

Re: USB on a breadboard?

Source: <http://coding.derkeiler.com/Archive/General/comp.arch.embedded/2006-08/msg00810.html>

- *From:* "Noway2" <no_spam_me2@xxxxxxxxxxx>
 - *Date:* 8 Aug 2006 06:24:08 -0700
-

brehob@xxxxxxxx wrote:

Hello all,

I'm an instructor at a large US school who has slowly been moving into teaching embedded systems. I know very little about embedded systems in the real world (I'm more of an architect), but I've learned a lot by teaching and working with students on embedded projects.

We currently have a class that uses a combination of an MPC823 and an FPGA to do software/hardware design. It is mostly in assembly (a bit of C) and covers the details of doing memory-mapped I/O, designing the hardware devices, interrupts (in gory detail), timers and A/D stuff.

I've been trying to upgrade our embedded systems offerings and so I am offering a class next semester that is a follow-on to the current one. I've chosen to go with two different platforms, an ARM based system from Intel (sitsang) with an OS, color touch LCD, and crazy amounts of I/O as well as an Atmel based portion (ATmega8) where the students will design (and get built) a board.

So now the questions:

#1 Is there a USB chip that I could possibly talk to with the ATmega8? I'd love something I can put on a breadboard without having to solder very small wires. Looking at the student projects I'm expecting, USB may be a pretty common desire. The sitsang board does USB in its sleep, but I'd like them designing their own hardware where possible.

#2 Same as #1 but for ethernet.

#3 What do you all think is important for them to know? If you were to hire one of our graduates, what specific skills would you be looking for? Are skills like being good with solder important to any/many of you? Just software?

#4 What am I missing?

More background:

I'm going to teach a fair bit about how to interface with an OS (for the sitsang) and the basics of Linux drivers. I'm also going to spend

Re: USB on a breadboard?

a short time on the AVR architecture and interrupts (which I'm still learning) as well as a bit on the basics of board design. I'm planning on having a few guest speakers come in and talk about what they do (one a WIMS person, one works for Rabbit, and maybe a Lockheed–Martin embedded person).

The major goal of the class is to give them the opportunity to design a system (probably involving board design) of their choosing. We have a reasonable budget (thanks Lockheed–Martin!) of around \$800 per group of 4 students. So we should be able to build some cool things!

Answers to the above or random thoughts gratefully accepted!

The first thing that I would like to say is that I am pleased to see that these types of classes are being offered and I am impressed at your initiative.

In response to your questions, hardware development with USB and Ethernet may be a little bit tricky, from the breadboard perspective. The reason being is that both technologies utilize differential signaling on lines with controlled impedances which would be difficult to achieve with a breadboard. I think the suggestions you received recommending a development kit or adapter are probably the way to go. This way you could at least focus on the concepts of the protocols and possibly hook up a scope or something to look at the electrical signals.

Second, I think that your focus on interrupt handling and uP interfacing is very good. As an EE who has worked with embedded systems for the 10+ years since graduation, I find that these skills have come in handy on just about every project. Based on my experience with a computer science person who reports to me, I think these skills could have used some more attention in his education as they made his eyes glaze over when he had to deal with low level issues that would normally be handled for you by an OS. Generally speaking, his ability to program software was excellent, but he would become befuddled when faced with hardware interfacing. Granted, he is a CS major, but not everybody who graduates from the program will go work for Microsoft.

Third, your work to interface with Linux is an excellent idea, for two reasons. First, there is the aspect of embedded systems that focuses on using single board computers (PCs) with a scaled down operating system, such as micro–Linux. Second, it is sometimes possible to obtain a development card that interfaces to a PC to do some early work prior to the creation of the target application. As Linux allows a user to modify the kernel, and place handlers and other types of code in the system it makes an excellent platform for this type of development. Windows (spelling deliberate) isn't nearly as flexible as with it, you are the tail and the OS is the dog.

Re: USB on a breadboard?

I also think that your idea about some soldering experience has merit. I believe that being able to perform some basic board rework is a valuable skill for both engineers and technicians. A course that provides IPC certification takes about two days so it should be possible to cover a fair amount of technique in a few class or lab sessions. I personally, populate all my new designs on the first prototpe. This allows me to populate sections like the power supply and reset circuit to verify it before populating the rest of the board. While it may be an unconventional approach, it certainly makes me more comfortable than the idea of throwing the switch and hoping that my last four months of effort doesn't go up in smoke.

I think that introducing FPGA design is an excellent idea too. Not only are FPGAs becoming increasing common in many designs, I believe that being able to work with them has helped my career based on employer comments in annual reviews.

You may want to consider introducing some of the aspects of real time design and programming. Real time issues are inherent to some degree in just about every embedded system. In contrast, though, in most PC based applications, the user has little control over the timing. I think uCOS-II might be an excellent avenue to pursue this.

Other areas to consider discussing would be the importance of system reliability and how this is achieved through the use of watchdog circuits and proper coding practices. Also you may want to discuss the importance of good user interface design and how often times the designers think very differently from those who use the systems.

The only other thing I can think of that would be helpfull, that you didn't explicitly mention is some experience with development tools like how to properly use oscilloscopes and logic analyzers and most importantly, understanding their limitations.

.