MEMOTECH

MFX SYSTEM

OPERATOR'S MANUAL

This page is intentionally blank

MEMOTECH

MFX SYSTEM

OPERATOR'S

MANUAL

Manual Revision : 1.14

FPGA Firmware Build : 01-01

ROM Firmware Build : 167

2nd February 2024

Credits :

Martin Allcorn / Dave Stevenson	: Co-designers, CFX, CFX-II, MFX
Bill Brendling	: 80 Column Card emulation (CFX-II), FTPD, HTTPD
Andy Key	: SDX ROM and enhancements
Memotech	: For the MTX series

Table of Contents

1 Introduction

MFX is a Multi-Function Expansion card for the Memotech MTX range of computers, designed to provide removable storage for Memotech owners who do not have one of Memotech's disk drive add-ons or a modern day alternative, such as CFX or REMEMOrizer.

MFX is a development of the CFX and CFX-II products. The CFX name was derived from the removable media used, ***Compact Flash***. For MFX, the removable storage media is SD or micro SD card and additional features have been added, resulting in a change of name to MFX to reflect the multiple functions available with the MFX product.

Features

- CP/M "disc" system using either SD or micro SD card
- RAM expansion up to a total of 512kB
- VGA output for CP/M (80 columns) and VDP (40/80 columns and graphics)
- Network connectivity supporting HTTP and FTP
- Designed for internal mounting

From a MTX perspective, MFX is fitted with a system extension ROM which allows access to the SD storage card in 2 ways :

- It provides a CP/M 2.2 environment based on Memotech's FDX/SDX

and

- It can provide extensions to the MTX's built in BASIC via USER command as used by the SDX system based on Andy Key's re-creation of the source code for the SDX ROM.

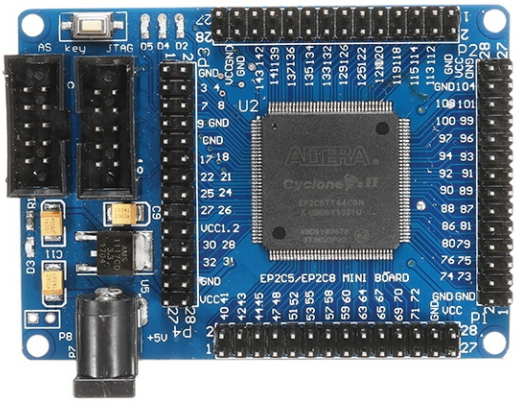
Other enhancements over the first version of CFX/CFX-II include :

- Optional 320kB RAM Disc
- Optional 32kB RAM expansion (to allow MTX500 computers to run CP/M)
- Compatible with the original MTX NMOS Z80 CPU (CFX-II required a CMOS CPU)
- Full emulation of the MTX VDP with output to a VGA monitor

In a change from the design philosophy of CFX and CFX-II, MFX has been designed to support internal mounting only, connecting via the internal MTX edge connector (J0). In another change, focus moved from minimising the PCB footprint to better integration of the I/O connectors with the MTX case; the PCB size increased markedly to allow all I/O connections to be made through a custom end plate on the right hand side of the MTX.

1.1 Architecture

The heart of MFX is a minimum system development board which utilises an Altera Cyclone II EP2C5 Field Programmable Gate Array (FPGA).



This FPGA is available only as a Surface Mount component in a TQFP-144 package, making its direct use somewhat difficult for the average hobbyist. Mounted on a carrier PCB, this credit card sized development board provides a convenient way to integrate the FPGA with the Memotech MTX. The majority of the FPGA's I/O pins are exposed on the 4 x 28 pin headers which can be mated with corresponding sockets on a carrier PCB. The FPGA supply voltage is 1.2VDC with a maximum voltage for the I/O pins of 3.3VDC. The development board supply is 5VDC and on board regulators deliver the required 1.2VDC and 3.3VDC voltages. To interface with the MTX's 5VDC TTL level voltages, the carrier PCB must perform level shifting between the FPGA and MTX's voltage levels.

1.2 Credits

Andy Key's awesome REMEMOrizer deserves a notable mention at this point. REMEMOrizer was the first SD card expansion available for the Memotech MTX, with extensive development work having been done by Andy to clone MTX functionality using a low cost FPGA development board incorporating a Xilinx XC3S500E Spartan 3E FPGA. The GODIL development board that Andy used now appears to be out of production and Andy has no plans to make any more REMEMOrizers, so an alternative SD card solution is needed. REMEMOrizer was itself a development of Andy's REMEMOTECH, the Re implementation of a Memotech computer entirely in an FPGA development board - the Altera DE1, built around an Altera Cyclone II 2C20 FPGA.

The firmware in MFX draws heavily on the code published by Andy and thanks are due to him for releasing his work into the public domain. However, it should be noted that our intention was not to recreate all of the functions of REMEMOrizer, for example, the Speculator emulation and Virtual Tape support is not included, but some other unique features have been added.

Credit is also due to Andy and Bill Brendling for developing the TCP/IP code that allows the MTX computer to use a commercially available network module – the WIZnet WIZ810SMJ. Andy wrote

the low level code to interface the WIZnet module first used with the “proof of concept” Memotech NFX. Bill enhanced the code to support HTTP and FTP.

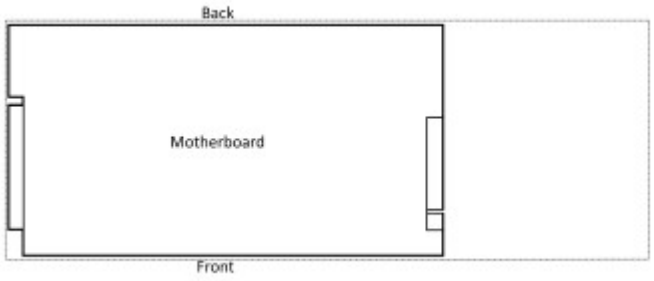
The sections of this manual that describe the use of CP/M draws heavily from the original Memotech FDX manual, the text has been edited to reflect the changes implemented for MFX. The contents of this section in particular are pitched at users with little or no knowledge of CP/M are likely overly simplistic for the experienced MTX user. However, this manual is intended to provide a stand-alone guide to the most often used features MFX without requiring the user to refer to additional Memotech or Digital Research documentation. For more advanced topics, such as programming, please refer to the original Memotech documentation – the FDX Operator’s Guide is the most relevant to this subject.

The most significant difference between Memotech’s disc systems and MFX is the user of SD card media to replace the floppy disc drives. References to “disc”, “disc drive”, etc. should be understood as referring to the MFX single SD card, divided into logical “discs” (partitions) as described in Chapter 3.

2. Installation

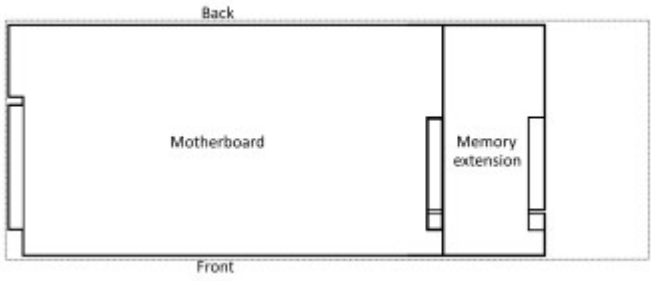
MFX has been designed to connect to the internal card edge connector (J0) on the Memtoech MTX computer board.

This sketch, reproduced from the FDX manual, although not dimensioned, is pretty much to scale; the dotted outline shows the size of the MTX case in relation to the motherboard. The area to the right of the motherboard was intended for Memotech's expansion cards - initially, RAM, ROM and the RS-232/FDX Interface boards.



Each of these boards was half the width of the expansion card area, i.e., up to two expansion cards could be installed internally. A combined 80-column/RS-232 board, taking up the full width of the expansion card area was released later but is not relevant to this discussion, since it was meant to support Memotech's own disk based CP/M add-on using a disk interface connected to the cartridge port on the left hand side of the case. If you have one of those, you probably don't need MFX.

The most common expansion card seen is the RAM extension card, due, in no small part, to Memotech's common practice of selling an "MTX512" that was made up of an MTX500 motherboard with an additional RAM board fitted with the extra 32kB of memory.



I expect that this was due to Memotech having a surplus of MTX500 boards when the MTX512 was released. This wasn't normally a problem unless the user wanted to install more than 1 additional board - at least until the 80 column board was released when MTX owners could get a nasty shock when opening up the case to try and fit it!

It may not be clear from the sketch, but the RAM (and ROM) boards had an edge connector plug on the left and the right hand side had a pass through board edge connector to allow a second card to be plugged into it.

To allow one of my expansion cards (CFX, CFX-II or MAGROM) to be installed internally by as many MTX owners are possible, the cards needed to be no wider than one of Memotech's expansion boards, i.e., ~80mm.

For MFX, the PCB takes up the whole of the available space between the edge of the MTX computer board and the right hand end plate. This creates a potential issue for prospective MFX users whose MTXs already have an internal expansion card fitted, either by accident (Memotech installed), or design (user installed) – any Memotech expansion boards must be removed to create sufficient space to install MFX.

For the majority of prospective users, this will not be a problem as summarised in the following table

Memotech Board	Impact / Mitigation
RAM Expansion	None – additional RAM, up to 512kB, can be provided by MFX
NewWord ROM	Low – the tape based program can be replaced with the CP/M version
Pascal ROM	Low – the tape based program can be replaced with the CP/M version
RS-232 Interface	Medium – no space for card, use network functionality of MFX
FDX Interface	High - FDX is incompatible with MFX in the same way as SDX was
MTX 80 Column Card	High – MFX is incompatible with the MTX 80 Column Card

The only material impact is for users who use the functionality of the RS232/FDX interface board. However, it is suggested that the loss of RS232 functionality is mitigated by MFX's TCP/IP networking capabilities.

(If an RS232 interface board has been fitted to the MTX, as well as removing the board, the 25 way "D" connectors must be removed from the plastic panel at the rear of the case as the RS232-1 connector would clash with the WIZnet module.)

MFX is compatible with an externally fitted MAGROM.

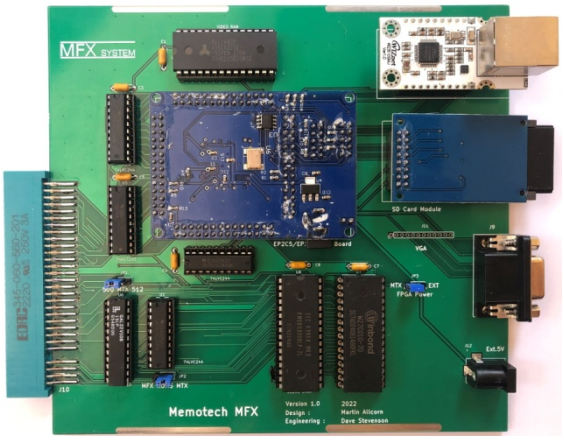
2.1 Preparation

Check that you have all of the parts you should have received :

MFX PCB

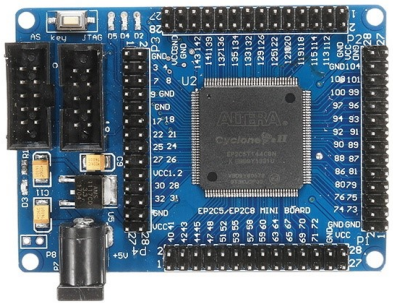
Pre-assembled MFX PCB with the FPGA, WIZnet and SD card modules installed.

(Connection to a VGA monitor is normally made using the VGA connector on the PCB, alternatively, an optional VGA breakout cable can be connected to the blank header.)



FPGA Development Board
(Usually pre-fitted to PCB)

If the FPGA module has been supplied unfitted, fit it to the PCB, ensuring that the header pins are correctly aligned with sockets J1, J2, J3, J4 and J8 on the PCB.

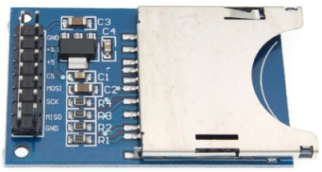


WIZ810SMJ Network Adapter
(Usually pre-fitted to PCB)

If the WIZnet module has been supplied unfitted, fit it to the PCB, ensuring that the header pins are correctly aligned with sockets J7 and J8 on the PCB.

SD Card Module
(Usually pre-fitted to PCB)

Either a full size or micro SD (TF) card slot



Full Size Slot

Micro SD (TF) Slot

SD Card

or

Micro SD card & adapter

Before installing the MFX PCB, check that the option selection jumpers on the MFX PCB match the configuration of your machine and the options that you wish to use.

2.1.1 MTX Selection Jumper (JP1)

The default configuration for MFX allows it to be used with either a MTX500 or a MTX512. Jumper JP1 is used to select the appropriate computer and works with the MFX expansion RAM to extend the available RAM for either machine to a total of 512kB.

MFX should also be compatible with the Memotech RS128 and MTX512S2 but will require the memory selection GAL (chip U6 on the PCB) to be programmed differently. Please contact the author to request support for using MFX in machines other than a MTX500 or MTX512.

2.1.2 ROMS Selection Jumper (JP2)

MFX can make up to 512kB of RAM available to the MTX. Since CP/M can only use 64kB memory for programs, MFX provides the option to use 320kB of the extra memory as a RAM disk in the same way that REMEMOrizer does. However, the MTX Operating System (OS) ROM performs a test of all of the available memory at power on or after a reset. This means that the contents of the RAM disk are not normally preserved after a reset which prevents a soft reboot of the MTX from booting from RAM disk.

For users who want the flexibility of having the MTX boot from RAM disk, it is possible to remove the MTX OS ROMs from their sockets and have MFX provide a patched version of the ROMs which does not corrupt the RAM disk on a reset. It is recognised that not all uses will want to mess with the MTX computer board in this way, so this is optional.

For users who do not want to remove the original OS ROMs, the RAM disk functionality is still available (see section 8.12), but it will not be bootable. In this case, ensure that the MFX “ROMS” jumper is set to “M***T***X”.

For users who want the ability to reboot from RAM disk, ensure that the MTX OS ROMs have been removed and set the MFX “ROMS” jumper JP2 is set to “M***F***X”. (Since the RAM disk will be empty at power on, the initial boot of CP/M will always be from the SD card.)

There is an added complication for some users who have a foreign language keyboard fitted to their MTX. Memotech installed a language ROM, piggy-backed onto one of the standard ROMS to support some international keyboards, including Danish, Finnish and Norwegian. Whilst the standard ROM and the piggy-backed ROM soldered to it can be easily removed from the socket, the language ROM is also hard wired to the GROM line on the MTX computer board. This patch wire must be removed in order to allow the ROMs to be removed. If MFX is supplying the ROM images, the MFX ROM must

also include the contents of the language ROM. Images are available for the Danish and Finnish language ROM; the required language must be specified when ordering MFX.

With these added complications, it is likely that users who have a language ROM installed will forgo the minor benefit of being to reboot CP/M from the RAM disk to simplify the installation of MFX.

2.1.3 FPGA Power Jumper (JP3)

Most MTX users will find that the 5v power available from the MTX's power supply is capable of proving sufficient power for the additional load imposed by MFX. However, during testing, it was observed that one machine suffered from somewhat degraded performance of the MTX Video Display Processor when MFX was connected, though the 80 column CP/M and emulated VDP displays on the VGA monitor were unaffected.

MFX has the facility to allow the user to connect an external 5v DC (5.5 x 2.1mm, centre positive) Power Supply to provide additional power. Jumper JP3 is used to select between powering the FPGA module from the MTX's 5v power line (jumper position "MTX") or from an external PSU connected to the barrel jack connector J12 (jumper position "EXT").

Since most users are likely to prefer to use the emulated VDP output through to the VGA monitor, a slightly degraded VDP output may be acceptable should it occur.

2.2 Opening the MTX

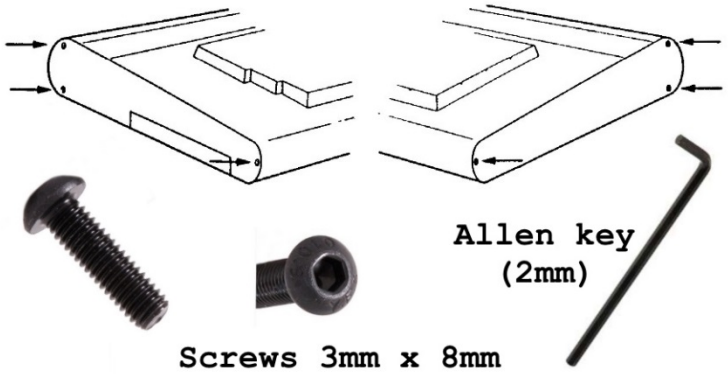
***** Turn OFF the MTX *****

The MTX and/or the MFX card may be damaged if the MFX PCB is installed or removed with power applied

The two halves of the MTX case are secured by six, 3mm socket head machine screws, three through each end plate.
The front edges of the two halves of the case have interlocking profiles that allow the keyboard to be swung upwards like a hinge.

Using a 2mm Allen key, remove the three screws from the right and left hand sides of the MTX.

Set the left hand end plate aside for later refitting. The right hand end plate will only be required should MFX be removed at some point in the future and can be put away until that time.

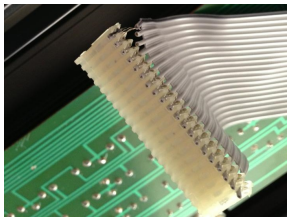


Lift the MTX keyboard at the rear, just above the plastic panel, taking care not to put strain on the keyboard interconnecting cable

This photo shows a ribbon cable attached to the MTX computer board and to the left hand side of the keyboard.

This cable is not the original MTX one, the Memotech cable is shorter and you will not be able to raise the keyboard to the same extent as in this photo without disconnecting the cable first.

The photo shows the type of cable damage that can result if care is not taken in opening the case or disconnecting the keyboard cable.



After the cable has been disconnected from the computer board, the keyboard is released from the base by sliding it completely to the left or right, leaving the MTX in two halves as shown here.

The right hand side of the computer board is very close to the AV connectors and has an overhanging capacitor as shown.

Make sure that you have decided whether you will use the ROMs on the MTX computer board, or have MFX provide their contents. Refer back to section 2.1.2, configure the ROMS jumper accordingly and remove the MTX computer board ROMs if you made that choice.

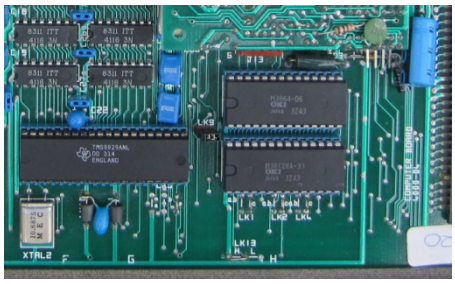
NB : The OS ROM must only be made available by either the MTX or MFX – not both. Prior to powering on the MTX for the first time after you have installed MFX, please double check that the MTX OS ROM has been removed or that the “ROMS” jumper is in the “MTX” position.

MTX 4000-04 computer boards have a 16 kB combined “OS” and “BASIC” ROM and an 8kB “ASSEM” ROM. MTX 4000-05/06 computer boards have individual 8kB ROMs for the “OS”, “BASIC” and “ASSEM” ROMs.

MTX 4000-04 Computer Board

8kB (OKI M3864-06) in the upper socket, and
16kB (OKI M38128-A in the lower socket

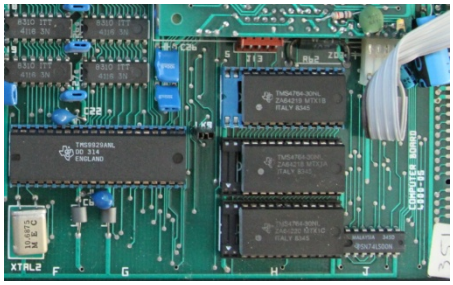
Remove both ROMs



MTX 4000-05/06 Computer Board

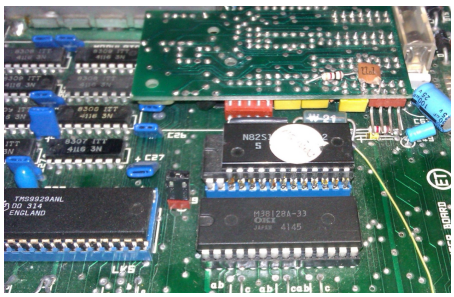
3 x 8kB (TMS 4764) marked MTX1B, MTX1A and MTX1C

Remove all three ROMs



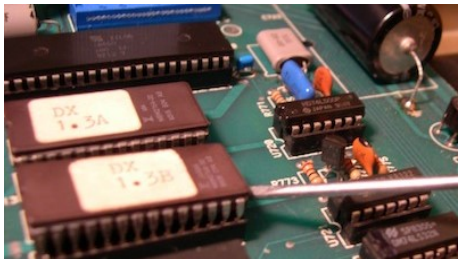
MTX4000-04 With Language ROM

If the MTX has a foreign language ROM and the OS ROMs are to be removed, the yellow GROM patch wire must also be removed.
NB: ensure that the MFX has been supplied with a ROM that includes the appropriate language ROM contents for your region before removing the original ROMs.



The “proper” way to remove socketed ICs such as the ROMs, is to use an IC extraction tool such as the ones shown here, most readers are unlikely to own one, but . . . . .

Alternatively, the IC can be removed with the aid of a small, flat blade, screwdriver. Carefully insert the tip of the screwdriver between the base of the chip and the socket and gently prise up each end in turn. Take care not to insert the tip under the socket and lift the socket off the computer board instead!



If you chose to remove the original ROMs, be sure to store them safely, you will need them again should you want to remove MFX at some point in the future – not that you ever will of course !

Proceed to fitting the PCB.

2.3 Fitting the PCB

The FPGA development board will normally have been fitted to the MFX PCB before shipping. If it has not been installed, mount it to the MFX PCB before installing the PCB in your MTX.

It is possible to install the FPGA board after the PCB has been fitted, but the number and size of the pin header strips on the FPGA/PCB means that the force required to insert the header plugs into the sockets risks putting unneccessary strain on the PCB.

The WIZnet and SD Card Slot modules will also normally have been fitted to the MFX PCB before shipping, but if supplied loose, they can be installed either before or after fitting the PCB into the case.

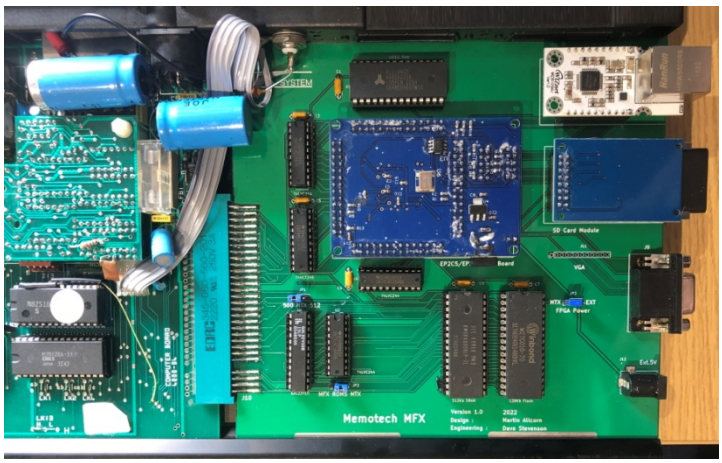
(It may be convenient to remove these modules from the MFX PCB before fitting it to ensure that you avoid any risk of damage to them when you are fitting the PCB.)

Memotech used two different methods of fixing rubber feet to the bottom of the keyboard shell. Most often, the feet are merely self adhesive pads fixed to the bottom of the case. However, in some cases, the rubber feet are inserted into holes drilled through the keyboard base. MFX can be installed in both case types, but fitting is definitely easier if the self adhesive pads have been used.

If the rubber feet are inserted into holes in the base, the one nearest the front right side may foul the MFX edge connector but you should be able to manoever the PCB over it. Alternatively, you may wish to remove that rubber foot until the PCB has been installed, then reinsert it.

The MFX edge connector has a “key” installed in position 5. Make sure that the key is in place and the MFX PCB is located in the slots in the keyboard shell, then, carefully attach the MFX board to the MTX computer board edge connector.

Take care not to damage the AV ribbon cable that connects to the rear panel, it is very fragile!



You may need to slightly reposition the capacitor that overhangs the computer board PCB.

You should check that MFX is working properly before refitting the keyboard and closing up the case.

Please refer to the Quick Start Guide in the next section to confirm that MFX is working properly.

If you do not see the MFX Boot Screen, power off the system and check that the PCB has been correctly mated with the edge connector and that the SD Reader and Card are installed correctly.

If you still do not see the MFX Boot Screen, please contact your supplier.

2.4 Quick Start Guide

This Operator's Guide contains everything that you should need to know to the get the most out of your MFX but the price for that is that it is somewhat "wordy". You are likely keen to fire up the system and get started, so hopefully, a few pointers here will allow you to do that without having to wade through the hundred or so pages in this document. However, should you have any problems or general questions, ***please take a few minutes to scan through the rest of the manual before you ask for help***!

2.4.1 Connect up the system

Connector	To / From	Required	Notes
MTX Power	MTX PSU	Yes	
MTX Video	Composite Monitor/TV	Optional	
MTX Audio	Monitor/TV Audio	Optional	
MFX VGA	VGA Monitor	Yes	
MFX Network	The world !	Optional	
MFX Power	5VDC PSU	Optional	See 2.1.3

In the first instance, having the MTX connected to the Memotech PSU and MFX connected to a VGA monitor should be sufficient to confirm that the system is working as expected.

Insert the SD card – note, the card should be inserted upside down

2.4.2 Power on the system

Though not essential, having the MTX audio output connected to a suitable TV/Monitor will allow you to quickly determine whether the system has booted correctly. With MFX running, the first thing that happens is that a 'Welcome' tune is sounded and the MFX opening menu is displayed on the VGA and VDP connected monitor(s).

If the 'Welcome' tones are absent and there is no output on the VGA or Composite Video monitor, switch off the MTX and re-check the installation of MFX.

Possible issues are :-

- The PCB is not correctly mated with the MTX edge connector
- The sub-modules (FPGA, WIZnet and SD) boards are not mounted correctly
- IC's may have worked loose during shipping

Having checked these points, if you still do not see the MFX Boot Screen, please contact your supplier.

On successful start-up, a boot screen is displayed on the monitor(s) connected to the MTX AV/TV outputs and/or the MFX VGA output, which displays the various boot options available :

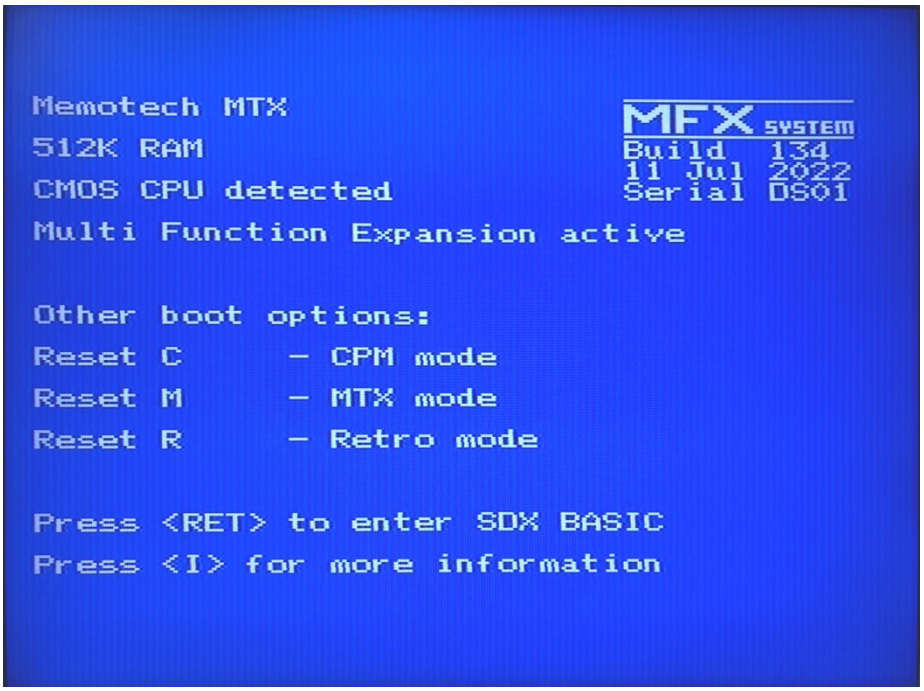


Figure 1 - MTX Video Output Boot Screen

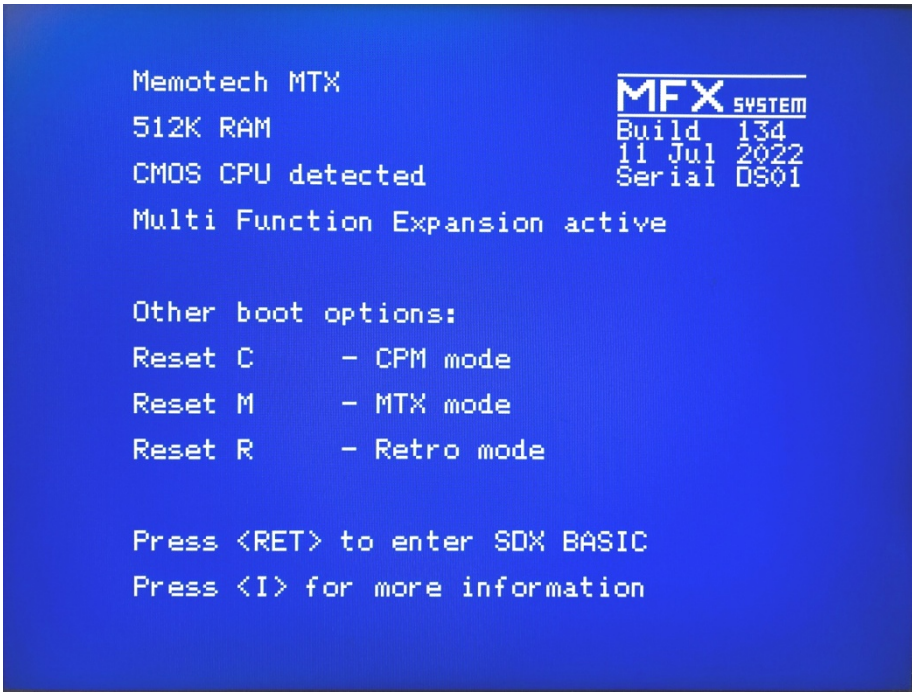


Figure 2 - VGA Output Boot Screen

2.4.3 Reassemble the MTX

Once you have established that MFX has been detected by the computer and the boot screen has been correctly displayed, power off the MTX and refit the keyboard.

The round profile of the upper shell/keyboard should be aligned with the mating channel in the lower half of the shell, and the keyboard slid back into position. Reconnect the keyboard interface cable, taking care not to damage the cable and that the connector pins and cable are correctly aligned.

Before refitting the keyboard end plates, you may want to confirm that the SD card and WIZNET module are operating as expected (see 2.4.4 and 2.4.5). Once you are satisfied, you can refit the original left hand endplate and either the optional 3D printed right hand endplate that may have been supplied with MFX, or another of your choosing. If a replacement end plate is not available, the system may be operated without one, but care should be taken when inserting/removing the SD card as the SD card slot is intended to be supported by the right hand end plate.

Holding down the relevant key when the system is powered on or reset will enter the associated mode as shown. Pressing <**RET**> when the boot screen is displayed will enter SDX BASIC mode, or pressing <**I**> will display a crib sheet showing SDX BASIC USER disk commands

Figure 3 MTX Video Output Crib Screen

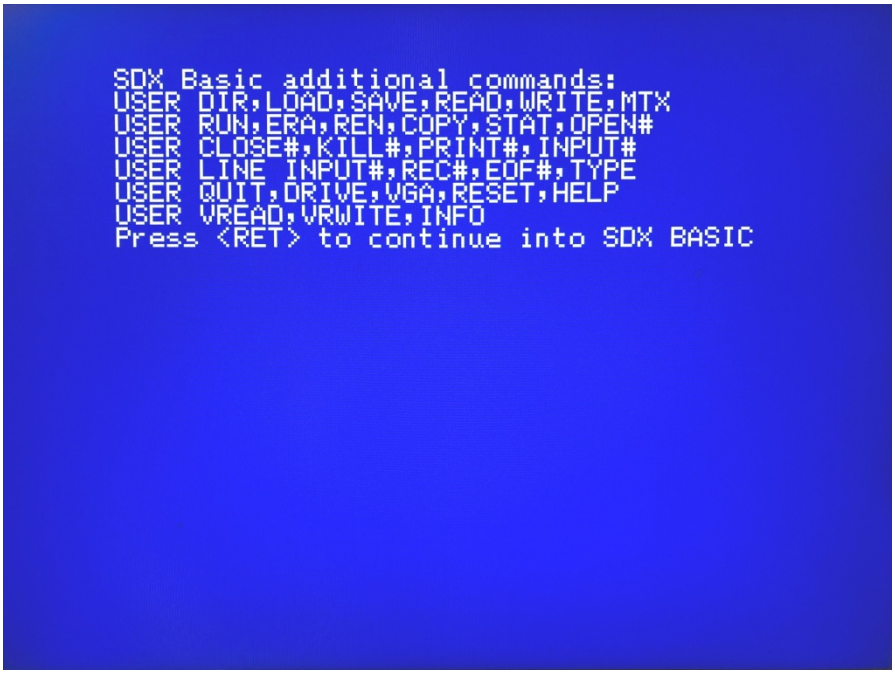


Figure 4 VGA Output Crib Screen

Display of the Boot and Info screens demonstrates that the system has successfully booted from the MFX ROM and you should be able to confirm that the system has correctly recognised the total of 512kB of RAM that's now available to your MTX. A full description of the available start-up options is available in Chapter 16 , but for now, we'll just load a couple of games . . . . . .

2.4.4 Start MFX – SDX Mode

From now on, we'll assume that you are connected to a VGA monitor.

Press <**RET**> to start MFX in SDX Mode

From the 'Ready' prompt, enter '**USER HELP**' to see the commands available from the SDX ROM. You should see the same Help screen as is available from the Boot screen.

Note: All of the USER commands, including **USER HELP**, call the same entry point into the MFX ROM and are all reliant on the SD card being available. If the SD card is not present or otherwise unreadable, the system will respond with a '**DISC ERROR**' message.

.MTX files are binary copies of MTX tape files, typically produced for use in emulators. Multi-part .MTX files rely on tape functions in the MTX ROM to load multiple parts of the program, since MFX does not use the ROM tape routines, multi-part MTX files will not run on MFX.

The .MTX files on the SD card distributed with MFX are on the "D" drive, so, make that drive the default, from the 'Ready' prompt, enter by '**USER DRIVE "D:"**'

From the 'Ready' prompt, enter '**USER DIR "*.MTX"**' to list the .MTX files on the SD Card

OK, let's pick a known good example . . .

From the 'Ready' prompt, enter '**USER MTX "ZARCOS.MTX"**' to load "*Escape from Zarcos*"

You should see the screen border flash magenta as the SD card is accessed.

After a few seconds, the game will load, the annoying Zarcos music will start and you can play your first game courtesy of MFX!

Once you're done with the game, reset the MTX and hit <Return> to start MFX in SDX Mode again and you can try some of the other loadable files:

'USER DIR "*.BAS"' and 'USER DIR "*.RUN"' will list the other files executable from SDX mode

.BAS files are files LOADed and SAVEed from MTX BASIC; run using '**USER LOAD "name.BAS"**'

.RUN files are programs, typically games, intended for use with SDX BASIC; run using '**USER RUN "name.RUN"**'

From the 'Ready' prompt, enter '**USER LOAD "ZARCOS.BAS"**' to load the same game as the one you loaded from its .MTX file.

A more detailed description of the SDX USER commands is given in Chapter 16

2.4.5 Start MFX – CP/M Mode

As shown on the Boot menu, '**Reset C'** is used to start MFX in CP/M mode, i.e., Reset the MTX by simultaneously pressing the two unmarked key while also holding down the '**C**' key. Release the '**C**' key after one or both of the Reset keys and the system will start CP/M.

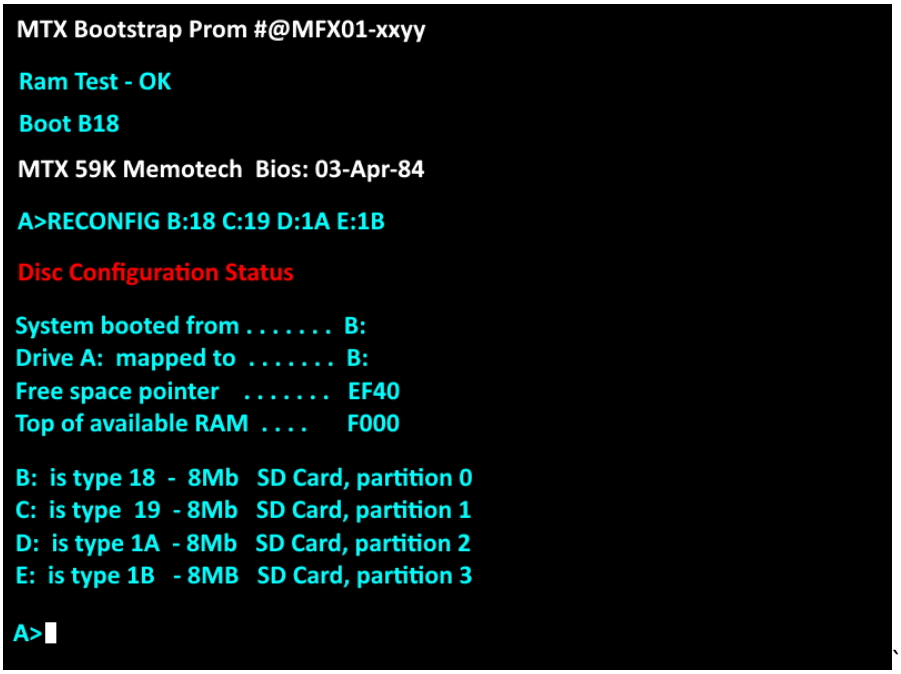


Figure 5 - MFX CP/M Boot Screen

The system has booted from CP/M and you have 4 'discs' immediately available for use, each 'disc' is an 8Mb partition on the SD card.

The '**A>**' is the CP/M command prompt and the flashing cursor is waiting for your input

Type '**DIR'** to see the files present on the currently logged 'disc' (A).

If you are unfamiliar with CP/M, now's the time to delve further into the detail of this User's Guide, starting with Chapter 3.

OK, you should now be up and running with MFX

Chapters 3 to 15 provide detailed information on the use of CP/M with MFX. The majority of users are likely to need minimal interaction with CP/M, those users might like to jump straight to Chapter 16 which covers the SDX aspects of MFX.

3 Getting Started With MFX CP/M

If you have set up your MTX-MFX according to the instructions in the previous section and booted into CP/M (<**Reset**><**C**>), the following sign on message should be on the screen.

```
MTX Bootstrap Prom #@MFX01-xxyy

Ram Test - OK

Boot B18
```

The “xxyy” portion of the sign on message is a serial number which will usually include two initials from the owner’s name and a unique two digit serial number.

If you do not see the above message carefully follow the instructions in the set-up section once more.

If the sign on message does appear and the SD card distributed with MFX is installed in the SD card slot, the system should boot CP/M

```
MTX 59K Memotech Bios:03-Apr-84

A> RECONFIG B:18 C:19 D:1A E:1B

Disc Configuration Status

System booted from ..... B:
Drive A: Mapped to ..... B:
Free Space pointer ..... EF40
Top of available RAM .... F000

B: is type 18 - 8Mb  SD Card, partition 0
C: is type 19 - 8Mb  SD Card, partition 1
D: is type 1A - 8Mb  SD Card, partition 2
E: is type 1B - 8Mb  SD Card, partition 3

A>
```

The screen should now appear as shown above. CP/M has been loaded or 'booted' into your computer.
The ‘**A>**‘ is called the '**prompt**' sign, and the flashing square is the cursor. The prompt shows that the computer is now ready to accept information typed at the keyboard. The cursor indicates the position at which the next character will be typed.
The '**A>**' will be replaced by a '**B>**' when drive B is accessed, and similarly '**C>**' will indicate that drive C is being accessed, etc. Before you ask the question, this does not mean you have five disc drives.
The MTX-MFX has one physical disc drive, or rather, one physical SD card slot. This card slot can emulate up to four physical drives, labelled B to E. Drive A is a logical drive which can be set to drive B, C, D or E or to drive F (a RAM disc as discussed later).

For the time being please accept that drive A and drive B are the same drive. A more thorough explanation of the drives can be found in the section **'MTX special CP/M utilities**.'

If the MFX SD card is inserted into the SD card slot and the system is asked to boot into CP/M (<**Reset**><**C**>), the system will automatically load CP/M into your computer's memory. Not all of the disc contents will be loaded into memory, only that part of the disc immediately required by CP/M will be loaded. Utility programs will only be loaded when called by the user.

CP/M enters your computer's memory in a few seconds and starts automatically. To tell you all is well your console displays the sign on message **'MTX CP/M 2.2** ..... 'etc. already shown above.

The first line is the sign on, CP/M's way of saying hello. The following lines show the Disc Configuration Status. The **Free Space Pointer** defines the highest address used by CP/M. The **Top Of Available Memory** Pointer indicates the lowest byte required for certain essential routines which are loaded above CP/M in high memory.

Points to note when using your SD card:

1 The SD card is a very convenient storage medium and will remain reliable as long as it is treated with care.

A Memory cards are precision devices. Do not drop the memory card or subject it to vibration. Physical shock or vibration may destroy the images recorded on the card.
B Do not spill any liquids on the memory card.
C Do not bend the card or subject it to any excessive force or physical shock.
D Do not touch the contact of a SD memory card with your fingers or any metal objects.
E Do not store or use a memory card near anything having a strong magnetic field such as a TV set, speakers, magnets, or in places prone to having static electricity. Such environments may destroy the images recorded on the card.
F Do not leave memory cards in direct sunlight or near a heat source. Heat can deform the cards and render them unusable.
G To protect the recorded image data, always store the memory card in a case.
H Do not store memory cards in hot, dusty, or humid locations.

2 Insertion. Insert the SD Card with the label facing downwards.

3 Write protect. There is a Lock switch on the left side of the SD card. Make sure the Lock switch is slid up (unlock position) towards the front of the card. You will not be able to modify or delete the contents on the memory card if it is locked. (This statement only applies when the SD card is used in an SD card slot external to MFX, e.g., on a PC. The SD card slot and device driver in MFX are not aware of the SD card write protect switch.)

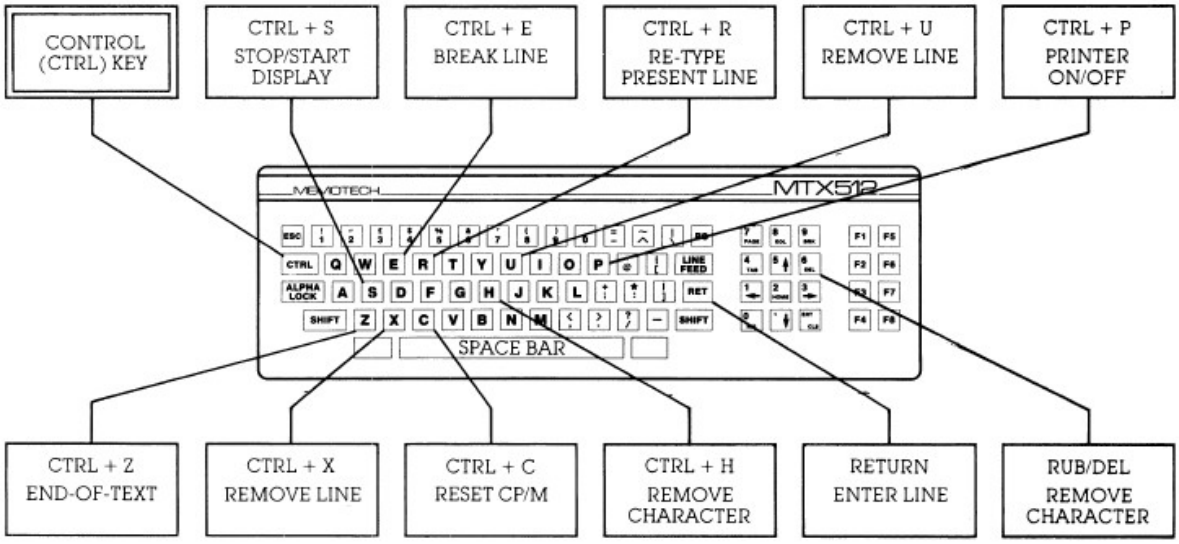
4 Back ups. Whilst the SD Card is much more robust than a floppy disk, it is still a relatively vulnerable recording medium; it is important to make copies or 'backups' of the SD Card to protect you from loss or damage. If regular backups are not made loss or damage to discs may mean the repetition of hours of work. This document describes a process for backup up logical discs (partitions) to other partitions on the SD card. To protect against physical damage to the SD card or other media errors, the SD card should be backed up externally using an imaging utility as described in APPENDIX A.

Now that the MTX-MFX has been set up correctly it is time to start getting to know your system. The disc operating system or DOS used by the MTX-MFX is CP/M. Many computers use CP/M, and therefore, there are many programs available to run on your computer. The following sections will guide you through CP/M, and will educate you in the use of your MTX-MFX.

You normally tell CP/M what to do by typing a command at the keyboard. Now don't get nervous - you can't do any damage by making a mistake. The most you can do is confuse CP/M, and then it will repeat your command followed by a question mark.

You may type using upper or lower case characters, CP/M automatically converts each to upper case for its own use. As you type, the letters and numbers you enter are shown on the display. If you make a mistake, merely press the back space or delete key. If you want to remove the whole line that you have just typed hold down the control key and press X.

Pressing control along with other keys will provide many other useful functions, they are shown here, but you will see how they are used later.



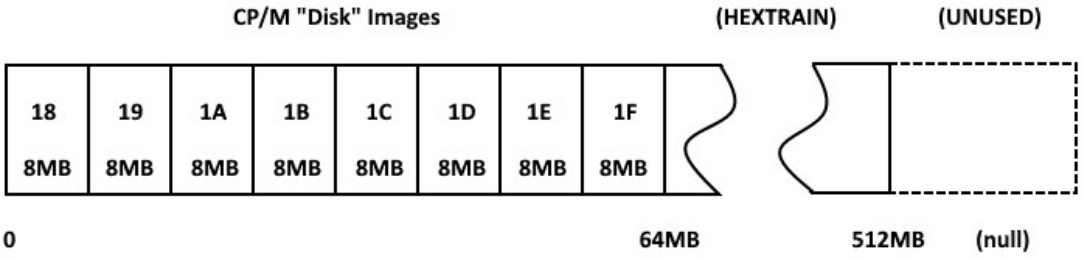
When you have typed in a command, it is entered by pressing the return key. CP/M will then perform the action you have requested.

Okay, your fingers are itching to type in a command, but where do you start? You have a choice. You could ask CP/M to run one of the programs on the disc (as you will see later), but, more likely, you will first want to use one of CP/M's built in commands. Here's what they are and how they work.

Before experimenting with the various CP/M commands it is advisable to make a copy of your system disc (SD Card). Since the MTX-MFX has only a single SD card slot, to make a backup copy of the entire contents of the SD card you will need to use a PC based imaging program, refer to Appendix A for further information.

However, it is possible to backup the contents of the boot partitions to other areas of the SD Card.

The structure of the MTX-MFX SD card is illustrated below



The SD Card supports up to 8 x 8MB partitions placed in a (logically) contiguous block of memory cells on the card, i.e., the maximum storage available to CP/M is 64MB, of which 32MB or 4 x 8MB partitions is accessible concurrently. A “hidden” block, up to 512MB is reserved for use by Andy Key’s HEXTRAIN. Any space above 512MB cannot be used by any of the SD card based systems based on Andy Key’s SD Card disk types, including MTX-MFX, so the use of larger sized cards is a waste.

When the system is booted in the default configuration as previously described, “discs” B to E are mapped to the first 4 x 8MB partitions on the SD card, addressed as disc types 18, 19, 1A and 1B.

Note: whilst a single 8MB partition is almost certainly large enough to store all of the data and programs that you may need, there is a limit of 512 directory entries for each partition – you are likely to run out of available directory entries long before the partition is actually full. (Since CP/M 2.2 does not support directories, anything approaching this limit is likely to be unmanageable anyway.)

By default, MTX-MFX is supplied with software loaded to some or all of the first 4 partitions, the remaining partitions are blank. These can provide additional storage for user files, but are also a convenient location to store backups of the distribution files.

To do this, simply follow the instructions listed below. They will be explained later in the manual. For example, with the system booted from the first partition, you want to back up the system “disc” to the first unused partition, 1C, temporarily mapping the partition to the E “disc”

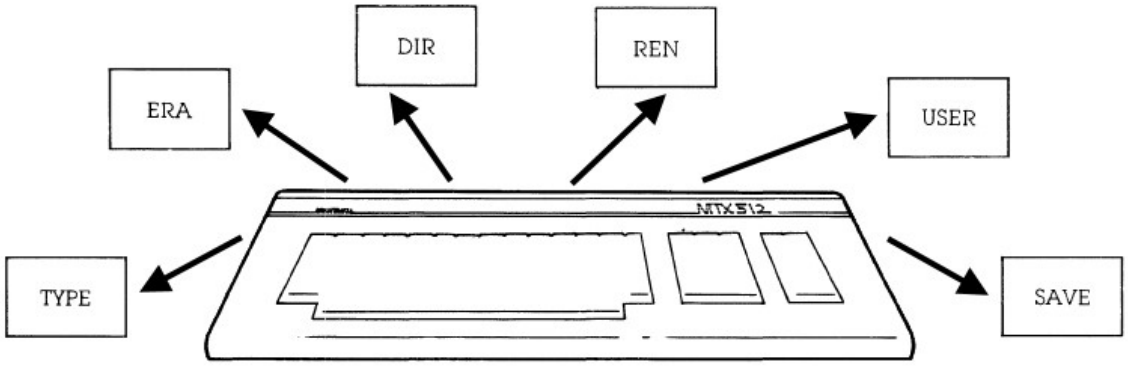
1 Type ‘**RECONFIG E: 1C**’
2 Type '**REFORMAT E:**'
3 Press the return key in response to the message
Ready to format the disc in drive E:?
Type ‘**CR**‘ to go ahead or "c” to abort
4 Type '**SYSCOPY E:**'
5 Type '**PIP E:= * .***'

When the computer has finished copying all the files to the new disc, you can type ‘**DIR**’ to list the files on the boot partition (A:) and ‘**DIR E:**’ to list the files on the newly created partition and you should find that the listings are the same.

At this point, you may either leave the new partition mapped to drive E, or revert to the original mapping by typing ‘**RECONFIG E: 1B**’

Now you can continue experimenting with CP/M without any fear of damaging your precious system disc.

4 Built-in Commands



When CP/M enters your computer. it is instantly ready to accept any one of six commands. This section will cover how each of these commands are used - but first, here's a quick look at what they are and what they do:

Command	**The result**
DIR file name (optional)	directory of disc
ERA filename	erase a file
TYPE filename	type file contents
REN new=old	rename a file
USER number	user number change
SAVE b filename	save a new file (advanced command)

Each command is typed on the keyboard and followed by pressing the return key. Most, however, require that you also specify a particular 'filename.' This is the name given to each filed program or group of data - saved on the disc.

You may have lots of files on a disc or you may have just a few. But every file has a different name - so you and CP/M can tell them apart. To see what the names are, you begin by using the DIR command.

4.1 DIR- Looking at filenames (directory)

Format: DIR
DIR NAME.TYP

Each program or group of data on a disc is held in its own individual 'file'. And, so you and CP/M can tell one from the other, each file is given a specific name. The DIR command is used for displaying the list - the directory - of filenames on the disc.

After the A> prompt. you type:

A>DIR

followed by a RETURN. This tells CP/M you'd like to see the names of all programs and data presently on the disc. Depending upon the files currently on the disc you're using, the result will appear on your console somewhat like this:

The A: in the left column tells you that drive A is presently being shown. (The bottom A> is the prompt for your next command.) The words you see are the names of each file contained on the disc. No two files will have the same name on the same disc.

Each file name is made up of two words, such as DUMP and COM. The first word is the file 'name' and can be up to eight characters long. The second word is the file 'type,' and it may be up to three characters long.

While the file name may be virtually anything, the file type usually describes the kind of data the file contains - such as TXT for the text of letters or VAL for the values in number tables. The COM file type is the most common. It stands for 'command' file - that merely means it is a program that may be run by your computer. (You'll see how to start a program file a little later.)

Eventually, as you add programs and data to a disc, the directory display becomes cluttered with various names - it then becomes tedious to look for just the file you want. But there is an easy way:

Suppose you want to check the directory for a particular file on the disc. The DIR command, in addition to the filename, will tell you if it's there. For example, type

A>DIR DUMP.COM

and you have asked to see a directory of a specific filename. The fullstop, '.', is used in your command to separate the file name from the file type. The result is

A>DIR DUMP.COM

A: DUMP COM
A>

on the console, if the file does indeed exist. If not. CP/M will display the words 'NO FILE' on your screen.

Can't find the file you want? Maybe it's not there or maybe you've forgotten its exact spelling and the DIR command can't find a match. Don't give up - here's another way to look for files:

Finding groups of files with DIR

Whenever you add a filename to the DIR command, you are asking CP/M to try and match every character you've typed. But you can make your request a lot less specific to make CP/M search for more than one filename at a time. That allows you to check for entire groups of files - all letters you've written, perhaps, or all files that contain numerical data.

It's done by using asterisks,'*', and question marks,'?', within the filename you give. The asterisk can be used in place of an entire file name or file type. The question mark can take the place of any character in the filename.

Wherever used, these characters tell CP/M that a match is not necessary in these positions. Only the characters in the remaining positions are checked.

For example, suppose you'd like to see a directory of all files that have a VAL file type. Give the command:

```
A>DIR *.VAL
```

Since the asterisk is used in place of a file name, any file name is acceptable for a match. Only the file type, .VAL will be checked. The result could be all your income tax files for the past few months:

```
A>DIR *.VAL

A: JAN      VAL  : FEB      VAL  : MAR      VAL  : APR      VAL
A: MAY      VAL  : JUN      VAL  : JUL      VAL  : JUL      VAL
A>
```

Or, you can use the asterisk in the file type you specify:

```
A>DIR JAN.*
```

will find all files with a JAN file name, regardless of their file type. That could result in:

```
A>DIR JAN.*

A: JAN      VAL  : JAN      TXT  : JAN      MEM
A>
```

all the JAN files - in this case, the values you used for your income tax (JAN.VAL), the text of the letter you wrote to the Inland Revenue to explain it (JAN.TXT), and the memo to your lawyer (JAN.MEM) asking him to explain it.

Just as the asterisk allows any file name or type to be accepted, you can use the question mark to substitute for any character in the filename. You may use as many as you wish, and anywhere you wish. The position in which they're used simply isn't checked for a match. For example

```
A>DIR J?N.VAL
```

asks for a directory of files with a three-letter file name that starts with the letter J and ends with N. The middle letter can be anything. The file type must be .VAL, Result:

```
A>DIR J?N.VAL

A: JAN      VAL  : JUN      VAL
A>
```

Only the JAN.VAL and JUN.VAL files are displayed since they are the only two that match your request.

You can also mix the asterisks and question marks in your specification. For instance

A>DIR J?N. *

again asks for all files with a three-letter file name that starts with J and ends with N. But this time, because an asterisk is used, the file type is not important. Or, to be even less specific, you could type:

A>DIR J??. *

for all files with a three-letter (or less) file name that start with the letter J.

Depending upon how you use the DIR command, it will show all or any particular group of files contained on a disc. It allows you to instantly locate all the important - and, not so important- files on the disc. The important ones you keep; the next section, erasing a file, explains what you can do about the others.

The only file the DIR command will not display is the CP/M program. That's because CP/M is kept in its own special area of the disc.

4.2 ERA - Erasing a file from disc

Format: ERA FILE NAME. TYP

Eventually, there will be a time when you want to erase files from a disc. You may want to eliminate old programs or data you never use. When erased, the space taken by these files is made available for future files.

The ERA command requires that you specify a filename. For example:

A>ERA MOTHER.LET

will remove the letter you wrote to mom. As before, the file name and type are separated by the fullstop. CP/M will not indicate that the file has been erased. But if you're the nervous type, use the directory command (DIR) to be sure the file has been removed.

If you try to erase a file that doesn't exist, CP/M will respond with

NO FILE
A>

You can remove more than one file at a time by using the asterisk or question mark in the name you've specified. All files that match your request will be erased. For example:

A>ERA JUNK.*

erases all files named JUNK regardless of their file type. Or if you type

A>ERA J??K.*

all text (TXT) files with a four-letter file name that start with J and end with K will be removed.

Naturally, when using the question marks or asterisks be sure you don't accidentally erase a good file simply because it too is a match. Once you erase a file, it cannot normally be restored. (A special 'UNERASE' program is required to restore an erased file, but it is not included with CP/M.) For that reason, if you ask for all the files to be removed with the command

A>ERA *.*

CP/M will check to be sure this is really what you want. On the console you'll see

A>ERA *.*
ALL (YIN) ?N

If you type 'Y' for yes, all the files will be removed. But if you've made a mistake, simply type a 'N' for no. Then CP/M will forgive you and merely display a new prompt. No files will be erased.

Finally, there is just one situation that will prevent files from being removed: you have purposely asked CP/M to protect them from accidental erasure. (Certain files or an entire disc may be protected.) This is done with the STAT utility program (described later) and CP/M will warn you with an error message if this occurs.

You don't know if you want the file erased or not? Then take a look to see what's in it. You can do that with the next command.

4.3 TYPE - Typing a file

Format: TYPE FILE NAME.TYP

The TYPE command displays the contents of a file on the console. You can use it as a quick way to show the contents of a letter, for example, or perhaps to display the raw numbers used by another program.

There are two rules for using this command, however. First the complete filename must be specified. The asterisks and question marks won't work. And second, only files that contain textual information (conventional letters and numbers) can be displayed. (Other kinds of files can be specified, but since they contain program information only usable by the computer, they produce gibberish on the console.) The command is used with a specific filename as in:

A>TYPE ABE.LET

If the file ABE. LET exists on the disc, you will instantly see the text of the letter you wrote to Abe on the console:

Dear Abe,

As your agent, I would like to apologise for the mix-up in your

last travel arrangements.

Yours Sincerely
G. Lee

On long files. the contents will continue to the bottom of your screen and move up (scroll) one line at a time until the file ends. You can temporarily stop and start the display scrolling (so you can read it without a speed-reading course) by pressing CONTROL and alternately pressing the S key. Or, you can prematurely end the file display by pressing any other key on the keyboard.

You may also print the file on paper if a printer is attached to your system. To do it press the CONTROL and P keys before finishing your command with a RETURN. From then on, everything you see on the console will also be printed on paper. To stop the printer from duplicating the console display, merely press the CONTROL and P keys once again.

4.4 REN - Renaming a file

Format: REN NEW.TYP=OLD.TYP

As your library of program and data files grows, their filenames become more important. Similar names on related data can be helpfuL for example. But similar names on unrelated files can become confusing. To help you organize your files, there is the Rename command. It allows you to give an old file a new name - without changing its contents.

The Rename command is used with the new filename set equal to the old filename as in

A>REN NEW.TXT=OLD.LET

In this, the file called OLD.LET will be changed to NEW.TXT. The new name is always specified first followed by an equal sign,'=', and old filename. If you forget which name goes first think of the command as 'let new=old.'

The old file name and type must be specified completely- no asterisks or question marks. The new file name can be up to eight characters long. The file type is optional but up to three characters can be used. You can use any standard character or number except for a space and

< > . , ; : = ? * ( )

These characters, such as'*' and'?' as you've seen, are reserved for special meanings. (Others will be shown later.)

There are only two mistakes you can make using the Rename command - and CP/M is ready for both of them. It you try to rename a file that doesn't exist the console will display

NO FILE
A>

And it you try to give an old tile a name that already exists on the disc, you'll see

FILE EXISTS
A>

You could, however, rename a file on one disc that matches the name of another totally different file on a second disc. That could cause confusion in the future. For that reason, it's wise to keep a log of all the filenames you create or change.

4.5 USER - Changing users

Format: USER number

In computer terms, you're called a 'User'. (Rather impersonal, but it's to the point.) And, when CP/M is first entered into your computer, it assumes that User number 0 is in control. You can run programs, save data, rename files - all the things CP/M is capable of doing.

But now your brother wants to use the computer. He wants to use the same disc, but he also wants to create and use his own library of programs or data. To do it he merely types

A>USER 1

and CP/M will again respond with a new A> prompt. The difference, however, is that all your files become inactive and only the files he creates will be in use. (Your files are still on the disc, but merely stored in the User 0 section.) All programs and data he saves will be put in the User 1 area, and, if he asks for a directory with the DIR command, only his files will be displayed.

Fine, but now your sister wants to 'compute.' And she too wants her own section of the disc. To do it she types

A>USER 2

and CP/M again responds with another A> prompt. Now both your files and your brother's become inactive and only files she creates can be used.

You have a big family? No problem. The User command will accept up to 15 different users – provided there's room on the disc. (You'll see how to check remaining disc space later.) Remember. the disc space has not been increased, only shared by many people.

You can change the User number at any time. Re-entering CP/M, as when first turning on the power to the computer or pressing the RESET button, will automatically reset the system to User number '0'.

4.6 SAVE- Advanced command to save data

Format: SAVE b FILENAME.TYP

Usually files are automatically created - saved - on the disc by outside programs. (Editors create text files, for example, or a spread-sheet program will save all the numbers you need.) But for more advanced programming, CP/M allows you to save a file directly with its Save command. (For normal CP/M operations, this command is seldom used.)

The command is used with a file length specification followed by a file name and optional file type. For instance:

A>SAVE 4 HELP.ME

will save four 'blocks' of data presently residing in memory. The file will have HELP as a name and ME as a file type assigned to it.

To determine the number of blocks you'll need, merely multiply the size of the file you're about to save by four. A 2K file (remember a 'K' is one thousand bytes) will require eight blocks to save it (2x4); a l5K file needs 60 blocks (15x4); a 30K file will need 120 blocks (30x4). The value of four is used simply because CP/M uses four blocks to save every l K.

To save a 20K file, for example, your command is

A>SAVE 80 FILE NAME. TYP

As in the RENAME command, the file name can be up to eight characters long. The file type is optional but up to three characters can be used. And as before, you can use any standard character or number except for a space and

< > . , ; : = ? * ( )

>.';: =? * ()

Be careful when making up the filename for the save command, however. If the file already exists, CP/M will not warn you –

IT WILL REPLACE THE EXISTING FILE WITH THE NEW ONE JUST SAVED.

While this is fine if you are updating an old program, it would be disastrous if a different program exists on the disc with the same name.

The largest numbers of blocks you can save is 255. If you specify more, CP/M will merely repeat the number with a question mark on the console and no data will be saved.

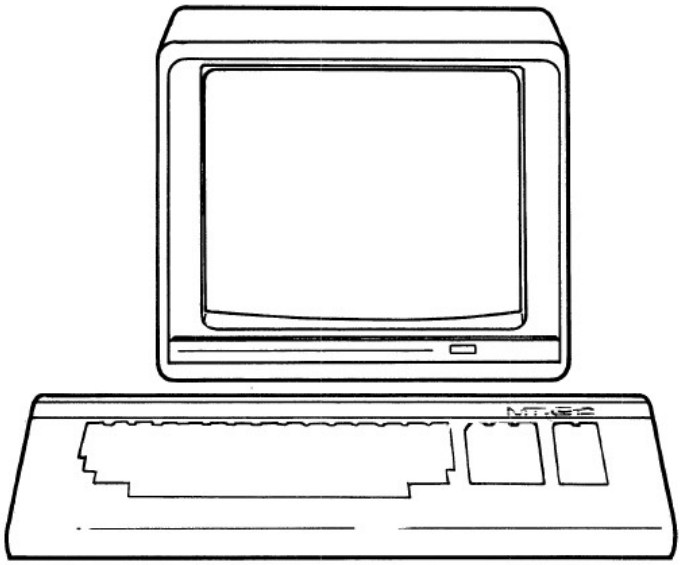
There is also a limit on how much data the disc can contain - and another limit on how many filenames the directory can hold. Both values will depend upon your disc system. If you exceed either while trying to save a file. you'll see

NO SPACE
A>

and, unfortunately, only part of your file will be saved.

The good part about seeing the NO SPACE message is that you get to try out the ERASE command to make more room on the disc. Or, if you are using a floppy disc, you can replace it with an empty one. But if you replace a disc, be sure to press the CONTROL and C keys before you try to save the data again. This tells C:P/M a new disc is in use. (**More on this later in the Changing Discs and Drives section.**)

5 Running a Program



Each time you type on the console, CP/M first checks to see if your entry is one of the previous built-in commands. If it's not CP/M knows you want to run a program on the disc.

Each program, of course, is saved on the disc as a file. The file name may be anything: EDITOR, for a text editor program for example, or maybe OTHELLO for a game. The file type, however, must be 'COM' (for command). CP/M only accepts a COM file type as a directly executable program. To start a program, you merely type its file name, as in

A>OTHELLO

The file name must be exact - no question marks or asterisks. But the file type is not used since COM is assumed.

After typing its name and RETURN, CP/M looks for the program on the disc. If it's not found, you'll merely see the filename and a question mark displayed on the console.

When found, the program is immediately entered into your computer and started. From then on, the commands you use will depend on the instructions supplied with that program.

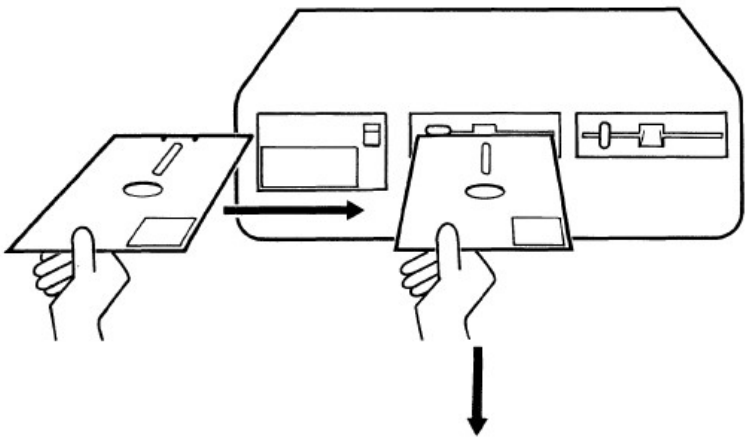
Some programs, such as a text editor, may require an additional filename to be specified. (The instructions with the program should tell you.) If so, merely add the filename to your command separated by a single space. For example:

A>EDITOR BUDGET.TXT

will enter a text editor called EDITOR into your computer and it will use a second file called BUDGET.TXT to save or retrieve all the words you've written.

Great but what happens when the program you want is on another disc? That's no problem - and that's in the next section: **Changing Discs and Drives**.

6 Changing Discs and Drives



The concepts in this section are based on the use of a system with one of more floppy drives but in principle, they are relevant to MFX too.

Often you'll need data or a program on a disc not presently in use. You have two choices: replace the present disc with the one you need, or place the new disc in another drive. Either is a simple chore - once you know how.

When you first turn on your computer. two things happen: CP/M is entered from drive A and the disc is 'mapped.' Mapping merely means CP/M has done its own internal directory of the disc - it knows where the files are located and how large they are.

But suppose the program you'd like to use is on another disc? Naturally, on a floppy-disc system, you can remove the present disc and replace it with the one you need. Simple- but remember, CP/M has the first disc mapped in its memory.

The cure, however, is to tell CP/M you've changed discs. To do it merely press the CONTROL and C keys. When you do, the new disc will start and you will see

A>^C

The '^' (called a caret) indicates that the letter which follows was typed with the CONTROL key pressed.

This is called a 'warm boot' and it makes CP/M re-enter some of its original programming information from the new disc in drive A and remap it. (Which means that whenever you press CONTROL and C, the disc in drive A must contain a copy of CP/M. You'll see how to put CP/M on a disc in the section on SYSCOPY to come later.)

You only need to remap a new disc once- and when done (the disc spins for a second), you may continue normally.

If you don't type CONTROL and C, you will still be able to do a directory or even run a program since CP/M can find the information it needs from the new disc. But there are dire consequences if you try to save data (either directly with the Save command or indirectly through another program), since you have managed to confuse CP/M by switching discs and not telling it ...

GOOD FILES CAN BE ERASED OR HAVE THEIR CONTENTS CHANGED

For this reason, always type CONTROL and C after changing a disc, unless specifically asked not to by a program you are using. (Disc-copying programs, for example, request that you change discs without doing a 'warm boot.' These programs, however, internally ask CP/M to map the disc.) All this is fine if you wish to use only one disc drive. But if you own more, a new disc can be placed on one of those, as well. But to use it, you'll have to tell CP/M your plans:

With MFX, physically changing discs when the system is powered on will rarely be needed. However, if you do want to do it, make sure that there is no current disc (SD card) activity - in CP/M mode, the cursor colour changes to red during disc activity - and follow the instructions to type CONTROL and C after changing the SD card.

Changing drives

Historically, disc drives were the physical units used to accept your discs. You may have had one drive in your system - you may have 2 or more. Each, however, is assigned a drive letter. With MFX, there is only one physical unit (SD card slot) in the system which is used to emulate up to 4 physical discs using logical drives A to E.

The drive used to originally start your system, is drive B (logical drive A). It is the present 'active', (or 'logged-in') drive. The remaining "drives" are lettered C to E.

It you place a program disc in drive C, for example, you now must tell CP/M where it is before it can be used. That can be done in two ways.

The first is to make the new drive you wish to use the 'active' drive. To make drive C active, for example, simply type

A>C:

followed by a RETURN. The colon tells CP/M that the letter preceding it is a drive specification. (Otherwise, without the colon, CP/M will assume you want to run a program called 'C'.) CP/M will respond with

C>

Look familiar? It is the same prompt as before (A>) except the A has been replaced with C. You are now 'logged-in' on drive C. You may perform all the built-in commands or run a program on disc as before- but CP/M will now use drive C as the source. (The first time you change to another drive, CP/M will map it. If you change discs after that however, remember to have it remapped by pressing CONTROL and C.)

Any drive may be selected by merely typing its letter and a colon. You can return to drive A if you wish by typing A:. Each time you change the drive, CP/M will respond with a new prompt.

The only problem you can possibly create is by asking for a drive that does not exist.

Another way

The second way you can use a program on another drive is to precede the program or data

you want with the drive letter on which it's to be found. For example:

A>C:EDITOR

will tell CP/M to go to drive C and run a program called EDITOR. The difference here, however, is that drive C is not made the active drive - it is merely the temporary source of the program. Any data saved by the EDITOR program will be saved on the active disc which, in this case, is still drive A. CP/M's built-in commands work similarly. For example:

A>DIR C:

will show all the files on the disc in drive C. The result may be

```
A>DIR C:
C:  PIP      COM : FORMAT   COM : CONFIG   COM : TECHREP1 BAK
C:  TECHREP1 DOC : WSMSGS   OVR : WSOVLY1  OVR : WINSTAL  BAK
C:  WS       COM : WINSTAL  COM : CONRAID  BAK : MTXPAY   BAK
C:  MTXA/C   COV : MTXPAY   DOC : CONRAID  SNA : MSSGE
C:  STAT     COM : SCREENS  FDX
A>
```

The C: on the left tells you the directory is of disc-drive C, but the A> prompt that follows shows that drive A is still active.

Similarly, to display only one or a group of files on another disc, a file name may be added. The asterisk and question mark may also be included. For instance:

A>DIR C: * .TXT

will display all files with a TXT file type presently on the disc in drive C. But once done, the active drive remains A

You may also type, erase, rename, and save files on another disc. In each case, the disc-drive letter merely precedes the filename and tells CP/M to temporarily change drives before executing. When your command is completed, CP/M always returns to the active drive:

A>TYPE C:LETTER.TXT
Type LETTER.TXT file on drive C. (Active drive is A)

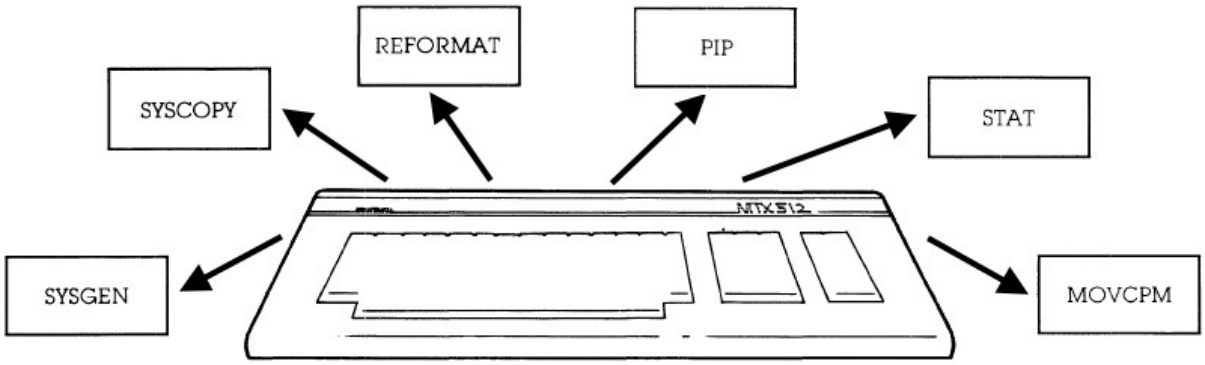
C>ERA B: *.LET
Erase all files with the LET file type on drive B. (Active drive is C.)

C>REN B:NEW.TYP=OLD.TYP
Rename OLD.TYP file to NEW.TYP on drive B. (Active drive is C.)

C>SAVE 10 B:PROGRAM
Save 10 blocks on drive B and call the file PROGRAM. (Active drive is C.)

All of these commands will be performed on another drive. But when the action is complete, CP/M will return to the active drive and display its letter with the > prompt.

7 CP/M Utility Programs



CP/M comes with a number of 'utility' programs. These programs are just like any other you run on your system, except they are specifically written to help you use CP/M - and your computer more efficiently.

Although some have unique applications and are provided tor computer programmers (these programs are covered later), many are for general use: they'll help you prepare a new disc to accept data, for example, or they'll make copies of CP/M and your programs.

Here are the names of the general utility programs and a brief summary of what they do:

REFORMAT	Format a SD card partition - prepares a new partition to accept data. You'll use this program first to remove random data from the partition (caused by the manufacturing process) and to set the disc (partition) to run on your MTX-MFX. (Replaces the standard FORMAT program)
SYSCOPY	System generation - copies the CP/M program onto another disc (partition). (No files are affected.) The disc (partition) can then be used to start your system.
PIP	Peripheral Interchange Program - is often used to copy individual files from one disc (partition) to another. You can create a 'master' disc, for example, that contains all your standard programs. PIP can also exchange data between peripherals attached to your system or change as it is copied.
STAT	Status of system - displays the present status of disc files and peripherals. It can show the size of files, for example, or tell you how much space is left on the disc. It can also change the status of files and peripherals tor special uses.
MOVCPM	Move CP/M - moves CP/M so it may use additional memory you've installed in your computer.
WRTCPM	Writes the CCP and BDOS section of CP/M 2.2 on to the Bootstrap area of the disc in drive specified.
WRTBIOS	Writes the custom BIOS and the COLD-START loader section of CP/M 2.2 on the Bootstrap area of disc (partition) specified.

The order in which you use these programs will of course, depend upon what you wish to do. Often, however, they are used in a particular sequence to perform three routine jobs: make a backup copy of your CP/M disc, make a backup copy of a new program disc you just bought or make a general disc of various programs you use everyday.

While the sections to come explain exactly how to use these programs, here's a quick look at the order in which they may be used to complete these three basic functions:

Make a backup copy of your CP/M disc

1 Use REFORMAT to prepare a new disc (partition) to accept programs.
2 Use PIP *.* to copy the entire original CP/M disc onto the empty disc.

Copy a new program disc

1 Use REFORMAT to prepare a new disc (partition) to accept programs.
2 Use PIP *. * to copy the entire new program disc onto the empty disc.
3 Use SYSCOPY to copy CP/M onto the disc so it may be used in drive A to start your system. (Since a new program disc you buy does not contain CP/M, it will not have been copied onto the empty disc.)

Create a general disc for everyday use

1 Use REFORMAT to prepare a new disc (partition) to accept programs.
2 Use SYSCOPY to copy CP/M onto the disc so it may be used to start your system.
3 Use PIP to copy individual programs from other discs onto this empty one.

In the sections to follow, you'll see exactly how each of these utility programs is used - and you'll see what other jobs they can do for you.

As with any program, the utility you want can be started by merely typing its file name on the command line, followed by a RETURN. Utilities that work on a disc file require its name in the command line as well. Here are the details ...

7.1 REFORMAT - Preparing a new disc (partition)

Format : REFORMAT drive:

REFORMAT replaces the FORMAT program supplied with Memotech FDX/SDX systems and whilst similar in operation, there are some fundamental differences. Much of the FDX content is reproduced here and while not directly applicable to MFX, helps describe MFX's REFORMAT

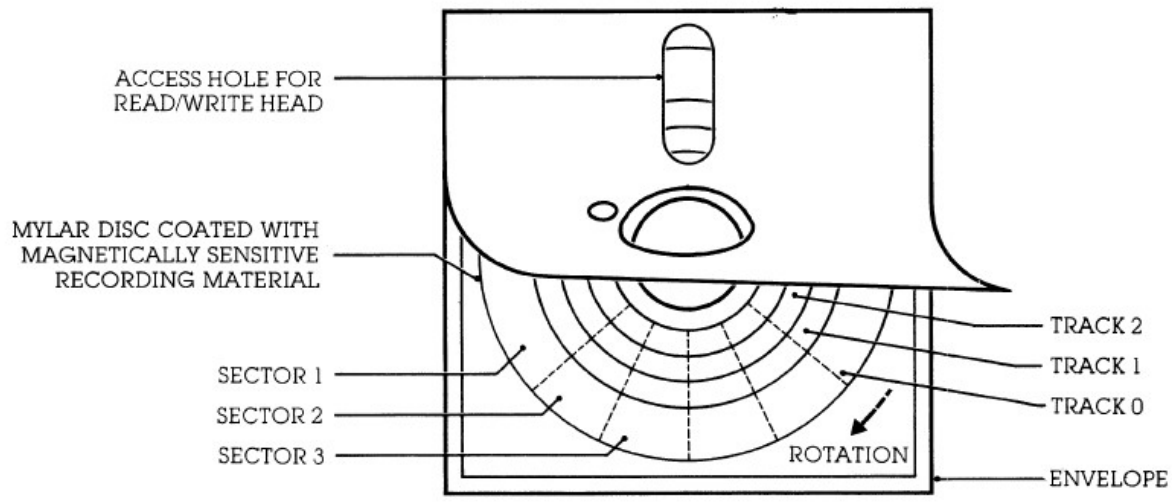
Write Protect

Physical floppy disks have a Write Protect notch that may be either closed (protected) or open (unprotected) that allows the user to protect the contents of a disk from being accidentally overwritten by format, copy and erase operations. The SD card hardware used in MFX does not support hardware write protection; even if the SD card "Lock" switch is used, the operating system will ignore its position and write to the SD card (unless the software Read Only attribute is set – see later.)

What FORMAT did to your disc

With a floppy disk, FORMAT is actually recording a character (an 'E5' in computer code) throughout the disc that CP/M later recognizes as an 'empty' spot. The characters are recorded together in small groups, called 'sectors: and sectors are recorded end-to-end in concentric circles around the disc in 'tracks.'

In this way, CP/M will later know where to save - and retrieve - your data by merely going to a particular sector and track number.



The distance between each empty character placed on the disc is carefully controlled by FORMAT. The denser the characters, naturally, the more data the disc will eventually be able to hold.

RUNNING FORMAT

FORMAT d:

This command formats the disc in drive d. No writing is done until confirmation of the command is received and its validity checked. The 'd' must be an actual physical drive, i.e., B,C,D,E,F,G,H or I.

FORMAT generates the appropriate sector interleave pattern, optimised for the current drive configuration. Type 10 drives (8 inch S/S,S/D) are always IBM compatible but other configuration types may not retain compatibility across controller types - ie prom version numbers.

What REFORMAT does to your disc (partition)

As described in Chapter 2, the “discs” in MFX are partitions on the SD memory card, each providing 8MB of storage, split into logical sectors that CP/M sees as consisting of 128 bytes. In a similar way to FORMAT, the REFORMAT program overwrites the reserved “tracks” (CP/M system) and directory areas with ‘E5’, effectively wiping the “disc”.

In addition, unlike the original FDX Silicon Disc, the MFX RAM disk does not require the use of the SiDisc driver or format utility and REFORMAT can also be used to format the RAM disk (see 8.12).

RUNNING REFORMAT

REFORMAT d:

This command formats the disc in drive d.

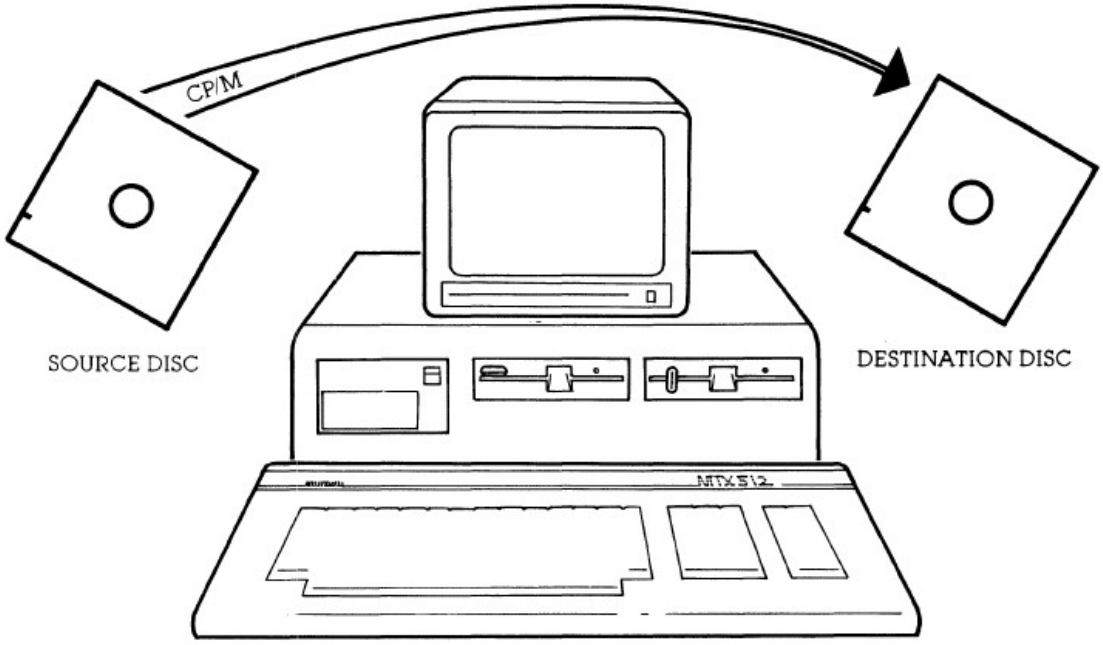
Note: Unlike the original FORMAT program, REFORMAT does NOT require confirmation of the command before it starts; provided that a valid disc is selected, the process will start as soon as you enter the command.

MAKE SURE THAT YOU REALLY WANT TO DELETE ALL OF THE DATA ON THE TARGET DISC BEFORE USING THIS COMMAND!

The 'd' must be one of the four discs (partitions) currently visible to CP/M i.e., B,C,D and E, or the RAM disk, F.

The current boot disk cannot be formatted. Formatting the extended partitions (1C, 1D, 1E and 1F) requires that the partition is first mapped into one of the “visible” drives as described in Chapter 2.

7.2 SYSCOPY - Copying CP/M



Any disc (partition) in MFX can be used to initially start your system. But naturally, there's a catch – the disc must contain a copy of CP/M. You can place CP/M on any disc you wish with the SYSCOPY utility program.

SYSCOPY records CP/M onto its own special place on the disc called the 'system tracks.' These are simply the first two recording areas (created by the REFORMAT program) on the disc.

This area is reserved for the CP/M program only - other files cannot use it. That means you can place CP/M on a disc at any time- just after the disc is formatted or later, after it is filled with files. You can also use SYSCOPY to replace the present CP/M program on a disc with a newer, updated version.

SYSCOPY d_{dest}:<=d_{src}:>

This command initiates a simple copy of the bootstrap area from a specified source disc d_{src} , (or the currently-logged disc if omitted) to that of the specified destination disc d_{dest}. If a new startup command or configuration default is required, different to that of the source disc, it will be necessary to perform a STARTUP operation (see Chapter 8). Note that if 'dest' and 'src' are the same then a pause, prior to writing, is generated to allow the user to change discs before typing CR in response to the prompt.

CP/M System Size

The original FDX was supplied with two CP/M system sizes – one which left 54kB free for user programs and one 59kB. (It is believed that the smaller system size was to allow RAM space for the

Silicon disc system drivers). All SDX systems were supplied with 59kB systems. The standard system size used by MFX is the 59kB version.

Andy Key provided a 54kB system disc image for use in REMEMOrizer to allow the user to run one game (Icicle Works) that would only run under the 54kB version of CP/M. The SD card distributed with MFX also includes a copy of the 54kB system and the Icicle Works game.

Make sure that you know which version of CP/M you are running and which system you want to clone before using SYSCOPY on MFX.

7.3 PIP – The Peripheral Interchange Program

WHAT IS PIP?

One of the most powerful utility programs you own is PIP - the Peripheral Interchange Program. PIP can copy individual files - one at a time, or many at once. That allows you to place many different programs from a variety of sources onto a single disc.

But copying files from one disc to another is just part of PIP's abilities. It can also exchange data between peripheral devices - or change data during the copying process.

You can use the PIP program in two ways. The first is to type its file name along with your command:

A> PIP your command here

With this, PIP will enter memory, execute a command, and return to CP/M. This is used when only one PIP operation is needed.

To perform many operations, however, it's faster to keep PIP in memory (so you don't have to wait for it to enter memory for each command). You start this second command form by typing just

A> PIP

followed by a RETURN. When you do, you'll see the PIP '*' prompt:

```
A>
A>PIP
*
```

You may now enter your command on the line, followed by a RETURN. It will be carried out but you will not return to CP/M. Instead, PIP remains in memory and you'll see another * prompt for another command on the next line:

```
A>PIP
*command 1
*command 2
*
```

When you're through, you can exit PIP and return to CP/M by pressing just the RETURN key (entering an empty line).

The first command form is the most common - it will be used in the examples that follow. But in all cases, either form will work.

Naturally, the commands you use will determine the job PIP will perform. And in the major sections to come - covering PIP's copying commands and PIP's special commands - you'll see how it can be used in a variety of ways.

7.3.1 USING PIP TO COPY DISC FILES

PIP's most useful function is to make copies of disc files. You can make a backup file of am original on the same disc or you can copy a file - or group of files - on another disc.

The new file you wish PIP to create is always specified first. It's directly followed by an equal sign, '=', (no spaces) and the old file you'd like copied. (Think of the command as 'let NEW=OLD.') The basic command is

A>PIP NEW.TYP=OLD.TYP

When drive letters are not specified, the active drive, the one that appears in the prompt is assumed. Here, a file called NEW.TYP will be created on the disc in drive A and it will contain the same information as the OLD.TYP file on drive A. This is useful for making backup copies of a file on a single disc. (If NEW.TYP already existed on the disc. it will be replaced by this latest version.)

Most often, however, you'll want to copy a file from one disc to another. To do it, add the drive letter and a colon (:) before the name:

A>PIP C:NEW.TYP=OLD.TYP

As before, a file called NEW.TYP will be created. But since the drive letter (C:) was specified, the new file will be on the disc in drive C. The file will be identical to the OLD.TYP on the active drive (A). By adding other drive specifications, you can copy a file from one disc to any other. For example:

A>PIP C:NEW.TYP=B:OLD.TYP

Here, the new file will be created on drive C, and it will be identical to the old file on drive B.
Just as PIP assumes the active drive if you leave out a drive specification, the program can assume more. If you merely specify the destination drive letter and leave out the filename as in

A>PIP C:=OLD.TYP

then PIP will assume you wish the new copy to have the same name as the original. In this case, PIP will create a file called OLD.TYP on drive C and it will contain the same information as the original OLD.TYP file on drive A.

Similarly, one of the more powerful ways to use PIP is to include the asterisks (*) or question marks (?) in the filename you specify. All files that match your specification will be copied - and that allows more than one file to be copied with one command. For instance:

A> PIP C:= * .TXT

instructs PIP to copy all files with the TXT file type on the disc in the active drive (A) to the disc in drive C. The filenames of the newly created files will be identical to the originals. (PIP will display each file on the console as it is copied.)

And to copy all the files from one disc to another, use the asterisks in both the file name and file type, as in

A> PIP C:= *. *

All files on the active drive's disc (A) will be copied to the disc in drive C under their original names.

Old files on the destination disc are not destroyed, PIP merely adds the new files from the source disc to the files already present on the destination disc.

Placing additional drives in your command makes PIP transfer all files from any disc to any other. For instance:

A> PIP B:=C: *·. *

will copy all files from drive C to drive B.

Great! But what happens if there's a (gulp) power failure or any other technical interruption during these copying procedures? For safety, PIP automatically copies a file in two steps:

First, it creates a temporary file on the destination disc using the file name you've specified and a '$$$' file type. The data from the original file is then copied to this file. If the transfer was successful (no power failures, earthquakes, etc.), the second step is done: the new temporary file is renamed to the new file name and file type you've specified.

If there were a power failure - or your kids decided to push the computer RESET button during the transfer - only the temporary file will lose the data. You simply use PIP again. But all this also means you'll need extra room on the destination disc (equal to the size of the file you are copying) for the temporary file. The **STAT** utility program, described later, will show how much space a file requires and how much room is left on the disc.

7.3.2 USING PIP TO COPY BETWEEN PERIPHERALS

While PIP is used most often to copy files between discs. it can also be used to exchange data between 'peripherals'.

Everything connected to your computer is a peripheral - from required devices, such as the disc drive and keyboard, to more elaborate add-ons, such as a modem for sending and receiving data over a phone line. PIP will allow you to exchange data between all peripherals attached to your system - and for a variety of tasks:

Have a disc file placed on paper by sending it to the printer. Type a quick memo on the keyboard and have PIP send it to the disc for saving - then read it later by asking PIP to show it on your console screen. Send data to a friend by using PIP to control a phone modem. Or receive data from a friend and have PIP put it on your console for viewing now - or on your printer for reading later.

Still using a typewriter for small jobs? With one command, PIP can turn your computer into an 'electronic' typewriter. Everything you type on the keyboard instantly goes to the printer.

To do all this and more, you use PIP as before except the name of the peripheral device you'd like to use is placed in your command. To make things simple, PIP accepts four major device names: CON:, LST:, RDR:, and PUN:. Here's what those names mean to PIP:

NAME	MEANING
CON:	Console- The keyboard you use for entering data and the display (usually a video screen) you use for seeing the results.
LST:	List- The printer you use. (It 'lists' data.)
RDR:	Reader - Any auxiliary device that 'reads' data into your computer. This could be any device you've attached to your system, such as a modem. The data received over the phone line by the modem (from another computer) is read into your system.
PUN:	Punch-Any auxiliary device that 'punches out' data from your computer. This too can be any device you've attached to your system, such as a modem. The data sent to the modem from your computer is transmitted over the phone line.

You can use any of the four peripheral names in your command line to send or receive data. As before, you refer to a disc file by using its filename and optional drive letter. For example, to send a disc file to the printer. Type

A>PIP LST:=FILENAME.TYP

and the printer (LST:) will receive the contents of the file you've specified. Don't forget the colon - without it PIP will think you want to make a new file called 'LST'.

The receiving device is always specified first followed by an equal sign (no spaces), and the source of the data. (The order is as before: 'let LST:=FILENAME.TYP'.)

With no drive letter specified, the default drive is used. But you can change that as in

A>PIP LST:=C:DOGS.DAT

Here, the data from a file on drive C will be placed on paper for future reference.
Similarly, you could send a file to your console screen for viewing by giving the command

A>PIP CON:=FILENAME.TYP

The console (CON:) receives the contents of the file you've given and displays it. As is, this command will give the same results as when using the built-in TYPE command. But later you'll see how PIP can be used this way to change the file before it's displayed. (Lines too wide for the video display can be automatically truncated, for example.)

By merely reversing the order of your command, you can make a disc file the receiving device.
For instance, type

A>PIP FILENAME.TYP=CON:

and PIP will create the file you've specified and begin filling it with characters you type at the console keyboard (CON:). This is handy for writing quick memos to yourself. When you're through, merely press the CONTROL and Z keys (which produce an end-of-text computer code). PIP will get the message and return you to the CP/M prompt. Later, when you're ready, use the TYPE command to display the memo on your console.

Often, disc files are not part of your command. Turn your computer into an electronic typewriter, for instance, with:

A>PIP LST:=CON:

The printer (LST:) will receive all characters typed on the console keyboard (CON:). (End it with CONTROL and Z keys.)

The Reader (RDR:) and Punch (PUN:) devices may be used as well, but these usually refer to an auxiliary connection on the rear of your computer: Whatever is plugged into the connector will be used when the RDR: or PUN: device names are used in your command. (The connector is called a 'port' and you'll have to check your computer technical manual to see how different devices may use it.)

If a modem is plugged into the auxiliary connector. for instance, and you wish to use it for receiving data from a friend over the phone line, give the command

A>PIP CON:=RDR:

and incoming information will be read in (RDR:) from the modem and displayed on the console screen (CON:). Or, to put the incoming data on your printer, type

A>PIP LST:=RDR:

and you can read it later. Then when you're ready to give a reply, use PIP again with the command

A>PIP PUN:=CON:

and all characters you type at your keyboard (CON:) will now be punched- sent- out to the modem for transmission over the phone to your friend.

Or if you're clever- and hove an answer for everything- store your pre-planned replies on disc. Then send the one you need by specifying its filename as in

A>PIP PUN:=SCRAM.TXT

and your friend receives the contents of your carefully worded disc file.

Disc files of text may be exchanged this way with PIP. The file can be sent over a modem to a remote computer or you could connect the two computers directly together through their rear auxiliary connectors. (Your computer dealer or local electronics shop can supply the necessary cable.)

To ready the computer that is to receive the file, you use the command

A>PIP FILENAME.TYP=RDR:

And to send the file from the second computer, type

A>PIP PUN:=FILENAME.TYP

The file you've specified in this command will be sent- character-by-character- to the Punch (PUN:) output of the sending computer. through the connecting wires (or modem), and into the Reader (RDR:) input of the receiving computer. It will then be placed in the disc file specified in the receiving PIP command.

Whenever you are exchanging data between peripherals (other than disc-to-disc drives or keyboard-to-disc), the process can be immediately stopped by touching any key on the keyboard. When you do, PIP will tell you it is aborting its present command.

PIP will also tell you if you've made a mistake in your command. You can't, for example, ask to receive data from the printer or send data to the Reader device. PIP will re-display your request and tell you it is invalid.

Although PIP normally associates the console. printer. and auxiliary connector of your computer with CON:, LST:, RDR:, and PUN:, that can be changed by changing your system's status- and you'll see how that's done later in the section on **STAT**. Also, PIP can use other device names for special uses - and you'll see those shortly, too. But for now ...

7.3.3 OPTIONS FOR CHANGING FILES

Until now, PIP has been basically shuffling complete pieces of data from one place to another. But during the copy process you can alter the final data in a variety of ways. It's done by using one or more of PIP's special options.

Each option is basically a letter enclosed in brackets. ( ), and directly follows the source file or device in your command. You may specify more than one option- for many changes at once - by merely placing one letter after another within the brackets. You can use spaces between them for clarity if you like.

Options to change text files

The option letters specifically designed for changing files of text - conventional characters and numbers - are:

D	Delete characters	**E**	Echo characters
F	Form-feeds removed	**L**	Lowercase only
N	Number lines	**P**	Page form-feeds
Q	Quit copying	**S**	Start copying
T	Tab space	**U**	Uppercase only
Z	Zero parity bit		

Here's how they work:

Dn Delete characters

A file that is too wide to print or display on the console may be truncated - shortened – by cutting off all characters that extend past the column number you specify (n).

EXAMPLE: A>PIP:=LONGTEXT(D80)

A file called LONGTEXT from the default drive (A) will be displayed on the console (CON:). But only the first 80 characters in each line will be shown. You can use this option as a quick check of a wide file. for example, before it is placed permanently on paper.

E Echo characters

With the E option, all copying is echoed- displayed on the console. It means you get to see the data as it is copied.

EXAMPLE: A>PIP B:=C:SONGS(E)

With this command. all the words from the file of your favourite SONGS on drive C will be copied to drive B under the same name. You'll be able to sing along, however, since all characters will be echoed on the console.

F Form-feeds removed

'Form-feeds' are special codes that force a printer to the top of the next page (if your printer will respond to this code). They may be imbedded within text to prevent printing over the tear line on fanfold paper, for example, or may be used to start a fresh page at strategic points in your writing (such as the start of a new chapter). After editing the text. however. the present form-feeds may be in the wrong place. This command automatically removes them in the destination file. (They may be put back in the correct position with the P option explained later.)

EXAMPLE: A>PIP C:WORKCOPY=ORIGINAL(F)

Here, a new working file called WORKCOPY will be created on drive C. It will be a duplicate of the ORIGINAL file on the default drive (A) except all form-feeds will be removed.

L Lowercase only

This option coverts all copied characters to lowercase- an 'A' becomes an 'a', for example. The only hard part about this option is finding a use for it.

EXAMPLE: A>PIP C:LOWER=D:UPPER(L)

A file called LOWER will be created on drive C and will contain a copy of the information from file UPPER in drive D. All uppercase characters will become lowercase characters in the copy.

N Number lines
N2

This line-numbering option works with text-like tiles, but it is most often used with a 'text' of programming lines for creating a program. The option can be specified in two ways. The first N, numbers each line being transferred starting at one and continuing in increments of one. Leading zeroes are not used (001 is just 1) and each number is followed by a colon. It the N2 option is used.

leading zeroes are used and a tab is inserted after the number.

EXAMPLE: A>PIP VERSIONl.ASM=ORIGINAL.ASM(N)

A file called VERSIONl.ASM will be created from the ORIGINAL.ASM tile (both on default drive A). The new version will have all lines numbered, starting with '1:'. This option is useful for correcting errors in a program source file created by a line-oriented text editor (such as CP/M's ED program.) The numbers allow you to easily reference any line you wish for correcting with the editor. The (N2) option is similar, only lines will begin with '001' and tab characters will be inserted.

P Page form-feeds added
Pn

This option has two forms: the first, a single letter P, adds a form-feed code to the copy every 60 lines (to advance the printer to the top of the next page.) This normally leaves six empty lines at the bottom of each page. It you wish to change this, you can use the second option form to specify the number of lines per page (n) before each form-feed is added.

EXAMPLE: A>PIP LST:=MAKENEAT(P50)

The file called MAKENEAT from the default drive (A) will be sent to the list device (LST:). Formfeeds will be included after every 50 lines of text. (If your printer cannot use form-feeds, it will not advance and the text will simply continue down the page.) It only the P option is used, form-feeds will default to every 60 lines. It the F and P options are used together as in (FP50), then old form-feeds will be removed before the new ones are inserted.

Qphrase ^z Quit copying

With the Q option, you can ask PIP to look for a particular word or phrase in the original text as it's being copied. When found, the copying automatically stops. The rest of the text will be ignored. This allows you to copy just the beginning of a tile up to any point you wish. The word or phrase must be exact. Its end is marked by pressing the CONTROL and Z keys, which will produce the caret sign (^) and Z on the command line.

EXAMPLE: A>PIP EDIT.TXT=TOTAL.TXT(QBE?"Z)

It you were editing Shakespeare, this command will do nicely. The TOTAL.TXT tile from the default drive (A) contains 'TO BE OR NOT TO BE? THAT IS THE QUESTION.' But since the option specifies the phrase 'BE?' (terminated with CONTROL and Z), the result in the EDIT.TXT copy file will be just the beginning of the original file: 'TO BE OR NOT TO BE?' (Notice punctuation counts too. Without the question mark in your option, the EDIT.TXT file would have terminated on the first 'BE'.)

Now if you're alert, you'll see that all of the words in this example are in capital letters. That's because CP/M automatically converts any letter you type - small or large - into capital letters for its own use (even though they may appear in lowercase on your console). If you type 'be?' in your command it will be converted to 'BE?' for the search and a capital 'BE?' must be in the text for a match to occur. To search for lowercase characters in a word or phrase, the second command form of PIP must be used:

A> PIP
*** EDIT.TXT=TOTAL.TXT(Qbe?^Z)**
*** A>**

The second form of PIP is used by typing just its name to first enter it into memory. PIP responds with its '*' prompt and you may now type your command.

The difference, however, is that PIP will not convert lowercase characters into capitals. Now both the upper- and lowercase letters in your search phrase (be?) will be used. Once the copy is complete, PIP gives you another prompt for your next command. When you're through, type just a RETURN.

Sphrase^Z Start copying

The S option is very similar to the Q option except the phrase or word you specify gives PIP a point at which to start copying. The original text is checked for the phrase you've given. When found, the copying begins. Only the end of the original file, including the phrase, will be copied. The phrase is ended by pressing CONTROL and Z keys, producing the caret(^) and Z on the console.

EXAMPLE: A>PIP EDIT.TXT=TOTAL.TXT(STHAT^Z)

Again, as in the example used for the Q option, suppose the file called TOTAL.TXT contains 'TO BE OR NOT TO BE? THAT IS THE QUESTION.' Using the S option along with the word THAT in your command, will tell PIP to start copying only when 'THAT' is found in the original text. The result in the EDIT.TXT file will be 'THAT IS THE QUESTION.' As before, to search for a phrase or word that uses lowercase, use the second command form of PIP.

Since the S option tells PIP when to start copying and the Q option tells it when to quit you can use both letters in your command to copy extracts of a file. For example:

EXAMPLE: A>PIP EDIT.TXT=TOTAL.TXT(STHAT^ZQTHE^Z)

will tell PIP to start copying with the word 'THAT' and end when it finds 'THE'.

Tn Tab space

Tab characters in your text such as at the start of a new paragraph, can be expanded to any number of spaces in the copy using the T command. The number of spaces (n) you wish to 'tab over' is specified in your command directly following the T.

EXAMPLE: A>PIP FINAL=ORIGINAL(T5)

Here, the FINAL file will be identical to the ORIGINAL file (both on default drive A) except that all tabs in the original text will create five spaces in the copy.

U Uppercase only

This option converts all copied characters to uppercase - an 'a' becomes an 'A' in the copy, for

Example, This command is similar to the L option (Lowercase only).

EXAMPLE: A>PIP CON:=C:LOW(U)

All characters in the LOW file on drive C will be displayed on the console (CON:) as uppercase only.

Z Zero parity bit

Every character in your text requires seven separate pieces of information (called bits). The exact pattern of these bits tells the computer which character you've typed. But the computer always works with eight bits at a time - that leaves one left over. It's called the 'parity bit.' While many text editors may use this bit for special handling of the characters (it's set to l), standard CP/M programs usually require the parity bit to be set to 0. To make the conversion, use the Z option:

EXAMPLE: A>PIP STANDARD=UNIQUE(Z)

The STANDARD file will contain the same information as the UNIQUE file (both from default drive A) except that the parity bit of all characters will be set to 0. The new file will then be compatible with conventional CP/M programs.

7.3.4 OPTIONS FOR GENERAL COPYING

Besides options specifically designed for text copying, PIP also has a few more designed for any type of file transfer. They, too, me enclosed in square brackets, ( ), and may be used singularly or together. The general options are

B	Block mode copy	G	Get from user	R	Read system file
V	Verify copy	W	Write over		

Here's how they work:

B Block mode copy

When PIP copies your data, it normally takes small pieces at a time. The sending device merely waits for the recipient to copy the first chunk before giving the next. But some peripheral devices, such as a magnetic-tape player, can't stop: once started, they send all the data, from beginning to end. The B option, however, tells PIP cm entire block of data is coming. Instead of copying small chunks at a time, PIP sends all the incoming data into memory. Then, when the block transfer is complete, PIP copies the data from memory to the recipient device.

EXAMPLE: A>PIP ALL=RDR:(B)

All the data from the reader (RDR:) is first sent directly to memory. When the file is complete, it is copied into the ALL file on the default disc drive A. The size of the block you are copying will be

limited to the amount of memory in your system. If the block is too big, PIP will tell you and will abort the copy.

Gn Get file from another user area

The G option allows PIP to get a source file from another User area. The User area number, n, is given with the command and may be from 0 to 15 (as explained earlier in the Built-in Commands section under 'Changing Users.')

EXAMPLE: A>PIP A:=NEEDIT(G2)

This command will tell PIP to find the NEEDIT file in User area 2 and copy it under the same name in the present User area on drive A. (You could also change the destination file name with: PIP WANTIT=NEEDIT(G2). All conventional filename specifications, including asterisks and question marks, will work)

Since PIP is needed in a new User area to copy files, how do you first copy PIP into the new area? The answer is to place PIP in memory while in User area 0, change to the User area you want and resave PIP with the built-in CP/M SAVE command. Suppose you'd like to place PIP in User area 3. Here's how to do it:

While in User area 0, type
A> PIP
Pip will enter memory and sign on with
A> PIP
*****
Press the RETURN to exit to CP/M
A> PIP
*****
A>
Now change to the User number you wish to contain PIP
A> PIP
*** A>USER 3**
and CP/M will again show another A> prompt
A> PIP
*** A>USER 3**
A>
Next save PIP (8K x 4 = 32 blocks long and in memory) with
A> PIP
*** A>USER 3**
A>SAVE 32 PIP.COM

The PIP utility will now be on the disc in User area 3 and you may use it to copy all other files.

R Read system file

Any file on a disc can be made a 'system' file (with the **STAT** utility described later). The file can be used, but it will not appear on the console with the Directory (DIR) command. Also, the file can't be copied with PIP ... unless you use the R option.

EXAMPLE: A>PIP C:=HIDDEN(R)

The system file HIDDEN is copied to the disc in drive C.

V Verify copy

The Verity command is for nervous people; it tells PIP to go back and check the data that has been copied against the original. If a bad piece of data was somehow placed in the file copy, PIP will tell you and you can try again. (A worn disc could cause this, for example.) With this command, the destination copy must be a disc file.

EXAMPLE: A> PIP C:= *.*(V)

Here, all the files from the default drive (A) will be copied to drive C and checked against the original. The V option causes the copying process to take a bit longer, but the few extra seconds can be worth it. In fact for safety, it is normal practice to ...

ALWAYS USE THE (V) OPTION WHEN COPYING FILES.

W Write over protected files

Any file on a disc may be protected against accidental erasure. (This is done with the STAT utility program described later.) For PIP, however, this creates a problem if you wish to update one of these files- it can't be erased and replaced with a new version. Normally, when PIP finds a protected file, it will stop and ask you if it's all right to continue. (You answer with a simple Y or N.) But to quicken this process, you can use the W option to tell PIP to make the copy without bothering you with questions.

EXAMPLE: A> PIP C : =SAFE(W)

The file SAFE will be copied from the default drive (A) to drive C. If a protected file called SAFE already exists on drive C, it will be automatically overwritten. If you wish to join many files into one already protected file, the W option is merely placed at the end of all the files you've specified.

PIP can 'join' files? Yes- and you'll see how it's done in the next section covering PIP's special uses.

7.3.5 SPECIAL USES FOR PIP

This next section covers the special jobs PIP can do. They won't be used very often, but they can save you hours of work when needed.

Joining text files

You're writing a book. Each chapter is a separate file on disc, but now you'd like to put them together. To do that you can use PIP to join (concantenate) any number of files into one:

A>PIP BOOK=CHAPTER1,CHAPTER2,CHAPTER3,CHAPTER4

The receiving tile is set equal to all the filenames you'd like included, separated by commas (no spaces). Here, a file called BOOK will be created and will contain first CHAPTER 1 then CHAPTER2, and so on. All files me copied (from the left and working to the right of your command) as one large final file. The original files remain unchanged.

Without specifying a drive letter. PIP will assume you mean the active drive. But as before, you con change that by merely adding a drive letter and colon(:). For example:

A>PIP C:BOOK=B:CHAPTER1 ,B:CHAPTER2,C:CHAPTER3,C:CHAPTER4

With this command, PIP creates a file called BOOK on drive C and fills it with CHAPTER1 from drive B, CHAPTER2 from drive B, CHAPTER3 from drive C and CHAPTER4 from drive C.

Also, since PIP first makes <X temporary file ($$$) for the actual copying, you can even use the destination file as one of the source files:

A>PIP BOOK=BOOK,CHAPTER5,CHAPTER6

The new BOOK tile you are creating will first contain the old file BOOK, which has the first four chapters from the previous joining, and the two newest chapters you've written. That makes PIP extremely flexible. You can join any files together- including the file that is to be updated.

Joining non-text files

Joining textual files together is easy tor PIP; that's because every text file ends with a special end-of-text code (comparable to CONTROL and Z keys). PIP merely copies a file until it sees the code then goes to the next file you've specified. The problem, however, is when you wish to join raw data files not in textual form or files with a HEX file type.

When joining raw data tiles, PIP should not stop it if sees the end-of-text code. Reason: in this case, the code just happens to be a piece of raw data. It PIP stops, the rest of the file will not be copied.
The cure? Tell PIP the file you are joining is an 'object' tile and it should not stop copying until the file's end. To do that you use PIP's option for data copying: the letter 0 (for object) enclosed by brackets. For instance:

A>PIP NEW=DATAl(O),DATA2(0),DATA3(0)

will join DATA1, DATA2, and DATA3 together in the NEW tile. The copying will stop only after DATA3 has been completed.

A HEX type tile is used (by the LOAD utility) to re-create an executable (COM) program file, and it gives PIP other trouble. This file does contain the end-of-text character, but it also has another built-in code to signal the program end (:00). It you join two or more HEX files together. this program end code will be included in every file. It you later try to use the LOAD utility (to re-create one large COM program file) it will simply stop when it sees the end of the first program. The other tiles you've joined will not be used.

The fix? Right- another option: I (for ignore) enclosed by brackets (I). This tells PIP to ignore the program end it is presently joining. For example:

A>PIP NEW.HEX=FILE1.HEX(I),FILE2.HEX(I),FILE3.HEX

As the NEW file is created, PIP will ignore the program end of FILE 1 ond FILE2 - only the program end on FILE3 will be copied. Also, when the I option is used, PIP knows you are using a HEX file and will automatically check the data being transferred for accuracy. It something is wrong in any of the files, PIP will tell you.

If you like the idea of having the computer check its own data, you may use the last option, H, tor checking any HEX format data.
The data in a HEX format (short tor 'hexadecimal' the number system used by your computer) is saved in rows. To check tor accuracy, all the numbers in each row of data are added together and must equal the value of the last number in the row. This is called the Intel Checksum format (developed by the Intel Corporation).

Naturally, it any data was copied incorrectly, the total value of the row will not equal the last number - and the error will be reported to you by PIP. For instance:

A>PIP FILE.HEX=RDR:(H)

As a program from the reader device (RDR:) is copied to a disc file called FILE.HEX in the default drive (A), the data is checked tor the proper sum.

Special devices

Sometimes external peripherals need a bit more than just raw data supplied by PIP. They may need additional codes to make the data more usable.

You can easily make these additions by using PIP's special device names. There are five special device names that can be used exactly as the standard device names (CON:, LST:, RDR:, and PUN:) explained earlier, but each has a particular effect on the final data.

The first three, 'EOF:', 'NUL:', and 'PRN:' are dedicated to do one job. The remaining two, 'INP:', and 'OUT:', tell PIP to get its data from a program you must supply. Here are the first three special device codes and what they do:

DEVICE	RESULT
EOF:	End-of-File- This sends the end-of-file character to the receiving device or tile. This is normally sent automatically by PIP when copying text tiles, but this command allows you to send it manually tor special tiles.

EXAMPLE: A>PIP PUN:=NEW,EOF:

A file called NEW on the default drive (A) will first be sent to the Punch (PUN:). Then the tile will be terminated with the standard end-of-file character (EOF:).

NUL: Null- This sends 40 null codes (0) to a receiving device. This is normally used to produce leaders or trailers at the beginning or end of magnetic or paper tape. (It merely produces an empty area of tape making it easier to load into the machine.)

EXAMPLE: A>PIP PUN:=NUL:C:XMAS.LST,NUL:

The Punch device (PUN:) will first receive 40 nulls as a tape leader, followed by your Christmas card list (XMAS.LST) from a disc in drive C. Then another 40 nulls are added to end the copy.

PRN: Print- data is sent to the print device as it would be if you used LST: except that tab characters in the text (such as at the beginning of a paragraph) create 8 spaces, all lines are numbered, and, after 60 lines, the printer is automatically advanced to the top of the next form (so it doesn't print over the paper tear perforations.)

EXAMPLE: A>PIP PRN:=DATA.ASM

The DATAASM file from the default disc drive (A) will be copied to the printer. The printed information will be formatted neatly on the paper and each line will be numbered tor future reference. (Some editor programs, such as ED supplied with CP/M, allow you to instantly move to a particular line in your text to make corrections.) Using the PRN: special device is comparable to using the LST: standard device name with an option of (T8NP).

The remaining two special devices, INP: and OUT:, instruct PIP to use an external program for processing the information. The program can do anything you'd like to the data, but there's a catch – you must write it. Since the program must be written to produce machine code, these two devices are reserved for professional programming uses. But here's a quick look at what they do:

INP: Input data

EXAMPLE: A>PIP FIXED=INP:

A file called FIXED is created on the default drive (A) and is filled with data from your program.

OUT: Output data

EXAMPLE: A>PIP OUT:=CON:

Each character you type on the console (CON:) is sent to your program.

7.4 STAT

7.4.1 Checking System Status

You want to know how much space is left on a disc? That's no problem with the STAT utility. It's an extremely flexible program that keeps you up-to-date on the status of your computer system by performing five basic jobs:

- Display remaining disc space
- Display the size of a tile
- Set disc and tile status
- Check peripheral assignments
- Set peripheral assignments

STAT will perform these tasks based on your entry on the command line. And, once the job is done, it automatically returns you to CP/M. Here's how they all work:

Displaying remaining disc space

One of STAT's most helpful functions is its ability to display the amount of space left on a disc. To do it type

A>STAT

Depending on the disc presently in your system, you'll see

A>STAT
A: RIW, Space: 124k

A>

Since no drive is specified, all discs presently in use will be displayed. Here, only the default drive (A) is in use and is shown on the left of the display as A:.
The R/W indicates the disc is set to read and write data. It the disc is protected against accidental erasure (with a STAT command you'll see shortly), the display will be R/O for read-only. And finally, the 124k shows there is room tor another 124,000 pieces (bytes) of data on the disc.

To check the remaining space on a disc in another drive, merely add the drive-letter specification to your command:

A>STAT C:

With a drive specified, however, only the remaining space will be shown. The read/write status will not be displayed.

Displaying file size

You can check the size of a disc tile by including its name on the command line as in

A>STAT PIP.COM

With this, you have instructed STAT to show the size of the PIP.COM file. The result:

```
A>STAT PIP.COM
Recs      Bytes   Ext      Ace
  64        8k     1       R/O  A:PIP.COM
Bytes Remaining On A: 124k

A>
```

You not only see the number of bytes taken up by the program (8k) but also the number of Records (64) and Extents (1l) it uses on the disc, and its Access mode (R/O). As a check, the drive letter (A:) and filename are displayed; as a bonus, the remaining disc space is also shown.

Records and extents are areas of the disc assigned to a file. Instead of saving a file as one long continuous piece of data, CP/M breaks it into small sections; each section is a 'record' of data. (A record holds 128 bytes of the file.) Up to 128 records are grouped together and called an 'extent.' If more records are needed to save your file, more extents are used.

By saving a file in sections, CP/M squeezes more onto a disc: If it can't find one large empty space to hold your program, it merely uses a few smaller ones.

The Access (Ace) tells you this file is set for read-only (R/O) use, so it can't be accidentally erased. If the file is not protected, the Access is 'R/WI' for read and write operation. (The Access mode is set using a STAT command you'll see in just a bit.)

You may check a file on another disc by adding the disc-drive specification:

```
A>STAT C:PIP.COM
```

The display you see with this command will be of the PIP.COM file on drive C.

Also, you may check the size of many files at once by using the asterisks or question marks in the filename:

```
A>STAT * .TXT
```

will display all files with a TXT file type. To be helpful STAT lists them all in alphabetical order:

```
A>STAT *·TXT
Recs      Bytes    Ext      Acc
  16        2k      1       R/W A:CHAP-A. TXT
   8        1k      1       R/W A:CHAP-B.TXT
  64        8k      1       R/W A:CHAP-C.TXT
  32        4k      1       R/W A:CHAP-D.TXT
  64        8k      1       R/W A:CHAP-E.TXT

A>
```

Long displays will continue to the bottom of your screen and move up (scroll) one line at a time until finished. You can temporarily stop and start the display by pressing CONTROL and alternately pressing the S key.
Although the size of any file can be checked this way with STAT, the result you see with some, called 'random access' files, may be misleading. These files have been placed at specific locations

on the disc under the direction of an outside program. Records and extents are assigned by CP/M, but each may not be filled with data. But you can check these special files with one addition to the STAT command:

A>STAT PIP.COM $S

The $S in your command will ask STAT to also show the actual size of a random access file.

Setting disc and file status

Until now, STAT has been passive - merely displaying the status of files. But with more commands, it can also do some work for you: prevent discs or files from being accidentally erased, for instance. To see what commands are available to you, type

A>STATVAL:

and the result is a display of nine lines you can use for reference:

STAT VAL:
Temp R/0 Disk : d:=R/O
Set Indicator : d:filename.typ $R/O $R/W $SYS $DIR
Disk Status : DSK : d:DSK
User Status : USR
Iobyte Assign:
CON: = TTY: CRT: BAT: UC1:
RDR: = TTY: PTR: UR1: UR2:
PUN: = TTY: PTP: UP1: UP2:
LST: = TTY: CRT: LPT: UL1:

Look confusing? It is at first glance, but each line is showing you the keywords you need to know to make STAT go to work. Here they are one at a time:

Line 1: Temp R/0 Disk : d:=R/O
Purpose: To protect your disc

With this you can protect an entire disc from accidental erasure. The command sets the disc for read-only operation - you can read data or run programs on the disc, but new files cannot be created and old ones cannot be erased. You use the disc-drive specification (d:) set equal to the letters R/O, as in

A>STAT A:=R/O

To set the disc in drive A to read only. The R/O status is temporary, however, remaining only until CONTROL and C keys are pressed or CP/M is re-entered into your computer.

Line 2: Set Indicator : d:filename.typ $R/O $R/W $SYS $DIR
Purpose: To protect a file or create a system file.

You can set the status of individual files in various ways with this command. To use it, the file drive (d:) and filename (filename.typ) are followed by a space, dollar sign ($), and one of the four

indicators: R/O, R/W, SYS, or DIR.

Each indicator sets the file for a particular use. You may also use asterisks and question marks in the filename to change the status of many files at once. The drive specification is optional. If not present the default drive is used. Here's how each indicator is used and what each does:

To protect a file against accidental erasure, type
A>STAT B:EXAMPLE.TXT $R/O

The file EXAMPLE.TXT on drive B will be read-only. You can use it but it cannot be erased or changed. This is permanent unless you use the next command:

A>STAT B:EXAMPLE.TXT $R/W

Now the same file will be reset to read-and-write status. It can now be erased or changed.

To make a file a system file, type

A>STAT * .COM $SYS

The SYS indicator sets the file for system use. It can be used, but it does not clutter the directory listing (DIR command) and it cannot be copied using PIP (unless the (R) option is used). This is helpful on utility programs, and, as shown above, all programs (files with a COM file type) on default-drive A can be set at once by using the asterisk in the file name.

To change a system file back to a standard file, type

A>STAT FORMAT.COM $DIR

The DIR in your command sets the file back to normal use. It will be displayed in the directory and it may be copied.

Line 3: Disk Status : DSK : d:DSK
Purpose: To check disc type

If you're curious about how data is saved on your disc, type the first form of this command:

A>STAT DSK:

and you'll see a rather lengthy display describing the characteristics of the disc in the default drive (A). You may check another drive by adding a drive specification:

A>STAT C:DSK:

In this, the disc in drive C will be shown. The display of numbers you'll see will depend upon the type of disc (single or double density, for example) being checked. The density can only be changed with the FORMAT utility program described earlier.

Line 4: User Status : USR
Purpose: To check users on disc

If you share your disc with another User (as described in the USER area of the CP/M Built-in Command section), then this next STAT command will be helpful:

A>STAT USR:

will request a display of User numbers presently storing data on the disc. Result:

A>
A>STAT USR:

Active User 0
Active Files: 012

You are the Active User (presently 0), but files are also present for User 1 and User 2.

Checking peripheral assignments

Obviously, for your computer to be useful it must communicate with external peripheral gear. It has to accept your commands from the keyboard, for example, and display the results of its work on your screen. But to do that it must also know where to look for the information it needs and where to send it when finished. Here's how you tell it:

The final five lines in the STAT VAL: display refer to the peripheral devices (video console, printer, etc.) attached to your system. The display you see is

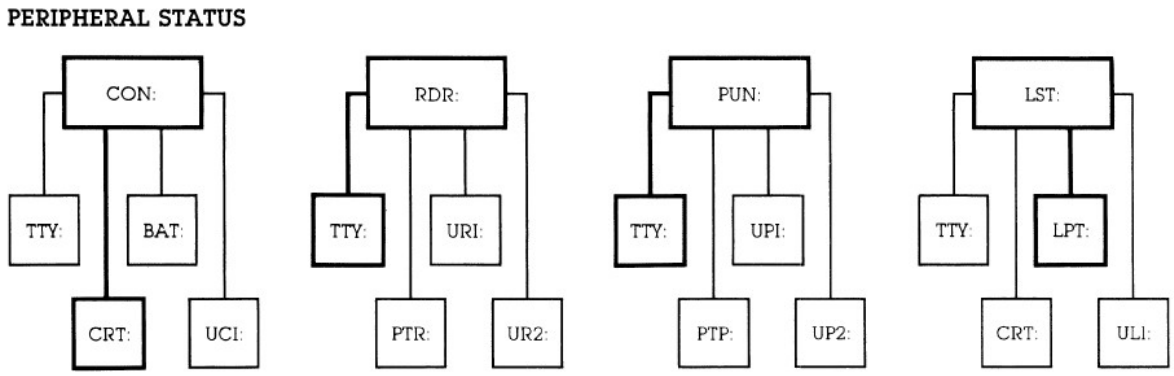
Iobyte Assign:
CON: = TTY: CRT: BAT: UC1:
RDR: = TTY: PTR: UR1: UR2:
PUN: = TTY: PTP: UP1: UP2:
LST: = TTY: CRT: LPT: UL1:

The CON: (Console), RDR: (Reader), PUN: (Punch), and LST: (List) down the left side of the display are the four main categories CP/M uses to input or output data.
As explained earlier in the PIP utility section, CP/M uses a specific category to send or receive data depending upon the job to be done. To receive commands and display results to you, it uses the console (CON:) category. To communicate with miscellaneous gear, it uses the Reader (RDR:) category to receive data and the Punch (PUN:) to send data out. And to print data, it uses the List (LST:) category.

But to allow greater flexibility, each category can be assigned to any one of four actual devices connected to your system. These devices are represented by the three letter abbreviations to the right of each category you see in the display. Here's what those abbreviations stand for:

BAT: Batch (RDR: is input; LST: is output)
CRT: Cathode Ray Tube (video console/keyboard)
LPT: Line Printer
PTP: Paper Tape Punch
PTR: Paper Tape Reader
TTY: Teletype
UC1: User Console 1

ULl: User Line Printer 1
UP1: User Punch 1
UP2: User Punch 2
UR1: User Reader 1
UR2: User Reader 2



These three-letter names are only suggested peripherals. They merely represent a particular input/output connection to your computer. CP/M acts as a traffic cop; first directing data to one of the main categories for a particular job, and then to one of the four connections for a specific peripheral. (Your computer technical manual will tell you which connection- called a 'port'- is for which device name.)

Suppose, for example, you ask to have a letter printed. CP/M will then send each character to the list (LST:) category. But from there, it can go to any of the four possible device connections of your computer: TTY:, CRT:, LPT:, or UL1:. Which one will be used?

To find out which device is assigned to each category, you first use STAT in the command:

A> STAT DEV:

and you'll see a display similar to

A>
A>STAT DEV:
CON: is TTY:
RDR: is UR2:
PUN: is UP2:
LST: is LPT:

A>

This display shows that all Console commands (CON:) must come from the video display/keyboard (CRT:). The Reader (RDR:) and Punch (PUN:) are assigned to the teletype (TTY:) connector on your computer. And the letter you asked to be printed will be routed through the LST: category to the Line Printer (LPT:) connection of your computer. If your printer is attached to the LPT: connector, it will begin to work whenever data is sent to the LST: category.

All four categories are automatically assigned a device name when you first enter CP/M. But

now watch what you can do by changing the assignments:

Changing peripheral assignments

Although CP/M automatically sets peripheral assignments each time it is entered in your computer, you can change them at any time with STAT. To do it you merely set the category equal to the device name as in

A>STAT LST: =CRT:

With this command, for example, information sent to the list (LST:) category for printing will be routed to your video screen (CRT:). You could use this command to display the exact contents and format of a document before it's permanently placed on paper. If it all looks right type

A>STAT LST:=LPT:

and now when you ask to print the document is sent through the list category to the printer at the LPT: connector.

You have two printers? Great. Attach the fast one to the LPT: connector and the slower, better printing model to the UL l: connector. Give the command

A>STAT LST:=LPT:

and you may print a rough draft on the high-speed printer. But for final work, change the assignment with

A>STAT LST:=UL1:

and when you ask to have the document printed again, it is now typed on the slower, letter-quality printer.

If you own a modem for using data over the phone line, attach it to the TTY: connector on your computer. Then type

A>STAT LST:=TTY:

and instead of sending the document to your printer, it goes through the modem to a friend's computer. Or type

A>STAT CON:=TTY:

and all console (CON:) commands and CP/M displays will be routed through the modem- your friend can now control your system remotely.
Remember that data can also be sent or received from peripherals with the PIP utility program. (See the section on PIP for more details.) But now that you know these other device names, you should also know that PIP can use them in its command line, too. For example:

A>PIP UL1:=FILENAME.TYP

will send a disc file directly to the slow-speed printer attached to the UL1: connector.

Automating your system with STAT

There is a device name that does not represent a particular connection to your computer. It's the Batch (BAT:) assignment available for console use.

When the Batch assignment is selected, CP/M gets its commands from whatever is assigned to the RDR: category; and the result of its work is sent to whatever is assigned to the LST: category. A paper tape reader could supply commands, for example. And the processed work could be sent to the printer for checking later. Enjoy your coffee.

7.5 MOVCPM

Usage : MOVCPM nn *

This is the standard CP/M utility, it is included here for the sake of completeness. The command builds an nn K version of CP/M 2.2, complete with Intel MDS BIOS, and leaves an image of the code in memory at 0980H onwards. Do NOT attempt any other variations on the MOVCPM syntax- a system crash will result.

7.5.1 WRTCPM <d:>

This command writes the CCP and BDOS section of CP/M 2.2 onto the bootstrap area of the disc in drive 'd' - it 'd' is omitted the current logged disc is used. The CCP and BDOS code is obtained from memory at 0980H; this command is intended for use immediately after a MOVCPM. Note that the BIOS and COLD-START LOADER sectors are left unaltered by the operation. On successful completion, WRTCPM issues a warning message to inform the user that warm boots from this disc could now cause a system crash.

7.5.2 WRTBIOS <d:>

This utility is used to write the custom BIOS and COLD-START LOADER section of CP/M 2.2 onto the bootstrap area of the specified disc 'd'. If 'd' is omitted the current logged disc is used. The code tor BIOS and C-START is obtained from the tile CBIOS.HEX which must be resident on the current logged disc. (CBIOS.HEX is generated by assembling the CBIOS.ASM source code tile). Note that the cold start loader code is contained within the CBIOS source at BIOS+380H.

7.5.3 Generating a new CPM system Size

This procedure makes use of the WRTCPM and WRTBIOS utilities described above. It is necessary to relocate the main CP/M body and the BIOS sections independently and then patch them together onto the disc. Follow the steps outlined below:

1. **>ED CBIOS.ASM**
   Only one line of this tile needs changing - the one that equates the value SYSIZE – Simply change the number here to the system size, in K bytes, that you require.
2. **>ASM CBIOS**
3. **>WRTBIOS <d:>**
   Write the BIOS code to the required disc.
4. **>MOVCPM nn ***
5. **>WRTCPM <d:>**

The new system has now been written and is ready to run.

NOTE: warm booting from this drive could result in a crash.

A sub tile CPMGEN.SUB is provided to automate the above process. You must use the correct

syntax when invoking this tile, ie: SUB CPMGEN nn d:- it assumes that a copy of **CBIOS.ASM** is resident on the current logged disc. Be careful when using this tile!

8 Overview of MTX-MFX CP/M 2.2 Special Features

This implementation of CP/M has been designed to support up to four logical disc drives (accessed under CP/M as B .... E- see table 2). All of the logical drives are partitions on the SD card. A fifth logical drive, F, can be created using the additional RAM provided by MFX.

Logical drives and drive types may be configured into the operating system dynamically on an as-and when-required basis by a single CP/M command.

By making drive A: a logical drive which is mapped onto one of the eight physical drives, it is possible to free drive A: from the constraints of any particular physical drive or drive type. The prom based bootstrap firmware is capable of booting from any of the four physical drives which may in turn be configured as any of the available drive types. Bootstrapping is carried out automatically at power up. Alternatively, a manual override facility may be invoked by interrupting the bootstrap sequence during its ram check phase by typing a carriage return at the console. This puts the bootstrap into a mode where it accepts a limited set of commands from the console, see the section headed '**The Bootstrap Prom**' in the FDX technical manual, reproduced as Chapter 8.11 below.

Once CP/M has been booted, it maps the logical drive A: onto the physical drive from which it was booted - for example if the system was booted from physical drive E: then this drive will be internally mapped to appear as drive A:, however it will still remain accessible as drive E:. The I/0 byte is assigned to the default value found in the boot area of the current disc and a pre-defined string of commands is automatically executed (unless aborted by a DEL character entered from the console). At this point CP/M is not aware of the existence of any drives other than A: and the drive from which it was booted - physically the same drive. Further drives may now be installed into the operating system using the RECONFIG command (RECONFIG normally forms part of the startup command string). When RECONFIG is run the bootstrap section of the current disc is searched for a default configuration table which is used to customise CP/M to the user's particular installation. Once RECONFIG has finished the customisation it prints out a disc configuration status table which lists the drive type number and description for each of the currently installed drives.

However, RECONFIG has a second mode of operation in which parameters are passed to it from the command line. These parameters allow new configurations to be made the mapping of SD card partitions to disc drives, for example, the partition pointed to by drive C can be changed from 19 to 1C to access one of the “hidden” partitions on the SD card. A STARTUP command may then be used to install the new configurations as the defaults for future cold boots. STARTUP also has a second mode of operation used to program the startup command string generated at cold boot time eg, 'STARTUP CONFIG\BASIC DEMO'; the\ symbol is used to represent carriage return.

There now follows a concise description of the utilities proved with MTX-FDX CP/M:

8.1 RECONFIG

RECONFIG <d:t, d:t . . . d:t>

This command is used to configure and re-configure CP/M for various combinations of disc/drive types. It is the users responsibility to inform the operating system of the type of drive and media for a given drive number. The command has 2 modes of operation:

1 Default mode, where no command string is specified, is used to initialise CP/M, usually at cold boot time, with the default configurations stored on the system tracks of the currently-logged disc.

These default configurations are set up initially using CONFIG, in mode 2, and STARTUP. Note that in this mode of operation RECONFIG will not re-configure any already configured drives, for example the bootstrap drive's initial configuration determined at boot time will remain unchanged. RECONFIG followed by a single space character may be used at any time to provide a listing of current drive configuration status.

2 When a valid command string is entered this will be analysed and used to generate new configurations and re-configurations. 'd' represents the drive and 't' represents the type to be allocated to that drive- t is a hex digit or digit pair, see below for a description of drive type numbers.

Once RECONFIG has completed its operation, a summary of current configuration status is listed. Among other things the current values of the FREE-SPACE and TOP-OF-AVAILABLE-MEMORY pointers are given. The FREE-SPACE pointer value increases when drives are installed as CHECK and ALLOCATION VECTORS are allocated, an error will occur if an attempt is made to increase it beyond the current value of the TOP-OF-AVAILABLE-MEMORY pointer. The TOAM pointer is set up at coldboot time by the disc handling software and points to the first byte of code or buffer needed by DISCxx. If user installed code is required in high memory, this should be slotted in between TOAM and FS, further, TOAM should be updated so as to afford protection to the new code. Note also that if an 'ENTER' string is currently active it MAY be terminated prematurely.

TABLE 1 - Config codes

TYPE	DESCRIPTION
18	8Mb SD Card, Partition 0
19	8Mb SD Card, Partition 1
1A	8Mb SD Card, Partition 2
1B	8Mb SD Card, Partition 3
1C	8Mb SD Card, Partition 4
1D	8Mb SD Card, Partition 5
1E	8Mb SD Card, Partition 6
1F	8Mb SD Card, Partition 7
50	RAM – 256kB
51	RAM – 512kB (Not supported by MFX)
52	RAM – 320kB

TABLE 2 - Logical and physical drives

LOGICAL	PHYSICAL
A	mapped to· one of the following at cold boot
B	mapped to SD Card Partition 0
C	mapped to SD Card Partition 1
D	mapped to SD Card Partition 2
E	mapped to SD Card Partition 3
[F	RAM Disc (optional, see 8.12)]

8.2 STARTUP

STARTUP <msg1\msg2\ ... msgn>

This utility is used to initialise the startup command string held on the currently-logged disc. It has three forms of syntax.

1 With no string specified the effect is to store the current I/0 byte and disc configuration status on the destination disc. These will then be used as the default parameters in future cold boots from that disc.
2 With a single space character entered, the effect is to delete the startup command string currently on the destination disc.
3 However if a string is specified then it will be written to the destination disc and used as the startup command string on future cold boots from that disc. Use the '\' symbol to generate carriage returns. Typing a single DEL character at the console will abort the execution of the command string. (Note that '\\' is interpreted as a single '\' character and any character preceded by a '^' will generate the corresponding control code - '^^' generates a single '^').

8.3 COLDBOOT

COLDBOOT <d:>

This command is used to generate a cold start from the specified drive 'd'. This drive becomes the source for all future warm boots and becomes assigned to logical drive A:. If 'd' is specified as '$', then the currently logged disc is used for the cold boot.

8.4 RCHECK

RCHECK <d:>

This command read-checks the specified drive, or if d is omitted. the currently-logged drive. A':' character is output to the console for every good sector read. If an error occurs a number between 0 (hard error) and 9 (soft error) will be printed in place of the colon. Typing any character during the read check will abort the process. The number of sectors read by the read checker is determined by the current configuration status of the drive. Note that a sector is a logical sector of 128 Bytes and is independent of the actual sector size for the disc.

8.5 BATCH

This utility performs a similar function to the CP/M transient SUBMIT except that once BATCH is invoked it prompts for CP/M command lines to be entered directly from the console for execution immediately after BATCH is terminated with the reserved command line '@END'.

8.6 ENTER

ENTER <msgl\msg2\ ... \msgn>

This utility, when used inside SUBMIT files or in BATCH command blocks, has the effect of simulating console character entry of the specified strings msg1..msgn with \ characters interpreted as carriage returns and a CR character assumed at the end of the line. Typing a DEL character at the console aborts execution of the strings. ENTER enables interactive programs such as DDT and ED to be driven by SUBMIT and BATCH files. Note that'\\' is interpreted as a single'\' character and that any character preceded by a '^' character generates the corresponding control code-'^^' generates a single '^". Note that ENTERed strings are cumulative, ie any number of ENTER commands may be used one after the other to build up a long string. ENTER makes use of free ram space in high memory; if an attempt is made to enter a string so long that the available free space is exhausted then the command will abort.

8.7 ERAQ

ERAQ file name

This utility provides an erase-with-query facility. Before erasure of the specified file(s) is performed the operator is prompted with the unambiguous file name and should respond with 'Y' to erase the file, 'N' to keep the file, or CR to abort the program. SYS files are not ignored and the operator is prompted for further confirmation in the case of R/0 files.

8.8 BAUD (Deleted)

MFX does not support the use of the Memotech RS232 Communications board and therefore the BAUD program has no function on MFX.

8.9 SIDISC (Depreciated, see RAM Disc)

Silicon Disc configuration program – not supported by MFX

8.10 SISPOOL (Deleted)

Silicon Disc print spooler program – not supported by MFX

8.11 THE BOOTSTRAP PROM

The 8kB bootstrap prom on the MTX-FDX Interface Card contains three software units: A bootstrap monitor program, ZMON.ASM, the VDU device driver, CRT.ASM and the Disc controller handler, DISC. The disc controller handler code is dependent on the particular controller board used in the system and thus determines its version number. At power up and reset time the prom is mapped-in in place of RAM from location O through to 7FFFh - leaving room for up to 32K bytes of prom. It subsequently gets control of the processor and loads itself into high RAM from 0F000h to 0FFFFh. Control is then passed to the newly initialised location 0F000h, at this point the prom is windowed out of the memory map and the Bootstrap Monitor Program signs on from RAM. After a system setup and RAM check sequence, provided no CR characters have been typed at the KBD, up to 10 attempts are made at booting, if all fail then the processor is halted. If a CR is typed during the ram check then the bootstrap monitor will enter its interactive mini-monitor mode and prompt the use for a command. The following commands are recognised:

S<addr>	Substitute memory starting at address addr. Enter spaces to skip over locations without altering them, or any valid hex digit to change the value in ram, LF causes a skip to the next block of 16 locations, CR exits the command.
R	Read hex file from SIOB at 9600 baud. Used to down-line-load programs via the serial port. (Not valid for MFX)
D	Dialogue - the computer becomes a dumb terminal device, characters entered at KBD are sent to SIOB, characters received by SIOB are sent to the display. Terminate the command by typing "'Q. (Not valid for MFX)
B <dnn>	The boot command may be used without parameters to simulate a poweron boot sequence. If parameters are given then d is used as the physical drive number B .. F and nn as the type code for the drive.
G<addr>	Begins execution at addr.

Bootstrap Prom Version 03

CONTROLLER: MFX01
NUMBER OF DRIVES: up to 4
PROM SIZE: 8K bytes
CONFIG CODES SUPPORTED: 18,19,1A,1B,1C,1D,1E,1F,50,52
TOAM VALUE: 0F000h

8.12 RAM Disc

In CP/M mode, MFX can use the additional memory in the SRAM chip to optionally add a single RAM Disc to the system. The RAM disc is a Type 51 disc with 360kB of available storage space.

For a system with only legacy floppy disk drives, such as the original Memotech FDX, a Silicon Disc (or RAM disk) potentially offered a huge speed increase over the use of floppy disks. Data could be written and read to and from memory much faster than physical disks. With the solid state “discs” used in CF/SD based systems such as CFX, MFX and REMEMOrizer, the speed benefits are much reduced RAM disc support is included with MFX for completeness, any potential benefits are very much down to an individual user’s usage case and personal preferences.

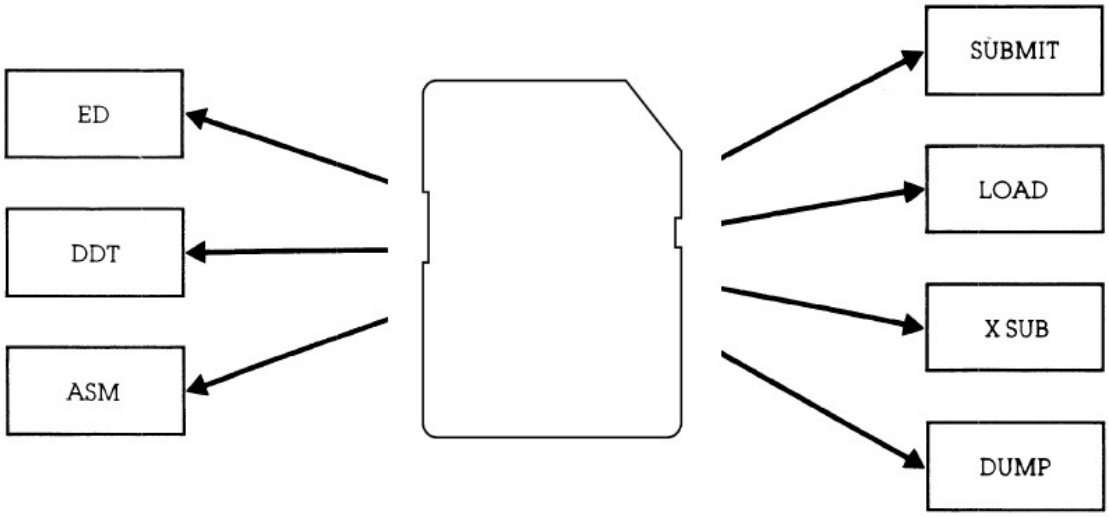
Since the SRAM has no battery backup, the contents of the RAM disk will be lost when the system is powered off. The behaviour is the RAM disk on system reset is dependent on the configuration of the ROMS jumper made during installation of the MFX PCB – see 2.1.1

Depending on the ROMS jumper selection, the RAM disc contents may, or may not, be preserved through a system reset. If the RAM disc contents are preserved and the appropriate system files present, the CP/M may be rebooted from the RAM disc.

To make best use of the RAM disc, the user should copy their preferred files to it when CP/M is loaded. The usual way of doing this is through invocation of RAMDISC.SUB which should be edited as required to put the desired files on the RAM disc. This task needs to be done after every power cycle – though, not necessarily after a reset as just described.

If you make any changes to files on the RAM disc, for example, if you have edited documents in NewWord, remember to save a copy to one of the emulated disc drives on the SD card (drives B: to E:).

9 Programming Utilities



The balance of the utility programs usually included with CP/M are provided for programmers knowledgeable in 8080 Assembly Language. The programs are:

ED	**ASM**	**LOAD**	**DUMP**
DDT	**SUBMIT**	**XSUB**	

The Assembly Language is just one of many languages used to create programs. (BASIC, FORTRAN, and COBOL, for example, are others). The difference, however, is that Assembly Language deals more closely with the direct operation of the microprocessor than the others. The result is a very compact, fast-running program. Its drawback, however, is that it is a more difficult language to learn - but there are a variety of excellent books available.

If you have not mastered Assembly Language yet, a complete description of how these programs work is meaningless. And if you are an experienced programmer, the CP/M manuals will supply you with thousands of words on the subject. So, to be logical, only the order in which these utilities are generally used and a brief description of what they do will be given here:

9.1 ED- Editor

Format: A>ED FILENAME.ASM

This is a line-orientated text-editing program used for creating the program source. It is here where the idea for your program is originally entered on separate lines. All lines are numbered - you must tell the editor which line and where in that line you'd like to enter or change a character.

To do that, ED has two modes of operation: the 'insert' mode for entering text and the 'command' mode for changing it. Special keys are used to switch from one mode to another and to give ED its instructions.

When complete, the source program you write is saved on disc under the filename you originally specified in the command line. The file name is up to you; the file type must match the

requirements of the Assembler you'll be using. To use the standard CP/M Assembler described next for example, the file type must be ASM.

9.2 ASM - Assembler

Format: A>ASM FILENAME.123

The ASM program converts the English-like words in your program (called mnemonics) into the 8080 microprocessor codes used by the computer. Your program must be on disc with an 'ASM' file type, but the file name may be anything. Using your source file, the Assembler then creates (optionally) two other disc files:

The first file contains the computer codes that represent your program in a form that may be tested for accuracy (Intel checksum). The file name is the same as your program, but the file type is automatically set to 'HEX.' This file will eventually be used with the LOAD utility (described later) to create a final executable version of your program.

The second file is a duplicate of your program except all the computer codes are added. Its file name is also - the same as your program, but the file type is set to 'PRN.' The PRN file is for your use - print it out and you'll see the assembled version of your program.

The three file-type positions (FILENAME. 123) used when ASM is first initiated specify where your original file is located and where the HEX and PRN files should go if desired. If left empty, the default drive is used - it must contain your source program and it will be used to hold both the HEX and PRN files after assembly.

But by adding a drive letter, or the letters X or Z, you may give the Assembler special instructions:

Position 1

A drive letter specification (AB,C, etc.) here will tell the Assembler the location of your source program. This allows you to have your program on any drive. The letters X and Z are not used in this position.

Position 2

A drive letter specification here indicates the drive on which you'd like the HEX file placed. If the letter Z is used, the file will not be created. (This saves time during the development of a program when you may only wish to see possible assembly errors.) The letter X is not used in this position.

Position 3

A drive letter specification here tells the Assembler where to place the PRN file. If the letter Z is used, it will not be created. If the letter X is used, it will be sent to the console instead of a disc drive. With this, you get to see your program being developed - line by line - as it is assembled.

During the assembly, the console will display any errors (heaven forbid) you've made. Your

program must be then corrected with the editor and re-assembled. Once all errors are removed, there is just one more step before it may be run in the computer - and that's to use the LOAD utility.

9.3 LOAD - Loader

Format: A>LOAD FILENAME

The LOAD utility converts the HEX file created by ASM into a directly executable command file and saves it on disk. The file name will be the one you originally chose; the file type will be automatically set to COM.

Once your program has been loaded, and its COM file created, you may use it by merely typing its file name on the command line. If it works, you're job is finished. If it doesn't there's always help from DDT:

9.4 DDT - Dynamic Debugging Tool

Format: A>DDT FILENAME. TYP

After you've created your program and tried to run it there's a chance (usually a good one) that it won't do precisely what you had intended- it has 'bugs.' But DDT has a variety of commands that allow you to examine and change the program execution - to 'debug' it.

When DDT starts, it automatically enters the file you've specified on the command line into memory. (If you hadn't specified a file, it may be entered later using one of DDT's commands.) It then gives you a dash (-) prompt to tell you it's ready.

DDT will accept single-letter commands to display, enter, or change the contents of the program. Here are the commands you can use and a brief description of what they do:

Command	Result
A	Assembly Language codes (mnemonics) may be entered
D	Display contents of memory address
F	Fill memory with data
G	Go to memory address and execute instruction
I	Input a file parameter
L	List memory contents using assembly codes
M	Move data in memory
R	Read in a disc file
S	Substitute a value in memory for another
T	Trace the program as it runs
U	Untrace the program (turn off trace)
X	Examine and optionally change microprocessor status

Besides these single letters, most commands also require additional information, such as the memory address you'd like to change. That command, for instance, would be S1000 to examine and optionally change address l1000.

Once you've made the changes you need, you may immediately try the program or return to CP/M by pressing the CONTROL and C keys. Then you may save the changed program with CP/M's built-in SAVE command.

9.5 DUMP - Display file

Format: A> DUMP FILENAME. TYP

The DUMP utility program displays the contents of the file you specify on the console. All data is shown as a 'hexadecimal' number. These numbers are used by the computer and represent textual data - conventional letters and numbers - or the machine codes used within a program.

Using DUMP to see a file called MESSAGE.TXT, for example, could produce a display somewhat like:

```
A>DUMP MESSAGE.TXT

0000 43 6F 6E 67 72 61 74 75 6C 61 74 69 6F 6E 73 2D
0010 79 6F 75 20 64 65 63 6F 64 65 64 20 74 68 69 73
A>
```

All numbers you see are in hexadecimal. The first four digits show the starting position for the 16 bytes of data (two digits per byte) contained on each line. This is the data actually contained in the file.

On long files, the display will continue to the bottom of your console and move up (scroll) line-by-line until finished, you may prematurely end the display, however, by pressing any standard key on the keyboard. Or, to stop and start the display for easier reading, press CONTROL and alternately press the S key.

Along with the DUMP.COM program file, you may also find a DUMP.ASM file. This is the program assembly listing and you may use CP/M's built-in TYPE command to display it on your console. You will see the program source and how it has been created. You may also use the Assembler and the LOAD utility to re-construct a new copy of DUMP for practice.

9.6 SUBMIT -Automatic operation

Format: SUBMIT FILENAME.SUB

Many of the jobs you do with CP/M will become routine - such as copying utilities and assembling programs. You can do it all manually, but you can also ask CP/M to do it for you by using the SUBMIT utility.

The idea is simple: using a text editor (such as ED), you create a disc file of the commands you'd like done. Then, by merely typing SUBMIT and the name of the file, each of your commands will be automatically placed on the command line for CP/M to execute. (You'll actually see each line appear on the console as SUBMIT does its work.) The file may have any file name you choose, but it must have a SUB file type.

For example, suppose you've created a file called DEMO.SUB and it contains the following command lines:

```
PIP B:=STAT.COM
PIP B:=PIP.COM
PIP B:=EDITOR.COM
DIR B:
```

Each command in your tile is on its own line and is typed just as if you were performing that function in CP/M. Now, to make it work, type SUBMIT and the name of the file you've created:

```
A>SUBMIT DEMO
```

The SUB file type is assumed and is not used in your command. In a moment, you'll see the disc activity indicator change as the process begin to work - and shortly, you'll see your commands automatically filled in on the command line and executed. Here, the disc in drive B will receive a copy of STAT.COM, PIP. CO M, and EDITOR.COM. The last command will then do a directory (DIR) of the disc in drive B. You have automated CP/M!

Using variables with SUBMIT

Suppose you routinely copy three files to drive C, but their names vary depending upon the type of disc you're creating. The SUBMIT program can still help by also accepting a variable.

Instead of using specific filenames in your submit tile, use a number preceded by a dollar($) sign. (The '$' tells SUBMIT that the number is a variable and not the name of a program). For example, suppose the 'DEMO.SUB' tile you create contains:

```
PIP C:=S1
PIP C:=$2
PIP C:=$3
DIR C:
```

With this, three files will still be copied on drive C and the directory (DIR) displayed, but the names of the files are variable. The names used by SUBMIT will be chosen from your command line when the tile is used. For instance, type

```
A>SUBMIT DEMO STAT.COM PIP.COM EDITOR.COM
```

and SUBMIT will replace the S l, $2, and $3 variables with the names you've given in the command line. The order of the names (separated by spaces). determines the number. To SUBMIT, the command appears as:

	file	variable	variable	variable	variable	etc.
A>SUBMIT	name	$1	$2	$3	$4	etc.

When DEMO.SUB is run, sit back and relax- the commands will begin filling the console screen and you'll see:

```
A>PIP C:=STAT.COM
A>PIP c:=PIP.COM
A>PIP c:~EDITOR.COM
```

A>DIR C:

Any phrase can be used as a variable: a complete filename, a separate file name and file type, or drive lette,. for example. That means you can make your submit files as flexible as you like. There is only one rule, however: the SUBMIT file must be on the disc in drive A and the disc cannot be write protected.

To keep track of things, SUBMIT writes data in its own special file on the disc in drive A

9.7 XSUB - Extended SUBMIT

Normally, SUBMIT can only enter CP/M command lines. Once a program is entered, you must take over the commands it needs. But if XSUB is placed on the first line of your SUB file, then even commands within a program will be automatically carried out.

While SUBMIT and XSUB may be used for programs other than utilities, they may not work for all. In fact because of the way most programs use CP/M, SUBMIT and XSUB are largely confined to standard utility operations.

9.8 SUB- Modified Submit

SUB file name

This is a modified version of SUBMIT which allows operation from any logged drive - not just A:, it also tolerates embedded 'A' characters. Use this command in preference to SUBMIT.

10 CP/M Control Character Summary

Keystroke	Action
CTRL-C	Aborts the current program and performs a warm start.
CTRL-F	Forces a physical carriage return, but does not send a command to CP/M.
CTRL-H	Same as back space.
CTRL-J	Line feed; terminates input at the console
CTRL-M	Same as carriage return.
CTRL-P	Echoes all console activity a t the Printer; A second CTRL-P ends printer echo.
CTRL-R	Retypes current command line, useful after using RUB or DEL key.
CTRL-S	Stops console listing temporarily; CTRL-S resumes the listing.
CTRL-U	Cancels line; displays#; cursor moves down one line and awaits a new command.
CTRL-X	Deletes all characters in command line.
CTRL-Z	String or field separator.
BACKSPACE	Moves cursor back one space; erases previous character.
DEL	Deletes characters to the left of cursor.
RETURN	Carriage return.

11 CP/M Filetypes

CP/M identifies every file by a unique file specification, which consists of a drive specification, a filename, and filetype. The filetype is an optional three character ending separated from the filename by a full stop. The filetype generally indicates a special kind of file. Common filetypes and their meanings follow.

Filetype	Indication
ASM	Assembly language source file; the CP/M assembler, ASM, assembles or translates an ASM file into machine language.
BAK	Back-up file created by text editor; the editor renames the source file with this filetype to indicate that the original file has been processed. The original file stays on disc as the back-up file, so you can refer to it.
BAS	CBASIC program source file.
COM	8080 executable file.
HEX	Program file in Intel hexadecimal format.
INT	CBASIC program intermediate language file.
IRL	Indexed REL file produced by LIB.
LIB	Used by MAC and RMAC for macro libraries. The ED R command reads files of type LIB. The ED X command writes files of type LIB. Printable file displayed on console or printer.
OVL	Program overlay file. PUI-80 compiler overlays files; you can create overlay files with LINK-80.
PLI	PL/I-80 source program filetype.
PRL	Page relocatable file, a file that does not require an absolute segment. It can be relocated in any page boundary (256 bytes).
PRN	Printable file displayable on console or printer.
REL	Relocatable file produced by RMAC and PUI-80 that can be linked by LINK-80.
SUB	Filetype required by submit file containing one or more CP/M commands. The submit program executes commands in files of TYPESUB, providing a batch execution mode for CP/M.
SYM	Symbol table file. MAC, RMAC, and LINK-80 output files of the type SYM. SID and ZSID read files of type SYM.
TEX	Source file for TEX-80, the Digital Research text formatter.
XREF	Cross reference file produced by XREF.
$$$	Temporary file.

Note : standard CP/M file types may have alternative usage on Memotech systems, including MFX. For example, a MFX .BAS file is most often a BASIC file associated with SDX BASIC.

12 CP/M Messages

? (DDT Error Message)

DDT. This message appears when DDT does not understand the assembly language instruction, the file cannot be opened, a checksum error occurred in a HEX file, or the assembler/disassemble was overlayed.

ABORTED

PIP. You stopped a PIP operation by pressing a key.

ASM Error Messages

D	Data error; data statement element cannot be placed in specified data area.
E	Expression error; expression cannot be evaluated during assembly.
L	Label error: label cannot appear in this context (might be duplicate label).
N	Not implemented: unimplemented features such as macros are trapped.
O	Overflow: expression is too complex to evaluate.
P	Phase error: label value changes on two passes through assembly.
R	Register error: the value specified as a register is incompatible with the code.
S	Syntax Error: improperly formed expression.
U	Undefined Label: label used does not exist.
V	Value error: improperly formed operand encountered in an expression.

BAD DELIMITER

STAT. Check command line for typing errors.

Bad Load

CCP error message, or S**AVE** error message.

Bdos Err On d:

Basic Disc Operating System (BOOS) Error on the designated drive. CP/M replaces d: with the drive specification of the drive where the error occurred. This message is followed by one of the four phrases in the situation described below.

Bdos Err On d: Bad Sector

This message appears when CP/M finds no disc in the drive, the disc is improperly formatted, the drive latch is open, or power to the drive is off. Check for one of these situations and try again. This message can also indicate a hardware problem or a worn or improperly formatted disc. Press CTRLC to terminate the program and return to CP/M, or

press the return key to ignore the error.

Bdos Err On d: File R/O

You tried to erase, rename, or set file attributes on a Read Only file. The file should first be set to Read-Write (R/W) with command: STAT filespec $R/W.

Bdos Err On d: R/O

The drive specified by d: has been assigned Read-Only status with the STAT command or the disc in the drive has been changed without being initialized with a CTRL-C. CP/M terminates the current program when any key is pressed.

Bdos Err On d: Select

CP/M has received a command specifying a nonexistent drive. CP/M terminates the current program as soon as you press any key. Press return key or CTRL-C to recover.

Break ':x:' at c

ED. 'x' is one of the symbols described below and c is the command letter being executed when the error occurred.

Search failure. **ED** cannot find the string specified in an **F**, **S** or **N** command.

? Unrecognized command letter c. **ED** does not recognize the indicated command letter, or an **E**, **H**, **Q** or **O** command is not alone on its command line.

O The file specified in an **R** command cannot be found.

> Buffer full. **ED** cannot put any more characters in the memory buffer, or the string specified in an **F**, **N** or **S** command is too long.

E Command aborted. A keystroke at the console aborted command execution.

F Disc or directory full. This error is followed by either the disc or directory full message. Refer to the recovery procedures listed under these messages.

CANNOT CLOSE DESTINATION FILE-(FILESPEC)

PIP. An output file cannot be closed. You should take appropriate action after checking to see if the correct disc is in the drive and that the disc is not write protected.

Cannot close, R/O
CANNOT CLOSE FILES

CP/M cannot write to the file. This usually occurs because the disc is write protected.

ASM. An output file cannot be closed. This is a fatal error that terminates ASM execution. Check to see that the disc is in the drive and that the disc is not write protected.

DDT. The disc file written by a W command cannot be closed. This is a fatal error that Terminates DDT execution. Check if the correct disc is in the drive, and that the disc is not write protected.

SUBMIT. This error can occur during SUBMIT file processing. Check if the correct system disc is in drive A. and that the disc is not write protected. The SUBMIT job can be restarted after rebooting CP/M.

CANNOT READ

PIP. PIP cannot read the specified source. Reader cannot be implemented.

CANNOT WRITE

PIP. The destination specified in the PIP command is illegal. You probably specified an input device as a destination.

Checksum error

PIP. A hex record checksum error was encountered. The hex record that produced the error must be corrected, probably by recreating the hex file.

CHECKSUM ERROR
LOAD ADDRESS hhhh
ERROR ADDRESS hhhh
BYTES READ: hhhh:

LOAD. File contains incorrect data.
Regenerate hex file from the source.

Command Buffer Overflow

SUBMIT. The SUBMIT butter allows up to 2048 characters in the input file.

Command too long

SUBMIT. A command in the SUBMIT file cannot exceed 125 characters.

CORRECT ERROR, TYPE RETURN OR CTRL-Z

PIP. A hex record checksum error was encountered during the transfer of a hex file. The hex file with the checksum error should be corrected, probably by recreating the hex file.

DESTINATION IS R/O, DELETE (Y/N):

PIP. The destination file specified in a PIP command already exists, and is Read-Only. If you type Y, the destination file is deleted before the file copy is done.

Directory full

ED. There is not enough directory space for the file being written to the destination disc. You can use the OX filespec command to erase any unnecessary files on the disc without leaving the editor.

SUBMIT. There is not enough directory space to write the $$$.SUB file on drive A, which is used for processing SUBMIT's. Erase some files or use a new disc and retry.

Disc Full

ED. There is not enough disc space for the output file. This error can occur on the W,E,H or X commands. If it occurs with, you can repeat the command, prefixing the filename with a different drive.

DISC READ ERROR - (filespec)

PIP. The input disc file specified in a PIP command cannot be read properly. This is usually the result of an unexpected end-of-file. Correct the problem in your file.

DISC WRITE ERROR - (filespec)

DDT. A disc write operation cannot be successfully performed during a W command, probably due to a full disc. You should either erase some unnecessary files or use another disc with more space.

PIP. A disc write operation cannot be successfully performed during a PIP command, probably due to a full disc. You should either erase some unnecessary files or use another disc with more space and execute PIP again.

SUBMIT. The SUBMIT program cannot write the $$$.SUB file to the disc. Erase some files or use a new disc and try again.

ERROR: BAD PARAMETER

PIP. You entered an illegal parameter in a PIP command. Retype the entry correctly.

ERROR: CANNOT OPEN SOURCE, LOAD ADDRESS hhhh

LOAD. Displayed if LOAD cannot find the specified file or if no filename is specified.

ERROR: CANNOT CLOSE FILE, LOAD ADDRESS hhhh

LOAD. Caused by an error code returned by a BDOS function call. Disc might be write protected.

ERROR: CANNOT OPEN SOURCE, LOAD ADDRESS hhhh

LOAD. Cannot find source file. Check disc directory

ERROR: DISC READ, LOAD ADDRESS hhhh

LOAD. Caused by an error code returned by a BDOS function call.

ERROR: DISC WRITE, LOAD ADDRESS hhhh

LOAD. Destination disc is full.

ERROR: INVERTED LOAD ADDRESS, LOAD ADDRESS hhhh

LOAD. The address of a record was too far away from the previously processed record. This is an internal limitation of LOAD, but it can be circumvented. Use DDT to read the hex file into memory, then use a SAVE command to store the memory image on disc.

ERROR: NO MORE DIRECTORY SPACE, LOAD ADDRESS hhhh

LOAD. Disc directory is full.

Error on line nnn message

SUBMIT. The SUBMIT program displays its messages in the format shown above, where nnn represents the line number of the SUBMIT file. Refer the message following the line number.

FILE ERROR

ED. Disc or directory is fulL and ED cannot write anything more on the disc. This is a fatal error, so make sure there is enough space on the disc to hold a second copy of the file before invoking ED.

FILE EXISTS

You asked CP/M to create or rename a file using a file specification that is already assigned to another file. Either delete the existing file or use another file specification.

REN. The new name specified is the same as a file that already exists. You cannot rename a file with the name of an existing file. If you want to replace an existing file with a newer version of the same file, either rename or erase the existing file, or use the PIP facility.

File Exists, erase it

ED. The destination filename already exists when you are placing the destination file on a different disc than the source. It should be erased, or another disc selected to receive the output file.

*** * FILE IS READ/ONLY * ***

ED. The file specified in the command to invoke ED is Read-Only. ED can read the file so that you can examine it but ED cannot change a Read-Only file.

File Not Found

CP/M cannot find the specified file. Check that you have entered the correct drive specification and that you have the correct disc in the drive.

ED. ED cannot find the specified file. Check that you have entered the correct drive specification and that you have the correct disc in the drive.

STAT. STAT cannot find the specified file. The message might appear if you omit the drive specification. Check to see if the correct disc is in the drive.

FILE NOT FOUND - (filespec)

PIP. An input file that you have specified does not exist.

Filename Required

ED. You typed the ED command without a filename. Re-enter the ED command, followed by the name of the file you want to edit or create.

hhh??=dd

DDT. The?? indicates DDT does not know how to represent the hexadecimal dd encountered at address hhhh in 8080 assembly language, dd is not an 8080 machine-instruction opcode.

Insufficient memory

DDT. There is not enough memory to load the file specified in an R orE command.

Invalid Assignment

STAT. You specified an invalid drive or file assignment or misspelled a device name. This error message may be followed by a list of the valid file assignments that can follow a filename. If an invalid drive assignment was attempted, the message 'Use: d:=RO' is displayed showing the proper syntax for drive assignments.

Invalid Control Character

SUBMIT. The only valid control characters in the SUBMIT files of type SUB are A through' Z. Note that in a SUBMIT file, the control character is represented by typing the circumflex,~, not by pressing the CTRL key.

INVALID DIGIT- (filespec)

PIP. An invalid hex digit has been encountered while reading a hex file. The hex file with the invalid hex digit should be corrected, probably by recreating the hex file.

Invalid Disc Assignment

STAT. Might appear if you follow the drive specification with anything except =R/0.

INVALID DISC SELECT

CP/M received a command line specifying a nonexistent drive or the disc in the drive is improperly formatted. CP/M terminates the current programme as soon as you press any key.

INVALID DRIVE NAME (Use A,B,C,D,E or F,G,H.I- see bootstrap rom)

SYSCOPY. SYSCOPY recognizes only the drives supported by the resident bootstrap prom, as valid destinations for system generation.

Invalid File Indicator

STAT. Appears if you do not specify RO, RW, DIR. or SYS.

INVALID FORMAT

PIP. The format of your PIP command is illegal. See the description of the PIP command.

INVALID HEX DIGIT
LOAD ADDRESS hhhh
ERROR ADDRESS hhhh
BYTES READ:
hhhh

LOAD. File contains incorrect hex digit.

INVALID MEMORY SIZE

MOVCPM. Specify a value less than 64K or your computers actual memory size.

INVALID SEPARATOR

PIP. You have placed an invalid character for a separator between two input filenames.

INVALID USER NUMBER

PIP. You have specified a user number greater than 15. User numbers are in the range 0-15. n?

USER. You specified a number greater than 15 for a user area number. For example, if you Type USER 18 <cr>, the screen displays 18?

NO DIRECTORY SPACE

ASM. The disc directory is full. Erase some files to make room for PRN and HEX files. The directory can usually hold only 64 (512 for MFX) filenames.

NO DIRECTORY SPACE- (filespec)

PIP. There is not enough directory space for the output tile. You should either erase some unnecessary files or get another disc with more directory space and execute PIP again.

NO FILE - (filespec)

DIR, **ERA**, **REN**, **PIP**. CP/M cannot find the specified file, or no files exist.

ASM. The indicated source or include file cannot be found on the indicated drive.

DDT. The file specified in an R or E command cannot be found on the disc.

NO INPUT FILE PRESENT ON DISC

DUMP. The file you requested does not exist.

No Memory

There is not enough memory available for loading the programme specified.

NO SOURCE FILE ON DISC

SYSCOPY. SYSCOPY cannot find CP/M either in CPM:xx.COM form or on the system tracks of the source disc.

NO SOURCE FILE PRESENT

ASM. The assembler cannot find the file you specified. Either you mistyped the file specification in your command line, or the filetype is not ASM.

NO SPACE

SAVE. Too many files are already on the disc or no room is left on the disc to save the information.

No SUB File Present

SUBMIT. For SUBMIT to operate properly, you must create a file with filetype of SUB. The SUB file contains usual CP/M commands. Use one command per line.

NOT A CHARACTER SOURCE

PIP. The source specified in your PIP command is illegal. You have probably specified an output device as a source.

*** * NOT DELETED * ***

PIP. PIP did not delete the file, which might have had the R/0 attribute.

NOT FOUND

PIP. PIP cannot find the specified file.

OUTPUT FILE WRITE ERROR

ASM. You specified a write protected diskette as the destination for the PRN and HEX files, or the diskette has no space left. Correct the problem before assembling your programme.

Parameter error

SUBMIT. Within the SUBMIT file or type SUB, valid parameters are SO through $9.

QUIT NOT FOUND

PIP. The string argument to a Q parameter was not found in your input file.

Read Error

TYPE. An error occurred when reading the file specified in the type command. Check the disc and try again. The STAT filespec command can diagnose trouble.

READER STOPPING

PIP. Reader operation interrupted.

Record Too Long

PIP. PIP cannot process a record longer than 128 bytes.

START NOT FOUND

PIP. The string argument to an S parameter cannot be found in the source file.

SOURCE FILE NAME ERROR

ASM. When you assemble a tile, you cannot use the wildcard characters * and? in the filename. Only one file can be assembled at a time.

SOURCE FILE READ ERROR

ASM. The assembler cannot understand the information in the file containing the assembler language programme. Portions of another file might have been written over your assembly language file, or information was not properly saved on the diskette. Use the TYPE command to locate the error. Assembly language files contain the letters, symbols, and numbers that appear on your keyboard. If your screen displays unrecognizable output or behaves strangely, you have found where computer instructions have crept into your file.

SYNCHRONIZATION ERROR

MOVCPM. The MOVCPM utility is being used with the wrong CP/M system.

'SYSTEM' FILE NOT ACCESSIBLE

You tried to access a file set SYS with the SET command.

****TOO MANY FILES ****

STAT. There is not enough memory for STAT to sort the files specified, or more than 512 files were specified.

UNEXPECTED END OF HEX FILE - (filespec)

PIP. An end-of-file was encountered before a termination hex record. The hex file without a termination record should be corrected, probably by recreating the hex file.

Unrecognized Destination

PIP. Check command line for valid destination.

USE: STAT d:= RO

STAT. An invalid STAT drive command was given. The only valid drive assignment in STAT is STAT d:=RO.

VERIFY ERROR: - (:filespec)

PIP. When copying with the V option, PIP found a difference when rereading the data just written and comparing it to the data in its memory buffer. Usually this indicates a failure of either the destination disc or drive.

XSUB ACTIVE

SUBMIT. XSUB Has Been Invoked.

XSUB ALREADY PRESENT

SUBMIT. XSUB is already active in memory.

Your Input?

If CP/M cannot find the command you specified, it returns the command name you entered, followed by a question mark. Check that you have typed the command name correctly, and that the command you requested exists as a .COM file on the default or specified disc.

13 CP/M BDOS Functions

NO	HEX	FUNCTION NAME	INPUT	OUTPUT
0	00H	SYS RESET	NONE	NONE
1	01H	CON INPUT	NONE	A=CHAR
2	02H	CON OUTPUT	E=CHAR	NONE
3	03H	READER INPUT	NONE	A=CHAR
4	04H	PUNCH OUTPUT	E=CHAR	NONE
5	05H	LIST OUTPUT	E=CHAR	NONE
6	06H	DIRECT CON I/0*	E=0FFH INPT	A=0 OR CHAR
			0FEH STAT	0 OR NONZERO
			CHAR OUTPT	NO VALUE
7	07H	GET I/O BYTE	NONE	A=IOBYTE
8	08H	SET I/O BYTE	E=IOBYTE	NONE
9	09H	PRINT STRING	DE=BUFFER	NONE
10	0AH	READ CON BUF	DE=BUFFER	CHARS IN BUF
11	0BH	GET CON STATUS	NONE	A=00/NONZERO
12	0CH	RTN VERSION #	NONE	HL=VERS **
13	0DH	RESET DISC SYS	NONE	NONE
14	0EH	SELECT DISC	E=DISC NUM	NONE
15	0FH	OPEN FILE	DE=FCB	A=DIR CODE
				A=FF IF NOT FOUND
16	10H	CLOSE FILE	DE=FCB	A=DIR CODE
				A=FF IF NOT FOUND
17	11H	SEARCH FOR FIRST	DE=.FCB	A=DIR CODE
				A=FF IF NOT FOUND
18	12H	SEARCH FOR NEXT	NONE	A=DIR CODE
				A=FF IF NOT FOUND
19	13H	DELETE FILE	DE=FCB	A=DIR CODE
				A=FF IF CODE NOT FOUND
20	14H	READ SEQ	DE=FCB	A=0 IF OK
				A=ERR CODE
21	15H	WRITE SEQ	DE=FCB	A=0 IF OK
				A=ERR CODE
22	16H	MAKE FILE	DE=FCB	A=DIR IF NO DIR
				SPACE
23	17H	RENAME FILE	DE=FCB	A=DIR CODE
				A=FF IF NOT FOUND
24	18H	RTN LOGIN VECT	NONE	HL=LOGIN VECT **
25	19H	RTN CUR DISC	NONE	A=CUR DISC#
26	1AH	SET DMAADDR	DE=DMA	NONE
27	1BH	GET ADDR (ALLOC)	NONE	HL=ALLOC
28	1CH	WRITE PROT DISC	NONE	NONE
29	1DH	GET R/O VECT	NONE	HL= R/O VECT **
30	1EH	SET FILE ATTRIB	DE=FCB	NONE
31	1FH	GET ADDR (DISC PARM)	NONE	HL=DPB
32	20H	SET/GET USR CODE	E=0FFH GET	USER#
			E=00FH SET	

33	21H	READ RAN	DE=FCB	A=ERR CODE
34	22H	WRITE RAN	DE=FCB	A=ERR CODE
35	23H	COMPUTE FILE SIZE	DE=FCB	R0,R1,R2
36	24H	SET RAN REC	DE=FCB	R0,R1,R2
37	25H	RESET DRIVE ***	DE=DRV VECT	A= 0
40	28H	WRITE RAN W/FILL	DE=FCB	A=ERR CODE

* NOTE THAT FUNCTION 6 MUST BE USED WITH FUNCTIONS, 1, 29 OR 11.
** NOTE THAT A= 11 AND B=H UPON RETURN.
*** DEFAULT DRIVE CANNOT BE RESET

THE PRECEEDING TABLE USED THESE ABBREVIATIONS:

ADDR = ADDRESS
ALLO =ALLOCATION
ATTRIB = ATTRIBUTE
BUF =BUFFER
CHAR = ASCII CHARACTER
CON = CONSOLE
CUR = CURRENT
DIR = DIRECTORY
DSK =DISC
ERR= ERROR
PARM = PARAMETER
PROT = PROTECT
RAN= RANDOM
REC =RECORD
RTN =RETURN
SEQ = SEQUENTIAL
STAT = STATUS
SYS =SYSTEM
USR =USER
VECT = VECTOR
VERS = VERSION

14 Commands at a Glance

Keyboard Control Keys:

CONTROL+ C	Warm boot CP/M
E	Continue on next line
H	Delete character
P	Printer on/off
R	Retype line
U	Delete line
X	Delete line and backup
BACKSPACE	Delete character
DELETE or RUBOUT	Delete and echo character

Built-in Commands:

DIR name.typ	Directory of disc
ERA name.typ	Erase file
TYPE name.typ	Type file on console
REN new.type=old.typ	Rename a file
USER n	User number set
SAVE b name.typ	Save a file

MTX-MFX Special CP/M Commands:

STARTUP	Initialises a startup command string
COLDBOOT	Generates a cold start
BAUD	Sets RS232 baud rate – not supported
RCHECK	Read checks drive
RECONFIG	Configures CP/M for disc drive variations
ENTER	Used inside submit or batch files
ERAQ	Erase with query

Standard Utility Programs:

REFORMAT	Prepare a new disc
MOVCPM	Adapt CP/M to memory size
PIP	Peripheral Interchange Program
STAT	Display and set system status
SYSCOPY	System generator to copy CP/M

Programmer utilities:

ASM	Assembler
DUMP	Dump HEX code
DDT	Debugger
ED	Editor
LOAD	Load HEX file
SUBMIT	Submit work
XSUB	Extend SUBMIT
SUB	Modified SUBMIT

Copying a program with PIP:

PIP d:destination=d:sourcel,d:source2(options)

PIP Options:

B	Block mode	**Pn**	Page eject
Dn	Delete characters	**Qx^Z**	Quit copying
E	Echo on terminal	**R**	Read system file
F	Form-feed remove	**Sx^Z**	Start copying
Gn	Get from user	**Tn**	Tab set
H	Hex transfer	**U**	Uppercase only
I	Ignore end record	**V**	Verify data
L	Lowercase only	**W**	Write over R/O
N	Number lines	**Z**	Zero parity bit
O	Object file		

PIP Standard Devices:		**PIP Special Devices:**	
CON:	Console	**EOF:**	End of file
LST:	List	**NUL:**	Null
RDR:	Reader	**PRN:**	Print
PUN:	Punch		
INP:	Input		
OUT:	Output		

Displaying STATus:

STAT d:	Remaining disc space on drive d:
STAT DEV:	Present device assignments
STAT d:DSK:	Display disc parameters for drive d:
STAT NAME.TYP	Display file space
STAT VAL:	Available commands
STAT USR:	Active users

Setting STATus:

STAT NAME.TYP $R/O	Set file to Read Only
$R/W	Set file to Read or Write
$SYS	Set file for system use
$DIR	Set file for directory use
STAT d:R/O	Set disc to Read Only until warm boot

15 80 Column Board Emulation

Visual Display Software (CRT.ASM) Control Code Definitions

The visual display system built into the 80 Column Board is a powerful and complex piece of hardware, and it therefore has a correspondingly complicated set of control codes to manipulate it, these are described below. A brief account of the visual display hardware will be given in order that the control code set can be related to it.

The display hardware contains 2K x 16 bit words of memory, each of the 1920 (80 x 24) character locations has one 16 bit data word associated with it. Two character generator proms are provided, one for ALPHA characters and one for the bit-mapped GRAPHICS characters. They each contain 256 shapes, one of which is addressed by the most significant byte of the character word. The least significant byte of the character word, referred to as the attribute byte, simply controls the way in which the character byte is displayed and amongst other things selects which character generator is used. In general each bit of the attribute byte has a particular and individual function associated with it, these are summarised in the table below.

The 256 entries in the ALPHA character generator prom are split up into 3 groups: the STANDARD character set of 96 symbols, the ALTERNATE set of 96 slightly different symbols and 64 special graphics symbols (independent of the bit-mapped graphics set). The ALPHA character generator regions are given below. The GRAPHICS character generator simply contains all the 256 possible combinations of the 8 pixels available in a graphics character.

On the Memotech FDX and SDX, the effects of the bits making-up the attribute byte are different depending on whether the display is being used with a colour or monochrome TV monitor - it controls either foreground and background colour or the various monochrome attributes such as under-line, bright-up, or reverse video. On MFX, it is assumed that the system will be connected to a colour VGA monitor, therefore, the bits in the attribute byte will always generate the colour effects.

Note that the attributes are NOT mutually exclusive and may be combined at will.

Effects of Bits in the Attribute Byte.

BIT	MONOCHROME DISPLAY	**COLOUR DISPLAY**
1	Underline	Red foreground
2	No effect	Green foreground
3	Bright-up	Blue foreground
4	No effect	Red background
5	Reverse video	Green background
6	Background	Blue background
7	---------- Blink	----------
8	---------- Graphics Mode	----------

Effects of Bits in the ASCII Byte

CHARACTER GENERATOR REGIONS

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0				
0	0	0	X	X	X	X	X	0	. .	31	Special Graphics set (upper case equiv)
0	0	1	X	X	X	X	X	32	. .	63	Standard Alpha set (numerals)
0	1	0	X	X	X	X	X	64	. .	95	Standard Alpha set (upper case)
0	1	1	X	X	X	X	X	96	. .	127	Standard Alpha set (lower case)
1	0	0	X	X	X	X	X	128	. .	159	Special Graphics set (lower case equiv)
1	0	1	X	X	X	X	X	160	. .	191	Alternate Alpha set (numerals)
1	1	0	X	X	X	X	X	192	. .	223	Alternate Alpha set (upper case)
1	1	1	X	X	X	X	X	224	. .	255	Alternate Alpha set (lower case)

GRAPHICS CHARACTER CODE INTERPRETATION

DO .. D7 represent the pixels of the graphics block:

b0	b1
b2	b3
b4	b5
b6	b7

CONTROL CODE SET

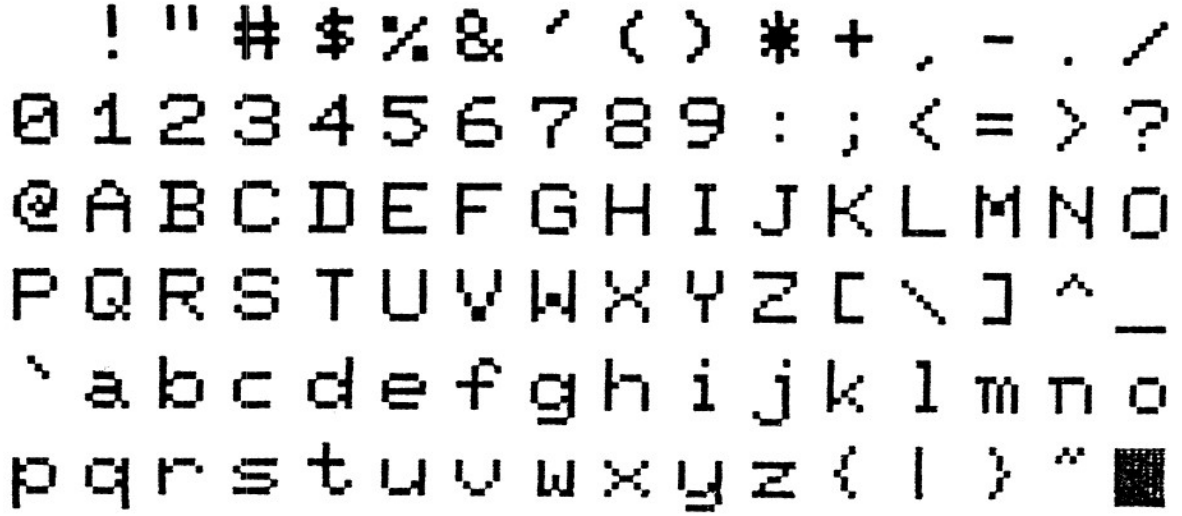
ASC	CNTL	NAME	PARAMS	DESCRIPTION
0	@	NUL		null
1	A	DOT	x,y	Plot point at X,Y in (160 x 96)
2	B	VCT	x,y,x,y	Plot vector between the 2 co-ords
3	C	CXY	x,y	Position cursor to X,Y in (80 x 24)
4	D	BKG	n	Set background colour-attrib for printed and non-printed characters: n=O to 7 corresponds to D3 .. D5 in the attribute byte specification
5	E	EOL		Erase to end of line
6	F	ATR	m	m=0 .. 255 sets up the attrib bytes
7	G	BEL		Sounds the bell
8	H	BS		Back space, cursor left.
9	I	TAB		Tabulate to next block of 8 columns
10	J	LF		LF Line feed, cursor down
11	K	UP		Cursor up
12	L	CLR		Clear screen and home cursor
13	M	CR		Carriage return, cursor to LHS
14	N	BL		Blink-on
15	O	BLO		Blink-off
16	P	BLK	*	Black foreground
17	Q	RED	*	Red foreground
18	R	GRN	*	Green foreground
19	S	YEL	*	Yellow foreground
20	T	BLU	*	Blue foreground
21	U	MAG	*	Magenta foreground
22	V	CYN	*	Cyan foreground
23	W	WHT		White foreground
24	X	INI		Initialise to standard attrib config.
25	Y	FWD		Cursor forwards
26	Z	HME	ch..	Cursor to top LHS, home position
27	^(	ESC		Escape sequence, see next table
28	^/	SCT		Scroll mode
29	^)	PGE		Page mode or ROLL
30	^^	CON		Cursor ON
31	^	CSO		Cursor OFF
			x,y	Are in offset binary. Add 32 to the required co-ords
			m,n	Must be specified in binary
			*	May be binary or ASCII
				The effect of these codes when used on a monochrome display will correspond to setting the appropriate bits in the field DO .. D2 of the attribute byte. NOTE: these codes affect printing characters only.

Printing characters are those which actually appear on the screen: Non-printing characters are generated by control codes such as CLR, EOL and ESC L graphics plots and dots are also treated as non-printing characters. Non-printing characters may be given different attributes to the printing characters if desired.

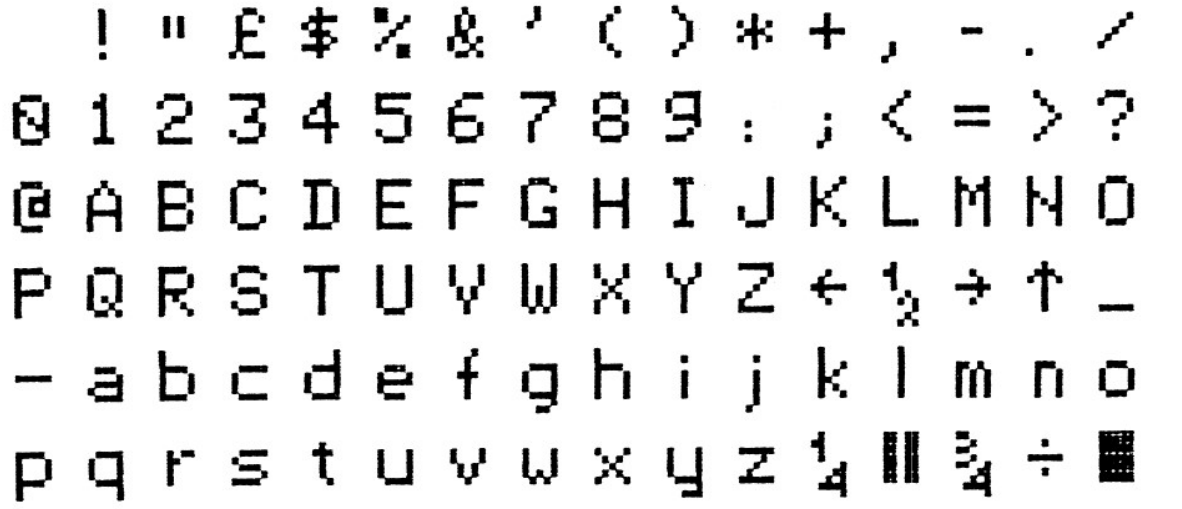
ESCAPE SEQUENCE CODES

CHR	PARAM	DESCRIPTION
S		Use standard character font
A		Use alternate character font
G		Use special graphics font
C		Scroll mode
D		Page mode
E		Cursor on
F		Cursor off
X	ch	Simulate control character ch
I		Insert a blank line at cursor line, shifting the lines below down
J		Delete the current cursor line, shifting the lines below up
W	n	Set up the write mask:
	m	n=O : allow ascii and attrib writes
	m	n= 1 : allow only ascii writes
	m	n=2 : allow only attrib writes
T	n	Set printing character attrib byte to m (0 .. 255)
U	n	Set non-printing attribute byte to m (0 .. 255)
V	n	Set both attribute bytes to m (0 .. 255)
P		Set bit n (1..8) of printing attrib byte *
N		Set bit n (1.:8) of non-printing attrib byte *
B		Set bit n (1..8) of both attribute bytes *
Y		Sets 80 x 24 character mode (MFX enhancement, see 19.3)
Z		Sets 80 x 48 character mode (MFX enhancement, see 19.3)
	n	May be in ascii or binary
	m	Must be in binary
	*	If n=O then whole byte is reset

STANDARD CHARACTER SET



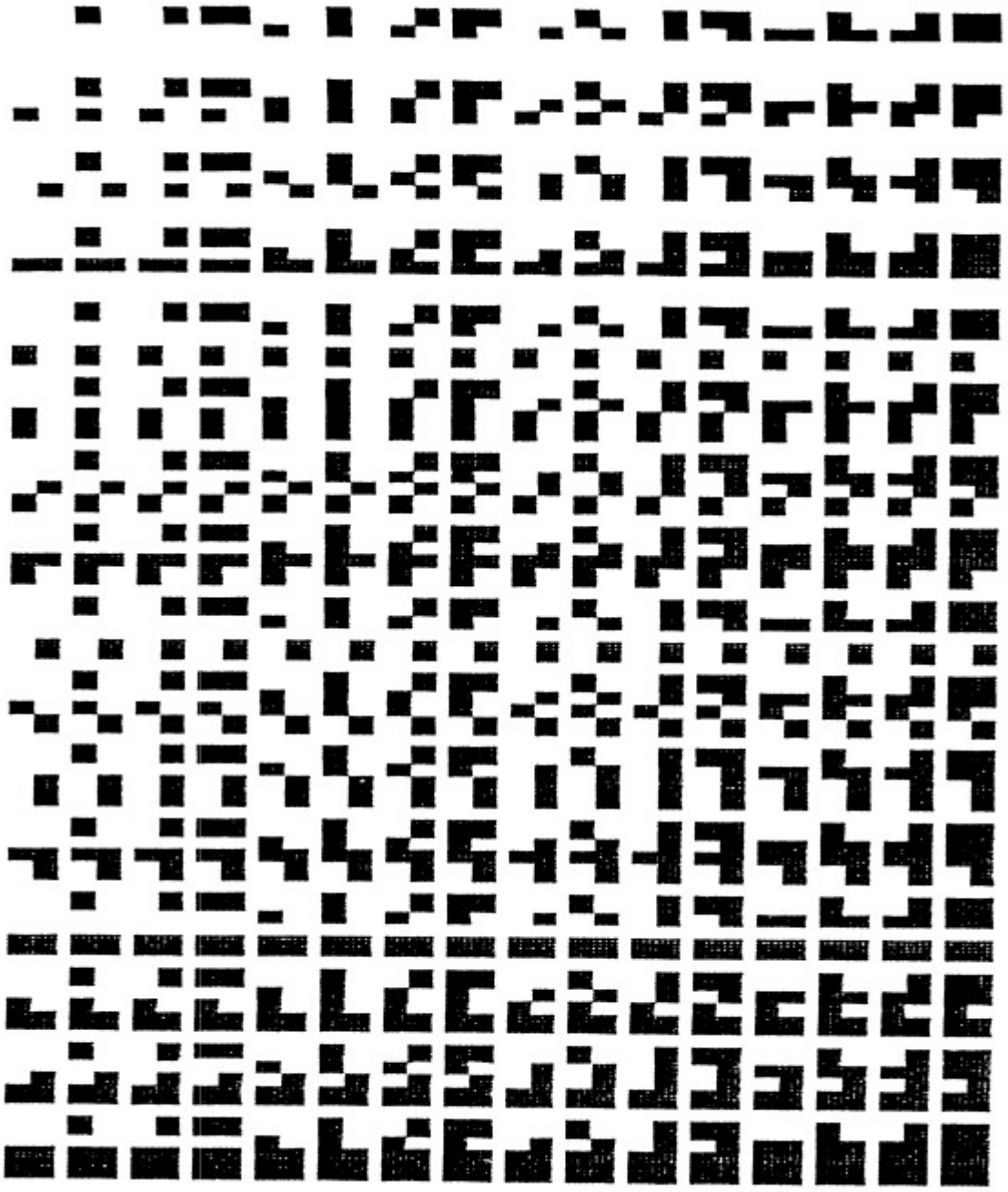
ALTERNATE CHARACTER SET



SPECIAL GRAPHICS SET

FULL ALPHA PROM SET

FULL GRAPHICS PROM SET



16 MFX SDX ROM

Pressing <**RET**> from the start-up screen enters the familiar blue screen of the BASIC environment. The auto run ROM will initialise the SD card on start-up so all the additional commands are ready from the outset.

The SDX ROM uses CP/M for all disc accesses and so retaining full compatibility with the CP/M ROM.

There is no need to manage separate memory cards for each option. Needing to reserve memory space for CP/M does reduce the amount of memory available for BASIC and will cause issues with some games that are only intended to run from tape.

On an unexpanded MTX500 the amount of memory for BASIC programs is reduced to approximately 20k.This is not an issue for MFX users as MFX supplies the extra 32kB of RAM that you'd normally get with a MTX512.

As the MFX implementation of the SDX ROM is mapped into ROM bank 5, typing "***ROM 5***" will re-initialise the SD card if required, without disturbing the program in memory.

All of the additional commands are accessed via the USER command. The full set of SDX extensions are provided, with two omissions; attempting to format a drive or copy the system tracks, will result in an error message being produced. All format actions have to be done from the CP/M side.

Like the CP/M side, drive letter "A" can be used as an alternative for the boot drive as there is no separate "A" drive.

If you've used the SDX or single disc FDX, you will find that along with the 2 commands that have been disabled there are two extra commands added.

In addition the control ROM has been extended so that it can see the first 4 partitions, and access them as drives "B" to "E", unlike the CP/M side, it's not possible to re-map the drive letter to point to the last 4 partitions. Increasing the number of partitions means that the NODE networking software will not coexist with MFX

Attempting to access drive letters above "E" will result in an error being displayed, and may need the CF re-initialising with "*ROM 5*".

The RAM drive is not available in SDX mode even if memory is available.

16.1 MFX Specific SDX USER Commands

16.1.1 HELP

USER HELP

This non-standard SDX extension will display the same crib sheet as that invoked by the “I” on the boot screen. Using this command does not disturb the program currently in memory.

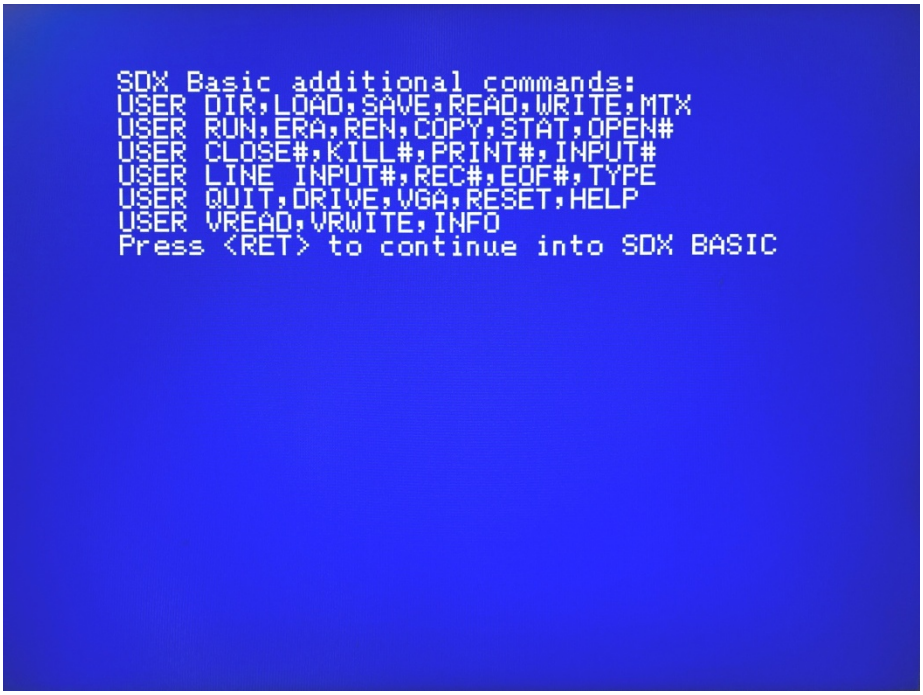


Figure 6 - MFX CP/M Boot Screen

The use of these SDX BASIC commands are broken down into functional groups and described in the following sections in this chapter

16.1	MFX Specific SDX USER Commands	HELP, INFO, VGA
16.2	SDX Basic USER Commands	SAVE, LOAD, RUN, MTX, DRIVE, RESET
16.3	MTX RAM/VRAM Read/Write	READ, WRITE, VREAD, VWRITE
16.4	Data File Handling Commands	OPEN, CLOSE, KILL, INPUT, LINE INPUT, REC, EOF
16.5	Disc Commands	DIR, ERA, STAT, TYPE, REN, COPY, QUIT

16.1.2 INFO

USER INFO

This non-standard SDX extension will display information on the status of the MFX, including the hardware and firmware version numbers and current configuration parameters, some of which may be changed by the user.

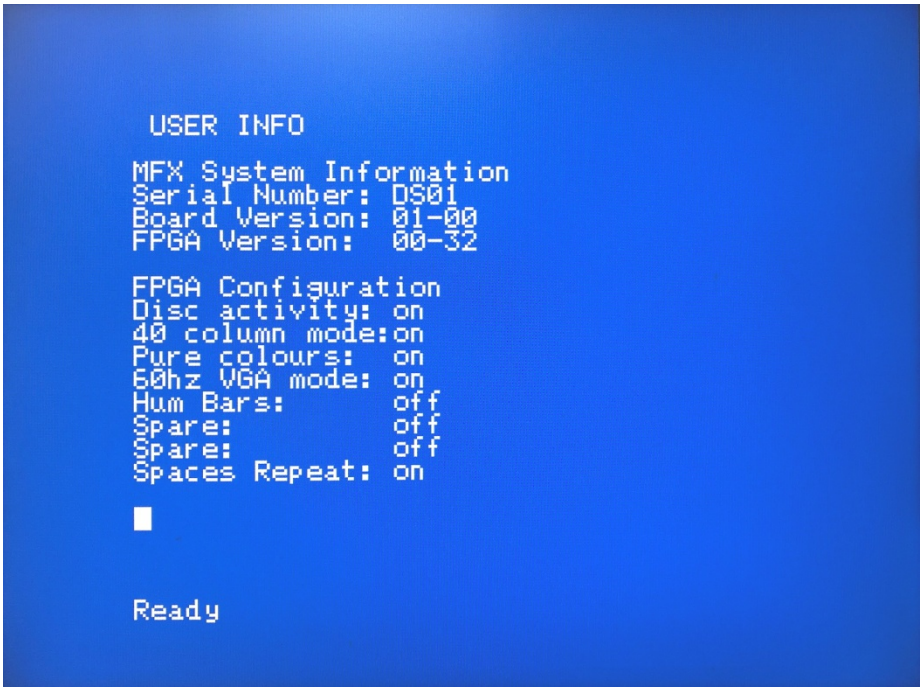


Figure 7 - MFX CP/M Boot Screen

MFX System Information

- Serial number, including the users initials and unique serial number
- PCB hardware Major-Minor version number
- FPGA firmware Major-Minor version number

FPGA Configuration

- Disc activity indicator
  Since the SD card module and FPGA development board are installed inside the MTX, it is not practical to have a physical "disc" activity indicator. This option provides an on-screen indication of SD card activity.
  In SDX mode, the screen border flashes red during disc activity.
  In CP/M mode, the cursor colour changes to red during disk activity.
  (This parameter cannot be changed at run-time, it is an FPGA compile time option.)
- 40 column mode
  The default mode for the VGA shadow output from the VDP is to use 40 columns, the same as the VDP. It is possible to switch the output to 80 columns to provide enhanced editing in SDX BASIC, display more output when listing files, etc.
  See the USER VGA command for details
- Pure colours

The VGA shadow output from the VDP can use one of two different palettes. The default mode is for the VGA output to use a vivid colour scheme (pure colours) that look better on VGA monitors but don't accurately reflect the VDP output. If you prefer the 'washed out' colours that you may remember from back in the day, start SDX in 'Retro' mode (<**Reset**><**R**>).

- 60Hz VGA mode
  As might be expected, sets the system to use a 60Hz monitor
- Hum Bars
  The default mode is for the VGA shadow output from the VDP to display a homogeneous background colour that you'd expect from a colour monitor. If you prefer to see the "hum bar" artefacts that you may remember from back in the day, start SDX in 'Retro' mode (<**Reset**><**R**>).
- (spare)
- (spare)
- Spaces Repeat
  This setting uses some custom programming to optimise the VGA display output when printing multiple repeating characters to the screen.
  (This parameter cannot be changed at run-time, it is an FPGA compile time option.)

16.1.3 VGA

USER VGAx

USER VGA1 to switch the SDX BASIC screen to 80 columns
USER VGA0 to reset the SDX BASIC screen to 40 columns

This non-standard SDX extension provides the facility for the user to switch the normal SDX BASIC 40 column screen into an 80 column mode. This significantly improves the experience of editing BASIC programs and provides an enhanced view of SD card directory listings.

Figure 8 – USER VGA1 80 Column Screen

The USER DIR command is aware of the 80 column mode and will display the file listing accordingly.

However, there is a potential issue with loading auto-run files in this mode. Program files created on an MTX without MFX, i.e., all of the .BAS, .RUN and .MTX files included on the SD card, are unaware of MFX 80 column mode, and if the screen was not returned to standard VDP 40 column mode prior to loading, graphical output would not be presented on the VGA screen, thought the VDP output would be presented as normal.

To work around this issue, the USER MTX and USER RUN commands on MFX have been modified to ensure that the 40 column/graphics screen mode is output to the VGA monitor when these commands are used.

However, this auto switching has not been implemented for USER BAS, otherwise, 80 column editing of BASIC programs would be disabled. You must ensure that you manually reset the system to 40 column mode (**USER VGA0**) before loading auto-run BASIC programs, or using the BASIC 'RUN' command to start a BASIC program which uses VDP graphics modes.

16.2 SDX Basic USER Commands

The basic commands for loading and saving programs using SDX BASIC with the executable file types, .BAS, .RUN and .MTX

16.2.1 SAVE

USER SAVE "FILENAME.EXT"

Will save to the B drive, partition 18, by default

USER SAVE "[x:] FILENAME.EXT"

Will save to the specified drive x:, as long as it's in the A-E range

Note: SDX BASIC accepts optional drive qualifiers [x:] in the range A: to E: when loading and saving programs.

File names don't have to be in upper case, however files with lower case characters won't be accessible from the CP/M ROM. File name extension is optional, the BASIC extensions will attempt to load any file when requested. Specifying the drive is optional for all the disc commands, the default is the A/B drive if the letter isn't specified.

16.2.2 LOAD

USER LOAD "[x:] FILENAME.EXT"

Will attempt to load the named file as a BASIC program, whatever the extension

Does NOT automatically reset the USER VGA mode to 0 (See USER VGA)

16.2.3 RUN

USER RUN "[x:] FILENAME.EXT"

Attempts to load and run a binary game image file, the first 4 bytes providing the start address and length of the data. Used to load a number of, mostly early Continental Software, games that have been converted using the tape to disc software that was available for the FDX/SDX or that were converted for MAGROM. Conventionally the extension used for these types of file is RUN but this does not have to be the case.

Automatically resets the USER VGA mode to 0 (See USER VGA)

16.2.4 MTX

USER MTX ”[x:] FILENAME.EXT”

Will attempt to load a MTX format tape image file, any multi part images are likely to fail. The command may enable some games that are not available in the other supported formats to load. As with RUN , the file extension for this type of file is normally MTX

Automatically resets the USER VGA mode to 0 (See USER VGA)

16.2.5 DRIVE

USER DRIVE ”x:”

Changes the logged drive to “x:” where “x:” is in the range B: to E:.
Drive “x:” becomes the default drive for SDX BASIC USER commands.

16.2.6 RESET

USER RESET

Reloads CP/M and logs onto drive B:

16.3 Commands for loading and saving raw data directly from MTX memory

16.3.1 WRITE

USER WRITE "[x:] FILENAME.EXT",<start>,<size>

Save a block of memory to the named file starting at <***start***> and <***size***> bytes long.

16.3.2 READ

USER READ "[x:] FILENAME.EXT",<start>

Attempt to read a file into memory beginning at address <start>.

16.3.3 VWRITE

USER VWRITE "[x:] FILENAME.EXT",0

Save all 16K of Video RAM to a file

16.3.4 VREAD

USER VREAD "[x:] FILENAME.EXT",0

Load up to 16K from the named file into of Video RAM

16.4 Data File Handling Commands

16.4.1 OPEN

USER OPEN#<channel no>,"[x:] FILENAME.EXT",<type>(,<reclen>)

Open a file for input/output, valid channel numbers are 1-4. <***type***> is a string type, and can be "***O***" output," ***I***" input or "R" random I/O. the quotes are required. <***reclen***> is an optional record length for random files.

An error will be reported if the file is already open.

16.4.2 CLOSE

CLOSE#<channel no>

Closes a currently open file.

16.4.3 KILL

USER KILL#<channel no>

Close and erase a currently open file. Use with care.

16.4.4 INPUT

USER INPUT#<channel no>, arguments

Will read numeric or string data from an open file, an error will occur if the variable types don't match.

16.4.5 LINE INPUT

USER LINE INPUT#<channel no>, arguments

Read an entire line into a string variable.

16.4.6 PRINT

USER PRINT#<channel no>, arguments

Send numeric or string data to an open file.

16.4.7 REC

USER REC#<channel no>,<position>

Move the file pointer in a random access file to the specified record in preparation for the next action.

16.4.8 EOF

USER EOF#<channel no>,<line no>

Will do a GOTO <***line no***> in BASIC if the end of file marker has been reached for the channel selected.

16.5 Disc Maintenance Commands

16.5.1 DIR

USER DIR

Used in the same way as the CPM command, both "***?***" and "*******" wildcards are allowed, all 4 legal partitions can be accessed.

16.5.2 ERA

USER ERA

Used in the same way as the CPM command, both "***?***" and "*******" wildcards are allowed, all 4 legal partitions can be accessed, care should be taken as inappropriate wildcards could result in the entire partition being wiped.

16.5.3 STAT

USER STAT" [x:] FILENAME.EXT"

Will display basic file info, including total size in kB, number of 128 byte records and R/W status

USER STAT"x:"

Will display basic partition info, i.e., the free space on the partition in kB

16.5.4 TYPE

USER TYPE "[x:] FILENAME.EXT"

Will read and display a file on the screen, control characters are displayed, so typing a data file could have unexpected consequences!

16.5.5 REN

USER REN "[x:] NEWNAME.EXT"=" [x:] OLDNAME.EXT"

Rename a file in the same way as CPM, except lower case is allowed.

16.5.6 COPY

USER COPY "[x:] NEWFILE.EXT"=" [x:] OLDFILE.EXT"

Will copy files on the same partition, or across partitions.

16.5.7 QUIT

USER QUIT

Equivalent of NEW, will run NCPM.COM if found, otherwise enters BASIC.

16.6 Legal, but Disabled Commands

16.6.1 FORMAT

USER FORMAT is disabled and will return an error if used.

16.6.2 SYSCOPY

USER SYSCOPY is disabled and will report an error if used.

17 Network File Transfer - WIZnet

Perhaps saving the best for last . . . . this chapter describes the networking capabilities of MFX.

The biggest differentiator between MFX and other modern day flash memory replacement "disc" systems for the MTX is the networking capability provided by MFX.

The heart of this capability is a WIZnet 810SMJ network module that uses a WIZnet W5100S Ethernet controller and interface hardware to provide direct connectivity to an IPv4 Ethernet network. It supports hardwired Internet protocols, TCP, UDP, WOL over UDP, ICMP, IGMPv1/v2, IPv4, ARP, PPPoE and up to 4 independent SOCKETs simultaneously.

This sub-system builds on the work done in developing the NFX (Network File System) prototype which proved the concept and demonstrated that a 4MHz, Z80 based, computer from the 1980s can be equipped with a TCP/IP network interface and that its performance is more than satisfactory.

Targeting the WIZnet hardware used in NFX, Andy Key created a CP/M program (NFX.COM) that allowed the MTX to run a number of simultaneous network services, including ***echo***, ***chargen*** and ***http***, and demonstrated a simple web server running on an MTX at Memofest 2018.

Having a simple web server running on an MTX was an interesting exercise, but was likely to be of very limited practical use; however, building on Andy's clever multi-threaded code, Bill Brendling developed a functional ftp server that runs on the same WIZnet hardware incorporated into MFX.

This is a "game changer" for MTX users. Until now, transferring files between the MTX and other computers has been pretty awkward, requiring the use of conversion tools such as Andy's ***cpmcbfs*** to enable the PC to read/write MTX CF/SD cards or images. Being able to connect your MTX to your local network, then use FTP tools such as ***Filezilla*** to directly transfer files from the PC to MFX's SD card is, to my mind, a huge leap forward.

17.1 MFX Networking Support Programs

As well as his ftp server, FTPD.COM, Bill also enhanced Andy's HTTP server and released his own version, HTTPD.COM. To make user configuration of the network configuration parameters easier, on-startup, Bill's programs read the required configuration from a text format configuration file, default name, NFX.CFG.

17.1.1 Configuration File Structure

The configuration file should be located on the same "disc" as the executable programs. It is a text file which can contain one to five lines

Line	Parameter	Example	Default	Comment
1	IP Address	192.168.1.32	192.168.1.123	
2	Subnet Mask	255.255.255.0	255.255.255.0	
3	Network Gateway	192.168.1.254	192.168.1.1	
4	MAC Address	00:DC:DC:4D:46:58	00:08:DC:4E:46:58	
5	WIZNet Port	90	90	90h For MFX, A0h for NFX

IP Address - Defaults to 192.168.1.123 if no configuration file found
Subnet Mask - Defaults to 255.255.255.0 if not specified.
Network Gateway - Defaults to 192.168.1.1 if not specified.
MAC Address - Defaults to 00:08:DC:4E:46:58 if not specified.
WIZnet Port base address – Defaults to 90 if not specified.

17.1.2 HTTPD

USAGE: HTTPD <logfile>

Hit <**q**> to exit the program

A simple web server, allows the MTX to serve files over the network, accessible by tools such as ***curl*** or any Web browser. If an optional log file name is specified, the programs activity will be logged to a text file with name as specified on the command line.

The application knows about a number of file types and attempts to help the remote web browser handle them accordingly

EXT	Type	Expected Browser Behaviour
HTM	text/html	Render (Display)
CSS	text/css	Render
JS	text/javascript	Render
TXT	text/plain	Render
SUB	text/plain	Render
DOC	text/plain	Render
XML	text/xml	Render
GIF	image/gif	Render
PNG	image/png	Render
JPG	image/jpeg	Render
TIF	image/tiff	Render
PDF	application/pdf	Render
COM	application/binary	Download
OVR	application/binary	Download
MTX	application/binary	Download
RUN	application/binary	Download
BAS	application/binary	Download
ZIP	application/binary	Download
other	unknown	Download

Note, the “Expected Browser Behaviour” is dependent on the browser used and their configuration. For example, on my PC, Microsoft Edge renders (displays) .TXT plain text files, whereas Google Chrome downloads them.

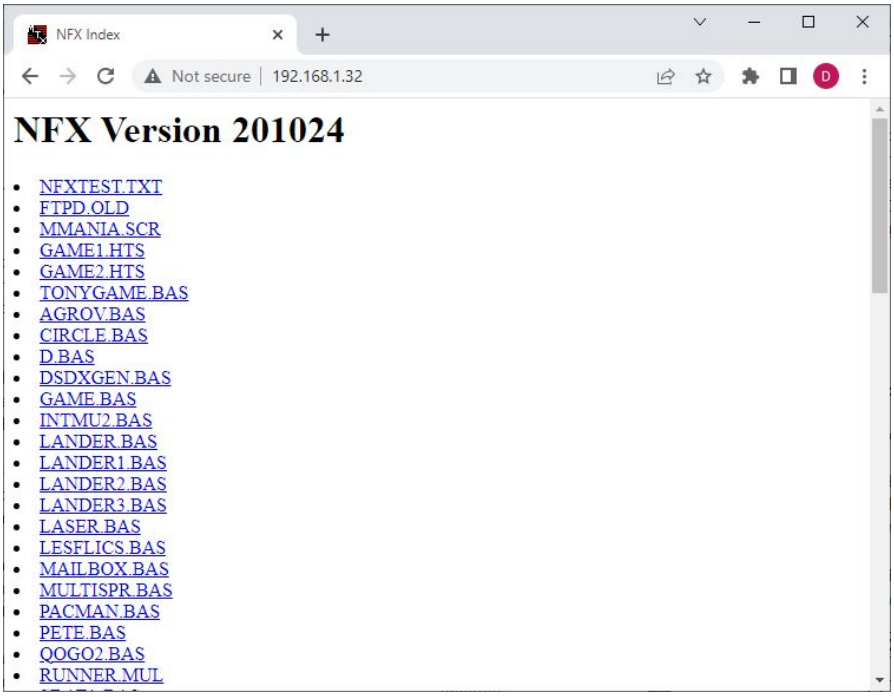


Figure 9 – Brower (Google Chrome) view of the HTTPD Web Server

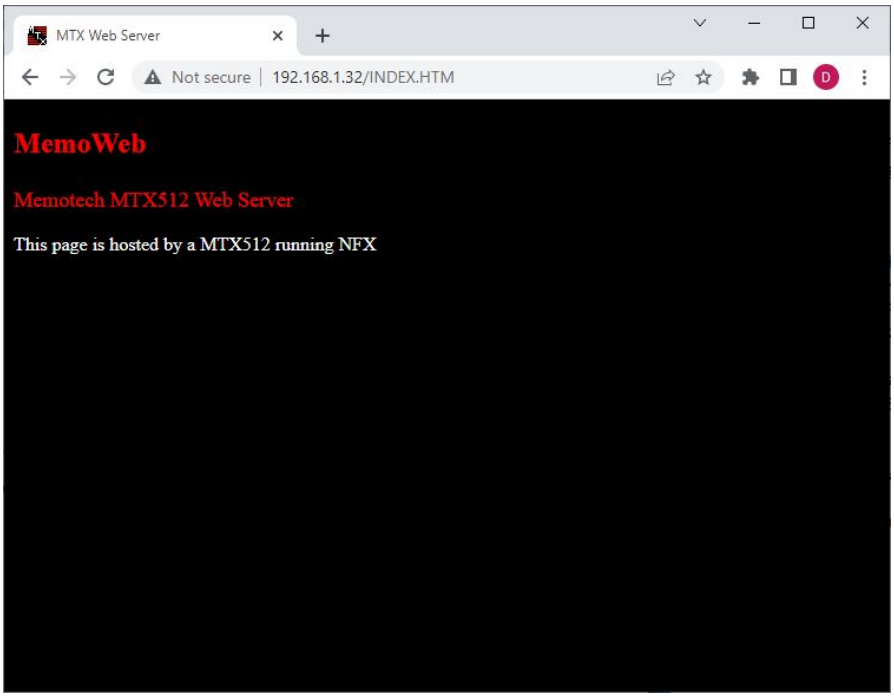


Figure 10 – Brower (Google Chrome) Rendering an .HTM file

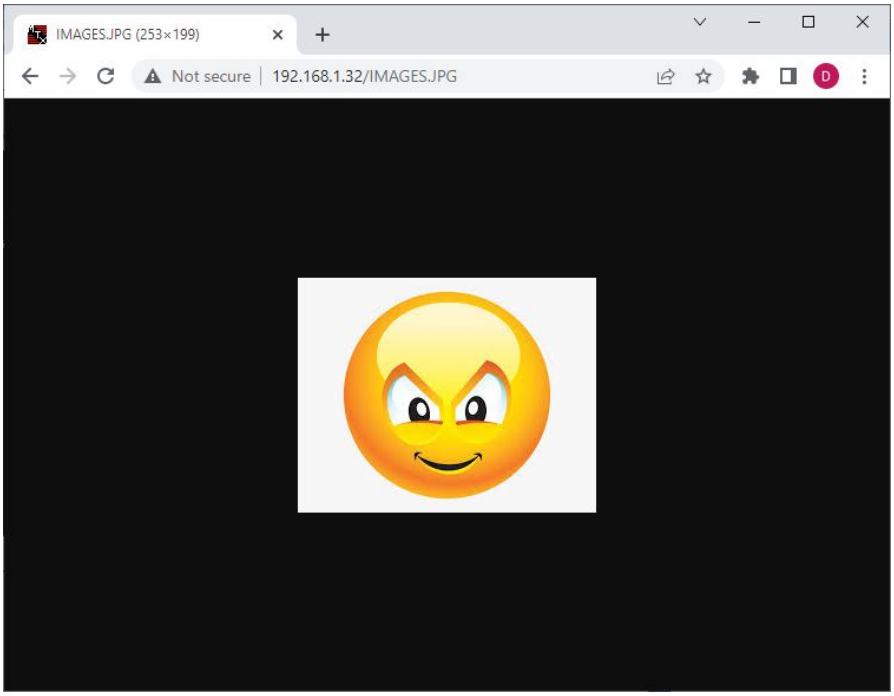


Figure 11 – Brower (Google Chrome) Rendering an .JPG file

17.1.3 FTPD

USAGE: FTPD <logfile>

Hit <**q**> to exit the program

A File Transfer Program server, allows the MTX to respond to and service requests from an FTP client. If an optional log file name is specified, the programs activity will be logged to a text file with name as specified on the command line.

You should be able to use any standard FTP client; recommended by Bill Brendling, I have had good results with Filezilla (https://filezilla-project.org/) client for Windows.

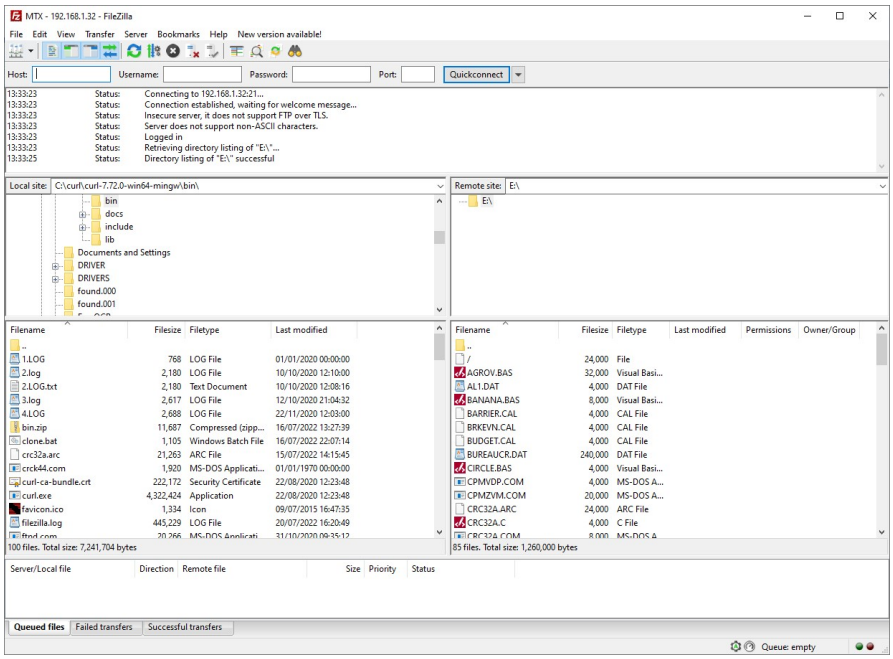


Figure 12 – FileZilla with an open connection to MFX

Screen shot of FileZilla running under Windows 10 and connected to an MTX512 with MFX hardware and FTPD running. Here you can see the right hand pane displaying the contents of MFX's E: drive, connected over TCP/IP to IP address 192.168.1.32.

As well as allowing easy file transfer between the MTX and a PC, the tool makes housekeeping on the MTX disk (SD) much simpler. File management is much easier than using native CP/M commands or CP/M add-in programs..

18 Additional CP/M Features

18.1 Start Games From CP/M

A number of MTX games are available in CP/M .COM format. When these programs are run under CP/M on original FDX/SDX hardware, the game output defaults to the 40 column VDP output. This behaviour is unchanged with MFX, however, the video output can be shadowed on the VGA monitor by switching the VGA output to shadow the VDP when the game is started.

To achieve this, a program on the SD card called ***CPMVDP.COM*** is used. If the program is run interactively and no game filename passed to it, the VGA screen will be blanked as soon as the mode is switched, so you will not be able to see any output on the screen.

To run a game .COM file, you would enter, for example, ***cpmvdp star***

where "***star***" is the COM file for Star Command

A number of games are also available in .RUN format. This format was designed for running games from SDX BASIC using the USER RUN command. Andy Key has written a program that allows most .RUN format games to be executed directly from CP/M using RUN.COM. As with the .COM format games, when invoking them from CP/M, the .RUN programs need to have the VGA VDP shadow mode enabled first.

An sub file **VDPRUN.SUB** is provided to achieve this :-

- VDP
- RUN $1.RUN

(The RUN.COM program requires that the full filename, including extension, is passed to it.)

To run the game .RUN file, you would enter, for example, ***sub vdprun alpha***

where "***alpha***" is the RUN file for Mission Alphatron

19 Bonus Features

19.1 Multi-colour sprites

The TMS9918/9929 Video Display Processor used in the MTX supported user defined sprites. There can be up to 32 monochrome sprites of either 8×8 or 16×16 pixels on screen, each sprite with its own, single colour. The illusion of multi-colour sprites can be created by stacking multiple sprites on top of each other.

The Yamaha V9938/V9958 used in MSX2 and MSX2+ was a development of the TI VDP with expanded capabilities, including the ability to display sprites with multiple colours on the same line. With the extra Video RAM available to MFX, it was possible to add multi-colour sprite capability to MFX.

As well as the option to write more colourful games from scratch, the addition of multi-colour sprite capability to MFX means that it is possible to hack existing MTX games to change the sprite attributes to display multi-colour sprites.

To demonstrate this feature, the MFX SD card includes two versions of Chris Bayne's Hawkwars game – the original (HAWKWARS.RUN" and one modified to use multi-coloured sprites (HWMC.RUN).

19.2 Additional SDX BASIC Virtual Screen Types

Two extra CRVS screen types are supported:

Type 2 virtual screens are 80x48 mode

Type 3 virtual screens are 80x24 mode

19.3 Double Height CP/M Console

The 80 column card emulation has additional RAM available to it that the original Memotech 80 column card didn't. This allows an expanded 80 x 48 character mode. Additional Escape sequences have been added to turn this feature on (Esc Z) and off (Esc X).

The 48 line mode is really handy for directory listings, allowing twice as many filenames to be visible at the same time. By default, standard CP/M application programs will not be aware of this 48 line mode, but many include a configuration program that allows custom screen parameters to be used.

An example use for this double height screen mode is editing documents with NewWord using the 80 x 48 character screen. The SD card includes a version of NewWord that has been patched to use 48 lines, invoke it as N48.COM.

19.4 System Status Information

The USER INFO commands available from SDB BASIC (see 10.1.2) provides a high level way for the user to interrogate the FPGA and determine the status of a number of MFX specific parameters. At a lower level, interrogating the status of the I/O ports directly can provide information that may not be available at the higher level.

These parameters can be read using the BASIC INP(xx) command and written, where permitted, using the BASIC OUT(xx) command – see 13.3.4 for the I/O Port mapping

I/O Port Allocation (For a full list of MFX I/O ports, see Appendix B)

Port			Function	Access
Hex	**Dec**	**Bit**		
35	53	-	Serial Number register	Read
3A	58	0	1 = Character speed up [1]	R/W
3A	58	1	(Not used)	n/a
3A	58	2	(Not used)	n/a
3A	58	3	1 = hum bars enabled if Bit 5 = 0	R/W
3A	58	4	VGA Freq. (1 = 60Hz)	R/W
3A	58	5	Colour Palette (0 = Retro, 1 = Vivid)	R/W
3A	58	6	VGA Output (0 = 80C board, 1=VDP) [2]	R/W
3A	58	7	SDX display disk activity (1 = display on)[3]	R/W
3B	59	-	Read Access to the Page Port Register [4]	Read

1 If this bit is set, it turns on the 80 column character DMA for doing the 1 character per cycle speed up used by CLS in 80 column mode. It speeds up the display of repeating characters.

2 If this bit is set, the VGA output shadows the VDP, is unset, the VGA output is from the emulated 80 column board.

3 If this bit is set, the flashing border to indicate SD card activity in SDX mode is enabled. In CP/M mode, the cursor always flashes red when SD card activity is taking place.

4 The Page Port register controls access to ROM & RAM. In a standard MTX, the page port (0) is Write only. On MFX, the page port is copied to Port 59 to allow its status to be read.

19.5 Readable Page Port

The MTX uses I/O Port 0 to configure its memory, normally this Page Port is Write Only. To allow it to be accessible to the programmer, MFX makes the status of the Page Port accessible via Port 59.

19.6 Serial Number Register

Each MFX has a unique serial number assigned during construction, it is displayed in the system boot menu screen and on the CP/M boot screen. It can also be read as required via port 53 (35h).

The serial number is stored in an array of 8 registers in the FPGA

Offset	Function	Data Type	Example
0	(spare)	Byte	x
1	Initial 1	ASCII	D
2	Initial 2	ASCII	S
3	Serial	BCD	1
4	PCB Major Version	BCD	1
5	PCB Minor Version	BCD	0
6	FPGA Major Version	BCD	0
7	FPGA Minor Version	BCD	32

Writing to the port sets a 3 bit pointer, reading the port then reads the data for that location, and increments the pointer.

For example, from BASIC

```
10  OUT 53,0              Set the pointer to the first location
20  DIM A(8)              Create an array of 8 integers
30  FOR X=1 to 8          Create a loop counter
40  LET A(X)=INP(53)      Stores the values from the serial number
50  NEXT X                (Port 53 location counter auto-increments)
60  FOR X=1 TO 2          Read the initials as integers (ASCII)
70  PRINT CHR$(A(X));     Get the character equivalent and concatenate
80  NEXT X
```

Would print the initials stored in the FPGA registers, i.e., “DS”

The five numeric values at offsets 3 to 7 are stored in BCD format and would need to have BCD to decimal conversion performed on them to make the values meaningful. For example, an FPGA minor version number stored as 32 (0011 0010) in BCD would return a decimal value of 50 using PRINT (A(7)) in the example above.

19.7 SD Card Contents

The SD card distributed with MFX is partitioned as described in Chapter 3.

The draft contents of the SD card are summarised in the following table:

Partition	Type	Disc	Description
0	18	B	59k CP/M System with CP/M programs
1	19	C	59k CP/M System with .BAS, .RUN
2	1A	D	59k CP/M System with .MTX
3	1B	E	59k CP/M System, scratch disc
4	1C	-	59k CP/M System, with minimal boot files
5	1D	-	59k CP/M System, with minimal boot files
6	1E	-	Backup of Partition 0
7	1F	-	Sample 54k CP/M System
8	n/a	-	Hidden partition with HEXTRAIN data

The contents of the SD card are likely to change at short notice and are not included in this manual. For a detailed breakdown of the card's contents, please see the document that you will have been pointed to when you received your MFX.

(The document will be hosted on the MFX web pages for ease of maintenance.)

For maximum flexibility, it is suggested that all partitions are made bootable and a minimum set of the most commonly used .COM files copied to every partition,

"Essential" Programs
STARTUP.COM Set/Change start-up commands
RECONFIG.COM Set/Change disc/partition mapping
STAT.COM Display file/disc info
PIP.COM File copy utility

Less Commonly Needed
REFORMAT.COM (Re)format a disc/partition
SYSCOPY.COM Make a disc/partition bootable

19.8 SD Card Media

The SD card logic used in the first versions of the MFX ROM were cloned from REMEMORize, and as such, were subject to the same limitations. For REMEMORizer, Andy stipulated that “SD Cards between 64MB and 1GB are supported” (although SD cards up to 2GB were found to work with MFX).

This restriction is due to the default format used in the original SD card specification, i.e., FAT16, giving typical card capacities of between 128MB and 2GB. The maximum capacity of such SD cards is far greater than would ever be needed for a device that targeted the Memotech MTX. However, development of the SD card specification, with the resultant increase in card capacities, means that original SD format cards are becoming much harder to source and correspondingly more expensive.

To allow a wider range of SD cards to be used, Bill Brendling has modified the SD card routines in the MFX ROM to cater for both SD and SDHC format cards. This functionality is available in MFX ROM versions 167 and above. (No changes are required to the FPGA programming to support the additional capacity.)

From Version 167, the MFX Boot process checks the SD card type and enables SD/SDHC support as appropriate.

Note: More advanced SD card formats, such as SDXC are unlikely to ever be compatible with MFX.

APPENDIX A. SD Card Maintenance

This appendix describes how the MFX SD card can be maintained outside of MFX itself. The tools described in this chapter are predominantly Windows based. Equivalent, and perhaps better, tools are available for other operating systems, including Linux. Since the author does not have experience using them and little knowledge of Linux, MFX users who need help with Linux or other OS based tools are advised to seek help on the Memotech forum (http://mtxworld.dk/memorum/index.php), some of whose users are “Linux aware”.

The flash memory cards used in CFX, MFX and REMEMOrizer are not formatted in the same way as they would be for use in a PC which, for example, might use FAT32. Consequently, the cards are not normally accessible using standard Windows tools, such as Explorer etc. Attempting to use such tools to format or read/write the card will likely destroy their contents and render them useless to MFX etc. until they are “reformatted” and their contents replaced.

The memory card is used in a RAW format, accessed by direct read/write to the sectors on the card. Any manipulation of the data on an external device, e.g., a PC, must be done using tools that support block level access to the card.

In addition, the card must not contain any high level partition data; any pre-existing partition data created when the card was manufactured must be removed before it can be used with MFX.

i Prepare a new card for use with MFX

Tools, such as ***Mini Tool Partition Wizard Free*** (https://www.partitionwizard.com/free-partition-manager.html), are necessary to remove the partition data from pre-formatted cards.

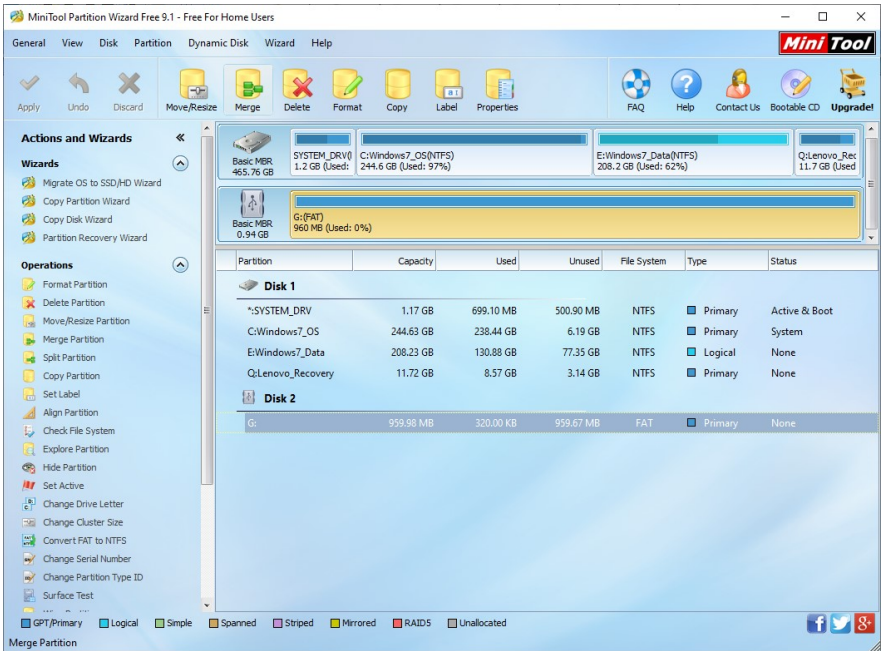


Figure 13 – Mini Tool Partition Wizard

Note: Careless use of low level disk tools such as this can trash your hard drive – be sure that you know what you're doing, and most importantly, that you are taking actions on the right disk/partition !

In this image, you can see **Mini Tool Partition Wizard** running on my laptop. I have one 500GB hard disk (Disk 1) with four partitions configured, shown as a hidden system partition, with three additional partitions C:, E: and Q:.

The SD card appears as Disk 2. Out of the box, this SD card with an advertised capacity of 1GB, has a capacity of approximately 960MB when formatted with a FAT file system and is shown as G:.

For use with MFX, the SD card partition must be removed.

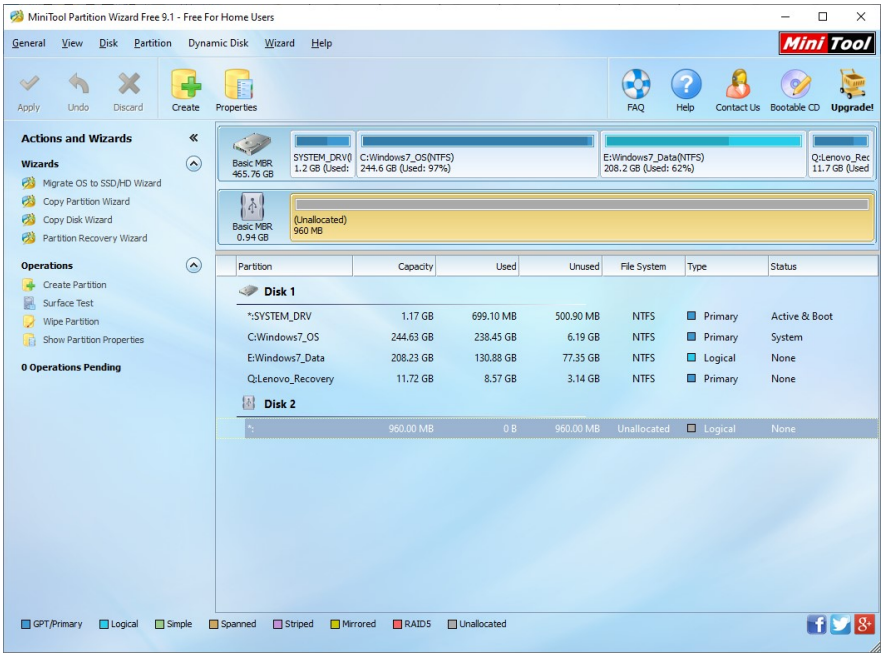


Figure 14 – Mini Tool Partition Wizard

After deleting the FAT partition, the SD card now shows the file system as "Unallocated".

Note: when the SD card is subsequently inserted into a Windows PC, you will likely be prompted for format it:

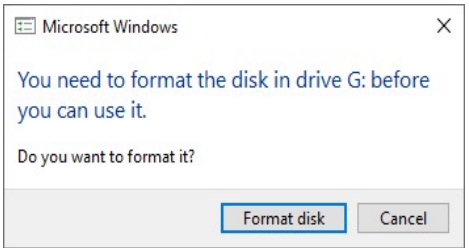


Figure 15 – Windows Explorer Format Dialogue

Obviously, you should cancel this operation, otherwise, you'll be back to where you started!

The blank SD card can now be used with MFX.

ii Disc Image Tools

a. HDRawCopy

Tools such as ***HDRawCopy*** (https://hddguru.com/HDD-Raw-Copy-Tool/) can be used to backup and restore images of the SD card. It is recommended that you create an image backup of the MFX SD card when you receive it.

When HDRawCopy is opened, it will ask for the source and destination files/devices.

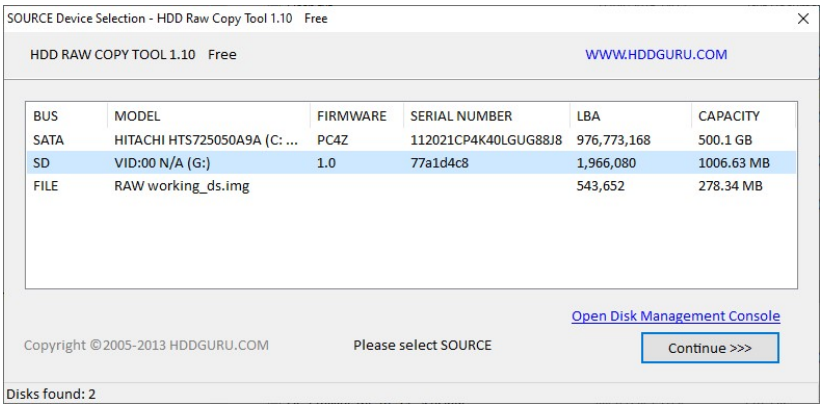


Figure 16 – HDRawCopy Source Selection

Here, I have selected the SD card slot (G:) which has my MFX SD card inserted with a capacity shown as ~1GB as expected. Selecting “Continue” allow the target file or device to be selected.

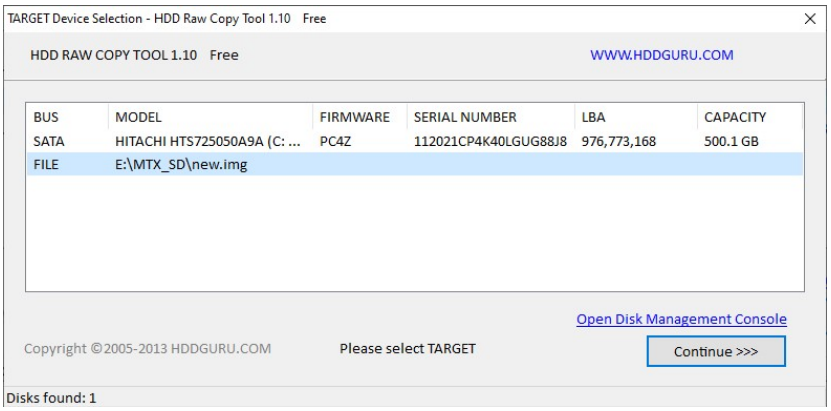


Figure 17 – HDRawCopy Target Selection

Here, I have chosen a file called new.img (which currently does not exist). Selecting “Continue” displays the Source and Target.

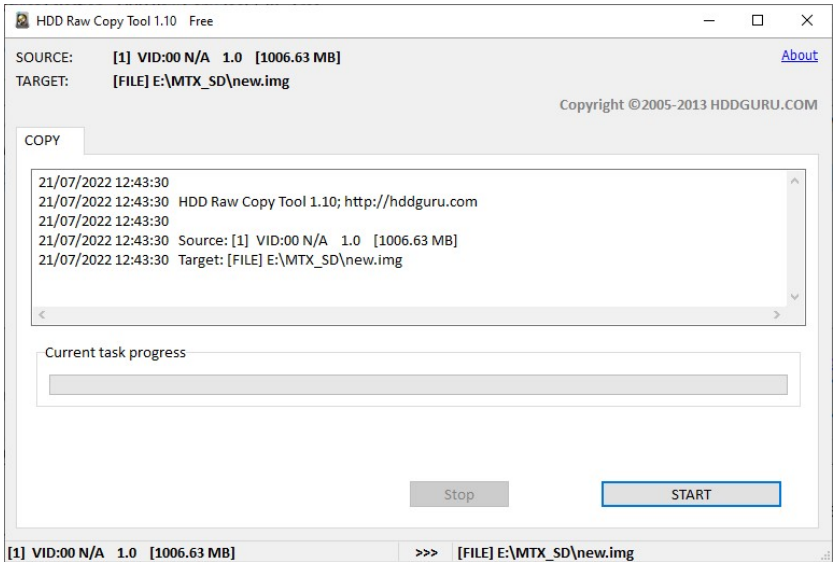


Figure 18 – HDRawCopy Start Selection

"Start" creates the image specified

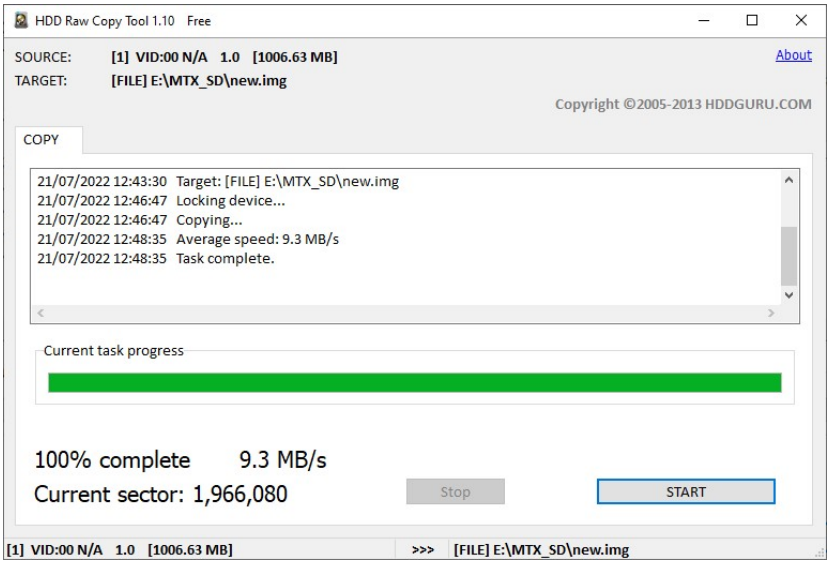


Figure 19 – HDRawCopy Completed

As should be expected, the RAW image is a copy of the full SD card, including the large amount of free space at the end of the card; this is quite wasteful of hard disk space and you may wish to consider alternative tools that allow for more flexibility when imaging the SD card, or portions of it.

b. RAWIO

Andy Key has released a tool called RAWIO (http://www.nyangau.org/rawio/rawio.htm) which allows a Windows based PC to do RAW read/writes under Microsoft Windows it allows you to do block transfers between image files and an SD card.

Entering **rawio** on the command line lists the available program options

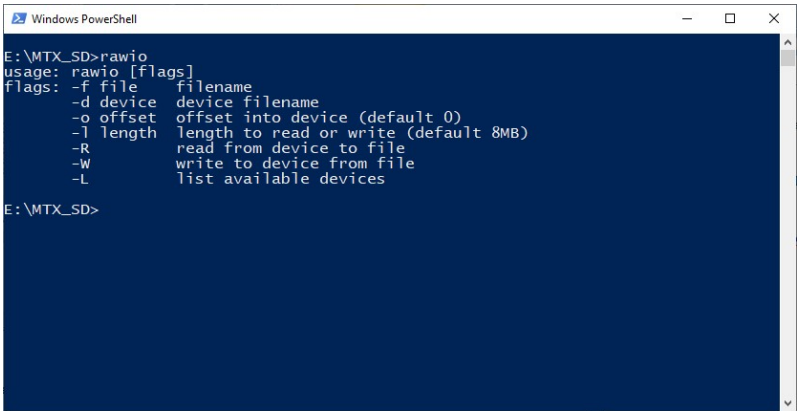


Figure 20 – RAWIO Options list

Entering **rawio -L** from an elevated Command Prompt lists the available devices

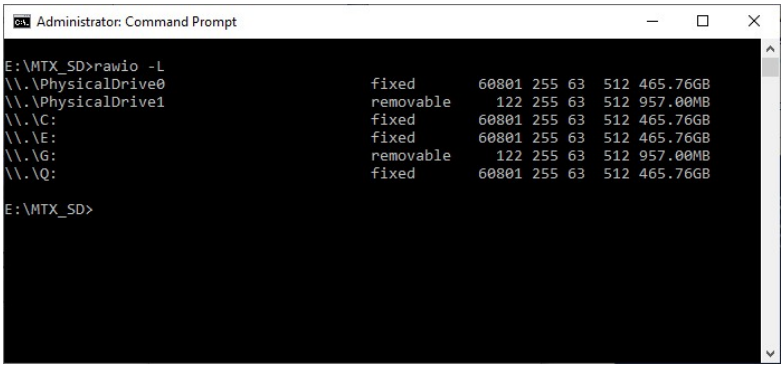


Figure 21 – RAWIO Options list

Note, it is not enough to be logged is a user with Administrator privileges, you need to specifically start the Command Prompt with Administrator privileges to list the available physical devices

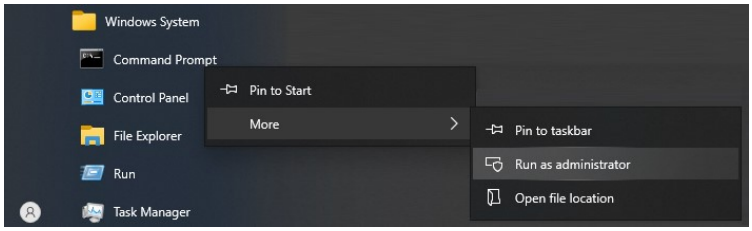


Figure 22 - Administrator Command Prompt

The image shown illustrates how to achieve this on Windows 10; from the Start menu, under Windows System, right click on Command Prompt and under the More tab, select Run as administrator.

The format of each line is devicename, type, cylinders, heads, sectors, block size and total device size.

As with other low level tools, careless use can destroy your hard disk – make sure that you are interacting with the SD card. In this example, you can see the hard disk attached to my system, identified as PhysicalDrive0, with a size of 465GB, and my MFX SD card, identified as PhysicalDrive1, with a size of 957MB.

The advantage of **rawio** over, say, **HDRawCopy**, is that it is particularly useful for working with image files of a size appropriate the MFX image requirements, rather than the whole size of an SD card which may be considerable larger than the image file needed.

Examples of usage are the insertion of blocks of data into larger image files in the correct place for a Memotech SD data type partition image, for example, inserting an 8MB block at an offset of 16MB places the data in the correct place for a Type 1A partition.

To read an 8MB chunk from offset 16MB in the device to file :-

```
c:\rawio>rawio -d \\.\PhysicalDrive1 -o 16MB -l 8MB -f myfile.bin –R
```

To read the whole useable area of the SD card and save to file :-

```
c:\rawio>rawio -d \\.\PhysicalDrive1 -l 270MB -f myfile.bin –R
```

(This will save the data from the 8 drive images as well as the hidden HEXTRAIN data.)

To write an 8MB from a file to offset 24MB on a device :-

```
c:\rawio>rawio -f myfile.bin -d \\.\PhysicalDrive1 -o 24MB -l 8MB –W
```

Offsets and lengths are specified in bytes, unless you use a suffix such as KB, MB, ...etc.

iii File Level Access

The most convenient way to exchange files between the MTX and a PC is to use network file transfer (see Chapter 17), but it is possible to perform direct file transfer between the PC and the SD card if the user wishes.

As previously noted, since the MFX SD card does not use a Windows compatible format, additional tools are required.

Andy Key has released a range of tools that facilitate this. Andy's tools are available for download on the Memotech area of his website (http://nyangau.org/memotech/memotech.htm) along with guidance on their use. This section will briefly describe the tools that I have found most useful, but for comprehensive instructions on their use, please refer to Andy's site.

CPMTOOLS

cpmtools was written by Michael Haardt and is available for download from his website (http://www.moria.de/~michael/cpmtools/). The program allows a PC to access CP/M file systems, provided that the disk definition file (***diskdefs***) contains details of the disk/image format.

Andy has created a ***diskdefs*** file that includes details of the format of Memotech's physical disks, Silicon discs and RAM discs, as well as the SD card format used in MFX etc. Using cpmtools with Andy's ***diskdefs*** file, it is possible to transfer files between the PC and SD card image file from the command line. This tool provides a basic level of functionality, but the author recommends Andy's newer offering, **cpmcbfs**.

Andy's ***diskdefs*** file is available in the ***mfloppy*** (http://www.nyangau.org/mfloppy/mfloppy.htm) download.

CPMCBFS

cpmcbfs makes use of the ***EldoS Callback Filesystem*** library (https://www.callback.com/cbfsstorage/), combined with ***cpmtools***, to allow Windows to mount CP/M filesystems.

[For Linux users, Andy has released an equivalent program ***cpmfuse*** (CP/M Filesystem in userspace) also available on his website (http://www.nyangau.org/cpmcbfs/cpmcbfs.htm).]

The description on Andy's webpage includes the text "At the moment, it's only a command line program and there is no Windows GUI. It would be nice to be able to pick the file or device to mount and the drive to mount it as, and perhaps even be able to do this from Explorer. I don't know how to do this at present." This may be a little misleading, at least, it was to me. Although ***cpmcbfs*** itself is only a command line tool, once the program is running, the image file or device is exposed to Windows Explorer and the usual Windows file drag & drop features can be used.

This tool provides a really convenient mechanism for transferring files between the MFX SD card or image files. Because ***cpmcbfs*** is built using code from ***cpmtools*** it uses the same ***diskdefs*** configuration file. Using the appropriate Memotech Type number, it is possible to interact with all partitions on the SD card or image.

Once ***cpmcbfs*** has been installed (see Andy's site), to interact with an image file, you can enter, for example,

c:\cpmcbfs>cpmcbfs -f memotech-type18 -i sddisc.bin -v

The system should respond with,

Drive Z: should exist until you press Enter (Z: is the default drive)

In my case, using my image backup from MFX, and drive letter 't:',

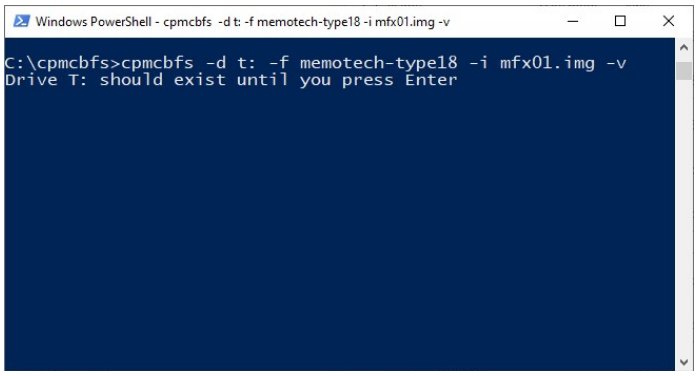


Figure 23 – cpmcbfs pointed to an image file

The first partition on the image is now available in Windows Explorer

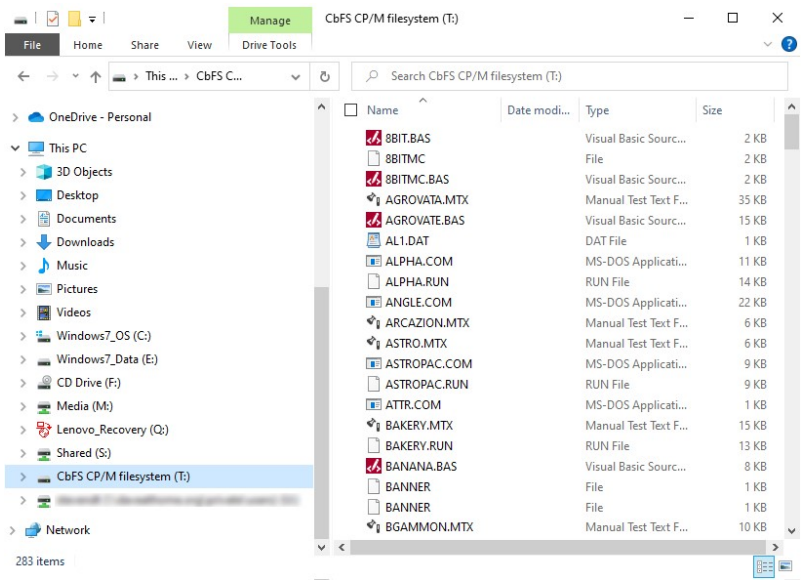


Figure 24 – cpmcbfs Image partition in Windows Explorer

The first partition on the image is now visible as drive T:, the file types reported by Windows may be misleading, they are Windows "best guesses", based on its current file associations – obviously, the .BAS files are SDX BASIC programs – not VB source, but apart from this minor niggle, the tool does a really great job.

By loading two instances of the tool, you can even interact with two partitions on the image file simultaneously and transfer files between them.

In this example, **cpmcbfs** is being used to work with the SD card inserted into the PCs SD card slot configured as drive G: which would expose the contents to Windows Explorer as shown above.

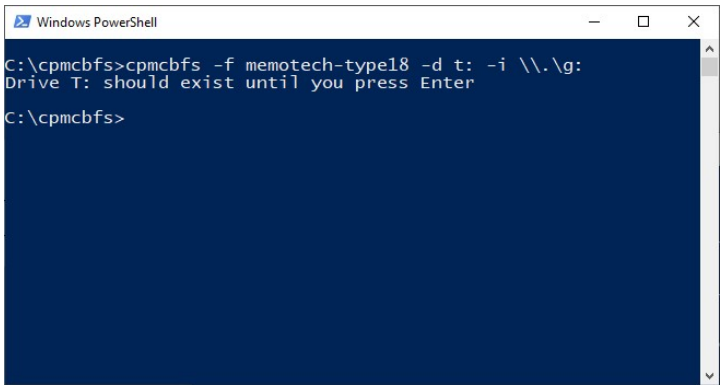


Figure 25 – cpmcbfs pointed to the SD card

(It is not possible to access two partitions of the same SD card simultaneously.)

APPENDIX B. MTX & MFX I/O PORT ALLOCATION

Port		Direction	Description	Key
Dec	Hex			
0	0	Output	Copy of the main Page port	
1	1	Output	Data to write to the shadow VDP screen memory	1
2	2	Output	Shadow VDP address and register setup	1
48	30	Output	Set 80 Column Display low address Initiate data transfer	2
49	31	Output	Set 80 Column Display high address and mask bits	2
50	32	Output	Set 80 Column ASCII character code	2
		Input	Read 80 Column ASCII character code	2
51	33	Output	Set 80 Column character attribute	2
		Input	Read 80 Column character attribute	2
52	34	Output	Set 80 Column repeat count	
		Input	Read 80 Column repeat count	
53	35	Output	Set Serial number pointer	
		Input	Read serial number data and increment pointer	
56	38	Output	Set emulated CRTC register pointer	2
		Input	Read emulated CRTC register pointer	2
57	39	Output	Set emulated CRTC register	2
		Input	Read emulated CRTC register	2
58	3A	Output	Set FPGA control register	
		Input	Read FPGA control register	
59	3B	Input	Read page port	
144	90	Output	Write WIZnet mode register	
		Input	Read WIZnet mode register	
145	91	Output	WIZnet address register High	
146	92	Output	WIZnet address register low	
147	93	Output	Write to WIZnet register	
		Input	Read WIZnet register	
212	D4	Output	SD card control	3
		Input	Read SD card status	3
214	D6	Output	Write SD card	3
		Input	Read SD card	3
215	D7	Output	Write SD card	3
		Input	Read SD card/ Send #FF byte	3

Key

1 Shadow VDP access
2 80 column board, as per FDX
3 SD card as per REMEMOTECH & REMEMOrizor

APPENDIX C. MFX Revision History

Release History	FPGA	ROM	GAL
Initial Production Release	01-00	134	2.0
Support for SDHC cards	01-00	166	2.0
Fix for USER VGA issue from ROM v166	01-00	167	2.0
Fix for Shadow Page Port issue	01-01	167	2.0

ROM Revision 166

Adds the ability to use SDHC cards with MFX. Legacy SD cards are becoming less common and more expensive, this ROM update allows SDHC cards to be used with MFX in the same RAW format as SD cards. To maintain compatibility with the imaging tools described in Appendix A, the same card structure must be used and FAT formatting is still not supported.

SDHC initialisation code contributed by Bill Brendling

NB: SDHC cards are not compatible with HEXTRAIN's data partition. Users who want to continue to use HEXTRAIN should continue to use an SD card for HEXTRAIN.

Andy's HEXTRAIN program uses customized code to access the hidden HT data area and will be incompatible with the new ROM. Bill is working on modifying the HT program to correctly handle the SDHC card format. An updated HT program will be available for download in due course.

ROM Revision 167

Fix for the error introduced to the **USER VGA** command in ROM version 166.

In tandem with this, Bill created a new version of the HEXTRAIN executable that can handle both SD and SDHC cards. The program is available from the ***MFX Firmware*** page on the website and has been added to the MFX SD card image file. Though both are present in the image file, HTSD.COM supersedes HT.COM for use owithboth SD and SDHC ards.

FPGA Revision 01-01

Fix for the Shadow Page Port logic error identified in very early shipments of MFX. This error is only present on MFX boards with serial numbers below xx-10 and unlikely to actually impact any users unless they are writing programs that read the Shadow Page port status.

Index