The Embedded Muse 22

Editor: Jack Ganssle (jack@ganssle.com) June 15, 1998

Hardware Helpers

Last issue I gave a few ideas of how we can improve system debuggability when designing the hardware. A number of folks sent more thoughts along, which are listed here.

Thanks for all of your feedback, and please do keep those ideas and comments coming!

From: Leonard Picone
Make control registers in an ASIC readable. I'm currently working on a project with write only control registers and have seen them in the past as well.

From Ed Sutter
On the schematics, add some comments...
a. For all off-page routes, provide some comments on what page the route goes to. Some CAD packages apparently do this automatically, but I haven't seen it. I spend a lot of time just trying to find routes on schematics.
b. For a part that has the Vcc and GND pins invisible on the schematic, provide a comment indicating which pins are connected to where.
c. For each complex part on a schematic, comment on physically where pin #1 is and what direction the pin numbering is in (CW or CCW).

Provide convenient clip-lead access to all distinct power points, to include ground.

Provide a watchdog timer and make it disable-able! What I recommend is that the input pin that must be toggled by firmware be optionally (via jumper) tied to a hardware pin that should always be toggling (ALE on x86 for example). This provides the firmware guy with something that will keep the watchdog happy when necessary, but still provides a hardware watchdog.

If the application has "leftover" communication ports, make them accessible! For example, some CPUs have two UARTs and only one is needed for the design. At least provide easy access to the TX, RX and GND pins so that a Maxim part can be glued down for a spare debug port.

From: Steve Leibson
Like ground, put the board's master clock on a pin that's easy to grab. Makes things much easier to scope or trace on an analyzer.

Always check a bare PC board fresh from the fab for a short between Vcc and Ground. Because there are so many access points for these two "nodes," they're the easiest to short. If there is a short, connect the bare board to a honking power supply and run some current through the short. You'll either blow it or you'll be able to find it using the "burn your finger" heat test. Either way, you'll locate the short.

Either install a signal LED for each power supply directly on the board (so you'll know they're on) or install a small connector and develop a standard power-monitor clip with the required LEDs and resistors so you can be sure that all of the power supplies are connected properly and that they're turned on.

From: Miguel Flores
Put reference designators in some sort of logical order on the board so you can find them. It's also helpful to include an assembly drawing in the schematic package in case the silk-screen is too dense.

For firmware, any sort of diagnostic I/O signal or LED should be dynamic, not static, so you can tell if the CPU has halted or is stuck in a loop.

Label switches and jumpers used for configuration (such as I/O bus address) with a legend on the silk-screen.

Make all connectors keyed so there is no guessing about which way the cable is supposed to go.

From: Jim Thomas
Make lots of narrow memory-mapped control/status ports instead of a few wide ones (address space is cheap!). I am always flabbergasted to get a board that has two or three 16-bit wide memory-mapped control ports, each containing four of five different functions. I hate having to be careful not to accidentally clear an interrupt when I'm trying to toggle an LED. It is better to wire different functions to different addresses. That way, the programmer can turn LED's on and off with wild abandon, never worrying about accidentally toggling the card's bus lock bit.

As an added bonus, this approach can use far fewer pins on a PLD (if that's where the control registers reside). Four data bits and four address bits will control sixteen 4-bit registers, for a total of 64 control bits. Doing this with four 16-bit wide registers (each register having many functions) would require 16 data and two address bits. Do the arithmetic! And it's a ton easier to program.

From: Scott Finneran
Add a couple of spare I/O bits to the design that firmware folks can use to instrument their code. If space (and decode logic) permits, include an entire 8 bit register connected to a row of .1 spaced vias or headers. Software state-machines can output their current "state" to this port which can be captured with a logic analyser. This has the advantage of not effecting timing as serial debug messages tend to do. Make it separate to any status LEDs.

If you are including a serial port for debugging, make sure that it has interrupt capability, nothing slows down software when debugging (causing it's own set of problems) than polling a serial port.

Don't just provide a GND post/test-point. Also include one for Vcc close by. This allows easy connection of the old single input, 1-bit trace memory logic analyser (the logic probe). Many simple "did it happen or not" problems can be solved using one of these babies without tying up the heavy artillery.

Two words: Flashing LEDs.

Four words: and lot's of 'em.

From: Jeff Corwin
Put an LED on a spare I/O output pin as a "confidence" indicator for the users. In my most recent project, a remote control interface for some broadcast equipment, I have a LED which flashes at a 1 second rate. Since there is no user interface for these boxes, seeing the LED's flash every second (synchronously, from a signal from the master unit) let's our maintenance engineers know at a glance that the system is working.

From: David Timmins
When tracking a PCB that has non-power pin of an IC tied high or low, don't use the pin as a convenient way to get the supply to the other side of the pins, say, by tracking the supply through the pin and out the other side. I call this "tracking through" and it makes debugging the PCB harder if that pin needs to be wired to somewhere else.

When powering up a new PCB for the first time, meter out the power supply pins from the power connector to some of the ICs on the PCB to make sure that it is tracked correctly and that power connector hasn't been put on the wrong way round. This can save you the time and embarrassment of blowing up the PCB before you even start to debug it.
(Editor's Note: I like to power boards from a current-limited lab supply that has an ammeter. I look at the current from time to time to make sure I'm not doing anything expensively stupid… Jack).

Don't parallel inputs unless you have to, as it increases the load on the device driving them and slows the circuit down. For example, if you are using a 3 input NAND gate as an inverter, tie the two unused inputs high rather than connect all three inputs to the signal.

Thought for the Week

Alcohol and calculus don't mix. Never drink and derive.

About The Embedded Muse

The Embedded Muse is an occasional newsletter sent via email by Jack Ganssle. Send complaints, comments, and contributions to him at jack@ganssle.com.

To subscribe, send a message to majordomo@ganssle.com, with the words "subscribe embedded *your-email-address*" in the body. To unsubscribe, change the message to "unsubscribe embedded *your-email-address*".

The Embedded Muse is supported by The Ganssle Group, whose mission is to help embedded folks get better products to market faster. We offer seminars at your site offering hard-hitting ideas - and action - you can take now to ***improve firmware quality and decrease development time***. Contact us at info@ganssle.com for more information.