ZARC CP/M USERS' GUIDE

Written by Merlin Skinner-Oakes, 13/5/2022

Last updated .

1 DESCRIPTION

The Z80 Anachronistic Retro Computer (ZARC) supports CP/M V2.2. This is started from the monitor programme using the following command:

```
> cpm
```

This document describes the customised CP/M BIOS, memory card format and ZARC-specific utilities. Information on CP/M itself is widely available online.

In this document, the following style conventions are used:

```
Machine output
User commands
Comments
```

2 BOOT PROCESS

The monitor programme implements the boot process. This is started by the user entering the following command:

```
> cpm
ZARC CP/M V2.2 BIOS V1.1

A>
```

The overall flow is as follows:

1) Load the master boot record (MBR) from sector 0.
2) Verify boot signature is present (0x55 0xaa at 0x01be).
3) Scan the partition table, looking for an "experimental" partition, which ZARC uses for CP/M. *$_1$
4) Allocate four memory pages ""CPM0" to "CPM3" to provide a 64 kB space.
5) Load the BIOS from the start of the partition.
6) Verify that it is a ZARC BIOS.

7) Execute it.

*$_1$ no official CP/M 2.2 partition type is coded, so this seems the most sensible choice for this application.

3 MEMORY MAP

While CP/M is active, memory is mapped as follows:

```
0 (0000 - 3FFF): 00 [CPM0]   writeable
1 (4000 - 7FFF): 01 [CPM1]   writeable
2 (8000 - BFFF): 02 [CPM2]   writeable
3 (C000 - FFFF): 03 [CPM3]   writeable, supervisor
```

Note that bank 3 is the only one with I/O (supervisor) access. This contains the BIOS, and is the one bank is maintained when calls are made to the monitor. For this reason, interrupt code is maintained in this bank. Page numbers for CP/M may vary to avoid clashes with other applications, and are allocated using the monitor's mpall function.

When the monitor is active for debugging or when its support routines are called, the map is modified:

```
0 (0000 - 3FFF): 3f [MONP]   read only, supervisor
1 (4000 - 7FFF): 3e [MOND]   writeable
2 (8000 - BFFF): (not used)
3 (C000 - FFFF): 03 [CPM3]   writeable
```

Pages are mapped into bank 2 (8000 - BFFF) as required to allow access to the full CP/M address space.

4 CP/M BIOS

This is based on details in Digital Research CP/M Operating System Manual, especially appendix A, "The MDS Basic I/O System (BIOS)" and appendix B, "A Skeletal CBIOS", both available at:

http://www.gaby.de/cpm/manuals/archive/cpm22htm/.

4.1 Extended Functions

In addition to CP/M 2.2 BIOS functions, there are some additional ones. These are defined in cpm.i. Applications must check for the presence of the ZARC BIOS before using extended functions. The function zrcdet in cpm_utility.z80 is a convenient method.

Jump Table Entry	*Function*
BIOS_ID	"ZRC"
BIOS_VER	<reserved>, <major>, <minor>
startup	ZARC CP/M startup entry point
mon_call	Call monitor function
set_super	Set supervisor state
int_disable	Disable interrupt and keep count
int_enable	Enable interrupts if safe.
tird	Read time and date
set_timer	Set elapsed time
get_timer	Get elapsed timer

The following sub-sections provide an overview of each function. For full details such as register usage, see the source file bios.z80.

4.1.1 BIOS_ID

The first extended entry is simply the three-byte string “ZRC”. This identifies the BIOS are intended for ZARC, and may be used as a rationalness check before relying on ZARC-only functions. Three bytes are used to maintain alignment with the other jump table entries.

4.1.2 BIOS_VER

The version is reported as follows:

Byte	*Function*
0	Reserved (zero). May be used to flag capabilities in the future.
1	Major version
2	Minor version

Three bytes are used to maintain alignment with the other jump table entries.

4.1.3 startup

startup is an entry point to the BIOS, and is used to pass initial parameters such as memory mapping information from the monitor programme to the BIOS. See source files bios.z80 and its counterpart cpm.z80 for details.

4.1.4 mon_call

This routine allows CP/M applications to call some monitor functions. Any function that accesses memory must be used with caution as banks 0 and 1 will be changed to select the monitor's environment.

4.1.5 set_super

Conventional CP/M programmes exclusively use the BIOS to access I/O. However, it is possible to write code that directly accesses the I/O space. Without supervisor mode, any attempt to access I/O will result in a NMI and subsequent return to the monitor. In order to circumvent this, CP/M programmes should call this function to enable or disable supervisor access for banks 0 to 2 as required. Bank 3 (0xc000 to 0xffff) is left unchanged as the BIOS always requires supervisor mode in order to function.

4.1.6 int_disable and int_enable

Nested interrupt enable / disable pairs present an issue in that the interrupts are enabled after the first enable is encountered, which is likely unsafe. Use of int_disable and int_enable avoid this as interrupts are only enabled when the outermost enable is executed. This is achieved by maintaining a record of the disable count.

4.1.7 tird

Reads time and date as a single 32-bit integer. This is similar to the monitor's routine of the same name, but this version reads the interrupt maintained time instead. The system keeps track of time as a Unix-style 32-bit number. Unlike Unix, this is treated as an unsigned integer and so will not suffer from the year 2038 problem as it doubles the allowable postive time. It does, however, mean that it is not possible to represent dates before 1970. Leap seconds are ignored. 32-bit time does not include BST adjustment and so always represents GMT. However, the monitor’s tiitos (convert integer time to structure) and tistoi (convert structure time to integer) add or remove the appropriate offset and so describe "clock time" without further adjustment.

4.1.8 set_timer

Set elapsed time as a 32-bit integer. This is incremented by MS_PER_TICK on each tick interrupt to maintain elapsed time in ms.

4.1.9 get_timer

Read elapsed time as a 32-bit integer.

4.2 Console Terminal Translation

Console input and output sequences are translated to provide a simple interface to CP/M applications while using a widely available ANSI / VT-100 terminal or emulator externally.

Translation may be disabled by starting CP/M from the monitor with the "rawcon" option. Translation of console input depends on interrupts, so is disabled if CP/M is started with the "noints" option.

4.2.1 Console Output

Translations performed are:

ZARC Sequence	***VT-100 / ANSI Sequence***	***Function***	***Comment***
Esc A	Esc [A	Cursor up	Move cursor one line upwards.
Esc B	Esc [B	Cursor down	Move cursor one line downwards.
Esc C	Esc [C	Cursor right	Move cursor one column to the right.
Esc D	Esc [D	Cursor left	Move cursor one column to the left.
Esc H	Esc [H	Cursor home	Move cursor to the upper left corner.
Esc J	Esc [J	Clear to end of display	Clear screen from cursor onwards.
Esc K	Esc [K	Clear to end of line	Clear line from cursor onwards.
Esc Y <y> <x>	Esc [<y> ; <x> H	Set cursor position	Move cursor to position <x - 32>, <y - 32>, encoded as single characters. Origin (top left-hand corner) is (0, 0).
Esc p	Esc [7 m	Reverse video on	
Esc q	Esc [27 m	Reverse video off	
Esc r	Esc [4 m	Underline on	
Esc u	Esc [24 m	Underline off	

Note that Esc L (insert line) and Esc M (remove line) are not supported. There is no equivalent VT-100 sequence for these operations.

4.2.2 Console Input

Input escape codes are translated to single characters where appropriate. This is the approach taken by later CP/M machines such as the Amstrad PCW range and Spectrum +3. Translations performed are:

Key	*Sequence*	*ZARC Code*	*Origin*
Cursor up	Esc [A	0x1f (^_)	Amstrad PCW and Spectrum +3
Cursor down	Esc [B	0x1e (^^)	Amstrad PCW and Spectrum +3
Cursor right	Esc [C	0x06 (^F)	Amstrad PCW and Spectrum +3
Cursor left	Esc [D	0x01 (^A)	Amstrad PCW and Spectrum +3
Home	Esc [1 ~	0x02 (^B)	Unique to ZARC
Insert	Esc [2 ~	0x16 (^V)	Unique to ZARC
Delete	Esc [3 ~	0x7f	Common
End	Esc [4 ~	0x04 (^D)	Unique to ZARC
Page up	Esc [5 ~	0x1d (^])	Unique to ZARC
Page down	Esc [6 ~	0x1c (^\)	Unique to ZARC

Unrecognised codes are passed on untranslated, so it is possible for software to communicate with other terminal types without disabling the translation.

4.2.3 Rationale

VT-52 codes were chosen for output as it offers relatively simple control codes and is the basis for a variety of later CP/M machines such as the Amstrad PCW and Spectrum +3.

Few terminal emulators support ADM-3A, and some CP/M programmes cannot support VT-100 sequences as these are relatively complex to generate. Some software such as BBC Basic cannot generate these codes as the space allocated to customisation is too small.

ADM-3A codes are initially attractive as earlier Kaypro machines mimic this. Later ones (“2.2G” and later) use different cursor control codes. These include ^S, so may interfere with XON / XOFF flow control.

Console input is translated to single character codes for ease of customisation. This appears to be what the Spectrum +3 and Amstrad PCW range used.

5 MEMORY CARD FORMAT

This section describes the format of data on the memory card.

5.1 CP/M Partition

The space used by the CP/M system is located in a single 17 MB partition partition on the card. A. This allows for 16 1 MB disks with an extra (very generous) 1 MB for CP/M's BDOS and CCP. Note that one large disk would be hard to organise given there are no sub-directories in CP/M.

The partition table may be loaded and examined using the monitor.

```
Setup memory mapping.
> map 0 0                    Map page 0 to bank 0
> set mmap 1                 Ensure memory mapping is enabled
> readsec 0 1 0000           Read the boot sector
> dump 0000 0200
0000  FA B8 00 10 8E D0 BC 00  |........|
0008  B0 B8 00 00 8E D8 8E C0  |........|
0010  FB BE 00 7C BF 00 06 B9  |...|....|
0018  00 02 F3 A4 EA 21 06 00  |.....!..|
0020  00 BE BE 07 38 04 75 0B  |....8.u.|
0028  83 C6 10 81 FE FE 07 75  |.......u|
0030  F3 EB 16 B4 02 B0 01 BB  |........|
0038  00 7C B2 80 8A 74 01 8B  |.|...t..|
0040  4C 02 CD 13 EA 00 7C 00  |L.....|.|
0048  00 EB FE 00 00 00 00 00  |........|
0050  00 00 00 00 00 00 00 00  |........|
                ...
01B8  06 97 85 77 00 00 00 04  |...w....|
01C0  01 04 7F 47 02 48 00 08  |...G.H..|
01C8  00 00 00 88 00 00 00 00  |........|
01D0  00 00 00 00 00 00 00 00  |........|
                ...
01F8  00 00 00 00 00 00 55 AA  |......U.|
```

01fe is the boot signature (0x55 0xaa).

0x01BE is the start of partition entry no. 1

Partition type (0x7f) is manually edited by me (readsec, modify, then writesec).

Red - LBA of first absolute sector in the partition (0x00000800 [2048d] or 1024 KB [1 MB])

Green - number of sectors in partition (0x00008800 [34816d] sectors or 17408 KB [17 MB])

Hence start sector for the CP/M partition is 0x0800 (2048d).

5.2 CP/M Disk Parameter Block (DPB)

This CP/M structure defines the disk geometry. The ZARC BIOS presents 16 disks, all with the same BPB:

Data	*CP/M Name*	*Description*
0100	SPT	sectors per track
04	BSH	data allocation block shift factor
0F	BLM	data allocation block mask (2^[BSH-1])
00	EXM	extent mask
01FF	DSM	maximum block number
01FF	DRM	maximum directory entry number
FF, 00	AL0, AL1	determine reserved directory blocks
0000	CKS	size of the directory check vector
Varies	OFF	number of reserved tracks

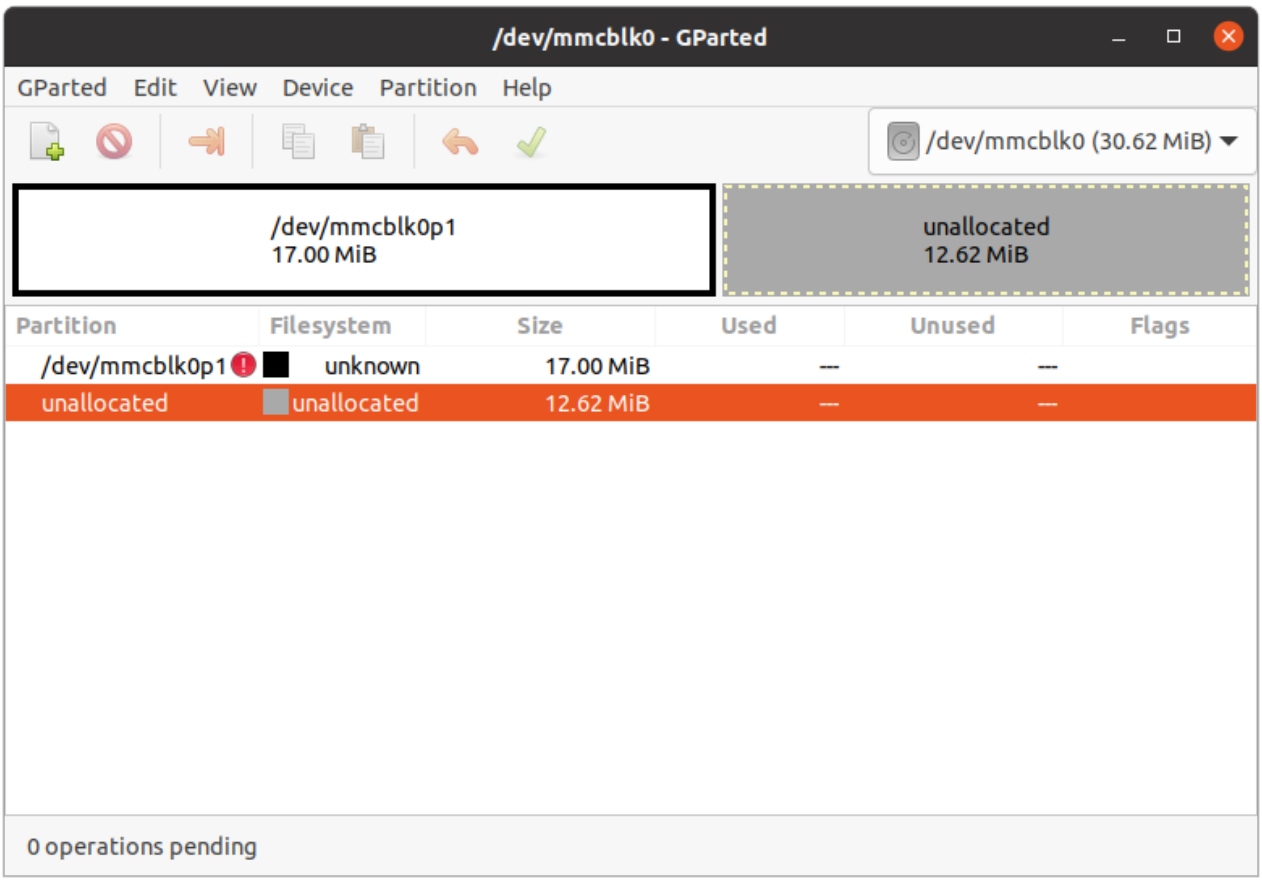
OFF is different for each disk. It is used to define the start of the disk within the partition.

6 INSTALLATION

This section describes how to initialise an memory card and bring up the CP/M system.

6.1 Creating the Partition

The required partition was created using GParted on a Linux system.



Unfortunately, it seems common, if not universal, on PC systems to start the first partition 1 MB from the start of the disk. Manual editing could overcome this, but it doesn't seem worthwhile, in part due to the complications of the CHS addressing system.

GParted cannot create the correct partition type, so this was changed using the monitor:

```
Setup memory mapping.
> map 0 0                   Map page 0 to bank 0
> set mmap 1                Ensure memory mapping is enabled
> readsec 0 1 0000          Read the boot sector
> write 01c2 7f
> dump 01C0 10
01C0  01 04 7F 47 02 48 00 08  |...G.H..|
01C8  00 00 00 88 00 00 00 00  |........|
> writesec 0 1 c000         Write the modified boot sector
```

6.2 Installing the BIOS, BDOS and CCP

The BIOS, BDOS and CCP must be written to the correct locations in the partition. The following steps achieve this. Note that commands entered at the "$" prompt are to be issued from the Linux system.

The components of CP/M are downloaded using a script called zarc_dl. This overcomes the limitations of the USB serial adaptor used to connect the PC to ZARC's console port.

6.2.1 Install BIOS

Follow these steps to install or update the BIOS:

```
Setup memory mapping.
> map 3 3                   Map page 3 to bank 3
> set mmap 1                Ensure memory mapping is enabled
> bindownload ec00          Download to BIOS start
$ zarc_dl bios.cim /dev/ttyUSB1
[reset]
> writesec 2059 8 ec00      Write to memory card
```

6.2.2 Install BDOS and CCP

ccp.bin is 2048 bytes (16 CP/M sectors) – loaded to 0xd600

bdos.bin is 3584 bytes (28 CP/M sectors) – loaded to 0xde00

This is 5632 bytes (0x1600) or 11 MMC sectors in total.

Follow these steps to install or update the BDOS and CCP.

```
Setup memory mapping.
> map 3 3                   Map page 3 to bank 3
> set mmap 1                Ensure memory mapping is enabled
> bindownload d600          Download to CCP start
```

```
The BDOS immediately follows the CCP, so we can transfer it in
one tranfer.
$ zarc_dl ccp.bin /dev/ttyUSB1
$ zarc_dl bdos.bin /dev/ttyUSB1
[reset]
> dump d600                    CCP
D600  C3 5C D9 C3 58 D9 7F 00  |.\..X...|
D608  20 20 20 20 20 20 20 20  |        |
 ...
D618  43 4F 50 59 52 49 47 48  |COPYRIGH|
D620  54 20 28 43 29 20 31 39  |T (C) 19|
D628  37 39 2C 20 44 49 47 49  |79, DIGI|
D630  54 41 4C 20 52 45 53 45  |TAL RESE|
D638  41 52 43 48 20 20 00 00  |ARCH  ..|
> dump de00                    BDOS
DE00  00 00 00 00 00 00 C3 11  |........|
DE08  DE 99 DE A5 DE AB DE B1  |........|
DE10  DE EB 22 43 E1 EB 7B 32  |.."C..{2|
DE18  D6 EB 21 00 00 22 45 E1  |..!.."E.|
DE20  39 22 0F E1 31 41 E1 AF  |9"..1A..|
DE28  32 E0 EB 32 DE EB 21 74  |2..2..!t|
DE30  EB E5 79 FE 29 D0 4B 21  |..y.).K!|
DE38  47 DE 5F 16 00 19 19 5E  |G._....^|
> writesec 2048 11 d600      Write to memory card
```

6.2.3 “Format” drive A

No low-level formatting is required as the sectors are already defined on the memory card. However, it is necessary to mark the directory entry blocks as “unused”. The remainder of the drives may be formatted from within CP/M using the format command.

```
Setup memory mapping.
> map 0 0                      Map page 0 to bank 0
> set mmap 1                   Ensure memory mapping is enabled
> fill 0 4000 e5               Ensure memory mapping is enabled
> writesec 4096 32 0           Write to memory card
```

The values above assume 512 directory entities (DPB.DRM is 511). If this value were changed, the directory size would be different. The required size may be calculated:

Number of entries (extents) = DPB.DRM + 1 = 512

Size in bytes: 512 x 32 = 16384 (0x4000)

Number of memory card sectors = 16384 / 512 = 32

6.3 Initial File Transfer

At this point, the CP/M system can be started:

```
> cpm
ZARC CP/M V2.2 BIOS V1.1

A>
```

If the system is newly installed, there will be no commands or utilities available beyond those built into the CCP. It is possible to transfer files using the monitor:

```
> tpadownload                Start binary download to TPA
$ zarc_dl <file> /dev/ttyUSB1
[reset]
> cpm                        Start CP/M
ZARC CP/M V2.2 BIOS V1.1
A>save <size> <CP/M filename>
```

<size> is the number of 256 byte pages to save.

A reasonable first file to transfer is the ZARC utility STRAN.COM, as this allows easier and more reliable file transfer.

7 FORMATTING THE REMAINING DISKS

The ZARC utility FORMAT is recommended for this. It formats a "drive" on the MMC card ready for use by the filing system. Unlike formatters for physical disks, the sectors are already prepared so most of the work is done. All that is required it so mark the directory entries as unused.