GMD128

Morrow MD128 Z80 Computer Emulator.

4th December 2020

This Page is left Blank

TABLE OF CONTENTS

GMD128 APPENDICES

IMAGES AND PRINTOUTS

1. INTRODUCTORY INFORMATION

1.1 Overview

GMD128 emulates the early 1980s Morrow Micro Decisions Z80 Computer Series.

GMD128 uses disk-image files which may contain Morrow MDxx Operating Systems of the period: mainly CP/M. Other Morrow Operating Systems may work, but haven't been tested.

Most of the Morrow Micro Decision systems used either two floppy disks or one floppy disk plus one hard disk. The size of the hard drive generally indicated the name of the system. Thus, the Morrow MD11 system had a floppy disk plus an 11-megabyte hard disk. There was no such animal as a Morrow MD 128. If it had existed, it probably would have cost in the vicinity of $50,000 – an absolute fortune in the mid 1980s. I have used that name to indicate that it has 128 megabytes of hard disk storage. This is made up of four 32 megabyte hard drive image-files, each of which holds four 8-megabyte CP/M virtual disk drives.

GMD128 is constructed from two modules. The first module, the GXE Z80 Emulator Toolkit, contains the 64K of RAM, a Z80 microprocessor emulator, and a display screen. Drop-down menus emulate the operator's interaction with the hardware, such as inserting or removing floppy disks, and organizing the interaction between the host linux machine and the virtual Z80 machine. This first module is installed as a Graphical User Interface (GUI) and can be called by GMD128 or any other Z80-based emulator software, such as CPZ (ICMS CPZ-48000 emulator) or NSE (North Star HORIZON emulator).

The second module consists of the Morrow system-specific components, such as the data and control ports of the serial and parallel I/O, the Disk Jockey DMA floppy-disk controllers, with associated boot PROMS, and the HDDMA fixed-disk controllers

1.2 ATTRIBUTIONS FOR OTHERS' CODE in GMD128

GMD128's Z80 emulation code pretty much comes from yaze, a CP/M emulator written by Frank Cringle. Morrow-specific code amendments such as the floppy-disk I/O, and a few other additions such as Mode 2 interrupt code were made by Jack Strangio.

GMD128's Z80 disassembly code comes from Marat Fayzullin's 1999 DAsm code with some local alterations.

The rest of GMD128 cannot be blamed on anyone else but myself.
Jack Strangio, November 2020

1.3 THANKS

I have only the greatest appreciation for all those who have helped me in my rather idiosyncratic quest to write emulators of the North Star Horizon, and several other ‘home computers’ as they were called in the late 1970s and early 1980s.

The North Star Horizon was my first computer which took more than 40 hours to build over the course of several weeks in late 1978. The thousands of solder-joints literally burned-out a new soldering iron. It says a lot for the quality of the instruction manual that most of the time I really had no idea what each step did but at the end (once my half-dozen wiring mistakes were fixed) I had assembled a computer which worked perfectly.

Often, just a little bug in the emulators would get people off writing to me and setting off another round of debugging and coding. Thanks to you all.

1.4 FLOPPY DISKS AND FOUR HARD DISKS SUPPLIED WITH GMD128

Several floppy disks are supplied with GMD128 to get you up and going quickly. They are stored in the 'disks' subdirectory. These archive disks have been renamed to allow their uses to be self-explanatory.

boot-8hx-b114.fd8

8-inch boot disk (8hx) The BIOS version B114 is configured for 2 8-inch floppies A: and B: and two of the JX32 hard drives (C:,D:,E:,F: and G:, H:, I:, J:).

empty-8hx.fd8

8-inch blank floppy disk.

jx32_abcd.img

32-megabyte hard drive with boot system track. (hxx) This BIOS B113 is configured for zero floppy disks, and four 8-megabyte hard drives, giving the maximum number of sixteen CP/M disk drives. The size of each CP/M virtual disk, 8-megabytes, is the maximum size allowed by CP/M.

jx32-efgh.img
jx32_ijkl.img
jx32_mnop.img

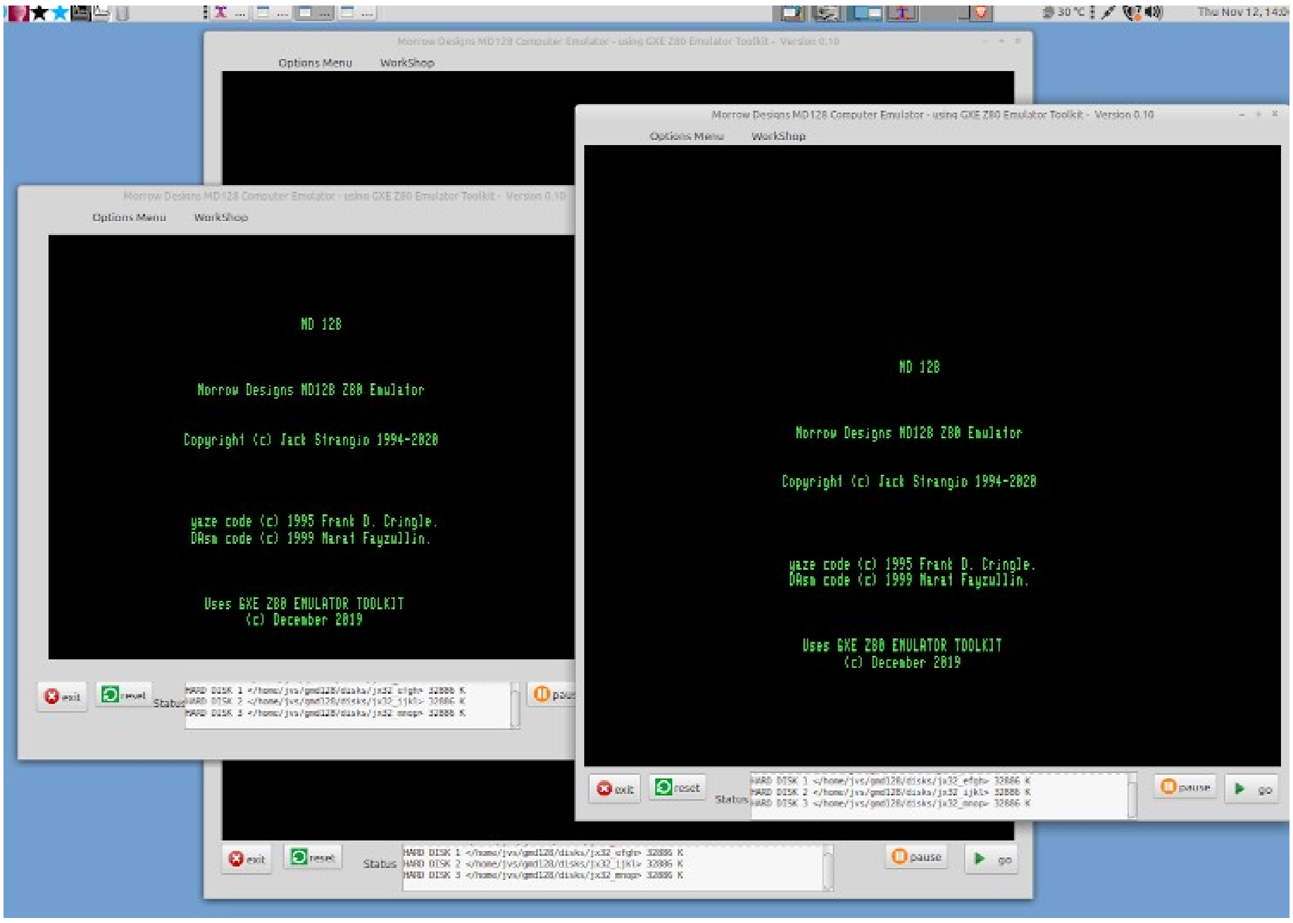
32-megabyte data drives.

Some of the data drives have been used for development, especially the B: drive which has several GMD128 BIOS source files.

1.5 SCREENSHOTS

(Note: Most of the screen images included in this User Guide do not render well. They look better when displayed dot-for-dot as screenshots. Find them in the ‘screenshots’ directory of the downloaded tarball.)

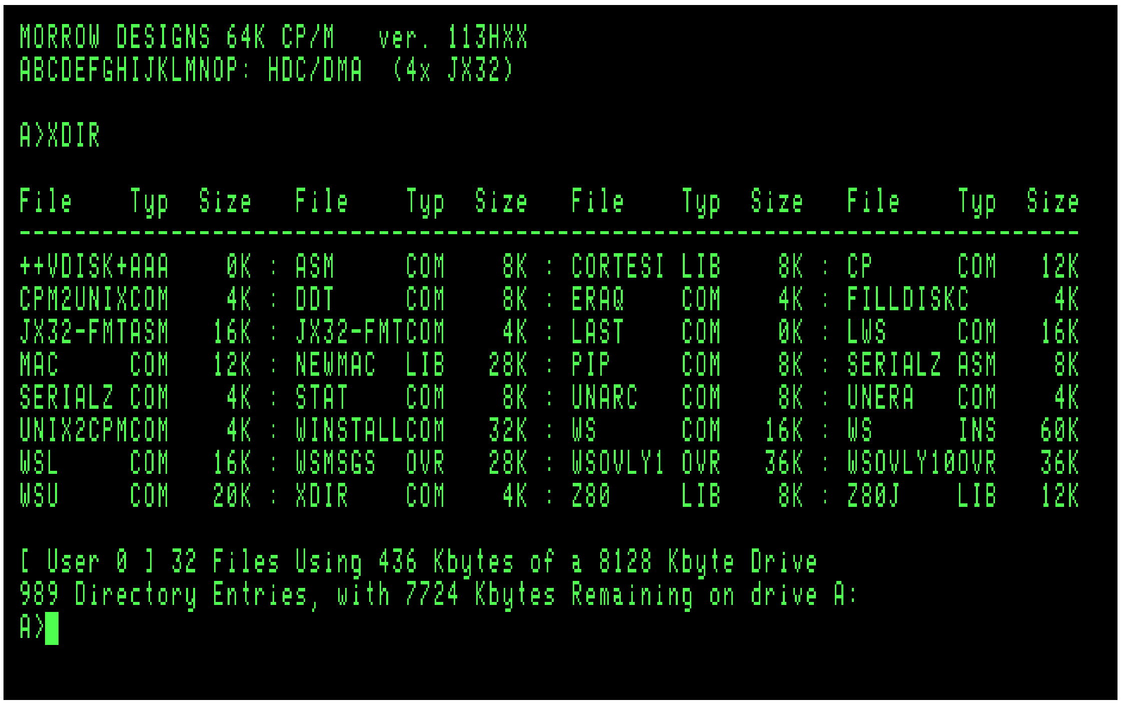
GMD128 is GTK+ based. When GMD128 starts up it will look at the screen-resolution and display a ‘terminal’ of a size that is suitable for that resolution. The ‘large’ terminal (110 chars wide, 45 lines) will fit on a 1920x1080 screen. The ‘medium’ terminal (96 chars wide, 36 lines) will fit on an intermediate resolution laptop. Then there is the ‘small’ terminal which is actually the historical standard-sized serial terminal of 80 chars wide and 25 lines high. Both the medium and small terminal displays can be specified with a command-line option.


Morrow Designs MD128 Computer Emulator - using GXE Z80 Emulator Toolkit - Version 0.10
Options Menu
WorkShop
exit
reset
Status
pause
go

```
MD 128

Morrow Designs MD128 Z80 Emulator

Copyright (c) Jack Strangio 1994-2020

yaze code (c) 1995 Frank D. Cringle.
DAsm code (c) 1999 Marat Fayzullin.

Uses GXE Z80 EMULATOR TOOLKIT
(c) December 2019
```

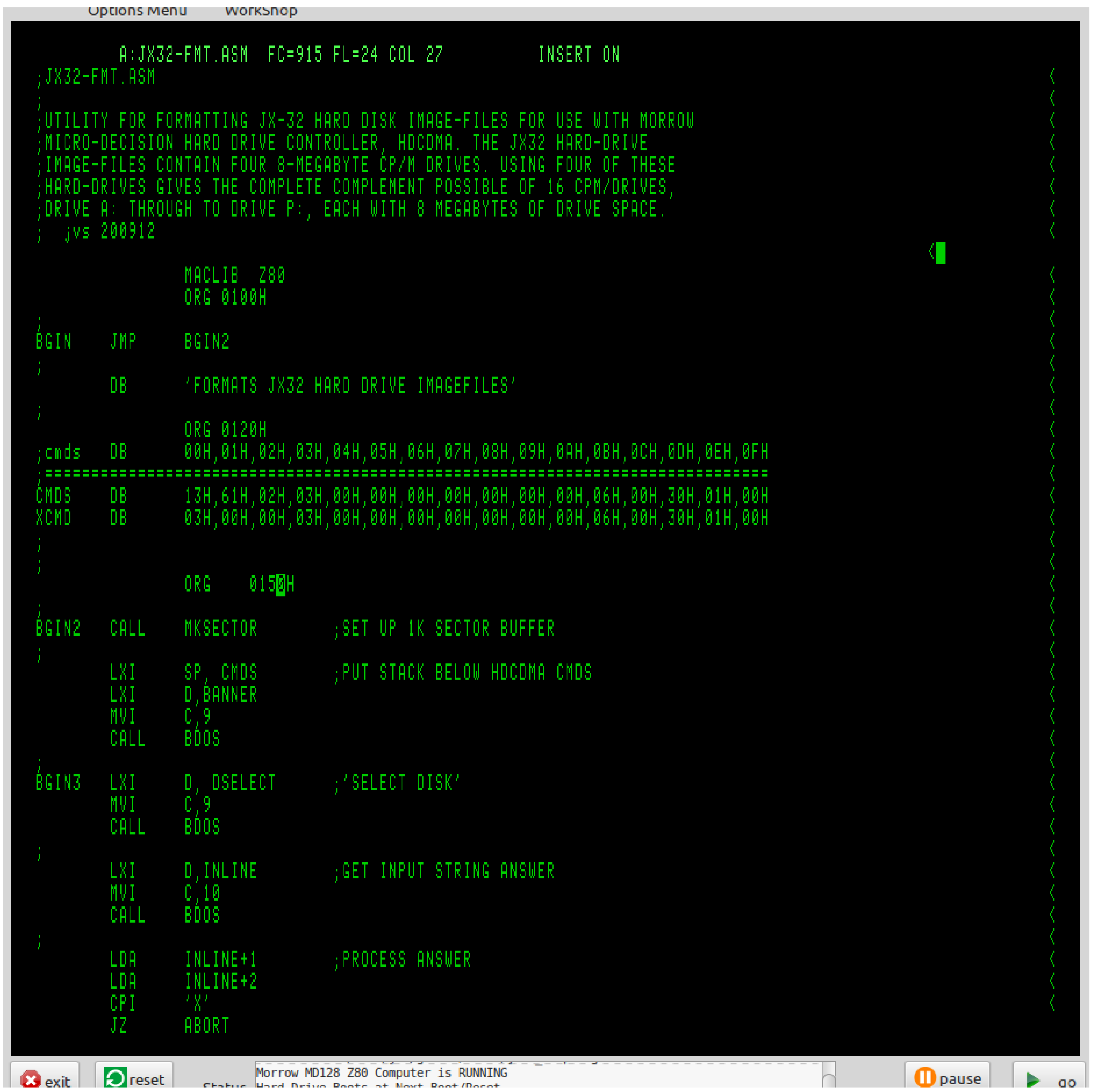
```
HARD DISK 1 </home/jvs/gmd128/disks/jx32_efgh> 32886 K
HARD DISK 2 </home/jvs/gmd128/disks/jx32_ijkl> 32886 K
HARD DISK 3 </home/jvs/gmd128/disks/jx32_mnop> 32886 K
```

Fig 1. Three GMD128 startup screens, running on a 1920x1080 resolution screen in 110x45 display, 96x36 display, and in 80x25 character screen formats.



```
MORROW DESIGNS 64K CP/M   ver. 113HXX
ABCDEFGHIJKLMNOP: HDC/DMA  (4x JX32)

A>XDIR

File     Typ  Size   File     Typ  Size   File     Typ  Size   File     Typ  Size
-------------------------------------------------------------------------------
++VDISK+AAA    0K : ASM      COM    8K : CORTESI LIB    8K : CP       COM   12K
CPM2UNIXCOM    4K : DDT      COM    8K : ERAQ    COM    4K : FILLDISKC       4K
JX32-FMTASM   16K : JX32-FMTCOM    4K : LAST    COM    0K : LWS      COM   16K
MAC      COM  12K : NEWMAC  LIB   28K : PIP     COM    8K : SERIALZ ASM     8K
SERIALZ COM    4K : STAT     COM    8K : UNARC   COM    8K : UNERA    COM    4K
UNIX2CPMCOM    4K : WINSTALLCOM   32K : WS      COM   16K : WS       INS   60K
WSL      COM  16K : WSMSGS  OVR   28K : WSOVLY1 OVR   36K : WSOVLY10OVR   36K
WSU      COM  20K : XDIR     COM    4K : Z80     LIB    8K : Z80J    LIB   12K

[ User 0 ] 32 Files Using 436 Kbytes of a 8128 Kbyte Drive
989 Directory Entries, with 7724 Kbytes Remaining on drive A:
A>
```

Fig 2. GMD128 splash screen followed by CP/M directory listing. GMD128 looks like a typical "green-screen" terminal of the 70's-80's period, in particular it will default to be a terminal which acts very similar to Televideo 920/ Soroc 120/ ADM3A terminals.



```
         A:JX32-FMT.ASM  FC=915 FL=24 COL 27         INSERT ON
;JX32-FMT.ASM
;
;UTILITY FOR FORMATTING JX-32 HARD DISK IMAGE-FILES FOR USE WITH MORROW
;MICRO-DECISION HARD DRIVE CONTROLLER, HDCDMA. THE JX32 HARD-DRIVE
;IMAGE-FILES CONTAIN FOUR 8-MEGABYTE CP/M DRIVES. USING FOUR OF THESE
;HARD-DRIVES GIVES THE COMPLETE COMPLEMENT POSSIBLE OF 16 CPM/DRIVES,
;DRIVE A: THROUGH TO DRIVE P:, EACH WITH 8 MEGABYTES OF DRIVE SPACE.
;  jvs 200912

                MACLIB  Z80
                ORG 0100H
;
BGIN    JMP     BGIN2
;
        DB      'FORMATS JX32 HARD DRIVE IMAGEFILES'
;
                ORG 0120H
;cmds   DB      00H,01H,02H,03H,04H,05H,06H,07H,08H,09H,0AH,0BH,0CH,0DH,0EH,0FH
;=============================================================================
CMDS    DB      13H,61H,02H,03H,00H,00H,00H,00H,00H,00H,00H,06H,00H,30H,01H,00H
XCMD    DB      03H,00H,00H,03H,00H,00H,00H,00H,00H,00H,00H,06H,00H,30H,01H,00H
;
;
                ORG    0150H
;
BGIN2   CALL    MKSECTOR        ;SET UP 1K SECTOR BUFFER
;
        LXI     SP, CMDS        ;PUT STACK BELOW HDCDMA CMDS
        LXI     D,BANNER
        MVI     C,9
        CALL    BDOS
;
BGIN3   LXI     D, DSELECT      ;'SELECT DISK'
        MVI     C,9
        CALL    BDOS
;
        LXI     D,INLINE        ;GET INPUT STRING ANSWER
        MVI     C,10
        CALL    BDOS
;
        LDA     INLINE+1        ;PROCESS ANSWER
        LDA     INLINE+2
        CPI     'X'
        JZ      ABORT
```

```
Morrow MD128 Z80 Computer is RUNNING
```

Fig 3. A custom-configured version of Word Star running in a high and wide screen format (110 chars,45 lines)

1.6 GMD128 COMMAND-LINE START-UP OPTIONS

```
gmd128   [-c config-file]  [-m or -l]
```

-c <config-file>

Use an alternate config-file instead of the default '/home/username/gmd128/gmd128.conf' file. The alternate file should also be placed in the '/home/username/gmd128' directory.

-m

Specify the use of a display 96 characters wide by 36 lines.

-l

Specify the use of a display 110 characters wide by 45 lines.

1.7 GMD128 COMMAND-LINE STARTUP EXAMPLES:

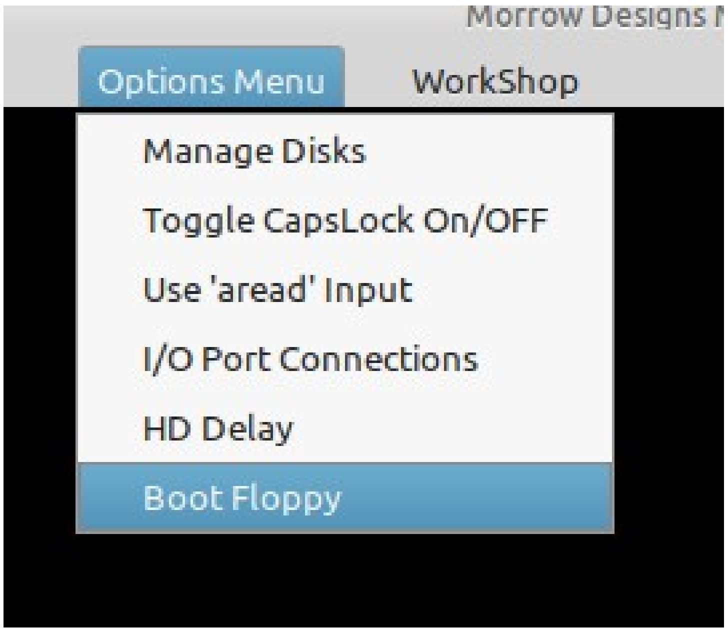
```
gmd128  -c morrow.zzz
```

Start GMD128 using the floppy or hard disk controller, booting from the disk-image file which is specified in the '/home/username/gmd128/morrow.zzz' configuration file.

```
gmd128
```

Start GMD128 using the floppy or hard disk controller, booting from the disk-image specified in the default configuration file.

NOTE: The default boot device is the hard drive. To boot the with the floppy disk, open the Options Menu, and toggle the 'Boot Floppy' menu item to 'ON'. The emulator will boot from the floppy the next time the emulator is started or reset. To go back to booting from the hard drive again, toggle the 'Boot Floppy' menu item to 'OFF'.



1.8 GETTING THE EMULATOR’S START-UP CONFIGURATION

1.8 THE WORK DIRECTORY

All of the emulators do their work in a directory which has the general look of

/home/user_name/emulator_name/

thus if user ‘fred’ is working with the ‘gmd128’ emulator, the gmd128 work-directory is installed at

/home/fred/gmd128

In this work-directory will be found any logfiles, such as the **screenlog** which contains a record of all the output that was displayed by the screen during the emulator’s activity. If any debug logging was required there will be a debug log called **xlog** written into the work-directory.

Also found in the work-directory are any configuration files which specify which floppy and hard disk images will be used while the emulator is working.

Subdirectories in the work directory are **disks**, **documentation**, and **info**.

The **disks** subdirectory is where floppy and hard disk images may be found. It is a good idea to put any other disk-images in there also. That is the first place that the emulator will usually look for disks.

The **documentation** subdirectory is where official Manufacturer and CP/M documentation is found.

The **info** subdirectory is where other useful information may be placed.

1.9 CONFIGURATION FILES:

A user's default configuration file is found at

/home/username/work_directory/emulator_name.conf

thus user **fred** will find his *default* **gmd128** configuration file at

/home/fred/gmd128/gmd128.conf

Bear in mind, though, that any other configuration filename can be specified on the command-line by using the **-c** option, as in

/home/fred/.local/bin/gmd128 **-c gmd128-001.conf**

or even just simply

gmd128 -c gmd128-001.conf

if gmd128 is located in one of your $PATH directories, and gmd128-001.conf is located in the work-directory.

1.10 USER CONFIGURATION FILES: gmd128.conf

This is an actual configuration file.

```
###
### Configuration File for Morrow GMD128 Z80 Emulator (c) 201112
###
###            Avoid Editing This File Manually.
###
### Any Changes You Make Are Liable To Be Overwritten at Any Time.
###
fd0           /home/jvs/gmd128/disks/empty-8hx.fd8
fd1
fd2
fd3
hdd0          /home/jvs/gmd128/disks/jx32_abcd.img
hdd1          /home/jvs/gmd128/disks/jx32_efgh.img
hdd2          /home/jvs/gmd128/disks/jx32_ijkl.img
hdd3          /home/jvs/gmd128/disks/jx32_mnop.img
disk_dir      /home/jvs/gmd128/disks/
hd_delay      off
capslock      on
s2_in         /home/jvs/gmd128/serial2_in
s2_out        /home/jvs/gmd128/serial2_out
pl_in         /home/jvs/gmd128/parallel_in
pl_out        /home/jvs/gmd128/parallel_out
=======
log           /home/jvs/gmd128/xlog
screenlog     /home/jvs/gmd128/screenlog
debug_level   0013
break_addr    FF00
break_on      off
trap_addr     FFFE
trap_on       off
boot_floppy   off
```

Note that although there is provision for 4 floppies and four hard drives, this config file only specifies 1 floppy and four hard drives. The hard drives named each hold 4 quite large (8 megabyte) virtual disks out of the 16 CP/M drives possible.

The **disk_dir** indicates where the last disk used was located, and where the emulator will look first for any other disks that are wanted.

The **hd_delay** is a way of slowing down the emulator's hard drive, usually during emulator development. Apart from that situation, the hd_delay can be left off as the normal case. See section XX for adjusting this.

Capslock is as described. Most people using CP/M will want the capslock on, but won't want it 'on' for their host machine. See section XX to vary the condition.

All the items under the **=======** separator are normally used only during the development of the emulator itself and so will rarely be used (if ever) by most **gmd128** emulator users.

NOTE: While it is possible to edit the configuration file manually, your changes will be overwritten when any of the 'Options Menu' or 'WorkShop' menu items are used.

2. Obtaining and Building ‘GMD128’

2.1 Linux Libraries required

Very few Linux libraries are required, apart from the standard packages installed on most Linux Distros.

The GUI Toolkit used is GTK+ Version 3, apart from a few deprecated functions from GTK+ Version 2.

This Toolkit can be installed using your Package Manager. If you’re using one of the Debian derivatives such as Debian itself, Mint, or Ubuntu, this can be done by installing **libgtk-3-dev** and **libglib2.0-dev** using Synaptic or even just

```
sudo apt install libgtk-3-dev  libglib2.0-dev
```

from the command line.

2.2 Get the source files

Download the GMD128 source code from https://itelsoft.com.au/code/gmd128_gtk_latest.tar.gz and move it to any convenient work directory. Untar and decompress the tarfile:

```
tar xvfz gmd128_gtk_latest.tar.gz
```

This will produce a subdirectory called gmd128. Move there.

```
cd gmd128
```

Compilation should be initiated with a simple **make** on the command-line**.**

If all goes well and the compile completes successfully, install the gmd128 package with

```
make install
```

This will install the package in the **/home/*username*/gmd128** work directory. So user ‘fred’ will find a directory called **/home/fred/gmd128**.

A launcher icon will appear on the username’s Desktop. Clicking on that should launch the emulator. It can be ‘Drag n Dropped’ to the Desktop Panel with most Desktops. Alternatively, GMD128 can be invoked from the command-line if the **gmd128** executable file is to be found somewhere within your $PATH list.

```
gmd128
```

2.3 What’s in the /home/username/gmd128 work-directory?

The /home/username/advantage directory has several important files:

gmd128.conf	the default configuration file for GMD128 which holds most of your personal preferences and: designates which CP/M disk image-files are mounted. specifies what I/O files will be attached to the machine’s I/O ports. preferred settings for capslock, hard-drive ‘speed’. preferred gmd128-development settings.

Avoid editing the gmd128.conf file manually. It gets updated automatically every time you make different choices on the Options and WorkShop menus, and will hold those settings indefinitely over more than one session.

pio_out	destination of text from the parallel-out port:	the ‘LST:’ device in CP/M
sio_out	destination of text from the serial-out port:	the ‘PUN:’ device in CP/M

2.4 Starting up GMD128

Starting GMD128 can be done from the Desktop with one of the emulator icons or from the command-line. On start-up, the program will show the title (splash screen) and will then wait for user input. Usually, the user will then just hit the 'go' button because the installation process also provides the default configuration file, **gmd128.conf**, which will be found in the GMD128 work directory, **/home/username/gmd128**

gmd128.conf contains the default settings which are expected by the Morrow Micro Decision 128 computer:

A boot floppy *must* be in 'floppy drive' 1 at the minimum when booting from the floppy.

Several GMD128 settings are also stored in the gmd128.conf file. Such as Capslock ON/OFF, and whether the hard drives runs FAST or SLOW. The **gmd128.conf** file should not be edited manually. While that can actually be done, any changes you make may not be permanent.

If for some reason, the default configuration is not present in the top directory, then a new configuration file needs to be made. This is simply done by providing the user's settings with the 'Options Menu', and/or the WorkShop menu. See Section 3, page 18. Any time a setting is altered with these two menus, the new setting is saved automatically into the **/home/username/gmd128/gmd128.conf** file.

'Options Menu': Things to be changed by the everyday user.

'WorkShop' menu:	Settings for use during GMD128 development. Most users won't need to bother with these.

2.5 Running Morrow's CP/M. The 'go button.

Now hit the 'go' button. The screen will clear, followed almost immediately by the CP/M Banner

```
MORROW DESIGNS 64K CP/M   ver. 113HXX
ABCDEFGHIJKLMNOP: HDC/DMA  (4x JX32)

A>
```

indicating that CP/M has been loaded from the hard disk boot, with the 'A> ' prompt indicating that it is waiting for a normal CP/M command-line as user input.

The BIOS version shown here has a serial number followed by 'HXX' which idicates that the BIOS has provision for

H:	boots from JX32 hard disk
X:	NO provision for 8-inch floppy drives
X:	NO provision for 5-inch floppy drives

The BIOS version provided on the 8-inch floppy disk-image is '8HX', which indicates:

8:	boots from 8-inch floppy disk
H:	has provision for JX32 hard drives (x2)
X:	NO provision for 5-inch floppy drives.

2.6. Pausing the Emulator. The 'pause' button.

In most cases you won't usually need to use the pause button unless things happen to move too fast for you, for instance to change floppies before the software moves on. Otherwise, using the Emulator is just like using a normal computer.

2.7 Rebooting/Resetting the Computer. The 'reset' button.

Just like the real thing, a reset will wipe the screen and reboot the emulator from scratch. Use this button sparingly, your work may be lost.

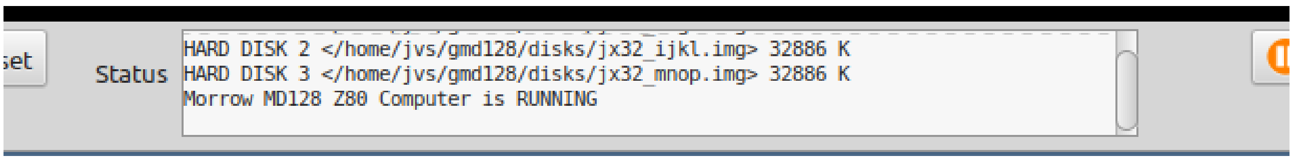
2.8. Finishing Up. The 'exit' button.

Pack it up and put it away. The GMD128 program closes down and the emulator window is closed. Settings in gmd128.conf will remain for next session.

2.9 The 'Status' window

In between the two pairs of buttons, left and right, is a small window which displays short one-line messages. This is used to show information or warnings regarding the progress of the emulator. A short beep may be heard when some messages are shown. Examples:

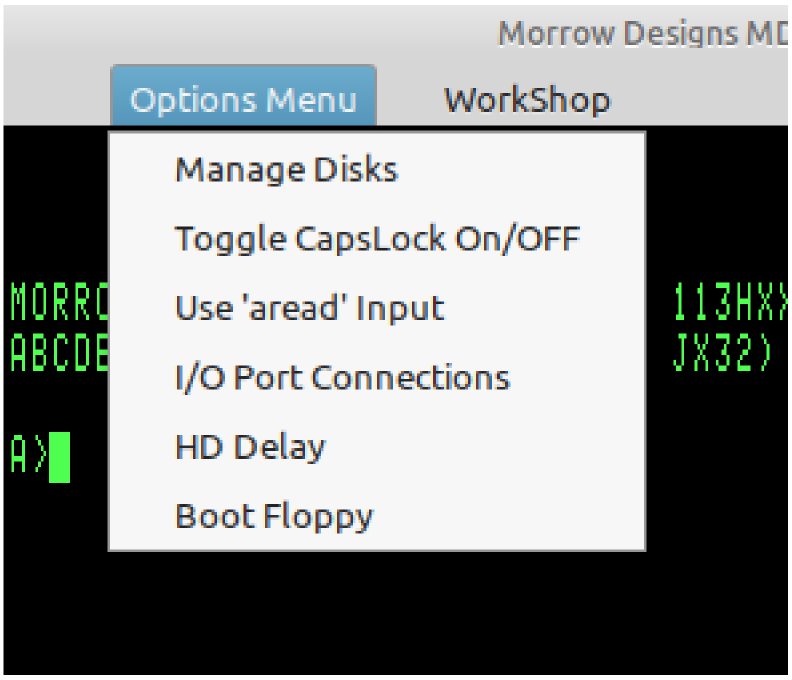
Morrow MD128 Z80 Computer is RUNNING
Capslock is now ON
HARD DISK 3 </home/fred/gmd128/disks/jx32_mnop.img> 32886 K
Morrow MD128 Z80 Computer RESET. Rebooted.



```
HARD DISK 2 </home/jvs/gmd128/disks/jx32_ijkl.img> 32886 K
HARD DISK 3 </home/jvs/gmd128/disks/jx32_mnop.img> 32886 K
Morrow MD128 Z80 Computer is RUNNING
```

Fig 5. The Status Window

3.0 The Options Menu



```
MORRO                                   113HX
ABCDE                                   JX32)

A>
```

Fig. 6. The Options Menu

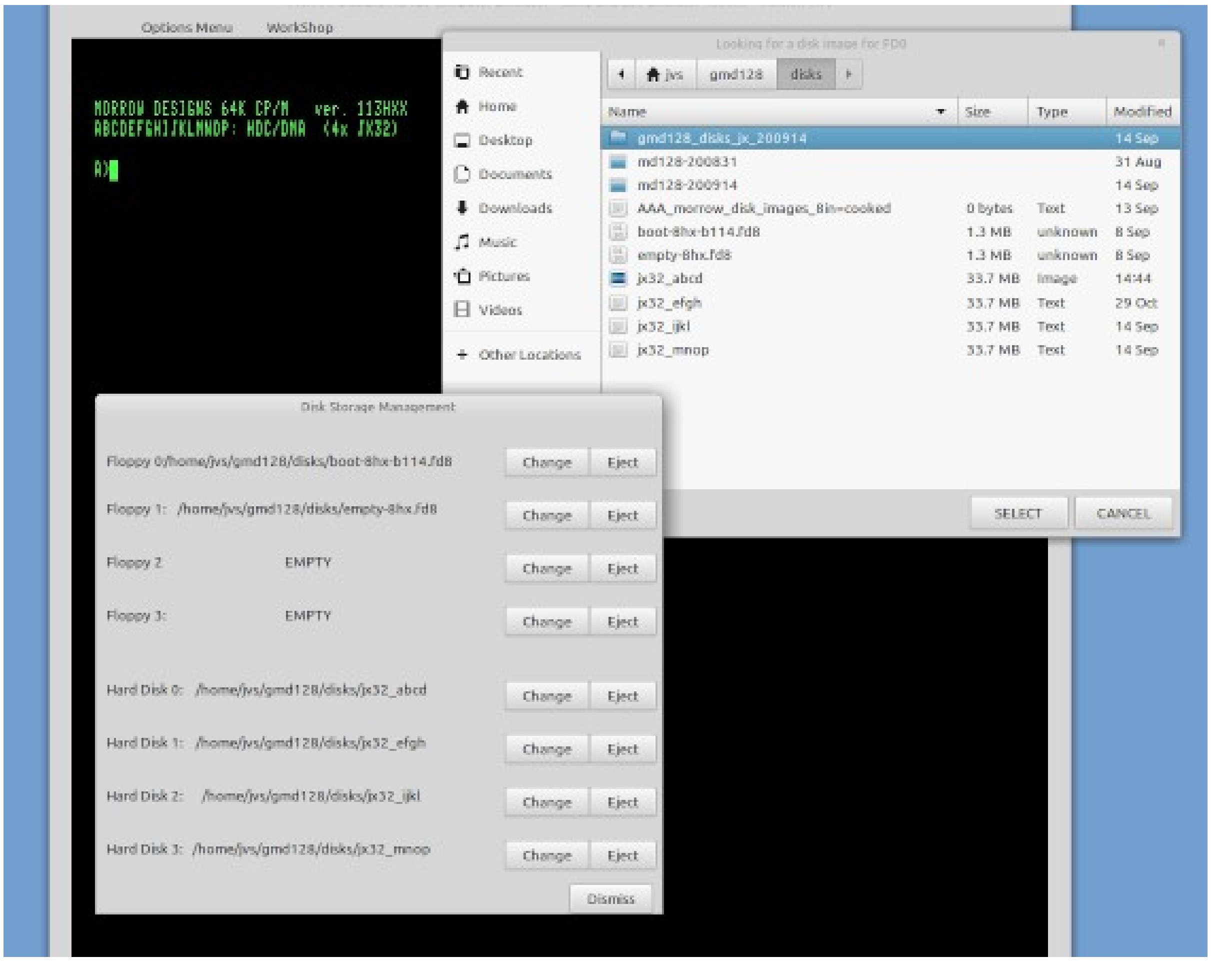
3.1 Disk Management.

The Disk Management menu item allows the user to ‘eject’ floppies and hard drives from the Emulator. The first displayed window shows what floppy disks happen to be ‘inserted’ in Floppy 1 or in Floppy 2. It also shows which hard drives were installed when the Emulator was booted.

Each disk-drive has two buttons: a ‘Change’ button which will install a different floppy-image or hard-disk image. And an ‘Eject’ button which removes any image-file which was previously installed.

If the ‘Change’ button is hit, a file-chooser dialog window opens and allows the user to browse through the whole file-system looking for a floppy-disk image to install. Once the file is selected, hit the ‘Select’ button to confirm your choice. The file-chooser window will close, the floppy-image is ‘inserted’ into the selected floppy-drive and is then ready for use.

The directory which the floppy-image came from will be used as the default disk directory in future disk-image searches. For this reason it is handy to store all your emulator floppy and hard-drive image-files in one or two directories.



```
MORROW DESIGNS 64K CP/M   ver. 113HXX
ABCDEFGHIJKLMNOP: HDC/DMA  (4x JX32)

A>
```

Looking for a disk image for FD0

jvs / gmd128 / disks

Name	Size	Type	Modified
gmd128_disks_jx_200914			14 Sep
md128-200831			31 Aug
md128-200914			14 Sep
AAA_morrow_disk_images_8in=cooked	0 bytes	Text	13 Sep
boot-8hx-b114.fd8	1.3 MB	unknown	8 Sep
empty-8hx.fd8	1.3 MB	unknown	8 Sep
jx32_abcd	33.7 MB	Image	14:44
jx32_efgh	33.7 MB	Text	29 Oct
jx32_ijkl	33.7 MB	Text	14 Sep
jx32_mnop	33.7 MB	Text	14 Sep

Disk Storage Management

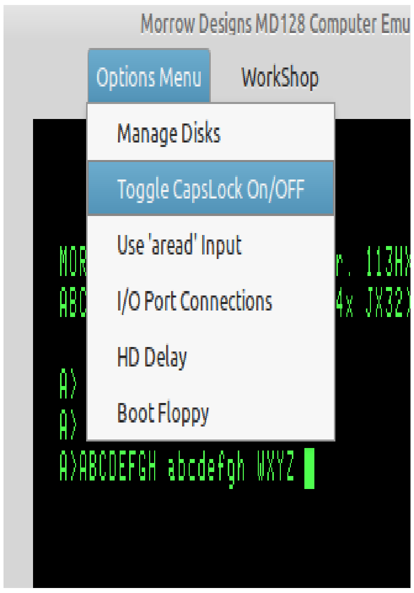
Floppy 0:	/home/jvs/gmd128/disks/boot-8hx-b114.fd8
Floppy 1:	/home/jvs/gmd128/disks/empty-8hx.fd8
Floppy 2:	EMPTY
Floppy 3:	EMPTY
Hard Disk 0:	/home/jvs/gmd128/disks/jx32_abcd
Hard Disk 1:	/home/jvs/gmd128/disks/jx32_efgh
Hard Disk 2:	/home/jvs/gmd128/disks/jx32_ijkl
Hard Disk 3:	/home/jvs/gmd128/disks/jx32_mnop

Fig. 7.   Disk Selection Pop-Ups

3.2 Toggle Capslock ON and OFF

Many of the older Operating Systems will not recognise the use of lower-case characters. While one can use the actual Caps Lock Key to turn on the CapsLock, it would also turn on upper-case for the host Operating System as well. This can be a nuisance.

North Star DOS only understands uppercase commands, so it's necessary to toggle Capslock ON when using DOS. CP/M automatically converts command-line lowercase to uppercase anyway, so the Capslock setting can be set to personal preference.



```
MOR                          r. 113H
ABC                          4x JX32

A>
A>
A>ABCDEFGH abcdefgh WXYZ
```

Fig 8.   Capslock Toggled ON/OFF

In the screenshot above, Capslock starts out as being ON, showing the 'ABCDEFGH' in uppercase. Then Capslock is toggled to OFF, as shown in the Status Window, with the next set of characters being lowercase 'abcdefgh'. The Capslock is then toggled back ON, again showing in the Status Window, and the final 'WXYZ' is again uppercase

3.3 Use 'aread' Input.

Read in an ASCII file from disk instead of having to type it all in manually. The ASCII file is read in line-by-line until it has all been entered. The keyboard then waits for user input, as it does normally

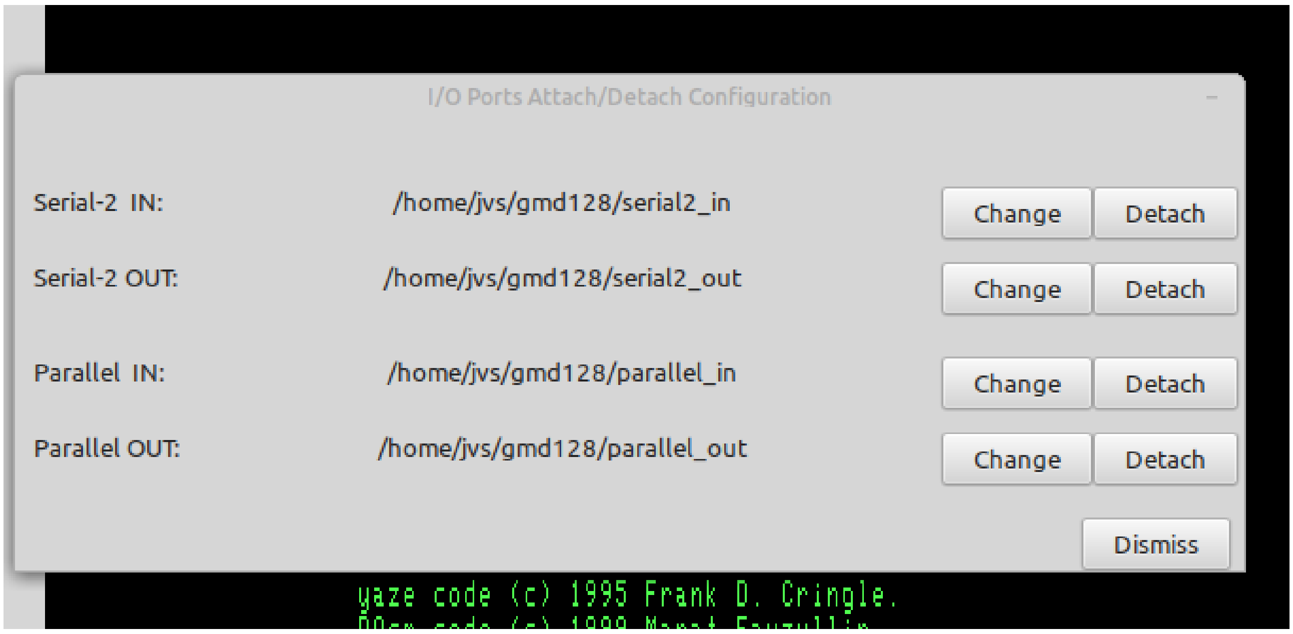
The Input File is selected with a file-chooser window. It is read in immediately after being selected.

Files read in with 'aread' will be processed in exactly the same way as they would if typed in at the keyboard. Excessively long lines will be rejected by the command-line processor of some operating systems, WordStar can 'choke' temporarily because it is unable to keep up with the faster input, but it usually recovers well.

3.3 Toggle HD Delay ON/OFF

GMD128's 20 x speed emulation of the floppy disk drives and the hard disk drives is extremely fast! Even the 'slow' hard drive speed is very fast. It's unlikely that you will want to slow down the hard drive. The default speed is 'Fast'.

3.4 Allocate I/O Port Files


I/O Ports Attach/Detach Configuration

Serial-2 IN:	/home/jvs/gmd128/serial2_in	Change	Detach
Serial-2 OUT:	/home/jvs/gmd128/serial2_out	Change	Detach
Parallel IN:	/home/jvs/gmd128/parallel_in	Change	Detach
Parallel OUT:	/home/jvs/gmd128/parallel_out	Change	Detach

Dismiss

```
yaze code (c) 1995 Frank D. Cringle.
```

Fig 9. Allocating I/O Files to the gmd128 I/O Ports

Attach or detach a unix file to or from a gmd128 I/O Port. There is a parallel I/O port. And there is a serial I/O port. In unix, everything is a file so one unix file or pipe is attached to the second serial-in port, and another to the second serial-out port.

Example: The ‘List’ device is allocated to the parallel port. Anything sent to the ‘List’ device will therefore show up as data in the file attached to the parallel output port.

3.5 TEXT COLOR OF THE EMULATOR OUTPUT

A selection of colors is available for the screen display. As this is pretty much a ‘set and forget forever’ option, it was decided against having a color-selection window as one of the ‘Options Menu’ items. To make a change it is simply a matter of selecting suitable values for the 24-bit RGB components, RED_LEVEL, BLUE_LEVEL, GREEN_LEVEL in the ‘gxe.h’ file and recompiling. Some examples -

Green on Black: (as default)

RED_LEVEL	0x3F
GREEN_LEVEL	0xFF
BLUE_LEVEL	0x3F

Amber on Black:

RED_LEVEL	0xFF
GREEN_LEVEL	0xBF
BLUE_LEVEL	0x3F

Yellow on Black:

RED_LEVEL	0xFF
GREEN_LEVEL	0xFF
BLUE_LEVEL	0x3F

White on Black:

RED_LEVEL	0xFF
GREEN_LEVEL	0xFF
BLUE_LEVEL	0xFF

4.0 GMD128 DEVELOPMENT ASSISTANCE

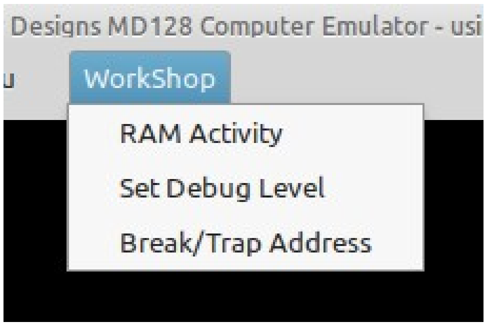
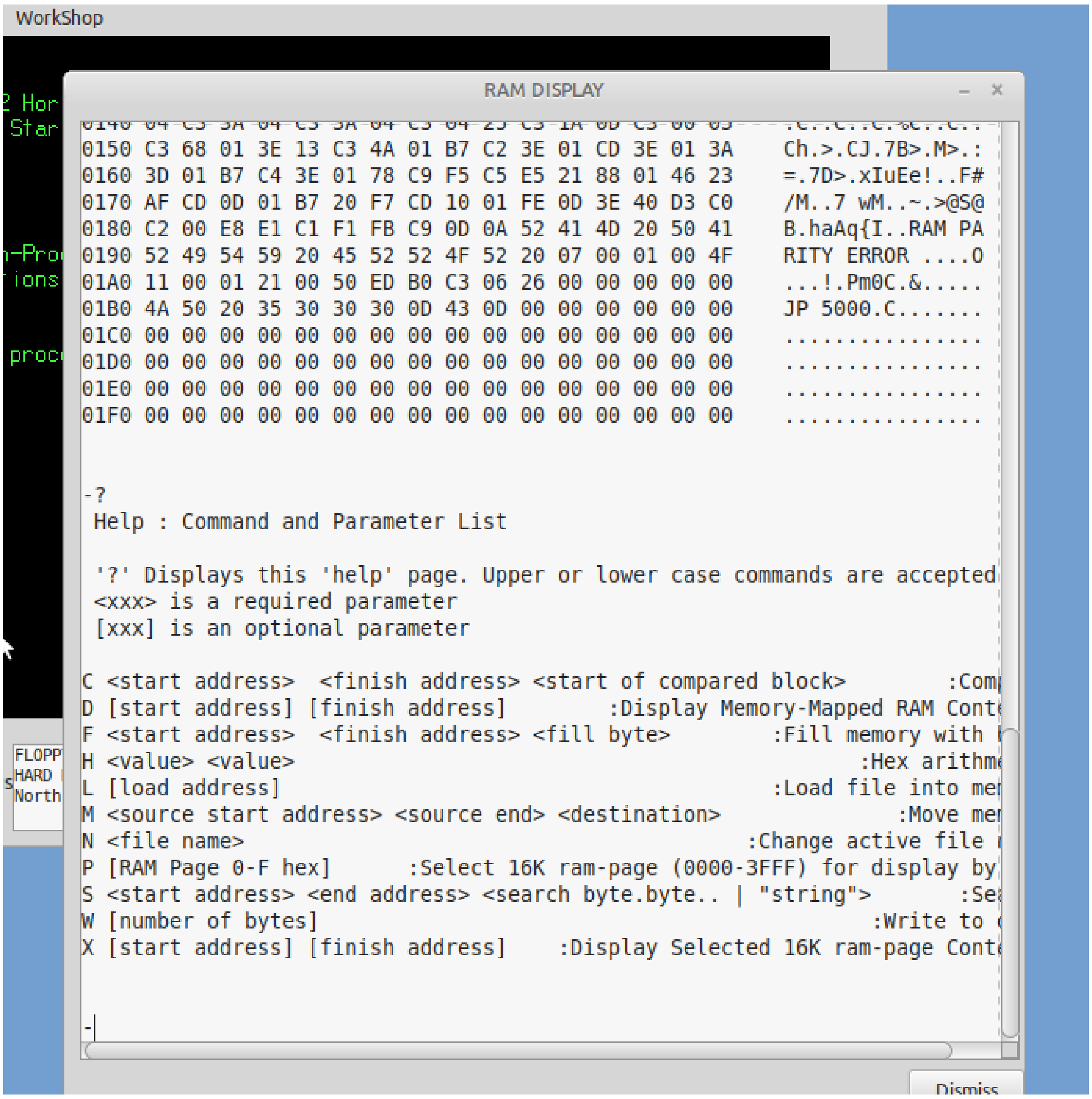


Fig 10. GMD128 Development menu: 'WorkShop'

4.1 Display RAM in the Morrow gmd128 virtual machine.



```
0150 C3 68 01 3E 13 C3 4A 01 B7 C2 3E 01 CD 3E 01 3A    Ch.>.CJ.7B>.M>.:
0160 3D 01 B7 C4 3E 01 78 C9 F5 C5 E5 21 88 01 46 23    =.7D>.xIuEe!..F#
0170 AF CD 0D 01 B7 20 F7 CD 10 01 FE 0D 3E 40 D3 C0    /M..7 wM..~.>@S@
0180 C2 00 E8 E1 C1 F1 FB C9 0D 0A 52 41 4D 20 50 41    B.haAq{I..RAM PA
0190 52 49 54 59 20 45 52 52 4F 52 20 07 00 01 00 4F    RITY ERROR ....O
01A0 11 00 01 21 00 50 ED B0 C3 06 26 00 00 00 00 00    ...!.Pm0C.&.....
01B0 4A 50 20 35 30 30 30 0D 43 0D 00 00 00 00 00 00    JP 5000.C.......
01C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
01D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
01E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
01F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................


-?
 Help : Command and Parameter List

 '?' Displays this 'help' page. Upper or lower case commands are accepted
 <xxx> is a required parameter
 [xxx] is an optional parameter

C <start address>  <finish address> <start of compared block>        :Comp
D [start address] [finish address]         :Display Memory-Mapped RAM Conte
F <start address>  <finish address> <fill byte>        :Fill memory with
H <value> <value>                                              :Hex arithme
L [load address]                                       :Load file into mem
M <source start address> <source end> <destination>          :Move mem
N <file name>                                        :Change active file
P [RAM Page 0-F hex]      :Select 16K ram-page (0000-3FFF) for display by
S <start address> <end address> <search byte.byte.. | "string">       :Sea
W [number of bytes]                                             :Write to
X [start address] [finish address]    :Display Selected 16K ram-page Conte


-|
```

Fig 11. Display RAM Dialog

This subsystem has usage similar to CP/M 'DDT' or MSDOS 'DEBUG' Commands.
Upper or lower case commands are accepted
<xxx> is a required parameter, [xxx] is an optional parameter

compare
C <start address> <finish address> <start of compared block>
c 1a00 2000 2a00
Compare two equal-length blocks of memory. Only the bytes which are different will be displayed with location and values.

display
D [start address] [finish address]
d 0 12FF

Display the block of memory selected, showing bytes as hexadecimal and ASCII. If no start and end address specified, the command will continue for 100 H bytes from where it ended last.

examine/substitute
E <start address>
E 2CFF

Examine/change values at memory locations. The operation is stopped when no new value is entered, just a plain 'enter'.

fill
F <start address> <finish address> <fill byte>
f 1000 2000 55

Fill a block of memory with byte-value specified by <fill byte>.

hex
H <value> <value>
h 1267 abcd

Hex arithmetic results of the addition of two values and the subtraction of the second value from the first value.

load
L [load address]
l 2a00

Load the file (previously specified by the 'N' command) into memory. If a load-address is not specified the file will be loaded into location 0000 H.

move
M <source start address> <source end> <destination>
M 4d00 5000 6d00

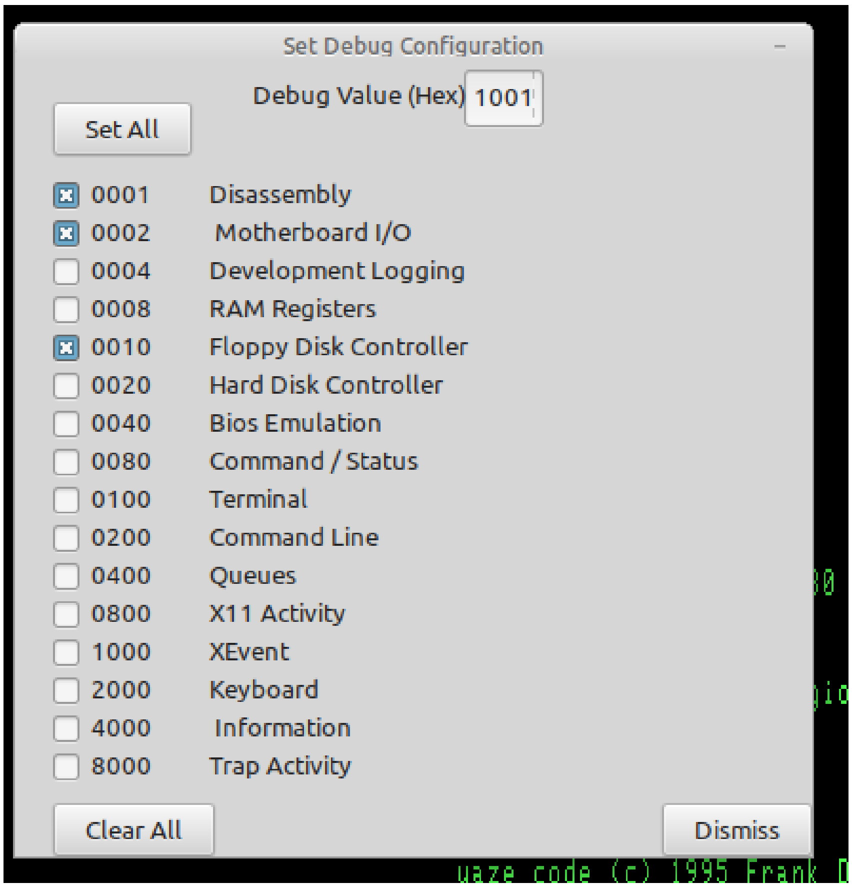
Move the block of memory specified by the block's start and end into memory beginning at the destination address.

name
N <file name>
N xtest.bin.bas

Change active file-name which specifies which unix file will be used for 'load' and 'write' operations.

quit
Q
Quit from the RAM display subsystem back to the emulator's control console.

4.2 Setting the Debug Parameters for the 'xlog' Debugging File Output


Set Debug Configuration

Debug Value (Hex) 1001

Set All

☒	0001	Disassembly
☒	0002	Motherboard I/O
☐	0004	Development Logging
☐	0008	RAM Registers
☒	0010	Floppy Disk Controller
☐	0020	Hard Disk Controller
☐	0040	Bios Emulation
☐	0080	Command / Status
☐	0100	Terminal
☐	0200	Command Line
☐	0400	Queues
☐	0800	X11 Activity
☐	1000	XEvent
☐	2000	Keyboard
☐	4000	Information
☐	8000	Trap Activity

Clear All

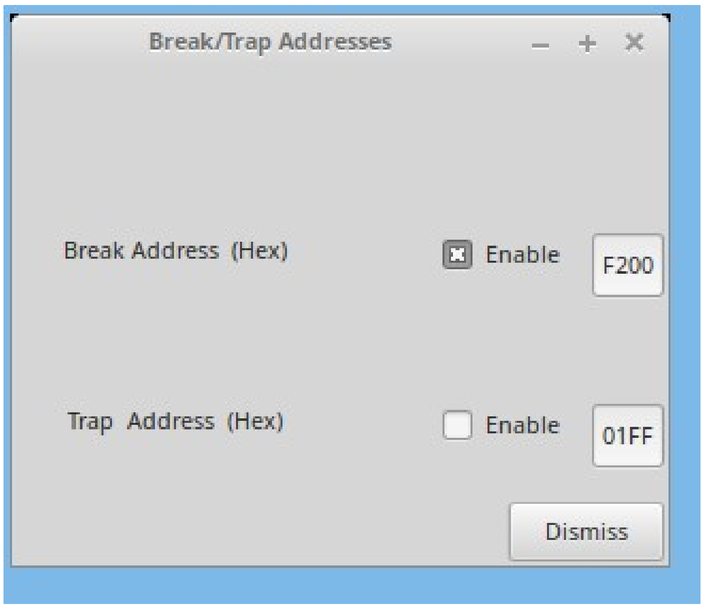
Dismiss

uaze code (c) 1995 Frank D

Fig 12. Setting Debug Logging Parameters

Depending on which items are selected for debug logging, *a lot* of logging output can be produced. Take care that your filesystem does not get over-filled.

4.3 Setting Execution Breakpoint Address, and Trap Address


Break/Trap Addresses

Break Address (Hex)	☒ Enable	F200
Trap Address (Hex)	☐ Enable	01FF

Dismiss

Fig 13. Enabling and Setting Break And Trap Addresses
break

Set a breakpoint address to stop the emulator at a pre-specified address. This is equivalent to the “PAUSE” button, but it occurs at a desired execution address. The contents of the RAM can then be examined by using the Ram Display functions. Hitting the ‘go’ button will resume execution from that breakpoint address and it will continue until that breakpoint address is again reached, unless the breakpoint is disabled while execution is stopped.

trap

Set a trap address to stop the emulator, perform a user-specified unix operation, return to the emulator and continue.

A dummy ‘trap’ function is included in the emulator source (trap.c) which merely prints the trap address and the register values. The trap function could be used to access parts of the host unix system or perform any other required operation.

Both the ‘break’ and ‘trap’ functions are enabled and disabled by the Check Buttons associated.

4.4 Log the debug information to Unix Disk File.

Automatically sends debugging/information output to the ‘xlog’ unix file.Take care, because the quantity of information sent to the log file can reach the maximum size (2 Gig in 32-bit systems, whole disk or whole filesystem in 64-bit systems) within a fairly short time.

Unless you're doing development on the GMD128 Emulator itself, it probably will not be useful to use any debug logging at all.

4.5 Log the Screen Output to Unix Disk File.

Automatically sends all ASCII screen text output to the ‘screenlog’ unix file. This can be handy to refer to if text output scrolls off the top of the screen before you can read it.

5. HELPER PROGRAMS

5.1 backup.gmd128

Run **backup.gmd128** from the work directory. This will look in the **disks** subdirectory for the four jx32 hard drive images (jx32_abcd.img, jx32_efgh.img, jx32_ijkl.img, jx32_mnop.img) and extract the individual files on the CP/M disk drives into directories designated A, B, C, ….. O, P. Each time the utility is run, a complete set of backup drives is produced and placed into a folder named **gmd128-<date of backup>** thus **gmd128-200914** or **gmd128-201130** and so forth

5.2 jx32_sysgen

jx32_sysgen is used to transfer the system boot track from one hard drive image file to another.

```
                JX32-SYSGEN

     Installs a bootable system track on a JX32 hard
     drive. The system may be copied from a hard drive
     which already contains a bootable system track.
     Otherwise this program will supply a suitable
     bootable system.


     Enter SOURCE hard drive filename or else 'Enter' to
       install the default system : <ENTER KEY>

     Using default version of boot system
```

If instead of just hitting the ‘Enter’ key, a hard drive image filename is given, the boot track will be taken from that hard disk image;

```
     Enter SOURCE hard drive filename or else 'Enter' to
       install the default system : disks/jx32_abcd.img

     Getting system track from hard drive "disks/jx32_abcd.img"
```

Having obtained the boot track, now give the DESTINATION hard drive image filename:

```
     Enter DESTINATION hard drive filename or else 'Enter' to
       exit this program : disks/jx32_efgh.img

     Putting system track on hard drive "disks/jx32_efgh.img"

      Enter DESTINATION hard drive filename or else 'Enter' to
       exit this program :
```

After writing the boot track to the destination hard drive image, the program will request more destination disks. If you don’t want to write to any other disk drives, just hit the ‘Enter’ key to finish.

5.3 mkmd128floppy

mkmd128floppy can generate a new blank floppy diskimage. Though I admit I find it easier to just copy a blank floppy image using the Linux **cp** utility.

```
MD11 Virtual Floppy Disk Maker and Formatter
 (c) Jack Strangio 2014

This program prepares 8" and 5" Virtual Floppy Disk Files

The floppies can be formatted Double-Density containing 16 512-byte
or 8 1024-byte sectors per track, or Single-Density containing 26 128-byte
sectors per track. Either a single-side or both sides of the disk
can be formatted.

Is this a 5" or an 8" disk? 8
Is this a Double-Density Disk? y
Is this a Double-Sided Disk? y
 128-byte (A), 512-byte (B),  or 1024-byte (C) sectors?  B
Sector Type = B,  16  512-byte sectors. track_code = 90.   77 tracks

Input disk name (omitting '.fd-type'  suffix): mynewfloppy
Formatting disk: "mynewfloppy.fd8"
```

5.4 jdz80 (Z80 disassembler)

jdz80 is a slightly improved version of Marat Fayzullin's 1999 DAsm, in which relative jump destination addresses are calculated and displayed rather than just displaying the relative jump offsets.

5.5 cpmtools

Life is simpler with cpmtools-2.7 (or later) which can be obtained from most linux repositories. This set of utilities can be used to copy files directly between CP/M disk-images in the .fd8 and JX32 formats and the unix/linux file space. It will be necessary to add the disk definitions specified in the **gmd128_diskdefs** file to the cpmtools config-file **diskdefs** which is usually at /etc/cpmtools/diskdefs.

The added disk-definitions will enable cpmtools to understand the GMD128 CP/M disk formats, both the floppy-disk images and the larger CP/M Virtual Disk Images on the hard disk.

The utilities in cpmtools include:

cpmls	list files in the North Star CP/M disk-image
cpmcp	copy files to and from the North Star CP/M disk-image
cpmrm	delete files from the North Star CP/M disk image
mkfs.cpm	prepare stub disk for CP/M. In my experience, this does not work properly. Instead, use **mkmd128floppy** to produce a floppy disk then FORMAT it for CP/M.

5.6 screenlog

screenlog is not a tool as such but a record of GMD128's screen output.

5.7 xlog

xlog is not a tool but is a record of all debugging information. Can make very large log files.

6.1 OTHER FILES REQUIRED

Various floppy-disk image files:

These are available from various sources on the Internet.

6.2 COMPILING LIBRARIES REQUIRED

The linux libraries required are GTK+ version 3

6.3 VARIOUS USEFUL MANUALS

Most of the manuals are available from https://itelsoft.com.au.

Probably the most useful are:

This gmd128 User Guide
CPM 2.2 Manual

These are all included in the 'documentation' directory

6.4 BUGS

I feel I have got many bugs out, which makes GMD128 very usable. But there are still a few to go, apart from the things that could be done to make GMD128 not quite so rough-edged. It certainly is not yet anywhere near as elegant as I would like, and the fault-lines between the several programs that GMD128 is based upon are still very visible. Please inform me of any bugs that you discover. Email me at: jackstrangio@yahoo.com

6.5 AUTHOR and SUPPORT

Jack Strangio <jackstrangio@yahoo.com>

Website: `https://itelsoft.com.au`

APPENDIX A.

GCPZ & GMD128 EMULATORS:
STRUCTURE OF THE .fd8 FLOPPY-DISK IMAGE FILE

These emulators can run up to four eight-inch floppy drives. These drives simulate an 8" double-sided double-density floppy. They contain 77 tracks on each side. On track 0, side 0, the sectors are formatted to be 26 sectors of 128 byes, all the rest of the tracks on both sides are formatted to 16 sectors of 512 bytes. This gives a total of about 1.2megs per disk.

The disk image is laid out as follows: every track is allocated 8K (16x512 or 0x2000 bytes), whether it contains 26 sectors of 128 bytes (0xD00 bytes) , or 16 sectors of 512 bytes (0x2000 bytes). The disk-image is built up of Track0, side 0; followed by Track 0, Side 1; then Track 1, Side 0; Track 1, Side 1, etc.

```
                           Side 0                                                  Side 1
Track  0:   S S S S S S S S S S S S S S S S S S S S S S S S S S         DD  DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD
Track  1:   DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD             DD  DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD

   Tracks Omitted ........

Track 75:   DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD             DD  DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD
Track 76:   DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD             DD  DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD
```

Because the disk-image is laid out regularly, the unix cpmtools can be used quite well to copy files to and from disk-images, using the cpmtools disk_defs supplied with the two emulators. Append the disk_defs supplied to the file /etc/cpmtools/diskdefs using any suitable text editor.

System tracks: With the .fd8 disk-images, the system tracks take up Track 0, Side 0 plus one other track. If the disk is single-sided the systems tracks are Track 0, Side 0 and Track 1, Side 0. If the disk is double-sided, the systems tracks are Track 0, Side 0 and Track 0, Side 1.

APPENDIX B. MORROW HARD DISKS WITH MULTIPLE PARTITIONS

PHYSICAL FORMAT OF MORROW HARD DRIVES:

Each track contains nine 1024-byte sectors.
Each cylinder contains eight tracks (8 heads).

Eight tracks will give (8x9) 1024-byte sectors per cylinder, making a total of 72 KB per cylinder.

HARD DISKS HOLDING 7 x 4MB CP/M DRIVES:

For 4MB CP/M drives (4096k) we need 4096/72 = 57 cylinders
57 cylinders x 7 drives (or partitions) = 399 cylinders + 1x system cylinder = 400 cylinders

HARD DISKS HOLDING 4 x 8MB CP/M DRIVES (JX_32_Drives):

For 8MB disks (8192k) we need 114 cylinders
in a 32MB disk we need (4*114) + 1 system cylinder = 457 cylinders

BIOS Version B113 PARAMETERS

Standard sectors (128-byte sectors) per cylinder = (72x8) = 576 sectors

Bad sector map at 4800 Hex from beginning of disk image in sector 0018 [0012 Hex] which is the first sector of track 2, on cylinder 0.

The System Track(s) starts on Track 0, on Cylinder 0. The System CP/M BIOS version B113 expects the HDC-DMA hard disk controller to be used

Bios Version B114 PARAMETERS

CP/M BIOS Version B114 boots from an 8-inch floppy disk in floppy drive A: The BIOS will access two floppy drives and two JX_32 hard drives. This BIOS uses both the Disk Jockey DJ-DMA floppy disk controller and the Morrow HDC-DMA hard disk controller. This BIOS is not as useful as the B113 BIOS which gives access to four JX_32 hard drives. It also needs more work to be fully functional.