RomWBW

Applications Guide

Version 3.6

Updated 13 Oct 2025

RetroBrew Computers Group
www.retrobrewcomputers.org

MartinR & Phillip Summers

Contents

1 Summary **1**

2 Boot Menu **3**

3 ROM Applications **6**

3.1 Monitor . . . 6
3.1.1 Monitor Commands . . . 7
3.2 CP/M 2.2 . . . 11
3.3 Z-System . . . 11
3.4 BASIC . . . 11
3.4.1 RomWBW specific features . . . 12
3.4.2 RomWBW unsupported features . . . 12
3.5 TastyBASIC . . . 12
3.5.1 Features / Limitations . . . 12
3.5.2 Direct Commands . . . 12
3.5.3 Statements . . . 12
3.5.4 Functions . . . 12
3.5.5 Operators . . . 13
3.6 FORTH . . . 13
3.6.1 Important things to know . . . 13
3.6.2 Structure of Forth source files . . . 13
3.6.3 RomWBW Additions . . . 14
3.7 Play a Game (2048) . . . 14
3.8 Xmodem Flash Updater . . . 16
3.8.1 Usage . . . 17
3.8.2 Console Options . . . 17
3.8.3 Programming options . . . 18
3.8.4 Exit options . . . 19
3.8.5 CRC Utility options . . . 19
3.8.6 Tera Term macro configuration . . . 19
3.8.7 Serial speed guidelines . . . 20

3.8.8 Notes . . . 20
3.9 User Application . . . 21

4 CP/M Applications - ROM-Based & Disk-Based 22
4.1 ASSIGN . . . 25
4.2 BBCBASIC (BBC BASIC) . . . 30
4.3 CLRDIR (Clear Directory) . . . 31
4.4 CPUSPD (CPU Speed) . . . 32
4.5 COPYSL (Copy Slice) . . . 34
4.6 FAT (FAT Utility) . . . 36
4.7 FDISK80 . . . 39
4.8 FDU (Floppy Disk Utility) . . . 41
4.9 FLASH (Flash EEPROM) . . . 42
4.10 FORMAT . . . 44
4.11 HTALK (HBIOS Talk) . . . 45
4.12 MODE . . . 46
4.13 REBOOT . . . 48
4.14 RTC (Real Time Clock) . . . 49
4.15 SLABEL (Slice Label) . . . 51
4.16 SURVEY . . . 52
4.17 SYSCONF (System Configuration) . . . 53
4.18 SYSCOPY (System Copy) . . . 57
4.19 TALK . . . 59
4.20 TBASIC (Tasty BASIC) . . . 60
4.21 TIMER . . . 61
4.22 TUNE . . . 63
4.23 VGMPLAY (Video Game Music Play) . . . 66
4.24 WDATE (WBW DATE) . . . 68
4.25 XM (X-Modem) . . . 70
4.26 ZMD (Z-Modem) . . . 72
4.27 ZMP (Z-Modem Program) . . . 74

Chapter 1

Summary

RomWBW is supplied with a suite of software applications that enhance the use of the system. Some of these applications have been written entirely from scratch for RomWBW. Others are pre-existing software that has been customized for the RomWBW environment. This document serves as a reference for these RomWBW-specific applications.

The primary usage documentation for RomWBW is the RomWBW User Guide. It is assumed that the reader is generally familiar with this document.

RomWBW also includes many generic software applications that have not been modified for RomWBW (e.g., MSBASIC). These generic applications are not documented here. Please refer to the application specific documentation for these generic applications. The documentation for some of these generic applications is included in the Doc folder of the RomWBW distribution.

The applications described in this document fall into two general categories.

1. **ROM Applications** are software applications that are loaded from the the ROM memory of your RomWBW system.
2. **CP/M Applications** are software applications that are loaded from disk using a previously loaded CP/M (or CP/M like) operating system using its command line.

Note that some applications are available in both forms. For example, Microsoft BASIC is available as a ROM application and as an application that runs under CP/M. Only the ROM variant is documented here because the CP/M variant is not RomWBW-specific.

You will see that two of the RomWBW operating systems are included here as ROM Applications. Although operating systems are normally loaded from disk, RomWBW does include a way to launch CP/M 2.2 and Z-System directly from ROM.

Most RomWBW systems include a ROM disk. A running operating system can load applications from the ROM disk just like a floppy or hard disk. Applications loaded from the ROM disk by CP/M are considered to be CP/M applications, **not** ROM applications.

Chapter 2

Boot Menu

The system start-up process is described in some detail in the RomWBW User Guide, and for the sake of completeness there is some overlap here.

When a RomWBW system is started the user is presented with a sign-on message at the default console detailing the RomWBW version and build date. The system follows this with the list of hardware that it has discovered, a list of devices and the system units assigned to them.

If autoboot is configured then the message (below) will count down and once 0 is reached the system will automatically boot with the configured options

```
AutoBoot in 3 Seconds (<esc> aborts, <enter> now)...
```

Pressing `esc` - will bypass the auto boot process going immediately to the `Boot` prompt, or pressing `enter` - will proceed with autoboot immediately. Auto boot is configured using the `W` boot menu option.

If autoboot is bypassed (or not configured) the user is asked to select a boot device with the prompt:

```
Boot [H=Help]:
```

At this point, the user may specify a unit, optionally with a slice, to boot from. Note that it is not possible to boot from from the serial (ASCI) or memory disk (MD) devices.

Alternatively the user may select one of the built-in Boot Loader commands. A menu of which may be displayed by pressing the H or ? keys (for Help). Furthermore, a ROM application may also be started from this prompt.

This start-up process is described in some detailed in the RomWBW User Guide, and there is some overlap here.

Help

After pressing H or ? at the boot prompt the user will be presented with the following list of available commands:

```
Boot [H=Help]: H

  L           - List ROM Applications
  D           - Device Inventory
  S           - Slice Inventory
  R           - Reboot System
  W           - RomWBW Configure
  I <u> [<c>] - Set Console Interface/Baud code
  V [<n>]     - View/Set HBIOS Diagnostic Verbosity
  N           - Network Boot
  <u>[.<s>]   - Boot Disk Unit/Slice
```

The function performed by each command is described below:

L: Lists the applications and operating systems that are built into the RomWBW ROM - e.g., low-level monitor utility, CP/M, or BASIC.

D: Displays the list of system devices that was first displayed when the system was started.

S: Displays the list of disk Slices that contain a label indicating that they may be bootable. See SLABEL (Slice Label) for more details about labels.

R: Will restart the system. Note that this does not reset hardware devices in the same way that power-on or pressing the reset button would.

W: Runs the SYSCONF (System Configuration) utility allowing RomWBW configuration stored in Non Volatile memory to be changed.

I: Allows the user to select the interface connected to the console, and optionally the Baud rate. This could be used to allow the system to be operated from a second console.

V: Enables the display of invalid RomWBW HBIOS API calls. This option is very unlikely to be used by a user and is used for development purposes.

N: Boot into CP/M via an RCBus Wiznet MT011 network module if configured. Section 10 of the RomWBW User Guide provides complete instructions for setting up a CP/NET based network under RomWBW including network booting.

And, finally, the system may be booted by specifying the unit number, and optional slice, separated by a period('.'), of where the disk operating system software is located - eg 2, 4.1, 5.3

Alternatively, a RomWBW ROM application may be started by pressing the appropriate key from the applications menu, shown in the following section.

List ROM Applications

If the user presses the L key at the Boot Loader prompt then the system will display the list of ROM applications that are built into RomWBW. If a command letter is known, then it may be entered directly at the prompt rather than first displaying the menu.

The ROM applications available from the boot prompt are:

```
Boot [H=Help]: L

ROM Applications:

  M: Monitor
  Z: Z-System
  C: CP/M 2.2
  F: Forth
  B: BASIC
  T: Tasty BASIC
  P: Play a Game
  X: XModem Flash Updater
  U: User App
```

Each of these will now be described in greater detail.

Chapter 3

ROM Applications

3.1 Monitor

The Monitor program is a low-level utility that can be used for testing and programming. It allows programs to be entered, memory to be examined and modified, and input/output devices to be read or written to.

It's key advantage is that is available at boot up.

Its key disadvantages are that code cannot be entered in assembly language and there is no ability to save to persistent storage (disks).

The available memory area for programming is `0100h-EDFFh`. The following areas are reserved:

Memory Area	Function
`0000-00FFh`	Jump and restart (RST) vectors
`EE00-FDFFh`	Monitor
`FE00-FFFFh`	HBIOS proxy

The monitor uses a prompt in the format of `xx>` where xx is the RomWBW bank id number. For example, the prompt may look like this and means that Bank Id 0x8E is currently mapped into the low 32K of processor memory.

```
8E>
```

Please refer to Section 4 of the $doc_sys# for a description of the RomWBW Bank Id and how it relates to the physical bank of memory being mapped to the lower 32K of the processor.

The method of assigning banks for specific RomWBW functions is also described.

Commands can be entered at the command prompt. Automatic case conversion takes place on command entry and all numeric arguments are expected to be in hex format.

The Monitor allows access to all memory locations but ROM and Flash memory cannot be written to. At startup, the Monitor will select the default "User" bank. The `S` command is provided to allow selecting alternate banks.

There now follows a more detailed guide to using the RomWBW Monitor program:

3.1.1 Monitor Commands

? - Will display a summary of the available commands.

```
Monitor Commands (all values in hex):
B                    - Boot system
D xxxx [yyyy]        - Dump memory from xxxx to yyyy
F xxxx yyyy zz       - Fill memory from xxxx to yyyy with zz
H                    - Halt system
I xxxx               - Input from port xxxx
K                    - Keyboard echo
L                    - Load Intel hex data
M xxxx yyyy zzzz     - Move memory block xxxx-yyyy to zzzz
O xxxx yy            - Output value yy to port xxxx
P xxxx               - Program RAM at address xxxx
R xxxx [[yy] [zzzz]] - Run code at address xxxx
                       Pass yy and zzzz to register A and BC
S xx                 - Set bank to xx
U                    - Set bank to previous bank
T xxxx               - X-modem transfer to memory location xxxx
X                    - Exit monitor
```

Cold Boot

B - Performs a cold boot of the RomWBW system. A complete re-initialization of the system is performed and the system returns to the Boot Loader prompt.

Dump Memory

`D xxxx [yyyy]` - Dump memory from hex location xxxx to yyyy on the screen as lines of 16 hexadecimal bytes with their ASCII equivalents (if within a set range, else a '.' is printed). If the

end address is omitted then 256 bytes are displayed.

A good tool to see where code is located, check for version id, obtain details for chip configurations and execution paths.

Example: `D 100 1FF`

```
0100: 10 0B 01 5A 33 45 4E 56 01 00 00 2A 06 00 F9 11  ...Z3ENV...*..ù.
0110: DE 38 37 ED 52 4D 44 0B 6B 62 13 36 00 ED B0 21  Þ87íRMD.kb.6.í°!
0120: 7D 32 E5 21 80 00 4E 23 06 00 09 36 00 21 81 00  }2å!..N#...6.!..
0130: E5 CD 6C 1F C1 C1 E5 2A C9 8C E5 CD 45 05 E5 CD  åÍl.ÁÁå*É.åÍE.åÍ
0140: 59 1F C3 00 00 C3 AE 01 C3 51 04 C3 4C 02 C3 57  Y.Ã..î.ÃQ.ÃL.ÃW
0150: 02 C3 64 02 C3 75 02 C3 88 02 C3 B2 03 C3 0D 04  .Ãd.Ãu.Ã..ò.Ã..
0160: C3 19 04 C3 22 04 C3 2A 04 C3 35 04 C3 40 04 C3  Ã..Ã".Ã*.Ã5.Ã@.Ã
0170: 48 04 C3 50 04 C3 50 04 C3 50 04 C3 8F 02 C3 93  H.ÃP.ÃP.ÃP.Ã..Ã.
0180: 02 C3 94 02 C3 95 02 C3 85 04 C3 C7 04 C3 D1 01  .Ã..Ã..Ã..ÃÇ.ÃÑ.
0190: C3 48 02 C3 E7 04 C3 56 03 C3 D0 01 C3 D0 01 C3  ÃH.Ãç.ÃV.ÃÐ.ÃÐ.Ã
01A0: D0 01 C3 D0 01 C3 D0 01 C3 D0 01 01 02 01 CD 6B  Ð.ÃÐ.ÃÐ.ÃÐ....Ík
01B0: 04 54 68 69 73 20 66 75 6E 63 74 69 6F 6E 20 6E  .This function n
01C0: 6F 74 20 73 75 70 70 6F 72 74 65 64 2E 0D 0A 00  ot supported....
01D0: C9 3E FF 32 3C 00 3A 5D 00 FE 20 28 14 D6 30 32  É>ÿ2<.:].þ (.Ö02
01E0: AB 01 32 AD 01 3A 5E 00 FE 20 28 05 D6 30 32 AC  «.2.:^.þ (.Ö02¬
01F0: 01 C5 01 F0 F8 CF E5 26 00 0E 0A CD 39 02 7D 3C  .Å.ðøÏå&...Í9.}<
```

Fill Memory

`F xxxx yyyy zz` - Fill memory from hex xxxx to yyyy with a single value of zz over the full range. The Dump command can be used to confirm that the fill completed as expected. A good way to zero out memory areas before writing machine data for debug purposes.

Halt System

`H` - Halt system. A Z80 HALT instruction is executed. The system remains in the halt state until the system is physically rebooted. Interrupts will not restart the system. On systems that support a HALT status LED, the LED will be illuminated.

Input from Port

`I xxxx` - Input data from port xxxx and display to the screen. This command is used to read values from hardware I/O ports and display the contents in hexadecimal.

Keyboard Echo

K - Echo any key-presses from the terminal. Press 'ESC' key to quit. This facility provides that any key stroke sent to the computer will be echoed back to the terminal. File down loads will be echoed as well while this facility is 'on'.

Load Hex

L - Load a Intel Hex data via the terminal program. The load address is defined in the hex file of the assembled code.

The terminal emulator program should be configured to give a delay at the end of each line to allow the monitor enough time to parse the line and move the data to memory.

Keep in mind that this will be transient unless the system supports battery backed memory. Saving to memory drive is not supported.

Move Memory

M xxxx yyyy zzzz - Move hex memory block xxxx to yyyy to memory starting at hex location zzzz. Care should be taken to insure that there is enough memory at the destination so that code does not get over-written or memory wrapped around.

Output to Port

O xxxx yy - Output data byte xx to port xxxx. This command is used to send hexadecimal values to hardware I/O ports to verify their operation and is the companion to the I operation. Use clip leaded LEDs to confirm the data written.

Program Memory

P xxxx - Program memory location xxxx. This routine will allow you to program a hexadecimal value 'into memory starting at location xxxx. Press 'Enter' on a blank line to return to the Monitor prompt.

The limitation around programming memory is that it must be entered in hexadecimal. An alternative is to use the L command to load a program that has been assembled to a hex file on the remote computer.

An excellent online resource for looking up opcodes for entry can be found here: https://clrhome.org/table.

Run Program

R xxxx [[yy] [zzzz]] - Run program at location xxxx. If optional arguments yy and zzzz are entered they are loaded into the A and BC register respectively. The return address of the Monitor is saved on the stack so the program can return to the monitor. On return to the monitor, the contents of the A, HL, DE and BC registers are displayed.

Set Bank

S xx - Set the physical memory bank to the RomWBW Bank Id indicated by xx. Memory addresses 0x0000-0x7FFF (i.e. bottom 32k) are affected. Because the interrupt vectors are stored in the bottom page of this range, this function is disabled when interrupt mode 1 is being used (IM1). Interrupt mode 2 is not affected as the associated jump vectors are stored in high memory.

Changing the bank also impacts the restart vectors (RST), so executing code that calls the HBIOS using the `RST 08` assembly code will not work.

The monitor stack resides in high memory and is not affected but any code that changes the stack to low memory will be affected.

The U command may be used to undo the change and return the selected memory bank back to the previously selected one.

Section 4 of the RomWBW System Guide provides detail on how Bank Ids map to the physical memory of the system and also how specific banks are utilized by RomWBW.

Undo Bank

U - Change the bank in memory back to the previously selected bank. This command should be used in conjunction with the S command.

X-Modem Transfer

T xxxx - Receive an X-modem file transfer and load it into memory starting at location xxxx.

128 byte blocks and checksum mode is the only supported protocol.

Exit Monitor

X - Exit the monitor program back to the main boot menu.

3.2 CP/M 2.2

This option will boot the CP/M 2.2 disk operating system from an image contained within the ROM. Please refer to the CPM User Manual in the Doc/CPM folder of the distribution for CP/M usage. There are also many online resources.

During the build process the system will create a ROM disk containing a number of curated CP/M applications, and also a RAM drive. The capacity of each will depend upon the size of the ROM and RAM available to the system. A more complete set of utilities are provided within the disk image files provided as part of RomWBW.

A number of the applications provided are generic to CP/M, while others rely on particular hardware or aspects of RomWBW itself.

Those that are written specific to RomWBW include: ASSIGN, CPUSPD, FDU, FORMAT, FLASH, FDISK80, MODE, REBOOT, RTC, SYSCOPY, TALK, TIMER, XM, and COPYSL.

The CP/M utilities supplied with RomWBW warrant more detailed descriptions, and so are described in some detail in their own section of this user guide. In summary they provide the initial capability to manage and update your RomWBW system, to create other bootable media (hardware dependent) and to write/debug code using assembler and BASIC.

3.3 Z-System

Z-System is a complete alternative, but entirely compatible, disk operating system to CP/M.

Z-System is comprised of ZSDOS 1.1 which is a replacement for CP/M's Basic Disk Operating System (BDOS), and ZCPR which is a replacement for the Console Command Processor (CCP). Either or both may be used, although using both together will allow ZCPR to make use of specific ZSDOS features.

Documentation for Z-System may be found in the Doc/CPM folder of the RomWBW distribution and the reader is referred to those.

3.4 BASIC

For those who are not familiar with BASIC, it stands for Beginners All Purpose Symbolic Instruction Code.

RomWBW contains two versions of ROM BASIC, a full implementation and a "tiny" BASIC.

The full implementation is a version of Microsoft BASIC from the NASCOM Computer.

A comprehensive instruction manual is available in the Doc/Contrib directory.

3.4.1 RomWBW specific features

- Sound
- Graphics
- Terminal Support

3.4.2 RomWBW unsupported features

- Cassette loading
- Cassette saving

3.5 TastyBASIC

TastyBASIC offers a minimal implementation of BASIC that is only 2304 bytes in size. It originates from Li-Chen Wang's Palo Alto Tiny BASIC from around 1976. It's small size is suited the tiny memory capacities of the time. This implementation is by Dimitri Theulings and his original source can be found at https://github.com/dimitrit/tastybasic.

3.5.1 Features / Limitations

- Integer arithmetic, numbers -32767 to 32767
- Singles letter variables A-Z
- 1-dimensional array support
- Strings are not supported

3.5.2 Direct Commands

- `LIST`,`RUN`, `NEW`, `CLEAR`, `BYE`

3.5.3 Statements

- `LET`, `IF`, `GOTO`, `GOSUB RETURN`, `REM`, `FOR TO NEXT STEP`, `INPUT`, `PRINT`, `POKE`, `END`

3.5.4 Functions

- `PEEK`, `RND`, `ABS`, `USR`, `SIZE`

3.5.5 Operators

- >=, #, >, =, <=, <
- Operator precedence is supported.

Type ***BYE*** to return to the boot menu.

3.6 FORTH

CamelForth is the version of Forth included as part of the boot ROM in RomWBW. It has been converted from the Z80 CP/M version published at https://www.camelforth.com/page.php?5. The author is Brad Rodriguez who is a prolific Forth enthusiast, whose work can be found here: https://www.bradrodriguez.com/papers.

For those are who are not familiar with Forth, I recommend the wikipedia article https://en.wikipedia.org/wiki/Forth_(programming_language) and the Forth Interest Group website https://www.forth.org/.

3.6.1 Important things to know

Forth is case sensitive.

To exit back to the boot loader type ***bye***

To get a list of available words type ***WORDS***

To reset Forth to its initial state type ***COLD***

Most of the code you find on the internet will not run unless modified or additional Forth words are added to the dictionary.

This implementation does not support loading or saving of programs. All programs need to be typed in. Additionally, screen editing and code blocks are not supported.

A CP/M version is not provided with RomWBW, this is only a ROM application. If you need to run it under CP/M you would need to download it from the camelforth web site, the link is above.

3.6.2 Structure of Forth source files

File	Description
camel80.azm	Code Primitives

File	Description
camel80d.azm	CPU Dependencies
camel80h.azm	High Level words
camel80r.azm	RomWBW additions
glosshi.txt	Glossary of high level words
glosslo.txt	Glossary of low level words
glossr.txt	Glossary of RomWBW additions

3.6.3 RomWBW Additions

Extensions and changes to this implementation compared to the original distribution are:

- The source code has been converted from Z80mr assembler to Hector Peraza's zsm.
- An additional file camel80r.azm has been added for including additional words to the dictionary at build time. However, as currently configured there is very little space allocated for addition words. Exceeding the allocated ROM space will generate an error message when building.
- James Bowman's double precision words have been added from his RC2014 version: https://github.com/jamesbowman/camelforth-z80.

Word	Syntax	Description
D+	d1 d2 – d1+d2	Add double numbers
2>R	d –	2 to R
2R>	d –	fetch 2 from R
M*/	d1 n2 u3 – d=(d1*n2)/u3	double precision mult. div
SVC	hl de bc n – hl de bc af	Execute a RomWBW function
P!	n p –	Write a byte to a I/O port
P@	p – n	Read a byte from and I/O port

3.7 Play a Game (2048)

2048 is a puzzle game that can be both mindless and challenging. It appears deceptively simple but failure can creep up on you suddenly.

It requires an ANSI/VT-100 compatible colour terminal to play.

2048 is like a sliding puzzle game except the puzzle tiles are numbers instead of pictures. Instead of moving a single tile all tiles are moved simultaneously in the same direction. Where

two tiles of the same number collide, they are reduced to one tile with the combined value. After every move a new tile is added with a starting value of 2.

The goal is to create a tile of 2048 before all tile locations are occupied. Reaching the highest points score, which is the sum of all the tiles is a secondary goal. The game will automatically end when there are no more possible moves.

Play consists of entering a direction to move. Directions can be entered using any of three different keyboard direction sets.

```
Direction | Keys
----------|----------
Up        | w ^E 8
Down      | s ^X 2
Left      | a ^S 4
Right     | d ^D 6
```

The puzzle board is a 4x4 grid. At start, the grid will be populated with two 2 tiles. An example game sequence is shown below with new tiles to the game shown in brackets.

```
Start             Move 1 - Up       Move 2 - Left     Move 3 - Left
+---+---+---+---+ +---+---+---+---+ +---+---+---+---+ +---+---+---+---+
|   |   |   |(2)| |   |   |   | 4 | | 4 |   |   |   | | 4 |   |   |   |
+---+---+---+---+ +---+---+---+---+ +---+---+---+---+ +---+---+---+---+
|   |   |   |   | |   |   |   |   | |   |   |   |(4)| | 4 |   |   |   |
+---+---+---+---+ +---+---+---+---+ +---+---+---+---+ +---+---+---+---+
|   |   |   |(2)| |   |   |   |   | |   |   |   |   | |   |   |   |   |
+---+---+---+---+ +---+---+---+---+ +---+---+---+---+ +---+---+---+---+
|   |   |   |   | |   |   |(2)|   | | 2 |   |   |   | | 2 |   |(2)|   |
+---+---+---+---+ +---+---+---+---+ +---+---+---+---+ +---+---+---+---+

Move 4 - Left     Move 5 - Up       Move 6 - Right    Move 7 - Up
+---+---+---+---+ +---+---+---+---+ +---+---+---+---+ +---+---+---+---+
| 4 |   |   |   | | 8 |   |   | 4 | |   |   | 8 | 4 | |   |   | 8 | 8 |
+---+---+---+---+ +---+---+---+---+ +---+---+---+---+ +---+---+---+---+
| 4 |   |   |(4)| | 4 |   |   |   | |   |   |   | 4 | |   |   |   | 2 |
+---+---+---+---+ +---+---+---+---+ +---+---+---+---+ +---+---+---+---+
|   |   |   |   | |   |   |   |   | |   |   |   |   | |   |   |   |   |
+---+---+---+---+ +---+---+---+---+ +---+---+---+---+ +---+---+---+---+
| 4 |   |   |   | |(2)|   |   |   | |(2)|   |   | 2 | |(2)|   |   |   |
+---+---+---+---+ +---+---+---+---+ +---+---+---+---+ +---+---+---+---+
```

This is how I lost this game:

```
+---+---+---+---+
| 4 | 2 | 16| 4 |
+---+---+---+---+
| 32| 64| 8 | 2 |
+---+---+---+---+
| 4 | 8 |128| 32|
+---+---+---+---+
|(2)| 16| 8 | 4 |
+---+---+---+---+
```

Press Q at any time to bring up the option to Quit or Restart the game.

3.8 Xmodem Flash Updater

The RomWBW Xmodem flash updater provides the capability to update RomWBW from the boot loader using an x-modem file transfer. It offers similar capabilities to Will Sowerbutts FLASH4 utility except that the flashing process occurs during the file transfer.

These are the key differences between the two methods are:

Xmodem Flash Updater	FLASH.COM (aka FLASH4)
Available from the boot loader	Well proven and tested
Xmodem transfer is integrated	Wider range of supported chips and hardware
Integrated checksum utilities	Wider range of supported platforms
Capability to copy a ROM image	Only reprograms sectors that have changed
More convenient one step process	Ability save and verify ROM images
No intermediate storage required	Progress display while flashing
.	Displays chip identification information
.	Faster file transfer

The major disadvantages of the Updater is that it is new and relatively untested. There is the risk that a failed transfer will result in a partially flashed and unbootable ROM. There are some limitations on serial transfer speeds.

The updater utility was initially intended to support the Retrobrew SBC-V2-005 platform using Atmel 39SF040 flash chips but has now been extended to be more generic in operation.

Supported flash chips are 39SF040, 29F040, AT49F040, AT29C040, M29F040 , MX29F040, A29010B, A29040B

The Atmel 39SF040 chip is recommended as it can erase and write 4Kb sectors. Other chips require the whole chip to be erased.

3.8.1 Usage

In most cases, completing a ROM update is a simple as:

1. Booting to the boot loader prompt
2. Selecting option X - Xmodem Flash Updater
3. Selecting option U - Update
4. Initiating an X-modem transfer of your ROM image on your console device
5. Selecting option R - Reboot

If your console device is not able to transfer a ROM image i.e. your console is a VDU then you will have to use the console options to identify which character-input/output device is to be used as the serial device for transfer.

When your console is the serial device used for the transfer, no progress information is displayed as this would disrupt the x-modem file transfer. If you use an alternate character-input/output devices as the serial device for the transfer then progress information will be displayed on the console device.

Due to different platform processor speeds, serials speeds and flow control capabilities the default console or serial device speed may need to be reduced for a successful transfer and flash to occur. The **Set Console Interface/Baud code** option at the Boot Loader can be used to change the speed if required. Additionally, the Updater has options to set to and revert from a recommended speed.

See the RomWBW Applications guide for additional information on performing upgrades.

3.8.2 Console Options

Option (C) - Set Console Device

Option (S) - Set Serial Device

By default the updater assumes that the current console is a serial device and that the ROM file to be flashed will also be transferred across this device, so the Console and Serial device are both the same.

Either device can be can be change to another character-input/output device but the updater will always expect to receive the x-modem transfer on the **Serial Device**

The advantage of transferring on a different device to the console is that progress information can be displayed during the transfer.

Option (>) - Set Recommended Baud Rate

Option (<) - Revert to Original Baud Rate

3.8.3 Programming options

Option (U) - Begin Update

The will begin the update process. The updater will expect to start receiving an x-modem file on the serial device unit.

X-modem sends the file in packets of 128 bytes. The updater will cache 32 packets which is 1 flash sector and then write that sector to the flash device.

If using separate console, bank and sector progress information will shown

```
BANK 00 s00 s01 s02 s03 s04 s05 s06 s06 s07
BANK 01 s00 s01 s02 s03 s04 s05 s06 s06 s07
BANK 02 s00 s01 s02 s03 s04 s05 s06 s06 s07 etc
```

The x-modem file transfer protocol does not provide any filename or size information for the transfer so the updater does not perform any checks on the file suitability.

The updater expects the file size to be a multiple of 4 kilobytes and will write all data received to the flash device. A system update file (128kb .img) or complete ROM can be received and written (512kb or 1024kb .rom)

If the update fails it is recommended that you retry before rebooting or exiting to the Boot loader as your machine may not be bootable.

Option (D) - Duplicate flash #1 to flash #2

This option will make a copy of flash #1 onto flash #2. The purpose of this is to enable making a backup copy of the current flash. Intended for systems using 2x512Kb Flash devices.

Option (V) - Toggle Write Verify

By default each flash sector will be verified after being written. Slight performance improvements can be gained if turned off and could be used if you are experiencing reliable transfers and flashing.

3.8.4 Exit options

Option (R) - Reboot

Execute a cold reboot. This should be done after a successful update. If you perform a cold reboot after a failed update then it is likely that your system will be unusable and removing and reprogramming the flash will be required.

Option (Q) - Quit to boot loader.

The SBC Boot Loader is reloaded from ROM and executed. After a successful update a Reboot should be performed. However, in the case of a failed update this option could be used to attempt to load CP/M and perform the normal x-modem / flash process to recover.

3.8.5 CRC Utility options

Option (1) and (2) - Calculate and display CRC32 of 1st or 2nd 512k ROM. Option (3) - Calculate and display CRC32 of a 1024k (2x512Kb) ROM.

Can be used to verify if a ROM image has been transferred and flashed correctly. Refer to the Tera Term section below for details on configuring the automatic display of a files CRC after it has been transferred.

In Windows, right clicking on a file should also give you a context menu option CRC SHA which will allow you to select a CRC32 calculation to be done on the selected file.

3.8.6 Tera Term macro configuration

Macros are a useful tool for automatic common tasks. There are a number of instances where using macros to facilitate the update process could be worthwhile if you are:

- Following the RomWBW development builds.
- Doing lots of configuration changes.
- Doing development on RomWBW drivers

Macros can be used to automate sending ROM updates or images and for my own purposed I have set up a separate macro for transferring each of the standard build ROM, my own custom configuration ROM and update ROM.

An example macro file to send an *.upd file, using checksum mode and display the crc32 value of the transmitted file:

```
Xmodem send, checksum, display crc32
xmodemsend '\\desktop\users\phillip\documents\github\romwbw\binary\sbc_std_cust.upd' 1
crc32file crc '\\desktop\users\phillip\documents\github\romwbw\binary\sbc_std_cust.rom'
```

```
sprintf '0x%08x' crc
messagebox inputstr 'crc32'
```

3.8.7 Serial speed guidelines

As identified in the introduction, there are limitations on serial speed depending on processor speed and flow control settings. Listed below are some of the results identified during testing.

Configuration	Processor Speed	Maximum Serial Speed
UART no flow control	2MHz	9600
UART no flow control	4MHz	19200
UART no flow control	5MHz	19200
UART no flow control	8MHz	38400
UART no flow control	10MHz	38400
USB-fifo	2MHz+	n/a
ASCI no flow control	18.432MHz	9600
ASCI with flow control	18.432MHz	38400

The **Set Recommend Baud Rate** option in the Updater menu follows the following guidelines.

Processor Speed	Baud Rate
1MHz	4800
2-3MHz	9600
4-7MHz	19200
8-20MHz	38400

These can be customized in the updater.asm source code in the CLKTBL table if desired. Feedback to the RomWBW developers on these guidelines would be appreciated.

3.8.8 Notes

Notes * All testing was done with Tera Term x-modem, Forcing checksum mode using macros was found to give the most reliable transfer. * Partial writes can be completed with 39SF040 chips. Other chips
require entire flash to be erased before being written. * An SBC V2-005 MegaFlash or Z80 MBC required for 1mb flash support. The Updater assumes both chips are same type * Failure handling has not been tested. * Timing broadly calibrated on a Z80 SBC-v2 * Unabios not supported

3.9 User Application

RomWBW provides the facility for a user to build, include and execute their own custom application directly from the applications menu at boot-up. All that's needed is for the user to create their custom code ready for inclusion, recognising that there are certain constraints in doing this.

In order to build properly, the build process requires that the file `usrrom.asm` be found in the /Source/HBIOS folder of the RomWBW tree.

This source file needs to assemble using TASM and it must start at (ORG) address 00100H as the RomWBW HBIOS reserves locations 00000H to 000FFH for internal use. Further, the user application must assemble to a maximum of `USR-SIZ` bytes.

During execution, the user application may make use of HBIOS calls as necessary, and at exit it should return to the RomWBW boot loader using the HBIOS warm reset. Note that no disk operating system (eg CP/M) functions will be available as no disk operating system will have been loaded.

There is a sample `usrrom.asm` supplied in Source/HBIOS and it is recommended that, at least initially, users create their own application based on this as a template because it already creates the necessary variables, starts at (ORG) 00100H, and ensures that the assembled file is padded to create a file `USR-SIZ` in length. Equally, should the the user's application prove too large for the space available then assembly will be terminated with an error. Users should not remove this check from the templated code.

If required, the user application may make use of the Z80 interrupt system but if the user application wishes to rely on HBIOS functionality then it must adhere to the HBIOS framework for managing interupts. Alternatively, if the user appliction has no need for the HBIOS then it may use its own custom code for handling interrupts. In that case, a hard reset, rather than an HBIOS warm start, would be necessary to return control to RomWBW.

Chapter 4

CP/M Applications - ROM-Based & Disk-Based

There now follows a more detailed guide to using the small suite of custom applications included with RomWBW. 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. XM/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 during 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 table below clarifies where each of the applications may be found. It is not an exhaustive list, with further applications existing on both the ROM-based and disk-based versions of CP/M. All of the Applications included within RomWBW may be found within the Binary/Apps directory.

Application	ROM Disk	Boot Disks
ASSIGN	Yes	Yes
BBCBASIC	No	Yes
CLRDIR	Yes	Yes
COPYSL	No	Yes
CPUSPD	Yes	Yes
FAT	No	Yes
FDISK80	Yes	Yes
FDU	Yes	Yes
FLASH	Yes	Yes
FORMAT	Yes	Yes
HTALK	Yes	Yes
MODE	Yes	Yes
REBOOT	Yes	Yes
RTC	Yes	Yes
SURVEY	Yes	Yes
SYSCOPY	Yes	Yes
TALK	Yes	Yes
TBASIC	No	Yes
TIMER	Yes	Yes
TUNE	No	Yes
VGMPLAY	No	Yes
WDATE	No	Yes
XM	Yes	Yes
ZMD	No	Yes
ZMP	No	Yes

All of the CP/M applications may be found in the RomWBW Binary/Apps directory and a user may copy those they need to their own customized disk/slice.

Independently of whether the CP/M system was started from ROM or a boot disk, such as a floppy disk or a slice on a CF or uSD memory card, applications may be located on and executed from either the ROM-disk itself or from other media. There are multiple disk images available for CP/M (eg floppy, legacy hard-disk and new hard-disk formats) and they all contain essentially the same set of applications.

There are particular advantages for each method of booting into CP/M.

ROM-based CP/M:

- A clean and reliable copy of CP/M with no possibility of corruption
- No additional hardware required
- Fast to boot
- Rolled forward with new releases of RomWBW

Disk-based CP/M:

- Greater capacity allows for a larger number of applications
- Allows for user-customisation of applications available
- Allows individual disks to be tailored to a particular purpose, eg word processor

For systems starting CP/M from a disk created from an image file, there are a small number of additional applications stored in the `USER 2` area of the disk. These applications do not form part of CP/M, but rather are small utilities used for test purposes during develpment work. They may, or may not, fuction correctly with any given hardware or software configuration. Documentation for these untilities is very limited, though the source files maybe found in the /Source folder. Note that these utiltites are not available when starting CP/M from the ROM image or from a floppy disk.

A number of the CP/M applications available are described in more detail in the following sections, each with an indication as to whether that application may be found on the ROM-disk, a boot-disk, or both.

4.1 ASSIGN

ASSIGN	
ROM-based	Yes
Disk-based	Yes

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
ASSIGN [<drv>],...
ASSIGN <drv>=[<device>:[<slice>]],...
ASSIGN <tgtdrv>=<srcdrv>,...
ASSIGN /B='*'<option>'*'['*'<option>'*'['*'<option>'*'...]]
```

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 A:` just specifying the drive letter will display the assignment for the drive letter

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

Usage (Specific)

The following describes how to assign drive specifically by identifing each drive by its unique device and slice id's

`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.

The `ASSIGN` command will not allow you to specify a slice (other than zero) for devices that do not support slices. A slice should only be specified for hard disk devices (SD, IDE, PPIDE). Floppy disk drives and RAM/ROM drives do not have slices.

`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 its actions. It will also display the remaining space available in disk buffers.

Usage (Bulk)

The following describes how to assign drives in bulk without having to specify the identifiers of each drive being mapped. Instead bulk mode has a predefined set options (identified by single letter) which will map drives. Bulk mode works by assigning drives sequentially starting at A: until all drives are used, or there are no more options to process. Each option will typically map between 0 and N drives depending on the option and the available hardware in your system.

`ASSIGN /B=<option><option>…` will perform bulk assignment .

The following options will assign a small number of devices, typically you would place at beginning of an option list.

Option	Name	Description	Assigned
B	Boot	The boot device	1
A	RAM	Ram drive	0,1
O	ROM	Rom drive	0,1
F	Floppy	All floppy devices, with/without media	0,1,2,..
P	Preserve	Skip and preserve the next drive assignment	1
X	Exclude	Un-assign / Exclude the next drive	1

A drive e.g. RAM, ROM, FLOPPY can only be assigned if it exists. if you system doesn't have the hardware that supports the device, then no devices will be assigned, and the next option will be processed.

`B` assigns the boot device. If used the `Boot` drive should typically be assigned first.

`P` will not make any changes to the next drive, it will skip over it. While the `X` option will un-assign the next drive, leaving a gap.

The remaining options will fill drives mostly to end, from hard drive slices, generally choose 1 of the following:

Option	Name	Description	Assigned
S	Slices	Assign slices from boot hard drive	…max
H	Hard Drive	Assign slices evenly from all hard drives	…max
L	Legacy HD	Assign slices from all hard drives (legacy)	6,…max
Z	Exclude All	Un-assign all remaining drives	…max

`Slices` assignment will map all remaining drives to slices from the boot device. If I have other hard drives present these will not be mapped by this option.

e.g. `ASSIGN /B=BAOS`

Will first assign drives `A:(Boot), B:(RAM), C:(ROM)` this leaves 13 drives which will be assigned to slices from the boot hard drive (D: thru P:), leaving no unused drives.

'H'ard drive assignment will attempt to fill all remaining drive letters by splitting the number of drives remaining evenly across all.

e.g. `ASSIGN /B=BAOH`

Will first assign drives `A:(Boot), B:(RAM), C:(ROM)` this leaves 13 drives available. If I have 3 hard disks then (13/3) = 4 slices from each hard drive will be assigned to drives (D: thru O:), leaving a single unused drive (P:).

`Legacy` hard drive assignment is identical to how the startup hard disk assignment works. ie. Attempt to assign up to 8 hard drives split across hard drives detected at boot.

e.g. `ASSIGN /B=BAOL`

Will first assign drives `A:(Boot), B:(RAM), C:(ROM)`. If I have 3 hard disks then (8/3) = 2 slices from each hard drive will be assigned to drives (D: thru I:), leaving 7 unused drives (J: thru P:).

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 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.

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 existing drive letter first.

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.

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.

The `ASSIGN` command does **not** prevent you from assigning a drive letter to a slice that does not fit on the physical media. However, any subsequent attempt to refer to that drive letter will result in an immediate OS error of “no disk”. Refer to “Hard Disk Capacity” in the RomWBW User Guide for a discussion of the exact number of slices that will fit on a specific physical disk size.

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.

Additionally, the `ASSIGN` command must be able to adjust to CP/M 2.2 vs. CP/M 3. If you utilize an RSX that modifies the BDOS version returned, you are likely to have serious problems. In this case, be sure to use `ASSIGN` prior to loading the RSX or after it is unloaded.

Etymology

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

4.2 BBCBASIC (BBC BASIC)

BBCBASIC	
ROM-based	No
Disk-based	Yes

BBC BASIC is an interpreted version of the BASIC programming language originally developed by Acorn Computers for the 6502 CPU. Compared to other BASIC implementations, BBC BASIC adds structured programming as well as a built-in Z80 assembler.

Syntax

`BBCBASIC` [*<filename>*]

Usage

The full documentation for BBC BASIC (Z80) is found online at https://www.bbcbasic.co.uk/bbcbasic/mancpm/index.html.

Notes

The cursor and screen management assumes the use of an ANSI/VT-100 terminal which is generally correct for RomWBW. Support for a hardware system timer is also implemented. If your system does not have a hardware timer, the TIME function will always return 0 and the timeout parameter of the INKEY(n) function will not be observed (will never timeout).

Etymology

This is a RomWBW HBIOS adaptation of BBCBASIC v5.00 by R.T.Russell. This implementation was adapted from the source code found at https://github.com/rtrussell/BBCZ80.

The adaptation to RomWBW was minimal and includes:

- VT100 terminal control
- TIME function

4.3 CLRDIR (Clear Directory)

CLRDIR	
ROM-based	Yes
Disk-based	Yes

The `CLRDIR` command is used to initialise the directory area of a drive.

Syntax

`CLRDIR<drv>`

Usage

`CLRDIR<drv>` will initialise the directory area of the specified drive. The drive may take any form - eg floppy disk, hard-disk, CF, uSD etc.

The use of FDISK80 to reserve space, or slices, for CP/M use as drives will not initialise the directory areas of those slices. The resultant directory areas will contain garbage left over from a previous use of the disk (or media) and using them in this state with CP/M will very likely lead to failed or corrupted data storage. Use `CLRDIR` to initialise the directory properly.

FDU will initialise the directory of a floppy disk as part of the formatting process and so `CLRDIR` is unnecessary for a floppy disk. `CLRDIR` is, therefore, primarily used with other types such as hard-disk, CF and uSD.

The `CLRDIR` command may also be used to effectively 'reformat' a used disk by reinitialising its directory area and effectively making it blank again.

Use `CLRDIR` with caution as changes made to disks by `CLRDIR` cannot be undone.

Notes

If `CLRDIR` is used on disk containing data then the directory area will be reinitialised and the data previously stored will be lost.

WARNING: Earlier versions of the `CLRDIR` application do not appear to check for disk errors when it runs. If you attempt to run `CLRDIR` on a drive that is mapped to a slice that does not actually fit on the physical disk, it may behave erratically.

4.4 CPUSPD (CPU Speed)

CPUSPD	
ROM-based	Yes
Disk-based	Yes

The `CPUSPD` application is used to change the running speed and wait states of a RomWBW system. It can also be used to invoke a warm or cold reboot of the system.

The functionality is highly dependent on the capabilities of your system.

Syntax

```
CPUSPD [<speed>[,[<memws>][,[<iows>]]]
CPUSPD (W)armBoot
CPUSPD (C)oldBoot
```

`<speed>` is one of (H)alf, (F)ull, (D)ouble, or (Q)uad. *`<memws>`* is a number specifying the desired memory wait states. *`<iows>`* is a number specifying the desired I/O wait states.

Usage

Entering `CPUSPD` with no parameters will display the current CPU speed and wait state information of the running system. Wait state information is not available for all systems.

To modify the running speed of a system, you can specify the `**` parameter. To modify either or both of the wait states, you can enter the desired number. Either or both of the wait state parameters may be omitted and the current wait state settings will remain in effect.

Notes

The ability to modify the running speed and wait states of a system varies widely depending on the hardware capabilities and the HBIOS configuration settings.

Note that it is frequently impossible to tell if a system is capable of dynamic speed changes. This function makes the changes blindly. If an attempt is made to change the speed of a system that is definitely incapable of doing so, then an error result is returned.

Z180-based systems will be able to adjust their CPU speed depending on the specific variant of the Z180 chip being used:

Z180 Variant	Capability
Z80180 (original)	Half
Z8S180 Rev. K	Half, Full
Z8S180 Rev. N	Half, Full, Double

SBC and MBC systems may be able to change their CPU speed if the hardware supports it and it is enabled in the HBIOS configuration.

The `CPUSPD` command makes no attempt to ensure that the new CPU speed will actually work on the current hardware. Setting a CPU speed that exceeds the capabilities of the system will result in unstable operation or a system stall.

In the case of Z180 CPUs, it is frequently necessary to add memory wait states when increasing the CPU speed.

Some peripherals are dependent on the CPU speed. For example, the Z180 ASCI baud rate and system timer are derived from the CPU speed. The CPUSPD application will attempt to adjust these peripherals for correct operation after modifying the CPU speed. However, in some cases this may not be possible. The baud rate of ASCI ports have a limited set of divisors. If there is no satisfactory divisor to retain the existing baud rate under the new CPU speed, then the baud rate of the ASCI port(s) will be affected.

Etymology

The `CPUSPD` application was custom written for RomWBW. All of the hardware interface code is specific to RomWBW and the application will not operate correctly on non-RomWBW systems.

The source code is provided in the RomWBW distribution.

4.5 COPYSL (Copy Slice)

COPYSL	
ROM-based	No
Disk-based	Yes

The purpose of this utility is to allow the copying of whole disk slices from one disk slice to another slice

This tool is only supported by RomWBW HBIOS, it uses HBIOS for all its disk IO. UNA UBIOS is not supported by this tool.

This tool is running on CP/M 2.2 or 3.0 and has access to full 64kb of RAM, with a minimum of 48kb TPA

This tool only works with hard disk devices, other media types like floppy, are not supported at this time. This tool works across different hard disk device types, even of different physical type

Both hd1k and hd512 are fully supported, however copying from one layout type to the other is not supported.

During operation data is copied in a single read/write pass, data is not verified by default. If there is a write error, it will be reported, and operation will stop.

Syntax

This tool operates at the disk level via RomWBW, thus all disk identifiers are in the RomWBW <disk>.<unit> format.

The syntax (similar to copy) for the command is:

`COPYSL`*<destunit>*[`.`*<slice>*]=*<srcunit>*[`.`*<slice>*] [`/`*<options>*]

E.g.

COPYSL 3.3=2.10 /U

Means copy from slice 10 on disk 2, onto disk 3 slice 3. This is in unattended mode, so you will not be asked to confirm the copy operation.

Options

F - Full disk copy. Copies the complete disk slice, all sectors.
U - Unattended. Will complete copy without confirmation from the user.
V - Verify. Does an additional read and verify after write.

Usage

When run COPYSL will perform command line argument validation and display an error if they are illegal. Also any disk IO errors will cause COPYSL to exit.

When specifying slice number(s) a check is made that the slice number is valid, i.e. not too large that it would extend past the end of the partition (hd1k), or the end of the media (hd512). For hd512 a check is also performed to ensure that the slice would not extend into the first defined partition.

The copy operation will be faster if the source disk has been formatted with the CP/M file system, since during copy the CP/M directory is scanned, and unused blocks are not copied.

If a filesystem is not found, (or the /F option is chosen) all data is copied.

Verification (if option chosen) will do an aditional read (after write) and compare the data read matches what was written. This compare is only on every 32'nd byte. This is done for efficiency.

During copy dots "." will be displayed to indicate progress of the copy. Each "." represents 16 kBytes of data. Each line of "." 's is 1 mBytes.

Notes

This tool has been tested on both SD and CF media types and with hd1k and hd512 formatted media.

You cannot copy slices between different hard disk formats (hd512 and hd1k) because the slices are incompatible.

Etymology

The `COPYSL` application was custom written for RomWBW and contributed by Mark Pruden.

4.6 FAT (FAT Utility)

FAT	
ROM-based	Yes
Disk-based	Yes

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 can occupy up to 8MB. So, make sure to start your FAT partition beyond (slice count) * 9MB.

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.

Only the first 8 RomWBW disk units (0-7) can be referenced.

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.

Known Issues

CP/M (and workalike) OSes have significant restrictions on filename characters. The FAT application will block any attempt to create a file on the CP/M filesystem containing any of these prohibited characters:

```
< > . , ; : ? * [ ] |/ \
```

The operation will be aborted with "`Error: Invalid Path Name`" if such a filename character is encountered.

Since MS-DOS does allow some of these characters, you can have issues when copying files from MS-DOS to CP/M if the MS-DOS filenames use these characters. Unfortunately, FAT is not yet smart enough to substitute illegal characters with legal ones. So, you will need to clean the filenames before trying to copy them to CP/M.

The FAT application does try to detect the scenario where you are copying a file to itself. However, this detection is not perfect and can corrupt a file if it occurs. Be careful to avoid this.

4.7 FDISK80

FDISK80	
ROM-based	Yes
Disk-based	Yes

`FDISK80` allows you to create and manage traditional partitions on your hard disk media. Depending on the hard disk format and features you are using, RomWBW may need hard disk partitions defined.

Please refer to the RomWBW User Guide for more information on the use of partitions within RomWBW. It is very important to understand that RomWBW slices are completely different from disk partitions.

This application is provided by John Coffman. The primary documentation is in the file "FDisk Manual.pdf" found in the Doc directory of the RomWBW distribution.

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.

If 'slices' for CP/M have been created using `FDISK80`, then these will need to have their directory areas initialised properly using `CLRDIR`. Failure to do this will likely result in corrupted data.

There is also more information on using FAT partitions with RomWBW in the RomWBW User Guide document in the Doc directory of the distribution.

Notes

Hard disk partition tables allow a maximum of 1024 cylinders when defining partitions. However, RomWBW uses exclusively Logical Block Addressing (LBA) which does not have this limitation. When defining partitions is usually best to define the start and size of of the partition using bytes or sectors.

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.

4.8 FDU (Floppy Disk Utility)

FDU	
ROM-based	Yes
Disk-based	Yes

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.

4.9 FLASH (Flash EEPROM)

FLASH	
ROM-based	Yes
Disk-based	Yes

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 can prevent the ROM chip from being recognized.

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.

4.10 FORMAT

FORMAT	
ROM-based	Yes
Disk-based	Yes

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.

4.11 HTALK (HBIOS Talk)

HTALK	
ROM-based	Yes
Disk-based	Yes

`HTALK` is a variation of the `TALK` utility, but it works directly against HBIOS Character Units.

Syntax

`HTALK`

Usage

`HTALK` operates at the HBIOS level.

The parameter to `TALK` is a single number referring to an HBIOS character unit. Upon execution all characters typed at the console will be sent to the device specified and all characters received by the specified device will be echoed on the console.

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

Notes

Etymology

The `TALK` command was created and donated to RomWBW by Tom Plano. It is an original product designed specifically for RomWBW.

4.12 MODE

MODE	
ROM-based	Yes
Disk-based	Yes

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.

4.13 REBOOT

REBOOT	
ROM-based	Yes
Disk-based	Yes

The `REBOOT` application is used to restart a running system from an operating system prompt. It can invoke either a warm or cold restart via command line switches.

Syntax

```
REBOOT /W
REBOOT /C
REBOOT /?
```

- /C initiates a cold restart
- /W initiates a warm restart
- /? displays command line usage

Usage

Entering `REBOOT` with no parameters will display the usage and version information. Use /C or /W to immediately initiate a cold or warm restart.

Notes

A warm restart just returns to the Boot Loader menu.

A cold restart will reinitialize the system as though power had been recycled.

Etymology

The `REBOOT` application was custom written for RomWBW by MartinR. All of the hardware interface code is specific to RomWBW and the application will not operate correctly on non-RomWBW systems.

The source code is provided in the RomWBW distribution.

4.14 RTC (Real Time Clock)

RTC	
ROM-based	Yes
Disk-based	yes

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 function 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.
`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 Lynch as part of the original ECB SBC board development. It has since been modified to support most of the hardware variations included with RomWBW.

4.15 SLABEL (Slice Label)

SLABEL	
ROM-based	No
Disk-based	Yes

Display or change the label of a disk slice.

The label applied is only used as informational purposes, displayed by RomWBW when an OS is booted. It has no correlation with any OS volume label scheme that may exist. i.e. It does not affect the CP/M 3 disk label as applied by the `SET` command

Syntax

```
SLABEL [unit.slice=label] [/?]
```

`unit.slice` the disk unit and slice number to apply the new label to. This is in the same format as when booting the system to a disk

`label` is the new disk label to apply to the disk

Usage

`SLABEL` with no arguments will list All existing labels across All disks

`SLABEL 2.5=MYDRIVE` will set the disk label of the 6th slice of disk unit 2

`SLABEL /?` (or other unrecognised parameters) will display a usage message.

Notes

There is no capability to update a label onto media that currently does not have existing media information in the third sector, typically this means only bootable media.

This will only display labels for the first 64 slices of any device. Slices higher than this are currently ignored.

Only bootable RomWBW disk images have a label, which is defined by the OS which is booted. i.e. NZ-COM has a label of “ZSDOS 1.1” since that is the booted OS. Prior to RomWBW 3.5 all disk images were defined with the label “Unlabeled”.

Etymology

The SLABEL application was custom written for RomWBW and contributed by Mark Pruden.

4.16 SURVEY

SURVEY	
ROM-based	Yes
Disk-based	Yes

The SURVEY command interrogates the system for information on disk usage, memory usage and I/O ports used, and reports it to the user.

Syntax

The SURVEY command takes no arguments.

SURVEY

Usage

The results presented by SURVEY include:

1. Information about any drives, within the first eight (ie A: to H:), which have been logged by the system. This includes: the total number of files; the storage capacity occupied by those files; and the capacity remaining on that drive.
2. Information about the the 64KByte CP/M memory map, which is shown diagramatically, and includes: locations and sizes of the TPA (Transient Program Area), CP/M's CCP (Console Command Processor),and BDOS (Basic Disk Operating System).
3. The addresses of active CPU I/O ports.

Notes

The mechanism by which SURVEY discovers I/O ports is very conservative and therefore the list returned may not be exhaustive. In particular, it may fail to discover ports that are 'write-only'.

4.17 SYSCONF (System Configuration)

SYSCONF	
ROM-based	Yes
Disk-based	Yes

System Configuration (`SYSCONF`) is a utility that allows system configuration to be set, dynamically and stored in NVRAM provided by an RTC chip.

(`SYSCONF`) is both a ROM utility ('W' Menu option), and a CP/M application. Noting however the CP/M application is not included on an disk image, it is found in the `Binary/Applications` folder of the RomWBW distribution.

The section "Setting NVRAM Options" in the RomWBW User Guide has additional information on the use of NVRAM to set your system configuration.

Syntax

The application is an interactive application; it does not have a command line syntax. Instead commands are executed from within the application in a command line structure. `SYSCONF` command takes no arguments.

```
SYSCONF
```

Usage

When you first start the (`SYSCONF`) utility it will display the current switches followed by a command listing. e.g.

```
RomWBW System Config Utility

Current Configuration:
  [BO] / Boot Options: ROM (App = "H")
  [AB] / Auto Boot: Disabled

Commands:
  (P)rint - Display Current settings
  (S)et {SW} {val}[,{val}[,{val}]]- Set a switch value(s)
  (R)eset - Init NVRAM to Defaults
  (H)elp [{SW}] - This help menu, or help on a switch
  e(X)it - Exit Configuration
```

$

When you run (SYSCONF) for the first time the NVRAM will be uninitialised, and can be initialised using the (R)eset command, which writes default values to NVRAM.

Updates are done immediately to NVRAM as you enter them, i.e. there is no confirm changes step. If you make any incorrect changes, you simply need to enter a new command to set the Switch value correctly.

Once a change has been made it is available, however it may not take effect until the next system reboot. This is dependent on the Switch itself.

If no NVRAM is provided by your hardware, then running this application will just report the missing hardware and exit immediately.

To exit from the application use the (Q)uit command.

Commands and Syntax

The following are the accepted commands, unless otherwise specified a “Space” character is used to delimit parameters in the command.

Command	Argument(s)	Description
(P)rint	-none-	Display a list of the current switch value(s)
(S)et	{SW} {val},…	Sets an Switch {SW} with specific values(s)
(R)eset	-none-	Reset all setting to default
(H)elp	{SW}	Provides help on the syntax (values)
e(X)it	-none-	Exit the application

Where

Argument	Description
{SW}	Switch ID, typically this is 2 character name to identify the switch
{val},…	a “Comma” separated list of values to set into the switch

Switch Options

Auto Boot (AB)

This switch will define if the system will perform auto boot at the RomWBW boot prompt. Enabling this will not prevent a user from typing a boot command, so long as the timeout

is not exceeded. When configured this replaces the (`AUTO_CMD`) variable defined in build configuration.

Making changes to auto boot has no affect until the next reboot.

Arguments

Type	Arguments	Description
Enable	'E'	Auto Boot. eg. "E,10" will auto boot, after 10 seconds
	Timout	Timeout in seconds in the range 0-15, 0 = immediate
Disabled	'D'	No Auto Boot. e.g. "D" will disable autoboot

Examples

Command	Description
S AB E,10	Enable Auto Boot with 10 second delay
S AB D	Disable Auto Boot

Boot Options (BO)

This switch will define the boot command to be executed when auto boot is enabled. When configured this replaces the (`AUTO_CMD`) variable defined in the ROM build configuration.

Making changes to boot options has no affect until the next reboot.

Arguments

Type	Arguments	Description
Disk	'D'	Disk Boot. eg. "D,2,14" will boot, disk unit 2, slice 14
	Disk Unit Number	Unit number in the range 0-127
	Disk Slice	Slice in the range 0-255, use 0 for floppy boot
ROM	'R'	ROM App. e.g. "R,M" will boot the Monitor App
	Rom App Name	single character used on the Menu to identify the app

Examples

Command	Description
S BO D,2,14	Set the default boot from Disk; Unit 2, Slice 14

Command	Description
S BO R,M	Set the default boot to be the (M)onitor Rom Application

Etymology

The `SYSCONF` utility is an original product specific to RomWBW, source code is included. `SYSCONF` was contributed by Mark Pruden.

4.18 SYSCOPY (System Copy)

SYSCOPY	
ROM-based	Yes
Disk-based	Yes

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 its own operating system. Each slice can be made bootable with its 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 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.

4.19 TALK

TALK	
ROM-based	Yes
Disk-based	Yes

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 typed at the console will be sent to the device specified and all characters received by the specified device will be echoed 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.

4.20 TBASIC (Tasty BASIC)

TBASIC	
ROM-based	No
Disk-based	Yes

Tasty Basic is a basic interpreter for CP/M and RomWBW based on the Z80 port of Palo Alto Tiny Basic.

Syntax

`TBASIC` [<*filename*>]

Usage

Notes

Tasty Basic is provided on RomWBW as both a ROM implementation and as a CP/M application. The CP/M version should be used if you wish to save Tasty Basic files.

Etymology

The implementation of Tasty Basic included in RomWBW is the work of Dimitri Theulings. The primary distribution site for this work is https://github.com/dimitrit/tastybasic.

4.21 TIMER

TIMER	
ROM-based	Yes
Disk-based	Yes

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
TIMER /Z
```

Usage

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

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

Use `TIMER /Z` to zero the seconds counter.

The display of the counter will be something like this:

```
2859 Ticks   24.18 Seconds   0:00:24.18 HH:MM:SS
```

The first number is the total number of ticks since system startup, where there are 50 ticks per second. The second number is the total number of seconds since system startup. Numbers are displayed in decimal format.

Notes

Not all systems will have a system timer. In this case, the `TIMER` command will output 0 for both ticks and seconds and never increment.

The resolution of the timer is determined by the system timer frequency which is typically 50Hz. This means that the seconds fraction will increment 0.02 seconds with each timer tick.

The primary use of the `TIMER` application is to test the system timer functionality of your system. However, it can be used to capture the value before and after some process you want to measure elapsed runtime.

Etymology

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

4.22 TUNE

TUNE	
ROM-based	No
Disk-based	Yes

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.

Note: TUNE will detect an AY-3-8910/YM2149 Sound Module re-gardless of whether support for it is included in the RomWBW HBIOS configuration

Syntax

`TUNE<filename> **`

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

Option	Description
`-MSX`	Force MSX port addresses A0H/A1H (no PSG detection)
`-RC`	Force RCBus port addresses D8H/D0H (no PSG detection)
`--HBIOS`	Utilise HBIOS' sound driver
`-DELAY`	Force delay mode (don't use hardware timer)
`+T1`	Play tune an octave higher
`+T2`	Play tune two octaves higher
`-T1`	Play tune an octave lower
`-T2`	Play tune two octaves lower

The +t and -t options apply only to HBIOS mode operation. The `-MSX`, `-RC`, and `--HBIOS` options are mutually exclusive. See Notes below.

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.

Some hardware (notably Why-Em-Ulator) cannot be detected due limitations of the emulation. In such cases, you can force the use of the two most common port addresses using the `-MSX` or `-RC` options.

On Z180 systems, I/O wait states are added when writing to the sound chip to avoid exceeding its 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 playback. If no system timer is available, a delay loop is calculated instead. The delay loop will not be as accurate as the system timer. If the `-DELAY` options is specified on the command line, then the delay loop will be used regardless of whether the system has a hardware timer. This is useful if the hardware timer does not run at the 50Hz desired for sound playback.

There are two modes of operation. A direct hardware interface for the AY-3-8910 or YM2149 chips, or a compatibility layer thru HBIOS supporting both the AY-3-8910 and 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 following summarizes the different modes of operation for the application:

- If you use `TUNE` with no options, it will use it's original behavior of searching for and detecting a sound chip. `TUNE` will play sound files directly to the PSG hardware. In this mode it does not matter if HBIOS does or does not know about the sound chip.
- If you use `TUNE` with the `--HBIOS` option, it will not detect a sound chip and will use the RomWBW HBIOS interface. This will only work if HBIOS was configured for the installed sound card and HBIOS detects the sound chip.
- If you use `TUNE` with `-RC` or `-MSX`, it will play tunes directly to the PSG hardware (not via HBIOS) and will bypass detection. In this mode it does not matter if HBIOS does or does not know about the sound chip.

Note that the HBIOS API for sound cards is pretty good, but does not implement everything that the sound card can do. For best fidelity, use `TUNE` without the `--HBIOS` option.

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.

4.23 VGMPLAY (Video Game Music Play)

VGMPLAY	
ROM-based	No
Disk-based	Yes

This application will allow you to play Video Game Music files. VGM files contain music samples from a range of different sound chips that were used in arcade games, game consoles and personal computer systems.

Video Game Music files have a .VGM file extension and each file contains an embedded header that identifies the hardware it is intended for and also the title of the music.

All RomWBW operating system boot disks include a selection of sound files in user area 3. Additional music files can be found at:

VGMRIPS website

PROJECT2612 website

Sound files are loaded into memory for playback, so the maximum size file that can be played is around 52Kb.

Sound chips currently supported are:

- AY-3-8190 (and equivalent YM2149)
- YM2612 (and equivalent YM3848)
- SN76489 (single chip mono and dual chip stereo)
- YM2151

VGMPLAY supports playback of files with multiple combinations of these chips.

Syntax

`VGMPLAY<filename>`

<filename> is the name of a sound file ending in .VGM

Usage

VGMPLAY does not automatically detect the hardware platform or sound hardware that you are using. This means a version customized for your system must be assembled before use. However, the version as distributed will work with ECB bus SBC systems.

To play a sound file, just use the VGMPLAY command and specify the file to play after the command. So, for example, `VGMPLAY TEDDY` will load the TEDDY.VGM sound file into memory and begin playing it.

Playback can be stopped by pressing a key. There may be a delay before playback stops.

Notes

The default build configuration for VGMPLAY is:

CPU speed: Autodetected

chip	number	port	notes
AY-3-8910	1st	09ah	stereo
AY-3-8910	2nd	not set	stereo
YM2612	1st	0c0h	stereo
YM2612	2nd	0c4h	stereo
SN76489	1st	0c8h	mono/left
SN76489	2nd	0c9h	mono/right
YM2151	1st	0cah	stereo
YM2151	2nd	0cbh	stereo

Inconsistant, garbled or distorted playback can be an indication that your CPU clock speed is too high for your sound chip. In this case, if your platform supports speed switching, then the CPUSPD application can be used to reduce your processor speed.

VGMPLAY is still under development. The source code is provided in the RomWBW distribution.

4.24 WDATE (WBW DATE)

WDATE	
ROM-based	No
Disk-based	Yes

`wdate` is a utility for CP/M systems that have Wayne Warthen's RomWBW firmware. It reads or sets the real-time clock, using function calls in the BIOS. It should work on any RTC device that is supported by RomWBW, including the internal interrupt-driven timer that is is available on some systems.

`wdate` differs from the `rtc.com` utility that is provided with the RomWBW version of CP/M in that it only gets and sets the date/time. `rtc.com` can also manipulate the nonvolatile RAM in certain clock devices, and modify the charge controller. However, `wdate` is (I would argue) easier to use, as it takes its input from the command line, which can be edited, and it's less fussy about the format. It doesn't require the date to be set if you only want to change the time, for example. In addition, `wdate` has at least some error checking.

`wdate` displays the day-of-week and month as English text, not numbers. It calculates the day-of-week from the year, month, and day. RTC chips usually store a day-of-week value, but it's useless in this application for two reasons: first, the BIOS does not expose it. Second, there is no universally-accepted way to interpret it (which day does the week start on? Is '0' a valid day of the week?)

Syntax

```
WDATE
WDATE <hr> <min>
WDATE <hr> <min> <sec>
WDATE <year> <month> <day> <hr> <min> <sec>
```

Usage

```
A> wdate
Saturday 27 May 13:14:39 2023
```

With no arguments, displays the current date and time.

```
A> wdate hr min
```

With two arguments, sets the time in hours and minutes, without changing date or seconds

```
A> wdate hr min sec
```

With three arguments, sets the time in hours, minutes, and seconds, without changing date

```
A> wdate year month day hr min sec
```

With six arguments, sets date and time. All numbers are one or two digits. The two-digit year starts at 2000.

```
A> wdate /?
```

Show a summary of the command-line usage.

Notes

I've tested this utility with the DS1302 clock board designed by Ed Brindly, and on the interrupt-driven timer built into my Z180 board. However, it does not interact with hardware, only BIOS; I would expect it to work with other hardware.

wdate checks for the non-existence of RomWBW, and also for failing operations on the RTC. It will display the terse "No RTC" message in both cases.

The RomWBW functions that manipulate the date and time operate on BCD numbers, as RTC chips themselves usually do. wdate works in decimal, so that it can check that the user input makes sense. A substantial part of the program's code is taken up by number format conversion and range checking.

Etymology

The `WDATE` application was written and contributed by Kevin Boone. The source code is available on GitHub at https://github.com/kevinboone/wdate-cpm.

4.25 XM (X-Modem)

XM	
ROM-based	Yes
Disk-based	Yes

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>
```

The following may be added to the action codes: | `S`: Send a file | `L`: Send a file from a library | `R`: Receive a file | `K`: Use 1K blocksize (send operations) | `C`: Force use of checksum (receive operations) | `X`: Force 128-byte protocol (receive operations) | `0-9`: Specifies HBIOS character unit for transfers

`<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

For example, the following command will receive a file using checksums on HBIOS character unit 3 and will name the received file `MYFILE.TXT`.

```
XM RC3 MYFILE.TXT
```

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 default to using the current HBIOS console port for transfers. Note that if you change your console port at the OS level (e.g., STAT CON:=UC1:), this does not change the HBIOS console.

`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 on 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.

4.26 ZMD (Z-Modem)

ZMD	
ROM-based	No
Disk-based	Yes

An adaptation of Robert Kramer's Remote CP/M File Transfer Program with support for XModem and YModem transfers.

NOTE: ZMD does not do ZModem transfers. The Z in ZMD refers to Z-System compatibility.

Syntax

`ZMD` *<mode><protocol><unit>* [*<filename>*]

where *<mode>* can be:
`S` – Send file from BBS
`SP` – Send from private area
`A` – Send ARK/ARC/LBR member
`R` – Receive file from YOU
`RP` – Receive in private area
`RW` – Receive without description(s)
`F` – Displays available upload space

<protocol> can be:
`X` – Xmodem 128 byte blocks (CRC)
`C` – Xmodem 128 byte blocks (Checksum)
`K` – Ymodem 1024 byte blocks (CRC only)

and *<unit>* can specify a single digit (0-9) that specifies the RomWBW Character Unit to use for the file transfer.

Usage

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

1. Enter one of the `ZMD` receive commands specifying the name you want to give to the received file (no filename required for ZModem transfers).
2. On your host computer select a file to send and initiate an XModem or YModem send operation.

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

1. Enter one of the `ZMD` 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 an XModem or YModem 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 ZMD adaptation that comes with RomWBW will default to using the current HBIOS console port for transfers. Note that if you change your console port at the OS level (e.g., STAT CON:=UC1:), this does not change the HBIOS console.

`ZMD` 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.

`ZMD` 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

ZMD v1.50 was produced by Robert Kramer. The RomWBW adaptation just uses the RomWBW HBIOS serial API.

4.27 ZMP (Z-Modem Program)

ZMP	
ROM-based	No
Disk-based	Yes

`ZMP` is a terminal program for interacting with a modem attached to your system. It includes X/Y/ZModem file transfer protocols. An actual modem is not required, but you must have a port for ZMP to use that is independent of the console running `ZMP`.

Syntax

`ZMD` *[<unit>]*

<unit> can specify a single digit (0-9) indicating the RomWBW Character Unit to use for the modem port.

Usage

Refer to the file `ZMP.DOC` found on all disk images that include the `ZMP` application.

Notes

`ZMP` requires access to multiple overlay and configuration files to run. It will look for these on the default driver and user area. Depending the operating system used, you may be able to set up a search path and locate these files in other locations. The files used by `ZMP` are:

- `ZMP.HLP`
- `ZMP.DOC`
- `ZMP.CFG`
- `ZMP.FON`
- `ZMXFER.OVR`
- `ZMTERM.OVR`
- `ZMINIT.OVR`
- `ZMCONFIG.OVR`

The `ZMP` console is always the active OS console. If no *<unit>* is specified on the command line, `ZMP` will default to using HBIOS Character Unit 1 as the modem port. Take care to avoid using the same HBIOS Character Unit as both the console and the modem or various strangeness will occur.

`ZMP` is a full screen application and is configured to use ANSI/VT-100 screen control.

`ZMP` does not support the range of port configurations provided by RomWBW. The RomWBW adaptation of `ZMP` ignores the port configuration options within `ZMP`. Instead, you should configure the HBIOS Character Unit using the RomWBW MODE command before launching `ZMP`.

`ZMP` is written in C. As a result, file transfers will be noticeably slower than other assembly language file transfer tools.

Etymology

ZMP was produced by Ron Murray and was based on HMODEM II. Wayne Hortensius updated the source to compile with the latest version of Hi-Tech C and implemented a few enhancements.

The RomWBW overlay was developed by Phil Summers.