```
*****************************
*                           *
*     MAP 80 SYSTEMS LTD    *
*         BIOS NOTES        *
*       VERSION 4.00        *
*         CP/M PLUS         *
*                           *
*****************************
```

MAP 80 SYSTEMS LTD
Unit 2
Stoneylands Road
EGHAM
Surrey. TW20 9QR
Tel 0784 37674
Issue 4 February 1985

CP/M PLUS - THE MAP 80 IMPLEMENTATION

CP/M PLUS has been supplied to you in two parts, a SYSTEM disk which you use to boot CP/M PLUS and a MASTER disk containing the utility files supplied by DRI.

Details of the files provided by DRI can be found in the CP/M PLUS manuals, which, on the whole, will be found instructive and a great improvement over the CP/M 2.2 manuals. We will add a few extra explanations when explaining our implementation in the hopes of clarifying what is a necessarily complex situation.

HARDWARE REQUIREMENTS

For systems other than MAP CPU the following mods are required

G811 with VFC

Link the G811 to power on reset at 0000H (see G811 manual)
Link the VFC to auto boot (see VFC manual)

G813 with VFC

Remove RP/M 2 from the G813 and replace it with BOOT 813 supplied.
Link the VFC to non-auto boot (see VFC manual)

NASCOM 2 with VFC

Remove header plug LKS1.
Set the DIL switches 1234 on LSW1 so that your NASCOM power on resets at 0000H
Link the VFC to auto boot
Remove the header from LKS1

MEMORY REQUIREMENTS

CP/M PLUS is supplied by DRI in two forms, one for an unbanked system (maximum 64k of memory), and one which operates under a system where more than 64k can be made available. The sophisticated features that CP/M PLUS is capable of requires a lot of memory, and we have therefore only implemented the banked system. However with just one additional bank of 64k the full facilities of a banked CP/M PLUS can be taken advantage of. A Z80 can address 64k of memory using 16 address lines, a system is thus required to access more than 64k. At present 3 systems exist on the NAS/80 BUS, one is a 64k paging system utilised by Gemini, this is quite unusable because CP/M PLUS requires portions of memory to be common (accessible at all times). The second is a 32k paging system available on the MAP RAM card when used with the G811 or Nascom CPU cards, this is very suitable for CP/M PLUS but even better is the 4k memory mapping system which the MAP RAM card also supports and is available when using a MAP CPU or a G813. You will therefore require at least one MAP RAM card and total system memory of at least 128k, the RAM on board the MAP CPU or G813 counts towards this.

BOOTING

To boot CP/M PLUS power up your system and then insert your SYSTEM CP/M PLUS diskette in your first floppy drive. A BOOT routine of some sort loads track 0 sector 0 from your master disk. This routine may be the BOOT EPROM on the MAP CPU, or VSOFT on the MAP VFC or varied EPROMs on other systems. The source file 3LDR.MAC, when assembled (3LDR.COM), provides this sector. When 3LDR.COM has been loaded, it is executed and proceeds to load sectors from track 0 sector 1 into memory starting at address 100H. The system loaded is a mini CP/M system (CPMLDR.COM) made up of a mini BDOS, provided by DRI (CPMLDR.REL), and a mini loader BIOS, (assembled from LDRBIOS.MAC). CPMLDR.COM then proceeds to load the main CP/M PLUS system from the first floppy disk, this file resides on the booting disk and is called CPM3.SYS. After CPM3.SYS has been loaded, it is executed, and finally you are into CP/M PLUS. Booting is complete when the CP/M PLUS BIOS loads the console command processor file (CCP.COM) from the last 4k of the system tracks of the SYSTEM disk. CCP.COM is stored in memory, and then loaded and executed to enable command entry. You will be greeted with the CP/M prompt A)

BOOTING ERRORS

If you insert a diskette which has not had the system tracks initialised with a valid system the message SYSTEM? will appear. Insert a properly initialised diskette and booting will then take place (It may be necessary to RESET your machine).

If the boot routine is unable to read a sector from the system track (most likely due to physical damage to the diskette) then the message ERROR will be displayed. Replace the offender with a properly initialised diskette and press RESET.

ON SCREEN EDITING

The normal cursor when the BIOS is calling for console input is a blinking underline character, when console input is not being called for the cursor remains visible but is non-blinking. The MAP BIOS provides an additional feature i.e on screen editing.

To enter the edit mode press EDIT (cntl) @ or (cntl shift @ Nascom keyboards) to obtain an entry of 00H. This informs the BIOS to enter edit mode 1. In this mode the cursor will change to a non-blinking block key entries will be echoed to the screen but will not be returned to CP/M. The cursor control and screen edit keys may be used at will. When CR is pressed the entire current cursor line, with some exceptions, is returned to CP/M. Exceptions are:

1) Trailing blank spaces.
2) CP/M utility prompts *,£,-,: and . when they occur in the first column
3) The CP/M prompt A) when) occurs in the second, third or fourth columns.

After returning the line the BIOS returns to non-edit mode, unless whilst in edit mode 1 the edit key is pressed a second time. Then the cursor will change to a blinking block and the BIOS will keep returning to the edit mode until the edit key is pressed again. Note that the entire line is returned which means that the edit mode may be unsuitable within some programs. It should also be noted that some editing features are not available on NASCOM 1 keyboards.

DISK ERROR MESSAGES

The BIOS traps and displays disk errors which may occur whilst attempting to read or write a sector. This error could occur, for instance, by inserting a disk the wrong way, attempting to write to disk with a write protect tab fitted, forgetting to close the drive door, forgetting to insert a disk or because of disk damage. Having displayed the error you will be prompted for retry, if the problem is a write protect tab fitted or non-inserted disk etc. you may cure the problem and continue, by pressing Y, without the BDOS or user program knowing anything untoward has happened, if the problem is more serious e.g a damaged disk press N and you will be returned to the BDOS which will process the error and pass it to the user program as necessary.

POSSIBLE ERROR MESSAGES

Not Ready

Occurs if a drive door is open or there is no disk, or an unformatted disk.

Disk Write Protected

Occurs when an attempt is made to write to a diskette which has a write protect tab fitted.

Write Fault

This is a fault signal provided by a drive. Drives supplied by MAP 80 do not provide this signal so this fault should never occur.

Record not found

This fault occurs if the FDC has been unable to find a sector's address header or unable to find a sector's data marker. Retries are unlikely to cure this error. If it should occur copy as much information as possible off the offending diskette and reformat it. If the error persists the diskette will most likely have been subject to physical damage and is best discarded (or returned to your supplier if new).

CRC Error

This error occurs when a sector has been found and loaded but a checksum shows that it is corrupt. Again retries are unlikely to resolve this but, as it is at least possible to load the sector, it may be possible to edit it with for example DDT to recover the data.

DEVICE INPUT/OUTPUT

CP/M PLUS exercises IO via CONSOLE (in and out), AUXILLARY (in and out), and LIST (out). These IO systems can be assigned none, one or more of up to 16 devices.

CONSOLE

This is the main interface with you, and as supplied the BIOS obtains input from your keyboard and outputs to your CRT. Normally these will be the only devices assigned for console and the BIOS is optimised for this as the CP/M PLUS console handling is disappointingly slow.

AUXILLARY

The serial device is assigned to auxillary IO in the supplied BIOS. (SIO channel A on MAP CPU).

LIST

The Centronics output via the PIO is assigned to LIST in the supplied BIOS.

The initial assignments can be altered by BIOS re-assembly or by using the DEVICE command, if necessary this can be done at cold boot time using the PROFILE.SUB cold start auto-run facility, see CP/M PLUS User Guide.

CENTRONICS

A parallel printer routine is provided via the PIO device option. Connect the Centronics data inputs of your printer to the B0 thru' B7 outputs of your PIO, connect the Centronics "BUSY" signal to A0 of your PIO and the Centronics "STROBE" to A1 of the PIO.

SERIAL (MAP CPU)

Serial output is handled via the Z80 SIO clocked by the Z80 CTC. SIO channel A is clocked by CTC channel 0 which is clocked at 2MHz. SIO channel B is clocked by CTC channel 1 which is clocked at 1Mhz, as supplied channel A is set to 1200 baud with handshake on CTS and DCD, and channel B is set to 300 baud with no handshake. Each channel is set for 8 bits, 1 stop, no parity. Baud rate and handshake can be dynamically altered using the DEVICE command, note here that enabling XON-XOFF enables hand shake via CTS and DCD. With the combination set up as above SIO channel B can be used at baud rates 75 to 4800 inclusive and SIO channel A at rates 134.5 to 9600 inclusive, additionally each channel can be used at 19200 baud, but this uses the the SIO in *1 mode and is only suitable for transmission.

SERIAL (NASCOM,G811,G813)

Serial output is handled via the 8250 UART on the G811 or G813 or via the 6402 UART on the NASCOM. As supplied the G811/G813 BIOS has handshaking enabled with the 8250 set for 8 bits, one stop, no parity. using the CTS line to provide handshake. NASCOM UARTS do have handshake facilities. The CTS line from the printer should therefore be connected to bit 7 of port 0 if handshake is required. NOTE: IF YOUR PRINTER PROVIDES AN RS232 LEVEL SIGNAL FOR HANDSHAKE (CTS), THEN THIS MUST BE CONVERTED TO TTL LEVELS. Baud rate of the 8250 UART and handshake on either G811, G813 or Nascom can be dynamically altered using the DEVICE command, note here that enabling XON-XOFF enables

hand shake via CTS.

BIOS WORKSPACE

There are certain parts of the BIOS which you may require access to, e.g the interrupt vector table. In order to find out where these useful bits can be found a table of addresses has been left in the BIOS, the address of the start of this table has been placed at a known location i.e the end of the jump table at BIOS+0063H, The addresses within the address table are assigned as follows.

```
;Pointers to useful locations
POINTR:
```
Address of floppy data base, used for assigning formats to drives
```
        DEFW DBASE              ;Disk data tables
```
Number of logical floppies and number of logical Winchesters
```
        DEFW NDRVF*256+NDRVW    ;No of Winchesters + No of floppies
```
Initial assignments for IO, could be changed in CPM3.SYS
```
        DEFW IASSG              ;Initial IO assignment
```
Port initialisation data setting serial baud rate and mode, could be changed in CPM3.SYS
```
        DEFW PORTI              ;Initialisation data sent to serial
        DEFW WRKSPC             ;Data
```
Location of VFC workspace, this workspace is also used by IVC/SVC keyboards for edit mode and programmable keys
```
        IF VFC OR IVC
        DEFW VFCW               ;VFC workspace, used by IVC too
        ELSE
        DEFW 0
        ENDIF
```
Interrupt vectors, patched for user requirements, remember interrupt routines must lie in common memory
```
        IF MAP
        DEFW INTVEC             ;Interrupt vectors
        ELSE
        DEFW 0
        ENDIF

        DEFW DISPT              ;Dispatch tables
```
The word prior to PRGTAB shows length of following table, table format is:-
```
     Character to be programmed,"Returned String",0FFH
```
Table is terminated by a null
```
E.G       DEFB 81H,"THIS IS A FUNCTION KEY",0FFH
          DEFB 82H,"THIS IS ANOTHER FUNCTION KEY",0FFH
          DEFB 0

        DEFW PRGTAB             ;Programmable key table
```
Address of a 128 byte buffer in common memory used by VDISK and MOVE routines, could be used by user for bank transfer
```
        DEFW BUFF80             ;128 byte buffer in common memory
```
Bank info including current bank selected
```
        DEFW CURBNK             ;Bank info
```
BIOS version number in format N.NNxy where N.NN defines version number e.g 4.00, y defines CPU M=MAP, 3=G813, 1=G811 and N=Nascom, y defines

main video/keyboard, V=VFC, I=IVC/SVC, S=Serial (Terminal).

```
        DEFW BVERS              ;4byte BIOS version + CPU + VIDEO
```

Address of space inside BIOS and thus common memory which can be used by a users program so that he may call BIOS inter bank moves for instance but insure the integrity of his stack during bank changes, stack space is at least 80 bytes.

```
        DEFW USRSTK             ;User stack address
Yet to be defined
        DEFW 0
        DEFW 0
        DEFW 0
```

The table will be found in common memory but some of the addresses may reside in banked memory.

INTERRUPTS

The MAP CPU uses interrupts via the CTC for keyboard entry, an interrupt structure has therefore already been set up in the BIOS. Interrupt mode 2 has been used so an interrupt vector table has been set up, at the moment this is very under used so adding more Z80 interrupting devices is simple. The interrupt vector table can be found via the POINTR table above and its format is as follows:-

```
;Interrupt vectors, these must all be on the same 100H page
INTVEC:
;MPI SIO, This must occur on a 10H boundary
MPSIOV: DEFW 0
        DEFW 0
        DEFW 0
        DEFW 0
        DEFW 0
        DEFW 0
        DEFW 0
        DEFW 0
;On board SIO
CPSIOV: DEFW 0
        DEFW 0
        DEFW 0
        DEFW 0
        DEFW 0
        DEFW 0
        DEFW 0
        DEFW 0
;MPI CTC
MPCTCV: DEFW 0
        DEFW 0
        DEFW 0
        DEFW 0
;On board CTC
CPCTCV: DEFW 0
        DEFW 0
        DEFW 0
        DEFW KEYINT             ;Read keyboard
;On board PIO
```

```
CPPIOV: DEFW 0
        DEFW 0
        DEFW 0
        DEFW 0
```

TABLE 2

As can be seen only the CPU's CTC has been allocated, the other vectors are reserved for other MPI and CPU devices but may be used by yourself as you wish, simply place the address of your interrupt service routine into the appropriate place in the table and initialise the Z80 device's interrupt register with the low byte at the start of the appropriate part of the table. Interrupt mode 2 has already been set up as has the I register. Don't forget that you are operating in a banked environment where memory is being mapped in and out, your interrupt routines must therefore lie in common memory and not at a location where the VFC is being paged in. On the MAP CPU the best place for such routines is below 1000H.

With the G813,G811 or Nascom interrupts have not been enabled and no interrupt table exists, users will therefore have to set up their own interrupt structure, with the G813 place the interrupt vectors and interrupt routines below 1000H, with a G811 or Nascom place them above 8000H. Note also that the IVC has a hardware bug and does not handle interrupt acknowledges correctly.

If your interrupts are not for a particular program they can of course be included in the BIOS where they must reside in the resident portion. Furthur information on interrupts can be found in the BIOS modification section.

MEMORY ALLOCATION

The CP/M PLUS system you have been supplied with assumes only 128k of RAM and has been configured in what we consider the most useful way in those circumstances. If you wish to change this configuration this is possible without having to undertake re-assembly of the BIOS, by using the GENCPM.COM utility. This utility takes the raw BDOS supplied by DRI and raw BIOS supplied by MAP 80 and produces a system (CPM3.SYS) configured according data supplied by the user (and stored in GENCPM.DAT). In order to use GENCPM you will require an understanding of how CP/M PLUS allocates memory. A Z80 can only address 64k at any one time, in order to extend memory addressing part of this memory must be switched out and replaced by another bank, only part of the memory must be switched otherwise you will switch out the very program that you are running. CP/M PLUS requires the switched portion to be in the lower part of the 64k map and the non-switched, common part to be in the upper. We tell GENCPM.DAT where the first and last pages (100H) of common memory are. On G811 and Nascom systems the last 2k of common memory is reserved and the MAP VFC screen is left there so as to be accessible to the user. The 32k paging system used on G811 and Nascom forces the bank boundary to be at 8000H, but with the memory mapping system used by the MAP CPU and G813 the bank boundary can be set at any 4k boundary. Following discussions will assume a bank boundary at 8000H as this can be common to both systems. On power up or reset bank 0 is selected in the lower part of memory and common memory in the upper part. The loader BDOS and BIOS are

loaded from floppy at address 100H and is then executed. The file CPM3.SYS is then loaded from floppy, this file contains the BDOS and BIOS of the main CPM system. Part of the BDOS and BIOS are placed by the loader system at the top of the common memory at cold boot, where they can control the switching of memory in the banked portion. The rest is placed in bank 0 at the top of the switchable portion of memory. Control is then passed to the BDOS in common memory, and it switches bank 1 into the lower part of memory. The boot routines also load the CP/M console command processor (CCP.COM) from the system tracks of the first floppy and places it in the bottom of bank 2 from where it can be quickly loaded, after any warm boot, into the bank 1 TPA. The system looks like FIGURE 1

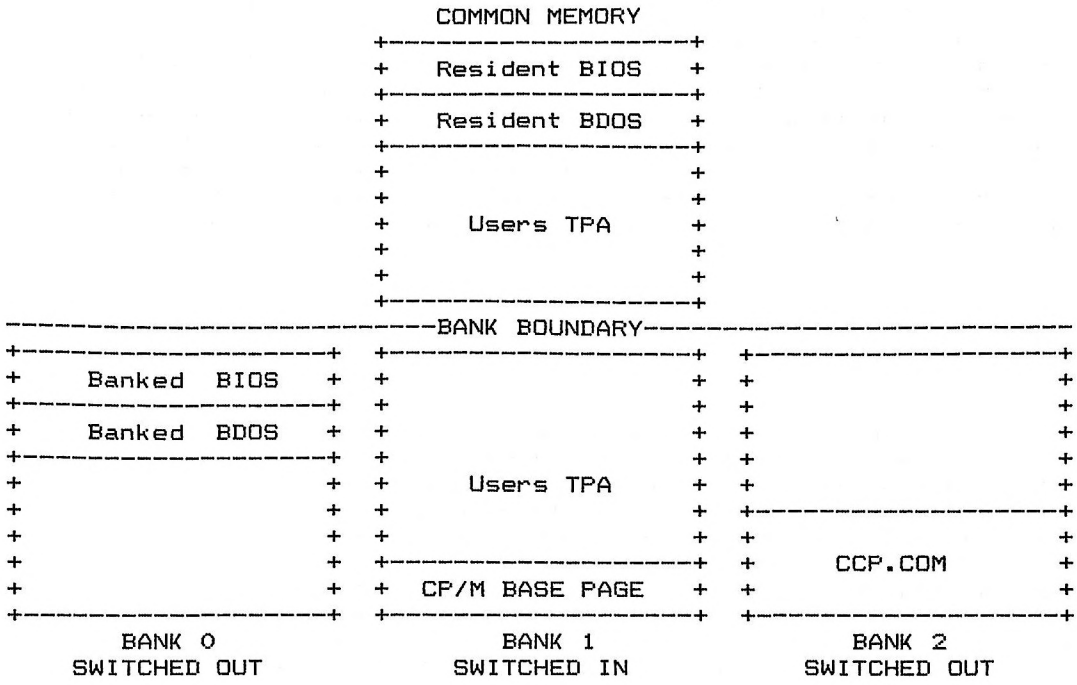
COMMON MEMORY
Resident BIOS
Resident BDOS
Users TPA
BANK BOUNDARY
Banked BIOS
Banked BDOS
Users TPA
CP/M BASE PAGE
CCP.COM
BANK 0
SWITCHED OUT
BANK 1
SWITCHED IN
BANK 2
SWITCHED OUT

FIGURE 1

Standard CP/M compatible programs can be loaded in the normal way, these start at 100H in bank 1 and follow up through the TPA across the bank boundary into common memory, standard BDOS calls can be made via the CP/M base page, these are sent to the resident BDOS in common memory. If necessary the resident BDOS will select bank 0, execute routines in the banked BDOS/BIOS and then reselect bank 1 before returning to the user program.

This takes care of the allocation of program parts of the system but not the data areas. There are 6 kinds of data areas to be allocated:-

ALLOCATION VECTORS (ALVs)
CHECKSUM VECTORS (CKSs)
HASH TABLES
DIRECTORY BUFFERS
DEBLOCKING DATA BUFFERS
BUFFER CONTROL BLOCKS (BCBs)

It is possible to allocate these data areas in the BIOS at assembly time or, as we have done, allocation can be left to GENCPM. By doing this it is possible to alter system configuration without having to resort to BIOS re-assembly. GENCPM places the ALLOCATION and CHECKSUM VECTORS together with the BUFFER CONTROL BLOCKS at the top of the banked BIOS but below the bank boundary, the banked BDOS and BIOS are moved down accordingly. The directory buffers are allocated starting at the base of bank 0 up to a maximum at the base of the BDOS. The HASH TABLES are allocated at the first available slot available in bank 2,3,4 etc. (above the stored CCP). And the DATA BUFFERS allocated above this. Optionally the DATA BUFFERS can be allocated in common memory, when they are placed above the resident BIOS, the resident BDOS and BIOS are moved down accordingly. DATA BUFFER allocation is however limited as each buffer eats into the users TPA. At cold boot the BIOS searches to find how much memory your system contains, it then looks to see how much has been allocated to the data areas above, and memory left unallocated is set up as virtual disk on drive P. We now add one furthur complication. This applies only to the memory mapped CPU boards the MAP CPU and the G813. CP/M PLUS doesn't care if bank 0 and banks 2+ don't start at 0000H, and its very convenient for us if they don't, if we make the bottom 4k slot common memory it becomes somewhere where we can always put a stack and be sure of it's integrity, somewhere we can put interrupt routines and be sure the BDOS won't take them out at an embarrassing moment, the CP/M base page always remains accessible as do the RST addresses and NMI vector. Having made the base 4k common the system looks like FIGURE 2.

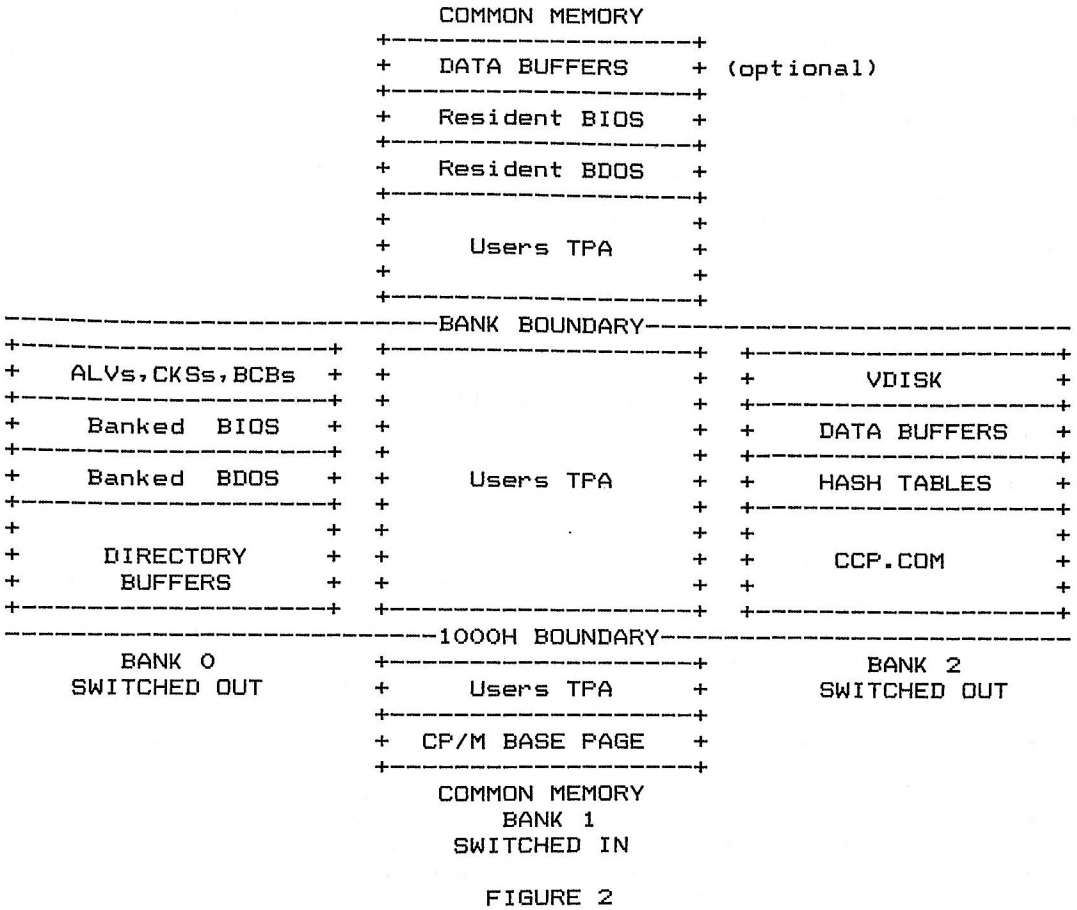
COMMON MEMORY
DATA BUFFERS
(optional)
Resident BIOS
Resident BDOS
Users TPA
BANK BOUNDARY
ALVs,CKSs,BCBs
Banked BIOS
Banked BDOS
DIRECTORY BUFFERS
Users TPA
VDISK
DATA BUFFERS
HASH TABLES
CCP.COM
1000H BOUNDARY
BANK 0 SWITCHED OUT
Users TPA
CP/M BASE PAGE
BANK 2 SWITCHED OUT
COMMON MEMORY BANK 1 SWITCHED IN

FIGURE 2

With a bank boundary of 8000H the system shown above would in fact be made up of RAM as follows:- The base 4k from 0000H to 0FFFH will be the base 4k from 0000H to 0FFFH of your first 64k block of memory, Bank 0 which when paged in occupies 1000H to 7FFFH will come from 1000H to 7FFFH from your first 64k block. Bank 1 which when paged in also occupies 1000H to 7FFFH will come from 0000H to 6FFFH from your second 64k block. Bank 2 which when paged in also occupies 1000H to 7FFFH will come from 7000H to DFFFH from your second 64k block. The common memory above the bank boundary occupying 8000H to FFFFH comes from 8000H to FFFFH of your first 64k block of memory. NOTE this only applies to the MAP CPU and G813, Figure 1 gives the representation for G811 and Nascom systems.

Now to using this information whilst answering GENCPMs questions.

Common memory base page (CO) ?

If you are running a G811 or Nascom system you must answer 80 to this question. With a MAP CPU or G813 the base of common memory can be at any 4k boundary. Selecting the best boundary depends on your system and/or your requirements. Looking at figure 2 we see that bank 0 holds the banked BDOS and BIOS, allocation and checksum vectors, buffer control blocks and directory buffers. Setting a large bank size permits a large amount of buffering of directory sectors for Winchester drives and opens up the possibilities of a large amount of data buffering. (CP/M PLUS can handle banked memory up to 16 times the bank size). However if you are just running a couple floppy disk drives, only 16 buffers are required and the rest of bank 0 will be wasted. If we set the bank size to that required for the directory buffers we limit the total size of memory available to data buffers because we can only have a total of 16 banks. Precise calculation of memory requirments in bank 0 isn't easy. Each directory buffer is the same size as a physical sector on the disk, this is usually 200H but may change according to disk format, VDISK for example has 80H sectors. In addition to the buffer size itself each buffer has a 15 byte BCB allocated, and to add to our problems each data buffer also has a BCB allocated in bank 0 and we haven't specified our data buffer requirements yet. There is a furthur restriction in that the VFC must be paged in over a block of memory occupying its true position in the memory map, we don't want to use 0000H to 0FFFH for the reasons described earlier so we must leave the 4k directly below the base of the resident BIOS as common memory. This normally limits the bank boundary to a maximum of E000H.

Number of memory segments (£2) ?

Enter the number of banks (at least 2). With a G811 or Nascom system you will have (TOTAL MEMORY CAPACITY - 64)/32. With a MAP CPU or G813 system you will have (TOTAL MEMORY CAPACITY - 64)/BANK SIZE. If this calculation leaves a remainder you may add an extra bank and then limit it's size to that of the remainder when specifying the memory segment sizes later.

Enter memory segment tables:

For each bank enter base of banked memory (always 0 for Nascoms or G811s), bank size in 100H pages, and the bank number, GENCPM will trim any over allocation that it knows about. Exceptions to this are,

when allocating bank 2 specify a base of 10H so as to leave space for the stored CCP.COM. When allocating the final bank which is smaller than the bank size, specify a size according to memory available.

Enable hashing for drive X: ?

Directory hashing allocates 4 bytes for each possible directory entry. With the MAP format 128 directory entries are possible the hash table therefore is 200H long.

Number of directory buffers for drive X: ?

Enter the number of sector size buffers required, the MAP format has 8*200H directory sectors but you don't have to specify them all if you don't want to, in fact all drives can share a single buffer if required.

Number of data buffers for drive X:

Enter number of sector sized buffers required, remember that a 15 byte BCB will be generated in bank 0.

It is possible to reserve memory for your own purposes in one of two ways. Firstly by limiting the size of a memory segment when specifying a bank size, it is suggested you start from above the stored CCP in bank 2. In order that the BIOS search of memory allocation can pick this up you must select at least one drive with directory hashing enabled or have allocated at least one data area outside of common. The other way is to reserve a specific number of 4k VDISK tracks by setting the VRTRK equate in SYSTEM.MAC, VDISK starts at the base of bank 1, you must therefore reserve at least sufficient tracks to cover bank 1 and the saved CCP, additional tracks must be reserved to cover hash and data buffers and finally sufficient for your own requirements. This will of course require full re-assembly of the BIOS.

Don't be surprised if you find that you have to run GENCPM several times before you get what you want, also keep a check on whats going on as GENCPM can be a little unforgiving and generate a system incapable of operation. To run GENCPM you will require GENCPM.COM, BNKBDOS3.SPR, RESBDOS3.SPR, BNKBIOS3.SPR and optionally GENCPM.DAT.

MODIFYING THE BIOS

The source files of the BIOS are provided to enable customisation of your system. All MAC files are for assembly using the Microsoft 8080/Z80 assembler M80. Simple modifications can be made by changing EQUates in the equate files DATA.MAC and SYSTEM.MAC and then re-assembling and linking, these can be made with only limited knowledge. More dynamic changes to the BIOS MAC files will require intimate knowledge of the CPM3 system guide and proficiency in using M80. The time/date routines have been kept completely separate from the other MAC files so that you may add your own RTC routines even if you do not have M80, write your own routine using RMAC the entry point label is ?TIME and must declared as public, use LINK to link BASE3.REL and SCBK.REL provided together with your new TIME.REL and create a new BNKBIOS3.SPR. If your RTC clock interface requires initialisation, use a flag in TIME.MAC to show first time entry, if the flag is 0 initialise the interface and then set flag non-zero before executing the actual time routine.

The BIOS supplied only uses interrupts for keyboard entry on a MAP CPU, this is a relatively slow interrupt rate and causes no problems to disk read and write operations, and interrupts are not disabled. If a rapid interrupt rate is envisaged, around 40ms, it will be necessary to disable interrupts during actual sector read/write. If interrupt handling routines are short and fast, as they should be, disable only once the FDC has called for data transfer, in this case interrupts will stop for about 15ms. If necessary guaranteed interrupt free disk access can be obtained by disabling before the FDC command is issued. Interrupts will be disabled for up to 230ms but typically 15-50ms depending on sector skew.

```
;Read/write floppy
RWFLOP: LD A,(DMABNK)           ;Select DMA bank
        CALL BANK
        LD A,(RWFLAG)
        OR A                    ;Read or write ?
        DI                      ;DISABLE HERE FOR GUARANTEED ACCESS
        JR Z,WRTOP
        LD A,(COMM)             ;Get command
        OUT (FDCCOM),A          ;Send command
        JR RDTEST

DATIN:  IN A,(FDCDAT)           ;Read byte
        LD (HL),A               ;Store buffer byte
        INC HL
RDTEST: IN B,(C)                ;Get flags
        JR Z,RDTEST             ;Nothings happening
        DI                      ;DISABLE HERE FOR MINIMAL DISABLE
        JP M,DATIN              ;DRQ
        JR RWDONE               ;Finished, either IRQ or motor off

WRTOP:  CALL INVERT             ;Invert the data if necessary
        LD A,(COMM)             ;Get command
        OUT (FDCCOM),A          ;Send command
DATOUT: LD A,(HL)               ;Get buffer byte
        INC HL
```

```
WRTEST: IN B,(C)                ;Get flags
        JR Z,WRTEST             ;Nothings happening
        OUT (FDCDAT),A          ;Send byte
        DI                      ;DISABLE HERE FOR MINIMAL DISABLE
        JP M,DATOUT             ;DRQ
        DEC HL                  ;Back to end of buffer
        LD A,2
        CALL DELAY              ;Delay 2 ms after write
;Reselect page 0
RWDONE: EI                      ;RE-ENABLE INTERRUPTS
        CALL INVERT             ;Invert the data if necessary
        XOR A
        CALL BANK
        RET
```

Note that in the DI instruction for minimal disable is inside the data transfer loop this will make the timing very tight for 8" double density, and may not work.

The SYSTEM disk contains the CP/M PLUS system configured for your machine together with all source code MAP 80 files are :-

3LDR.MAC
The MAC file for assembly by M80 of the Track 0 Sector 0 cold boot routine.

LDRBIOS.MAC
The MAC file of the mini BIOS used by the CPM3 loader BDOS to load and execute CPM3.SYS.

CPMLDR.SYS
The combined programs produced by 3LDR.MAC,CPMLDR.REL,LDRBIOS.MAC and CCP.COM. This is used to initialise the system tracks of a disk from which you wish to cold boot. To use it type COPYSYS CPMLDR.SYS(CR) see CPM3 manuals for furthur details.

BASE3.MAC
The Base MAC file of the CPM3 BIOS, this file INCLUDEs DATA.MAC, SYSTEM.MAC, LOW.MAC, MAP4.MAC, HIDEV.MAC, HIDISK.MAC, DSKSTRUC.MAC and LODISK.MAC.

BASE3.REL
Assembled REL file of BASE3.MAC

TIME.MAC
Customise this file for your specific RTC.

TIME.REL
Assembled REL file of TIME.MAC

DATA.MAC
BIOS MAC file, containing data equates

SYSTEM.MAC
BIOS MAC file containing equates which specify system configuration.

CHRIO.MAC
Character IO routines.

MAP4.MAC
Memory management routines for bank,move and virtual disk. This file is for memory mapping on the MAP CPU or G813, use MAP32.MAC for G811 or Nascom.

3BOOT.MAC
Warm and Cold Boot routines.

HIDISK.MAC
High level disk IO

DSKSTRUC.MAC
Disk structure generation (DPBs,DPHs etc)

FLOPPY.MAC
Low level floppy disk IO

SCBK.REL
The assembled file SCBK.ASM, this is used when LINKing the 3BIOS REL file.

BNKBIOS3.SPR
The assembled and linked BIOS. With this file it is possible to make system generation changes using GENCPM.COM

GENCPM.DAT
The data file used by GENCPM.COM when generating CPM3.SYS from BNKBDOS3.SPR, RESBDOS3.SPR and BNKBIOS3.SPR.

CPM3.SYS
The CPM3 system file produced by GENCPM.COM. This file must reside on the disk from which you cold boot.

MU.COM
The MAP 80 multi purpose disk format/copy/verify utility.

MU.MAC
Source of MU.COM

COPYSYS.COM
The DRI utility for system disk generation as modified for your system.

COPYSYS.MAC
Source of COPYSYS.COM, modified from DRIs COPYSYS.ASM

CONV2.RSX
RSX to be attached to command programs which make direct BIOS calls for disk access under CP/M 2.2

In order to assemble a new system proceed as follows:-

```
    M80 =BASE3
```
This assembles BASE3.MAC to produce BASE3.REL.

```
    LINK BNKBIOS3[B]=BASE3,TIME,SCBK
```
This links BASE3.REL, TIME.REL and SCBK.REL to produce BNKBIOS3.SPR.

Now use GENCPM.COM to create a new customised CPM3.SYS.

ASSEMBLING FOR DIFFERENT DRIVES/FORMATS

The disk routines are very flexible and permit a wide range of Winchesters, 5", 8", and several 3" drives to be used together with many disk formats. First level of system configuration is done in SYSTEM.MAC, modifications here can be done without having to insert any new data.
First select the CPU you are operating

```
MAP     EQU T                   ;Set true for MAP CPU
;Select CPU board
G811    EQU F AND NOT MAP
G813    EQU F AND NOT MAP
NAS     EQU F AND NOT MAP
```

Then your video card, select both F if you are using a Terminal, note that if you wish to use an IVC or SVC with a MAP CPU you will have to obtain a custom PROM from MAP 80.

```
VFC     EQU T                   ;True for VFC
IVC     EQU F                   ;True for IVC/SVC
```

If you are using a MPI as your floppy controller set ALT T for VFC at it's alternate location.

```
ALT     EQU T                   ;VFC port at alternate location
```

If you are using an MPI and wish to be able to read a format having a bad side 2 flag in the ID address headers on side 2 you must modify your MPI as follows:- Remove the link L14 connecting a to m and solder a wire between L14 a and L13 n. Note that once this modification is done you can only use BIOS version 4.00 or later.

```
MPILNK  EQU F AND ALT           ;Set only if MPI is linked
                                ;..for side change
```

Now tell the system about your floppy drives, see DATA.MAC for available drives, FPPY1 is the first physical floppy FPPY2 the second etc.

```
FPPY1   DEFL STNDRD
FPPY2   DEFL STNDRD
FPPY3   DEFL PERTEC
FPPY4   DEFL DR7200
```

Next assign CP/M logical drives (A,B,C,D...) to a floppy and a format, FASG1 and FORM1 refer to A: FASG2 and FORM2 refer to B: etc. FASGx is assignment to a floppy, 1 to FPPY1, 2 to FPPY2 etc. FORMx is assignment of a format, see DATA.MAC for available formats. Entering a 0 in either FASGx or FORMx will terminate drive assignment, it is best

to assign no more drives than you really need. The NULLF format is unusable as itself but is a special format designed to allocate maximum space to cope with any format and is used by ASSIGN.COM for dynamic format allocation. FASG1 must be 1 and FORM1 must be MAP96D or MAP48.

```
FASG1   DEFL 1                  ;Floppy assignment
FORM1   DEFL MAP96D             ;Format

FASG2   DEFL 2
FORM2   DEFL MAP96D
FASG3   DEFL 2
FORM3   DEFL NULLF
FASG4   DEFL 2
FORM4   DEFL NULLF
FASG5   DEFL 2
FORM5   DEFL NULLF
FASG6   DEFL 2
FORM6   DEFL NULLF
FASG7   DEFL 2
FORM7   DEFL NULLF
FASG8   DEFL 0
FORM8   DEFL 0
```

If you are running a MAP CPU, MKBD must be T unless you are running an IVC/SVC in which case it may be set false and the IVC/SVC keyboard can be used. When using the MAP CPU keyboard it is possible to stop a running program and warm boot at any time. The key which actions this is set by the BREAK equate, A1H is CTRL/SHIFT F1 on a MAP keyboard. It is possible to use a 7 bit keyboard which returns ESC xx for function keys on the MAP CPU keyboard port by setting ESCKEY T, the codes returned from the function keys are set in the programmable key table in CHRIO.MAC

```
MKBD    EQU T AND MAP AND (VFC OR IVC)  ;Set true for MAP keyboard
BREAK   EQU 0A1H                ;Break key for CPU keyboard
ESCKEY  EQU F AND MKBD          ;T for 7 bit ESC keyboard with MKBD
;Select keyboard with Nascom
NKBD    EQU T AND NAS AND (VFC OR IVC)  ;T for Nascom with Nascom keyb
```

Note when using an IVC with a MAP CPU keyboard the IVC must be modified to overcome a bug on the IVC. Invert M1 (bus line 25) by connecting it to IC15 pin 3, lift out pin 14 of the decode PROM VID 1 IC35 and connect the floating pin to the inverted M1 IC15 pin 4.

Setting either DRVW or NDRVW to 0 will cancel Winchester assembly, else select Winchester in DRVW and the number of logical CP/M drives you wish it split into in NDRVW, see DATA.MAC for available Winchesters and SYSTEM.MAC for furthur Winchester options

```
;Select Winchester type and logical drives it is split into
DRVW    DEFL 0                  ;Winchester type
NDRVW   DEFL 0                  ;Drive allocation to Winchester
```

As supplied a MAP CPU has CTC0 clocked at 2Mhz to provide baud rate clocks for SIO channel A and CTC1 clocked at 1Mhz to provide baud rate clocks for SIO channel B, if you change these links it must be reflected here.

```
SIOCHA  EQU FAST                ;SIO A CTC0 clock, slow=1Mhz fast=2Mhz
SIOCHB  EQU SLOW                ;SIO B CTC1 clock, slow=1Mhz fast=2Mhz

Power up baud rates are assigned below
IMODEA  EQU MBIO+MBSERL+MBSOFT+MBXNXF   ;Initial IO mode
IMODEB  EQU MBIO+MBSERL+MBSOFT          ;Initial IO mode
IBAUDA  EQU B9600               ;Initial baud rate
IBAUDB  EQU B300                ;Initial baud rate

And initial device assignment next
CONSIN  EQU 8000H               ;Console input from keyboard
CONSOT  EQU 4000H               ;Console output to video
LSTOUT  EQU 1000H               ;List to SIO-A
AXIN    EQU 0800H               ;Auxillary in from SIO-B
AXOUT   EQU 0800H               ;Auxillary out to SIO-B

If    DATA.MAC    does    not    contain    data    for    your    particular
Winchester/Floppy/Format, these can be inserted using existing data as
a model. To insert a Winchester:-
Name the Winchester
JIMW    EQU x
Enter Winchester data
;JIM's Winchester
CYLx    EQU 320                 ;Cylinders
RWCx    EQU 132                 ;Reduced write cylinder
HDSx    EQU 2                   ;Heads
WPCx    EQU 0                   ;Write pre-comp cylinder

To insert a Floppy:-
Name the Floppy
JIMF    EQU x
Enter Floppy data
;JIM's Floppy
STPx    EQU 3                   ;Step rate
HDLx    EQU 10                  ;Head load timing
STLx    EQU 20                  ;Track settling time
DTYx    EQU TP96                ;96tpi drive

To insert a format
Name the format
JIMFM   EQU x
;JIM's Format
        FN DEFL MAP96D
;                   _FTY,_DEN,_SEC,__BLK,DAL,TPS,OFF,SPS
        FDAT1 %FN,TP96,DDEN, 512, 4096,  1, 80,  2, 10
;                  UAL,EXM,___CGS,CGT, SKW,SOF,INV,BSF,SCY,DDM
        FDAT2 %FN,  0,  F,TRKCRY,  0,   0,  F,  F,  F,  F,  F

Where     FTY = Format Type (TP96,TP48,TP8)
          DEN = Density (SDEN,DDEN)
          SEC = Physical Sector Size (128,256,512,1024)
          BLK = CP/M Allocation Size (1024,2048,4096,8192,16384)
          DAL = Directory Allocation Blocks
          TPS = Physical Tracks per side
          OFF = Logical System Tracks
```

```
SPS = Physical Sectors per side
UAL = User reserved allocation blocks (usually 0)
EXM = T=Zeroed EXM, F=Standard CP/M 2.2 / 3.1 EXM
CGS = Side Handling (TRKCRY,TRKREV,SSID,TRKCYL,SECCYL)
CGT = Track to change side with TRKCRY (usually set to 0)
SKW = Logical sector skew, or SPSK for user defined skew
SOF = T=Sectors start at 1, F=Sectors start at 0
INV = T=Data is inverted, F=Data is not inverted
BSF = T=Side 2 has bad side flag in side 2 ID header
      F=Good side flag, See note on MPILNK above
SCY = T=Physical sector numbers on side 2 continue from
      side 1 (i.e side 1 sectors 0 to 9 on side 2 10 to19)
      F=Physical sector numbers on side 2 start from 0/1
DDM = T=Sector data is preceeded by deleted data mark
      F=Standard data mark (usual)
```

Note that if you wish to use the MAP 80 ASSIGN program the format number x that you assign to a format should correspond with the number issued in FORMATS.DAT

Having inserted your new requirements use M80, LINK and GENCPM to produce a new system. If you have specified a new drive for DRV1 it may well be necessary to produce a new loader system.

MODIFYING THE LOADER BIOS

```
M80 =3LDR
L80 3LDR,3LDR/N/E
```
To produce 3LDR.COM, track 0 sector 0 cold loader

```
M80 =LDRBIOS
```
To produce LDRBIOS.REL

```
LINK CPMLDR[L100]=CPMLDR,LDRBIOS
```
To produce CPMLDR.COM, note this can be run directly from CPM2.

Then to produce CPMLDR.SYS

```
SID 3LDR.COM
RCPMLDR.COM 200
RCCP.COM 1800
WCPMLDR.SYS 100 2900
GO
```

THE MAP 80 MULTI-UTILITY PROGRAM

On your SYSTEM DISK you will find a file called MU.COM. This program enables you to FORMAT, COPY and VERIFY a diskette. Run the program by entering:-

MU<CR>

You will be greeted by the command prompt. Enter one, two or all the commands in any order according to your requirements..F C V. Commands will be executed in the order FORMAT COPY VERIFY regardless of the order in which they are entered. Having selected the commands

you require press CR to continue.

If you selected F you will now be prompted for a sector skew. To optimise disk access time sectors are laid down on the disk in a staggered sequence. For a general purpose disk use a skew of 2. A CR is not required.

If you selected C you will be prompted for the drive which will hold the disk which is to be copied. A CR is not required.
You will then be prompted for the drive or drives to be FORMATTED and/or COPIED TO and/or VERIFIED. Enter the drive/drives followed by CR.

You will then be prompted to insert the appropriate disks into the appropriate drives.

Where a single drive system is being used it is possible to copy a disk in a drive to another disk in the same drive . In this case the copy routine will halt and prompt you when it is necessary to change disks. It would be advisable to put a write protect tab on the disk to be copied from in order to avoid possible mix ups.

Whenever you are being prompted for keyboard entry (blinking cursor) you may exit MU by pressing (cntl) C.

BUGS AND PATCHES

It is possible to have the date printed in English format, i.e DD/MM/YY by ammending the following files:-

DATE.COM	change	0951 to 97
		0958 to 96
DIR.COM	change	3529 to CB
		3530 to CA
SHOW.COM	change	0CEF to FE
		0CF6 to FD

PIP doesn't handle password protected files correctly, if you PIP to replace a password protected file the transfer appears to happen normally but in fact the protected file is not removed and the PIPped file remains in it's temporary .$$$ state. At least this is a failure on the safe side.

DATE [CONTINUOUS] only works if you have an interrupting RTC as DATE simply reads the SCBK and does not call the BIOS's ?TIME routine. Users of the MAP RTC should use our CLOCK.COM instead. The MAP BIOS does not support DATE.COM's clock setting routine, MAP RTC users should use CLKSET.COM.

SAVE doesn't permit maximum sized file name entry, i.e B:12345678.123 will have the file type "3" truncated.

DRI insist that you should use BDOS function 50 to call the BIOS, however if you write a customised user function via BIOS function 30 be warned that HL,DE,BC will be corrupted on return if a BDOS 50 call is used, better to call the BIOS directly using address 0001 and do any necessary banking in the BIOS routine.

If you start using banks in one of your programs, be warned any BDOS

call will return with bank 1 selected, contact MAP 80 should you run into problems as we an RSX to get round this.

INCOMPATIBILITY WITH CPM 2.2

CP/M PLUS, despite its far greater sophistication and many modifications, retains remarkable compatibility with CP/M 2.2, 99% of CP/M programs work without any trouble, those that do are the ones that are a little "naughty" in that they make direct BIOS calls for disk IO, this shouldn't really be done even under CP/M 2.2 but there are many fine programs of a utilitarian nature which do these "naughties" and which we would prefer not to lose. Using these programs under CP/M 3 however causes disaster, the disk routines are in banked memory and therefore not directly accessable, furthur more the CP/M 3 BIOS returns physical sector sized chunks of data rather than 128 byte records. Enclosed on your system disk is a little bit of magic to overcome all your problems (famous last words!!!), in the form of CONV2.RSX. This is an RSX which can be attached to any .COM file, which uses direct BIOS access for disk IO, and makes it look like CP/M 2.2, to attach it, for instance, to the old copy of DU from the CPM user group enter GENCOM CONV2 DU. The new DU.COM generated will work a treat.