

Project Final Report
Zilog Ethernet Port Controller
April 2006
Dean Linsalata, deanl@gwu.edu

Project Abstract

The idea of this project was to have a micro-controller system in your home that is connected to your home LAN. Then you or your family, from a remote location, that has access to the Internet and a standard browser, can control the port settings of your home controller. The controller could have many devices controlled by these ports. Such as household lighting, electronic door locks, or even your electric coffee maker. The idea was to show what the future can be like and to learn more on networking.

Originally, to add extra complexity to the project, I was going to add a feature that would give the user an opportunity to upload a software program to the Z8. Once uploaded, the software would be burned to Flash Memory and executed on a system reset. Due to time constraints and the complexity of the feature, I replaced this feature with a text box that gives the user an opportunity to send a text message to the Z8. The message would then be fed to a display, connected to the serial port, where the message can be read. I used Hyper Terminal to demonstrate this feature.

Status

Most everything went OK except for my software upload feature. I felt I could have accomplished this with more time. It took some effort getting used to the TCP protocol. Since using a web browser, to connect, I had to abide by the standards. Also, the 4K of Ram that the Z8 provides does limit what you can do, as far as dynamic memory. This would have put a tight squeeze on flashing ROM since half of my Ram was used for the web program. The whole feature would have took some clever programing and patience in debugging.

I would have liked to make my home page display a little more graphical. It would have been nice if

each port pin had its own radio button. Where the user could have just clicked on a button to turn on-off a pin, instead of entering hex numbers for values of the ports.

I did run into a couple of problems along the way. The first was actually figuring out the right pin to pin hook up locations for connecting the Nicholas to the Z8. The schematics from EDTP were confusing and there were multiple copies with different definitions. I have included my connection diagram that works.

The second problem, which I did not completely solve, was terminating my HTTP session correctly with the web browser. I had to use a reset of the NIC to terminate correctly. I'm sure there is a more proper way but I did not have the time to correct this. No matter what state I put the TCP connection in, the browser was always looking for something else to close. Even if I shut the browser down there were packets coming in once and awhile looking for a response. I did not know what to feed it. I tried multiple configurations but nothing I tried worked. I downloaded TCP diagram examples and looked at some state diagrams from books that I have, but all varied in their definitions of a web browser-web server TCP/HTTP connection session. The only way to debug this problem was trial and error and running through different configurations.

Maybe the extra packets were coming from the NIC itself and maybe the NIC needed some kind of data reset instead of a full reset but I ran out of time looking for the proper procedure. It could be the NIC I have is defective itself but I was not thorough enough in my testing to be sure.

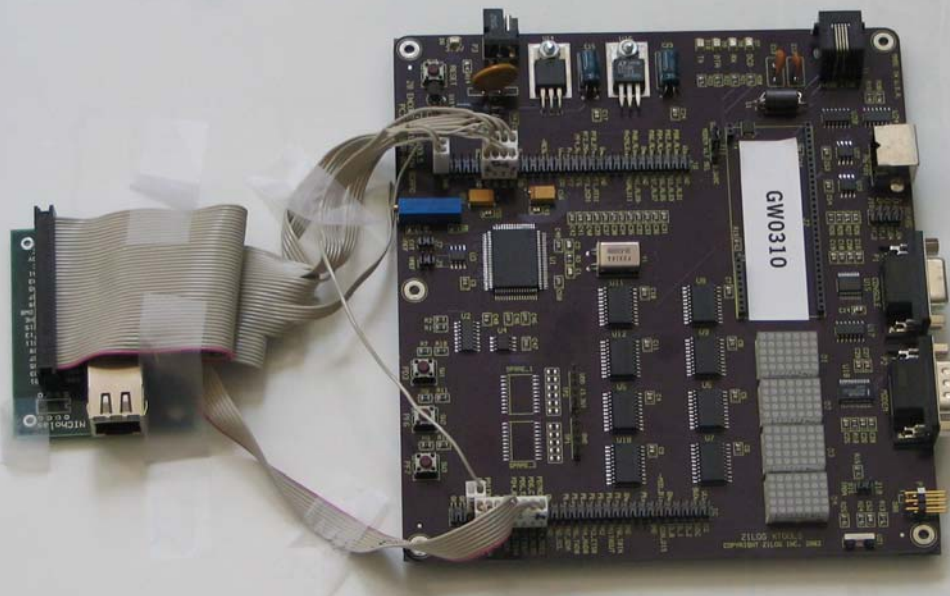
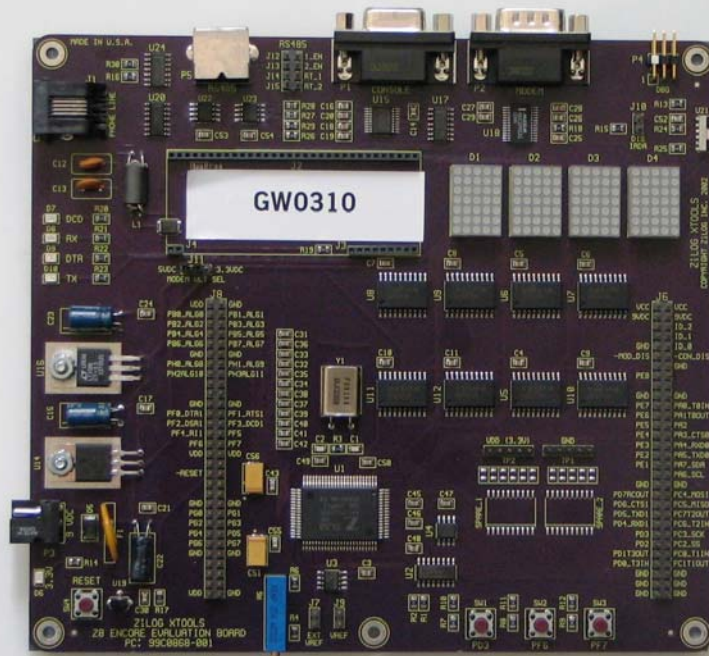
Next, below, I will provide a list of items and resources, that I have used the most, in completing this project. I will also try to explain how I got started and what I had to do to connect the devices and what software I needed to have in place, to begin writing my code.

Specification

Parts List

- Z8 Encore Z8F6403 micro-controller board
 - Zilog
 - <http://www.zilog.com/products/partdetails.asp?id=Z8F6403>
- Nicholas NIC card
 - EDTP Electronics, Inc.
 - <http://www.edtp.com/>
- Cables and Connectors
 - 40-pin ribbon cable and connector (I used a standard 40-pin hard drive cable)
 - IDC PCB .100" socket connectors
 - 1, 8-pin
 - 2, 4-pin
 - 2, 2-pin
 - Ethernet RJ45 cable
 - Serial Cable

- USB to Serial dongle (if needed)
- Software
 - ZiLOG Developer Studio II (ZDS II) 4.96
 - <http://www.zilog.com/software/zds2.asp>
 - Standard web browser
 - Z8 ASIX Code Package
 - <http://www.edtp.com/downloads.htm>
- Knowledge needed
 - Z8 Encore Z8F6403 micro-controller board and processor
 - C programming language
 - TCP/IP protocol
 - HTML and a HTTP session
 - soldering (if needed)
- Recommended documentation and reading
 - Z8 Encore!™ Z8F640 Manual PS017610-0404
 - Easy Ethernet/NICholas Schematic Package
 - <http://www.edtp.com/downloads.htm>
 - AX88796 Documentation (processor on the Nicholas board)
 - <http://www.asix.com.tw/download.php?op=searchresult>
 - TCP/IP Embedded Internet Applications by Edward Insam
 - Embedded Hardware by John Catsoulis
 - C Pocket Reference by Prinz & Kirch-Prinz



Z8		NIC		Z8		NIC		NIC		Z8		NIC		Z8
PC0	<--	SD0		N/A	<--	SA8		SD1	-->	PC1		SA9	-->	+3.3V
PC2	<--	SD2		N/A	<--	SA6		SD3	-->	PC3		SA7	-->	N/A
PC4	<--	SD4		PG4	<--	SA4		SD5	-->	PC5		SA5	-->	N/A
PC6	<--	SD6		PG2	<--	SA2		SD7	-->	PC7		SA3	-->	PG3
N/A	<--	SD8		PG0	<--	SA0		SD9	-->	N/A		SA1	-->	PG1
N/A	<--	SD10		PD0	<--	RESET		SD11	-->	N/A		NC	-->	N/A
N/A	<--	SD12		N/A	<--	AEN		SD13	-->	N/A		RDY	-->	N/A
N/A	<--	SD14		GND	<--	/CS		SD15	-->	N/A		NC	-->	N/A
PG7	<--	IRQ		GND	<--	GND		/BHE	-->	N/A				
PG5	<--	/IORD		+3.3V	<--	+3.3V		/IOWR	-->	PG6				

Implementation/Construction

To get started with the project you would need to assemble the hardware. Once you have your two boards you will need to figure on how to connect them. I have provided a wiring diagram above. There are 21 connections to be made. You can either make a custom ribbon cable as I have shown above or devise your own wiring method. Either way the 21 connections need to be made as shown above.

Once your board is connected to the Nicholas NIC you would just need to connect the Z8 to your PC. You will need a serial cable and or a USB-Serial dongle if your PC does not provide a standard serial port. Either cable would have be connected to the one-wire debugger pin port on the Z8. Zilog provides the one-wire to serial adapter for this connection.

Next would just be connecting power to the Z8 board and a network cable from your PC to the NIC. If using a direct PC/NIC-Nicholas/NIC connection with nothing in between such as a router or switch or hub, you will need a cross-over network cable.

To setup your software side you will need to download the Zilog Developer Studio II as stated above. There is a patch for it so you might as well download and install that also. You will need Windows 98 or above. You can then familiarize your self with the IDE. Then you will need to download the Nicholas code for the Z8. The download site is mentioned above. From these downloaded EDTP files, I just used the z8_asix.c source file to get me started. Create an appropriate project in the IDE and then import the z8_asix.c source file.

I did not use the UART functionality of this code because it was a little confusing on hoew it was being setup, so I just commented out these references and created my own. To familiarize myself with the code and the NIC functional, I studied the ASIX manual that you could download as stated above and followed the code.

Once you get your project compiled correctly, you should be able to do a simple ping to the Z8. The code provides a variable MYIP[] to set the NIC with an IP of your choice and there should be no need to change the hardware address MYMAC[] unless using your system directly on the Internet.

If you are able to ping the device you are ready to move on and learn TCP. Change the #DEFINE

MY_PORT_ADDRESS to 0x50 this will be the standard web server address that web browsers default to send to. When the Z8 code sees a TCP packet coming in will direct the flow of the code to the “void tcp(void)” method. This is where I took over and wrote the functionality to respond to the browser's request.

Before writing your own TCP code you will need to study and understand TCP/IP. I found the book TCP/IP “Embedded Internet Applications by Edward Insam” to be most helpful. After reading and understanding the protocols, you will find the TCP flags(SYN,FIN,ACK), TCP sequence number, and TCP acknowledgments number to be the most important and most difficult concept to understand.

To help understand the TCP concept, I create my own incoming and out going packet variables and setup the UART to display some results. I was able to see what values were coming in and what needed to be sent out. I will not explain further because this is up to you figure out and understand.

Retrospective

Some important things I have learned while doing this project and others.

Take your time when wiring your devices. Make solid connections and test each end point. The extra time spent will pay off when there are problems down the line. You will be confident that the problems do not lie in your connections. Modularize your connections to make your devices easy to connect and disconnect. And make sure your connections are insulated.

Simplify and make your development environment as efficient as possible and keep a clean work area. Such things as limited screen resolution & size, slow processor speed, and clutter will bog you down.

Work on your debugging techniques. I was a little lazy in configuring my UART and writing some code to display and format the results of the incoming and out going packet information. If done from the start this might have sped up my project. I tried to use what the IDE provided but this was limited. They should provided a quick way of displaying arrays of information side by side to debug results.

But the main thing I have learned from this project is that by the use of a micro-processor board and a device of your choice makes a great tool for learning and understanding different technologies. PC device specifications and their software driver code are hidden from the user. Todays operating systems hide the functionality that needs to be seen for a computer scientist to understand. The embedded boards and some of the modules sold for them expose this important information.

Network configuration, network protocols, and the infamous network stack were never fully understood to me. I have a pretty good understanding now!