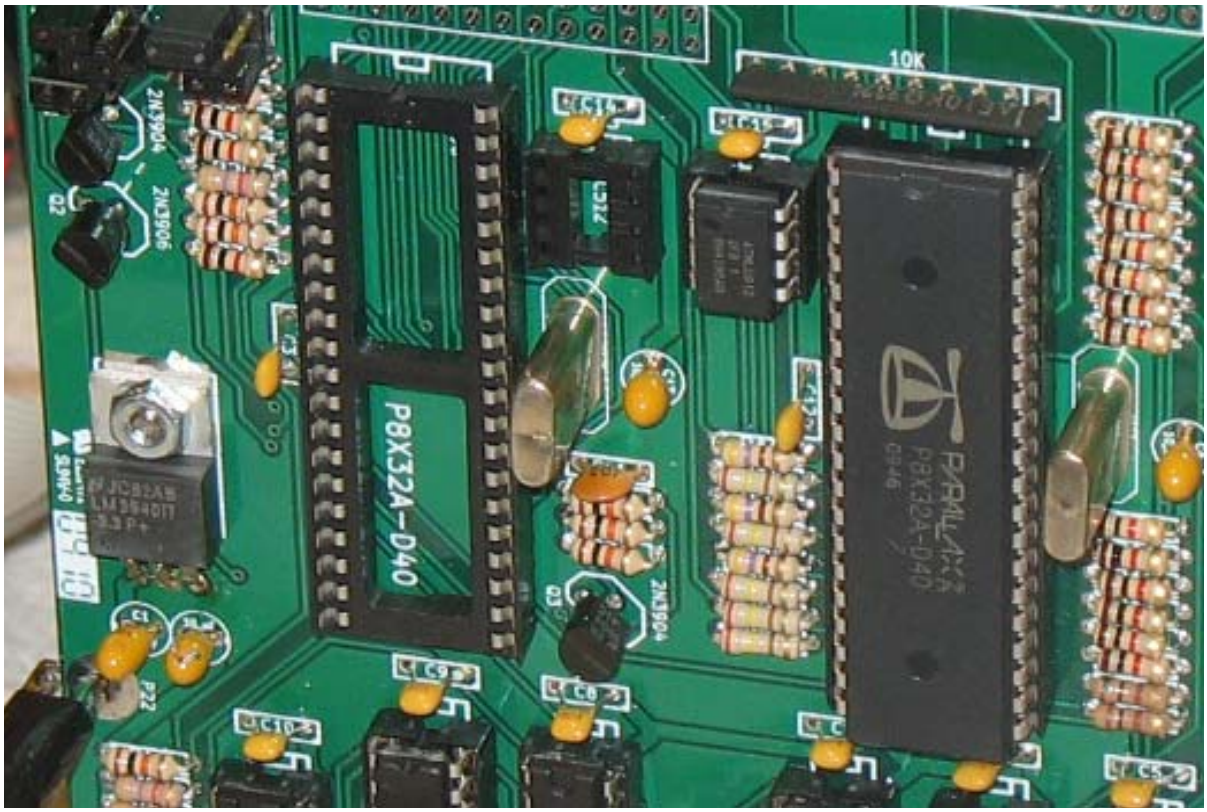
**PROPIO - VGA Video and Keyboard Card with SD Disk:**

The PropIO card is interesting because it provides a way to avoid allocating a PC as a separate terminal device. The PS/2 keyboard becomes the input and the VGA driver local to the propeller chip becomes the video output device. The bonus is the micro SD card on the board which provides a disk for handling larger files than the RAM disk built into the SBC memory domain.

There are, however, a few challenges that come up when one gets around to actually building and booting the PropIO card. The keyboard and VGA video plugs are something of a challenge to find as the PS/2 keyboard and VGA display are considered obsolete by many, although there are a lot of these devices currently in use. The SD connector comes in a variety of shapes and sizes, but in this case there is a part number and it is available, or there is a pre-configured Spark Fun substitute if you prefer.

The next issue is to find a few of the resistor values called for on the BOM. The 240 ohm resistors (R19, R21, and R23) are part of the analog video-out lines supporting the Red, Green, and Blue color signals. These resistors are not commonly found, with the closest substitutes being 220 and 270 ohms. These resistors are connected in series with commonly available 470 ohm resistors in a divider network aimed at providing four analog voltage levels for color intensity (0, 0.35. 0.68, and 1.0 volts DC). If you work out the divider resistances including the 75 ohm input impedance of the monitor it will become clear that 220 ohm resistors will work. The same is true for the vertical and horizontal sync current limit resistors (R24 and R25). Another strategy is to buy larger tolerance resistors (10% rather than 5% or 1% types) and selecting the values closest to 240 ohms with an ohm meter.

There is another hardware issue noted in the documentation in case you plan to use the serial circuit for booting the propeller. The silk screen layer component outline is incorrect for transistors Q1 and Q3. Thus, it is important to compare the PCB traces on the card with the schematic and put the emitter and collector leads in the correct holes rather than use the outline orientation for pin placement. Look at the orientation of these transistors in this picture:




The Propeller processor chip has a firmware EEPROM (24C512) which stores the operating code for the processor set supporting the VGA and keyboard operation. This code is downloaded to the Propeller when the card powers up. The code in the prom therefore has to be loaded prior to first use of the card. The code to be loaded can be found in the Support folder of the ROMWBW directory; and is the only firmware that will properly interface with the CP/M Bios code in PRP.asm. The procedure is as follows:

1. Open the ROMWBW directory, unzip the files, and find the folder marked SUPPORT. Under SUPPORT you will find the folder marked PROPIO. Under PROPIO, there is a file named PropIO.eeprom, which is the code that goes in the 24C512 boot EEprom. There is also a folder marked SPIN which is the set of executable programs necessary for the Prop P8X32A to handle the keyboard input and the VGA output driver facilities; the source code for the EEprom code.

2. The Parallax website has a propeller development tool called Propeller Tool v1.3.2. This is a firmware development environment, which supports

developing code or downloading RAM/ROM code. Down load this file to your computer and run the installer. Appendix 1 briefly describes the install and use of this tool for those who are working with these tools for the first time.

At start-up, the Propeller chip looks for a reset (low to high) on pin 11, and then attempts to communicate with a terminal device via pins 39 and 40 through the serial port built around Q1-Q3. Alternatively the parallel Prop Plug (a USB to 3.3 TTL and protocol converter) can be employed using the same chip serial IO scheme. If serial input is not available (operating rather than in a boot mode), the chip loads the 24C512 EEprom code into Cog0 along with the SPIN interpreter and executes the code from the ROM.

3. When it finds the Programming Plug, the propeller and PC PropToolv1.3.2 work together to load the PropIO.eeprom code down to the propeller which in turn writes it to the EEprom, where it can be retrieved for the next operating cycle. This pass-through-write has to be accomplished once before the card will operate from power-up.

4. Note that the PropIO.eeprom is the pre-built code segment aimed at operational use, it does not include the Parallax Serial Terminal facility which is only built when debug is the conditional assembly choice. If implemented in a custom build, this terminal runs at 9600 baud to a terminal or the development tool.

5. The entire VGA Terminal facility is contained in the Propeller and the EEprom. The rest of the chips are associated with bus access to the Prop terminal. These can be inserted and tested separately when the backplane is in use.

6. Plug in the Prop Plug and the computer will tell you it has found new hardware, followed by finding the driver pre-loaded above. The Prop Plug has pin identification on the back side, and the VSS goes to the Prop Plug ground which is the lead outboard and away from the video plug. Look

for a red LED flash when the cpu and the plug acknowledge each other.

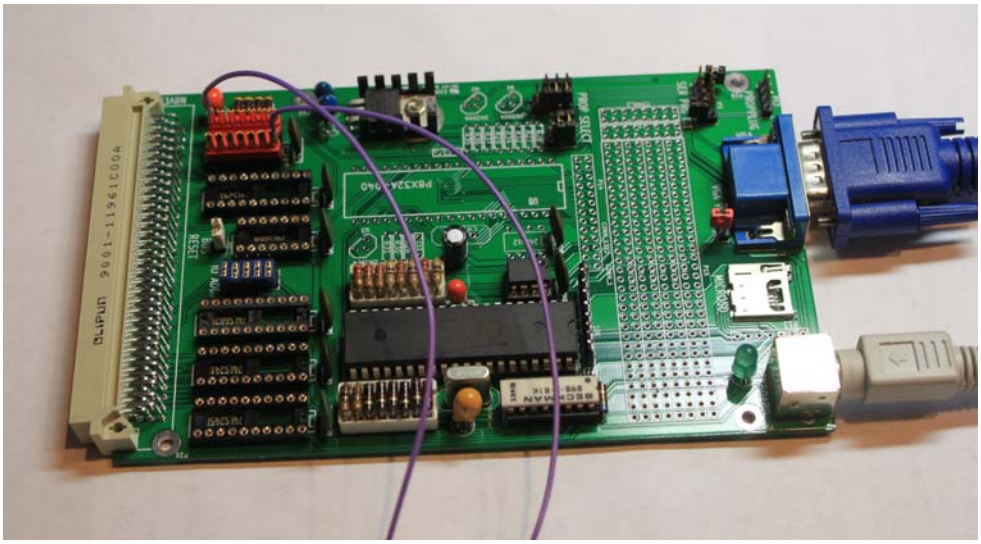
7. Using the Prop Tool, open the PropIO.eeprom file (file/open/PropIO.eeprom) and it comes up in the Object Window. Select Load RAM, and it will load the file to RAM and then to Prom, with a host of red and blue LED flashes on the Prop Plug. When both loads are complete, the program will execute and the board will sign-on to the VGA monitor. The keyboard still has to echo through the CP/M system which is not yet connected so it will pass init OK, but not work to the screen.

8. The monitor will write the Startup and Initialize messages you found in the PripIO.spin file, along with the "OK" feedback messages telling you that all of the init details were completed OK. The last message is the "PropIO Ready!" Message. The version 0.92 prints in the lower right corner of the screen on the blue line. Below is a picture of the VGA screen (with some tilt to minimize the camera flash) as it initializes:


9. The last step is to place both the N8VEM CPU and the PropIO in the backplane sockets and power up the pair with CPU JP2 removed. The VGA will run

the sign-on messages and then work with the CPU as the CP/M terminal serial port.

10. Below is a picture of the assembled PropIO with a red 5 volt power socket installed because my VGA did not source 5 volts through the red JP1 jumper.



Note that the bus interface chips were not in place for the initial confirmation of the VGA terminal. Correct voltages were confirmed on all sockets before any chips or peripheral devices were plugged into the board.

Jumper JP1 is open to prevent sourcing 5 volts into the VGA. I had intended to use the 5 volts from the VGA to power the board for the first terminal test and EEprom download, but this older display was not destined to cooperate.

Before launching into the Write the Prom Exercise, lets take a look at the files that will be down-loaded through the prop chip to the EEPROM.

1. Go back to the ROMWBW folder and drill down to the SUPPORT folder and find the folder called PropIO. Inside, is the PropIO.eeprom file, which is the pre-built production code binary for the PropIO card 24C512 EEprom.

2. The propeller runs its applications from SPIN code using its built-in SPIN Interpreter and a small amount of assembly code. Start up the Prop Tool (aka Propeller Tool v1.3.2), and use it to open PropIO.spin, which is the starting point for all of the Spin applications. This file will set up Keyboard.spin to operate the PS2 keyboard, Safe_spi.spin to support the SD disk, and VGA_1024.spin to initialize the VGA and pass operating control to VGA_hires_txt.spin).

3. There is a Parallax Serial Terminal tool which can be built into the application binary which provides a terminal window into the details of the propeller execution of the spin code. This piece of diagnostic code is not built into the ROMWBW production binary.

4. For the inveterately curious, the files PRP.asm and Prp_dph.asm are the code and file descriptors for the CP/M bios end of the bios driver to terminal firmware conversation. These files must be matched with the provided PropIO SPIN firmware in order to obtain proper operation of the card.

5. After poking around in the PropIO support code on the card and in the Bios for a while, one just starts to appreciate how massive an undertaking ROMWBW really is; a matrix of multiple SBCs with unique bios code versus a host of function specific support cards with individual interface requirements, all structured in one huge complex conditional assembly environment. Some of the support cards use non-Z80 processors and

languages which have to be understood in order to bring these interfaces together. Wow!

Time for the Load the EEprom Exercise. The Prop Plug cable from Parallax is the connection (USB on one end a 4 pin connector on the other with a USB to 3.3 TTL converter) between the code development tool and the hardware:

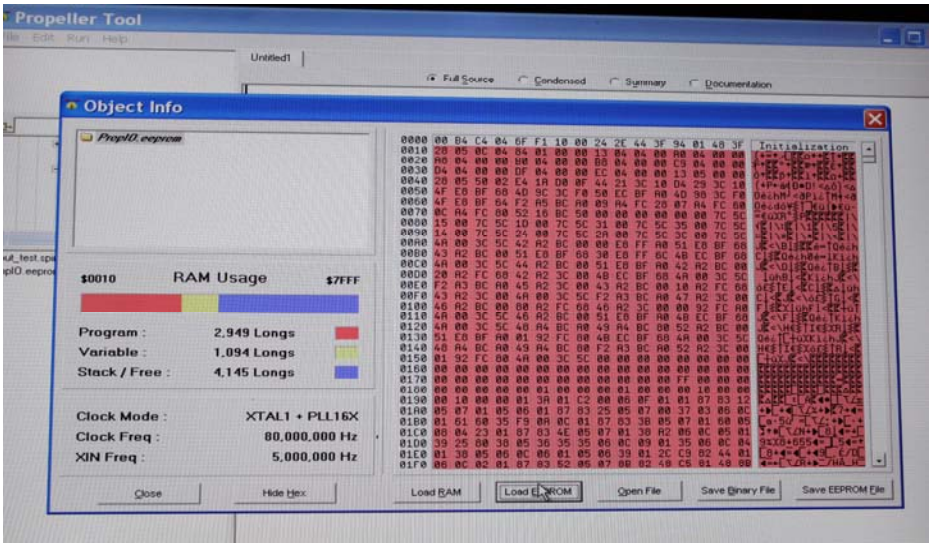
1. The Prop Plug needs to be connected between the computer USB port and the connector on the hardware board; orientation can be confirmed by locating the ground pin which is pin 1 on the 4 terminal connector.

2. The target board needs to have power applied but without chips in order to confirm proper power levels on all chips with a volt meter. First check the LM3940 regulator for 5.0 volts and 3.3 volts, and then check all the other chips with the help of the schematic.

3. With correct power verified, the power can be removed and the Propeller and EEprom memory chip placed in their respective sockets (3.3 volt level). Bus buffer chips are not needed for prop test or loading the prom code so they can be deferred to the backplane testing. All of the buffer chips run from the 5.0 volt supply.

4. Using the Propeller Tool, open the ROMWBW version of PropIO.eeprom file. The Tool will bring up an Object Info window and show a color coded representation of the memory span from 0 to 32K in Hex format. The right hand side of the display shows ASCII when found and the snipits that appear confirm the identity of the file that we loaded.

   The screen messages that will paint on the VGA screen can be found in the code on the ASCII side of the screen.




5. Read across the bottom of the display, and there are options to load this code to the ROM memory and the Prop internal RAM memory. Push the Load EEPROM button first to get the external ROM written. The Tool will report errors or success. After a successful load, a reset pulse to the card without the PropPlug in place will cause the PROM to be loaded, executed, and the startup sequence to be announced on the VGA screen.

6. The load strategy will be to power the board from the VGA with jumper JP1 shorted. This will supply power to the 24C512 EEProm and the Propeller via the 3.3 volt regulator, which is all we need to write the EEProm.

   Once the Prom is written, it should be feasible to reset the board without the Prop Plug and the terminal should boot up from the Prom and all of the start-up messages should be evident. Only the Propeller, the VGA, and the keyboard are required to run the terminal sans the backplane and external bus buffers.

   With the VGA Terminal system working, the external bus address must be set to the 40H range. All of the remaining bus interface chips must be installed on the PropIO. CPU jumper JP2 must be removed, and then the

PropIO should provide VGA terminal services over the external backplane and bus.

The PropIO SD disk facility was booted to the EEprom in the VGA exercise, and will be tested later with separate support documentation.  This testing will confirm all of the bus connections and buffers.

# APPENDIX 1 – Parallax Tool Set

Journey to the Parallax Inc. website (http://www.parallax.com) to find the propeller information set. The web page is probably not exactly what you expected, but if you select microcontrollers and then propeller, you will get a page with the heading **Propeller The Multicore Propeller Microcontroller** with a picture of the three chip packages that are available.

Half way down the page there is a heading for **Propeller Tools**. In that space is the description of the software development environment: "The Propeller Tool Software is the primary development environment for Propeller programming in Spin and Assembly Language. It includes many features to facilitate organized development of object-based applications: multi-file editing, code and document comments, color-coded blocks, keyword highlighting, and multiple window and monitor support aid in rapid code development. Optional view modes allow you to quickly drill down to the information you need—by hiding comment lines, method bodies, or by showing the object's compiled documentation only. Example objects, such as keyboard, mouse, and graphics drivers, come standard with the free Propeller Tool software."

Double click on the Propeller Tool Software vector, and it will take you to another page where you can download the tool in zip format:

| | | |
|---|---|---|
| P8X32A-Setup-Propeller-Tool-v1.3.2.zip | 19.69 MB | Mon, 2012-10-08 15:42 |

Double click on the tool line and you will get a down-load window asking if you would like to save this file. I tend to store these installables in the Program file for disk C so I always have the original files handy but separate, while I actually install the tool in a directory structure that makes sense for the development of the project code. Note that there is also an older version Propeller Tool v1.2 which works quite well (and the one I started with).

The installer asks you to register by name and organization (ORG is not required) and does the dirty work. You must accept the 2 drivers that support the USB interfaces. Create a shortcut for your desktop and you are ready to go.

When you first start **Propeller Tool 1.3.2**, it may tell you that it has found a file type that does not fit into its expected file types (it expects .spin and .eeprom, but apparently does not recognize .binary). I chose Cancel to ignore the .binary type and have not yet been slapped down for carelessness. The Propeller Tool then comes up and you can select to OPEN the Propeller SPIN codes in the SUPPORT directory of ROMWBW.

There is a slightly more "rustic" propeller development tool available that is command line oriented: "The Parallax Propellent software is a Windows-based tool for compiling and downloading to the Parallax Propeller chip without using the Propeller Tool development software. The Propellent Executable provides the ability to do things like compile Spin source, save it as a binary or EEPROM image, identify a connected Propeller chip, and download to the Propeller chip, all via simple command-line switches or drag-and-drop operations." As always, Beauty is in the eye of the beholder.

| | | |
|---|---|---|
| P8X32A-Propellent-v1.6.zip | 738.73 KB | Thu, 2013-04-04 10:18 |

It is also worthwhile to note that there is a Parallax Propeller Library Exchange which is a collection of applications from the public domain which are available without warranty or support. These applications are good examples of how things could (or should not?) be done.

There is an Open Propeller Project, not associated with Parallax, which is hosted by Ken Gracey which provides some forum support and example applications and code.

And last, but not least, there is an Open source GCC Project with language, tool, and example application support: "The Propeller GCC Compiler tool-chain is an open-source, multi-OS, and multi-lingual compiler that targets the Parallax Propeller's unique multi-core architecture. Parallax has collaborated with industry experts to develop all aspects of the tool chain, including the creation of a new development environment that simplifies writing code, compilation, and downloading to a Propeller board. Using

the Large Memory Model (LMM) and Extended Memory Model (XMM) gives the developer the ability to write C or C++ programs that run faster than Spin or exceed Spin's 32 KB program size limit, respectively. Example objects, including C objects, are available through the Propeller Object Exchange."

And just when you thought we were done, there are the USB drivers for the Parallax Prop Plug. These can be found under the USB DRIVERS area of the standard Parallax Website (WWW.Parallax.com/usbdrivers). These need to be downloaded before the hardware is plugged into the USB port so that the computer will be able to find the right drivers for the new hardware the computer system has found. There is a zip with all the stuff you need, and the Wizard executable which will do the hard work:

| | | |
|---|---|---|
| Install-Parallax-USB-Drivers-v2.08.24.exe | 2.43 MB | Fri, 2012-09-28 15:55 |
| Install-Parallax-USB-Drivers-v2.08.24.zip | 1.71 MB | Fri, 2012-09-28 15:55 |

# APPENDIX 2 PROPELLER STARTUP PROCESS

Upon power-up (+ 100 ms), RESn low-to-high, or software reset:

1. The Propeller chip starts its internal clock in slow mode (≈ 20 kHz), delays for 50 ms (reset delay), switches the internal clock to fast mode (≈ 12 MHz), and then loads and runs the built-in Boot Loader program in the first processor (Cog 0).

2. The Boot Loader performs one or more of the following tasks, in order:

   a. Detects communication from a host, such as a PC, on pins P30 and P31. If communication from a host is detected, the Boot Loader converses with the host to identify the Propeller chip and possibly download a program into Main RAM and optionally into an external 32 KB EEPROM.

   b. If no host communication was detected, the Boot Loader looks for an external 32 KB EEPROM (24LC256) on pins P28 and P29. If an EEPROM is detected, the entire 32 KB data image is loaded into the Propeller chip's Main RAM.

   c. If no EEPROM was detected, the boot loader stops, Cog 0 is terminated, the Propeller chip goes into shutdown mode, and all I/O pins are set to inputs (high impedance).

3. If either step 2a or 2b was successful in loading a program into the Main RAM, and a suspend command was not given by the host, then Cog 0 is reloaded with the built-in Spin Interpreter and the user code is executed from Main RAM.

**** The source for the above startup procedure is the Parallax Propeller chip datasheet documentation.