RomWBW Applications

Wayne Warthen

Thursday 3 December 2020

Contents

Summary **3**

ASSIGN **5**

SYSCOPY **9**

MODE **12**

FDU **14**

FORMAT **15**

XM **16**

FLASH **19**

FDISK80 **22**

TALK **24**

RTC **25**

TIMER **27**

INTTEST **29**

FAT **31**

TUNE **35**

Summary

RomWBW includes a small suite of custom applications to maximize the features available. In general, these applications are operating system agnostic – they run under any of the included operating systems. However, they all require RomWBW – they are not generic CP/M applications.

Most of the applications are custom written for RomWBW. However, some are standard CP/M applications that have been adapted to run under RomWBW (e.g., XModem). The applications are generally matched to the version of RomWBW they are distributed with. So, if you upgrade the version of RomWBW in your system ROM, you will want to copy the corresponding applications to any storage devices you are using.

Most of the applications are included on the RomWBW ROM disk, so they are easy to access.

The applications are also included with all of the operating system disk images provided with RomWBW. So, a simple way to ensure you have matching applications is to write the disk images onto your disk media when upgrading your ROM. Of course, this will destroy any existing data on your disk media, so don't do this if you are saving any data on the media.

Most of the applications are included as source code in the RomWBW distribution and are built in the normal build process. The source code is found in the Source\Apps directory of the distribution. The binary executable applications are found in the Binary\Apps directory.

The following table clarifies where each of the applications can be found:

Application	ROM Disk	Boot Disks	Apps Dir
ASSIGN	Yes	Yes	Yes
SYSCOPY	Yes	Yes	Yes
MODE	Yes	Yes	Yes
FDU	Yes	Yes	Yes

Application	ROM Disk	Boot Disks	Apps Dir
FORMAT	Yes	Yes	Yes
XM	Yes	Yes	Yes
FLASH	Yes	Yes	Yes
FDISK80	Yes	Yes	Yes
TALK	Yes	Yes	Yes
RTC	Yes	Yes	Yes
TIMER	Yes	Yes	Yes
INTTEST	Yes	Yes	Yes
FAT	No	Yes	Yes
TUNE	No	Yes	Yes

ASSIGN

RomWBW includes a flexible mechanism for associating the operating system drive letters (A: - P:) to the physical devices in the system. Drive letter assignments can be changed on a running operating system without rebooting. The ASSIGN command facilitates this by allowing you to display, assign, reassign, or remove the drive letter assignments.

Syntax

`ASSIGN /?`

`ASSIGN /L`

`ASSIGN [`*`<drv>`*`],...`

`ASSIGN`*`<drv>`*`=[`*`<device>`*`:[`*`<slice>`*`]],...`

`ASSIGN`*`<tgtdrv>`*`=`*`<srcdrv>`*`,...`

Usage

`ASSIGN /?` will display brief command usage and version information.

`ASSIGN /L` will display a list of all the devices available to be used in drive assignments in the running system. The devices listed may or may not contain media. Although some device types support the use of slices, the list does not indicate this.

`ASSIGN` with no parameters will list all of the current drive assignments.

`ASSIGN`*`<drv>`* will display the assignment for the specific drive For example, `ASSIGN C:` will display the assignment for drive C:.

`ASSIGN`*`<drv>`*`=`*`<device>`*`[:`*`<slice>`*`]` will assign (or reassign) a drive letter to a new device and (optionally) slice. If no slice is specified, then slice 0 is assumed. For example, `ASSIGN C:=IDE0` will assign drive letter C: to device IDE0, slice 0. `ASSIGN D:=IDE0:3` will assign drive letter D: to device IDE0 slice 3.

`ASSIGN<drv>=` can be used to remove the assignment from a drive letter. So, `ASSIGN E:=` will remove the association of drive letter E: from any previous device.

`ASSIGN<tgtdrv>=<srcdrv>` is used to swap the assignments of two drive letters. For example, `ASSIGN C:=D:` will swap the device assignments of C: and D:.

The `ASSIGN` command supports "stacking" of instructions. For example, `ASSIGN C:=IDE0:0,D:=IDE0:1,E:=` will assign C: and D: to the first two slices of IDE 0 and will unassign E:.

When the command runs it will echo the resultant assignments to the console to confirm it's actions. It will also display the remaining space available in disk buffers.

Notes

If the `ASSIGN` command encounters any rule violations or errors, it will abort with an error and **none** of the drive assignments will be implemented. In other words, the command is atomic and will either completely succeed or completely fail.

All assigned drives utilize disk buffer space from a limited pool. The ASSIGN command will display the amount of buffer space remaining after an assign command is executed. Buffer space is freed if a drive is unassigned. If the total assignments exceed the available disk buffer space available, the command will abort with an error message.

The `ASSIGN` command does not check to see if the device and slice being assigned actually contains readable media. If the assigned device has no media, you will receive an I/O error when you attempt to use the drive letter.

The `ASSIGN` command will not allow you to specify a slice (other than zero) for devices that do not support slices (such as floppy drives or RAM/ROM disks).

The ASSIGN command does not check that the media is large enough to support the slice you specify. In other words, you could potentially assign a drive letter to a slice that is beyond the end of the media in a device. In this case, subsequent attempts to use that drive letter will result in an I/O error.

Additionally, the ASSIGN command does not check to see if the slice specified refers to an area on your media that is occupied by other data (such as a FAT filesystem).

You will not be allowed to assign multiple drive letters to a single device and slice. In other words, only one drive letter may refer to a single filesystem at a time.

Drive letter A: must always be assigned to a device and slice. The ASSIGN command will enforce this.

The changes made by this command are not permanent. The assignments will persist through a warm start, but when you reboot your system, all drive letters will return to their default assignments. A SUBMIT batch file can be used to setup desired drive assignments automatically at boot.

Floppy disk drives and RAM/ROM drives do not have slices. A slice should only be specified for hard disk devices (SD, IDE, PPIDE).

Only one drive letter may be assigned to a specific device/unit/slice at a time. Attempts to assign a duplicate drive letter will fail and display an error. If you wish to assign a different drive letter to a device/unit/slice, unassign the the existing drive letter first.

Be aware that this command will allow you to reassign or remove the assignment of your system drive letter. This can cause your operating system to fail and force you to reboot.

This command is particularly sensitive to being matched to the appropriate version of the RomWBW ROM you are using. Be very careful to keep all copies of ASSIGN.COM up to date with your ROM.

Etymology

The `ASSIGN` command is an original product and the source code is provided in the RomWBW distribution.

SYSCOPY

To make disk media bootable, you must write a system boot image onto the system tracks of the of the media. The SYSCOPY allows you to read or write the system boot image of disk media.

Syntax

SYSCOPY*<dest>*=*<src>*

<dest> is the drive to receive the operating system image or alternatively a filename to save the operating system image

<src> is the drive containing an operating system image or alternatively a filename containing the system image to be placed on the destination

Usage

Both *<dest>* and *<src>* can refer to either a drive letter or a file. If a drive letter is specified, the system boot image will be read or written to the system tracks of the drive. If a filename is specified, the system boot image will be read or written to the specified filename.

SYSCOPY C:=ZSYS.SYS will read a system boot image from the file ZSYS.SYS and write it onto the system tracks of drive C:.

SYSCOPY A:OS.SYS=C: will capture the system boot image from the system tracks of drive C: and store it in the file A:OS.SYS.

SYSCOPY D:=C: will copy the system tracks from drive C: onto the system tracks of drive D:.

Notes

The RomWBW ROM disk contains files with the system boot image for Z-System and CP/M 2.2. These files are called CPM.SYS and ZSYS.SYS respectively. These files can be used as the source of a `SYSCOPY` command to make a disk bootable with the corresponding operating system.

CP/M 3 uses a two phase boot process. To make a CP/M 3 drive bootable, you need to put "CPMLDR.SYS" on the boot tracks of the disk and be sure that the drive also contains the "CPM.SYS" file. The "CPMLDR.SYS" file is not included on the ROM disk, but is found on the CP/M 3 disk image.

ZPM3 is similar to CP/M 3. You also put "CPMLDR.SYS" on the system tracks of the drive to make it bootable. The ZPM3 operating system is in the file called "CPM3.SYS" on the ZPM3 disk image. It may seem confusing that ZPM3 is in the file called CPM3.SYS, but it is normal for ZPM3.

For the purposes of booting an operating system, each disk slice is considered it's own operating system. Each slice can be made bootable with it's own system tracks.

`SYSCOPY` uses drive letters to specify where to read/write the system boot images. However, at startup, the boot loaded will require you to enter the actual disk device and slice to boot from. So, you need to be careful to pay attention to the device and slice that is assigned to a drive letter so you will know what to enter at the boot loader prompt. By way of explanation, the boot loader does not know about drive letters because the operating system is not loaded yet.

If you want to put a a boot system image on a device and slice that is not currently assigned to a drive letter, you will need to assign a drive letter first.

Not all disk formats include space for system tracks. Such disk formats cannot contains a system boot image and, therefore, cannot be made bootable. The best example of such disk formats are the ROM and RAM disks. To maximize usable file space on these drives, they do not have system tracks. Obviously,

ROM operating system is supported by choosing a ROM operating system at the boot loader prompt. Any attempt to write a system boot image to disk media with no system tracks will cause SYSCOPY to fail with an error message.

The system boot images are paired with the ROM version in your system. So, you must take care to update the system tracks of any bootable disk when you upgrade your ROM firmware.

The system boot images are **not** tied to specific hardware configurations. System boot images and operating systems provided with RomWBW will work with any supported RomWBW platform or hardware as long as they are the same version as the RomWBW firmware.

Etymology

The `SYSCOPY` command is an original product and the source code is provided in the RomWBW distribution.

MODE

The MODE command allows you to adjust the operating characteristics such as baud rate, data bits, stop bits, and parity bits of serial ports dynamically.

Syntax

`MODE /? MODE COM<n>: [<baud>[,<parity>[,<databits>[,<stopbits>]]]] [/P]`

`/?` displays command usage and version information

<n> is the character device unit number

<baud> is numerical baudrate

<parity> is (N)one, (O)dd, (E)ven, (M)ark, or (S)pace

<databits> is number of data bits, typically 7 or 8

<stopbits> is number of stop bits, typically 1 or 2

`/P` prompts user prior to setting new configuration

Usage

`MODE /?` will display basic command usage and version information.

`MODE` with no parameters will list all devices and their current configuration.

`MODE` <*n*> will display the current configuration of the specified character device unit.

`MODE COM<n>: [<baud>[,<parity>[,<databits>[, <stopbits>]]]] [/P]` requests that the specified configuration be set on the character device unit. You can use commas with no values to leave some values unchanged. As an example, `MODE COM0: 9600,,,2` will setup character device unit 0 for 9600 baud and 2 stop bits while leaving data bits and stop bits as is.

Appending `/P` in a command specifying a new configuration will cause the terminal output to pause and wait for the user to press a key. This allows the user to change the local terminal setup before continuing.

Notes

Specified baud rate and line characteristics must be supported by the serial unit. Any parameters not specified will remain unchanged.

Changes are not persisted and will revert to system defaults at next system boot.

Not all character devices support all `MODE` options. Some devices (notably ASCI devices) have limited baud rate divisors. An attempt to set a baud rate that the device cannot support will fail with an error message.

Etymology

The `MODE` command is an original product and the source code is provided in the RomWBW distribution.

FDU

The FDU application is a Floppy Disk Utility that provides functions to format and test floppy disk media.

Syntax

```
FDU
```

Usage

This application has an interactive user interface. At startup, you will be prompted to select the floppy interface hardware in your system. Following this, you will see the main menu of the program with many functions to manage floppy disk drives.

The primary documentation for this application is in a file called “FDU.txt” in the Doc directory of the RomWBW distribution. Please consult this file for usage information.

Notes

This application interfaces directly to the floppy hardware in your system. It does not use the RomWBW HBIOS. This means that even if your system is not configured for floppy drives, you can still use `FDU` to test your floppy drives and format floppy media. This also means it is critical that you choose the correct hardware interface from the initial selection when starting the application.

Etymology

The `FDU` command is an original product and the source code is provided in the RomWBW distribution.

FORMAT

This application is just a placeholder for a future version that will make it simpler to format media including floppy disks.

Syntax

```
FORMAT
```

Notes

This application currently just displays a few lines of information briefly instructing a user how to format media. It performs no actual function beyond this display currently.

Etymology

The `FORMAT` command is an original product and the source code is provided in the RomWBW distribution.

XM

An adaptation of Ward Christensen's X-Modem protocol for transferring files between systems using a serial port.

Syntax

`XM S`*<filename>*

`XM SK`*<filename>*

`XM L`*<library> <filename>*

`XM LK`*<library> <filename>*

`XM R`*<filename>*

`S`: Send a file `L`: Send a file from a library `R`: Receive a file `K`: Use 1K blocksize for transfer

<filename> is the name of a file to send or receive

<library> is the name of a library (.lbr) to extract a file to send

Usage

To transfer a file from your host computer to your RomWBW computer, do the following:

1. Enter one of the `XM` receive commands specifying the name you want to give to the received file.
2. On your host computer select a file to send and initiate the XModem send operation.

To transfer a file from your RomWBW computer to your host computer, do the following:

1. Enter one of the `XM` send commands specifying the name of the file to be sent.

2. On your host computer, specify the name to assign to the received file and initiate and XModem receive operation.

Please refer to the documentation of your host computer's terminal emulation software for specific instructions on how to use XModem.

Notes

The XModem adaptation that comes with RomWBW will automatically use the primary character device unit (character device unit 0) for the file transfer.

`XM` attempts to determine the best way to drive the serial port based on your hardware configuration. When possible, it will bypass the HBIOS for faster operation. However, in many cases, it will use HBIOS so that flow control can be used.

`XM` is dependent on a reliable communications channel. You must ensure that the serial port can be serviced fast enough by either using a baud rate that is low enough or ensuring that hardware flow control is fully functional (end to end).

Etymology

The `XM` application provided in RomWBW is an adaptation of a pre-existing XModem application. Based on the source code comments, it was originally adapted from Ward Christensen's MODEM2 by Keith Petersen and is labeled version 12.5.

The original source of the application was found in the Walnut Creek CD-ROM and is called XMDM125.ARK dated 7/15/86.

The actual application is virtually untouched in the RomWBW adaptation. The majority of the work was in the modem driver which was enhanced to detect the hardware being used and dynamically choose the appropriate driver.

The source code is provided in the RomWBW distribution.

FLASH

Most of the hardware platforms that run RomWBW support the use of EEPROMs – Electronically Erasable Programmable ROMs. The FLASH application can be used to reprogram such ROMS in-situ (in-place), thus making it possible to upgrade ROMs without a programmer or even removing the ROM from your system.

This application is provided by Will Sowerbutts.

Syntax

FLASH READ<*filename*> [options]
FLASH VERIFY<*filename*> [options]
FLASH WRITE<*filename*> [options]

<*filename*> is the filename of the ROM image file

FLASH4 will auto-detect most parameters so additional options should not normally be required.

Options:

/V: Enable verbose output (one line per sector)
/P or /PARTIAL: Allow flashing a large ROM from a smaller image file
/ROM: Allow read-only use of unknown chip types
/Z180DMA: Force Z180 DMA engine
/UNABIOS: Force UNA BIOS bank switching
/ROMWBW: Force RomWBW (v2.6+) bank switching
/ROMWBWOLD: Force RomWBW (v2.5 and earlier) bank switching
/P112: Force P112 bank switching
/N8VEMSBC: Force N8VEM SBC (v1, v2), Zeta (v1) SBC bank switching

Usage

To program your EEPROM ROM chip, first transfer the file to your RomWBW system. Then use the command `FLASH WRITE *'*`. The application will auto-detect the type of EEPROM chip you have, program it, and verify it.

You can use the `FLASH READ` form of the command to read the ROM image from your system into a file. This is useful if you want to save a copy of your current ROM before reprogramming it.

Although `FLASH WRITE` automatically performs a verification, you can manually perform a verification function with the `FLASH VERIFY` form of the command.

The author's documentation for the application is found in the RomWBW distribution in the Doc\Contrib directory.

Notes

The application supports a significant number of EEPROM parts. It should automatically detect your part. If it does not recognize your chip, make sure that you do not have a write protect jumper set – this jumper will cause the ROM chip type to be unrecognized.

Reprogramming a ROM chip in-place is inherently dangerous. If anything goes wrong, you will be left with a non-functional system and no ability to run the `FLASH` application again. Use this application with caution and be prepared to use a hardware ROM programmer to restore your system if needed.

Etymology

This application was written and provided by Will Sowerbutts. He provides it in binary format and is included in the RomWBW distribution as a binary file.

The source code for this application can be found at the FLASH4 GitHub repository.

FDISK80

RomWBW supports disk media with MS-DOS FAT filesystems (see FAT application). If you wish to put a FAT filesystem on your media, the FDISK80 application can be used to partition your media which is required in order to add a FAT filesystem.

This application is provided by John Coffman.

Usage

`FDISK80` is an interactive application. At startup it will ask you for the disk unit that you want to partition. When your RomWBW system boots, it will display a table with the disk unit numbers. Use the disk unit numbers from that table to enter the desired disk unit to partition.

`FDISK80` operates very much like other FDISK disk partitioning applications. Please refer to the file called "FDisk Manual.pdf" in the Doc directory of the RomWBW distribution for further instructions.

There is also more information on using FAT partitions with RomWBW in the "RomWBW Getting Started.pdf" document in the Doc directory of the distribution.

Notes

Partitioning of RomWBW media is **only** required if you want to add a FAT filesystem to your media. Do not partition your media if you are simply using it for RomWBW. To be clear, RomWBW slices do not require partitioning.

As described in "RomWBW Getting Started.pdf", you should be careful when adding a FAT partition to your media that the partition does not overlap with the area of the media being used for RomWBW slices. The "(R)eserve" function in `FDISK80` can help prevent this.

Etymology

The source for this application was provided directly by John Coffman. It is a C program and requires a build environment that includes the SDCC compiler. As such, it is not included in the RomWBW build process, only the binary executable is included.

Please contact John Coffman if you would like a copy of the source.

TALK

It is sometimes useful to direct your console input/output to a designated serial port. For example, if you were to connect a modem to your second serial port, you might want to connect directly to it and have everything you type sent to it and everything it sends be shown on your console. The `TALK` application does this.

Syntax

```
TALK [TTY:|CRT:|BAT:UC1:]
```

Usage

`TALK` operates at the operating system level (not HBIOS).

The parameter to `TALK` refers to logical CP/M serial devices. Upon execution all characters types at the console will be sent to the device specified and all characters received by the specified device will be echoes on the console.

Press Control+Z on the console to terminate the application.

Notes

This application is designed for CP/M 2.2 or Z-System. Use on later operating systems such as CP/M 3 is not supported.

Etymology

The `TALK` command is an original product and the source code is provided in the RomWBW distribution.

RTC

Many RomWBW systems provide real time clock hardware. The RTC application is a simple, interactive program allowing you to display and set the time and registers of the RTC.

Syntax

```
RTC
```

Usage

After startup, the application provides the following options:

Option	Function
`E)xit`	will terminate the application.
`T)ime`	will display the time as read from the RTC hardware.
`st(A)rt`	will restart the clock running if it is stopped.
`S)et`	will program the RTC clock with the date/time previously entered using the I)nit option.
`R)aw`	will read the minute/second of the RTC clock iteratively every time the space key is pressed. Press enter to end.
`L)oop`	will read the full date/time of the RTC clock iteratively every time the space key is pressed. Press enter to end.
`C)harge`	will enable the battery charging function of the RTC.
`N)ocharge`	will disable the battery charging functino of the RTC.
`D)elay`	allows you to test the built-in timing delay in the program. It is not unusual for it to be wrong.
`I)nit`	allows you to enter a date/time value for subsequent programming of the RTC using the S)et option.
`G)et`	allows you to read the value of a non-volatile register in the RTC.

Option	Function
P)ut	allows you to write the value of a non-volatile register in the RTC.
B)oot	will reboot your system.
H)elp	displays brief help.

Notes

When using Get and Put options, the register number to read/write is entered in hex. The non-volatile ram register numbers are 0x20-0x3F.

When entering values, you must enter exactly two hex characters. The backspace key is not supported. You do not use enter after entering the two hex characters. Yes, this should be improved.

The RTC application interacts directly with the RTC hardware bypassing HBIOS.

Etymology

The RTC application was originally written by Andrew Lync as part of the original ECB SBC board development. It has since been modified to support most of the hardware variations included with RomWBW.

TIMER

Most RomWBW systems have a 50Hz periodic system timer. A counter is incremented every time a timer tick occurs. The `TIMER` application displays the value of the counter.

Syntax

`TIMER` `TIMER /?` `TIMER /C`

Usage

Use `TIMER` to display the current value of the counter.

Use `TIMER /C` to display the value of the counter continuously.

The display of the counter will be something like this:

```
00045444 Ticks, 0000162A.10 Seconds
```

The first number is the total number of ticks since system startup. The second number is the total number of seconds since system startup.

Notes

The seconds value is displayed with a fractional value which is not a an actual fraction, but rather the number of ticks past the seconds rollover. All values are in hex.

The primary use of the `TIMER` application is to test the system timer functionality of your system.

In theory, you could capture the value before and after some process you want to time.

Etymology

The `TIMER` command is an original product and the source code is provided in the RomWBW distribution.

INTTEST

RomWBW includes an API allowing applications to “hook” interrupts. The `INTTEST` application allows you to test this functionality.

Syntax

```
INTTEST
```

Usage

`INTTEST` is an interactive application. At startup, it will display a list of the interrupt vector slots in your system along with the current vector address for each of them.

It then prompts you to enter the slot number (in hex) of a vector to hook. After entering this, the application will watch the hooked vector and countdown from 0xFF to 0x00 as interrupts are noted.

When the counter reaches 0x00, the interrupt is unhooked and the application terminates. The application can also be terminated by pressing .

Notes

If your system is running without interrupts active, the application will terminate immediately.

All slots have vectors even if the corresponding interrupt is not doing anything. In this case, the vector is pointing to the “bad interrupt” handler.

If you hook a vector that is not receiving any interrupts, the downcounter will not do anything.

Etymology

The `INTTEST` command is an original product and the source code is provided in the RomWBW distribution.

FAT

The operating systems included with RomWBW do not have any native ability to access MS-DOS FAT filesystems. The FAT application can be used overcome this. It will allow you to transfer files between CP/M and FAT filesystems (wildcards supported). It can also erase files, format, and list directories of FAT filesystems.

Syntax

`FAT DIR`*<path>*

`FAT COPY`*<src> <dst>*

`FAT REN`*<from> <to>*

`FAT DEL`*<path>[<file>|<dir>]*

`FAT MD`*<path>*

`FAT FORMAT`*<drv>*

<path> is a FAT path
<src>, *<dst>* are FAT or CP/M filenames
<from>, *<to>* are FAT filenames
<file> is a FAT filename
<dir> is a FAT directory name
<drv> is a RomWBW disk unit number

CP/M filespec: *<d>*`:FILENAME.EXT` (*<d>* is CP/M drive letter A-P)
FAT filespec: *<u>*`:/DIR/FILENAME.EXT` (*<u>* is RomWBW disk unit #)

Usage

The `FAT` application determines whether you are referring to a CP/M filesystem or a FAT filesystem based on the way you specify the file or path. If the file or path is prefixed with a number (n:), then it is assumed this is a FAT filesystem reference and is referring to the FAT filesystem on RomWBW disk

unit ‘n’. Otherwise, the file specification is assumed to be a normal CP/M file specification.

If you wanted to list the directory of the FAT filesystem on RomWBW disk unit 2, you would use `FAT DIR 2:`. If you only wanted to see the “.TXT” files, you would use `FAT DIR 2:*.TXT`.

If you wanted to copy all of the files on CP/M drive B: to the FAT filesystem on RomWBW disk unit 4, you would use the command `FAT COPY B:*.* 4:` If you wanted to copy the files to the “FOO” directory, then you would use `FAT COPY B:*.* 4:\FOO`. To copy files in the opposite direction, you just reverse the parameters.

To rename the file “XXX.DAT” to “YYY.DAT” on a FAT filesystem, you could use a command like “FAT REN 2:XXX.DAT 2:YYY.DAT”.

To delete a file “XXX.DAT” on a FAT filesystem in directory “FOO”, you would use a command like `FAT DEL 2:\FOO\XXX.DAT`.

To make a directory called “FOO2” on a FAT filesystem, you would use a command line `FAT MD 2:\FOO2`.

To format the filesystem on a FAT partition, you would use a command like `FAT FORMAT 2:`. Use this with caution because it will destroy all data on any pre-existing FAT filesystem on disk unit 2.

Notes

Partitioned or non-partitioned media is handled automatically. A floppy drive is a good example of a non-partitioned FAT filesystem and will be recognized. Larger media will typically have a partition table which will be recognized by the application to find the FAT filesystem.

Although RomWBW-style CP/M media does not know anything about partition tables, it is entirely possible to have media that has both CP/M and FAT file systems on it. This is accomplished by creating a FAT filesystem on the

media that starts on a track beyond the last track used by CP/M. Each CP/M slice on a media will occupy 8,320K (16,640 sectors). So, make sure to start your FAT partition beyond (< slice count> * 8,320K) or (<slice count * 16,640 sectors).

The application infers whether you are attempting to reference a FAT or CP/M filesystem via the drive specifier (char before ':'). A numeric drive character specifies the HBIOS disk unit number for FAT access. An alpha (A-P) character indicates a CP/M file system access targeting the specified drive letter. If there is no drive character specified, the current CP/M filesystem and current CP/M drive is assumed. For example:

`2:README.TXT` refers to FAT file "README.TXT" on disk unit #2
`C:README.TXT` refers to CP/M file "README.TXT" on CP/M drive C
`README.TXT` refers to CP/M file "README.TXT" on the current CP/M

drive

Files with SYS, HIDDEN, or R/O only attributes are not given any special treatment. Such files are found and processed like any other file. However, any attempt to write to a read-only file will fail and the application will abort.

It is not currently possible to reference CP/M user areas other than the current user. To copy files to alternate user areas, you must switch to the desired user number first or use an additional step to copy the file to the desired user area.

Accessing FAT filesystems on a floppy requires the use of RomWBW HBIOS v2.9.1-pre.13 or greater.

Files written are not verified.

Wildcard matching in FAT filesystems is a bit unusual as implemented by FatFs. See FatFs documentation.

Etymology

The `FAT` application is an original RomWBW work, but utilizes the FsFat library for all of the FAT filesystem work. This application is written in C and requires SDCC to compile. As such it is not part of the RomWBW build process. However, the full project and source code is found in the FAT GitHub Repository.

TUNE

If your RomWBW system has a sound card based on either an AY-3-8190 or YM2149F sound chip, you can use the `TUNE` application to play PT or MYM sound files.

Syntax

`TUNE` *<filename>*

<filename> is the name of a sound file ending in .PT2, .PT3, or .MYM

Usage

The TUNE application supports PT and YM sound file formats. It determines the format of the file from the extension of the file, so your tune filenames should end in .PT2, .PT3, or .MYM.

To play a sound file, just use the command and specify the file to play after the command. So, for example, `TUNE ATTACK.PT2` will immediately begin playing the PT sound file “ATTACK.PT2”.

Notes

The `TUNE` application automatically probes for compatible hardware at well known port addresses at startup. It will auto-configure itself for the hardware found. If no hardware is detected, it will abort with an error message.

On Z180 systems, I/O wait states are added when writing to the sound chip to avoid exceeding it’s speed limitations. On Z80 systems, you will need to ensure that the CPU clock speed of your system does not exceed the timing limitations of your sound chip.

The application probes for an active system timer and uses it to accurately pace the sound file output. If no system timer is available, a delay loop is

calculated instead. The delay loop will not be as accurate as the system timer.

There are two modes of operations. A direct hardware interface for the AY-3-8910 or YM2149 chips, or a compatibility layer thru HBIOS supporting the SN76489 chip.

By default the application will attempt to interface directly to the sound chip. The optional argument `--hbios` supplied after the filename, will enable the application to use the HBIOS sound driver.

The HBIOS mode also support other switch as desribed below.

Switch	Description
`--hbios`	Utilise HBIOS' sound driver
`+t1`	Play tune an octave higher
`+t2`	Play tune two octaves higher
`-t1`	Play tune an octave lower
`-t2`	Play tune two octaves lower

All RomWBW operating system boot disks include a selection of sound files in user area 3.

Etymology

The `TUNE` application was custom written for RomWBW. All of the hardware interface code is specific to RomWBW. The sound file decoding software was adapted and embedded from pre-existing sources. The YM player code is from MYMPLAY 0.4 by Lieves!Tuore and the PT player code is (c)2004-2007 S.V.Bulba vorobey@mail.khstu.ru.

The source code is provided in the RomWBW distribution.