88,LOADER.CMD
ERA LOADER.CMD
;SYSGEN LDR88.CMD
```

MAKLDR86.SUB consists of the following commands:

```
;Make LOADER.CMD file for CP/M 86
ASM86 TMXLOAD $$$1
PIP LOADER.H86=LDCPM.H86,LDBDOS.H86,TMXLOAD.H86
GENCMD LOADER 8080 code[A800]
PIP LDR86.CMD=TMXFBOOT.86,LOADER.CMD
ERA LOADER.CMD
;SYSGEN LDR86.CMD
```

After your submit routine is complete, you will need to SYSGEN the LOADER onto the system tracks. Perform this task as follows:

```
SYSGEN LDR??.CMD <CR>     (??=88 or 86 depending on CPU)
COPYSYS VER.??
Destination drive name (or return to terminate). B
```

That completes the regeneration process on both the operating system and the loader.

IV CLOSING AND REFERENCE SECTION

CLOSING

In short, we hope that this document helps you in the alteration and recreation of your operating system. CompuPro is open to any suggestions you might have concerning this information. If you find any inaccuracies, errors, or just have some comments or ideas for inclusion, please feel free to write a tell us. Address all correspondence to:

Kevin Fischer
Vice President of Customer Assurance
CompuPro
3506 Breakwater Court
Hayward, CA 94545

REFERENCE SECTION

For further information about CP/M 86 you may refer to the following manuals:

CompuPro CP/M 86 Operating System Technical Manual
Digital Research CP/M-86 Operating System System Guide
Operating System Programmer's Guide
Operating System User's Guide

You may wish to consult any of the CompuPro Hardware technical manuals for information specific to an individual board.

CP/M-68K BIOS and LOADER MODIFICATION GUIDE

SOFTWARE REV 1.1K
DOCUMENT REV 1.0, 29 MAY 84

This is a guide to modifying the CompuPro CP/M-68K BIOS and loader. You can change console or printer baud rates, change the floppy disk drive step rate, or set up the system to talk to CompuPro supported hard disk drives by following instructions in this document.

The information provided here is organized according to the following outline:

1) A status report on the features of the current version (1.1K) of CP/M-68K.

2) A list of of the files, documents and programs needed to accomplish the modifications as well as definitions of the files and their function.

3) A list of changes made since the last released revision of CP/M-68K (1.1H) and their significance to the user.

4) A tabular guide to which file(s) (BIOS or loader) must be modified in order to accomplish common changes to the operating system.

5) An explanation of what alternate configurations are already supplied with the system and how to put them in place of the standard configuration.

6) An explanation of what alternate loaders are already supplied with the system and how to put them in place of the standard loader.

7) A list of data constants that are likely to be changed in the BIOS and their locations.

8) A list of data constants that are likely to be changed in the loader and their locations.

9) A few words on using the CP/M-68K editor "ED.68K" to actually change the files.

10) A step-by-step procedural guide to generating a new CPM.SYS file from a modified BIOS.

11) A step-by-step procedural guide to generating a new loader and placing it on the system tracks of a diskette.

12) A step-by-step procedural guide to setting up a BIOS to automatically execute a command line at boot time.

13) And finally, a guide to where further information can be found regarding the subjects covered here.

THE CURRENT REVISION: CP/M-68K 1.1 K

Revision 1.1 K of CP/M-68K follows revision 1.1 H; revisions I and J were used for in-house development, and were not released to the field. There is one major difference between the K and H revisions: the K revision incorporates the FORTH COMPILER as a module called FORTH-83 that runs in the CP/M environment, the H revision included FORTH as a separate, stand-alone operating system called MAPFORTH. (MAPFORTH is still available as a separate package.) FORTH-83 is considerably different from MAPFORTH in that it uses CP/M to talk to disk and I/O. Also, a number of new words are supported. Refer to the FORTH-83 documentation provided with the CP/M-68K package and on distribution diskette #3 for further information.

There are some minor changes in the K revision that should be noted:

1) The logical order of the disk drives in a system with both hard disk and floppy disk can now be changed by changing an equate in the BIOS called "ORDER". (See the section on BIOS DATA CONSTANTS.)

2) The floppy disk drive step rate can now be changed by changing an equate in the loader called "STEPR8". (See the section on LOADER DATA CONSTANTS.)

3) The BIOS now supports two printer devices for the INTERFACER 3 or 4: device LPT at relative user 4, and device UL1 at relative user 5.

NEEDED FILES AND REFERENCE MATERIAL

Before you begin to use the instructions given here, you should become familiar with certain other documentation that is included with CP/M-68K. You should understand the material covered in the DIGITAL RESEARCH CP/M-68K OPERATING SYSTEM USER'S GUIDE (past users of CP/M 80 or CP/M-86 will already know most of this material). Practice with the line editor ED.68K in particular will be useful for those who have ignored ED on other CP/Ms in favor of word processing screen editors. An understanding of sections five and six of the Digital Research CP/M-68K PROGRAMMER'S MANUAL, though not necessary, might prove useful to the more experienced programmer. You should also read the CompuPro CP/M-68K manual to get some background information. (You may find that you do not need this guide after reading the CompuPro CP/M-68K manual.) You should also understand the function of the BIOS and loader in the CP/M operating system.

Most of the actual files and programs you will need are available on diskettes #1 and #2 of the distribution set of three diskettes. Those files that are not supplied can be created following instructions given below.

The assembler provided with CP/M-68K is different from assemblers provided with CP/M 80 or CP/M-86 (primarily because it creates relocatable code). Consequently, the fields of CP/M-68K filenames follow a different format than those of the other CP/Ms, and an explanation of filename fields will clarify your understanding of the function of files listed below.

In CP/M-68K, an assembly language source file is identified by the field ".S" (e.g. BIOS.S), similar to ".ASM" in CP/M 80. When that source file is assembled, the result is an object code file identified by the field ".O" (e.g. BIOS.O), similar to ".HEX" in CP/M 80.

The object code file must then be linked into an executable command file which may or may not be relocatable. If the linked file is to be relocatable, as is the case when linking BIOS.O, the result is a file identified by the field ".REL" (e.g. BIOS.REL).

The relocatable file is already executable, but it may be necessary to relocate it, as is the case with BIOS.REL. When a relocatable file is relocated, the result is an executable file identified by the field ".68K", similar to ".COM" in CP/M 80.

Here is a list of the names and functions of particular files that are used when modifying the BIOS or loader:

BIOS.S — the source file for the BIOS.

LBIOS.S — a source file for the part of the loader containing code which is mostly device specific. Modifications to the loader will be done in this file.

BOOT.S — a source file for the loader which contains the simple routines which load the rest of the loader. Since this file is so simple, it should not need modification when modifying the loader.

CPM.SYS — the executable BIOS file which is loaded into system memory by the loader at the boot time.

CPMLDR.SYS — the executable loader file which is loaded from the system tracks of the boot diskette at boot time. This file is not supplied on the distribution masters but can be easily generated.

CPMLDR??.SYS — files which are variations of the standard loader file.

CPM?????.SYS — files which are executable variations of the standard BIOS file.

AS.68K — the assembler program provided by Digital Research with CP/M-68K.

LO.68K — the linker program provided by Digital Research with CP/M-68K.

RELOC.68K — the relocator program provided by Digital Research with CP/M-68K.

PUTBOOT.68 — the program used to place a loader file onto the system tracks of a diskette.

ED.68K — the editor program provided by Digital Research with CP/M-68K.

DDT.68K — a debugging program provided by Digital Research with CP/M-68K.

DDT1.68K — a DDT overlay file.

CPMLIB — an archive file containing Digital Research supplied object code functions that are linked to the BIOS.S file using the program LO.68K.

LDRLIB — an archive file containing Digital Research supplied object code functions that are linked to the LBIOS.S file using the program LO.68K.

AS68SYMB.DAT — a data file that must be available on the currently logged disk when running the assembler program AS.68K

MAKESYS.SUB — a submit file which assembles and links the BIOS.S file. MAKESYS.SUB provides the correct command lines for proper assembly, and produces the relocatable file CPM.REL.

RC.SUB — a submit file which relocates the file CPM.REL. RC.SUB provides the correct command lines for proper relocation and produces the executable BIOS file CPM.SYS.

MAKELDR.SUB — a submit file which assembles and links the loader file LBIOS.S. MAKELDR.SUB provides the correct command lines for proper assembly and produces the executable loader file CPMLDR.SYS.

MODIFICATION REQUIREMENT TABLE

The following table explains which files must be modified in

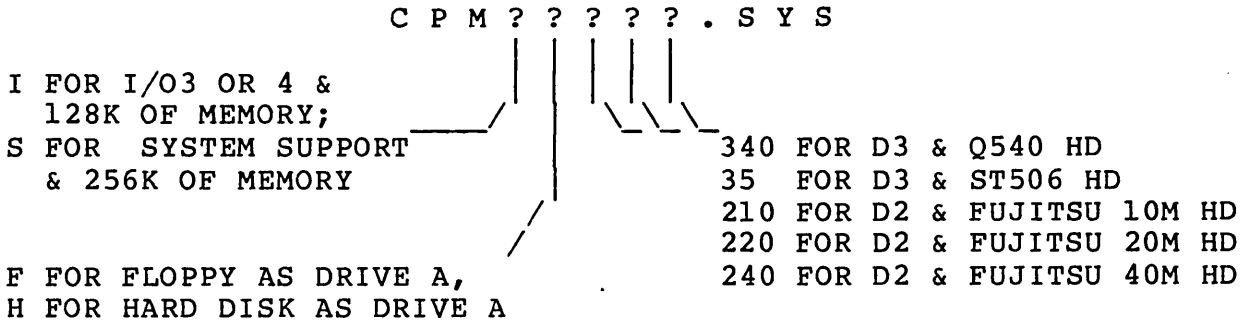
SUPPLIED VARIATIONS OF THE CPM.SYS FILE

In an effort to make it easier for you to modify your standard operating system to accommodate commonly supported options, the CP/M-68K distribution diskettes (specifically #1 and #2) come with several executable BIOS files which are variations of the standard BIOS. These files are named

CPM?????.SYS

where the ? s are optional characters which denote various configurations supported by a particular BIOS.

Here is a chart of what each ? character can contain:



For example, a file named CPMSH340.SYS supports the SYSTEM SUPPORT serial port as its system console port, 256 Kbytes of memory, and a DISK3 with a Quantum Q540 hard disk.

If you wish to set up a boot diskette that loads a BIOS that is supplied in one of the CPM?????.SYS files, then you simply rename that file to CPM.SYS and put it in place of the original CPM.SYS file on your boot diskette.

For example, let's set up a boot diskette for a SYSTEM SUPPORT, 256K of memory, and a DISK3 with a Quantum Q540 hard disk. First, make copies of your master diskettes and put the originals away for safe keeping. On the copy of distribution diskette #1, erase all CPM?????.SYS files except CPMSH340.SYS and CPM.SYS (this will make room on the diskette). Next, rename the CPM.SYS file to CPMFLPY.SYS. Finally, rename the CPMSH340.SYS file to CPM.SYS. The diskette will now boot and load the Quantum Q540 hard disk as drives A through E.

order to accomplish desired changes in the operating system.

DESIRED CHANGE	MODIFY BIOS	MODIFY LOADER
SYSTEM CONSLE I/O	X	X
I/O BAUD RATE		X
FLOPPY DISK STEP RATE		X
HARD DISK SELECT	X	
FLOPPY/HARD DISK ORDER	X	
MEMORY SIZE	X	
EXTRA FLOPPY DISK SELECT	X	

SUPPLIED VARIATIONS OF THE STANDARD LOADER

In order to make it easy for the user who wants to change the loader on his system boot diskette, distribution diskette #1 comes with two executable loader files:

1) CPMLDRSS.SYS - a loader that initializes the SYSTEM SUPPORT serial port as the system console I/O device, and the floppy disk drive step rate to 3 milliseconds. This loader is already on the system tracks of distribution diskette #1.

2) CPMLDRIO.SYS - a loader that initializes the I/O3 or 4 relative user 7 serial port as the system console I/O device, and the floppy disk drive step rate to 8 milliseconds. This loader is already in place on the system tracks of distribution diskette #2.

Now, let's put the loader for SYSTEM SUPPORT console, 256K, and 3 millisecond step rate on a blank diskette. First, format the write-enabled blank diskette, and PIP the following files to it: PUTBOOT.68K and CPMLDRSS.SYS. Next, log on to this newly formatted diskette, and type:

"PUTBOOT CPMLDRSS.SYS Y"

where "Y" is the logical letter of the currently logged drive.

This puts the loader file onto the system tracks of the diskette. Now the diskette only needs an executable BIOS file (CPM.SYS) to be a boot diskette.

BIOS DATA CONSTANTS

If neither the standard BIOS file nor the alternate executable BIOS files that are supplied with the distribution diskettes are adequate for your use, you may want to change the BIOS source file (BIOS.S) and reassemble it to produce a custom BIOS. The simplest way to change the BIOS.S file is to change certain data constants (assembly language "equates") that are declared in its first 50 lines. Here is a list of the constants, what line number of the source code they are found on, what they determine, and how to set them.

1) "SYSSUP", line 23 - determines console I/O device. Set SYSSUP to "0" for console device at SYSTEM SUPPORT serial port; set it to "1" for console device at INTERFACER 3 or 4 relative user 7. An equate of the same name must be set the same way in

the loader source.

2) "HIMEM", line 25 - determines top of transient program area (TPA). Change this when you want your system to reside in a larger block of memory than 128K or 256K (smaller blocks are not recommended as they do not leave enough transient program area). HIMEM is a hex value that should always follow this formula:

HIMEM = [(highest memory address + 1)H - B000H].

For example, in a system with 256 Kbytes of memory, the highest address in hex is 3FFFF. Using the above formula:

HIMEM = [(3FFFF + 1)H - B000H] = 35000

Note that a change in the value of HIMEM must be taken into account when relocating the assembled BIOS (see below).

3) "ORDER", line 27 - determines the order of disk drive logical letters when the system contains a floppy and a hard disk. Setting ORDER to "0" for a floppy/hard system makes the floppy drives correspond to logical drives A through D, and the hard drive correspond to logical drives E through I. (In systems with less than four floppies, the logical partitioning of the hard drive begins with C.) Setting ORDER to "1" for a floppy/hard system makes the hard drive correspond to logical drives A through E, and the floppy drives correspond to logical drives F through I. (In systems with an ST-506 type 5-Megabyte drive, the hard drive will only occupy logical drive A. But the logical partitioning of the floppy drives will still begin with E unless the equate "FDBASE" is changed.) In floppy-only systems, ORDER must be set to "0".

4) "XD8", line 29 - determines whether the system supports two or four floppies. Set XD8 to "0" for four floppies, and to "1" for only two floppies.

5) "FDBASE", line 31 - determines the logical number of the first floppy drive. Set FDBASE to "0" for floppy-only systems and floppy/hard systems where ORDER (see above) is set to "0"; set FDBASE to the number of logical partitions of the hard drive in hard/floppy systems where ORDER is set to "1". (Number of logical partitions is 5 for Q540 and 1 for ST-506.)

6) "D2????", lines 36 to 39 - equates to select DISK2 drivers for one of four types of hard drives. "????" variations depend on the type of hard drive; here are the options supported:

A) "M10" - 10 Mbyte Fujitsu drive (2301 series)
B) "M20" - 20 Mbyte Fujitsu drive (2302 series)
C) "F20B" - BE type 20 Mbyte Fujitsu drive (23XX-BE series)
D) "F40B" - BE type 40 Mbyte Fujitsu drive (23XX-BE series)

Set one of these equates to "0" to assemble the BIOS with drivers for the DISK2 with that particular drive. Only one hard disk driver equate (DISK2 or DISK3) should be set to "0".

7) "D3??", lines 43 and 44 - equates to select DISK3 drivers to one of two types of hard drive. "??" variations depend on the type of hard drive; here are the options supported:

A) "M5" - 5 Mbyte ST-506 type drive
B) "M40" - 40 Mbyte Quantum drive (Q540 series)

Set one of these equates to "0" to assemble the BIOS with drivers for the DISK3 with that particular drive. Only one hard disk driver equate (DISK2 or DISK3) should be set to "0".

LOADER DATA CONSTANTS

If neither of the alternate executable loader files that are provided with the distribution diskettes are adequate for your needs, you may want to change the loader source file and reassemble it to produce a custom loader. There are actually two source files for the loader: BOOT.S and LBIOS.S. Since BOOT.S is, by design, very simple, it should be left intact; any changes to the loader should be made in the file LBIOS.S.

The simplest way to modify the LBIOS.S file is to change the data constants (assembly language equates) or certain initialization bytes which are found in the first seventy lines of the file. Here is a list of those constants and initialization bytes, what line number of the source code they are found on, what they determine, and how to set them.

1) "SYSSUP", line 12 - determines console device. Set "SYSSUP" to "0" for console device at SYSTEM SUPPORT, and to "1" for console device at INTERFACER 3 or 4, relative user 7. An equate of the same name in the BIOS.S file must be set in the same way.

2) "STEPR8", line 29 - determines the step rate for the floppy disk drive(s). Set it to the number of the desired actual step rate in milliseconds (3 for 3 mSecs., 8 for 8 mSecs.).

3) I/O device protocol (baud rate, parity, etc.) bytes - lines 42 and 43 for SYSTEM SUPPORT as console device, lines 50 and 51 for INTERFACER 3 or 4 as console device, lines 57 and 58 for PRINTER 0, and lines 62 and 63 for printer 1. The specific bytes which can be changed are underlined in the following excerpts from the code:

```
console device routines          "move.b #$ee, ...
                                  move.b #$3e, ..."

printer routines                 "move.b #$ee, ...
                                  move.b #$7e, ..."
```

The underlined bytes are bytes which initialize the mode registers of the programmable USART chips on the I/O cards. They can be changed according to the charts provided on the top half of page eighteen of the INTERFACER 4 MANUAL.

For example, if the console I/O device is the SYSTEM SUPPORT, and you wish to change its baud rate to 19.2 kilobaud, replace the hex byte "3e" on line 43 with the hex byte "3f".

EDITING THE SOURCE FILES

Once you know where to find the constants and initialization bytes in the source files for the loader and BIOS, the next step is changing them with an editor. The only text editor that comes with CompuPro's CP/M-68K is the program ED.68K, supplied by Digital research with CP/M-68K. Instructions on its use are in the CP/M-68K USER'S GUIDE (pages 79-103), also supplied by Digital Research with CP/M-68K. If you are not already familiar with ED.68K (it is almost identical to ED.COM and ED.CMD) then you should practice the basic append, insert, delete and exit commands on a garbage file before attempting to edit an important source file. (Of course no mistakes could be at all disastrous because you have already copied and stored the distribution masters and are only working with the copies.)

One tip for those using ED.68K: after opening a file with ED.68K, you have to append lines from it into your working text buffer. The simplest way to do this is by typing "#A"; this will append all of the lines of the opened file into your buffer. Unfortunately, the file BIOS.S is too large to fit into the ED.68K buffer, so you should just append the first 200 or so lines as they include all of the code you would normally want to change. (Note that the file LBIOS.S will entirely fit into the ED.68K buffer.)

REASSEMBLING A MODIFIED BIOS

Once the desired changes have been edited into BIOS.S source file, it must then be assembled, linked with the library routines supplied by Digital Research, and relocated for the appropriate memory size. To make those steps as easy as possible, two submit files have been supplied on distribution diskette #1: MAKESYS.SUB which assembles and links the BIOS, and RC.SUB which relocates it.

MAKESYS.SUB is made up of the following three command lines:

1) "as -l BIOS.S" which assembles the file BIOS.S, ensures that all address constants are generated as longwords, and produces an object code output file called BIOS.O.

2) "lo -r -ucpm -o cpm.rel cpmlib BIOS.O" which links BIOS.O with the CPMLIB file of Digital Research routines, preserves the

relocation bits, and produces a relocatable output file called CPM.REL.

3) "era BIOS.O" which erases the post-assembled, pre-linked file BIOS.O to save disk space. (When MAKESYS.SUB is run on systems with large amounts of disk space, this line may be considered unnecessary, and edited out.)

In order to run MAKESYS.SUB, you will need the following files on the currently logged, write-enabled disk drive:

BIOS.S	AS.68K	LO.68K
AS68SYMB.DAT	CPMLIB	MAKESYS.SUB

You may want to run MAKESYS.SUB on M-DRIVE or a hard drive if either are available because it takes a few minutes to run on the floppy.

When you have the correct files on the correct drive, type "MAKESYS" and watch for errors. If no error messages appear during any of the phases of MAKESYS.SUB, you have created a relocatable BIOS file called CPM.REL and are now ready to relocate CPM.REL using RC.SUB. Be aware that since MAKESYS.SUB is a SUBMIT file, an error during the assembly phase will not prevent the running of the linking and erasing phases. If any errors are noted, refer to appendix tables E-2 (pp. 161-171) and E-9 (pp. 187-191) of the DIGITAL RESEARCH CP/M-68K PROGRAMMER'S GUIDE.

When you have successfully assembled the BIOS and created the CPM.REL file, you can relocate it using RC.SUB. RC.SUB is made up of the following two command lines:

1) "reloc -b35000 cpm.rel $1:cpm.sys" which relocates the file CPM.REL with a starting address 35000H, and produces an executable BIOS file called CPM.SYS on the drive of the logical letter invoked on the command line (e.g. typing "RC A" will put CPM.SYS on logical drive A). The use of parameter "-b35000" assumes that the BIOS.S constant HIMEM has not been changed (see BIOS DATA CONSTANTS above) If HIMEM has been changed, the RC.SUB file should be edited to reflect the new value or the correct relocation command sequence should be entered manually, without using the submit file RC.SUB.

2) "era cpm.rel" erases the CPM.REL file created by the MAKESYS.SUB file since it is no longer needed. Again, when using a system with ample disk space, you may want to edit out this line of the submit file.

In order to run RC.SUB, you need the following files resident on the currently logged disk drive:

RC.SUB	RELOC.68K	CPM.REL

In addition you should have a write-enabled floppy diskette ready

for the destination disk drive. (You can send the output file to a hard disk logical drive for speed and ease, but at this date, CP/M-68K cannot load the BIOS from the hard disk at boot time.)

When you have the appropriate files ready, type "RC Y", where "Y" is the logical letter of the destination drive. This completes the regeneration of a non-standard BIOS.

Obviously, the process executed by RC.SUB is not a complicated one and you may choose simply to use the file as a guide for typing out the actual command lines yourself.

REASSEMBLING A MODIFIED LOADER

Once you have made the desired changes to the LBIOS.S loader file, you must assemble both BOOT.S and LBIOS.S, and link them together with the library of CP/M routines supplied by Digital Research (the loader does not need to be relocated). To make this easy, a SUBMIT file called MAKELDR.SUB is provided on distribution diskette #1 which includes the correct command lines for successful assembly and linking of the loader files. SUB is comprised of the following five command lines:

1) "as boot.s" assembles the source file BOOT.S and produces the object code file BOOT.O.

2) "as -l lBIOS.S" assembles the source file LBIOS.S, assures that all addresses are longwords, and produces the object code file LBIOS.O.

3) "lo -s -t0 -uldr -o cpmldr.sys boot.o ldrlib lBIOS.O" links LBIOS.O, BOOT.O, and the loader library files; strips the symbol table and relocation bits to save space; specifies a starting address of 0H, and produces an executable loader file called CPMLDR.SYS.

4) "era boot.o" erases the assembled but unlinked file BOOT.O. (Some users may consider this line unnecessary, and may choose to edit it out of the file.)

5) "era lBIOS.O" erases the assembled but unlinked file LBIOS.O. (This line, too, may be considered unnecessary.)

To use MAKELDR.SUB, prepare a write-enabled diskette with the following files on it:

BIOS.S	BOOT.S	LDRLIB
AS.68K	LO.68K	AS68SYMB.DAT
MAKELDR.SUB		

Log on to the prepared diskette, type "MAKELDR", and watch the terminal for error messages. Be careful: as with MAKESYS.SUB, an error in one of the early assembly phases of the MAKELDR series of commands will probably not prevent the submit

program from continuing with the linking and erasing phases. So watch carefully for error messages throughout the whole process.

If the MAKELDR.SUB file finishes running without any errors, then an executable loader file named CPMLDR.SYS will be generated on the currently logged drive.

This loader can be placed on the system tracks of a floppy diskette using the program PUTBOOT.68K. Just PIP the new loader and the file PUTBOOT.68K onto the diskette which will contain the loader. Then log on to the prepared disk and type

```
PUTBOOT CPMLDR.SYS Y
```

where "Y" is the logical letter of the currently logged disk.

The diskette now only needs a CPM.SYS file and it will boot.

AUTOMATIC EXECUTION OF A COMMAND LINE AT TIME OF BOOT

It is sometimes necessary to set up a BIOS that will automatically execute a CP/M command line just after the sign-on at boot time. The easiest way to accomplish this is to patch the desired command line into the appropriate area of the relocatable system file CPM.REL.

The patch can be done using DDT.68K, a debugging program which is explained in detail in section eight (pp. 129-139) of the DIGITAL RESEARCH CP/M-68K PROGRAMMER'S GUIDE. First, enter DDT and load the CPM.REL file using the "r" command. Then replace the 00H byte at address 428H with the byte 01H which indicates that the following bytes are a command line which should be executed at boot time. Finally, place the upper case ASCII bytes which spell the desired command line in the locations following address 428H. Use the byte 00H to delimit, or indicate the end of, the command line.

For example, let's set up a BIOS that will run MFORM M X to format the M-DRIVE at boot time. Log onto a write-enabled disk which contains the files CPM.REL and DDT.68K. Then enter DDT and type in the following commands:

1) "rcpm.rel" which reads the CPM.REL file into a contiguous block of memory.

2) "s148" which enters into the byte substitution mode at address 148H. Place the following bytes at these addresses:

ADDRESS	BYTE	PURPOSE
428	01	set automatic execute flag
429	4D	ASCII M
42A	46	ASCII F
42B	4F	ASCII O
42C	52	ASCII R
42D	4D	ASCII M
42E	20	ASCII SPACE
42F	4D	ASCII M
430	20	ASCII SPACE
431	58	ASCII X
432	00	DELIMITER

3) "." which leaves the substitute mode.

4) "wcpm.rel which writes the patched CPM.REL file back to disk.

Now the patch is accomplished and you can leave DDT by pressing control-C. The resulting CPM.REL file is ready to be relocated and used as the CPM.SYS file.

REFERENCE MATERIAL

We hope that this document helps you in the alteration and recreation of your operating system. CompuPro is interested in any suggestions you might have concerning this information. If you find any errors or have any comments or ideas for inclusion, please feel free to write to us. Address all correspondence to:

Kevin Fischer
Vice President of Customer Assurance
CompuPro
3506 Breakwater Court
Hayward, CA 94545

For further information about CP/M-68K you may refer to these manuals:

COMPUPRO CP/M-68K OPERATING SYSTEM TECHNIAL MANUAL
DIGITAL RESEARCH OPERATING SYSTEM USER'S GUIDE
PROGRAMMER'S GUIDE
SYSTEM GUIDE

You may wish to consult any of the CompuPro Hardware technical manuals for information specific to any individual board.