|  |
| --- |
| RomWBW User Guide |
| RomWBW Version 2.8 June 15, 2016 |
| RetroBrew Computing Group <http://www.retrobrewcomputers.org>  Wayne Warthen [wwarthen@gmail.com](mailto:wwarthen@gmail.com) |



Table of Contents

[Overview 1](#_Toc453532690)

[System Requirements 1](#_Toc453532691)

[Acknowledgements 3](#_Toc453532692)

[RomWBW Distribution Package 4](#_Toc453532693)

[Getting Started 5](#_Toc453532694)

# Overview

RomWBW is system software that support the Z80/Z180 based computing platforms produced by the RetroBrew Computing Group (see <http://www.retrobrewcomputers.org>). The goal of the project is to provide all firmware and software needed to make a fully functional computing platform. This includes both firmware (ROM) and software (disk images). The source code is provided and is licensed under GPL v3. A GitHub repository is used to maintain all source code and can be found at <http://www.github.com/RomWBW>.

Essentially all Z80/Z180 based hardware produced by RetroBrew Computing is fully supported by RomWBW. Much of the software was adapted from software produced by others in the community (see Acknowledgements) and is packaged within RomWBW to provide an integrated solution. A companion document (RomWBW System Guide) provides substantial detail on the architecture and internal operation of this software.

It is worth noting that this software is a perpetual work-in-progress. While it has become fairly stable and robust over time, it is undergoing constant updates to support new and revised hardware produced by the community. Backward compatibility between releases should not be assumed.

In order to provide a complete solution, the RomWBW package incorporates a hardware BIOS (hardware drivers) and selected operating systems and application software. All software is derived from the CP/M era of 8-bit computing. The operating systems included have been adapted to run under the RomWBW architecture. In general, application software has simply been included as originally distributed by the vendors and required no adaptation.

The RomWBW distribution package includes all the tools required to easily build the software from the source that is included in the package. The package includes a range of pre-built ROM and disk images. These are usually sufficient to get your hardware up and running simply by programming a ROM and optionally copying disk image(s) to a floppy disk, CF Card, or SD Card. If you wish to highly customize your system software, it is straightforward to modify the source code and build your own. At present, this requires Microsoft Windows XP or greater. All of the tools have counterparts for Linux, so building the software under Linux should be possible with a little effort.

# System Requirements

RomWBW is purely a software project. It assumes you have a fully functional hardware platform on which to host the software. A great deal of information on procuring and building the appropriate hardware is found on the RetroBrew Computing Wiki at <http://www.retrobrewcomputing.org>. Additionally, the RetroBrew Computing Group has a very active forum at <http://www.retrobrewcomputing.org/forum>. This forum is the ideal place to ask questions and get guidance for hardware and software. It is the primary forum for supporting RomWBW.

The starting point for a hardware platform that will appropriately host RomWBW software is one of the following CPU boards:

* **SBC v1 or v2**  
  This is the original Z80 CPU board produced by the community. It remains a very functional platform and is relatively easy to build. Note that v1 has a design deficiency that may or may not prevent the proper operation of RomWBW (bank switching does not always function reliably). The SBC CPU board features an ECB bus connector which allows it to be expanded with a backplane and peripheral boards.
* **Zeta v1 or v2**  
  The Zeta is very similar to the SBC board and is generally compatible with it. However, the Zeta platform is optimized to be a compact, standalone system. In addition to the features of the SBC, it includes an onboard floppy disk controller and the form factor of the board allows it to be mounted directly to a 3.5” floppy disk for a complete computing solution. It optionally supports a single daughter board that provides SD Card storage, VGA Monitor interface, and PS/2 keyboard interface. Although it does not have a bus interface, Zeta is powerful, compact, and fully featured. The Zeta v2 primarily adds enhanced bank switching and an interrupt controller which is not required by RomWBW.
* **N8**  
  The N8 is a very robust SBC. It is significantly larger than the SBC and incorporates a wide range of peripherals right on the one board (although it also supports expansion via ECB bus). The N8 is based on the Z180 CPU and incorporates interfaces for 2 serial ports, 2 parallel ports, IDE Hard Disk / CF Card, SD Card, sound synthesizer, video display, PS/2 keyboard & mouse interface, and floppy disk controller. This board is very powerful, but more challenging to build. It is not compatible with the SBC/Zeta – it implements a different bank switching mechanism.
* **Mark IV**  
  The Mark IV by John Coffman is similar to the SBC in that it shares the same form factor and ECB for expansion. However, it is substantially more powerful featuring a Z180 CPU and onboard CF Card and SD Card interfaces.

RomWBW fully supports all of the above boards as a starting point. For Zeta, the ParPortProp is supported as an option. The other platforms all support the ECB bus for adding optional peripheral support boards.

In addition to the hardware listed above, RomWBW also runs well on the Microsoft Windows based SIMH Altair Z80 simulator which allows you to try all of the RomWBW features without any actual hardware. The distribution package contains a copy of the simulator software for MS Windows, so it is very easy to use it (see Getting Started).

Note that RomWBW assumes specific board configuration settings. You must ensure that you set the jumpers/switches of each board as required by RomWBW (unless you modify RomWBW and produce a custom version that supports your specific board configurations). The standard board configuration settings are documented in Appendix A.

Note that RomWBW assumes there is 512KB of ROM and 512KB of RAM for all systems. It is fine if your system has more RAM or ROM than this, but it is problematic if you have less. It would be very rare for a system to have less that these amounts, but be aware of this constraint. These assumptions can be modified via customization later, but the pre-built software must have these minimums.

All of the host boards include a serial port. RomWBW will use this serial port for output when you start your system. By default, RomWBW uses 38,400 baud, 8 data bits, 1 stop bit, and no parity. You will need to connect the primary serial port of the host board to a terminal (or PC running terminal emulation software) to see the system output when you start RomWBW.

The use of the ECB bus signals is standardized such that any ECB add-on board can generally be combined with any of the ECB host boards to provide enhanced functionality. Appendix A provides an inventory of the boards supported by RomWBW along with relevant notes and required board configuration settings. Appendix A also includes a compatibility/support matrix between the host boards and the peripheral support boards.

# Acknowledgements

First, I want to be clear that RomWBW is not the only option available for system software on RetroBrew Computing Z80/Z180 hardware. While many similar projects are no longer active, they are very useful and may contain functionality that has not been incorporated in RomWBW. All of the software projects (including RomWBW) are listed in the RetroBrew Computing Wiki.

The UNA Project from John Coffman is the other currently active software project for the Z80/Z180 projects. It is far more advanced than RomWBW in that it can support all 4 host boards with a single ROM image and allows dynamic system configuration via onboard setup. It does not yet support the full range of hardware or video capabilities of RomWBW. Note that RomWBW supports an UNA “hybrid” configuration in which the UNA BIOS is combined with the RomWBW OS and application layers.

The RetroBrew Computing Group has existed in various forms since about 2010 (?). Many individuals have contributed to the community. The original founder of the community has moved on and requested anonymity going forward. However, his initiative is greatly appreciated. While there is no formal structure to the community, Andrew Bingham has taken the mantle of responsibility for the wiki and discussion group. This is a critical function and he deserves substantial credit for this effort.

Earlier in the community’s history, there were multiple branches of software development. Frequently, when a new board was produced, someone would create an independent code branch to support it. This started to lead to a very fragmented set of software that made it very difficult to create an integrated system with selected boards. RomWBW came about as an effort to create a framework that would allow arbitrary hardware to be easily added without creating entirely separate branches of code.

RomWBW essentially became a semi-structured place to incorporate all of the many software efforts of the community. Initially, most of the RomWBW codebase was simply a “cut and paste” of the software produced by others. Over time, much of this software has been repeatedly revised such that it is no longer similar to the original, but RomWBW owes its existence to the contributions of many other individuals. A few of those people are listed below and I apologize for anyone that I may have inadvertently omitted. I have intentionally omitted the original founder of the community based on my understanding of his desire to be anonymous going forward.

Douglas Goodall worked in very close collaboration with me during the first year of the RomWBW Project. He produced an excellent set of supporting utility programs and provided a great deal of design input. Regrettably, his utilities no longer have a caretaker and have become unusable as RomWBW has evolved, but their legacy continues within the current codebase. The source for all of these utilities is still available if anyone wants to take responsibility for bringing them back to current status.

John Coffman has personally produced a great deal of the hardware designs within the community. RomWBW contains many portions of code that John contributed over time. Additionally, he has been instrumental in providing advice and guidance to me for many years now.

Dan Werner has been one of the most prolific coders within the community. A great deal of his code was incorporated in the early RomWBW releases. David Giles produced some code that also provided a more integrated set of software for each host board. Over time, much of his code was incorporated in RomWBW. Likewise, Max Scane has produced code that ultimately wound up in RomWBW – specifically, he contributed the CLRDIR application.

It is my belief that all code incorporated into RomWBW has been done so with the express or implied permission of the original authors. I realize there have been many other individuals that have contributed to RomWBW and apologize for not naming all of them.

# RomWBW Distribution Package

RomWBW is distributed as a complete package (a .zip file) that contains everything appropriate for the different hardware variations. In other words, don’t look for a specific distribution for your hardware, you just want the current package. Within the package, you will find documentation, source code, build tools, and pre-built ROM and disk images.

The distribution package is usually hosted at the following locations:

* **RetroBrew Computing Wiki:**  
  Navigate to [https://www.retrobrewcomputers.org](https://www.retrobrewcomputers.org/doku.php?id=software:firmwareos:romwbw:start). Then, using the navigation menu on the left, choose software 🡪 firmwareos 🡪 romwbw to reach the RomWBW Project Page. At the bottom of the page you will find the distribution files listed for download.
* **GitHub:**  
  Navigate to <https://github.com/wwarthen/RomWBW> to reach the RomWBW Project on GitHub. Select “releases” to reach the list of distribution files. Note that you will see both Prereleases and Releases listed. Unless you specifically want to test work-in-progress, please download only a Release version.

The package should be named something like RomWBW-2.8-Package.zip. Using any standard personal computer (Windows, Linux, Mac, etc.), download and extract the contents of the zip file using any of the standard zip tools. You will see that there are several directories that are used to organize the contents. Don’t get overwhelmed. Initially, all you really care about is the Output directory (and possibly the Doc directory):

|  |  |
| --- | --- |
| Directory | Contents |
| Doc | Contains documentation files for various parts of the RomWBW distribution including operating systems, applications, and various aspects of RomWBW itself. In most cases, the name of the file should identify the component being documented. |
| Hardware | Files that are specific to certain hardware components. For example, it has the font ROM images for the video display boards. You do not need any of these files for the host boards used initially. Appendix A describes the contents of this directory for relevant boards. |
| Images | Files that are used to create disk images. Since the disk images are all pre-built, you do not need to worry about this directory until you want to create custom disk images (documented later). |
| Output | The ROM and Disk Images that you need to get started as documented below in Getting Started. |
| Source | The source code files that are compiled or assembled to create RomWBW. Again, the output is pre-built, so you don’t need to worry about this directory until you want to customize your system. |
| Tools | Windows-based applications that are used to build RomWBW. It also contains applications that you can use to copy disk images to floppy disks, CF Cards, SD Cards, etc. It also has the SIMH simulation software. |

In most cases, you will find a ReadMe.txt file in the directory which describes the contents of the directory in more detail.

# Getting Started

Because of the wide variety of hardware combinations, there is no “one case fits all” approach to getting started. The good news is that RomWBW operates very consistently regardless of the specific hardware. The operating systems, applications, and storage formats are all common. However, building and testing your hardware is entirely outside the scope of this document. The RetroBrew Computing Forum (<https://www.retrobrewcomputers.org/forum>) is probably the best place to get advice if you get stuck on hardware issues.

It is not necessary, but I highly recommend running RomWBW under the SIMH Simulator as a first step. This requires no hardware and will allow you to see how it should look when you use it on real hardware. Since the SIMH software is included in the distribution package, you can start it with a single command. Using a command prompt window, navigate to the high level directory of the distribution. Enter the command “sim” and the simulator should start up and you will see the RomWBW boot display similar to the one below.

ROM Image: 'Output\SBC\_simh.rom'

HDSK0: WARNING: Unsupported disk capacity, assuming HDSK type with capacity 34078KB.

HDSK0: WARNING: Forcing WRTLCK.

HDSK0: WARNING: Fixing geometry.

HDSK1: WARNING: Unsupported disk capacity, assuming HDSK type with capacity 34078KB.

HDSK1: WARNING: Forcing WRTLCK.

HDSK1: WARNING: Fixing geometry.

RetroBrew HBIOS v2.8.0-pre.5, 2016-06-05

SBC Z80 @ 20.000MHz ROM=512KB RAM=512KB

UART0: IO=0x68 8250 MODE=38400,8,N,1

SIMRTC: Tue 2016-06-07 19:55:51

MD: UNITS=2 ROMDISK=384KB RAMDISK=384KB

HDSK: UNITS=2

Unit Device Type Capacity/Mode

---------- ---------- ---------------- --------------------

Disk 0 MD1: RAM Disk 384KB,LBA

Disk 1 MD0: ROM Disk 384KB,LBA

Disk 2 HDSK0: Hard Disk 65MB,LBA

Disk 3 HDSK1: Hard Disk 65MB,LBA

Serial 0 UART0: RS-232 38400,8,N,1

SBC Z80 Boot Loader

Boot: (C)PM, (Z)System, (M)onitor,

(L)ist disks, or Disk Unit # ===>

In all cases, you will want to start with a Z80/Z180 host board. Any of the boards listed in System Requirements will work fine. I strongly recommend that you initially work on getting just the single host board running by itself – don’t even plug it into an ECB backplane.

Given a host board that is assembled and passes any hardware checks recommended by the boards designer, you need to make sure the board is configured for RomWBW. Refer to the entry in Appendix A for your board and confirm that all switches and jumpers on the board are set as required by RomWBW.

Your initial goal is to locate and program a ROM image for your host board. The ROM images are located in the Output directory. You are looking for the files that end in “.rom”. Don’t worry about all of the other variations at this point. Refer to the following table to determine the ROM image that you want:

|  |  |
| --- | --- |
| Host Board | Rom Image File |
| SBC v1 or v2 | SBC\_std.rom |
| Zeta v1 | ZETA\_std.rom |
| Zeta v2 | ZETA2\_std.rom |
| N8 (2511 datestamp | N8\_2511.rom |
| N8 (2312 datestamp) | N8\_2312.rom |
| Mark IV | MK4\_std.rom |

Locate the appropriate ROM image file in the Output directory based on the table above. You should see that the file is exactly 512KB in size. As indicated above in System Requirements, your system should have a ROM capacity of 512KB or greater. You need to program the file to your ROM using whatever tool you have. Programming a ROM chip is beyond the scope of this document, but any feel free to ask for help at the RetroBrew Computing Forum. The ROM image files are pure binary and should be programmed into the ROM chip starting at address 0 thru address 7FFFH. Insert the programmed ROM chip in your system.

Initially, you will need two external connections to your board. Power and serial port. All of the CPU boards provide an onboard power connection. Refer to the board designer’s notes on the RBC Wiki for more information on the power connection your board requires.

Finally, you must connect the primary serial port of your host board to a terminal using 38,400 baud, 8 data bits, 1 stop bit, and no parity. You can use either a dedicated terminal or use terminal emulation software on your PC (Tera Term works well and is free). When connecting to a standard PC serial port, a null modem cable is required. There is a good document on the Wiki that explains cabling of serial ports at http://???????.

Assuming you have a host board that is assembled and seems