System Development and Firmware Build

The N8VEM project community has put together a set of ROM chips which provide support for all of the available accessory cards available for the processor in the bus configuration. This firmware changes over time as new boards become available or more effective code is developed. On occasion folks may want to add their own code or develop other custom variations, which require modifications to the existing current version of the firmware source and the building of a new system ROM.

When I bought my processor card, I also purchased an AT27C080 EPROM chip from Andrew because I knew that I needed a version of the firmware with CP/M in order to debug and get an initial system put together. The debug monitor on this early ROM was an ideal starting place and allowed me to explore the memory space and test IO facilities. This worked out as expected, and provided the Monitor and CP/M operating systems, but did not support all of the available bus capabilities. Time to start pushing the envelope!

With a system running, the next assignment was to learn how to get the latest ROM burned and installed so that bus based cards could be utilized. My primary motivation was to run the PropIO card so that I could use a standard PC keyboard and VGA rather than tie up a PC in terminal emulator mode. There is a specific ROM for this configuration, and it has an associated set of code for the Propeller chip ROM.

Prom images with current support for a number of bus cards can be found in the ROMWBW directory (here). This effort requires sorting out the ROM versions and getting a prom programmer up and running. Appendix B lists the currently available versions of the ROM with their individual feature set and boards supported. The ROM Change Log, Appendix C provides some useful insight into what changes have been made between versions of the ROM code over time. Up to date versions of these documents can be found in the Wiki.

There is a site for firmware developers which contains the set of more recent support code organized by Wayne Warthen and others (here). Each of the versions has a folder and that folder contains the source code, the configuration files, and the tools used to assemble the code. There is also a .xxx file in each version folder so that one can burn a prom the first time around without actually changing anything and having to rebuild from original source.

There is another folder labeled SUPPORT, which contains code for the accessory card such that the card will correctly interact with the specific SBC firmware. In my case, this was the location of the ROM code for the PropIO card which would work with the ROMWBW version selected.

1: The ROMWBW file is a zip file which needs to be downloaded to a CP/M Project Code development folder and unzipped by double clicking the icon. In the unzipped files, there is a ROMList.txt file which is the original version of Appendix B. There is another ReadMe.txt file (Appendix A) which provides some insight into fundamental choices between CP/M and ZSDOS, building or not building custom versions of the ROM, and some information about terminal baud rates and system responsiveness included here as Appendix C.

2: For the first pass through the ROM code, I elected to use an existing ROM image with PropIO support, which requires a hardware CPU and CPU clock operating above 6.0 mhz (if you are stuck at 4 mhz, Xmodem version 5 is an option). There is a version of the ROM built for a 512 byte ROM (the 27C080 is an OTP EPROM/ROM) with the default baud rate of 38400 bps.

3: Go to folder FFFFFFF and get a copy of the correct ROM version binary. This is the file to move to the prom programmer.

With a current version of the base firmware burned and in place, it is time to try a ROM build from scratch, using the development tools contained in the ROM file space. Lets start with the tools and then code organization:

1. A reasonable place to start is the TASM Assembler which is licensed for Project member use. There are a number of copies of TASM in the Project Wiki, and any of these will run on a 32 or 64 bit Windows System. File TASM32 from the ROMWBW build directory has a full distribution set and can be moved to a development directory tree of your choice.

   TASM is a DOS project which needs to run from the Windows DOS shell. Go to Start> All Programs> Accessories > and select Command Prompt. This shell facility will give you a DOS Command Prompt in a DOS Window. Type in TASM enter after the prompt, and TASM

will display a signon and give you a help message showing command strings and line format.

Type in TASM -80 –h testz80.asm followed by an enter; and TASM will assemble the test program in Z80 mnemonics, generate list and hex files, and screen list any errors found. Options exist for alternative ways to build and format object and other output files. Type DIR to see all of the old and new files in the directory. NotePad is a suitable text editor, and other options abound.

2. The Assembler is in place and working. Next is the collecting and organizing the pieces of code to be assembled and organizing the configuration files to automate this new custom build. An exercise for the skeptical is to build Dump.asm, compare the historical and new Hex files, and download and run the CP/M application on the SBC host board using Xmodem.

   (here we need to describe how the default prom code for one of the configurations was built. This shows how the sets of source code are selected for inclusion and how the whole thing gets put together with conditional assembly).

3. (here we need to describe how the output of the assembler is transferred to the buffer memory of the prom programmer). Wellems probably. Verify buffer. Install Prom & verify.

APPENDIX A – System Choices Across ROM Images

```
************************************************************
***                      R o m W B W                     ***
***                                                      ***
***       System Software for N8VEM Z80 Projects         ***
************************************************************
```

Builders: Wayne Warthen (wwarthen@gmail.com)
Douglas Goodall (douglas_goodall@mac.com)
David Giles (vk5dg@internode.on.net)

Updated: 2013-09-28
Version: 2.5.2

This is an adaptation of CP/M-80 2.2 and ZSDOS/ZCPR targeting ROMs for all N8VEM Z80 hardware variations including SBC, Zeta, and N8.

NOTE: This is very much a work-in-progress. It is severely lacking appropriate documentation. I am happy to answer questions and provide support though.

Acknowledgements

While I have heavily modified much of the code, I want to acknowledge that much of this is derived or copied from the work of others in the N8VEM project including Andrew Lynch, Dan Werner, Max Scane, David Giles, John Coffman, and probably many others I am not clearly aware of.

I especially want to credit Douglas Goodall for contributing code, time, testing, and advice. He has created an entire suite of application programs that substantially enhance this ROM. Everything in the Apps folder of the distribution came directly from Douglas and the list includes cpmname, writesys, assign, slices, termtype, drives, and others.

David Giles has contributed support for building the ROM under Linux and the CSIO support in the SD Card driver.

Usage Instructions

The distribution includes many pre-built ROM images in the Output directory. The simplest way of using this ROM is to simply pick the pre-built ROM that most closely matches your preferences, burn it, and use it.

Refer to the file called RomList.txt for a complete list of the ROMs that are included and the required hardware configuration that they support.

Upgrading from Previous Versions

<TBD>

CPU Speed & Baud Rate

The startup serial port baud rate in all pre-built RomWBW variants is 38.4Kbps. While this speed is nice in that it provides great display and file transfer performance, it does push the limits of slower hardware. Specifically, XModem v12.5 (the default XM.COM) on the distribution is unable to service the serial port fast enough if the CPU is running at 4MHz. Your options are to 1) use the old version of XModem (XM5.COM), put a faster CPU oscillator in your system (6MHz or above), or 3) decrease the baud rate by building a custom ROM.

CP/M vs. ZSystem

There are two OS variants included in this distribution and you may choose which one you prefer to use.

The traditional Digital Research (DRI) CP/M code is the first choice. The ROM images that DO NOT end in "_z" are built with the traditional CP/M components from DRI. The Doc directory contains a manual for CP/M usage (cpm22-m.pdf). If you are new to the N8VEM systems, I would currently recommend using the CP/M ROMs to start with simply because they have gone through more testing and you are less likely to encounter problems.

The other choice is to use the most popular non-DRI

CP/M "clone" which is generally referred to as ZSystem. The ROM images with a "_z" suffix are built using the ZSystem components (specifically ZSDOS 1.2 and ZCPR 1.0). These are intended to be functionally equivalent to CP/M and should run all CP/M 2.2 code. They are optimized for the Z80 CPU (as opposed to 8080 for CP/M) and have some potentially useful improvements. Please refer to the Doc directory and look at the files for zsdos and zcpr (zsdos.pdf & zcpr.doc as well as ZSystem.txt).

ZSystem builds contain ZSDOS specific files in the ROM Disk.

Building a Custom ROM

I strongly suggest you start with burning one of the pre-built ROMs and making sure that works first. Once you have gotten past that hurdle, you should consider building a custom ROM. It is very easy and the distribution comes with everything that is needed to run a build on a Windows 32 bit or 64 bit system -- basically Windows XP or above. There is also a Linux build now available.

Creating a custom ROM allows you to customize a lot of useful stuff like adding support for a DSKY if you have one.

Please refer to the Build.txt file in the Doc directory for detailed instructions for building a custom ROM. If you are using Linux, also read the LinuxBuild.txt file.

Formatting Media

<TBD>

Creating Bootable Media

<TBD>

Using Slices on Mass Storage Devices

<TBD>

Managing Console I/O

<TBD>

Notes

I realize these instructions are very minimal. I am happy to answer questions. You will find the Google Group 'N8VEM' to be a great source of information as well.

APPENDIX B -- LIST OF PREBUILT ROMWBW CONFIGURATIONS AVAILABLE:

You should find the following ROM images in the Output directory. Refer to the descriptions below to select one that matches your hardware configuration, burn it, and use it.

Note there are two set of ROM builds below, one for generic Digital Research Inc (DRI) CP/M 2.2C (BDOS & CCP) and one for ZSystem (ZSDOS & ZCPR).

Note that all builds are now set for 512KB ROMs. The builds will work fine in 1MB ROMs. If you want to use the full 1MB ROM address space, just do a custom build.

Note that all builds are now set for 38.4Kbsp baud rate on the console with 8 data bits, no parity, and 1 stop bit. The baud rate can be changed in the config file if you want to do a custom build.

DRI CP/M (BDOS & CCP)

N8VEM_std.rom for N8VEM Z80 SBC V1/V2:
- 512KB ROM, 512KB RAM
- 38.4KB serial console baud rate
- Basic ROM/RAM disk (no floppy/IDE)
- Drives A:=ROM, B:=RAM

N8VEM_diskio.rom for N8VEM Z80 SBC V1/V2 + DISKIO:
- 512KB ROM, 512KB RAM
- 38.4KB serial console baud rate
- Basic ROM/RAM disk
- Floppy support via DISKIO
- IDE support via DISKIO
- Drives A:=ROM, B:=RAM, C:=FD0, D:=FD1, E:=IDE0-00, F:=IDE0-01, G:=IDE0-02, H:=IDE0-03

N8VEM_dide.rom for N8VEM Z80 SBC V1/V2 + DUAL IDE:
- 512KB ROM, 512KB RAM
- 38.4KB serial console baud rate
- Basic ROM/RAM disk
- Floppy support via DISKIO
- IDE support via DISKIO
- Drives A:=ROM, B:=RAM, C:=FD0, D:=FD1, E:=IDE0-00, F:=IDE0-01, G:=IDE0-02, H:=IDE0-03

N8VEM_diskio3.rom for N8VEM Z80 SBC V1/V2 + DISKIO3:
  - EXPERIMENTAL!
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk
  - Floppy support via DISKIO3
  - IDE support via DISKIO3
  - Drives A:=ROM, B:=RAM, C:=FD0, D:=FD1, E:=IDE0-00,
    F:=IDE0-01, G:=IDE0-02, H:=IDE0-03

N8VEM_ppide.rom for N8VEM Z80 SBC V1/V2 + PPIDE:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk
  - IDE support via DISKIO
  - Drives A:=ROM, B:=RAM, C:=PPIDE0-00, D:=PPIDE0-01,
    E:=PPIDE0-02, F:=PPIDE0-03

N8VEM_ppisd.rom for N8VEM Z80 SBC V1/V2 + PPISD:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk
  - PPISD support
  - Drives A:=ROM, B:=RAM, C:=SD0-00, D:=SD0-01, E:=SD0-02,
    F:=SD0-03

N8VEM_dsd.rom for N8VEM Z80 SBC V1/V2 + Dual SD:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk
  - Dual SD support
  - Drives A:=ROM, B:=RAM, C:=SD0-00, D:=SD0-01, E:=SD0-02,
    F:=SD0-03

N8VEM_propio.rom for N8VEM Z80 SBC V1/V2 + PROPIO:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk
  - SD Card support via PropIO
  - VGA console support via PropIO
  - PS/2 Keyboard support via PropIO
  - Drives A:=ROM, B:=RAM, C:=PRPSD0-00, D:=PRPSD0-01,
    E:=PRPSD0-02, F:=PRPSD0-03

  - WARNING: You must use the RomWBW specific firmware
    for the Propeller found in the Support directory!

  - NOTE: With PROPIO Console defaults to VGA & PS/2
    Keyboard. Short JP2 (one bit input port) to use the
    serial port as the console.

N8VEM_mfp.rom for N8VEM Z80 SBC V1/V2:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk (no floppy)
  - Drives A:=ROM, B:=RAM, C:=PPIDE0-00, D:=PPIDE0-01,
    E:=PPIDE0-02, F:=PPIDE0-03
  - IDE support via Multifunction / PIC
  - Second UART via Multifunction / PIC

N8VEM_ci.rom for N8VEM Z80 SBC V1/V2:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk (no floppy/IDE)
  - Drives A:=ROM, B:=RAM
  - Cassette Interface mapped to RDR/PUN

N8VEM_simh.rom for N8VEM SIMH Simulator:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk (no floppy/IDE)
  - Drives A:=ROM, B:=RAM, C:=HDSK0-00, D:=HDSK0-01,
    E:=HDSK0-02, F:=HDSK0-03

N8VEM_rf.rom for N8VEM Z80 SBC V1/2 + RAM Flopppy:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk (no floppy/IDE)
  - RAM Floppy support
  - Drives A:=ROM, B:=RAM, C:=RF0, D:=RF1

N8VEM_vdu.rom for N8VEM Z80 SBC V1/V2:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk
  - VDU board support
  - Drives A:=ROM, B:=RAM

ZETA_std.rom for Zeta Z80 SBC:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk
  - Floppy support via built-in FDC
  - PPIDE support via built-in PPI

- Drives A:=ROM, B:=RAM, C:=FD0, D:=FD1, E:=PPIDE00-0, F:=PPIDE0-01, G:=PPIDE0-02, H:=PPIDE0-03

ZETA_ppp.rom for Zeta Z80 SBC w/ ParPortProp:
- 512KB ROM, 512KB RAM
- 38.4KB serial console baud rate
- Basic ROM/RAM disk
- Floppy support via built-in FDC
- SD Card support via ParPortProp
- VGA console support via ParPortProp
- PS/2 Keyboard support via ParPortProp
- Drives A:=ROM, B:=RAM, C:=FD0, D:=FD1, E:=PPPSD0-00, F:=PPPSD0-01, F:=PPPSD0-02, G:=PPPSD0-03

- WARNING: You must use the RomWBW specific firmware for the Propeller found in the Support directory!

- NOTE: Console defaults to VGA & PS/2 Keyboard. Short JP1 (CONFIG) to use the serial port as the console.

N8_2511.rom for N8 2511 Z180:
- Assumes oscillator frequency of 18.432MHz
- CPU clock at X1 (18.432MHz)
- 512KB ROM, 1MB RAM
- 38.4KB serial console baud rate
- Basic ROM/RAM disk
- Floppy support via built-in FDC
- SD card support via built-in SD card slot
- Drives A:=ROM, B:=RAM, C:=FD0, D:=FD1, E:=SD0-00, F:=SD0-01, G:=SD0-02, H:=SD0-03

N8_2312.rom for N8 2312 Z180:
- Assumes oscillator frequency of 18.432MHz
- CPU clock at X1 (18.432MHz)
- 512KB ROM, 1MB RAM
- 38.4KB serial console baud rate
- Basic ROM/RAM disk
- Floppy support via built-in FDC
- SD card support via built-in SD card slot
- Drives A:=ROM, B:=RAM, C:=FD0, D:=FD1, E:=SD0-00, F:=SD0-01, G:=SD0-02, H:=SD0-03

ZSYSTEM (ZSDOS & ZCPR)

N8VEM_std_z.rom for N8VEM Z80 SBC V1/V2:
- 512KB ROM, 512KB RAM

- 38.4KB serial console baud rate
- Basic ROM/RAM disk (no floppy/IDE)
- Drives A:=RAM, B:=ROM

N8VEM_diskio_z.rom for N8VEM Z80 SBC V1/V2 + DISKIO:
- 512KB ROM, 512KB RAM
- 38.4KB serial console baud rate
- Basic ROM/RAM disk
- Floppy support via DISKIO
- IDE support via DISKIO
- Drives A:=ROM, B:=RAM, C:=FD0, D:=FD1, E:=IDE0-00, F:=IDE0-01, G:=IDE0-02, H:=IDE0-03

N8VEM_dide_z.rom for N8VEM Z80 SBC V1/V2 + DUAL IDE:
- 512KB ROM, 512KB RAM
- 38.4KB serial console baud rate
- Basic ROM/RAM disk
- Floppy support via DISKIO
- IDE support via DISKIO
- Drives A:=ROM, B:=RAM, C:=FD0, D:=FD1, E:=IDE0-00, F:=IDE0-01, G:=IDE0-02, H:=IDE0-03

N8VEM_diskio3_z.rom for N8VEM Z80 SBC V1/V2 + DISKIO3:
- EXPERIMENTAL!
- 512KB ROM, 512KB RAM
- 38.4KB serial console baud rate
- Basic ROM/RAM disk
- Floppy support via DISKIO3
- IDE support via DISKIO3
- Drives A:=ROM, B:=RAM, C:=FD0, D:=FD1, E:=IDE0-00, F:=IDE0-01, G:=IDE0-02, H:=IDE0-03

N8VEM_ppide_z.rom for N8VEM Z80 SBC V1/V2 + PPIDE:
- 512KB ROM, 512KB RAM
- 38.4KB serial console baud rate
- Basic ROM/RAM disk
- IDE support via DISKIO
- Drives A:=ROM, B:=RAM, C:=PPIDE0-00, D:=PPIDE0-01, E:=PPIDE0-02, F:=PPIDE0-03

N8VEM_ppisd_z.rom for N8VEM Z80 SBC V1/V2 + PPISD:
- 512KB ROM, 512KB RAM
- 38.4KB serial console baud rate
- Basic ROM/RAM disk
- PPISD support
- Drives A:=ROM, B:=RAM, C:=SD0-00, D:=SD0-01, E:=SD0-02, F:=SD0-03

N8VEM_dsd_z.rom for N8VEM Z80 SBC V1/V2 + Dual SD:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk
  - Dual SD support
  - Drives A:=ROM, B:=RAM, C:=SD0-00, D:=SD0-01, E:=SD0-02,
    F:=SD0-03

N8VEM_propio_z.rom for N8VEM Z80 SBC V1/V2 + PROPIO:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk
  - SD Card support via PropIO
  - VGA console support via PropIO
  - PS/2 Keyboard support via PropIO
  - Drives A:=ROM, B:=RAM, C:=PRPSD0-00, D:=PRPSD0-01,
    E:=PRPSD0-02, F:=PRPSD0-03

  - WARNING: You must use the RomWBW specific firmware
    for the Propeller found in the Support directory!

  - NOTE: Console defaults to VGA & PS/2 Keyboard. Short JP2
    (one bit input port) to use the serial port for console.

N8VEM_vdu_z.rom for N8VEM Z80 SBC V1/V2:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk
  - VDU board support
  - Drives A:=ROM, B:=RAM

N8VEM_mfp_z.rom for N8VEM Z80 SBC V1/V2:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk (no floppy)
  - Drives A:=ROM, B:=RAM, C:=PPIDE0-00, D:=PPIDE0-01,
    E:=PPIDE0-02, F:=PPIDE0-03
  - IDE support via Multifunction / PIC
  - Second UART via Multifunction / PIC

N8VEM_ci_z.rom for N8VEM Z80 SBC V1/V2:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk (no floppy/IDE)
  - Drives A:=ROM, B:=RAM
  - Cassette Interface mapped to RDR/PUN

N8VEM_simh_Z.rom for N8VEM SIMH Simulator:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk (no floppy/IDE)
  - Drives A:=ROM, B:=RAM, C:=HDSK0-00, D:=HDSK0-01,
    E:=HDSK0-02, F:=HDSK0-03

N8VEM_rf_z.rom for N8VEM Z80 SBC V1/2 + RAM Flopppy:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk (no floppy/IDE)
  - RAM Floppy support
  - Drives A:=ROM, B:=RAM, C:=RF0, D:=RF1

ZETA_std_z.rom for Zeta Z80 SBC:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk
  - Floppy support via built-in FDC
  - PPIDE support via built-in PPI
  - Drives A:=ROM, B:=RAM, C:=FD0, D:=FD1, E:=PPIDE0-00,
    F:=PPIDE0-01, G:=PPIDE0-02, H:=PPIDE0-03

ZETA_ppp_z.rom for Zeta Z80 SBC w/ ParPortProp:
  - 512KB ROM, 512KB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk
  - Floppy support via built-in FDC
  - SD Card support via ParPortProp
  - VGA console support via ParPortProp
  - PS/2 Keyboard support via ParPortProp
  - Drives A:=ROM, B:=RAM, C:=FD0, D:=FD1, E:=PPPSD0-00,
    F:=PPPSD0-01, F:=PPPSD0-02, G:=PPPSD0-03

  - WARNING: You must use the RomWBW specific firmware
    for the Propeller found in the Support directory!

  - NOTE: Console defaults to VGA & PS/2 Keyboard. Short JP1
    (CONFIG) to use the serial port as the console.

N8_2511_z.rom for N8 2511 Z180:
  - Assumes oscillator frequency of 18.432MHz
  - CPU clock at X1 (18.432MHz)
  - 512KB ROM, 1MB RAM
  - 38.4KB serial console baud rate
  - Basic ROM/RAM disk

- Floppy support via built-in FDC
- SD card support via built-in SD card slot
- Drives A:=ROM, B:=RAM, C:=FD0, D:=FD1, E:=SD0-00, F:=SD0-01, G:=SD0-02, H:=SD0-03

N8_2312_z.rom for N8 2312 Z180:
- Assumes UART oscillator frequency of 18.432MHz
- CPU clock at X1 (18.432MHz)
- 512KB ROM, 1MB RAM
- 38.4KB serial console baud rate
- Basic ROM/RAM disk
- Floppy support via built-in FDC
- SD card support via built-in SD card slot
- Drives A:=ROM, B:=RAM, C:=FD0, D:=FD1, E:=SD0-00, F:=SD0-01, G:=SD0-02, H:=SD0-03

APPENDIX C -- FIRMWARE CHANGE LOG

This is the change log for the ROMWBW code and provides a view of the features and facilities that have been developed and installed.

Version 2.5.2

- WBW: Added initial support for RAM Floppy

Version 2.5.1

- WBW: Fix PPIDE bug

Version 2.5

- WBW: Implemented CRT driver model (video and keyboard)
- WBW: Implemented ANSI terminal emulation
- WBW: Updated build process to remove CPU specification
- WBW: Refactored VDU and Color VDU drivers for new cRT driver model
- WBW: Refactored keyboard drivers for new CRT driver model
- WBW: Refactored PPIDE driver to improve performance
- WBW: Implemented "boot" messaging
- WBW: Fixed debug monitor (I, O, and H commands corrected)
- WBW: Added support for SCSI2IDE (SD Card driver primarily)
- WBW: Revised System Architecture document
- WBW: Added support for Dual SD board (preliminary)
- WBW: Overhaul of SD driver (sd.asm) to clean up the conditionals
- WBW: Completed Douglas' N8 video driver support
- WBW: ASCI driver has been separated from UART driver
- WBW: Added ZX CPM emulator and integrated with build process
- WBW: Updated Apps built with Aztec C to use the TINY library
- WBW: Updated CPMNAME application to reflect latest config data block
- WBW: Support multiple UART devices (up to 2 for now)
- WBW: Partial support for Multifunction / PIC (UART & PPIDE)
- WBW: Add chip detection to UART driver
- WBW: Move ram/rom disk code into separate driver (md.asm)

Version 2.1.1

- WBW: Corrected setup of Z180 wait states
- WBW: Added hd0-3 geometries to diskdefs file for cpmtools

Version 2.1

- WBW: Implemented write caching in (de)blocking algorithm
- WBW: Added Architecture documentation
- WBW: Config jumper controls default vs. alternate console for N8VEM/Zeta
- DGG: Added support for PPISD in SD Card driver
- WBW: Implemented screen saver in PropIO and ParPortProp (5 minute timeout)

Version 2.0

- WBW: Implemented Banked BIOS (drivers in separate bank)
- DGG: Updated in-situ flash utility for greater chip compatibility
- WBW: Updated FDTST to latest version (improved support for 5.25" and 8" media)
- WBW: Added ParPortProp driver
- DWG: Entire new suite of Apps written in Aztec C
- DWG: BANKER.COM - displays bank identification and version information
- DWG: CPMNAME.COM - displays CBIOS header data and SYSCFG data, names and vaues
- DWG: CHARS.COM - displays ascii map as reference
- DWG: CLS.COM - clears screen
- DWG: LABEL.COM - displays and changes drive labels for drives with reserved tracks
- DWG: MAP.COM - like old map command, displays drives and logical unit labels and changes LU values
- DWG: META.COM - like old metaview command displays and edits drive metadata
- DWG: REM.COM - used in submit files
- DWG: SYSGEN.COM - replaces old writesys command, much nicer, more flexible
- DWG: TERMTYPE.COM - like old termtype, displays and changes terminal type
- DWG: VIEW.COM - displays drive DPH and DPB, with addresses, 4-up
- WBW: Updated FDTST to v3.0 (support for sector interleave in format)
- DGG: Support for CSIO based SD access for N8
- DWG: Added DWG-APPS.MAN file to ROM describing command line syntax of new applications
- WBW: Prebuilt ROMs are now all 512KB -- works fine on a 1MB ROM
- WBW: Added driver for SIMH AltairZ80 hard disk (HDSK)
- WBW: Added support for SDHC/XC card to SD Card driver

- DWG: Extra Apps can be downloaded from Apps/apps-bins
  (limited to 100K in ROM)
- DWG: /XSource/Makefile is work in progress for Mac OS X build
  (experimental)
- WBW: Updated SIMH build for latest SIMH release v3.9

Version 1.5.2

- DGG: Added in-situ flash programming application
- WBW: Added support for 8" floppy drives
- WBW: Upgraded FDTST.COM to version 2.7a on ROM disks
- DWG: Minor fixes to METAVIEW, and MAP
- DGG: Fixes for makefile.linux

Version 1.5.1

- WBW: Added ZSDOS clock drivers (see Support\Clock)
- WBW: Overhaul of ZSystem ROM Disk (see Doc\ZSystem.txt)
- WBW: Update PropIO ANSI emulation for compatiblity with
  ASSIGN
- DWG: Added version tags to all applications, and IDENT
  program to
  check version of utilities.
- DWG: Added MULTIFMT program which prepares new media for use
  by
  initializing the metadata and clearing the directory
  sectors of
  all logical units on a specific drive
  (IDE,PPIDE,PRPSD,SD).
- DWG: Enhanced MAP program combines the functionality of
  DRIVES,
  SLICES, and MAP. DRIVES and SLICES have been removed.
- DWG: ANALYSE and HELLO programs removed from ROM due space
  concerns
- DWG: Additional macro librarties added supporting program
  identification
  (IDENTITY.LIB/ASM) and access to drive metadata
  (METADATA.LIB/ASM),
  and realtime selection of logical units from within new
  application
  programs (LOGICALS.LIB/ASM).
- DWG: Added TERM_VT52 for VDU compatbility, all apps now
  compliant
- DGG: Contributed Linux build (see Doc\BuildLinux.txt)

Version 1.5

- WBW: Upgraded XModem to version 12.5
- WBW: Added support for PropIO (RomWBW specific firmware
  required on PropIO)
- WBW: Corrected RTC application for N8 (it now works)
- WBW: Included updated FDTST v2.6 w/ support for 5.25" floppy
  drives
- WBW: Added OS support for 5.25" drives
- DWG: New Apps ACCESS, ANALYSE,
  FINDFILE,HELLO,METAVIEW,NOACCESS,RTC2012
- DWG: RMAC macro files re-written as hybrid libraries making
  executables smaller and faster
- DWG: Loader displays logical unit label with other stats
- DWG: CPMNAME enhanced to support new PROP I/O SD
- DWG: Much more inline doc in Apps source code
- DWG: Add ACCESS to verify file present in submit
- DWG: Add ANALYSE as sample program demonstrating macro usage
- DWG: Add FINDFILE to locate file(s) in any Logical Unit
  (slice)
- DWG: Add HELLO as clasic hello world sample
- DWG: Add METAVIEW to display and manage file system metadata
- DWG: Add NOACCESS to verify file not present in submit

Version 1.4

- DWG: Add various .SUB files used for application maintenance
- DWG: Enhanced utility building .SUB files to only contain
  libs utilitized
- DWG: Add BUILD.SUB to build all applications and DEVFILES.LBR
- DWG: Add/update RMAC macro libraries used in Apps -
- DWG: BIOSHDR, STDLIB, STRCPY, STRLEN, CPMBIOS, CPMBDOS,
  TERMINAL, HARDWARE,
- DWG: CPMAPPL, GLOBALS, ATOI, LUBIND, APPLVERS,
  MEMORY(memcpy,memset), PORTAB
- DWG: Add/Repair BIOS support for Boot Drive login during CP/M
  Coldstart
- DWG: All Apps utilities now licensed with GNU Public License
  v3
- DWG: DRIVES utility now dispays labels for drives with
  reserved track(s)
- DWG: DEVFILES.LBR now include just .ASM, .LIB, and .SUB files
- DWG: Updated CPMNAME for latest config changes, added paging
- DWG: Add REM utility for use in SUBMIT files
- DWG: Add STOP utility to terminate execution of SUBMIT files
- DWG: Add PAUSE utility to pause the execution of SUBMIT files
- DWG: Add REQ1PARM utility to verify a parameter was specified
- DWG: Add HEADER utility to display addresses of BIOS header
  data items

- DWG: Add command line MAP utility "map A: 23" for use general use and in SUBMIT files
- DWG: Retired PPIDELUx utilities in favor of new MAP utility
- DWG: Add SLICES utility to display labels of all slices on current drive 3/line, formatted
- DWG: Add LABEL utility to insert label into drive/slice metadata
- DWG: Add 16 char label field to metadata
- DWG: ASSIGN utility displays and manipulates DPH/DPB & logical unit parameters
- DWG/WBW: Collaborated on design of Logical Unit DPH enhancemnt
- WBW: Proposed MAP utility functionality
- WBW: Implement slice selection API for DSK devices
- WBW: Record boot drive in config memory at load time
- WBW: Add DSKY_KBD flag to util.asm so that keyboard routines can be built only when needed to save space in CBIOS
- WBW: Support 16550 UART FIFO (selectively via config, enabled where available)
- WBW: Remove B: default from xmodem (default to current drive)
- WBW: Consolidate xmodem code variations using conditionals
- WBW: Add xmodem variation for N8 ASCI1 (N8 now has XM0 & XM1 instead of XM)
- WBW: Remove CCP extension that searches USER 0 area for executables
- WBW: Reset drives when exitting FDTST (media format may have changed)
- WBW: Switch from VDE to ZDE
- WBW: Added signature to system image prefix
- WBW: Modified SD card disk layout for consistency with other media (existing sd cards need reformatting!)
- WBW: Upgraded ZSDOS from v1.1 to v1.2
- WBW: Modified build so that separate ('_z') config files are no longer needed

Version 1.3.3

- WBW: Changed startup banner for ZSystem builds
- WBW: Modified XM for ZSystem builds to default to current drive
- WBW: Included zsdos.lbr in Support directory

Version 1.3.2

- WBW: ZSDOS/ZDDOS support added
- WBW: ZCPR support added

Version 1.3.1

- WBW: Updated FDTST application to handle faster (20MHz) systems, slower is OK
- WBW: Small fix to SD card driver to handle card init failure in rare situations
- DWG: Updated WRITESYS to improve SELDSK BIOS call compatibility

Version 1.3

- DWG: WRITEIMG renamed to WRITESYS, works on PPIDE, CPM.SYS added to ROM
- DWG: TERMTYPE gets and sets terminal type
- DWG: PPIDELUX programs will dynamically select storage "slice" on device
- DWG: DRIVES utility will show current drive mappings
- DWG: CPMNAME utility enhanced to include all new config settings
- DWG: DEVFILES Douglas Goodall's Development Environment added to Apps
- DWG: Configured Wordstar and front end utility added to Apps
- DWG: Added drive mapping display to loader
- DWG: Added "Logical Unit" support for PPIDE (allows full use of media)
- DWG: Designed Application Package format and Protocol
- WBW: Implemented IOBYTE and character device driver interface abstraction layer.
- WBW: Mapped VDU to CRT: device (N8VEM SBC w/ VDU hardware only)
- WBW: Implemented second UART for N8 as UC1: device
- WBW: Implemented SD driver (N8 only)
- WBW: Implemented hot swap for SD driver (N8 only)
- WBW: Added DSKY display for SD driver (N8 only)
- WBW: Corrected keymap in VDU driver (N8VEM SBC w/ VDU hardware only)
- WBW: Removed filler and allocated space to rom extension area
- WBW: Added new standard build configurations for N8 (fd, sd, ppide)
- WBW: Implement DBGCON selection (UART or VDU)
- WBW: Implement LDRCON selection (UART of VDU)
- WBW: Made ROM size selection part of build command

- WBW: Made processor selection (Z80 vs. Z180) part of build command

Version 1.2

- DWG: Updated CPMNAME and WRITEIMG utilities
- WBW: Added N8 support (minimal, based on work by David Giles, but not as robust as N8RomDG).
- WBW: Added support for VDU board (code from Andrew Lynch, Dan Werner and James Moxham)
- WBW: Boot loader configurable for auto-selection w/ timeout (as requested by Bob Devries)

Older Stuff

This work is all derived from JC110508. Note that JC110508 included
the fix for DPB in CBIOS for large ROM drive. Specifically, EXM_5
was changed from 1 -> 0 (as it should have been).

1) Fixed the stack location in loader-b.asm. LOADER.COM was not working
for me without this change.

2) Fixed the size of the ccp+bdos+cbios in bloader.asm and loader-b.asm.
It was too small before and would be a potential problem depending on
how many of the optional features were enabled.

3) Fixed the "MON" command in cpm. It was jumping to an old/bad location.
I modified the way it works a bit to handle the situation where it can
become overlaid. There is now a routine in cbios.asm to reload it and
branch to the warm start location. The address of the cbios routine is
now saved in the cbios scratch area of zero page (40H) so that ccp does
not need to have a hard coded location in it.

4) Corrected ROM memory layout a bit. Small ROM drive was starting at the

wrong location (4800H). It should have been 5000H. So, I gave the loader
2K more space. Required modifications to bloader and cbios. Renamed
'romdsk.dsk' to 'romdsk.dat' to help alert folks to the fact that this is
a different image and allows "clean" to remove anything with a suffix
of '.img'.

5) Changed bloader to allow selection of DSKY or UART monitor using JP2.
A single ROM can now be dynamically configured to start via DSKY or UART.

6) Fixed small ROM disk DPB in cbios. Changed DSM_6 from 31 -> 11. Based
on blocksize of 1024 and the fact that it is now a 12K area, 11 is the
correct value. Also fixed EXM_6 from 1 -> 0 (per DRI spec).

7) Moved startup message from warm boot to cold boot. Also moved VDU init
to cold boot -- seems appropriate, but I have no way to test this yet...

8) More DPB tweaks. I have reviewed the floppy and RAM/ROM DPB's. I have
not looked at ATAPI or IDE. May still be problems in those.

9) Complete overhaul of build scripts. Reworked makefile to be compatible
with the make utility from gcc and created MakeRom.cmd to invoke the gcc
make utility. Created a PowerShell script as an alternative way to create
ROM (BuildRom.ps1) and created BuildRom.cmd to invoke it. Final ROM
image is now called rom.img -- made more sense to me, but I have no
compelling justification for changing that. BuildRom2.bat is still
there and I think it works.

10) Played with the startup message in cbios a bit. Extracted the "build"

id and moved it to the top of the file. My intent is to make it easy to update. Ultimately, I wold prefer that it be updated with the current date of the build as part of the build scripts, but still debating how best to accomplish that.

11) Related to #10, I have extracted the 3rd party build tools into sibling directories. So, for example, tasm is now found at tasm32. This makes it very easy to update the 3rd party tools and to clearly differentiate the 3rd party tool files. All build script have been updated as needed.

12) Removed ALL enable interrupt (EI) instructions in CBIOS. By leaving interrupts disabled the BIOS will now start OK even if some vagrant hardware is asserting an interrupt (DISKIO). Seems like this is better anyway -- general idea is that we only enable interupts precisely when desired for very specific controled purposes since there is no concept of interrupt dispatching available.

T