11.0 Game Cartridge, EEPROM and Expansion Port Hardware

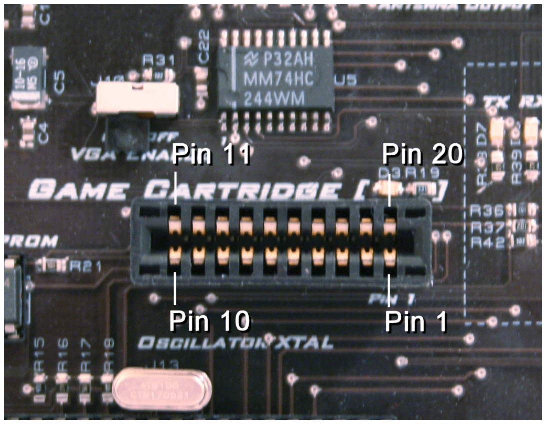
11.0 Game Cartridge, EEPROM and Expansion Port Hardware

In this chapter we are going to discuss the ***“expansion port”*** on the HYDRA as well as the design of the onboard serial EEPROM since they are related. The expansion port is the most powerful aspect of the HYDRA since it allows you to plug in enhancement products to upgrade the HYDRA including more memory, I/O, extra processors, or whatever you can think up! As is, the HYDRA comes with both a ***128K Game Card*** as well as a ***Blank Experimenter Card***. We are going to discuss all these topics and more, here’s what’s in store:

- Game expansion port design and signals.
- Blank Experimenter Card design.
- 128K Memory Card design.
- Onboard 128K serial EEPROM design.

11.1 HYDRA Expansion Port Design

Figure 11.1 – The Hydra Expansion port.

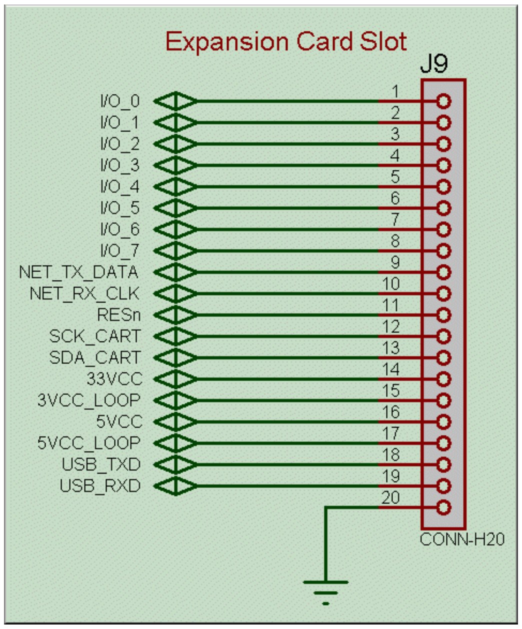


The HYDRA has a 20 pin expansion port (**J9**) as shown in Figure 11.1 that can be used for a number of expansion functions. The port itself is an industry standard female edge connector interface with 0.1” contact spacing on both sides. Typically, the idea of an ***“expansion port”*** is to export as much of the system busses and I/O as possible, so future upgrades and add on’s can be built on the platform. With this in mind, the expansion port exports;

- Power (both the 3.3V and 5.0V supplies).
- Networking (the built in RJ-11 ad-hoc serial network).
- USB (serial TX and RX along with ground).
- 8 I/Os (general I/Os from PChip also used for VGA interface).
- System reset.
- Serial EEPROM interface (so an EEPROM on the cartridge can be loaded rather than the onboard EEPROM).
- Loop back signals to detect cartridge insertions.

Everything is self explanatory except maybe the ***“loop back”*** power lines. There are used to indicate a cartridge is plugged in. The power lines are looped back on the cartridge and then “sensed” back on the HYDRA via the signal lines **33VCC_LOOP** and **5VCC_LOOP** this way the HYDRA can tell if something is plugged in and your code can take different execution paths or you can wire them directly to hardware selection logic to gate in/out various components. Referring to the image in Figure 11.1 of the expansion port, take a look at Figure 11.2, it’s the actual design of the expansion port and shows the lines that are exported.

Figure 11.2 – The design of the expansion port.



11.2 Expansion Port Signal Definitions

Table 11.1 below lists the actual signal names and their connections to the PChip as well as their functions for the expansion port.

Table 11.1 – The mapping of expansion port signals to the PChip.

Expansion Port Signal / Pin		PChip Pin # / Name	Description
I/O_0	1	21 / P16	General I/O (shared with VGA_VSYNC).
I/O_1	2	22 / P17	General I/O (shared with VGA_HSYNC).
I/O_2	3	23 / P18	General I/O (shared with VGA_BLUE_B0).
I/O_3	4	24 / P19	General I/O (shared with VGA_BLUE_B1).
I/O_4	5	25 / P20	General I/O (shared with VGA_BLUE_G0).
I/O_5	6	26 / P21	General I/O (shared with VGA_BLUE_G1).
I/O_6	7	27 / P22	General I/O (shared with VGA_BLUE_R0).
I/O_7	8	28 / P23	General I/O (shared with VGA_BLUE_R1).
NET_TX_DATA	9	3 / P2	Hydra Net Transmit Line.
NET_RX_CLK	10	2 / P1	Hydra Net Receive / Clocking Line.
RESn	11	11 / RESn	System Reset Active LOW.
SCK_CART	12	37 / P28	Serial Clock for Cartridge / EEPROM.
SDA_CART	13	28 / P29	Serial Data for Cartridge / EEPROM.
33VCC	14	POWER	Connects to 3.3V Power Supply.
3VCC_LOOP	15	POWER	Optional Feedback Loop from Cartridge to Hydra.
5VCC	16	POWER	Connects to 5.0V Power Supply.
5VCC_LOOP	17	POWER	Optional Feedback Loop from Cartridge to Hydra.
USB_TXD	18	N/A	USB Transmit Serial Line.
USB_RXD	19	N/A	USB Receive Serial Line.
GND	20	POWER	Connects to System GROUND.

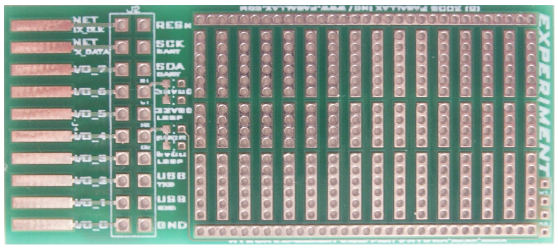
The only signals that deserve some extra explanation are the I/O_0..7 lines. These lines are shared with the VGA interface, so when you are driving a VGA monitor, you can't use these lines on your plug in card (unless of course that is the intent). The VGA Enable switch at J10 enables or disables the VGA outputs. So if you want to drive VGA with these lines then you would place the VGA Enable switch into the "on" position. When you are not driving VGA, its best to keep the VGA Enable in the "off" position. The VGA Enable switch as noted early in the book simply enables/disables the tristate buffer logic and doesn't let the IO's drive the VGA interface if they are not meant to.

Additionally, you might be concerned with power draw from the cartridge port and how much you can pull from the HYDRA? The HYDRA's power adaptor is 300-500mA, so as long as you leave enough to power the HYDRA and the systems in use then there is no limit to the power up to the supply current of the power adaptor. However, I suggest that if you do design an expansion card yourself, that you decouple the power coming in on the 3.3V and 5.0V supplies with a 0.01 – 0.1uF cap, and a 1.0-10.0uF tantalum cap to keep the power clean. Both the experimenter card and the 128K expansion card have power decoupling on them already.

11.2 Blank Experimenter Card

To get you started on your own experiments, the HYDRA kit comes complete with blank experimenter card as shown in Figure 11.3.

Figure 11.3 – The Blank Experimenter Card.

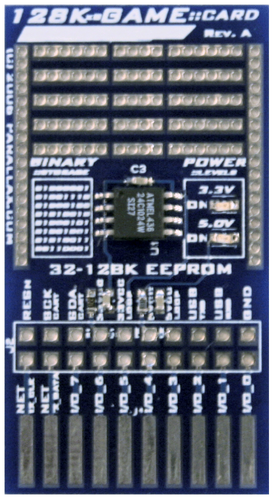


As you can see, the card has nothing on it, but decoupling caps and power LEDs. There is a header areas have a solder thru pin for each signal, so you can simply wire into them. Additionally there is space on the surface to place a couple DIP style TTL chips along with some passive components. There are 3 rows of 18 contacts, along with large "common" rails on top and bottom referring to figure. This is a great way to start experimenting with expanding the HYDRA and or adding small boards to the PChip's functionality via the expansion port. The 128K memory expansion game card was derived from this design.

11.3 128K Memory Card

Figure 11.3 shows the 128K BYTE memory expansion card that comes with the HYDRA kit. The serial EEPROM is based on the 8-pin 24C1024 model which a number of manufactures' sell (Atmel is my favorite).

Figure 11.4 – The 128K Memory Expansion / Game Card.



The 128K memory expansion card is more or less a copy of the 128K serial EEPROM design found on the HYDRA itself copied to an expansion card. The 32K and 128K expansion cards are similar, but simply have smaller EEPROMs on them. The magic of the card is that when plugged in the PChip on the HYDRA reads the card's serial EEPROM rather than onboard serial EEPROM. This is facilitated via the loop back signals. When the card is inserted, the loop back signals, specifically the **3VCC_LOOP** is used to enable the card's EEPROM while disabling the onboard serial EEPROM.

This is possible since the serial EEPROMs have addressing lines A0..2 that allow up to 8 devices to be chained together on the same buses as long as only one devices is selected at a time.

Figure 11.5 – The design for the 128K memory expansion / game card.

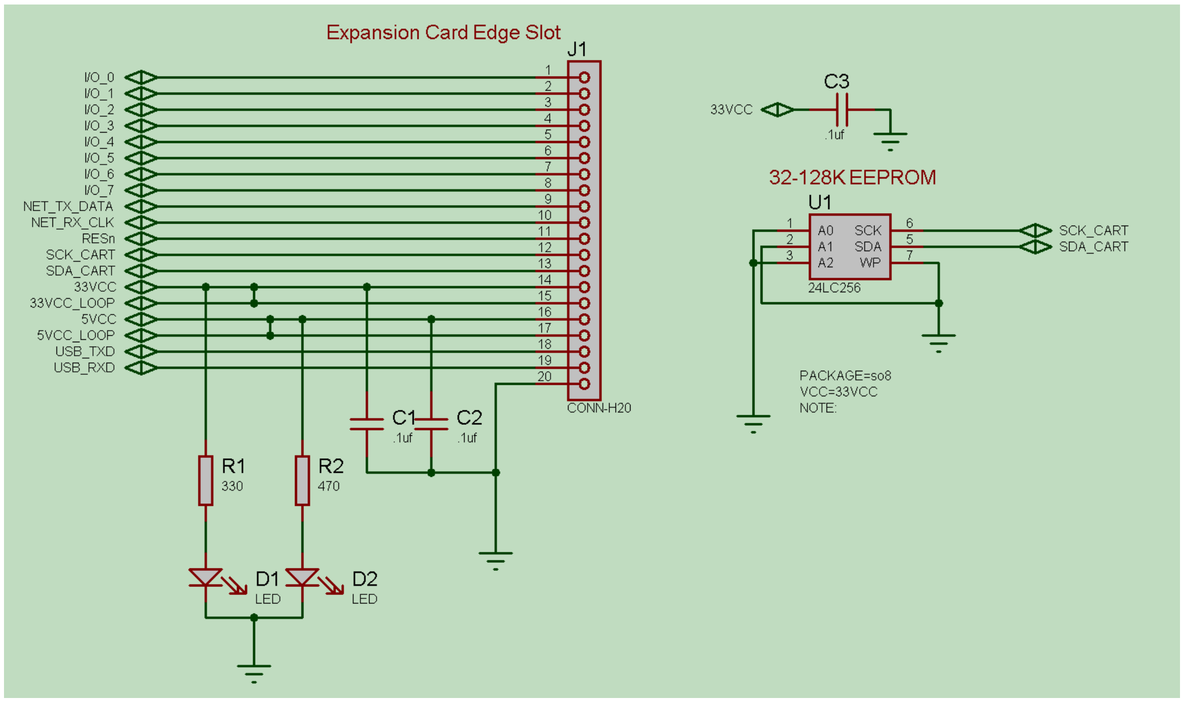


Figure 11.5 shows the design of the expansion card. Notice all 3 address selects are LOW, this always selects the EEPROM, thus when plugged into the HYDRA it will always be addressed, therefore, its up to the design of the HYDRA's onboard 128K serial EEPROM design which we will take a look at next.

11.4 Onboard 128K Serial EEPROM Design

The Propeller chip works with a 32K BYTE "image" that the IDE/compiler generates. All your code, assets, and data must fit into this 32K. However, after the PChip is done reading the 32K memory image from the serial EEPROM on boot, the lines are released and you can still access the EEPROM device yourself. The HYDRA uses the largest available serial EEPROMs which are 128K. This way you can fit your 32K Propeller program in the EEPROM, but you have an additional 96K of storage for assets, data, real-time monitoring, digitizing – you name it! Also, later when more tools are available from Parallax and others, you will be able to access the expanded 96K of the serial EEPROM and place assets there as part of the Propeller took itself. However, for now, we must as programmers write our own tools to do this and write our own drivers to talk to the serial EEPROMs expanded memory.

Figure 11.6 – The onboard 128K EEPROM design.

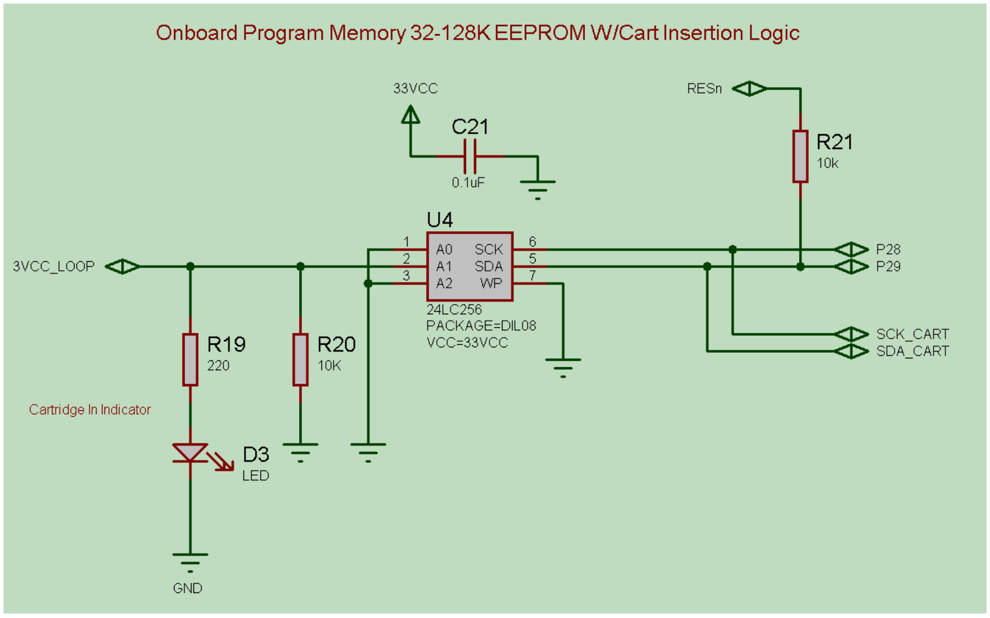


Figure 11.6 depicts the design for the onboard serial EEPROM, pin's 5 and 6 of the EEPROM are connected to the system lines P28/P29 which are the ***"serial clock"*** SCK and ***"serial data"*** SDA respectively on the PChip. Those connections are standard and nothing exciting is going on there. The only action happens on the addressing lines, notice there are 3 addressing lines A2..0. A0 and A1 are grounded, but A1 is driven by a weak pull down as well as the 3VCC_LOOP signal from the expansion port. Normally, when A1 is grounded the serial EEPROM is selected and PChip reads from it, but the moment a game card is plugged in the 3VCC_LOOP is driven HIGH disabling the onboard serial EEPROM and the EEPROM on the game card is gated in! Presto, plug and play that works!

If you're interested in learning more about serial EEPROMs and designing around them as well as programming them please review this file on the CD:

CD_ROOT:\HYDRA\DOCS\atmel 24c1024_datasheet.pdf

11.5 Summary

Having an open interface to expand any hardware device is the #1 feature as far as I am concerned. The Atari 800 for example was a feat of engineering, 10x more advanced than the Apple II, but the Apple II sold 10x more units because the Apple II allowed users to build their own expansion cards for the device and openly documented the hardware interface. The Atari 800 had expansion slots as well, but they were very hard to find information on to interface with. So the moral of the story, is the easier it is to build add on cards the more useful hardware will be. I personally, can't think of a fraction of the cool things that can be done with a Propeller chip, but I know that by having an expansion port with a commonly found edge connector interface others can create their own add on hardware and expand the HYDRA itself. At some point, I hope to see Ethernet interfaces, LCD, IDE, flash drive, and even SRAM, CPU, and FPGA cards! At very least take the free blank expansion card, throw a 7447 on there an a 7-segment display and use it for debugging!