THIS IS HUGE

CC Gold

OVER TWO DECADES OF ENGINEERING EXCELLENCE
At Your Fingertips

Now you can have the complete Circuit Cellar issue archive and article code stored on a stylish, durable and portable USB flash drive. You can easily keep your CC Gold archive up to date by purchasing subsequent issues from our webshop or by downloading issues with a Circuit Cellar Digital 2.0 Subscription. Issues appear in searchable PDF format.

Complete archive includes PDFs of all issues in print through date of purchase. (For display purposes) USB drive shown is larger than the actual size.

Visit www.cc-webshop.com to purchase
Design Development (Part 2)

Product Implementation

It’s time to dig deeper into the design process. This month schematic capture software is used to define the hardware portion of a design.

Last time we started a product design. I laid out the framework for what an instrument might look like. And I said I was going to design a spring tester because I wanted one. Well, I still want one, but that seems too narrow a focus to keep everyone interested and handing in their homework on time. And remember, homework is 30% of your final grade. (Doesn’t that thought just give you chills? Aren’t you glad it’s no longer true!?) So, let’s open up the design a bit and change it to an environmental monitor. Let’s try to make it useful to monitor your house’s environment or the environment of a container as it’s shipped across the country. It’s still basically a piece of instrumentation and, as you’ll see, the block diagram changes only slightly.

What did we really do to our requirements by changing the design? What additional interfaces on the input and output sides will be needed?

What would a customer like to have in a spring tester? If you recall, I listed our design requirements in my April 2011 column (Circuit Cellar 249). We want methods to do the following: measure the characteristics of a spring, compare the measurements of a spring to a standard, calibrate the device, save the results of a measurement, recall the results of a measurement, print the results of a measurement, perform a go/no-go type of test, gather statistical manufacturing data, update software, work with different spring rates, and measure leaf-type springs.

Let’s try the requirements again for our monitor. What would make a good environmental monitor? It should include methods to do the following: measure temperature, measure humidity, measure atmospheric pressure, measure accelerations in three axes, measure time/date (and to save and recall these readings), view the readings graphically, set alarms for out-of-limit conditions, update software, and operate from a battery.

So, it’s a different list, but just how different? Very little, I think. As I go about the design process for our monitoring product, I’ll use hardware schematics and software modules to implement the design. Let’s call this new list of requirements, “RequirementsListRevA.” I’ll make it a separate document and keep it up to date. We can then compare our design to the list and see if we are still on track.

HARDWARE DESIGN

At this point in the design process, we can take a first pass at either the hardware or the software. If we were designing a messaging system that was included in a large pool of flash memories, I would start the software design first. The software design probably would be more difficult than the hardware design in that case. So, starting with the software might uncover and flush out problems earlier in the design process.

In this case, I would start with the hardware design. I’m recommending this because there are so many unknowns in that area. I’m also going to draw a schematic for each of the hardware design modules. I believe I’m going to show you three different versions of the schematics. The first is the initial design capture, the second is a more refined version with parts identified, and the third would be a version almost completely captured in the design software. I use Altium’s Design Explorer (DXP) integration platform for schematic capture and PCB design. Altium has viewer software that you can download to view the design without
purchasing the complete software package. And, as a final thought in this first-pass hardware design, we should only generate schematics as a result of requirements. If there is no requirement, there is no need for a schematic.

TEMPERATURE & HUMIDITY
There are several ways to measure temperature. Thermocouples and resistance temperature detectors (RTDs) are probably the two most popular and common. I suspect that measuring humidity will have a temperature component in the calculation or circuitry. So, let’s also look at humidity sensors first. I start with component distributors such as Digi-Key or Mouser Electronics and search their websites. You will find digital temperature sensors from IC manufacturers (Texas Instruments and National Semiconductor come up). A search for humidity sensors shows units that give a voltage, frequency, or capacitance that’s proportional to humidity.

From experience, I know Sensirion’s SHT21 digital humidity sensor measures both temperature and humidity and has an FC interface. I know I took a big leap here, and if you didn’t know about this part, you would spend hours trading off the various temperature and humidity solutions. If your system requires several remote temperature measurements, the SHT21 is probably not a good choice.

The first pass on this function is done. The next task is to draw a schematic. I use Altium, but at this level most of the design packages are basically the same (see Photo 1). I needed to create the symbol for the SHT21 and that’s part of the package. I added power and ground connections that come from the datasheet. All the power connections will be connected together across all the sheets of the schematics. So the V 3.3 and GND_DIG just connect into one big net each across all the schematics.

Read Circuit Cellar columnist George Novacek’s articles about grounding for information about how to deal with connecting all the power and ground signals. I left this schematic rather rough since it’s a work in progress. I feel it’s important to show you the chaos that’s referred to as the design process. Also the SHT21 is an FC device. I put module ports (the elements that will eventually connect all the schematics together) because not all the microcontrollers have FC ports. Some have only SPI. I do not want to hide this design issue, so I left those connections dangling.

Choosing parts is like reading a Charles Dickens novel. Two hundred pages (days) later these decisions will come back. Some return as a long-lost friend with help and advice while others return and bite you in the rear (a technical term).

REAL-TIME CLOCK
Our requirements necessitate that we keep time of day. Tom Cantrell recently wrote about a device that does just that (“Time Traveler: Embedded Timekeeping and More,” Circuit Cellar 248, March 2011). Let’s use it. It’s Microchip Technology’s MCP794xx Real Time Clock/Calendar (RTCC). You’ll find that most of the major IC manufacturers have similar real-time clock (RTC) devices. This device also uses the FC interface, so I hope we choose a microcontroller that has an FC interface. When I draw the schematic, I show the MCP794xx, a crystal, a back-up battery and holder, and all the resistors and capacitors the datasheet recommends.

ACCELERATION
Our requirements specify the measurement of acceleration in three axes. This obviously means a three-axis accelerometer. Going to the distribution webpage again, I first come across an Analog Devices ADXL325 three-axis ±5-g accelerometer. This is straightforward. Power is with 3.3-V DC. Connect capacitors to each of the three outputs to set the bandwidth of the output response and you will have three voltages proportional to the acceleration on each axis. Each of those three outputs can go into three inputs of an ADC and then be digitized to get the data into the digital domain.

If you read the previous paragraph quickly, you might have skipped over the term “bandwidth.” What’s that reference to bandwidth? It was never specified in the requirements. And, if you think about it a bit, the higher the bandwidth, the higher the sampling frequency required to extract the information. Is the accelerometer going to be powered up all the time? What about shocks or bumps to the system? Do we have to capture them? We could add peak and hold detection circuits to each of the three outputs and digitize those additional three outputs. Actually, each output will be at 3.3/2 V for 0 g. So, the peak detectors need to capture positive and negative g’s. Ugh! It’s getting complicated.
If you keep searching, you will come across an STMicroelectronics LIS31DLF accelerometer. It's a digital version of the analog device previously described. It has outputs that can be I2C or SPI. It also has internal registers that can set limits and interrupt the system when these limits are tripped. You can set the sensitivity to 2, 4, or 8 g and it costs only $5 for one quantity. At this point, I would select the STMicro part and start a schematic page for the accelerometer.

ATMOSPHERIC PRESSURE

I searched Digi-Key for pressure transducers and got more than 19,000 results. Using its selection criteria, I found the transducers that would run off of a 3.3-V supply. That narrowed the results to 66 devices. Selecting surface-mount devices, I got one page of results ranging from $4 to $25. The requirements are to measure the pressure that the instrument encounters. I'm assuming a plane trip might be the most extreme condition we encounter, and I'm not going to consider monitoring for sudden loss of cabin pressure.

For the sake of this project, let's select a Bosch Sensortec BMP085 digital barometric pressure sensor. It costs $9 for one and less than $6 per unit for larger quantities. The pressure readings represent altitude from −500 to more than 19,000 m. I think this will meet the intended requirements.

According to the datasheet, the interface is I2C! It certainly is a digital world we live in today. We can design this instrument with purely serial digital interfaces. Also, the pressure transducer provides pressure and temperature readings. Hold that thought.

Again, I would generate a schematic for the atmospheric pressure signal. I would place the device on that schematic. At first pass, these schematics don't seem worth the effort. I find that once they are on paper I can review them, add devices such as decoupling capacitors, and refine the connections. Also, these pages give you a place to put more project-type information, such as power consumption, PCB square inches required, and manufacturing costs.

MEMORY & POWER

I was thinking that a small removable memory card, such as those used in cameras, would be a good choice for our memory. On the memory schematic, you’ll see the connector for that device. I’ve also included a connector for a compact flash-type memory card. The compact flash memory card is not a requirement. I wanted to show the hardware requirements necessary to support such a device. Basically, that interface requires address lines, data lines, chip selection, and read/write control. You can do the interface in either 8 or 16 bits of data.

Again, the schematic shows the interface. The connector looks simple, but it’s a much more complicated device. I put USB and Ethernet on one schematic at this point. The selection of the CPU will add a lot more detail to the parts required to connect the CPU to the appropriate interface connectors. Some CPUs contain more of the interface electronics than others. No need to detail the design at this point in the process.

This unit needs to run off of a battery. I try to talk my customers out of building battery chargers as part of a unit. Battery chargers have been known to catch fire and burn down buildings. No need to take on that responsibility. So, let’s assume we have a battery pack that can be externally charged. Also, when the unit is connected to the USB, Ethernet, or an external power source, power should come from the external supply and not from the battery. This means diode ORing of some sort. Again, until we select the CPU, we probably should not put much effort into the power supply details.

WHERE ARE WE NOW?

Is this ugly, or what? It is ugly and certainly a mess. I bet you've never seen the design process presented in this manner. Well, this is the way it evolves. Also, it's typical to run into a road block during an initial pass. We did with trying to capture peak accelerations from an analog sensor. But, let's say that one of the devices (humidity) comes in at too high a price or consumes too much power. I would generate a new schematic [Temperature and Humidity Version 2] to try and solve the problem.

We know there is a second source to temperature (the pressure transducer), so perhaps we should include a humidity transducer that gives an analog voltage or a frequency output that is proportional to humidity. Yes, it means restarting the search, finding the component, creating the schematic, and then looking at this approach in the overall system design.

This design process is not like building a house where a foundation is first built and then the house fits perfectly on that foundation. It's more like a sculpture. You've got a block of stone and you start chipping it with sculpting tools. After the first full day of hammering, it still looks like a piece of stone. It’s closer to becoming a statue, but it doesn’t show.

There is one other schematic that I generate to hold all of the detailed schematics. I typically call this the “Top-Level Schematic.” It’s mostly a place holder. I might move the removable battery and the On/Off switch to that level. On/Off switch? What On/Off switch? It's not in the requirements, but we probably need one. Again, passing over the design again and again reveals more details that need to be incorporated into the design.

Before we select the CPU, I would take a pass at the design of the software. And next time I will go over that process.

OUR UGLY DESIGN

It’s ugly, but we have a design. It won’t work and it’s not properly connected, but it’s a basis for all the next steps. If you’re looking into an interface, just follow the steps as if they were part of this project and put it in at the top-level schematic. Your requirements are, of course, different, but you should be able to duplicate these steps for your design.
George Martin (gmm50@att.net) began his career in the aerospace industry in 1969. After five years at a real job, he set out on his own and co-founded a design and manufacturing firm (www.embedded-designer.com). His designs typically include servo-motion control, graphical input and output, data acquisition, and remote control systems. George is a charter member of the Garcia Design Works Team. He is currently working on a mobile communications system that announces highway information. He is also a nationally ranked revolver shooter.

**PROJECT FILES**

**RESOURCES**


**SOURCES**
The Design Explorer (DXP) Integration Platform
Altium Ltd. | www.altium.com
ADXL325 Three-axis accelerometer
Analog Devices, Inc. | www.analog.com
BMP085 Digital barometric pressure sensor
Bosch Sensortec | www.bosch-sensortec.com
Humidity sensors and pressure transducers
Digi-Key Corp. | www.digikey.com
MCP794xx Real Time Clock/Calendar
Microchip Technology, Inc. | www.microchip.com
Humidity sensors
Mouser Electronics, Inc. | www.mouser.com
Digital temperature sensors
National Semiconductor Corp. | www.national.com
SHT21 Digital humidity sensor
Sensirion | www.sensirion.com
LIS331DLF Three-axis linear accelerometer
STMicroelectronics | www.st.com
Digital temperature sensors
Texas Instruments, Inc. | www.ti.com

---

**Got Serial, Need Network?**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bluetooth Qty 1</td>
<td>$145</td>
</tr>
<tr>
<td>Ethernet Qty 1</td>
<td>$99</td>
</tr>
<tr>
<td>Wireless Qty 1</td>
<td>$199</td>
</tr>
</tbody>
</table>

**Volume Discounts Available**

**STC Microcontroller**
The World's Largest 90S1 MCU Family

- Enhanced 80C51 CPU, 1T per machine cycle
- Binary level compatible with conventional 8051
- 0.5-52K on-chip flash ROM
- Up to 2048/32 RAM
- In-System program or In-Application program
- Internal oscillator, reset, WDT
- On-chip ADC, PWM, EEPROM, SPI
- Addressing up to 64K of external RAM
- Dual data pointer
- 6 vector address, 2 level priority interrupt
- Up to 3 UART with baud-rate generator
- Low power consumption
- 8 to 48Pins, 24/32/48/DFN/SOP/DIP/SSOP
- Ultra safe code protection for flash ROM
- Highly ESD protecting

**F700 USB ISP Tool**

**STC11F04E**
Unit Price:

**Compatible with AT89C4051**
6-8 times faster in average
2K EEPROM
14, 18, 20pins DIP/SOP/LSSOP

Reduce the cost dramatically while considerably improving the performance

Tel: +8610 8256 2708 Email: sales@strict-51.com
Website: http://www.strict-51.com

---

www.circuitcellar.com • CIRCUIT CELLAR®
Vehicle Diagnostics (Part 2)

Pre-CAN Protocol Standards

The first part of this series was a primer on automotive control systems and OBD-II data. This article explores the pre-CAN protocol standards still in use. You can use this information to diagnose trouble codes and more.

After reading my introduction last month, I hope you were curious enough to look for your car’s on-board diagnostics II (OBD-II) connector. They were installed around the driver’s seat in all automobiles manufactured after January 1, 1996. This single connector contains all of the networks in use today, as well as the control area network (CAN) standard, required for automobiles manufactured after January 1, 2008. We can thank the Society of Automotive Engineers (SAE)—encouraged by early EPA emissions standards—for the foresight of proposing an open-ended standard communications set for the control and monitoring of engines and other subsystems. And just as important was its plan for phasing out the various protocols in use for a single standard.

We are well past the date requiring new automobiles to use the single CAN standard. But, because vehicle lifetimes average 12 years, most of the automobiles on the road today still use one of the pre-CAN standards. Therefore, it is possible that your car uses one of the older standards. The purpose of this series is to explain each of the standards so you can make use of the information available on one of the OBD-II networks for some ulterior motive. This might be to merely diagnose a trouble code, to indicate where to look for a component problem, or to provide some additional information about a vehicle that isn’t currently presented (e.g., real-time fuel consumption versus distance or MPG).

As engine compartments become jammed with increasingly complex engine control systems, it is easy to understand that what began as emission control and verification has grown into much more. Today, we have “smart” transmissions, brakes, tire pressure monitoring, and vehicle suspension, as well as creature comfort systems such as cabin temperature controls, navigation, and entertainment. Our automobiles are morphing

<table>
<thead>
<tr>
<th>Mode</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$01</td>
<td>Identifies what powertrain information is available to the scan tool</td>
</tr>
<tr>
<td>$02</td>
<td>Displays freeze-frame data</td>
</tr>
<tr>
<td>$03</td>
<td>Lists emission-related “confirmed” DTCs as four-digit codes identifying the faults</td>
</tr>
<tr>
<td>$04</td>
<td>Clears emission-related diagnostic information including clearing stored pending/confirmed DTCs and freeze-frame data</td>
</tr>
<tr>
<td>$05</td>
<td>Displays the oxygen sensor monitor screen and the test results gathered about the oxygen sensor</td>
</tr>
<tr>
<td>$06</td>
<td>Requests on-board monitoring test results for a continuously and non-continuously monitored system (There is typically a minimum value, a maximum value, and a current value for each non-continuous monitor)</td>
</tr>
<tr>
<td>$07</td>
<td>Requests emission-related DTCs detected during current or last completed driving cycle</td>
</tr>
<tr>
<td>$08</td>
<td>Enables the off-board test device to control the operation of an on-board system, test, or component</td>
</tr>
<tr>
<td>$09</td>
<td>Retrieves vehicle information</td>
</tr>
<tr>
<td>$0A</td>
<td>Stores emission-related “permanent” DTCs</td>
</tr>
</tbody>
</table>

Table 1—Each command begins with a mode byte. These modes are used to perform general tasks. Each mode can have numerous formats. A second PID byte further defines functions supporting that mode.