RomWBW

User Guide

Version 3.5

Updated 16 Apr 2024

RetroBrew Computers Group
www.retrobrewcomputers.org

Wayne Warthen
wwarthen@gmail.com

Contents

1 Overview **3**

2 Getting Started **5**
- 2.1 Acquiring RomWBW . . . 5
- 2.2 Installation . . . 6
- 2.3 Supported Platforms . . . 7
- 2.4 System Startup . . . 9
- 2.5 Core System Information . . . 10
- 2.6 Hardware Discovery . . . 11
- 2.7 Device Unit Assignments . . . 12

3 Boot Loader Operation **14**
- 3.1 Starting Applications from ROM . . . 15
- 3.2 Starting Operating Systems from Disk . . . 17
  - 3.2.1 Auto-Submit Batch Files . . . 19
- 3.3 System Management . . . 19
  - 3.3.1 Listing Device Inventory . . . 19
  - 3.3.2 Rebooting the System . . . 20
  - 3.3.3 Changing Console and Console speed . . . 20
  - 3.3.4 HBIOS Diagnostic Verbosity . . . 21
- 3.4 Console Takeover . . . 21
- 3.5 Front Panel . . . 22

4 Disk Management **24**
- 4.1 Drive Letter Assignment . . . 25
  - 4.1.1 Default Drive Letter Assignment . . . 27
- 4.2 ROM & RAM Disks . . . 28
- 4.3 Floppy Disks . . . 29
- 4.4 Hard Disks . . . 31
- 4.5 Slices . . . 32
- 4.6 Hard Disk Layouts . . . 34

4.6.1 Checking Hard Disk Layout 35
4.7 Hard Disk Capacity 37

5 Disk Content Preparation 38
5.1 Disk Images 38
5.1.1 Floppy Disk Images 39
5.1.2 Hard Disk Images 40
5.2 In-situ Disk Preparation 45

6 Operating Systems 49
6.1 Digital Research CP/M 2.2 49
6.2 Z-System 50
6.3 NZCOM Automatic Z-System 52
6.4 Digital Research CP/M 3 53
6.5 ZPM3 54
6.6 QP/M 56
6.7 UCSD p-System 57
6.8 FreeRTOS 58
6.9 Fuzix 59

7 Custom Applications 62

8 FAT Filesystem 64
8.1 FAT Filesystem Preparation 65
8.2 FAT Application Usage 66

9 Real Time Clock 69
9.1 Date/Time Utilities 69
9.1.1 WDATE Utility 70
9.1.2 RTC Utility 70
9.2 Date/Time File Stamping 71
9.3 Timezone 72

10 CP/NET Networking 73
10.1 CP/NET Client Setup 74
10.2 CP/NET Sever Setup 76
10.3 CP/NET Usage 77
10.4 Network Boot 79

11 Transferring Files 82
11.1 Serial Port Transfers 82

11.2 Disk Image Transfers . . . 83
11.3 FAT Filesystem Transfers . . . 83

12 Customizing RomWBW 85
12.1 Startup Command Processing . . . 85
12.2 ROM Customization . . . 86

13 UNA Hardware BIOS 87
13.1 UNA Usage Notes . . . 88

14 Upgrading 89
14.1 Application Boot . . . 90
14.2 Upgrading via Flash Utility . . . 90
14.3 Upgrading via XModem Flash Updater . . . 91
14.4 Post Upgrade System Image and Application Update Process . . . 92
14.5 System Update . . . 93

15 Related Projects 94
15.1 Z88DK . . . 94
15.2 Paleo Editor . . . 94
15.3 p-System Volume Management Script . . . 94
15.4 Z80 fig-FORTH . . . 95
15.5 RomWBW Date/Time Utility . . . 95
15.6 Assembly Language Programming for the RC2014 Zed . . . 95

16 Acknowledgments 96

17 Licensing 99

18 Getting Assistance 101

19 Appendixes 102
19.1 Appendix A - Pre-built ROM Images . . . 103
19.1.1 RetroBrew Z80 SBC . . . 104
19.1.2 RetroBrew Z80 SimH . . . 105
19.1.3 RetroBrew N8 Z180 SBC . . . 106
19.1.4 Zeta Z80 SBC . . . 107
19.1.5 Zeta V2 Z80 SBC . . . 108
19.1.6 Mark IV Z180 SBC . . . 109
19.1.7 RCBus Z80 CPU Module . . . 110
19.1.8 RCBus Z180 CPU Module . . . 112

19.1.9 RCBus Z280 CPU Module 115
19.1.10 Easy Z80 SBC 117
19.1.11 Tiny Z80 SBC 118
19.1.12 Z80-512K CPU/RAM/ROM Module 119
19.1.13 Small Computer SC126 Z180 SBC 120
19.1.14 Small Computer SC130 Z180 SBC 122
19.1.15 Small Computer SC131 Z180 Pocket Computer 124
19.1.16 Small Computer SC140 Z180 CPU Module 125
19.1.17 Small Computer SC503 Z180 CPU Module 127
19.1.18 Small Computer SC700 Z180 CPU Module 129
19.1.19 Dyno Z180 SBC 131
19.1.20 Nhyodyne Z80 MBC 132
19.1.21 Rhyophyre Z180 SBC 134
19.1.22 Z80 ZRC CPU Module 135
19.1.23 Z80 ZRC512 CPU Module 138
19.1.24 Z180 Z1RCC CPU Module 140
19.1.25 Z280 ZZRCC CPU Module 141
19.1.26 Z280 ZZ80MB SBC 144
19.1.27 Z80-Retro SBC 145
19.1.28 S100 Computers Z180 146
19.1.29 Duodyne Z80 System 147
19.1.30 Heath H8 Z80 System 149
19.1.31 EP Mini-ITX Z180 150
19.1.32 NABU w/ RomWBW Option Board 151
19.2 Appendix B - Device Summary 152

Preface

This document is a general usage guide for the RomWBW software and is generally the best place to start with RomWBW. There are several companion documents you should refer to as appropriate:

- RomWBW System Guide discusses much of the internal design and construction of RomWBW. It includes a reference for the RomWBW HBIOS API functions.
- RomWBW ROM Applications is a reference for the ROM-hosted applications provided with RomWBW including the monitor, programming languages, etc.
- RomWBW Applications is a reference for the OS-hosted proprietary command line applications that were created to enhance RomWBW.
- RomWBW Disk Catalog is a reference for the contents of the disk images provided with RomWBW. It is somewhat out of date at this time.
- RomWBW Errata is updated as needed to document issues or anomalies discovered in the current software distribution.

Since RomWBW is purely a software product for many different platforms, the documentation does **not** cover hardware construction, configuration, or troubleshooting – please see your hardware provider for this information.

Each of the operating systems and ROM applications included with RomWBW are sophisticated tools in their own right. It is not reasonable to fully document their usage here. However, you will find complete manuals in PDF format in the Doc directory of the distribution. The intention of this document is to describe the operation of RomWBW and the ways in which it enhances the operation of the included applications and operating systems.

On a personal note, I found this document very difficult to write. Members of the retro-computing community have dramatically different experiences, skill levels, and desires. I realize some readers will find this document far too basic. Others will find it lacking in many areas. I am doing my best and encourage you to provide constructive feedback.

Conventions

Size Suffixes Within this document and in RomWBW in general, the use of size suffixes KB, MB, GB, and TB refer to the binary variant as shown below. The modern suffixes (KiB, MiB, etc.) are not used here because they were not prevalent during the time that the RomWBW OSes were used. This keeps all of RomWBW and associated applications consistent.

Suffix	Value	Meaning
KB	1024	1,024 bytes
MB	1024^2	1,048,576 bytes
GB	1024^3	1,073,741,824 bytes
TB	1024^4	1,099,511,627,776 bytes

Links and URLs Many of the references in this document to Internet addresses (URLs) do not provide the address in the text. However, these links are embedded and "clickable" within the documents. Your PDF viewer should highlight these links in some manner (typically an alternate color or an underline).

Chapter 1

Overview

RomWBW software provides a complete, commercial quality implementation of CP/M (and workalike) operating systems and applications for modern Z80/180/280 retro-computing hardware systems. A wide variety of platforms are supported including those produced by these developer communities:

- RetroBrew Computers (https://www.retrobrewcomputers.org)
- RC2014 (https://rc2014.co.uk),
  RC2014-Z80 (https://groups.google.com/g/rc2014-z80)
- Retro Computing (https://groups.google.com/g/retro-comp)
- Small Computer Central (https://smallcomputercentral.com/)

A complete list of the currently supported platforms is found in the Installation section.

General features include:

- Z80 Family CPUs including Z80, Z180, and Z280
- Banked memory services for several banking designs
- Disk drivers for RAM, ROM, Floppy, IDE ATA/ATAPI, CF, SD, USB, Zip, Iomega
- Serial drivers including UART (16550-like), ASCI, ACIA, SIO
- Video drivers including TMS9918, SY6545, MOS8563, HD6445
- Keyboard (PS/2) drivers via VT8242 or PPI interfaces
- Real time clock drivers including DS1302, BQ4845
- OSes: CP/M 2.2, ZSDOS, CP/M 3, NZ-COM, ZPM3, QPM, p-System, and FreeRTOS
- Built-in VT-100 terminal emulation support

RomWBW is distributed as both source code and pre-built ROM and disk images. Some of the provided software can be launched directly from the ROM firmware itself:

- System Monitor
- Operating Systems (CP/M 2.2, ZSDOS)
- ROM BASIC (Nascom BASIC and Tasty BASIC)
- ROM Forth

A dynamic disk drive letter assignment mechanism allows mapping operating system drive letters to any available disk media. Additionally, mass storage devices (IDE Disk, CF Card, SD Card, etc.) support the use of multiple slices (up to 256 per device). Each slice contains a complete CP/M filesystem and can be mapped independently to any drive letter. This overcomes the inherent size limitations in legacy OSes and allows up to 2GB of accessible storage on a single device.

The pre-built ROM firmware images are generally suitable for most users. However, it is also very easy to modify and build custom ROM images that fully tailor the firmware to your specific preferences. All tools required to build custom ROM firmware under Windows are included – no need to install assemblers, etc. The firmware can also be built using Linux or MacOS after confirming a few standard tools have been installed.

Multiple disk images are provided in the distribution. Most disk images contain a complete, bootable, ready-to-run implementation of a specific operating system. A "combo" disk image contains multiple slices, each with a full operating system implementation. If you use this disk image, you can easily pick whichever operating system you want to boot without changing media.

By design, RomWBW isolates all of the hardware specific functions in the ROM chip itself. The ROM provides a hardware abstraction layer such that all of the operating systems and applications on a disk will run on any RomWBW-based system. To put it simply, you can take a disk (or CF/SD/USB Card) and move it between systems transparently.

A tool is provided that allows you to access a FAT-12/16/32 filesystem. The FAT filesystem may be coresident on the same disk media as RomWBW slices or on stand-alone media. This makes exchanging files with modern OSes such as Windows, MacOS, and Linux very easy.

Chapter 2

Getting Started

2.1 Acquiring RomWBW

The RomWBW Repository (https://github.com/wwarthen/RomWBW) on GitHub is the official distribution location for all project source and documentation. The fully-built distribution releases are available on the RomWBW Releases Page (https://github.com/wwarthen/RomWBW/releases) of the repository. On this page, you will normally see a Development Snapshot as well as recent stable releases. Unless you have a specific reason, I suggest you stick to the most recent stable release. Expand the "Assets" drop-down for the release you want to download, then select the asset named RomWBW-vX.X.X-Package.zip. The Package asset includes all pre-built ROM and Disk images as well as full source code. The other assets contain only source code and do not have the pre-built ROM or disk images.

All source code and distributions are maintained on GitHub. Code contributions are very welcome.

Distribution Directory Layout

The RomWBW distribution is a compressed zip archive file organized in a set of directories. Each of these directories has its own ReadMe.txt file describing the contents in detail. In summary, these directories are:

Directory	Description
Binary	The final output files of the build process are placed here. Most importantly, the ROM images with the file names ending in ".rom" and disk images ending in .img.

Directory	Description
Doc	Contains various detailed documentation, both RomWBW specifically as well as the operating systems and applications.
Source	Contains the source code files used to build the software and ROM images.
Tools	Contains the programs that are used by the build process or that may be useful in setting up your system.

2.2 Installation

In general, installation of RomWBW on your platform is very simple. You just need to program your ROM with the correct ROM image from the RomWBW distribution. Subsequently, you can write disk images on your disk drives (IDE disk, CF Card, SD Card, etc.) which then provides even more functionality.

NOTE: The pre-built ROM images distributed with RomWBW are based on the default system configurations as determined by the hardware provider/designer. This document does not provide hardware construction or configuration information. Please contact your hardware provider/designer as needed.

The pre-built ROM images will automatically detect and support typical devices for their corresponding platform including serial ports, video adapters, on-board disk interfaces, and PropIO/ParPortProp boards without building a custom ROM. The distribution is a .zip archive. After downloading it to a working directory on your modern computer (Windows/Linux/Mac) use any zip tool to extract the contents of the archive.

Depending on how you got your hardware, you may have already been provided with a pre-programmed ROM chip. If so, use that initially. Otherwise, you will need to use a ROM programmer to initially program your ROM chip. Please refer to the documentation that came with your ROM programmer for more information.

The Binary directory of the distribution contains the pre-built ROM and disk images. Refer to Supported Platforms below to identify the correct ROM image for your system.

Once you have a running RomWBW system, you can generally update your ROM to a newer version in-situ with the included ROM Flashing tool (Will Sowerbutts' FLASH application) as described in the Upgrading chapter of this document.

2.3 Supported Platforms

The table below summarizes the hardware platforms currently supported by RomWBW along with the standard pre-built ROM image(s). RomWBW does allow for the creation of ROM images with custom configurations. This is discussed in Customizing RomWBW.

Description	Bus	ROM Image File	Baud Rate
RetroBrew Z80 SBC[1]	ECB	SBC_std.rom	38400
RetroBrew Z80 SimH[1]	-	SBC_simh.rom	38400
RetroBrew N8 Z180 SBC[1] (date code >= 2312)	ECB	N8_std.rom	38400
Zeta Z80 SBC[2], ParPortProp	-	ZETA_std.rom	38400
Zeta V2 Z80 SBC[2], ParPortProp	-	ZETA2_std.rom	38400
Mark IV Z180 SBC[3]	ECB	MK4_std.rom	38400
RCBus Z80 CPU Module[4], 512K RAM/ROM	RCBus	RCZ80_std.rom	115200
RCBus Z80 CPU Module[4], 512K RAM/ROM, KIO	RCBus	RCZ80_kio.rom	115200
RCBus Z180 CPU Module[4] w/ external banking	RCBus	RCZ180_ext.rom	115200
RCBus Z180 CPU Module[4] w/ native banking	RCBus	RCZ180_nat.rom	115200
RCBus Z280 CPU Module[4] w/ external banking	RCBus	RCZ180_ext.rom	115200
RCBus Z280 CPU Module[4] w/ native banking	RCBus	RCZ180_nat.rom	115200
Easy Z80 SBC[2]	RCBus	RCZ80_easy.rom	115200
Tiny Z80 SBC[2]	RCBus	RCZ80_tiny.rom	115200
Z80-512K CPU/RAM/ROM Module[2]	RCBus	RCZ80_skz.rom	115200
Small Computer SC126 Z180 SBC[5]	BP80	SCZ180_sc126.rom	115200
Small Computer SC130 Z180 SBC[5]	RCBus	SCZ180_sc130.rom	115200
Small Computer SC131 Z180 Pocket Computer[5]	-	SCZ180_sc131.rom	115200
Small Computer SC140 Z180 CPU Module[5]	Z50	SCZ180_sc140.rom	115200
Small Computer SC503 Z180 CPU Module[5]	Z50	SCZ180_sc503.rom	115200
Small Computer SC700 Z180 CPU Module[5]	RCBus	SCZ180_sc700.rom	115200
Dyno Z180 SBC[6]	Dyno	DYNO_std.rom	38400
Nhyodyne Z80 MBC[1]	MBC	MBC_std.rom	38400
Rhyophyre Z180 SBC[1]	-	RPH_std.rom	38400
Z80 ZRC CPU Module[7]	RCBus	RCZ80_zrc.rom	115200
Z80 ZRC CPU Module[7] ROMless	RCBus	RCZ80_zrc_ram.rom	115200
Z80 ZRC512 CPU Module[7]	RCBus	RCZ80_zrc512.rom	115200
Z180 Z1RCC CPU Module[7]	RCBus	RCZ180_z1rcc.rom	115200
Z280 ZZRCC CPU Module[7]	RCBus	RCZ280_zzrcc.rom	115200
Z280 ZZRCC CPU Module[7] ROMless	RCBus	RCZ280_zzrcc_ram.rom	115200
Z280 ZZ80MB SBC[7]	RCBus	RCZ280_zz80mb.rom	115200
Z80-Retro SBC[8]	-	Z80RETRO_std.rom	38400

Description	Bus	ROM Image File	Baud Rate
S100 Computers Z180[9]	S100	S100_std.rom	57600
Duodyne Z80 System[1]	Duo	DUO_std.rom	38400
Heath H8 Z80 System[10]	H8	HEATH_std.rom	115200
EP Mini-ITX Z180[11]	RCBus?	EPITX_std.rom	115200
NABU w/ RomWBW Option Board[10]	NABU	NABU_std.rom	115200

[1] Designed by Andrew Lynch
[2] Designed by Sergey Kiselev
[3] Designed by John Coffman
[4] RCBus compliant (multiple products/designers)
[5] Designed by Stephen Cousins
[6] Designed by Steve Garcia
[7] Designed by Bill Shen
[8] Designed by Peter Wilson
[9] Designed by John Monahan
[10] Designed by Les Bird
[11] Designed by Alan Cox

RCBus refers to Spencer Owen's RC2014 bus specification and derivatives including RC26, RC40, RC80, and BP80.

Additional information for each of the system configurations supported by the ROM images listed above is found in Appendix A - Pre-built ROM Images.

The RCBus Z180 & Z280 require a separate RAM/ROM memory module. There are two types of these modules and you must pick the correct ROM for your type of memory module. The first option is the same as the 512K RAM/ROM module for RC/BP80 Bus. This is called external ("ext") because the bank switching is performed externally from the CPU. The second type of RAM/ROM module has no bank switching logic – this is called native ("nat") because the CPU itself provides the bank switching logic. Only Z180 and Z280 CPUs have the ability to do bank switching in the CPU, so the ext/nat selection only applies to them. Z80 CPUs have no built-in bank switching logic, so they are always configured for external bank switching.

All pre-built ROM images are pure binary files (they are not "hex" files). They are intended to be programmed starting at the very start of the ROM chip (address 0). Most of the pre-built images are 512KB in size. If your system utilizes a larger ROM, you can just program the image into the first 512KB of the ROM for now.

Initially, don't worry about trying to write a disk image to any disk (or CF/SD/USB) devices you have. This will be covered later. You will be able to boot and check out your system with just the ROM.

Connect a serial terminal or computer with terminal emulation software to the primary serial port of your CPU board. You may need to refer to your hardware provider's documentation for details. A null-modem connection may be required. Set the baud rate as indicated in the table above. Set the line characteristics to 8 data bits, 1 stop bit, no parity, and no flow control. If possible, select ANSI or VT-100 terminal emulation. Hardware flow control is not required for terminal operation, but may be necessary for Serial Port Transfers.

RomWBW will automatically attempt to detect and support typical add-on components for each of the systems supported. More information on the required system configuration and optional supported components for each ROM is found in Appendix A - Pre-built ROM Images.

2.4 System Startup

Upon power-up, your terminal should display a sign-on banner within 2 seconds followed by hardware inventory and discovery information. When hardware initialization is completed, a boot loader prompt allows you to choose a ROM-based operating system, system monitor, application, or boot from a disk device.

Here is an example of a fairly typical startup. Your system will have different devices and configuration, but the startup should look similar.

```
RomWBW HBIOS v3.1.1-pre.183, 2022-10-04

RCBus [RCZ80_kio] Z80 @ 7.372MHz
0 MEM W/S, 1 I/O W/S, INT MODE 2, Z2 MMU
512KB ROM, 512KB RAM
ROM VERIFY: 00 00 00 00 PASS

KIO: IO=0x80 ENABLED
CTC: IO=0x84 TIMER MODE=TIM16
AY: MODE=RCZ80 IO=0xD8 NOT PRESENT
SIO0: IO=0x89 SIO MODE=115200,8,N,1
SIO1: IO=0x8B SIO MODE=115200,8,N,1
DSRTC: MODE=STD IO=0xC0 NOT PRESENT
MD: UNITS=2 ROMDISK=384KB RAMDISK=256KB
FD: MODE=RCWDC IO=0x50 NOT PRESENT
IDE: IO=0x10 MODE=RC
IDE0: NO MEDIA
IDE1: NO MEDIA
```

```
PPIDE: IO=0x20
PPIDE0: LBA BLOCKS=0x00773800 SIZE=3815MB
PPIDE1: NO MEDIA

Unit        Device      Type              Capacity/Mode
----------  ----------  ----------------  --------------------
Char 0      SIO0:       RS-232            115200,8,N,1
Char 1      SIO1:       RS-232            115200,8,N,1
Disk 0      MD0:        RAM Disk          256KB,LBA
Disk 1      MD1:        ROM Disk          384KB,LBA
Disk 2      IDE0:       Hard Disk         --
Disk 3      IDE1:       Hard Disk         --
Disk 4      PPIDE0:     CompactFlash      3815MB,LBA
Disk 5      PPIDE1:     Hard Disk         --
```

If your system completes the ROM-based boot process successfully, you should see the RomWBW Boot Loader prompt. For example:

```
RCBus [RCZ80_kio] Boot Loader

Boot [H=Help]:
```

If you get to this prompt, your system has completed the boot process and is ready to accept commands. Note that the Boot Loader is not an operating system or application. It is essentially the point where you choose which operating system or application you want RomWBW to execute.

The Boot Loader is explained in detail in the next section. For now, you can try a few simple commands to confirm that you can interact with the system.

At the Boot Loader prompt, you can type `H <enter>` for help. You can type `L <enter>` to list the available built-in ROM applications. If your terminal supports ANSI escape sequences, you can try the 'P' command to play a simple on-screen game. Instructions for the game are found in RomWBW ROM Applications.

If all of this seems fine, your ROM has been successfully programmed. See the Boot Loader Operation section of this document for further instructions on use of the Boot Loader.

2.5 Core System Information

During startup, the first few lines of information displayed provide the most basic information on your system. In the example above, these lines are the Core System Information:

```
RomWBW HBIOS v3.1.1-pre.183, 2022-10-04

RCBus [RCZ80_kio] Z80 @ 7.372MHz
0 MEM W/S, 1 I/O W/S, INT MODE 2, Z2 MMU
512KB ROM, 512KB RAM
ROM VERIFY: 00 00 00 00 PASS
```

The first line is a version identification banner for RomWBW. After that you see a group of 4 lines describing the basic system. In this example, the platform is the RCBus running a configuration named "RCZ80_kio". The CPU is a Z80 with a current clock speed of 7.372 MHz. There are 0 memory wait states and 1 I/O wait state. Z80 interrupt mode 2 is active and the bank memory manager is type "Z2" which is standard for RCBus. The system has 512KB of ROM total and 512KB of RAM total. Finally, a verification of the checksums of the critical ROM banks is shown (all 4 should be 00).

RomWBW attempts to detect the running configuration of the system at startup. Depending on your hardware, there may be inaccuracies in this section. For example, in some cases the CPU clock speed is assumed rather than actually measured. This does not generally affect the operation of your system. If you want to correct any of the information displayed, you can create a custom ROM which is described later.

2.6 Hardware Discovery

The next set of messages during boot show the hardware devices as they are probed and initially configured. In the example above, these lines are:

```
KIO: IO=0x80 ENABLED
CTC: IO=0x84 TIMER MODE=TIM16
AY: MODE=RCZ80 IO=0xD8 NOT PRESENT
SIO0: IO=0x89 SIO MODE=115200,8,N,1
SIO1: IO=0x8B SIO MODE=115200,8,N,1
DSRTC: MODE=STD IO=0xC0 NOT PRESENT
MD: UNITS=2 ROMDISK=384KB RAMDISK=256KB
FD: MODE=RCWDC IO=0x50 NOT PRESENT
IDE: IO=0x10 MODE=RC
IDE0: NO MEDIA
IDE1: NO MEDIA
PPIDE: IO=0x20
PPIDE0: LBA BLOCKS=0x00773800 SIZE=3815MB
PPIDE1: NO MEDIA
```

What you see will depend on your specific system and ROM, but should match the hardware present in your system. Each device has a tag that precedes the colon. This tag identifies the driver and instance of each device. For example, the tag “SIO0:” refers to the SIO serial port driver and specifically the first channel. The “SIO1:” tag refers to the second channel.

In many cases you will see IO=0xNN in the data following the tag. This identifies the base I/O port address of the hardware device and is useful for identifying hardware conflicts.

Note that you may see some lines indicating that the associated hardware is not present. Above, you can see that the FD driver did not find a floppy interface. Lines such as these are completely normal when your system does not have the associated hardware.

Finally, be aware that all ROMs are configured to identify specific hardware devices at specific port addresses. If you add hardware to your system that is not automatically identified, you may need to build a custom ROM to add support for it. Building a custom ROM is covered later.

Appendix B - Device Summary contains a list of the RomWBW hardware devices which may help you identify the hardware discovered in your system.

2.7 Device Unit Assignments

In order to support a wide variety of hardware, RomWBW HBIOS uses a modular approach to implementing device drivers and presenting devices to an operating system. In general, all devices are classified as one of the following:

- Disk (RAM/ROM Disk, Floppy Disk, Hard Disk, CF Card, SD Card, etc.)
- Character (Serial Ports, Parallel Ports, etc.)
- Video (Video Display/Keyboard Interfaces)
- Sound (Audio Playback Devices)
- RTC/NVRAM (Real Time Clock, Non-volatile RAM)
- System (Internal Services, e.g. Timer, DMA, etc.)

HBIOS uses the concept of unit numbers to present a generic set of hardware devices to the operating system. As an example, a typical system might have a ROM Disk, RAM Disk, Floppy Drives, and Disk Drives. All of these are considered disk devices and are presented to the operating system as generic block devices. This means that each operating system does not need to embed code to interact directly with all of the different hardware devices – RomWBW takes care of that.

In the final group of startup messages, a device unit summary table is displayed so that you can see how the actual hardware devices have been mapped to unit numbers during startup.

```
Unit        Device      Type              Capacity/Mode
----------  ----------  ----------------  --------------------
Char 0      UART0:      RS-232            38400,8,N,1
Char 1      UART1:      RS-232            38400,8,N,1
Disk 0      MD1:        RAM Disk          384KB,LBA
Disk 1      MD0:        ROM Disk          384KB,LBA
Disk 2      FD0:        Floppy Disk       3.5",DS/HD,CHS
Disk 3      FD1:        Floppy Disk       3.5",DS/HD,CHS
Disk 4      IDE0:       CompactFlash      3815MB,LBA
Disk 5      IDE1:       Hard Disk         --
Disk 6      PRPSD0:     SD Card           1886MB,LBA
Video 0     CVDU0:      CRT               Text,80x25
```

In this example, you can see that the system has a total of 7 Disk Units numbered 0-6. There are also 2 Character Units and 1 Video Unit. The table shows the unit numbers assigned to each of the devices. Notice how the unit numbers are assigned sequentially regardless of the specific device.

There may or may not be media in the disk devices listed. For example, the floppy disk devices (Disk Units 2 & 3) may not have a floppy in the drive. Also note that Disk Unit 4 shows a disk capacity, but Disk Unit 5 does not. This is because the PPIDE interface of the system supports up to two drives, but there is only one actual drive attached. A unit number is assigned to all available devices regardless of whether they have actual media installed at boot time.

Note that Character Unit 0 is the initial system console unless modified in a customized ROM image.

If your system has an RTC/NVRAM device, it will not be listed in the unit summary table. Since only a single RTC/NVRAM device can exist in one system, unit numbers are not required nor used for this type of device. Also, System devices are not listed because they are entirely internal to RomWBW.

Chapter 3

Boot Loader Operation

Once your system has completed the startup process, it presents a Boot Loader command prompt. The purpose of the Boot Loader is to select and launch a desired application or operating system. It also has the ability to configure some aspects of system operation.

After starting your system, following the hardware initialization, you will see the RomWBW Boot Loader prompt. Below is an example. Note that the text preceding "Boot Loader" will vary and identifies your specific system and configuration.

```
Mark IV [MK4_wbw] Boot Loader

Boot [H=Help]:
```

From the Boot Loader prompt, you can enter commands to select and launch any of the RomWBW operating systems or ROM applications. It also allows you to manage some basic settings of the system. To enter a command, just enter the command followed by ***<enter>***.

For example, typing `H<enter>` will display a short command summary:

```
Boot [H=Help]: h

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

Likewise the `L` command will display the list of ROM Applications that you can launch right from the Boot Loader:

```
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
  N: Network Boot
  X: XModem Flash Updater
  U: User App
```

3.1 Starting Applications from ROM

To start a ROM application you just enter the corresponding letter at the Boot Loader prompt. In the following example, we launch the built-in Microsoft BASIC interpreter. From within BASIC, we use the `BYE` command to return to the Boot Loader:

```
Boot [H=Help]: b

Loading BASIC...
Memory top?
Z80 BASIC Ver 4.7b
Copyright (C) 1978 by Microsoft
55603 Bytes free
Ok
bye


Mark IV [MK4_wbw] Boot Loader

Boot [H=Help]:
```

The following ROM applications and OSes are available at the boot loader prompt:

Application	Description
Monitor	Z80 system debug monitor w/ Intel Hex loader
CP/M 2.2	Digital Research CP/M 2.2 OS
Z-System	ZSDOS 1.1 w/ ZCPR 1 (Enhanced CP/M compatible OS)
Forth	Brad Rodriguez's ANSI compatible Forth language
Tasty BASIC	Dimitri Theuling's Tiny BASIC implementation
Play	A simple video game (requires ANSI terminal emulation)
Network Boot	Boot system via Wiznet MT011 device
Flash Update	Upload and flash a new ROMWBW image using xmodem
User App	User written application placeholder

The User App is provided as a way to access a custom written ROM module. In the pre-built ROMs, selecting User App will just return to the Boot Loader menu. If you are interested in creating a custom application to run here, review the "usrrom.asm" file in the Source/HBIOS folder of the distribution.

Each of the ROM Applications is documented in RomWBW ROM Applications. Some of the applications (such as BASIC) also have their own independent manual in the Doc directory of the distribution. The OSes included in the ROM (CP/M 2.2 & Z-System) are described in the Operating Systems chapter of this document.

In general, the command to exit any of these applications and restart the system is `BYE`. The exceptions are the Monitor which uses `B` and Play which uses `Q`.

Two of the ROM Applications are, in fact, complete operating systems. Specifically, "CP/M 2.2" and "Z-System" are provided so that you can actually start either operating system directly from your ROM. This technique is useful when:

- You don't yet have any real disk drives in your system
- You want to setup real disk drives for the first time
- You are upgrading your system and need to upgrade your real disk drives

The RAM disk and ROM disk drives will be available even if you have no physical disk devices attached to your system.

Booting an operating system from ROM is not intended as a way to use your operating system on a long-term basis. The ROM disk has only a small subset of the operating system files. Additionally, you cannot easily customize your ROM disk because you cannot write to it. For any significant use of an operating system, you should boot directly to the disk/slice that contains the complete operating system. This is described in the next section.

3.2 Starting Operating Systems from Disk

In order to make use of the more sophisticated operating systems available with RomWBW, you will need to boot an operating system from a disk. Setting up disks is described in detail later. For now, we will just go over the command line for performing this type of boot.

From the Boot Loader prompt, you can enter a number (***<diskunit>***) and optionally a dot followed by a second number (***<slice>***). The ***<disk unit>*** unit number refers to a disk unit that was displayed when the system was booted – essentially it specifies the specific physical disk drive you want to boot. The ***<slice>*** numbers refers to a portion of the disk unit to boot. If no slice is specified, then it is equivalent to booting from the first slice (slice 0). Disk units and slices are described in more detail later.

Following this, you should see the operating system startup messages. Your operating system prompt will typically be `A>` and when you look at the drive letter assignments, you should see that A: has been assigned to the disk and slice you selected to boot.

If you receive the error message "Disk not bootable!", you have either failed to properly initialize the disk and slice requested or you have selected an invalid/unavailable disk/slice.

The following example shows a disk boot into the first slice of disk unit 4 which happens to be the CP/M 2.2 operating system on this disk. This is accomplished by entering just the number '4' and pressing ***<enter>***.

```
Boot [H=Help]: 4

Booting Disk Unit 4, Slice 0, Sector 0x00000800...

Volume "Unlabelled" [0xD000-0xFE00, entry @ 0xE600]...

CBIOS v3.1.1-pre.194 [WBW]

Formatting RAMDISK...

Configuring Drives...

        A:=IDE0:0
        B:=MD0:0
        C:=MD1:0
        D:=FD0:0
        E:=FD1:0
        F:=IDE0:1
```

```
        G:=IDE0:2
        H:=IDE0:3
        I:=PRPSD0:0
        J:=PRPSD0:1
        K:=PRPSD0:2
        L:=PRPSD0:3

        1081 Disk Buffer Bytes Free

CP/M-80 v2.2, 54.0K TPA

A>
```

Notice that a list of drive letters and their assignments to RomWBW devices and slices is displayed during the initialization of the operating system.

Here is another example where we are booting disk unit 4, slice 3 which is the CP/M 3 operating system on this disk:

```
Boot [H=Help]: 4.3

Booting Disk Unit 4, Slice 3, Sector 0x0000C800...

Volume "Unlabelled" [0x0100-0x1000, entry @ 0x0100]...

CP/M V3.0 Loader
Copyright (C) 1998, Caldera Inc.

 BNKBIOS3 SPR  F600  0800
 BNKBIOS3 SPR  4500  3B00
 RESBDOS3 SPR  F000  0600
 BNKBDOS3 SPR  1700  2E00

 60K TPA

CP/M v3.0 [BANKED] for HBIOS v3.1.1-pre.194

A>
```

Some operating systems (such as CP/M 3 shown above) do not list the drive assignments during initialization. In this case, you can use the `ASSIGN` command to display the current assignments.

The Boot Loader simply launches whatever is in the disk unit/slice you have specified. It does not know what operating system is at that location. The layout of operating systems on disk media is described in the Disk Images section of this document.

3.2.1 Auto-Submit Batch Files

All of the operating systems supplied with RomWBW have the ability to execute a "batch" of commands by creating a batch submission file containing the commands to be executed. The specifics of using batch files in a specific operating system is covered in its specific documentation.

At boot, the operating system will look for a specific batch file (`PROFILE.SUB` for CP/M 2.2 and 3) on the boot drive and execute that batch file automatically. This allows you to automatically customize your operating system with any commands desired at boot. CP/M 2.2 did not originally have the ability to automatically excute a batch file at boot, but the CBIOS in RomWBW has added this capability.

Since RomWBW can utilize many disk slices, it is very easy to create slices for specific workflows (editing, software development, games, etc.). You can then just boot to the slice that is optimized for the task you want to perform.

3.3 System Management

3.3.1 Listing Device Inventory

The device units available in your system are listed in the boot messages. However, if that list has scrolled off of your screen, you can use the 'D' command to display a list of them at any time from the Boot Loader prompt.

```
Unit        Device      Type              Capacity/Mode
----------  ----------  ----------------  --------------------
Char 0      ASCI0:      RS-232            38400,8,N,1
Char 1      ASCI1:      RS-232            38400,8,N,1
Char 2      UART0:      RS-232            38400,8,N,1
Char 3      UART1:      RS-232            38400,8,N,1
Char 4      UART2:      RS-232            38400,8,N,1
Char 5      UART3:      RS-232            38400,8,N,1
Char 6      TERM0:      Terminal          Video 0,ANSI
Char 7      PRPCON0:    Terminal          Term Module,ANSI
Disk 0      MD0:        RAM Disk          352KB,LBA
Disk 1      MD1:        Flash Drive       384KB,LBA
```

```
Disk 2      FD0:        Floppy Disk      3.5",DS/HD,CHS
Disk 3      FD1:        Floppy Disk      3.5",DS/HD,CHS
Disk 4      IDE0:       CompactFlash     3815MB,LBA
Disk 5      IDE1:       Hard Disk        --
Disk 6      IDE2:       CompactFlash     3823MB,LBA
Disk 7      IDE3:       Hard Disk        --
Disk 8      IDE4:       Hard Disk        --
Disk 9      IDE5:       Hard Disk        --
Disk 10     SD0:        SD Card          --
Disk 11     PRPSD0:     SD Card          15193MB,LBA
Video 0     TMS0:       CRT              Text,40x24
Sound 0     SND0:       AY-3-8910        3+1 CHANNELS
```

3.3.2 Rebooting the System

The 'R' command within the Boot Loader performs a software reset of the system. The system will perform a startup just like powering up or pressing the hardware reset button (although the hardware is not physically reset).

There is generally no need to do this, but it can be convenient when you want to see the boot messages again or ensure your system is in a clean state.

```
Boot [H=Help]: r

Restarting System...
```

3.3.3 Changing Console and Console speed

Your system can support a number of devices for the console. They may be VDU type devices or serial devices. If you want to change which device is the console, the ***I*** menu option can be used to choose the unit and its speed.

The command format is `I <unit> [<baudrate>]`

where ***<unit>*** is the character unit to select and ***<baudrate>*** is the optional baud rate.

Supported baud rates are:

```
 75   450  1800   7200  38400  115200   460800  1843200
150   600  2400   9600  28800  153600   614400  2457600
225   900  3600  14400  57600  230400   921600  3686400
300  1200  4800  19200  76800  307200  1228800  7372800
```

Here is an example of changing the console to unit #1 (the second serial port) and switching the port to 9600 baud:

```
Boot [H=Help]: i 1 9600

  Change speed now. Press a key to resume.

  Console on Unit #1
```

At this point, the Boot Loader prompt will be displayed on character unit #1.

Note that not all character devices support changing baud rates and some only support a limited subset of the baud rates listed. If you attempt to select an invalid baud rate for your system, you will get an error message.

3.3.4 HBIOS Diagnostic Verbosity

The 'V' command of the Boot Loader allows you to view and optionally change the level of diagnostic messages that RomWBW will produce. The normal verbosity level is 4, which means to display only fatal errors. You can increase this level to see more warnings when function calls to RomWBW HBIOS detect problems.

The use of diagnostic levels above 4 are really intended only for software developers. I do not recommend changing this under normal circumstances.

3.4 Console Takeover

If your system has more than one character unit, then the Boot Loader will "poll" all of the character devices for a request to make any of the alternate character devices the active console. This is called a console takeover request. This functionality must be enabled in the ROM build configuration, but currently it is for all standard ROMs.

To request a console takeover, you just press the <space> character twice in a row at the port or terminal that you want to move the console to. The terminal or communication software **must** be configured for the default serial port speed and data bits for this to work.

A takeover request is only possible while the active console is showing the Boot Loader prompt prior to typing any characters at the active console. In other words, once you start typing at the active console prompt, the takeover polling is suspended. If you have started typing characters, you can press <enter> at the active console to get a fresh Boot Loader prompt and reactivate the polling.

If you have built a custom ROM that includes an automatic boot command with a timeout, then performing a console takeover will abort the timeout process and the automatic boot command will not be performed.

3.5 Front Panel

RomWBW supports the concept of a simple front panel. The following image is a conceptual view of such a front panel. If your system has a front panel, it should look similar to the RomWBW Front Panel.

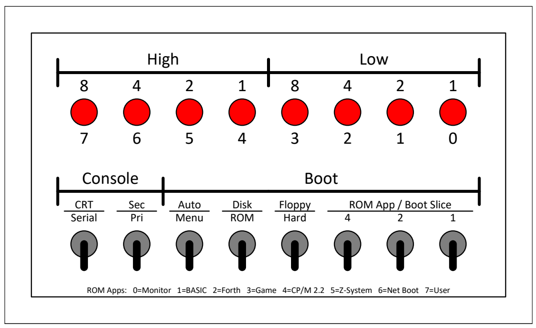

Figure 3.1: RomWBW Front Panel

The LEDs in the top row of the panel are used for multiple purposes. They are initially used to display the progress of the system boot. This may help in diagnosing a hardware or configuration issue in a system that does not progress far enough to display text output on the console. The meaning of the LEDs is:

0-------	System Boot has started
00------	Common RAM bank activated
000-----	HBIOS transitioned to RAM
0000----	Basic initialization done
00000---	CPU detection complete
000000--	System timer configured
0000000-	Pre-console device init done
00000000	Console activation

Once the system has booted, the LEDs are used to indicate disk device activity. Each LED numbered 7-0 represents disk units 7-0. As each disk device performs I/O, the LED will light while the disk is active. This is only possible for the first 8 disk units.

The second row of the front panel is composed of switches that allow you to control a few aspects of the system startup.

The first two switches affect the device used as the console initially. Setting the CRT/Serial switch will cause the system to boot directly to an attached CRT device (if available). Setting the Pri/Sec switch will cause the system to boot to the secondary Serial or CRT device (depending on the setting of the first switch).

The final six switches allow you to cause the system to automatically boot into a desired function. The Auto/Menu switch must be set to enable this, otherwise the normal ROM Loader prompt will be used. If the Disk/ROM switch is not set, then you can use the last 3 switches to select a ROM app to auto-start. If the Disk/ROM switch is set, then the system will attempt a disk boot based on the following switches. The Floppy/Hard switch can be used to boot to a Floppy or Hard Disk. In either case, the first Floppy or Hard Disk will be used for the boot. If a Hard Disk boot is selected, then the last three switches can be used to select any of the first 8 slices.

Chapter 4

Disk Management

The systems supported by RomWBW all have the ability to use persistent disk media. A wide variety of disk devices are supported including floppy drives, hard disks, CF Cards, SD Cards, etc. RomWBW also supports the use of extra RAM and ROM memory as pseudo-disk devices.

RomWBW supports a variety of storage devices which will be discussed in more detail later.

- ROM Disk
- RAM Disk
- Floppy Disk
- Hard Disk (includes CF Cards, SD Cards, USB Drives, etc.)

We will start by discussing each of these types of storage devices and how to prepare them so files can be stored on them. Subsequently, we will describe how to install the pre-built disk images with bootable operating systems and ready-to-run content.

Some systems have disk interfaces built-in, while others will require add-in cards. You will need to refer to the documentation for your system for your specific options.

In the RomWBW boot messages, you will see hardware discovery messages. If you have a disk drive interface, you should see messages listing device types like FD:, IDE:, PPIDE:, SD:. Additionally, you will see messages indicating the media that has been found on the interfaces. As an example, here are the messages you might see if you have an IDE interface in your system with a single CF Card inserted in the primary side of the interface:

```
IDE: IO=0x80 MODE=MK4
IDE0: 8-BIT LBA BLOCKS=0x00773800 SIZE=3815MB
IDE1: NO MEDIA
```

The messages you see will vary depending on your hardware and the media you have installed. But, they will all have the same general format as the example above.

Once your system has working disk devices, they will be accessible from any operating system you choose to run. Disk storage is available whether you boot your OS from ROM or from the disk media itself.

Referring back to Starting Applications from ROM, you could start CP/M 2.2 using the 'C' command. As the operating system starts up, you should see a list of drive letters assigned to the disk media you have installed. Here is an example of this:

```
Configuring Drives...

   A:=MD1:0
   B:=MD0:0
   C:=IDE0:0
   D:=IDE0:1
```

You will probably see more drive letters than this. The drive letter assignment process is described below in the Drive Letter Assignment section. Be aware that RomWBW will only assign drive letters to disk interfaces that actually have media in them. If you do not see drive letters assigned as expected, refer to the prior system boot messages to ensure media has been detected in the interface. Actually, there is one exception to this rule: floppy drives will be assigned a drive letter regardless of whether there is any media inserted at boot.

Notice how each drive letter refers back to a specific disk hardware interface like IDE0. This is important as it is telling you what each drive letter refers to. Also notice that mass storage disks (like IDE) will normally have multiple drive letters assigned. The extra drive letters refer to additional "slices" on the disk. The concept of slices is described below in the Slices section.

4.1 Drive Letter Assignment

In legacy CP/M-type operating systems, drive letters were generally mapped to disk drives in a completely fixed way. For example, drive A: would **always** refer to the first floppy drive. Since RomWBW supports a wide variety of hardware configurations, it implements a much more flexible drive letter assignment mechanism so that any drive letter can be assigned to any disk device.

At boot, you will notice that RomWBW automatically assigns drive letters to the available disk devices. These assignments are displayed during the startup of the selected operating system. Additionally, you can review the current drive assignments at any time using the `ASSIGN` command. CP/M 3 and ZPM3 do not automatically display the assignments at startup, but you can use ASSIGN to display them. Refer to RomWBW Applications for more information on use of the ASSIGN command.

Here is an example of the list of drive letter assignments made during the startup of Z-System:

```
Loading Z-System...

CBIOS v3.1.1-pre.194 [WBW]

Formatting RAMDISK...

Configuring Drives...

        A:=MD0:0
        B:=MD1:0
        C:=FD0:0
        D:=FD1:0
        E:=IDE0:0
        F:=IDE0:1
        G:=IDE0:2
        H:=IDE0:3

        1081 Disk Buffer Bytes Free

ZSDOS v1.1, 54.0K TPA
```

Above you can see that drive A: has been assigned to MD0 which is the RAM Disk device. Drives C: and D: have been assigned to floppy drives. Drives E: thru L: have been assigned to the IDE0 hard disk device. The 4 entries for IDE0 are referring to 4 slices on that disk. Slices are discussed later.

WARNING: Drive letter assignments do **not** ensure that the slice referenced by the drive letter actually fits on the media you are using. For example, a typical 64MB CF Card (which is typically a bit smaller than 64MB) will only fit 7 slices. At startup, you will typically see 8 drive letters assigned to the CF Card. Attempting to access the last drive letter will result in a "no disk" error from the operating system.

The drive letter assignments **do not** change during an OS session unless you use the ASSIGN command yourself to do it. Additionally, the assignments at boot will stay the same on each boot as long as you do not make changes to your hardware configuration. Note that the assignments **are** dependent on the media currently inserted in hard disk drives when the operating system is started. So, notice that if you insert or remove an SD Card, CF Card or

USB Drive, the drive assignments will change. Since drive letter assignments can change, you must be careful when doing destructive things like using `CLRDIR` to make sure the drive letter you use is referring to the desired media.

When performing a ROM boot of an operating system, note that A: will be your RAM disk and B: will be your ROM disk. When performing a disk boot, the disk you are booting from will be assigned to A: and the rest of the drive letters will be offset to accommodate this. This is done because most legacy operating systems expect that A: will be the boot drive.

4.1.1 Default Drive Letter Assignment

As shown above, when an operating system is booted, RomWBW will automatically assign drive letters to physical disk devices. The assignment process varies depending on: 1) the drive/slice you choose to boot from, and 2) the number and type of physical drives in your system.

If you boot an operating system from ROM, then the first two drive letters will be assigned to your RAM disk (A:) and your ROM disk (B:). It may seem odd that the RAM disk is assigned to A: in this case. The reason for this is to accommodate certain functions that require that A: be a writable disk drive. For example, A: **must** be writable in order to submit batch files.

If you boot to a physical disk device, then the first drive letter (A:) will be assigned to the disk/slice that you chose to boot from. The A: drive letter is considered special by most operating systems and is automatically used in some cases. By making the selected disk/slice the A: drive, you can setup different disks/slices for specific uses and just boot to it.

After the first drive letter is assigned (as well as the second drive letter in the case of a ROM boot), RomWBW will assign additional drive letters based on the disk drives in the system. Additional drive letters will be assigned in the following order:

- RAM Disk
- ROM Disk
- Floppy Disk(s)
- Hard Disk(s)

If a disk/slice was already assigned as the A: (or B:) drive letter, then it will not be assigned again.

In the case of floppy, RAM, and ROM disks, a single drive letter will be assigned to each physical disk (even if there is no disk media in the drive).

In the case of hard disks, 1-8 drive letters will be assigned to the initial 1-8 slices of the disk drive. The number of drive letters assigned to each hard disk depends on the number of hard disks in the system:

- 1 Hard Disk: 8 drive letters (slices)
- 2 Hard Disks: 4 drive letters (slices) per disk
- 3+ Hard Disks: 2 drive letters (slices) per disk

This somewhat complicated algorithm is used to try and maximize the limited number of operating system drive letters available (16) to the available disk devices as evenly as possible.

Note that for hard disk devices, drive letters will only be assigned to disk devices that actually contain media. So, for example, if you have an SD Card slot in your system, but it has no SD Card inserted, then no drive letters will be assigned to it.

Since drive letter assignments are easily changed at any time using the `ASSIGN` command, you can customize your assignments as desired after starting the operating system. Even better, you can use an auto-submit batch file to customzie the assignments at startup without any user intervention.

4.2 ROM & RAM Disks

A typical RomWBW system has 512KB of ROM and 512KB of RAM. Some portions of each are dedicated to loading and running applications and operating system. The space left over is available for an operating system to use as a pseudo-disk device (ROM Disk and RAM Disk).

The RAM disk provides a small CP/M filesystem that you can use for the temporary storage of files. Unless your system has a battery backed mechanism for persisting your RAM contents, the RAM disk contents will be lost at each power-off. However, the RAM disk is an excellent choice for storing temporary files because it is very fast. You will notice that the first time an operating system is started after the power was turned off, you will see a message indicating that the RAM disk is being formatted. If you reset your system without turning off power, the RAM disk will not be reformatted and it's contents will still be intact.

Like the RAM disk, the ROM disk also provides a small CP/M filesystem, but it's contents are static – they are part of the ROM. As such, you cannot save files to the ROM disk. Any attempt to do this will result in a disk I/O error. The contents of the ROM disk have been chosen to provide a core set of tools and applications that are helpful for either CP/M 2.2 or ZSDOS. Since ZSDOS is CP/M 2.2 compatible, this works fairly well. However, you will find some files on the ROM disk that will work with ZSDOS, but will not work on CP/M 2.2. For example, `LDDS`, which loads the ZSDOS date/time stamper will only run under ZSDOS.

Unlike other types of disk devices, ROM and RAM Disks do not contain an actual operating system and are not "bootable". However, they are accessible to any operating system (whether the operating system is loaded from ROM or a different disk device).

Neither RAM nor ROM disks require explicit formatting or initialization. ROM disks are pre-formatted and RAM disks are formatted automatically with an empty directory when first used.

Flash ROM Disks

The limitation of ROM disks being read-only can be overcome on some platforms with the appropriate selection of Flash ROM chip and system configuration. In this case the flash-file system can be enabled which will allow the ROM disk to be read and written to. Flash devices have a limited write lifespan and continual usage will eventually wear out the device. It is not suited for high usage applications. Enabling ROM disk writing requires building a custom ROM.

4.3 Floppy Disks

If your system has the appropriate hardware, RomWBW will support the use of floppy disks. The supported floppy disk formats are generally derived from the IBM PC floppy disk formats:

- 5.25” 360K Double-sided, Double-density
- 5.25” 1.2M Double-sided, High-density
- 3.5” 720K Double-sided, Double-density
- 3.5” 1.44M Double-sided, High-density

When supported, RomWBW is normally configured for 2 3.5” floppy drives. If a high-density drive is used, then RomWBW automatically detects and adapts to double-density or high-density media. It cannot automatically detect 3.5” vs. 5.25” drive types – the ROM must be pre-configured for the drive type.

Floppy media must be physically formatted before it can be used. This is normally accomplished by using the supplied Floppy Disk Utility (FDU) application. This application interacts directly with your hardware and therefore you must specify your floppy interface hardware at application startup. Additionally, you need to specify the floppy drive and media format to use for formatting.

Below is a sample session using FDU to format a 1.44M floppy disk in the first (primary) floppy disk drive:

```
B>fdu

Floppy Disk Utility (FDU) v5.8, 26-Jul-2021 [HBIOS]
Copyright (C) 2021, Wayne Warthen, GNU GPL v3
```

```
SELECT FLOPPY DISK CONTROLLER:
  (A) Disk IO ECB Board
  (B) Disk IO 3 ECB Board
  (C) Zeta SBC Onboard FDC
  (D) Zeta 2 SBC Onboard FDC
  (E) Dual IDE ECB Board
  (F) N8 Onboard FDC
  (G) RCBus SMC (SMB)
  (H) RCBus WDC (SMB)
  (I) SmallZ80 Expansion
  (J) Dyno-Card FDC, D1030
  (K) RCBus EPFDC
  (L) Multi-Board Computer FDC
  (X) Exit
=== OPTION ===> D-IDE

===== D-IDE ===========<< FDU MAIN MENU >>======================
(S)ETUP: UNIT=00  MEDIA=720KB DS/DD    MODE=POLL        TRACE=00
----------------------------------------------------------------
(R)EAD          (W)RITE         (F)ORMAT        (V)ERIFY
(I)NIT BUFFER   (D)UMP BUFFER   FDC (C)MDS      E(X)IT
=== OPTION ===> SETUP
ENTER UNIT [00-03] (00):
00: 3.5" 720KB - 9 SECTORS, 2 SIDES, 80 TRACKS, DOUBLE DENSITY
01: 3.5" 1.44MB - 18 SECTORS, 2 SIDES, 80 TRACKS, HIGH DENSITY
02: 5.25" 320KB - 8 SECTORS, 2 SIDES, 40 TRACKS, DOUBLE DENSITY
03: 5.25" 360KB - 9 SECTORS, 2 SIDES, 40 TRACKS, DOUBLE DENSITY
04: 5.25" 1.2MB - 15 SECTORS, 2 SIDES, 80 TRACKS, HIGH DENSITY
05: 8" 1.11MB - 15 SECTORS, 2 SIDES, 77 TRACKS, DOUBLE DENSITY
06: 5.25" 160KB - 8 SECTORS, 1 SIDE, 40 TRACKS, DOUBLE DENSITY
07: 5.25" 180KB - 9 SECTORS, 1 SIDE, 40 TRACKS, DOUBLE DENSITY
08: 5.25" 320KB - 8 SECTORS, 1 SIDE, 80 TRACKS, DOUBLE DENSITY
09: 5.25" 360KB - 9 SECTORS, 1 SIDE, 80 TRACKS, DOUBLE DENSITY
ENTER MEDIA [00-09] (00): 01
00: POLLING (RECOMMENDED)
01: INTERRUPT (!!! READ MANUAL !!!)
02: FAST INTERRUPT (!!! READ MANUAL !!!)
03: INT/WAIT (!!! READ MANUAL !!!)
04: DRQ/WAIT (!!! NOT YET IMPLEMENTED!!!)
ENTER MODE [00-04] (00):
```

```
ENTER TRACE LEVEL [00-01] (00):

===== D-IDE ===========<< FDU MAIN MENU >>======================
(S)ETUP: UNIT=00  MEDIA=1.44MB DS/HD   MODE=POLL        TRACE=00
----------------------------------------------------------------
(R)EAD          (W)RITE         (F)ORMAT        (V)ERIFY
(I)NIT BUFFER   (D)UMP BUFFER   FDC (C)MDS      E(X)IT
=== OPTION ===> FORMAT (T)RACK, (D)ISK ===> DISK
ENTER INTERLEAVE [01-12] (02):



RESET DRIVE...
PROGRESS: TRACK=4F HEAD=01 SECTOR=01

===== D-IDE ===========<< FDU MAIN MENU >>======================
(S)ETUP: UNIT=00  MEDIA=1.44MB DS/HD   MODE=POLL        TRACE=00
----------------------------------------------------------------
(R)EAD          (W)RITE         (F)ORMAT        (V)ERIFY
(I)NIT BUFFER   (D)UMP BUFFER   FDC (C)MDS      E(X)IT
=== OPTION ===> EXIT
```

Since the physical format of floppy media is the same as that used in a standard MS-DOS/Windows computer, you can also physical format floppy media in a modern computer. However, the directory format itself will not be compatible with CP/M OSes. In this case, you can use the `CLRDIR` application supplied with RomWBW to reformat the directory area.

Once a floppy disk is formatted, you can read/write files on it using any of the RomWBW operating systems. The specific commands will depend on the operating system or application in use – refer to the appropriate OS/application documentation as needed.

WARNING: Some of the operating systems provided with RomWBW require that a soft-reset be performed when swapping floppy disk media. For example, under CP/M 2.2, you must press control-C at the CP/M prompt after inserting a new floppy disk. The consequences of failing to perform the soft-reset vary from unexpected error messages to full disk directory corruption.

4.4 Hard Disks

Under RomWBW, a hard disk is similar to a floppy disk in that it is considered a disk unit. However, RomWBW has multiple features that allow its legacy operating systems to take advantage of modern mass storage media.

To start with, the concept of a hard disk in RomWBW applies to any storage device that provides at least 8MB of space. The actual media can be a real spinning hard disk, a CompactFlash Card, a SD Card, etc. In this document, the term hard disk will apply equally to all of these.

RomWBW uses Logical Block Addressing (LBA) to interact with all hard disks. The RomWBW operating systems use older Cylinder/Head/Sector (CHS) addressing. To accommodate the operating systems, RomWBW emulates CHS addressing. Specifically, it makes all hard disks look like they have 16 sectors and 16 heads. The number of tracks varies with the size of the physical hard disk.

It is recommended that hard disk media used with RomWBW be 1GB or greater in capacity. The reasons for this are discussed later, but it allows you to use the recommended disk layout for RomWBW that accommodates 64 CP/M filesystem slices and a 384KB FAT filesystem.

> Although we have not yet discussed how to get content on your disk units, it is necessary to have a basic understanding of how RomWBW handles disk devices as background. The following sections explain how disk units are managed within the operating systems. We will subsequently discuss how to actually setup disk devices with usable content.

4.5 Slices

The vintage operating systems included with RomWBW were produced at a time when mass storage devices were quite small. CP/M 2.2 could only handle filesystems up to 8MB. In order to achieve compatibility across all of the operating systems supported by RomWBW, the hard disk filesystem format used is 8MB. This ensures any filesystem will be accessible to any of the operating systems.

Since storage devices today are quite large, RomWBW implements a mechanism called slicing to allow up to 256 8MB filesystems on a single large storage device. This allows up to 2GB of usable space on one media. You can think of slices as a way to refer to any of the first 256 8MB chunks of space on a single media.

Note that although you can use up to 256 slices per physical disk, this large number of slices is rarely used. The recommended RomWBW disk layout provides for 64 slices which is more than enough for most use cases.

Of course, the problem is that CP/M-like operating systems have only 16 drive letters (A:-P:) available. Under the covers, RomWBW allows you to use any drive letter to refer to any slice of any media. The ASSIGN command is used to view or change the current drive letter mappings at any time. At startup, the operating system will automatically allocate a reasonable number of drive letters to the available storage devices. The allocation will depend on the number of mass storage devices available at boot. For example, if you have only one hard disk type media, you will see that 8 drive letters are assigned to the first 8 slices of that media. If you have two large storage devices, you will see that each device is allocated 4 drive letters.

Referring to slices within a storage device is done by appending a `:<n>` where <n> is the device relative slice number from 0-255. For example, if you have an IDE device, it will show up as IDE0: in the boot messages meaning the first IDE device. To refer to the fourth slice of IDE0, you would type "IDE0:3". Here are some examples:

`IDE0:0`	First slice of disk in IDE0
`IDE0:`	First slice of disk in IDE0
`IDE0:3`	Fourth slice of disk in IDE0

So, if you wanted to use drive letter L: to refer to the fourth slice of IDE0, you could use the command `ASSIGN L:=IDE0:3`. There are a couple of rules to be aware of when assigning drive letters. First, you may only refer to a specific device/slice with one drive letter at a time. Said another way, you cannot have multiple drive letters referring to a the same device/slice at the same time. Second, there must always be a drive assigned to A:. Any attempt to violate these rules will be blocked by the ASSIGN command.

In case this wasn't already clear, you **cannot** refer directly to slices using CP/M. CP/M only understands drive letters, so to access a given slice, you must assign a drive letter to it first.

While it may be obvious, you cannot use slices on any media less than 8MB in size. Specifically, you cannot slice RAM disks, ROM disks, floppy disks, etc. All of these are considered to have a single slice and any attempt to ASSIGN a drive letter to a slice beyond that will result in an error message.

It is very important to understand that RomWBW slices are not individually created or allocated on your hard disk. RomWBW uses a single, large chunk of space on your hard disk to contain the slices. You should think of slices as just an index into a sequential set of 8MB areas that exist in this large chunk of space. The next section will go into more detail on how slices are located on your hard disk.

Although you do not need to allocate slices individually, you do need to initialize each slice for CP/M to use it. This is somewhat analogous to doing a FORMAT operation on other systems. With RomWBW you use the CLRDIR command to do this. This command is merely "clearing out" the directory space of the slice referred to by a drive letter and setting up the new empty directory. Since CLRDIR works on drive letters, make absolutely sure you know what media and slice are assigned to that drive letter before using `CLRDIR` because CLRDIR will wipe out any pre-existing contents of the slice.

WARNING: The `CLRDIR` application does 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.

Here is an example of using `CLRDIR`. In this example, the `ASSIGN` command is used to show the current drive letter assignments. Then the `CLRDIR` command is used to initialize the directory of drive 'G' which is slice 2 of hard disk device IDE0 ("IDE0:2").

```
B>assign

   A:=MD0:0
   B:=MD1:0
   C:=FD0:0
   D:=FD1:0
   E:=IDE0:0
   F:=IDE0:1
   G:=IDE0:2
   H:=IDE0:3

B>clrdir G:
CLRDIR Version 1.2 April 2020 by Max Scane

Warning - this utility will overwrite the directory sectors of Drive: G
Type Y to proceed, any key other key to exit. Y
Directory cleared.
B>
```

4.6 Hard Disk Layouts

As previously discussed, when RomWBW uses a hard disk, it utilizes a chunk of space for a sequential series of slices that contain the actual CP/M filesystems referred to by drive letters.

Originally, RomWBW always used the very start of the hard disk media for the location of the slices. In this layout, slice 0 referred to the first chunk of ~8MB on the disk, slice 1 referred to the second chunk of ~8MB on the disk, and so on. The number of slices is limited to the size of the disk media – if you attempted to read/write to a slice that would exceed the disk size, you would see I/O errors. This is considered the "legacy" disk layout for RomWBW.

Starting with v3.2, RomWBW has been enhanced to support the concept of partitioning. The partition mechanism is entirely compliant with Master Boot Record (MBR) Partition Tables introduced by IBM for the PC. The Wikipedia article on the Master Boot Record is excellent if you are not familiar with them. This is considered the "modern" disk layout for RomWBW. RomWBW uses the partition type id 0x2E. This partition type id does not equate to any existing well-known partition types – it was chosen because it is not generally used. RomWBW does not support extended partitions – only a single primary partition can be used.

Both the legacy and modern disk layouts continue to be fully supported by RomWBW. There are no plans to deprecate the legacy layout. In fact, the legacy format takes steps to allow a partition table to still be used for other types of filesystems such as DOS/FAT. It just does not use a partition table entry to determine the start of the RomWBW slices.

There is one more difference between the legacy and modern disk layouts that should be highlighted. The CP/M filesystem in the slices of the legacy disk layout contain 512 directory entries. The modern disk layout filesystems provide 1024 directory entries. In fact, you will subsequently see that the prefixes "hd512" and "hd1k" are used to identify disk images appropriate for the legacy and modern format. These prefixes were chosen specifically to highlight the number of directory entries supported.

You **cannot** mix disk layouts on a single disk device. To say it another way, the existence of a partition table entry for RomWBW on a hard disk makes it behave in the modern mode. The lack of a RomWBW partition table entry will cause legacy behavior. Adding a partition table entry on an existing legacy RomWBW hard disk will cause the existing data to be unavailable and/or corrupted. Likewise, removing the RomWBW partition entry from a modern hard disk layout will cause the same problems. It is perfectly fine for one system to have multiple hard disks with different layouts – each physical disk device is handled separately.

If you are setting up a new disk, the modern (hd1k) layout is recommended for the following reasons:

- Larger number of directory entries per filesystem
- Simplifies creation of coresident FAT filesystem
- Reduces chances of data corruption

4.6.1 Checking Hard Disk Layout

If you are not sure which hard disk layout was used for your existing media, you can use the OSes command to display the number of directory entries for a drive letter on the corresponding hard disk. For example, the `STAT` command is used in CP/M 2.2 to determine this. This command displays the number of directory entries on a filesystem. If it indicates 512, your disk layout is legacy (hd512). If it indicates 1024, your disk layout is modern (hd1k).

Here is an example of checking the disk layout. We want to check the CompactFlash Card inserted in IDE interface 0. We start the system and boot to Z-System in ROM by using the 'Z' command at the Boot Loader. As Z-System starts, we see the following disk assignments:

```
Boot [H=Help]: c

Loading CP/M 2.2...

CBIOS v3.1.1-pre.194 [WBW]

Configuring Drives...

        A:=MD0:0
        B:=MD1:0
        C:=FD0:0
        D:=FD1:0
        E:=IDE0:0
        F:=IDE0:1
        G:=IDE0:2
        H:=IDE0:3
        I:=PRPSD0:0
        J:=PRPSD0:1
        K:=PRPSD0:2
        L:=PRPSD0:3

        1081 Disk Buffer Bytes Free

CP/M-80 v2.2, 54.0K TPA
```

You can see that the IDE0 interface (which contains the CF Card) has been assigned to drive letters E: to H:. We can use the STAT command on any of these drive letters. So, for example:

```
B>stat e:dsk:

    E: Drive Characteristics
65408: 128 Byte Record Capacity
 8176: Kilobyte Drive  Capacity
 1024: 32  Byte Directory Entries
    0: Checked  Directory Entries
  256: Records/ Extent
   32: Records/ Block
   64: Sectors/ Track
```

```
  2: Reserved Tracks
```

It is critical that you include "dsk:" after the drive letter in the `STAT` command line. The important line to look at is labelled "32 Byte Directory Entries". In this case, the value is 1024 which implies that this drive is located on a modern (hd1k) disk layout. If the value was 512, it would indicate a legacy (hd512) disk layout.

4.7 Hard Disk Capacity

Although RomWBW can support many CP/M filesystem slices on a single hard disk, you are still constrained by the physical capacity of the actual hard disk. RomWBW does not prevent you from assigning slices to drive letters even if the location of the slice does not fit on the physical disk. Any attempt to access a drive letter mapped to a slice that does not fit will result in an error such as "no disk" from the operating system.

The exact number of CP/M filesystem slices that will fit on your specific physical hard disk can be determined as follows:

- For hd512 disk layouts, it is slices * 8,320KB.
- For hd1k disk layouts, it is 1024KB + (slices * 8192KB). Since 1024KB is exactly 1MB, it is equivalent to say 1MB + (slices * 8MB).

WARNING: In this document KB means 1024 bytes and MB means 1048576 bytes (frequently expressed as KiB and MiB in modern terminology). In general, hard disk capacities use KB to mean 1000 bytes and MB to mean 1,000,000 bytes.

As an example, hardware distributors frequently supply a "64MB" CF Card with a RomWBW system. Such a hard disk probably has less than 62.5MB of actual space (using the RomWBW definition that 1MB is 1048576 bytes). Such a drive will not support 8 slices. It will support 7 slices just fine because 7 * 8,320KB = 58.24MB (hd512) or 1024KB + (7 * 8192MB) = 57MB (hd1k).

The cost of high capacity CF/SD/USB Media has become very reasonable. I highly recommend upgrading to 1GB or greater media. This size will support all features of the RomWBW Combo Disk Image with 64 slices and a 384MB FAT filesystem (see Combo Hard Disk Image).

Chapter 5

Disk Content Preparation

With some understanding of how RomWBW presents disk space to the operating systems, we need to go over the options for actually setting up your disk(s) with content.

Since it would be quite a bit of work to transfer over all the files you might want initially to your disk(s), RomWBW provides a much easier way to get initial contents on your disks. You can use your modern Windows, Linux, or Mac computer to copy a disk image onto the disk media, then just move the media over to your RomWBW computer. RomWBW comes with a variety of disk images that are ready to use and have a much more complete set of files than you will find on the ROM disk. This process is covered below under Disk Images.

If you do not want to start with pre-built disk images, you can alternatively initialize the media in-place using your RomWBW system. Essentially, this means you are creating a set of blank directories on your disk so that files can be saved there. This process is described below under Disk Initialization. In this scenario, you will need to subsequently copy any files you want to use onto the newly initialized disk (see Transferring Files).

You will notice that in the following instructions there is no mention of specific hardware. Because the RomWBW firmware provides a hardware abstraction layer, all disk images will work on all hardware variations. Yes, this means you can remove disk media from one RomWBW system and put it in a different RomWBW system. The only constraint is that the applications on the disk media must be up to date with the firmware on the system being used.

5.1 Disk Images

As mentioned previously, RomWBW includes a variety of disk images that contain a full set of applications for the operating systems supported. It is generally easier to use these disk images than transferring your files over individually. You use your modern computer (Windows, Linux, MacOS) to write the disk image onto the disk media, then just move the media over to your system.

The disk image files are found in the Binary directory of the distribution. Floppy disk images are prefixed with "fd_" and hard disk images are prefixed with either "hd512_" or "hd1k_" depending on the hard disk layout they are for.

Each disk image has the complete set of normal applications and tools distributed with the associated operating system or application suite. The following table shows the disk images available.

Disk Image	Description	Boot
xxx_cpm22.img	DRI CP/M 2.2 Operating System	Yes
xxx_zsdos.img	ZCPR-DJ & ZSDOS 1.1 Operating System	Yes
xxx_nzcom.img	NZCOM ZCPR 3.4 Operating System	Yes
xxx_cpm3.img	DRI CP/M 3 Operating System	Yes
xxx_zpm3.img	ZPM3 Operating System	Yes
xxx_qpm.img	QPM Operating System	Yes
xxx_dos65.img	DOS/65 Operating System	Yes
xxx_ws4.img	WordStar v4 & ZDE Applications	No

You will find 3 sets of these .img files in the distribution. The "xxx" portion of the filename will be "fd_" for a floppy image, "hd512" for a legacy layout hard disk image, and "hd1K" for a modern layout hard disk image.

In the case of xxx_dos65.img, only an hd512 variant is provided. This is a constraint of the DOS65 distribution.

There is also an image file called "psys.img" which contains a bootable p-System hard disk image. It contains 6 p-System filesystem slices, but these are not interoperable with the CP/M slices described above. This file is discussed separately under p-System in the Operating Systems section.

5.1.1 Floppy Disk Images

The floppy disk images are all intended to be used with 3.5" high-density, double-sided 1.44 MB floppy disk media. This is ideal for the default floppy drive support included in RomWBW standard ROMs.

For floppy disks, the .img file is written directly to the floppy media as is. The floppy .img files are 1.44 MB which is the exact size of a single 3.5" high density floppy disk. You will need a floppy disk drive of the same type connected to your modern computer to write this image. Although modern computers do not come equipped with a floppy drive, you can still find USB floppy drives that work well for this.

The floppy disk must be physically formatted **before** writing the image onto it. You can do this with RomWBW using FDU as described in the Floppy Disks section of this document. You can also format the floppy using your modern computer, but using FDU on RomWBW is preferable because it will allow you to use optimal physical sector interleaving. FDU defaults to a sector interleave of 2 which will result in faster floppy disk I/O. Other interleaves will work, but will be slower.

RomWBW includes a Windows application called RawWriteWin in the Tools directory of the distribution. This simple application will let you choose a file and write it to an attached floppy drive. For Linux/MacOS, I think you can use the dd command (but I have not actually tried this). It is probably obvious, but writing an image to a floppy disk will overwrite and destroy all previous contents.

Once the image has been written to the floppy disk, you can insert the floppy disk in your RomWBW floppy disk and read/write files on it according to the specific operating system instructions. If the image is bootable, then you will be able to boot from it by entering the floppy drive's corresponding unit number at the RomWBW Boot Loader command prompt.

5.1.2 Hard Disk Images

Keeping in mind that a RomWBW hard disk (including CF/SD/USB devices) allows you to have multiple slices (CP/M filesystems), there are a couple ways to image hard disk media. The easiest approach is to use the "combo" disk image. This image is already prepared with 6 slices containing 5 ready-to-run OSes and a slice with the WordStar application files. Alternatively, you can create your own hard disk image with the specific slice contents you choose.

Standard Hard Disk Physical Layout

As previously described in Hard Disk Layouts, the exact placement of slices and optional FAT partition will vary depending on which disk layout (hd512 or hd1k) you are using and your partition table entries. To simplify the use of hard disk images, RomWBW has adopted standard partition table entries for disk image files provided.

These partition sizes and locations were chosen to:

- Fit entirely on 1GB media
- Allow for 64 CP/M filesystem slices
- Allow for a 384KB FAT filesystem

NOTE: RomWBW is not limited to these partition table entries. You can change the size and location of the RomWBW and/or FAT partitions to increase/decrease the number of slices or FAT filesystem size.

	— Legacy (hd512) —		— Modern (hd1k) —	
	Byte(s)	Sector(s)	Byte(s)	Sector(s)
RomWBW (slices) Start	0	0	1,048,576	2,048
RomWBW (slices) Size	545,259,520	1,064,960	536,870,912	1,048,576
FAT Filesystem Start	545,259,520	1,064,960	537,919,488	1,050,624
FAT Filesystem Size	402,653,184	786,432	402,653,184	786,432
<end>	947,912,704	1,851,392	940,572,672	1,837,056

The above partition table entries will result in the following locations and sizes of filesystems on the RomWBW disk images.

	— Legacy (hd512) —		— Modern (hd1k) —	
	Byte(s)	Sector(s)	Byte(s)	Sector(s)
Prefix Start	–	–	0	0
Prefix Size	–	–	1,048,576	2,048
Slice Size	8,519,680	16,640	8,388,608	16,384
Slice 0 Start	0	0	1,048,576	2,048
Slice 1 Start	8,519,680	16,640	9,437,184	18,432
Slice 2 Start	17,039,360	33,280	17,825,792	34,816
Slice 3 Start	25,559,040	49,920	26,214,400	51,200
Slice 4 Start	34,078,720	66,560	34,603,008	67,584
Slice 5 Start	42,598,400	83,200	42,991,616	83,968
Slice 6 Start	51,118,080	99,840	51,380,224	100,352
Slice 7 Start	59,637,760	116,480	59,768,832	116,736
Slice 63 Start	536,739,840	1,048,320	529,530,880	1,034,240
FAT Filesystem Start	545,259,520	1,064,960	537,919,488	1,050,624
FAT Filesystem Size	402,653,184	786,432	402,653,184	786,432
<end>	947,912,704	1,851,392	940,572,672	1,837,056

Combo Hard Disk Image

The combo disk image is essentially just a single image that has several of the individual filesystem images (slices) already concatenated together. The combo disk image contains the following 6 slices in the positions indicated:

Slice	Description
Slice 0	DRI CP/M 2.2 Operating System
Slice 1	ZCPR-DJ & ZSDOS 1.1 Operating System
Slice 2	NZCOM ZCPR 3.4 Operating System
Slice 3	DRI CP/M 3 Operating System
Slice 4	ZPM3 Operating System
Slice 5	WordStar v4 & ZDE Applications

You will notice that there are actually 2 combo disk images in the distribution. One for an hd512 disk layout (hd512_combo.img) and one for an hd1k disk layout (hd1k_combo.img). Simply use the image file that corresponds to your desired hard disk layout. Review the information in Hard Disk Layouts if you need more information of the disk layout options.

Although the combo disk images contain only 6 slices of content, they reserve space to store 64 CP/M filesystem slices as well as a single 384MB FAT filesystem. Keep in mind that the slices beyond the first 6 are not yet initialized. You will need to use the `CLRDIR` application to initialize them before their first use. Likewise, the pre-allocated FAT partition must still be formatted using `FAT FORMAT` in order to actually use it (see FAT Filesystem Preparation). Alternatively, the FAT partition can be formatted on a modern computer.

The combo disk image layout was designed to fit well on a 1GB hard disk. The 64 CP/M slices (approximately 512MB) and 384MB FAT filesystem all fit well within a 1GB hard disk. This size choice was a bit arbitrary, but based on the idea that 1GB CF/SD/USB Media is easy and cheap to acquire. It is fine if your hard disk is smaller than 1GB. It just means that it will not be possible to use the pre-allocated FAT filesystem partition and any CP/M filesystem slices that don't fit. You will get "no disk" errors if you attempt to access a slice past the end of the physical hard disk.

WARNING: Your hard disk may be too small to contain the full 64 CP/M filesystem slices. The true number of CP/M filesystem slices that will fit on your specific physical hard disk can be calculated as described in Hard Disk Capacity.

For RomWBW systems with a single hard disk (typical), you will notice that the OS will pre-allocate 8 drive letters to the hard disk. If the combo disk image is being used, only the first 6 drive letters (typically C: - H:) will have any content because the combo disk image only provides 6 slices. The subsequent drives (typically I: - J:) will have no content and will not be pre-initialized. If you want to use any slices beyond the first 6 on the hard disk, then you must initialize them using `CLRDIR` first.

A great way to maintain your own data on a hard disk is to put this data in slices beyond the first 6. By doing so, you can always "re-image" your drive with the combo image without overlaying the data stored in the slices beyond the first 6. Just be very careful to use the same combo image layout (hd512 or hd1k) as you used originally. Also remember to calculate the maximum number of slices your hard disk will support and do not exceed this number.

WARNING: The combo disk image includes a partition table at the start of the image. If you re-image drive with the combo image, you will overwrite this partition table. This is fine as long as you don't make any changes to the partition table. If you manually customize the partition table (using `FDISK80` or other partition management software), those changes will be lost if you re-image your disk with a new combo disk image.

Custom Hard Disk Image

If you want to use specific slices in a specific order, you can easily generate a custom hard disk image file.

For hard disks, each .img file represents a single slice (CP/M filesystem). Since a hard disk can contain many slices, you can just concatenate the slices (.img files) together to create your desired hard disk image. For example, if you want to create a hard disk image that has slices for CP/M 2.2, CP/M 3, and WordStar in the hd512 format, you would use the command line of your modern computer to create the final image:

Windows:

```
COPY /B hd512_cpm22.img + hd512_cpm3.img + hd512_ws hd.img
```

Linux/MacOS:

```
cat hd512_cpm22.img hd512_cpm3.img hd512_ws >hd.img
```

NOTE: For the hd1k disk layout, you **must** prepend the prefix file called hd1k_prefix.dat which contains the required partition table. So, for an hd1k layout you would use the following:

Windows:

```
COPY /B hd1k_prefix.dat + hd1k_cpm22.img + hd1k_cpm3.img + hd1k_ws hd.img
```

Linux/MacOS:

```
cat hd1k_prefix.dat hd1k_cpm22.img hd1k_cpm3.img hd1k_ws >hd.img
```

In all of the examples above, the resulting file (hd.img) would now be written to your hard disk media and would be ready to use in a RomWBW system.

If you wish to further customize or create new disk image definitions, please refer to the ReadMe.txt file in the Source/Images directory.

Writing Hard Disk Images

Once you have chosen a combo hard disk image file or prepared your own custom hard disk image file, it will need to be written to the media using your modern computer. Note that you **do not** run `CLRDIR` or `SYSCOPY` on the slices that contain the data. When using this method, the disk will be partitioned and setup with 1 or more slices containing ready-to-run bootable operating systems.

To write a hard disk image file onto your actual media (actual hard disk or CF/SD/USB Media), you need to use an image writing utility on your modern computer. Your modern computer will need to have an appropriate interface or slot that accepts the media. To actually copy the image, you can use the `dd` command on Linux or MacOS. On Windows, in the "Tools" directory of the distribution, there is an application called Win32DiskImager. In all cases, the image file should be written to the media starting at the very first block or sector of the media.

You are not limited to the number of slices that are contained in the image that you write to your hard disk media. You can use additional slices as long your media has room for them. However, writing the disk image will not initialize the additional slices. If these additional slices were previously initialized, they will not be corrupted when you write the new image and will still contain their previous contents. If the additional slices were not previously initialized, you can use `CLRDIR` to do so and optionally `SYSCOPY` if you want them to be bootable.

To be entirely clear, writing a disk image file to your hard disk media will overwrite an pre-existing partition table and the number of slices that your image file contains. It will not overwrite or corrupt slices beyond those in the image file. As a result, you can use additional slices as a place to maintain your personal data because these slices will survive re-imaging of the media. If you setup a FAT partition on your media, it will also survive the imaging process.

WARNING: In order for your additional slices and/or FAT partition to survive re-imaging, you **must** follow these rules:

- Do not modify the partition table of the media using FDISK80 or any other partition management tools.
- Ensure that your hard disk image file uses the same disk layout approach (hd512 or hd1k) as previously used on the media.

Once you have copied the image onto the hard disk media, you can move the media over to your RomWBW system. You can then boot to the operating system slices by specifying "." at the RomWBW Boot Loader command prompt.

5.2 In-situ Disk Preparation

If you do not wish to use the pre-built disk images, it is entirely possible to setup your disks manually and transfer contents to them.

In this scenario, you will initialize the disk media entirely from your RomWBW system. So, you need to start by inserting the disk media, booting RomWBW, and confirming that the media is being recognized. If RomWBW recognizes the media, it will indicate this in the boot messages even though the media has not yet been prepared for use.

The following instructions are one way to proceed. This does not mean to imply it is the only possible way. Also, note that RAM/ROM disk media is prepared automatically. ROM disks are part of the ROM image and RAM disks are initialized when an operating system is started.

Start by booting RomWBW and launching either CP/M 2.2 or Z-System from ROM using the Boot Loader 'C' or 'Z' commands respectively. You can now use the tools on the ROM disk to prepare your disks. Note that you will see the operating system assign disks/slices to drives even though the disks/slices are not yet initialized. This is normal and does not mean the disks/slices are ready to use.

Preparation of floppy disk media is very simple. The floppy disk must be physically formatted as discussed in Floppy Disks previously using `FDU`. If a floppy is already physically formatted, you can wipe out it's contents (make it empty again) by running `CLRDIR` on it. You can confirm a floppy disk is ready for content by simply running a `DIR` command on it. The `DIR` command should complete without error and should list no files. At this point, you can proceed to copy files to the floppy disk and (optionally) make the floppy bootable using `SYSCOPY`.

The rest of this section will cover preparation of hard disk media. To start, it is critical that you decide which disk layout approach to use (either hd512 or hd1k). Review the Hard Disk Layouts section if you are not sure.

Partition Setup

WARNING: Modifying the partition table of existing media will make the data on the media inaccessible.

Since the disk layout is determined by the existence (or lack) of a RomWBW partition, you must start by running `FDISK80`. When FDISK80 starts, enter the disk unit number of the new media. At this point, use the 'I' command to initialize (reset) the partition table to an empty state. If you are going to use the hd512 layout, then use 'W' to write the empty table to the disk and exit. Remember that the lack of a partition for RomWBW implies the legacy (hd512) layout.

If you are going to use an hd1k layout, then you must create a partition for the RomWBW CP/M slices. The partition can be placed anywhere you want and can be any size >= 8MB. Keeping the size of the partition to increments of 8MB makes sense. The partition type **must** be set to '2e'. The typical location for the RomWBW partition is at 1MB with a size of 512MB (64 slices). Below is an example of creating a RomWBW partition following these guidelines.

```
FDISK80 for RomWBW, UNA, Mini-M68k, KISS-68030, SBC-188  ----
       Version 1.1-22 created 7-May-2020
                 (Running under RomWBW HBIOS)

HBIOS unit number [0..11]: 4
Capacity of disk 4:  (  4G)  7813120      Geom 77381010
Nr  ---Type- A --      Start         End   LBA start  LBA count  Size
 1             00       *** empty ***
 2             00       *** empty ***
 3             00       *** empty ***
 4             00       *** empty ***
>>i
>>n
New partition number: 1
Starting Cylinder (default 0): 1Mb
Ending Cylinder (or Size= "+nnn"): +512Mb
>>t
Change type of partition number: 1
New type (in hex), "L" lists types: 2e
>>p
Nr  ---Type- A --      Start         End   LBA start  LBA count  Size
 1    RomWBW   2e      8:0:1  1023:15:16        2048    1048576  512M
 2             00       *** empty ***
 3             00       *** empty ***
 4             00       *** empty ***
>>w
Do you really want to write to disk? [N/y]: y
Okay
FDISK exit.
```

At this point, it is best to restart your system to make sure that the operating system is aware of the partition table updates. Start CP/M 2.2 or Z-System from ROM again.

You are now ready to initialize the individual slices of your hard disk media. On RomWBW, slice initialization is done using the CLRDIR application. Since the CLRDIR application works on OS drive letters, you must pay attention to how the OS drive letters are mapped to your disk devices which is listed when the OS starts. Let's assume that C: has been assigned to slice 0 of the disk you are initializing. You would use `CLRDIR C:` to initialize C: and prepare it hold files. Note that CLRDIR will prompt you for confirmation and you must respond with a **capital** 'Y' to confirm.

After CLRDIR completes, the slice should be ready to use by the operating system via the drive letter assigned. Start by using the `DIR` command on the drive (`DIR C:`). This should return without error, but list no files. Next, use the `STAT` command to confirm that the disk is using the layout you intended. For example, use `STAT C:DSK:` and look at the number of "32 Byte Directory Entries". It should say 512 for a legacy (hd512) disk layout and 1024 for a modern (hd1024) disk layout.

Assuming you want to use additional slices, you should initialize them using the same process. You may need to reassign OS drive letters to access some slices that are beyond the ones automatically assigned. You can use the `ASSIGN` command to handle this.

Once you have your slice(s) initialized, you can begin transferring files to the associated drive letters. Refer to the Transferring Files chapter for options to do this. If you want to make a slice bootable, you will need to use `SYSCOPY` to setup the system track(s) of the slice. The use of `SYSCOPY` depends on the operating system and is described in the Operating Systems chapter of this document.

As an example, let's assume you want to setup C: as a bootable Z-System disk and add to it all the files from the ROM disk. To setup the system track you would use:

```
B>SYSCOPY C:=B:ZSYS.SYS

SYSCOPY v2.0 for RomWBW CP/M, 17-Feb-2020 (CP/M 2 Mode)
Copyright 2020, Wayne Warthen, GNU GPL v3

Transfer system image from B:ZSYS.SYS to C: (Y/N)? Y
Reading image... Writing image... Done
```

Then, to copy all of the files from the ROM disk to C:, you could use the `COPY` command as shown below. In this example, the list of files being copied has been truncated.

```
B>copy *.* m:
COPY  Version 1.73 (for ZSDOS)   2 Jul 2001
Copying B0:????????.??? to M0:
 -> ASM     .COM..Ok  Verify..Ok
 -> ASSIGN  .COM..Ok  Verify..Ok
 -> CLRDIR  .COM..Ok  Verify..Ok
 -> COMPARE .COM..Ok  Verify..Ok
```

```
-> COPY    .COM..Ok  Verify..Ok
-> CPM     .SYS..Ok  Verify..Ok
0 Errors
```

Once this process succeeds, you will be able to boot directly to the disk slice from the boot loader prompt. See the instructions in Starting Operating Systems from Disk for details on this.

Chapter 6

Operating Systems

One of the primary goals of RomWBW is to expose a set of generic hardware functions that make it easy to adapt operating systems to any hardware supported by RomWBW. As a result, there are now 8 operating systems that have been adapted to run under RomWBW. The adaptations are identical for all hardware supported by RomWBW because RomWBW hides all hardware specifics from the operating system.

By design, the operating systems provided with RomWBW are original and unmodified from their original distribution. Patches published by the authors are generally included or applied. The various enhancements RomWBW provides (such as hard disk slices) are implemented entirely within the system adaptation component of each operating system (e.g., CP/M CBIOS). As a result, each operating system should function exactly as documented by the authors and retain maximum compatibility with original applications.

Note that all of the operating systems included with RomWBW support the same basic filesystem format from DRI CP/M 2.2 (except for p-System). As a result, a formatted filesystem will be accessible to any operating system. The only possible issue is that if you turn on date/time stamping using the newer OSes, the older OSes will not understand this. Files will not be corrupted, but the date/time stamps will not be maintained.

The following sections briefly describe the operating system options currently available and brief operating notes.

6.1 Digital Research CP/M 2.2

This is the most widely used variant of the Digital Research operating systems. It has the most basic feature set, but is essentially the compatibility metric for all other CP/M-like operating systems including those listed below.

If you are new to the CP/M world, I would recommend using this CP/M variant to start with simply because it is the most stable and you are less likely to encounter compatibility issues.

Documentation

- CPM Manual

Boot Disk

To make a bootable CP/M disk, use the RomWBW `SYSCOPY` tool to place a copy of the operating system on the boot track of the disk. The RomWBW ROM disk has a copy of the boot track call "CPM.SYS". For example:

```
SYSCOPY C:=B:CPM.SYS
```

Notes

- You can change media, but it must be done while at the OS command prompt and you **must** warm start CP/M by pressing ctrl-C. This is a CP/M 2.2 constraint and is well documented in the DRI manual.
- `SUBMIT.COM` has been patched per DRI to always place submit files on A:. This ensures the submitted file will always be properly executed.
- The original versions of DDT, DDTZ, and ZSID used the RST 38 vector which conflicts with interrupt mode 1 use of this vector. The DDT, DDTZ, and ZSID applications in RomWBW have been modified to use RST 30 to avoid this issue.
- Z-System applications will not run under CP/M 2.2. For example, the `LDDS` date stamper will not work.

6.2 Z-System

Z-System is the most popular non-DRI CP/M workalike "clone" which is generally referred to as Z-System. Z-System is intended to be an enhanced version of CP/M and should run all CP/M 2.2 applications. It is optimized for the Z80 CPU (as opposed to 8080 for CP/M) and has some significant improvements such as date/time stamping of files.

Z-System is a somewhat ambiguous term because there are multiple generations of this software. RomWBW Z-System is a combination of both ZCPR-DJ (the CCP) and ZSDOS 1.1 (the BDOS) when referring to Z-System.

The latest version of Z-System (ZCPR 3.4) is also provided with RomWBW via the NZ-COM adaptation (see below).

Documentation

- ZCPR Manual
- ZCPR-DJ
- ZSDOS Manual

Boot Disk

To make a bootable Z-System disk, use the RomWBW `SYSCOPY` tool to place a copy of the operating system on the boot track of the disk. The RomWBW ROM disk has a copy of the boot track call "ZSYS.SYS". For example:

```
SYSCOPY C:=B:ZSYS.SYS
```

Notes

- Although most CP/M 2.2 applications will run under Z-System, some may not work as expected. The best example is PIP which is not aware of the ZSDOS paths and will fail in some scenarios (use `COPY` instead).
- Although ZSDOS can recognize a media change in some cases, it will not always work. You should only change media at a command prompt and be sure to warm start the OS with a ctrl-C.
- ZSDOS has a concept of fast relog of drives. This means that after a warm start, it avoids the overhead of relogging all the disk drives. There are times when this causes issues. After using tools like CLRDIR or MAP, you may need to run "RELOG" to get the drive properly recognized by ZSDOS.
- RomWBW fully supports both DateStamper and P2DOS file date/time stamping. You must load the desired stamping module (`LDDS` for DateStamper or `LDP2D` for P2DOS). This could be automated using a `PROFILE.SUB` file. Follow the ZSDOS documentation to initialize a disk for stamping.
- ZSVSTAMP expects to be running under the ZCPR 3.X command processor. By default, RomWBW uses ZCPR 1.0 (intentionally, to reduce space usage) and ZSVSTAMP will just abort in this case. It will work fine if you implement NZCOM. ZSVSTAMP is included solely to facilitate usage if/when you install NZCOM.
- FILEDATE only works with DateStamper style date stamping. If you run it on a drive that is not initialized for DateStamper, it will complain `FILEDATE, !!!TIME&.DAT missing`.

This is normal and just means that you have not initialized that drive for DateStamper (using PUTDS).

- ZXD will handle either DateStamper or P2DOS type date stamping. However, it **must** be configured appropriately. As distributed, it will look for P2DOS date stamps. Use ZCNFG to reconfigure it for P2DOS if that is what you are using.
- Many of the tools can be configured (using either ZCNFG or DSCONFIG). The configuration process modifies the actual application file itself. This will fail if you try to modify one that is on the ROM disk because it will not be able to update the image.
- DATSWEEP can be configured using DSCONFIG. However, DSCONFIG itself needs to be configured first for proper terminal emulation by using SETTERM. So, run SETTERM on DSCONFIG before using DSCONFIG to configure DATSWEEP!
- After using PUTDS to initialize a directory for ZDS date stamping, I am finding that it is necessary to run RELOG before the stamping routines will actually start working.
- Generic CP/M PIP and ZSDOS path searching do not mix well if you use PIP to copy to or from a directory in the ZSDOS search path. Best to use COPY from the ZSDOS distribution.

6.3 NZCOM Automatic Z-System

NZCOM is a much further refined version of Z-System (ZCPR 3.4). NZCOM was sold as an enhancement for existing users of CP/M 2.2 or ZSDOS. For this reason, (by design) NZCOM does not provide a way to boot directly from disk. Rather, it is loaded after the system boots into a host OS. On the RomWBW NZCOM disk images, the boot OS is ZSDOS 1.1. A `PROFILE.SUB` file is included which automatically launches NZCOM as soon as ZSDOS loads.

NZCOM is highly configurable. The RomWBW distribution has been configured in the most basic way possible. You should refer to the documentation and use `MKZCM` as desired to customize your system.

NZCOM has substantially more functionality than CP/M or basic Z-System. It is important to read the "NZCOM Users Manual.pdf" document in order to use this operating system effectively.

Documentation

- NZCOM Users Manual

Boot Disk

Since NZ-COM boots via Z-System, you can make a bootable NZ-COM disk using `ZSYS.SYS` as described in Z-System above. You will need to add a `PROFILE.SUB` file to auto-start NZ-COM itself.

Notes

- All of the notes for Z-System above generally apply to NZCOM.
- There is no `DIR` command, you must use `SDZ` instead. If you don't like this, look into the `ALIAS` facility.

6.4 Digital Research CP/M 3

This is the Digital Research follow-up product to their very popular CP/M 2.2 operating system. While highly compatible with CP/M 2.2, it features many enhancements and is not 100% compatible. It makes direct use of banked memory to increase the user program space (TPA). It also has a new suite of support tools and help system.

Documentation

- CPM3 Users Guide
- CPM3 Command Summary
- CPM3 Programmers Guide
- CPM3 System Guide

Boot Disk

To create (or update) a CP/M 3 boot drive, you must place `CPMLDR.SYS` on the system track of the disk. You must also place `CPM3.SYS` and `CCP.COM` on the target drive as regular files. Do **not** place CPM3.SYS on the boot track. `CPMLDR.SYS` chain loads `CPM3.SYS` which must exist as a regular file on the disk. Subsequently, `CPM3.SYS` loads `CCP.COM`.

The CP/M 3 boot files are not included on the ROM disk due to space constraints. You will need to transfer the following files to your system from the RomWBW distribution directory Binary/CPM3. You can use XModem for this (or any of the mechanisms in Transferring Files.

- `CPMLDR.SYS`
- `CPM3.SYS` or `CPM3BNK.SYS`
- `CCP.COM`

The CPM3.SYS boot file is provided in 2 versions. In the Binary/CPM3 distribution directory, CPM3.SYS is the "non-banked" version of CP/M 3. The CPM3BNK.SYS file is the "banked" version of CP/M 3. You almost certainly want to transfer the banked CPM3BNK.SYS version.

After transferring the boot files to your RomWBW system, you will need to use SYSCOPY to place CPMLDR.SYS on the boot track of the target drive. CPM3.SYS and CCP.COM can be copied to the target drive using any standard file copy tool such as PIP or COPY.

You do not need to be booted into CP/M 3 to create or update a CP/M 3 disk. The recommended approach is to boot CP/M 2.2 or Z-System from ROM. Transfer the boot files to the RAM disk. Then simply copy the files onto the CP/M 3 disk. Assuming the target CP/M 3 disk is F:, you can use the following commands to place the files on the target drive:

```
SYSCOPY F:=A:CPMLDR.SYS
COPY A:CPM3BNK.SYS F:CPM3.SYS
COPY A:CCP.COM F:
```

Note in the example above that CPM3BNK.SYS is renamed to CPM3.SYS in the copy command.

Notes

- The COPYSYS command described in the DRI CP/M 3 documentation is not provided with RomWBW. The RomWBW SYSCOPY command is used instead.
- Although CP/M 3 is generally able to run CP/M 2.2 programs, this is not universally true. This is especially true of the utility programs included with the operating system. For example, the SUBMIT program of CP/M 3 is completely different/incompatible from the SUBMIT program of CP/M 2.2.
- RomWBW fully supports CP/M 3 file date/time stamping, but this requires that the disk be properly initialized for it. This process has not been performed on the CP/M 3 disk image. Follow the CP/M 3 documentation to complete this process, if desired.

6.5 ZPM3

Simeon Cran's ZPM3 is an interesting combination of the features of both CP/M 3 and ZCPR3. Essentially, it has the features of and compatibility with both.

Due to this dual compatibility, the ZPM3 distribution image contains most of the standard CP/M 3 files as well as a variety of common ZCPR3 applications. However, you will notice that user area 0 of the disk has only a few files. Most of the files are distributed among other user areas which is standard practice for ZCPR3. Most importantly, you will see most of the applications in user area 15. The applications can be executed from any user area because ZPM3 has a default search path that includes User 15.

The ZPM3 distribution comes with essentially no utility programs at all. In addition to the standard CP/M 3 utilities, RomWBW includes a variety of common ZCPR3 utilities.

Documentation

ZPM3 has no real documentation. You are expected to understand both CP/M 3 and ZCPR 3.

Boot Disk

To create (or update) a ZPM3 boot drive, you must place `ZPMLDR.SYS` on the system track of the disk. You must also place `CPM3.SYS`, `ZCCP.COM`, `ZINSTAL.ZPM`, and `STARTZPM.COM` on the target drive as regular files. Do **not** place CPM3.SYS on the boot track. `ZPMLDR.SYS` chain loads `CPM3.SYS` which must exist as a regular file on the disk. Subsequently, `CPM3.SYS` loads `CCP.COM`.

The CP/M 3 boot files are not included on the ROM disk due to space constraints. You will need to transfer the following files to your system from the RomWBW distribution directory Binary/ZPM3. You can use XModem for this (or any of the mechanisms in Transferring Files.

- `ZPMLDR.SYS`
- `CPM3.SYS`
- `ZCCP.COM`
- `ZINSTAL.ZPM`
- `STARTZPM.COM`

You may be surprised to see the file called `CPM3.SYS`. This is not a typo. Although it is called `CPM3.SYS`, it is ZPM and not the same as `CPM3.SYS` in the CPM3 directory. Also, unlike CP/M 3, ZPM3 is always banked, so you will not find two versions of the file. `CPM3.SYS` is a banked implementation of ZPM3.

After transferring the boot files to your RomWBW system, you will need to use `SYSCOPY` to place `ZPMLDR.SYS` on the boot track of the target drive. The remaining boot files can be copied to the target drive using any standard file copy tool such as `PIP` or `COPY`.

You do not need to be booted into ZPM3 to create or update a ZPM3 disk. The recommended approach is to boot CP/M 2.2 or Z-System from ROM. Transfer the boot files to the RAM disk. Then simply copy the files onto the ZPM disk. Assuming the target ZPM3 disk is F:, you can use the following commands to place the files on the target drive:

```
SYSCOPY F:=A:ZPMLDR.SYS
COPY A:CPM3.SYS F:CPM3.SYS
```

```
COPY A:CCP.COM F:
COPY A:ZINSTAL.ZPM F:
COPY A:STARTZPM.COM F:
```

Notes

- The ZPM3 operating system is contained in the file called CPM3.SYS which is confusing, but this is as intended by the ZPM3 distribution. I believe it was done this way to make it easier for users to transition from CP/M 3 to ZPM3.

6.6 QP/M

QP/M is another OS providing compatibility with and enhancements to CP/M 2.2. It is provided as a bootable disk image for RomWBW.

Refer to the ReadMe.txt file in Source/Images/d_qpm for more details regarding the RomWBW adaptation and customizations.

Documentation

- QP/M 2.7 Installation Guide and Supplements
- QP/M 2.7 Interface Guide
- QP/M 2.7 Features and Facilities

Boot Disk

To create or update a bootable QP/M Z-System disk, a special process is required. QP/M is not provided in source format. You are expected to install QP/M over an existing CP/M installation using the `QINSTALL.COM` application.

To update an existing QP/M boot disk with the latest RomWBW CBIOS, you must use 2 steps: apply the generic CP/M system track, then reinstall the QP/M components. To do this, you can perform the following steps:

1. Boot to the existing QP/M disk. At this point, drive A should be the QP/M disk that you wish to update. You may receive a warning about CBIOS/HBIOS version mismatch.

2. Use RomWBW `SYSCOPY` to place the stock RomWBW CP/M OS image onto the system tracks of the QP/M boot disk:

   ```
   SYSCOPY A:=x:CPM.SYS
   ```

   where x is the drive letter of your ROM Disk.

3. Run `QINSTALL` to overlay the QP/M OS components on your QP/M boot disk.

WARNING: `QINSTALL` has no mechanism for retaining previous non-default settings. Any previous non-default settings you previously made with `QINSTALL` will need to be reapplied. The pre-built RomWBW QP/M disk image includes a couple of specific non-default settings to optimize use with RomWBW. Please review the notes in the ReadMe.txt file in Source/Images/d_qpm.

Notes

- QPM is not available as source. This implementation was based on the QPM binary distribution and has been minimally customized for RomWBW.
- When booted, the QPM startup banner will indicate CP/M 2.2. This is because QPM uses the CP/M 2.2 CBIOS code.
- QINSTALL is used to customize QPM. It is included on the disk image. You should review the notes in the ReadMe.txt file in Source/Images/d_qpm before making changes.
- In addition to the QPM disk image, all of the QPM distribution files can be found in the RomWBW distribution in the Source/Images/d_qpm/u0 directory.
- The QPM disk image is not included as one of the slices on the RomWBW combo disk image. If you want to include QPM, you can do so by following the directions in Source/Images/Readme.txt.

6.7 UCSD p-System

This is a full implementation of the UCSD p-System IV.0 for Z80 running under RomWBW. Unlike the OSes above, p-System uses its own unique filesystem and is not interoperable with other OSes.

It was derived from the p-System Adaptable Z80 System. Unlike some other distributions, this implements a native p-System Z80 Extended BIOS, it does not rely on a CP/M BIOS layer.

The p-System is provided on a hard disk image file called psys.img. This must be copied to its own dedicated hard disk media (CF Card, SD Card, etc.). It is booted by selecting slice 0 of the corresponding hard disk unit at the RomWBW Boot Loader prompt. Do not attempt to use CP/M slices on the same disk.

Refer to the ReadMe.txt file in Source/pSys for more details.

Documentation

- UCSD p-System Users Manual

Boot Disk

There is no mechanism provided to create a p-System boot disk from scratch under RomWBW. This has already been done as part of the porting process. You must use the provided p-System hard disk image file which is bootable.

Notes

- There is no floppy support at this time.
- The hard disk image contains 6 p-System slices which are assigned to p-System unit numbers 4, 5, 9, 10, 11, and 12 which is standard for p-System. Slices 0-5 are assigned sequentially to these p-System unit numbers and it is not possible to reassign them. Unit #4 (slice 0) is bootable and contains all of the p-System distribution files. Unit #5 (slice 1) is just a blank p-System filesystem. The other units (9-12) have not been initialized, but this can be done from Filer using the Zero command.
- p-System relies heavily on the use of a full screen terminal. This implementation has been setup to expect an ANSI or DEC VT-100 terminal or emulator. The screen output will be garbled if no such terminal or emulator is used for console output.
- There is no built-in mechanism to move files in and out of p-System. However, the .vol files in Source/pSys can be read and modified by CiderPress. CiderPress is able to add and remove individual files.

  Andrew Davidson has created a Python script that can extract p-System volumes from an existing disk image file. The script is also capable of inserting a modified volume back into the disk image file. This tool is available at https://github.com/robosnacks/psysimg.

6.8 FreeRTOS

Phillip Stevens has ported FreeRTOS to run under RomWBW. FreeRTOS is not provided in the RomWBW distribution. FreeRTOS is available under the MIT licence and further general information is available at FreeRTOS.

Phillip may be contacted via his GitHub Page.

6.9 Fuzix

Fuzix is a Unix-ish operating system for small systems. It is the work of Alan Cox and is hosted on GitHub at https://github.com/EtchedPixels/FUZIX. Fuzix itself is a standalone operating system, but it frequently utilizes RomWBW to boot and launch on RomWBW-supported platforms.

For those Fuzix platforms that leverage RomWBW for startup, you will program your ROM with the normal RomWBW ROM – there is no Fuzix-specific ROM. A Fuzix disk image for your system is then written to your disk media. After booting your system via the normal RomWBW ROM, you start Fuzix simply by choosing the disk device containing the Fuzix image at the RomWBW Loader prompt.

To create a Fuzix disk image:

- Locate and download the Fuzix disk image for your system from https://www.fuzix.org/. For each platform, you will typically find two image files. An emulator image (`emu-xxx.img`) and a disk image (`disk.img`). You want the disk image file.
- Write the disk image file to your physical media (CF Card, SD Card, etc.) starting at the beginning of the media (first sector). Do not combine the Fuzix image with the RomWBW disk images – they are entirely separate.

To boot into Fuzix:

- Insert your Fuzix disk media.
- Power-up or reset your system. RomWBW should load normally and bring you to the RomWBW Boot Loader prompt.
- Depending on the platform, Fuzix may be built to run at a different baud rate that the default RomWBW baud rate. If so, it is best to change your RomWBW baud rate prior to initiating the Fuzix startup. You can do this at the loader prompt with a command like this:

  ```
  I 0 38400
  ```

  Replace 38400 with the desired baud rate for Fuzix. You will be prompted to change your terminal's baud rate at this time.
- At the RomWBW Boot Loader prompt, enter the disk unit number of the Fuzix media. Fuzix should load and you will see device discovery/information messages that vary depending on your platform. This is a typical example:

  ```
  RCBus [RCZ180_nat_wbw] Boot Loader
  FP Switches = 0x00
  ```

```
Boot [H=Help]: 2

Booting Disk Unit 2, Slice 0, Sector 0x00000000...

Volume "Fuzix 126 Loader" [0xF200-0xF400, entry @ 0xF200]...
FUZIX version 0.4
Copyright (c) 1988-2002 by H.F.Bower, D.Braun, S.Nitschke, H.Peraza
Copyright (c) 1997-2001 by Arcady Schekochikhin, Adriano C. R. da Cunha
Copyright (c) 2013-2015 Will Sowerbutts <wi...@sowerbutts.com>
Copyright (c) 2014-2023 Alan Cox <al...@etchedpixels.co.uk>
Devboot
512kB total RAM, 448kB available to processes (15 processes max)
Enabling interrupts ... ok.
0000 : CF Card                                  - OK
0001 :  - absent
hda: hda1 hda2 (swap)
bootdev:
```

- At the `bootdev:` prompt, enter `hda1`. Fuzix should load and you will be prompted for a date/time. Here is a typical example:

```
bootdev: hda1
Mounting root fs (root_dev=1, ro): OK
Starting /init
init version 0.9.1
Checking root file system.
Current date is Fri 2023-08-18
Enter new date:
Current time is 13:30:24
Enter new time:

 ^ ^
 n n   Fuzix 0.4
 >@<
       Welcome to Fuzix
 m m

login:
```

- At the `login:` prompt, enter `root`. No password is required. You should then get a

Fuzix # command prompt.

```
login: root

Welcome to FUZIX.
#
```

You may now use Fuzix as desired. The general operation and use of Fuzix is outside of the scope of this document.

Chapter 7

Custom Applications

The operation of the RomWBW hosted operating systems is enhanced through several custom applications. You have already read about one of these – the `ASSIGN` command. These applications are functional on all of the OS variants included with RomWBW.

The applications discussed here are **not** the same as the built-in ROM applications mentioned previously. These applications run as commands within the operating systems provided by RomWBW. So, these commands are only available at an operating system prompt after an operating system has been loaded.

All of the RomWBW Custom Applications are built to function under all of the RomWBW Operating Systems (except for p-System). In general, the applications will automatically adapt as needed to the currently running operating system. One exception is `FDU` – the Floppy Disk Utility. This application requires that you pick the floppy disk interface you want to interact with.

There is more complete documentation of all of these applications in the related RomWBW manual "RomWBW Applications" found in the Doc directory of the distribution.

The following custom applications are found on the ROM disk and are, therefore, globally available.

Application	**Description
ASSIGN	Add, change, and delete drive letter assignments. Use ASSIGN /? for usage instructions.
SYSCOPY	Copy system image to a device to make it bootable. Use SYSCOPY with no parms for usage instructions.
MODE	Reconfigures serial ports dynamically.

Application	**Description
FDU	Format and test floppy disks. Menu driven interface.
FORMAT	Will someday be a command line tool to format floppy disks. Currently does nothing!
XM	XModem file transfer program adapted to hardware. Automatically uses primary serial port on system.
FLASH	Will Sowerbutts' in-situ ROM programming utility.
FDISK80	John Coffman's Z80 hard disk partitioning tool. See documentation in Doc directory.
FAT	Access MS-DOS FAT filesystems from RomWBW (based on FatFs).
TALK	Direct console I/O to a specified character device.
RTC	Manage and test the Real Time Clock hardware.
TIMER	Display value of running periodic system timer.
CPUSPD	Change the running CPU speed and wait states of the system.

Some custom applications do not fit on the ROM disk. They are found on the disk image files or the individual files can be found in the Binary/Apps directory of the distribution. They are also included on the floppy disk and hard disk images.

Application	Description
TUNE	Play .PT2, .PT3, .MYM audio files.
INTTEST	Test interrupt vector hooking.

Chapter 8

FAT Filesystem

The FAT filesystem format that originated with MS-DOS is almost ubiquitous across modern computers. Virtually all operating systems now support reading and writing files to a FAT filesystem. For this reason, RomWBW now has the ability to read and write files on FAT filesystems.

This is accomplished by running a RomWBW custom application called `FAT`. This application understands both FAT filesystems as well as CP/M filesystems.

- Files can be copied between a FAT filesystem and a CP/M filesystem, but you cannot execute files directly from a FAT filesystem.
- FAT12, FAT16, and FAT32 formats are supported.
- Long filenames are not supported. Files with long filenames will show up with their names truncated into the older 8.3 convention.
- A FAT filesystem can be located on floppy or hard disk media. For hard disk media, a valid FAT Filesystem partition must exist.
- Note that CP/M (and compatible) OSes do not support all of the filename characters that a modern computer does. The following characters are **not permitted** in a CP/M filename:

  `< > . , ; : = ? * [ ] _ % | ( ) / \`

  The FAT application does not auto-rename files when it encounters invalid filenames. It will just issue an error and quit. Additionally, the error message is not very clear about the problem.

8.1 FAT Filesystem Preparation

In general, you can create media formatted with a FAT filesystem on your RomWBW computer or on your modern computer. We will only be discussing the RomWBW-based approach here.

In the case of a floppy disk, you can use the `FAT` application to format the floppy disk. The floppy disk must already be physically formatted using RomWBW FDU or equivalent. If your floppy disk is on RomWBW disk unit 2, you could use `FAT FORMAT 2:`. This will overwrite the floppy with a FAT filesystem and all previous contents will be lost. Once formatted this way, the floppy disk can be used in a floppy drive attached to a modern computer or it can be used on RomWBW using the other `FAT` tool commands.

In the case of hard disk media, it is necessary to have a FAT partition. If you prepared your RomWBW hard disk media using the disk image process, then this partition will already be defined and you do not need to recreate it. This default FAT partition is located at approximately 512MB from the start of your disk and it is 384MB in size. So, your hard disk media must be 1GB or greater to use this default FAT partition.

You can confirm the existence of the FAT partition with `FDISK80` by using the 'P' command to show the current partition table. Here is an example of a partition table listing from `FDISK80` that includes the FAT partition (labelled "FAT16"):

```
Capacity of disk 4:  (  4G)  7813120      Geom 77381010
Nr  ---Type- A --      Start         End   LBA start  LBA count  Size
 1    RomWBW   2e      8:0:1  1023:15:16        2048    1048576  512M
 2     FAT16   06   1023:0:1  1023:15:16     1050624     786432  384M
 3             00       *** empty ***
 4             00       *** empty ***
```

If your hard disk media does not have a FAT partition already defined, you will need to define one using FDISK80 by using the 'N' command. Ensure that the location and size of the FAT partition does not overlap any of the CP/M slice area and that it fits within the size of your media.

Once the partition is defined, you will still need to format it. Just as with a floppy disk, you use the `FAT` tool to do this. If your hard disk media is on RomWBW disk unit 4, you would use `FAT FORMAT 4:`. This will look something like this:

```
E>fat format 4:

About to format FAT Filesystem on Disk Unit #4.
All existing FAT partition data will be destroyed!!!
```

```
Continue (y/n)?

Formatting... Done
```

Your FAT filesystem is now ready to use.

If your RomWBW system has multiple disk drives/slots, you can also just create a disk with your modern computer that is a dedicated FAT filesystem disk. You can use your modern computer to format the disk (floppy, CF Card, SD Card, etc.), then insert the disk in your RomWBW computer and access it using `FAT` based on its RomWBW unit number.

WARNING: Microsoft Windows will sometimes suggest reformatting partitions that it does not recognize. If you are prompted to format a partition of your SD/CF/USB Media when inserting the card into a Windows computer, you probably want to select Cancel.

8.2 FAT Application Usage

Complete instructions for the `FAT` application are found in RomWBW Applications. Here, we will just provide a couple of simple examples. Note that the FAT application is not on the ROM disk because it is too large to include there.

The most important thing to understand about the `FAT` application is how it refers to FAT filesystems vs. CP/M filesystems. It infers this based on the file specification provided. If you use a specification like `C:SAMPLE.TXT`, it will use the C: drive of your CP/M operating system. If you use a specification like `4:SAMPLE.TXT`, it will use the FAT filesystem on the disk in RomWBW disk unit 4. Basically, if you start your file or directory specification with a number followed by a colon, it means FAT filesystem. Anything else will mean CP/M filesystem.

Here are a few examples. This first example shows how to get a FAT directory listing from RomWBW disk unit 4:

```
E>fat dir 4:

Directory of 4:



E>
```

As you can see, there are currently no files there. Now let's copy a file from CP/M to the FAT directory:

```
E>fat copy sample.txt 4:
```

```
Copying...

SAMPLE.TXT ==> 4:/SAMPLE.TXT ... [OK]

    1 File(s) Copied
```

If we list the FAT directory again, you will see the file:

```
E>fat dir 4:

Directory of 4:

01/30/2023  17:50:14         29952  ---A  SAMPLE.TXT
```

Now let's copy the file from the FAT filesystem back to CP/M. This time we will get a warning about overwriting the file. For this example, we don't want to do that, so we abort and reissue the command specifying a new filename to use:

```
E>fat copy 4:sample.txt e:

Copying...

4:/SAMPLE.TXT ==> E:SAMPLE.TXT Overwrite? (Y/N) [Skipped]

    0 File(s) Copied

E>fat copy 4:sample.txt e:sample2.txt

Copying...

4:/SAMPLE.TXT ==> E:SAMPLE2.TXT ... [OK]

    1 File(s) Copied
```

Finally, let's try using wildcards:

```
E>fat copy sample*.* 4:

Copying...

SAMPLE.TXT ==> 4:/SAMPLE.TXT Overwrite? (Y/N) ... [OK]
SAMPLE2.TXT ==> 4:/SAMPLE2.TXT ... [OK]
```

```
    2 File(s) Copied
```

Chapter 9

Real Time Clock

RomWBW supports a variety of real time clock hardware. If your system has this hardware, then it will be able to maintain the current date and time even while your system is turned off. Additionally, depending on the operating system being used, you may be able to utilize date/time stamping of files.

You can determine if your system has a real time clock present (and functioning) by looking at the boot messages. Here is an example of a boot message reflecting the detection of a valid real time clock module:

```
DSRTC: MODE=STD IO=0x8A Thu 2023-10-19 14:07:11 CHARGE=ON
```

This example is from a DSRTC clock module. You may have a different one, but it will always display the current date/time.

In some cases, your real time clock will support charging of the battery or super-capacitor while the system has power. The status of this charging is displayed.

If the date/time of your RTC needs to be updated, you will need to do this with one of the utilities described below. There is no ability to update the date/time of the RTC in the RomWBW Boot Loader or Monitor.

9.1 Date/Time Utilities

RomwWBW includes two utilities for displaying or setting the date/time stored by the RTC. They are both a bit different and are briefly described below.

9.1.1 WDATE Utility

The `WDATE` utility (contributed by Kevin Boone) is an application that will display and/or update the current date/time. Its operation is described in RomWBW Applications. This utility works with any of the supported RomWBW RTC hardware. Here is an example of displaying and updating the date/time with this utility:

```
A>wdate
Thursday 19 October 14:14:43 2023

A>wdate 23 10 19 14 24 30

A>wdate
Thursday 19 October 14:24:34 2023
```

Note that `WDATE` does not have anything to do with date/time stamping of files. It merely displays and sets the real time clock value.

9.1.2 RTC Utility

Like `WDATE`, the `RTC` utility (contributed by Andrew Lynch) will let you display and set the current date/time. However, this utility only works with the DSRTC hardware (DS1302 chip). It is a "direct to hardware application". Its operation is described in RomWBW Applications. Here is an example of displaying and updatting the date/time with this utility:

```
A>rtc
Start RTC Program
RomWBW HBIOS, Mark 4 RTC Latch Port 0x8A

RTC: Version 1.9
Commands: E)xit T)ime st(A)rt S)et R)aw L)oop C)harge N)ocharge D)elay I)nit G)et P)ut B)oot W
start H)elp

RTC>t
Current time: 23-10-19 14:30:25-05

RTC>i
Init date/time.

YEAR:23
MONTH:10
DATE:19
```

```
HOURS:14
MINUTES:31
SECONDS:00
DAY:05
```

The `RTC` utility is also capable of turning the charging feature of the DS1320 chip on or off. Here is an example of turning if off and back on:

```
A>rtc
Start RTC Program
RomWBW HBIOS, Mark 4 RTC Latch Port 0x8A


RTC: Version 1.9
Commands: E)xit T)ime st(A)rt S)et R)aw L)oop C)harge N)ocharge D)elay I)nit G)et P)ut B)oot W
start H)elp

RTC>n
Trickle charger disabled.

RTC>c
Trickle charger enabled.
```

Do **not** enable charging unless you are sure that your system supports this. If your RTC is being powered by a normal battery, it would be dangerous to enable charging.

9.2 Date/Time File Stamping

If an RTC is available in your system, then most operating systems can use it to date/time stamp files. This just means recording the date/time of file creation, update, and or access in the directory. This capability is available in all of the RomWBW operating system except the original DRI CP/M 2.2.

In some cases (such as ZSDOS), you must load an RSX (memory resident utility) to enable date/time stamping of files. Additionally, you will need to initialize the directory. The procedure varies in each operation system, so you must review the associated documentation.

The date/time stamping mechanisms for each operating system are generally not compatible. If you initialize a directory for a type of stamping, you should be careful not to manipulate that directory with a different operating system with a different date/time stamping mechanism. Doing so may corrupt the directory.

The RomWBW disk images do not have date/time stamping initialized. This is to avoid any chance of directory corruption.

9.3 Timezone

None of the operating systems distributed with RomWBW have any concept of timezone. When files are date/time stamped, the date/time will simply be whatever date/time the RTC currently has.

The normal practice is to set the RTC to your local time. This implies that you would need to manually adjust the RTC for daylight savings time and/or when you travel to a different time zone.

The date/time stamps of files in directories will also be stored in local time. This includes files stored in a FAT filesystem. If you subsequently view the directory from modern machines (Windows, Linux, etc.), the date/time displayed will depend on the behavior of the modern system.

For example, Linux assumes that the date/time of files is UTC. So, if you create a file on a FAT filesystem with your RomWBW computer and then use Linux to view the directory, the date/time stamps will seem “off” by a few hours.

The only alternative you may consider is setting the date/time of your RTC to UTC. Since UTC is consistent across all timezones and daylight savings time, your file date/time stamps will also be consistent. Of course, this will mean that your RomWBW computer will display a date/time that seems wrong because it is not local time.

Chapter 10

CP/NET Networking

Digital Research created a simple network file sharing system called CP/NET. This allowed a network server running CP/NOS to host files available to network attached CP/M computers. Essentially, the host becomes a simple file sharing server.

RomWBW disk images include an adaptation of the DRI CP/NET client software provided by Douglas Miller. RomWBW does not support operation as a network server itself. However, Douglas has also developed a Java-based implementation of the DRI network server that can be used to provide host services from a modern computer.

Both CP/NET 1.2 and 3.0 clients are provided. Version 1.2 is for use with CP/M 2.2 and compatible OSes. Version 3.0 is for use with CP/M 3 and compatible OSes.

The CP/NET client software provided with RomWBW requires a supported ethernet interface module. At this time, the following are supported:

- RCBus MT011 w/ Ethernet Featherwing and (optionally) SPI FRAM (e.g., Adafruit SPI Non-Volatile FRAM Breakout)
- Duodyne Disk I/O w/ Wiz850IO and (optionally) SPI NVRAM (e.g., 25LC256)
- Generic Serial Interface

NOTE: The Generic Serial Interface is supported by RomWBW, but is not documented here. You must refer to the CP/NET documentation referenced below.

The client software interacts directly with this hardware. In a future version of RomWBW, I hope to add a generic networking API that will allow a greater range of network hardware to be used.

To use CP/NET effectively, you will want to review the documentation provided by Douglas on at his cpnet-z80 GitHub Project. Additionally, you should consult the DRI documentation which is not included with RomWBW, but is available on the cpnet-z80 site.

Below, I will provide the general steps involved in setting up a network using MT011 with RomWBW.

10.1 CP/NET Client Setup

The CP/NET client files are included on the RomWBW disk images, but they are found in user area 4. They are placed there to avoid confusing anyone that is not specifically trying to run a network client. They are only found on the CPM 2.2 and CP/M 3 slices. Using CP/NET on alternative OSes may work, but is not officially supported.

The CP/NET client files are packaged in `.LBR` library files. The library files are found in user area 4.

File	CP/NET Version	OS	Hardware
CPN12MT.LBR	CP/NET 1.2	CP/M 2.2	RCBus w/ MT011
CPN3MT.LBR	CP/NET 3	CP/M 3	RCBus w/ MT011
CPN12DUO.LBR	CP/NET 1.2	CP/M 2.2	Duodyne w/ Disk I/O
CPN3DUO.LBR	CP/NET 3	CP/M 3	Duodyne w/ Disk I/O
CPN12SER.LBR	CP/NET 1.2	CP/M 2.2	RomWBW Serial Port
CPN3SER.LBR	CP/NET 3	CP/M 3	RomWBW Serial Port

First, you need to merge the files from the correct library file into user area 0. This is done by extracting the files using the `NULU` library management utility application.

1. Start NULU specifying desired CP/NET library for <filename>:

   ```
   A>NULU 4:<filename>
   ```

2. At the NULU prompt, extract the files using the `-E *.*` command:

   ```
   -READY A0:>-E *.*
   ```

3. Exit NULU using the `-X` command:

   ```
   -Extract members A0:>-x
   ```

Here is an example of extracting the CP/NET 1.2 client files for an RCBus system w/ MT011. You should be in user area 0 when performing this operation.

```
A>nulu 4:cpn12mt
NULU 1.52  (07/12/87)
```

```
Copyright (C) 1984, 1985 & 1987 by Martin Murray
Bug fixes in version 1.52 by Mick Waters

TYPE -H FOR HELP

Library A4:CPN12MT.LBR open.
(Buffer size: 259 sectors)
Active entries: 27, Deleted: 0, Free: 5, Total: 32.
-READY A0:>-e *.*
Extracting...
  CCP     .SPR to A0:CCP     .SPR
  CPM2NET .HLP to A0:CPM2NET .HLP
  CPNBOOT .COM to A0:CPNBOOT .COM
  CPNET12 .HLP to A0:CPNET12 .HLP
  CPNETLDR.COM to A0:CPNETLDR.COM
  CPNETSTS.COM to A0:CPNETSTS.COM
  DSKRESET.COM to A0:DSKRESET.COM
  ENDLIST .COM to A0:ENDLIST .COM
  LOCAL   .COM to A0:LOCAL   .COM
  LOGIN   .COM to A0:LOGIN   .COM
  LOGOFF  .COM to A0:LOGOFF  .COM
  MAIL    .COM to A0:MAIL    .COM
  NDOS    .SPR to A0:NDOS    .SPR
  NETDOWN .COM to A0:NETDOWN .COM
  NETSTAT .COM to A0:NETSTAT .COM
  NETWORK .COM to A0:NETWORK .COM
  NVRAM   .COM to A0:NVRAM   .COM
  PIPNET  .COM to A0:PIPNET  .COM
  RDATE   .COM to A0:RDATE   .COM
  SNIOS   .SPR to A0:SNIOS   .SPR
  SRVSTAT .COM to A0:SRVSTAT .COM
  TR      .COM to A0:TR      .COM
  WIZCFG  .COM to A0:WIZCFG  .COM
  WIZDBG  .COM to A0:WIZDBG  .COM
  WIZTEST .COM to A0:WIZTEST .COM
  XSUBNET .COM to A0:XSUBNET .COM
-Extract members A0:>-x

Closing A4:CPN12MT.LBR...
```

At this point, you will need to configure your ethernet adapter for your local network using `WIZCFG`. The definitive guide to the use of `WIZCFG` is on the cpnet-z80 site in the document called "CPNET-WIZ850io.pdf". Here is an example of the commands needed to configure the WizNet:

`wizcfg w n F0`	set CP/NET node id
`wizcfg w i 192.168.1.201`	set WizNet IP address
`wizcfg w g 192.168.1.1`	set local network gateway IP address
`wizcfg w s 255.255.255.0`	set WizNet subnet mask
`wizcfg w 0 00 192.168.1.3 31100`	set server node ID, IP address, & port

You will need to use values appropriate for your local network. You can use the command `wiznet w` to display the current values which is useful to confirm they have been set as intended.

```
A>wizcfg w
Node ID:  F0H
IP Addr:  192.168.1.201
Gateway:  192.168.1.1
Subnet:   255.255.255.0
MAC:      98:76:B6:11:00:C4
Socket 0: 00H 192.168.1.3 31100 0
```

These values can be persisted across power-cycles if your system has NVRAM storage. To program the values into your NVRAM, you would use the same commands as above, but omit the `w` parameter. The "CPNET-WIZ850io.pdf" document is highly recommended to understand the operation of `WIZCFG`.

If you do not utilize NVRAM to persist your configuration, you will need to reapply these commands every time you power cycle your RomWBW computer, so I recommend putting them into a `SUBMIT` file.

After applying these commands, you should be able ping the WizNet from another computer on the local network. If this works, then the client-side is ready.

10.2 CP/NET Sever Setup

These instructions will assume you are using Douglas Miller's CpnetSocketServer to implement a CP/NOS server on your network. The definitive guide to this software is also on the [cpnet-z80] (https://github.com/durgadas311/cpnet-z80) site and is called "CpnetSocketServer.pdf".

The software is a Java application, so it can generally run anywhere there is a Java runtime environment available. I have normally used it on a Linux system and have had good results with that.

You will need to download the application called “CpnetSocketServer.jar” from the cpnet-z80 site. The application uses a configuration file. My configuration file is called “cpnet00.rc” and has these contents:

```
cpnetserver_host = 192.168.1.3
cpnetserver_port = 31100
cpnetserver_temp = P
cpnetserver_sid = 00
cpnetserver_max = 16
cpnetserver_root_dir = /home/wayne/cpnet/root
```

You will also need to setup a directory structure with the drive letters per the documentation.

To start the server, you would use a command like this:

```
java -jar CpnetSocketServer.jar conf=cpnet00.rc
```

At this point, the server should start and you should see the following:

```
CpnetSocketServer v1.3
Using config in cpnet00.rc
Server 00 Listening on 192.168.1.3 port 31100 debug false
```

Your CP/NET server should now be ready to accept client connections.

10.3 CP/NET Usage

With both the client and server configured, you are ready to load and use CP/NET on your RomWBW system. CP/NET documentation is available on the cpnet-z80 site. The document is called “dri-cpnet.pdf”.

Under CP/M 2.2, you will start the networking client using the command `CPNETLDR`. Under CP/M 3, you use the command `NDOS3`. If that works, you can map network drives as local drives using the `NETWORK` command. The `CPNETSTS` command is useful for displaying the current status.
Here is a sample session from CP/M 2.2:

```
A>cpnetldr
```

```
CP/NET 1.2 Loader
=================

BIOS         E600H  1A00H
BDOS         D800H  0E00H
SNIOS   SPR  D400H  0400H
NDOS    SPR  C800H  0C00H
TPA          0000H  C800H

CP/NET 1.2 loading complete.

A>network k:=c:[0]

A>dir k:
K: TELNET   COM : ZDENST   COM : CLRDIR   COM : RTC      COM
K: DDTZ     COM : MBASIC   COM : XSUBNET  COM : NETWORK  COM
K: WGET     COM : UNCR     COM : FLASH    COM : PIP      COM
K: TIMEZONE COM : COMPARE  COM : ZAP      COM

A>cpnetsts

CP/NET 1.2 Status
=================
Requester ID = F0H
Network Status Byte = 10H
Disk device status:
  Drive A: = LOCAL
  Drive B: = LOCAL
  Drive C: = LOCAL
  Drive D: = LOCAL
  Drive E: = LOCAL
  Drive F: = LOCAL
  Drive G: = LOCAL
  Drive H: = LOCAL
  Drive I: = LOCAL
  Drive J: = LOCAL
  Drive K: = Drive C: on Network Server ID = 00H
  Drive L: = LOCAL
  Drive M: = LOCAL
  Drive N: = LOCAL
```

```
  Drive O: = LOCAL
  Drive P: = LOCAL
Console Device = LOCAL
List Device = LOCAL
```

If you are using CpSocketServer to provide the CP/NOS server, then you will see some messages on your server console when clients connect. Here are the messages issued by the server in the above example:

```
Connection from 192.168.1.201 (31100)
Remote 192.168.1.201 is f0
Creating HostFileBdos 00 device with root dir /home/wayne/cpnet/root
```

At this point CP/NET is ready for general use. You should be able to access files on the network mapped drives just like files on your local drives.

10.4 Network Boot

It is possible to boot your MT011 equipped RomWBW system directly from a network server. This means that the operating system will be loaded directly from the network server and all of your drive letters will be provided by the network server. Duodyne is not yet supported in this mode of operation.

It is important to understand that the operating system that is loaded in this case is **not** a RomWBW enhanced operating system. Some commands (such as the `ASSIGN` command) will not be possible. Also, you will only have access to drives provided by the network server – no local disk drives will be available.

In order to do this, your MT011 Module **must** be enhanced with an NVRAM SPI FRAM mini-board. The NVRAM is used to store your WizNet configuration values so they do not need to be re-entered every time you power-cycle your system.

Using the same values from the previous example, you would issue the `WIZCFG` commands:

```
wizcfg n F0
wizcfg i 192.168.1.201
wizcfg g 192.168.1.1
wizcfg s 255.255.255.0
wizcfg 0 00 192.168.1.3 31100
```

Note that the 'w' parameter is now omitted which causes these values to be written to NVRAM.

As before, your network server will need to be running CpnetSocketServer. However, you will need to setup a directory that contains some files that will be sent to your RomWBW system when the Network boot is performed. By default the directory will be `~/NetBoot`. In this directory you need to place the following files:

- `cpnos-wbw.sys`
- `ndos.spr`
- `snios.spr`

All of these files are found in the Binary/CPNET/NetBoot directory of the RomWBW distribution.

You also need to make sure CpnetSocketServer is configured with an 'A' drive and that drive must contain (at an absolute minimum) the following file:

- `ccp.spr`

which is also found in the Binary/CPNET/NetBoot directory of RomWBW

Finally, you need to add the following line to your CpnetSocketServer configuration file:

```
netboot_default = cpnos-wbw.sys
```

To perform the network boot, you start your RomWBW system normally which should leave you at the Boot Loader prompt. The 'N' command will initiate the network boot. Here is an example of what this looks like:

```
RCBus [RCZ180_nat_wbw] Boot Loader

Boot [H=Help]: n

Loading Network Boot...
MT011 WizNET Network Boot

WBWBIOS  SPR  FD00 0100
COBDOS   SPR  FA00 0300
SNIOS    SPR  F600 0400
NDOS     SPR  EA00 0C00

58K TPA

A>
```

The CP/M operating system and the CP/NET components have been loaded directly from the network server. All of your drive letters are automatically mapped directly to the drive letters configured with CpnetSocketServer.

```
A>cpnetsts
```

```
CP/NET 1.2 Status
=================
Requester ID = F0H
Network Status Byte = 10H
Disk device status:
  Drive A: = Drive A: on Network Server ID = 00H
  Drive B: = Drive B: on Network Server ID = 00H
  Drive C: = Drive C: on Network Server ID = 00H
  Drive D: = Drive D: on Network Server ID = 00H
  Drive E: = Drive E: on Network Server ID = 00H
  Drive F: = Drive F: on Network Server ID = 00H
  Drive G: = Drive G: on Network Server ID = 00H
  Drive H: = Drive H: on Network Server ID = 00H
  Drive I: = Drive I: on Network Server ID = 00H
  Drive J: = Drive J: on Network Server ID = 00H
  Drive K: = Drive K: on Network Server ID = 00H
  Drive L: = Drive L: on Network Server ID = 00H
  Drive M: = Drive M: on Network Server ID = 00H
  Drive N: = Drive N: on Network Server ID = 00H
  Drive O: = Drive O: on Network Server ID = 00H
  Drive P: = Drive P: on Network Server ID = 00H
Console Device = LOCAL
List Device = LOCAL
```

At this point you can use CP/M and CP/NET normally, but all disk access will be to/from the network drives. There is no access to your local disk drives in this boot mode.

Chapter 11

Transferring Files

Transferring files between your modern computer and your RomWBW system can be achieved in a variety of ways. The most common of these are described below. All of these have a certain degree of complexity and I encourage new users to use the available community forums to seek assistance as needed.

11.1 Serial Port Transfers

RomWBW provides an serial file transfer program called XModem that has been adapted to run under RomWBW hardware. The program is called `XM` and is on your ROM disk as well as all of the pre-built disk images.

You can type `XM` by itself to get usage information. In general, you will run `XM` with parameters to indicate you want to send or receive a file on your RomWBW system. Then, you will use your modern computers terminal program to complete the process.

The `XM` application generally tries to detect the hardware you are using and adapt to it. However, you must ensure that you have a reliable serial connection. You must also ensure that the speed of the connection is not too fast for XModem to service. If your file transfer attempts are failing, try either of the following:

- Check that hardware flow control is enabled in your terminal emulation software.
- Reduce the speed of your serial port connection.

There is an odd interaction between XModem and partner terminal programs that can occur. Essentially, after launching `XM`, you must start the protocol on your modern computer fairly quickly (usually in about 20 seconds or so). So, if you do not pick a file on your modern computer quickly enough, you will find that the transfer completes about 16K, then hangs. The interaction that causes this is beyond the scope of this document.

11.2 Disk Image Transfers

It is possible to pass disk images between your RomWBW system and your modern computer. This assumes you have an appropriate media slot on your modern computer for the media you want to use (CF Card, SD Card, floppy drive, etc.).

The general process to get files from your modern computer to a RomWBW computer is:

1. Use `cpmtools` on your modern computer to create a RomWBW CP/M filesystem image.
2. Insert your RomWBW media (CF Card, SD Card, floppy disk, etc.) in your modern computer.
3. Use a disk imaging tool to copy the RomWBW filesystem image onto the media.
4. Move the media back to the RomWBW computer.

This process is a little complicated, but it has the benefit of allowing you to get a lot of files over to your RomWBW system quickly and with little chance of corruption.

The process can be run in reverse to get files from your RomWBW computer to a modern computer.

The exact use of these tools is a bit too much for this document, but the tools are all included in the RomWBW distribution along with usage documents.

Note that the build scripts for RomWBW create the default disk images supplied with RomWBW. It is relatively easy to customize the contents of the disk images that are part of RomWBW. This is described in more detail in the Source/Images directory of the distribution.

11.3 FAT Filesystem Transfers

The ability to interact with FAT filesystems was covered in FAT Filesystem. This capability means that you can generally use your modern computer to make an SD Card, CF Card, or USB Drive with a standard FAT32 filesystem on it, then place that media in your RomWBW computer and access the files.

When formatting the media on your modern computer, be sure to pick the FAT filesystem. NTFS and other filesystems will not work. As previously mentioned, the `FAT` application does not understand long filenames, only the traditional 8.3 filenames. If you have files on your modern computer with long filenames, it is usually easiest to rename them on the modern computer.

To copy files from your modern computer to your RomWBW computer, start by putting the disk media with the FAT filesystem in your modern computer. The modern computer should recognize it. Then copy the files you want to get to your RomWBW computer onto this media. Once done, remove the media from your modern computer and insert it in the RomWBW computer. Finally, use the `FAT` tool to copy the files onto a CP/M drive.

This process works just fine in reverse if you want to copy files from a CP/M filesystem to your modern computer.

WARNING: If you are using media that contains both a FAT partition and a RomWBW partition, your modern computer may be confused by the RomWBW partition. In some cases, it will prompt you to format the RomWBW partition because it doesn't know what it is. You will be prompted before it does this – just be careful not to allow it.

Chapter 12

Customizing RomWBW

12.1 Startup Command Processing

Most of the operating systems supported by RomWBW provide a mechanism to run commands at boot. This is similar to the AUTOEXEC.BAT files from MS-DOS.

With the exception of ZPM3 and p-System, all operating systems will look for a file called `PROFILE.SUB` on the system drive at boot. If it is found, it will be processed as a standard CP/M submit file. You can read about the use of the SUBMIT facility in the CP/M manuals included in the RomWBW distribution. Note that the boot disk must also have a copy of `SUBMIT.EXE`.

Note that the automatic startup processing generally requires booting to a disk drive. Since the ROM disk is not writable, there is no simple way to add/edit a `PROFILE.SUB` file there. If you want to customize your ROM and add a `PROFILE.SUB` file to the ROM Disk, it will work, but is a lot harder than using a boot disk.

In the case of ZPM3, the file called `STARTZPM.COM` will be run at boot. To customize this file, you use the ZCPR ALIAS facility. You will need to refer to ZCPR documentation for more information on the ALIAS facility.

p-System has its own startup command processing mechanism that is covered in the p-System documentation.

12.2 ROM Customization

The pre-built ROM images are configured for the basic capabilities of each platform. Additionally, some of the typical add-on hardware for each platform will be automatically detected and used. If you want to go beyond this, RomWBW provides a very flexible configuration mechanism based on configuration files. Creating a customized ROM requires running a build script, but it is quite easy to do.

Essentially, the creation of a custom ROM is accomplished by updating a small configuration file, then running a script to compile the software and generate the custom ROM and disk images. There are build scripts for Windows, Linux, and MacOS to accommodate virtually all users. All required build tools (compilers, assemblers, etc.) are included in the distribution, so it is not necessary to setup a build environment on your computer.

RomWBW can be built on modern Windows, Linux, or MacOS computers. The process for building a custom ROM is documented in the ReadMe.txt file in the Source directory of the distribution. Any modern version of Windows (32-bit or 64-bit), MacOS, or Linux released in the last 10 years should be able to run the build process.

For those who are interested in more than basic system customization, note that all source code is provided (including the operating systems). Modification of the source code is considered an expert level task and is left to the reader to pursue.

Note that the ROM customization process does not apply to UNA. All UNA customization is performed within the ROM setup script that is built into the ROM.

Chapter 13

UNA Hardware BIOS

John Coffman has produced a new generation of hardware BIOS called UNA. The standard RomWBW distribution includes its own hardware BIOS. However, RomWBW can alternatively be constructed with UNA as the hardware BIOS portion of the ROM. If you wish to use the UNA variant of RomWBW, then just program your ROM with the ROM image called "UNA_std.rom" in the Binary directory. This one image is suitable on **all** of the platforms and hardware UNA supports.

UNA is customized dynamically using a ROM based setup routine and the setup is persisted in the system NVRAM of the RTC chip. This means that the single UNA-based ROM image can be used on most of the RetroBrew platforms and is easily customized. UNA also supports FAT file system access that can be used for in-situ ROM programming and loading system images.

While John is likely to enhance UNA over time, there are currently a few things that UNA does not support:

- Floppy Drives
- Terminal Emulation
- Zeta 1, N8, RCBus, Easy Z80, and Dyno Systems
- Some older support boards

The UNA version embedded in RomWBW is the latest production release of UNA. RomWBW will be updated with John's upcoming UNA release with support for VGA3 as soon as it reaches production status.

Please refer to the UNA BIOS Firmware Page for more information on UNA.

13.1 UNA Usage Notes

- At startup, UNA will display a prompt similar to this:

  ```
  Boot UNA unit number or ROM? [R,X,0..3] (R):
  ```

  You generally want to choose 'R' which will then launch the RomWBW loader. Attempting to boot from a disk using a number at the UNA prompt will only work for the legacy (hd512) disk format. However, if you go to the RomWBW loader, you will be able to perform a disk boot on either disk format.

- The disk images created and distributed with RomWBW do not have the correct system track code for UNA. In order to boot to disk under UNA, you must first use SYSCOPY to update the system track of the target disk. The UNA ROM disk has the correct system track files for UNA: `CPM.SYS` and `ZSYS.SYS`. So, you can boot a ROM OS and then use one of these files to update the system track.

- Only Z-System and CP/M 2 are available OSes under UNA at this time. Since NZ-COM launches from CP/M 2, it is usable. p-System is not usable under UNA.

- Some of the RomWBW-specific applications are not UNA compatible.

Chapter 14

Upgrading

Upgrading to a newer release of RomWBW is essentially just a matter of updating the ROM chip in your system. If you have spare ROM chips for your system and a ROM programmer, it is always safest to retain your existing, working ROM chip and program a new one with the new firmware. If the new one fails to boot, you can easily return to the known working ROM.

For each ROM that comes with the RomWBW distribution, you will find that there are actually 3 different variants:

- `.rom` contains the complete ROM chip image
- `.upd` contains the system code, but omits the ROM Disk contents
- `.com` contains a CP/M executable version of the ROM code

So, for example, RCZ80_std contains the following files:

- `SBC_std.rom`
- `SBC_std.upd`
- `SBC_std.com`

The use of the `.com` variant is described below in Application Boot.

As previously discussed, the ROM in most RomWBW systems contains both the system code as well as a ROM Disk with files on it. The `.rom` variant of the ROM contains a full ROM chip image including both the system code and the Rom Disk contents. The `.upd` variant of the ROM contains only the system code portion of the ROM. If you apply the `.upd` variant to your system, it will overlay the system code, but will not overlay the ROM Disk contents (they remain intact). You may use either the `.rom` or the `.upd` file when updating your ROM chip (this does not apply to Application Boot). It is best to use the `.rom` file for your upgrade because the files on your ROM Disk should be updated whenever you update your system code. The advantage of the `.upd` variant is that is much smaller, so you can upload and apply it faster. The ROM update instructions below generally refer to using the `.rom` variant. However, you may substitute the `.upd` variant if desired.

14.1 Application Boot

Prior to attempting to reprogram your actual ROM chip, you may wish to "try" the update to ensure it will work on your system. With RomWBW, you can upload a new ROM image executable and load it from the command line. For each ROM image file (.rom) in the Binary directory, you will find a corresponding application file (.com). For example, for SBC_std.rom, there is also an SBC_std.com file.

You can upload the `.com` file to your system using XModem, then simply run the `.com` file. You will see your system go through the normal startup process just like it was started from ROM. However, your physical ROM has not been updated and the next time you boot your system, it will revert to the system image contained in ROM.

Using this `.com` version of a ROM is an excellent way to confirm that the new ROM code you intend to program will work correctly. If it does not, then you can just reboot and your old ROM will be loaded.

When the `.com` file is loaded, you will be taken to the normal Boot Loader menu. However, you will find that the only OS that is available to boot from ROM is ZSDOS. There is only room for a single OS in the .com file. Even if you don't normally use ZSDOS, this will still confirm that your system operates well under the new ROM code.

14.2 Upgrading via Flash Utility

If you do not have easy access to a ROM programmer, it is usually possible to reprogram your system ROM using the FLASH utility from Will Sowerbutts. This application, called FLASH.COM, can be found on the ROM drive of any running system. In this case, you would need to transfer the new ROM image (.rom) over to your system using XModem (or one of the other mechanisms described in the Transferring Files section). The ROM image is too large to fit on your RAM drive, so you will need to transfer it to a larger storage drive. Once the ROM image is on your system, you can use the FLASH application to update your ROM. The following is a typical example of transferring ROM image using XModem and flashing the chip in-situ.

NOTE: The FLASH utility **can not** determine the type of your ROM chip if it is write protected. Additionally, it has no way to determine if it is write protected. If the FLASH utility indicates it does not recognize your ROM chip, check to ensure the chip is not write protected.

```
E>xm r rom.rom

XMODEM v12.5 - 07/13/86
RBC, 28-Aug-2019 [WBW], ASCI

Receiving: E0:ROM.IMG
7312k available for uploads
File open - ready to receive
To cancel: Ctrl-X, pause, Ctrl-X

Thanks for the upload

E>flash write rom.rom
FLASH4 by Will Sowerbutts <will@sowerbutts.com> version 1.2.3

Using RomWBW (v2.6+) bank switching.
Flash memory chip ID is 0xBFB7: 39F040
Flash memory has 128 sectors of 4096 bytes, total 512KB
Write complete: Reprogrammed 2/128 sectors.
Verify (128 sectors) complete: OK!
```

Obviously, there is some risk to this approach since any issues with the programming or ROM image could result in a non-functional system.

To confirm your ROM chip has been successfully updated, restart your system and boot an operating system from ROM. Do not boot from a disk device yet. Review the boot messages to see if any issues have occurred.

14.3 Upgrading via XModem Flash Updater

Similar to using the Flash utility, the system ROM can be updated or upgraded through the ROM-based updater utility. This works by reprogramming the flash ROM as the file is being transferred.

Using the ROM-based updater has the advantage that secondary storage is not required to hold the new image. In other words, it is not necessary to have a mass storage device available to store the ROM image.

From the Boot Loader menu select X (Xmodem Flash Updater) and then U (Begin Update). Then initiate the Xmodem transfer of the `.rom` file. Since the XModem Flash Updater will be relatively slow, you may wish to use the `.upd` varient of the ROM.

More information can be found in the ROM Applications document.

14.4 Post Upgrade System Image and Application Update Process

Once you are satisfied that the ROM is working well, you will need to update the system images and RomWBW custom applications on your disk drives. The system images and custom applications are matched to the RomWBW ROM firmware in use. If you attempt to boot a disk or run applications that have not been updated to match the current ROM firmware, you are likely to have odd problems.

The simplest way to update your disk media is to just use your modern computer to overwrite the entire media with the latest disk image of your choice. This process is described below in the Disk Images section. If you wish to update existing disk media in your system, you need to perform the following steps.

If the disk is bootable, you need to update the system image on the disk using the procedure described in the Operating Systems section of this document.

Finally, if you have copies of any of the RomWBW custom applications on your hard disk, you need to update them with the latest copies. The following applications are found on your ROM disk. Use COPY to copy them over any older versions of the app on your disk:

- ASSIGN.COM
- SYSCOPY.COM
- MODE.COM
- FDU.COM
- FORMAT.COM
- XM.COM
- FLASH.COM
- FDISK80.COM
- TALK.COM
- RTC.COM
- TIMER.COM
- FAT.COM

For example: `B>COPY ASSIGN.COM C:`

Some RomWBW custom applications are too large to fit on the ROM disk. If you are using any of these you will need to transfer them to your system and then update all copies. These applications are found in the Binary/Apps directory of the distribution and in all of the disk images.

- TUNE.COM

The files normally contained on the standard ROM Disk is based on a 512K ROM. If your system has a smaller size ROM, then not all of these files will be included on your ROM Disk. You will need to copy them to your system from the /Binary/Apps folder of the RomWBW distribution.

WARNING: If you run a RomWBW-specific application that is not the appropriate for the version of RomWBW you are running, the application will generate an error message and abort.

14.5 System Update

As previously described, a RomWBW ROM contains ROM applications as well as a ROM disk image. If you are upgrading your ROM with a new patch level release, you may wish to upgrade just the application portion of the ROM. This is referred to as a System Update.

If the system running ROMWBW utilizes the SST39SF040 Flash chip then it is possible to do a System Update in place of a System Upgrade in some cases. A System Update would involve only updating the BIOS, ROM applications and ROM-hosted operating systems.

A System Update may be more favorable than a System Upgrade in cases such as:

- Overwriting of the ROM drive contents is not desired.
- Temporary disk space is unavailable to hold a full ROM image.
- To reduce the time taken to transfer and flash a full ROM.
- Configuration changes are only minor and do not impact disk applications.

The RomWBW build process generates a system update file along with the normal ROM image and can be identified by the extension ".upd". It will be 128Kb in size. In comparison the normal ROM image will have the extension ".rom" and be 512Kb or 1024Kb in size.

Transferring and flashing the System Update is accomplished in the same manner as described above in Upgrading via Flash Utility with the required difference being that the flash application needs to be directed to complete a partial flash using the /P command line switch.

```
E>FLASH WRITE ROM.UPD /P
```

Chapter 15

Related Projects

Outside of the hardware platforms adapted to RomWBW, there are a variety of projects that either target RomWBW specifically or provide a RomWBW-specific variation. These efforts are greatly appreciated and are listed below. Please contact the author if there are any other such projects that are not listed.

15.1 Z88DK

Z88DK is a software powerful development kit for Z80 computers supporting both C and assembly language. This kit now provides specific library support for RomWBW HBIOS. The Z88DK project is hosted at https://github.com/z88dk/z88dk.

15.2 Paleo Editor

Steve Garcia has created a Windows-hosted IDE that is tailored to development of RomWBW. The project can be found at https://github.com/alloidian/PaleoEditor.

15.3 p-System Volume Management Script

Andrew Davidson has created a Python script to automate the insertion and deletion of volumes within the p-System disk image. These scripts are hosted at https://github.com/robosnacks/psysimg.

15.4 Z80 fig-FORTH

Dimitri Theulings' implementation of fig-FORTH for the Z80 has a RomWBW-specific variant. This fig-FORTH is built into the RomWBW ROM. However, the project itself is hosted at https://github.com/dimitrit/figforth.

15.5 RomWBW Date/Time Utility

Kevin Boone has created a generic application that will display or set the date/time of an RTC on RomWBW. The application runs on all of the CP/M OS variants. This tool (`WDATE`) is included on the RomWBW OS disk images. The project is hosted at https://github.com/kevinboone/wdate-cpm.

15.6 Assembly Language Programming for the RC2014 Zed

Bruce Hall has written a very nice document that describes how to develop assembly language applications on RomWBW. It begins with the setup and configuration of a new RC2014 Zed system running RomWBW. It describes not only generic CP/M application development, but also RomWBW HBIOS programming and bare metal programming. The latest copy of this document is hosted at http://w8bh.net/Assembly for RC2014Z.pdf.

Chapter 16

Acknowledgments

I want to acknowledge that a great deal of the code and inspiration for RomWBW has been provided by or derived from the work of others in the RetroBrew Computers Community. I sincerely appreciate all of their contributions. The list below is probably missing many names – please let me know if I missed you!

- Andrew Lynch started it all when he created the N8VEM Z80 SBC which became the first platform RomWBW supported. Some of his original code can still be found in RomWBW.
- Dan Werner wrote much of the code from which RomWBW was originally derived and he has always been a great source of knowledge and advice.
- Douglas Goodall contributed code, time, testing, and advice in "the early days". He created an entire suite of application programs to enhance the use of RomWBW. Unfortunately, they have become unusable due to internal changes within RomWBW. As of RomWBW 2.6, these applications are no longer provided.
- Sergey Kiselev created several hardware platforms for RomWBW including the very popular Zeta.
- David Giles created support for the Z180 CSIO which is now included SD Card driver.
- Phil Summers contributed the Forth and BASIC adaptations in ROM, the AY-3-8910 sound driver, DMA support, and a long list of general code and documentation enhancements.
- Ed Brindley contributed some of the code that supports the RCBus platform.
- Spencer Owen created the RC2014 series of hobbyist kit computers which has expo-

nentially increased RomWBW usage. Some of his kits include RomWBW.

- Stephen Cousins has likewise created a series of hobbyist kit computers at Small Computer Central and is distributing RomWBW with many of them.
- Alan Cox has contributed some driver code and has provided a great deal of advice.
- The CP/NET client files were developed by Douglas Miller.
- Phillip Stevens contributed support for FreeRTOS.
- Curt Mayer contributed the original Linux / MacOS build process.
- UNA BIOS and FDISK80 are the products of John Coffman.
- FLASH4 is a product of Will Sowerbutts.
- CLRDIR is a product of Max Scane.
- Tasty Basic is a product of Dimitri Theulings.
- Dean Netherton contributed the sound driver interface and the SN76489 sound driver.
- The RomWBW Disk Catalog document was produced by Mykl Orders.
- Rob Prouse has created many of the supplemental disk images including Aztec C, HiTech C, SLR Z80ASM, Turbo Pascal, Microsoft BASIC Compiler, Microsoft Fortran Compiler, and a Games compendium.
- Martin R has provided substantial help reviewing and improving the User Guide.
- Jacques Pelletier has contributed the DS1501 RTC driver code.
- Jose Collado has contributed enhancements to the TMS driver including compatibility with standard TMS register configuration.
- Kevin Boone has contributed a generic HBIOS date/time utility (WDATE).
- Matt Carroll has contributed a fix to XM.COM that corrects the port specification when doing a send.
- Dean Jenkins enhanced the build process to accommodate the Raspberry Pi 4.
- Tom Plano has contributed a new utility (HTALK) to allow talking directly to HBIOS COM ports.
- Lars Nelson has contributed several generic utilities such as a universal (OS agnostic) UNARC application.
- Dylan Hall added support for specifying a secondary console.

- Bill Shen has contributed boot loaders for several of his systems.
- Laszlo Szolnoki has contributed an EF9345 video display controller driver.
- Ladislau Szilagyi has contributed an enhanced version of CP/M Cowgol that leverages RomWBW memory banking.

Contributions of all kinds to RomWBW are very welcome.

Chapter 17

Licensing

RomWBW is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

RomWBW is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with RomWBW. If not, see https://www.gnu.org/licenses/.

Portions of RomWBW were created by, contributed by, or derived from the work of others. It is believed that these works are being used in accordance with the intentions and/or licensing of their creators.

If anyone feels their work is being used outside of its intended licensing, please notify:

> Wayne Warthen
> wwarthen@gmail.com

RomWBW is an aggregate work. It is composed of many individual, standalone programs that are distributed as a whole to function as a cohesive system. Each program may have its own licensing which may be different from other programs within the aggregate.

In some cases, a single program (e.g., CP/M Operating System) is composed of multiple components with different licenses. It is believed that in all such cases the licenses are compatible with GPL version 3.

RomWBW encourages code contributions from others. Contributors may assert their own copyright in their contributions by annotating the contributed source code appropriately. Contributors are further encouraged to submit their contributions via the RomWBW source code control system to ensure their contributions are clearly documented.

All contributions to RomWBW are subject to this license.

Chapter 18

Getting Assistance

The best way to get assistance with RomWBW or any aspect of the RetroBrew Computers projects is via one of the community forums:

- RetroBrew Computers Forum
- RC2014 Google Group
- retro-comp Google Group

Submission of issues and bugs are welcome at the RomWBW GitHub Repository.

Also feel free to email Wayne Warthen at wwarthen@gmail.com.

Chapter 19

Appendixes

19.1 Appendix A - Pre-built ROM Images

This appendix contains a summary of the system configuration target for each of the pre-built ROM images included in the RomWBW distribution. It is intended to help you select the correct ROM image and understand the basic hardware components supported. Detailed hardware system configuration information should be obtained from your system provider/designer. I am happy to provide support adapting RomWBW to new or modified systems – see Getting Assistance for contact information.

The standard ROM images will detect and install support for certain devices and peripherals that are on-board or frequently used with each platform as documented below. If the device or peripheral is not detected at boot, the ROM will simply bypass support appropriately.

By default, RomWBW will use the first available character device it discovers for the initial console. Serial devices are scanned in the following order:

1. ASCI: Zilog Z180 CPU Built-in Serial Ports
2. Z2U: Zilog Z280 CPU Built-in Serial Ports
3. UART: 16C550 Family Serial Interface
4. DUART: SCC2681 or compatible Dual UART
5. SIO: Zilog Serial Port Interface
6. ACIA: MC68B50 Asynchronous Communications Interface Adapter

In some cases, support for multiple hardware components with potentially conflicting resource usage are handled by a single ROM image. It is up to the user to ensure that no conflicting hardware is in use.

The RomWBW `TUNE` application will detect an AY-3-8910/YM2149 Sound Module regardless of whether support for it is included in the RomWBW HBIOS configuration.

19.1.1 RetroBrew Z80 SBC

ROM Image File: SBC_std.rom

Default CPU Speed	8.000 MHz
Interrupts	None
System Timer	None
Serial Default	38400 Baud
Memory Manager	SBC
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- DSRTC: MODE=STD, IO=112
- UART: MODE=SBC, IO=104
- UART: MODE=CAS, IO=128
- UART: MODE=MFP, IO=104
- UART: MODE=4UART, IO=192
- UART: MODE=4UART, IO=200
- UART: MODE=4UART, IO=208
- UART: MODE=4UART, IO=216
- SIO MODE=ZP, IO=176, CHANNEL A
- SIO MODE=ZP, IO=176, CHANNEL B
- VGA: IO=224, KBD MODE=PS/2, KBD IO=224
- CVDU: MODE=ECB, IO=224, KBD MODE=PS/2, KBD IO=226
- CVDU occupies 905 bytes.
- KBD: ENABLED
- PRP: IO=168
- PRPCON: ENABLED
- PRPSD: ENABLED
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=DIO, IO=54, DRIVE 0, TYPE=3.5” HD
- FD: MODE=DIO, IO=54, DRIVE 1, TYPE=3.5” HD
- PPIDE: IO=96, MASTER
- PPIDE: IO=96, SLAVE

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.2 RetroBrew Z80 SimH

ROM Image File: SBC_simh.rom

Default CPU Speed	8.000 MHz
Interrupts	Mode 1
System Timer	SimH
Serial Default	38400 Baud
Memory Manager	SBC
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- SIMRTC: IO=254
- UART: MODE=SBC, IO=104
- UART: MODE=CAS, IO=128
- UART: MODE=MFP, IO=104
- UART: MODE=4UART, IO=192
- UART: MODE=4UART, IO=200
- UART: MODE=4UART, IO=208
- UART: MODE=4UART, IO=216
- SIO MODE=ZP, IO=176, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=ZP, IO=176, CHANNEL B, INTERRUPTS ENABLED
- FONTS occupy 0 bytes.
- MD: TYPE=RAM
- MD: TYPE=ROM
- HDSK: IO=253, DEVICE COUNT=2

Notes:

- Image for SimH emulator
- CPU speed and Serial configuration not relevant in emulator

19.1.3 RetroBrew N8 Z180 SBC

ROM Image File: N8_std.rom

Default CPU Speed	18.432 MHz
Interrupts	Mode 2
System Timer	Z180
Serial Default	38400 Baud
Memory Manager	N8
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- DSRTC: MODE=STD, IO=136
- ASCI: IO=64, INTERRUPTS ENABLED
- ASCI: IO=65, INTERRUPTS ENABLED
- UART: MODE=CAS, IO=128
- UART: MODE=4UART, IO=192
- UART: MODE=4UART, IO=200
- UART: MODE=4UART, IO=208
- UART: MODE=4UART, IO=216
- TMS: MODE=N8, IO=152
- PPK: ENABLED
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=N8, IO=140, DRIVE 0, TYPE=3.5” HD
- FD: MODE=N8, IO=140, DRIVE 1, TYPE=3.5” HD
- SD: MODE=CSIO, IO=136, UNITS=1
- AY38910: MODE=N8, IO=156, CLOCK=1789772 HZ

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present
- SD Card interface is configured for CSIO (N8 date code >= 2312)

19.1.4 Zeta Z80 SBC

ROM Image File: ZETA_std.rom

Default CPU Speed	8.000 MHz
Interrupts	None
System Timer	None
Serial Default	38400 Baud
Memory Manager	SBC
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- DSRTC: MODE=STD, IO=112
- UART: MODE=SBC, IO=104
- PPP: IO=96
- PPPCON: ENABLED
- PPPSD: ENABLED
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=DIO, IO=54, DRIVE 0, TYPE=3.5" HD

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present
- If ParPortProp is installed, initial console output is determined by JP1:
  - Shorted: console to on-board serial port
  - Open: console to ParPortProp video and keyboard

19.1.5 Zeta V2 Z80 SBC

ROM Image File: ZETA2_std.rom

Default CPU Speed	8.000 MHz
Interrupts	Mode 2
System Timer	CTC
Serial Default	38400 Baud
Memory Manager	Z2
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- DSRTC: MODE=STD, IO=112
- UART: MODE=SBC, IO=104
- PPP: IO=96
- PPPCON: ENABLED
- PPPSD: ENABLED
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=ZETA2, IO=48, DRIVE 0, TYPE=3.5" HD
- CTC: IO=32, TIMER MODE=COUNTER, DIVISOR=18432, HI=256, LO=72, INTERRUPTS ENABLED

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present
- If ParPortProp is installed, initial console output is determined by JP1:
  - Shorted: console to on-board serial port
  - Open: console to ParPortProp video and keyboard

19.1.6 Mark IV Z180 SBC

ROM Image File: MK4_std.rom

Default CPU Speed	18.432 MHz
Interrupts	Mode 2
System Timer	Z180
Serial Default	38400 Baud
Memory Manager	Z180
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- DSRTC: MODE=STD, IO=138
- ASCI: IO=64, INTERRUPTS ENABLED
- ASCI: IO=65, INTERRUPTS ENABLED
- UART: MODE=CAS, IO=128
- UART: MODE=MFP, IO=104
- UART: MODE=4UART, IO=192
- UART: MODE=4UART, IO=200
- UART: MODE=4UART, IO=208
- UART: MODE=4UART, IO=216
- VGA: IO=224, KBD MODE=PS/2, KBD IO=224
- CVDU: MODE=ECB, IO=224, KBD MODE=PS/2, KBD IO=226
- KBD: ENABLED
- PRP: IO=168
- PRPCON: ENABLED
- PRPSD: ENABLED
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=DIDE, IO=42, DRIVE 0, TYPE=3.5" HD
- FD: MODE=DIDE, IO=42, DRIVE 1, TYPE=3.5" HD
- IDE: MODE=MK4, IO=128, MASTER
- IDE: MODE=MK4, IO=128, SLAVE
- SD: MODE=MK4, IO=137, UNITS=1

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.7 RCBus Z80 CPU Module

ROM Image File: RCZ80_std.rom

Default CPU Speed	7.372 MHz
Interrupts	Mode 1
System Timer	None
Serial Default	115200 Baud
Memory Manager	Z2
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0, SWIO=0
- DSRTC: MODE=STD, IO=192
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- ACIA: IO=128, INTERRUPTS ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5” HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5” HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE
- CTC: IO=136

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present

ROM Image File: RCZ80_kio.rom

Default CPU Speed	7.372 MHz
Interrupts	Mode 2
System Timer	CTC
Serial Default	115200 Baud
Memory Manager	Z2
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0, SWIO=0
- DSRTC: MODE=STD, IO=192
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=STD, IO=136, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=STD, IO=136, CHANNEL B, INTERRUPTS ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5” HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5” HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE
- CTC: IO=132, TIMER MODE=TIMER/16, DIVISOR=9216, HI=256, LO=36, INTERRUPTS ENABLED

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present
- SIO Serial baud rate managed by CTC

19.1.8 RCBus Z180 CPU Module

ROM Image File: RCZ180_ext.rom

Default CPU Speed	18.432 MHz
Interrupts	Mode 2
System Timer	Z180
Serial Default	115200 Baud
Memory Manager	Z2
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0, SWIO=0
- DSRTC: MODE=STD, IO=12
- INTRTC: ENABLED
- ASCI: IO=192, INTERRUPTS ENABLED
- ASCI: IO=193, INTERRUPTS ENABLED
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5” HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5” HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE

Notes:

- For use with Z2 bank switched memory board (Z2 external memory management)

- CPU speed will be dynamically measured at startup if DSRTC is present

ROM Image File: RCZ180_nat.rom

Default CPU Speed	18.432 MHz
Interrupts	Mode 2
System Timer	Z180
Serial Default	115200 Baud
Memory Manager	Z180
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0, SWIO=0
- DSRTC: MODE=STD, IO=12
- INTRTC: ENABLED
- ASCI: IO=192, INTERRUPTS ENABLED
- ASCI: IO=193, INTERRUPTS ENABLED
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5" HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5" HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE

Notes:

- For use with linear memory board (Z180 native memory management)
- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.9 RCBus Z280 CPU Module

ROM Image File: RCZ280_ext.rom

Default CPU Speed	6.000 MHz
Interrupts	Mode 1
System Timer	None
Serial Default	115200 Baud
Memory Manager	Z2
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0, SWIO=0
- DSRTC: MODE=STD, IO=192
- Z2U: IO=16
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- ACIA: IO=128, INTERRUPTS ENABLED
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5" HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5" HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE

Notes:

- For use with Z2 bank switched memory board (Z2 external memory management)

ROM Image File: RCZ280_nat.rom

Default CPU Speed	6.000 MHz
Interrupts	Mode 3
System Timer	Z280
Serial Default	115200 Baud
Memory Manager	Z280
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0, SWIO=0
- DSRTC: MODE=STD, IO=192
- Z2U: IO=16, INTERRUPTS ENABLED
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5” HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5” HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE

Notes:

- For use with linear memory board (Z280 native memory management)

19.1.10 Easy Z80 SBC

ROM Image File: RCZ80_easy.rom

Default CPU Speed	10.000 MHz
Interrupts	Mode 2
System Timer	CTC
Serial Default	115200 Baud
Memory Manager	Z2
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0, SWIO=0
- DSRTC: MODE=STD, IO=192
- INTRTC: ENABLED
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=STD, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=STD, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5" HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5" HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE
- CTC: IO=136, TIMER MODE=COUNTER, DIVISOR=18432, HI=256, LO=72, INTERRUPTS ENABLED

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.11 Tiny Z80 SBC

ROM Image File: RCZ80_tiny.rom

Default CPU Speed	16.000 MHz
Interrupts	Mode 2
System Timer	CTC
Serial Default	115200 Baud
Memory Manager	Z2
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0, SWIO=0
- DSRTC: MODE=STD, IO=192
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=STD, IO=24, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=STD, IO=24, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5” HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5” HD
- IDE: MODE=RC, IO=144, MASTER
- IDE: MODE=RC, IO=144, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE
- CTC: IO=16, TIMER MODE=COUNTER, DIVISOR=18432, HI=256, LO=72, INTERRUPTS ENABLED

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.12 Z80-512K CPU/RAM/ROM Module

ROM Image File: RCZ80_skz.rom

Default CPU Speed	7.372 MHz
Interrupts	Mode 1
System Timer	None
Serial Default	115200 Baud
Memory Manager	Z2
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0, SWIO=0
- DSRTC: MODE=STD, IO=192
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- ACIA: IO=128, INTERRUPTS ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5” HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5” HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE
- CTC: IO=136

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.13 Small Computer SC126 Z180 SBC

ROM Image File: SCZ180_sc126.rom

Default CPU Speed	18.432 MHz
Interrupts	Mode 2
System Timer	Z180
Serial Default	115200 Baud
Memory Manager	Z180
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=13, SWIO=0
- DSRTC: MODE=STD, IO=12
- INTRTC: ENABLED
- ASCI: IO=192, INTERRUPTS ENABLED
- ASCI: IO=193, INTERRUPTS ENABLED
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5” HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5” HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE
- SD: MODE=SC, IO=12, UNITS=1
- AY38910: MODE=RCZ180, IO=104, CLOCK=1789772 HZ

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.14 Small Computer SC130 Z180 SBC

ROM Image File: SCZ180_sc130.rom

Default CPU Speed	18.432 MHz
Interrupts	Mode 2
System Timer	Z180
Serial Default	115200 Baud
Memory Manager	Z180
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0, SWIO=0
- DSRTC: MODE=STD, IO=12
- INTRTC: ENABLED
- ASCI: IO=192, INTERRUPTS ENABLED
- ASCI: IO=193, INTERRUPTS ENABLED
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5” HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5” HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE
- SD: MODE=SC, IO=12, UNITS=1
- AY38910: MODE=RCZ180, IO=104, CLOCK=1789772 HZ

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.15 Small Computer SC131 Z180 Pocket Computer

ROM Image File: SCZ180_sc131.rom

Default CPU Speed	18.432 MHz
Interrupts	Mode 2
System Timer	Z180
Serial Default	115200 Baud
Memory Manager	Z180
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- INTRTC: ENABLED
- ASCI: IO=192, INTERRUPTS ENABLED
- ASCI: IO=193, INTERRUPTS ENABLED
- MD: TYPE=RAM
- MD: TYPE=ROM
- SD: MODE=SC, IO=12, UNITS=1

Notes:

19.1.16 Small Computer SC140 Z180 CPU Module

ROM Image File: SCZ180_sc140.rom

Default CPU Speed	18.432 MHz
Interrupts	Mode 2
System Timer	Z180
Serial Default	115200 Baud
Memory Manager	Z180
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=160, SWIO=160
- DSRTC: MODE=STD, IO=12
- INTRTC: ENABLED
- ASCI: IO=192, INTERRUPTS ENABLED
- ASCI: IO=193, INTERRUPTS ENABLED
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5" HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5" HD
- IDE: MODE=RC, IO=144, MASTER
- IDE: MODE=RC, IO=144, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE
- SD: MODE=SC, IO=12, UNITS=1

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.17 Small Computer SC503 Z180 CPU Module

ROM Image File: SCZ180_sc503.rom

Default CPU Speed	18.432 MHz
Interrupts	Mode 2
System Timer	Z180
Serial Default	115200 Baud
Memory Manager	Z180
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=160, SWIO=160
- DSRTC: MODE=STD, IO=12
- INTRTC: ENABLED
- ASCI: IO=192, INTERRUPTS ENABLED
- ASCI: IO=193, INTERRUPTS ENABLED
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5” HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5” HD
- IDE: MODE=RC, IO=144, MASTER
- IDE: MODE=RC, IO=144, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE
- SD: MODE=SC, IO=12, UNITS=1

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.18 Small Computer SC700 Z180 CPU Module

ROM Image File: SCZ180_sc700.rom

Default CPU Speed	18.432 MHz
Interrupts	Mode 2
System Timer	Z180
Serial Default	115200 Baud
Memory Manager	Z180
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0
- DSRTC: MODE=STD, IO=12
- INTRTC: ENABLED
- ASCI: IO=192, INTERRUPTS ENABLED
- ASCI: IO=193, INTERRUPTS ENABLED
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60 S- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5” HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5” HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE
- SD: MODE=SC, IO=12, UNITS=1
- AY38910: MODE=RCZ180, IO=104, CLOCK=1789772 HZ

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.19 Dyno Z180 SBC

ROM Image File: DYNO_std.rom

Default CPU Speed	18.432 MHz
Interrupts	Mode 2
System Timer	Z180
Serial Default	38400 Baud
Memory Manager	Z180
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- BQRTC: IO=80
- ASCI: IO=192, INTERRUPTS ENABLED
- ASCI: IO=193, INTERRUPTS ENABLED
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=DYNO, IO=132, DRIVE 0, TYPE=3.5” HD
- FD: MODE=DYNO, IO=132, DRIVE 1, TYPE=3.5” HD
- PPIDE: IO=76, MASTER
- PPIDE: IO=76, SLAVE

Notes:

19.1.20 Nhyodyne Z80 MBC

ROM Image File: MBC_std.rom

Default CPU Speed	8.000 MHz
Interrupts	None
System Timer	None
Serial Default	38400 Baud
Memory Manager	MBC
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- PKD: IO=96
- DSRTC: MODE=STD, IO=112
- UART: MODE=SBC, IO=104
- UART: MODE=DUAL, IO=128
- UART: MODE=DUAL, IO=136
- SIO MODE=ZP, IO=176, CHANNEL A
- SIO MODE=ZP, IO=176, CHANNEL B
- PIO: IO=184, CHANNEL A
- PIO: IO=184, CHANNEL B
- PIO: IO=188, CHANNEL A
- PIO: IO=188, CHANNEL B
- LPT: MODE=SPP, IO=232
- CVDU: MODE=MBC, IO=224, KBD MODE=PS/2, KBD IO=226
- TMS: MODE=MBC, IO=152
- KBD: ENABLED
- ESP: IO=156
- ESPCON: ENABLED
- ESPSER: DEVICE=0
- ESPSER: DEVICE=1
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=MBC, IO=48, DRIVE 0, TYPE=3.5” HD
- FD: MODE=MBC, IO=48, DRIVE 1, TYPE=3.5” HD
- PPIDE: IO=96, MASTER
- PPIDE: IO=96, SLAVE
- SPK: IO=112

- CTC: IO=176

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.21 Rhyophyre Z180 SBC

ROM Image File: RPH_std.rom

Default CPU Speed	18.432 MHz
Interrupts	None
System Timer	None
Serial Default	38400 Baud
Memory Manager	RPH
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- DSRTC: MODE=STD, IO=132
- ASCI: IO=64
- ASCI: IO=65
- GDC: MODE=RPH, DISPLAY=EGA, IO=144
- KBD: ENABLED
- MD: TYPE=RAM
- MD: TYPE=ROM
- PPIDE: IO=136, MASTER
- PPIDE: IO=136, SLAVE

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.22 Z80 ZRC CPU Module

ROM Image File: RCZ80_zrc.rom

Default CPU Speed	14.745 MHz
Interrupts	Mode 1
System Timer	None
Serial Default	115200 Baud
Memory Manager	ZRC
ROM Size	512 KB
RAM Size	1536 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0, SWIO=0
- DSRTC: MODE=STD, IO=192
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- ACIA: IO=128, INTERRUPTS ENABLED
- VRC: IO=0, KBD MODE=VRC, KBD IO=244
- KBD: ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5” HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5” HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE
- CTC: IO=136

Notes:

- ZRC is actually contains no ROM and 2MB of RAM. The first 512KB of RAM is loaded from disk and then handled like ROM.
- CPU speed will be dynamically measured at startup if DSRTC is present

ROM Image File: RCZ80_zrc_ram.rom

Default CPU Speed	14.745 MHz
Interrupts	Mode 1
System Timer	None
Serial Default	115200 Baud
Memory Manager	ZRC
ROM Size	0 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0, SWIO=0
- DSRTC: MODE=STD, IO=192
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- ACIA: IO=128, INTERRUPTS ENABLED
- VRC: IO=0, KBD MODE=VRC, KBD IO=244
- KBD: ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- MD: TYPE=RAM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5” HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5” HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE
- CTC: IO=136

Notes:

- ROMless boot – HBIOS is loaded from disk at boot
- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.23 Z80 ZRC512 CPU Module

ROM Image File: RCZ80_zrc512.rom

Default CPU Speed	22.000 MHz
Interrupts	Mode 1
System Timer	None
Serial Default	115200 Baud
Memory Manager	ZRC
ROM Size	0 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0, SWIO=0
- DSRTC: MODE=STD, IO=192
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- ACIA: IO=128, INTERRUPTS ENABLED
- VRC: IO=0, KBD MODE=VRC, KBD IO=244
- KBD: ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- MD: TYPE=RAM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5” HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5” HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE
- CTC: IO=136

Notes:

- ROMless boot – HBIOS is loaded from disk at boot

- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.24 Z180 Z1RCC CPU Module

ROM Image File: RCZ180_z1rcc.rom

Default CPU Speed	18.432 MHz
Interrupts	Mode 2
System Timer	Z180
Serial Default	115200 Baud
Memory Manager	Z180
ROM Size	0 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0, SWIO=0
- DSRTC: MODE=STD, IO=12
- INTRTC: ENABLED
- ASCI: IO=192, INTERRUPTS ENABLED
- ASCI: IO=193, INTERRUPTS ENABLED
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- MD: TYPE=RAM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5” HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5” HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE

Notes:

- ROMless boot – HBIOS is loaded from disk at boot
- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.25 Z280 ZZRCC CPU Module

ROM Image File: RCZ280_zzrcc.rom

Default CPU Speed	14.745 MHz
Interrupts	Mode 3
System Timer	Z280
Serial Default	115200 Baud
Memory Manager	Z280
ROM Size	256 KB
RAM Size	256 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0, SWIO=0
- DSRTC: MODE=STD, IO=192
- Z2U: IO=16, INTERRUPTS ENABLED
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- VRC: IO=0, KBD MODE=VRC, KBD IO=244
- KBD: ENABLED
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5” HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5” HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE

Notes:

- ZZRCC actually contains no ROM and 512KB of RAM. The first 256KB of RAM is loaded from disk and then handled like ROM.
- CPU speed will be dynamically measured at startup if DSRTC is present

ROM Image File: RCZ280_zzrcc_ram.rom

Default CPU Speed	14.745 MHz
Interrupts	Mode 3
System Timer	Z280
Serial Default	115200 Baud
Memory Manager	Z280
ROM Size	0 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0, SWIO=0
- DSRTC: MODE=STD, IO=192
- Z2U: IO=16, INTERRUPTS ENABLED
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- VRC: IO=0, KBD MODE=VRC, KBD IO=244
- KBD: ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- MD: TYPE=RAM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5” HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5” HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE

Notes:

- ROMless boot – HBIOS is loaded from disk at boot
- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.26 Z280 ZZ80MB SBC

ROM Image File: RCZ280_zz80mb.rom

Default CPU Speed	12.000 MHz
Interrupts	Mode 3
System Timer	Z280
Serial Default	115200 Baud
Memory Manager	Z280
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0, SWIO=0
- DSRTC: MODE=STD, IO=192
- Z2U: IO=16, INTERRUPTS ENABLED
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- VRC: IO=0, KBD MODE=VRC, KBD IO=244
- KBD: ENABLED
- CH: IO=62
- CH: IO=60
- CHUSB: IO=62
- CHUSB: IO=60
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5” HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5” HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.27 Z80-Retro SBC

ROM Image File: Z80RETRO_std.rom

Default CPU Speed	14.745 MHz
Interrupts	Mode 2
System Timer	None
Serial Default	38400 Baud
Memory Manager	Z2
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- SIO MODE=Z80R, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=Z80R, IO=128, CHANNEL B, INTERRUPTS ENABLED
- MD: TYPE=RAM
- MD: TYPE=ROM
- SD: MODE=, IO=104, UNITS=1
- CTC: IO=64

Notes:

19.1.28 S100 Computers Z180

ROM Image File: S100_std.rom

Default CPU Speed	18.432 MHz
Interrupts	Mode 2
System Timer	Z180
Serial Default	57600 Baud
Memory Manager	Z180
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0
- INTRTC: ENABLED
- ASCI: IO=192, INTERRUPTS ENABLED
- ASCI: IO=193, INTERRUPTS ENABLED
- SCON: IO=0
- MD: TYPE=RAM
- MD: TYPE=ROM
- SD: MODE=SC, IO=12, UNITS=1

Notes:

- Z180 SBC SW2 (IOBYTE) Dip Switches:

Bit	Setting	Function
0	Off	Use Z180 ASCI Channel A for console
	On	Use Propeller Console
1	Off	Boot to RomWBW Boot Loader
	On	Boot to S100 Monitor

19.1.29 Duodyne Z80 System

ROM Image File: DUO_std.rom

Default CPU Speed	8.000 MHz
Interrupts	Mode 2
System Timer	CTC
Serial Default	38400 Baud
Memory Manager	Z2
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- DSRTC: MODE=STD, IO=148
- PCF: IO=86
- UART: MODE=SBC, IO=88
- UART: MODE=AUX, IO=168
- UART: MODE=DUAL, IO=112
- UART: MODE=DUAL, IO=120
- SIO MODE=ZP, IO=96, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=ZP, IO=96, CHANNEL B, INTERRUPTS ENABLED
- PIO: IO=104, CHANNEL A
- PIO: IO=104, CHANNEL B
- PIO: IO=108, CHANNEL A
- PIO: IO=108, CHANNEL B
- LPT: MODE=SPP, IO=72
- TMS: MODE=MBC, IO=160
- DMA: MODE=DUO, IO=64
- CH: IO=78
- CHUSB: IO=78
- CHSD: IO=78
- ESP: IO=156
- ESPCON: ENABLED
- ESPSER: DEVICE=0
- ESPSER: DEVICE=1
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=DUO, IO=128, DRIVE 0, TYPE=3.5” HD
- FD: MODE=DUO, IO=128, DRIVE 1, TYPE=3.5” HD

- PPIDE: IO=136, MASTER
- PPIDE: IO=136, SLAVE
- SD: MODE=, IO=140, UNITS=1
- SPK: IO=148
- CTC: IO=96, TIMER MODE=COUNTER, DIVISOR=18432, HI=256, LO=72, INTERRUPTS ENABLED
- AY38910: MODE=DUO, IO=164, CLOCK=1789772 HZ

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.30 Heath H8 Z80 System

ROM Image File: HEATH_std.rom

Default CPU Speed	7.372 MHz
Interrupts	Mode 1
System Timer	None
Serial Default	115200 Baud
Memory Manager	Z2
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- FP: LEDIO=0, SWIO=0
- DSRTC: MODE=STD, IO=192
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- SIO MODE=RC, IO=128, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=128, CHANNEL B, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL A, INTERRUPTS ENABLED
- SIO MODE=RC, IO=132, CHANNEL B, INTERRUPTS ENABLED
- ACIA: IO=128, INTERRUPTS ENABLED
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=RCWDC, IO=80, DRIVE 0, TYPE=3.5" HD
- FD: MODE=RCWDC, IO=80, DRIVE 1, TYPE=3.5" HD
- IDE: MODE=RC, IO=16, MASTER
- IDE: MODE=RC, IO=16, SLAVE
- PPIDE: IO=32, MASTER
- PPIDE: IO=32, SLAVE
- CTC: IO=136

Notes:

- CPU speed will be dynamically measured at startup if DSRTC is present

19.1.31 EP Mini-ITX Z180

ROM Image File: EPITX_std.rom

Default CPU Speed	18.432 MHz
Interrupts	Mode 2
System Timer	Z180
Serial Default	115200 Baud
Memory Manager	Z180
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- INTRTC: ENABLED
- ASCI: IO=192, INTERRUPTS ENABLED
- ASCI: IO=193, INTERRUPTS ENABLED
- UART: MODE=RC, IO=160
- UART: MODE=RC, IO=168
- TMS: MODE=MSX, IO=152
- MD: TYPE=RAM
- MD: TYPE=ROM
- FD: MODE=EPFDC, IO=72, DRIVE 0, TYPE=3.5” HD
- FD: MODE=EPFDC, IO=72, DRIVE 1, TYPE=3.5” HD
- SD: MODE=, IO=66, UNITS=1

Notes:

19.1.32 NABU w/ RomWBW Option Board

ROM Image File: NABU_std.rom

Default CPU Speed	3.580 MHz
Interrupts	Mode 1
System Timer	None
Serial Default	115200 Baud
Memory Manager	Z2
ROM Size	512 KB
RAM Size	512 KB

Supported Hardware (see Appendix B - Device Summary):

- UART: MODE=NABU, IO=72
- TMS: MODE=NABU, IO=160
- MD: TYPE=RAM
- MD: TYPE=ROM
- PPIDE: IO=96, MASTER
- PPIDE: IO=96, SLAVE
- AY38910: MODE=NABU, IO=65, CLOCK=1789772 HZ

Notes:

- TMS video assumes F18A replacement for TMS9918

19.2 Appendix B - Device Summary

The table below briefly describes each of the possible devices that may be discovered by RomWBW in your system.

ID	Type	Description
ACIA	Char	MC68B50 Asynchronous Communications Interface Adapter
ASCI	Char	Zilog Z180 CPU Built-in Serial Ports
AY	Audio	AY-3-8910/YM2149 Programmable Sound Generator
BQRTC	RTC	BQ4845P Real Time Clock
CH	System	CH375/376 USB Controller
CHSD	Disk	CH37x SD Card Interface
CHUSB	Disk	CH37x USB Drive Interface
CTC	System	Zilog Clock/Timer
CVDU	Video	MC8563-based Video Display Controller
DMA	System	Zilog DMA Controller
DS1307	RTC	Maxim DS1307 PCF I2C Real-Time Clock w/ NVRAM
DS1501RTC	RTC	Maxim DS1501/DS1511 Watchdog Real-Time Clock
DSRTC	RTC	Maxim DS1302 Real-Time Clock w/ NVRAM
DUART	Char	SCC2681 or compatible Dual UART
EF	Char	EF9345 Video Display Controller
EMM	Disk	Disk drive on Parallel Port emm interface (Zip Drive)
FD	Disk	8272 or compatible Floppy Disk Controller
FP	System	Simple LED & Switch Front Panel
GDC	Video	uPD7220 Video Display Controller
HDSK	Disk	SIMH Simulator Hard Disk
ICM	DsKy	ICM7218-based Display/Keypad on PPI
IDE	Disk	IDE/ATA/ATAPI Hard Disk Interface
IMM	Disk	IMM Zip Drive on PPI
INTRTC	RTC	Interrupt-based Real Time Clock
KBD	Keyboard	8242 PS/2 Keyboard Controller
KIO	System	Zilog Serial/ Parallel Counter/Timer
LPT	Char	Parallel I/O Controller
MD	Disk	ROM/RAM Disk
MSXKYB	Keyboard	MSX Compliant Matrix Keyboard
PCF	RTC	PCF8584-based I2C Real-Time Clock
PIO	Char	Zilog Parallel Interface Controller
PKD	DsKy	P8279-based Display/Keypad on PPI
PPA	Disk	PPA Zip Drive on PPI

ID	Type	Description
PPIDE	Disk	8255 IDE/ATA/ATAPI Hard Disk Interface
PPK	Keyboard	Matrix Keyboard
PPP	System	ParPortProp Host Interface Controller
PPPCON	Serial	ParPortProp Serial Console Interface
PPPSD	Disk	ParPortProp SD Card Interface
PRP	System	PropIO Host Interface Controller
PRPCON	Serial	PropIO Serial Console Interface
PRPSD	Disk	PropIO SD Card Interface
RF	Disk	RAM Floppy Disk Interface
RP5C01	RTC	Ricoh RPC01A Real-Time Clock w/ NVRAM
SCON	Char	S100 Console
SD	Disk	SD Card Interface
SIMRTC	RTC	SIMH Simulator Real-Time Clock
SIO	Char	Zilog Serial Port Interface
SN76489	Sound	SN76489 Programmable Sound Generator
SPK	Sound	Bit-bang Speaker
SYQ	Disk	Iomega SparQ Drive on PPI
TMS	Video	TMS9918/38/58 Video Display Controller
UART	Char	16C550 Family Serial Interface
USB-FIFO	Char	FT232H-based ECB USB FIFO
VDU	Video	MC6845 Family Video Display Controller
VGA	Video	HD6445CP4-based Video Display Controller
VRC	Video	VGARC Video Display Controller
YM	Audio	YM2612 Programmable Sound Generator
Z2U	Char	Zilog Z280 CPU Built-in Serial Ports