

Get hardware fast

BUILD A TESTBENCH USING THE MICROCONTROLLER IN YOUR PROJECT TO SIMPLIFY THE NEW-TASK LEARNING CURVE, GET A HEAD START ON HARDWARE/FIRMWARE INTEGRATION, AND SHORTEN THE OVERALL DEVELOPMENT CYCLE.

Testing the firmware for your microcontroller project without hardware is about as satisfying as looking at photos of food when you are hungry. But the lack of “real” hardware does not have to delay hardware-based testing of firmware. The same hardware that you use to evaluate a device can potentially serve as a testbench to verify your design. If you already have an evaluation kit, there is no added cost. Additionally, evaluation kits frequently use the same development tools you use to create production designs. With the widespread availability of evaluation boards and modules for low- to high-end devices, almost every engineer can take advantage of this development technique.

Early testbench development can significantly speed the time it takes to integrate the firmware once real hardware is available. But this improvement is not the only reason to take this approach. Having everyone on a project creating testbenches to verify their design will increase the quality of the overall firmware your team creates, and the end result will be more reusable on future projects and by other teams.

EVALUATION MODULES

Most microcontroller vendors provide suitable general-purpose evaluation modules for their products for as little as \$69, and you can secure even the more complex—that is, more capable—boards for less than \$500. Evaluation boards supply a range of hardware capabilities and demo or production-ready software. To make a good testbench, however, a board needs to provide some functions beyond testing the performance of a microcontroller for an application. Several key characteristics make an evaluation board a good testbench:

- **A generous prototyping area:** Although a breadboard with push-in capability is preferable, solder holes also work. The board should be able to accommodate connection to another microcontroller—the one to which you are applying the tests. Otherwise, you will need two boards.

- **One or more pushbutton or DIP (dual-inline-package) switches:** You can use these switches to

manually trigger events, such as selecting a mode, starting a data capture, injecting a one-off signal, or resetting data collection.

- **One or more potentiometers:** These components are worth adding to a board. You can use potentiometers as a simple analog stimulus or to dial in a selection to the testbench or the unit under test. For instance, you might use the pushbutton switch to select the test and the potentiometer to set the amount or duration of the test.

- **Several LEDs:** Use these lights to indicate pass/fail or state information from a test, as well as to signal the beginning or end of a test or state.

- **An LCD or some other form of display:** An LCD can provide more information than LEDs during testing. This ability is especially useful when prompting for action or parameters during a test, as well as for providing instantaneous and finer-grained feedback of key parameters. By employing the display in conjunction with a pushbutton and dial, you can easily devise a capable menu system.

- **A communications interface to a PC:** There is no substitute for simple, straightforward collection of raw data that you can send to a PC for later analysis. Until recently, this collection would occur via RS-232 serial communications with transceiver hardware. Because fewer and fewer PCs have serial

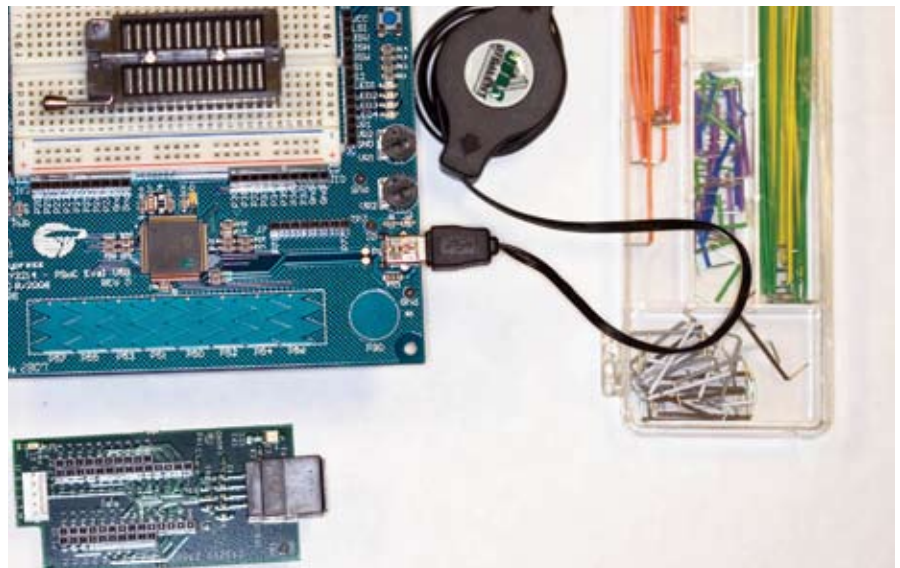


Figure 1 A low-cost evaluation kit, such as this Cypress CY3214-PSoCEvalUSB, can serve as a testbench for a range of devices.

ports, an evaluation board with USB comes in handy as long as the microcontroller vendor has tools to help the USB-novice succeed. A viable alternative is to use a USB-to-RS-232 adapter, which you can purchase for less than \$20.

These features meet the minimum requirements to support a workable general-purpose testbench. Most evaluation boards that meet these criteria will undoubtedly include other features that are intended to show off particular features of the microcontroller device. Even if your design does not currently use a feature, you can still include it in the testbench and learn how to use it. In the process, you may discover a useful implementation of the feature for a future design.

Depending upon your application, you may find it convenient to have two evaluation boards. One serves as a testbench that stimulates and monitors the main design in the device under test. This ability is especially important when the devices you are interested in are not readily available in through-hole packages, the prototyping area on the evaluation board is too small to support both the circuitry for the testbench and a device under test, or both situations occur. It is important that the device under test be as similar as possible to the unit you are designing into your end product with the exception of minor differences, such as package type and size. This requirement is not so critical for the testbench, although the more closely the device in the testbench/evaluation kit resembles the device in the end product, the more application-specific insight it can provide to the project at hand.

The only additional tools you should need are soldering equipment, jumper wires, and a PC. Soldering tools are useful for creating a permanent testbench, compared with temporary breadboarding, which you can continue to use once real hardware is available. In many cases, the low cost of an evaluation board makes it more cost-effective than creating a testbench from scratch. Vendors sometimes provide jumper wires for breadboarding with their evaluation boards but not always in sufficient quantity or variety of lengths. Finally, you'll need a PC that can run the development tools and interface to the testbench. If you are using RS-232 on your testbench and your PC does not have a serial port, remember to add a USB-to-RS-232-adaptor cable and to confirm its capabilities.

PRACTICE MAKES FASTER

The primary benefit of a testbench is to speed project design by allowing team members to begin integrating their firmware using the board as a reasonable hardware surrogate. Given access to the hardware and system requirements, they can construct a subsystem suitable for independently testing each subsection of a design. Thus, as soon as you design a function, you can implement, test, and verify it in a real-world environment with equivalent operating conditions. If everyone on the team follows this approach, the team members can begin integrating some or all of the functional areas together and run them on testbenches. When the real hardware arrives, final integration will be much smoother.

Using a testbench also provides a buffer against hardware delays. Because hardware is rarely completed on schedule, it delays initial testing. With a testbench, however, software de-

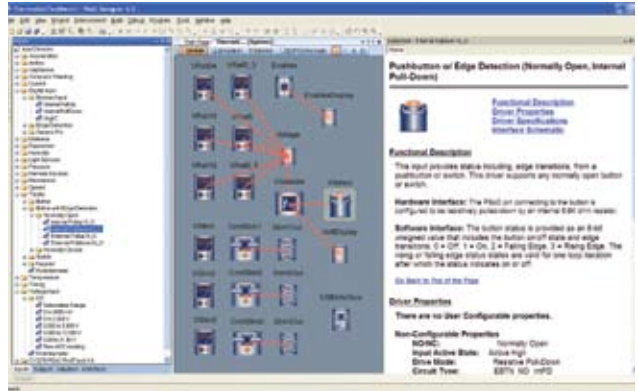


Figure 2 A typical PSoC Designer 5.0 screenshot shows the testbench in the middle; high-level functions on the left; and driver details, such as this pushbutton, on the right.

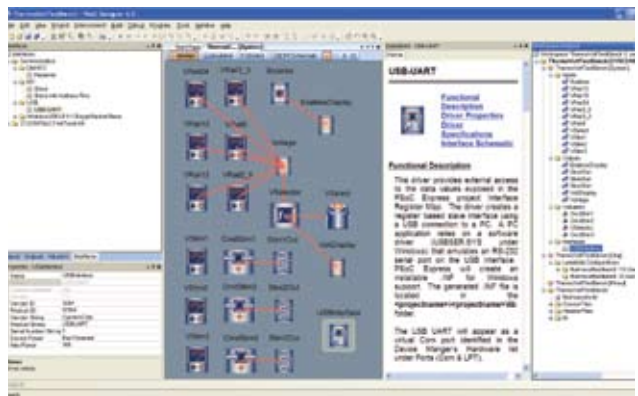


Figure 3 The USB-UART properties and data sheet generate a data-communications protocol that works with a standard Microsoft Windows driver.

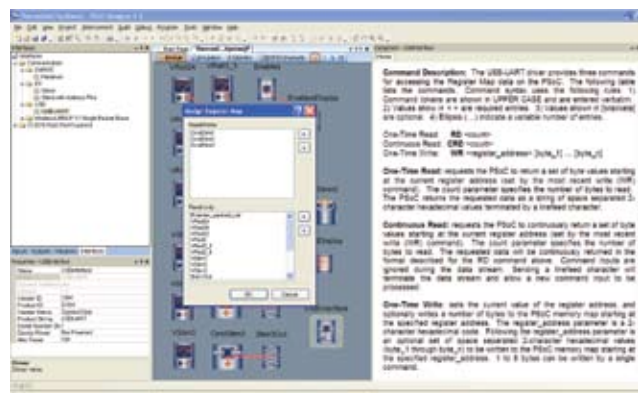


Figure 4 The testbench USB-UART-command protocol has a register map and driver.

velopers can use the delay in hardware availability to ferret out issues and better preserve the design schedule. Additionally, when hardware does arrive and problems arise, early familiarity with the system inspires a level of confidence and capability that allows the team to quickly isolate the cause of the problem. In many cases, the problem will likely be in hardware because software has been rigorously tested at this point. Finally, because real hardware tends to be in short supply when it does arrive, evaluation-board testbenches enable more team members to work concurrently. Such work in parallel results in faster overall development and the ability to meet deadlines.

Another consequence of using testbenches is increased quality of the project at hand, not just because the team is testing the firmware early and often, but also because, in working with the microcontroller to develop testbenches, team members must stretch themselves outside the bounds of their immediate feature or problem. Designing a testbench forces engineers to think about how the function should behave under all circumstances. Engineers must approach design from a preventive perspective with a testbench, rather than take a symptomatic approach to repair problems—as often happens when a system is complete before any real testing occurs. Working early on with this mindset strengthens the individual's skills with the microcontroller. These additional skills become critical to the project's success because, when a problem arises, designers have a greater array of tools at their disposal. More ideas will come from more people from all directions because each had his own testbench to play with.

Finally, if it puts the same rigor on the testbench developments as it puts on the rest of the project—that is, documenting the work and maintaining it in a configuration-management system—the team will have developed a set of testbenches that a formal test group can take advantage of. It can more quickly leverage these basic testbenches into better formal system-level tests. Also, because it can develop and test the functions from one designer with its own testbench, separate from other designers, the team has built a library of well-tested functions that it can

combine in many ways to speed a new or different project into production. As before, these regression tests and the hardware to run them are available for immediate use on new projects before any real hardware is available.

TESTDRIVING A TESTBENCH

An example shows how you can use an inexpensive evaluation kit as a testbench. The evaluation kit in this case is a Cypress (www.cypress.com) CY3214-PSoCEvalUSB (**Figure 1**). The evaluation board contains a Cypress PSoC device, CY8C24094, a special version of silicon that can emulate any member of the CY8C24x94 family of devices. Many evaluation boards include similarly flexible devices so that a single evaluation board can serve across a wider range of devices. You can program these devices as you would production silicon or use them with an in-circuit emulator to debug or emulate an application program. This feature is convenient and not uncommon for an evaluation kit. For the example, you could develop a testbench for a voltage-sequencing-and-monitoring application.

Note that, within families of microcontrollers, the differences between devices may be only the amount of memory available and peripheral choice; all other features are identical. Depending upon the portfolio a vendor offers, you'll have the option of selecting a device for your testbench with more features than you may be using in the device under test. For example, you might want to select a testbench microcontroller that supports USB even if your device under test does require USB for data transfer.

In this voltage-sequencing-and-monitoring example, the system has six voltage rails and six enables, one associated with each rail (**Reference 1**). The requirements for the testbench generally depend on what you need the system to do. The testbench in this example uses nine ADC inputs, three DAC outputs, a two-line LCD, seven digital inputs, and a full-speed USB interface. Six of the ADC inputs are for the six voltage rails, and six of the digital inputs are for the six enables. These inputs are the monitors for this voltage-sequencing-and-monitoring project. The three DACs handle voltage-signal injection and act as stimuli for test sequences. The other three

ADC inputs monitor the voltage that the DACs produce. One line of the LCDs displays the state of the enables, and the other line displays one of the six voltage rails, which you select with the pushbutton switch that is tied to the seventh digital input. The USB interface

lets you simultaneously access more data from the testbench and log it to a PC.

As assisting in software design continues to become a larger part of selling hardware, manufacturers often design microcontroller-evaluation kits to provide a fast, out-of-the-box experience that demonstrates a device's capabilities. As a result, development tools often include wizards, configuration GUIs, and graphical-design tools that enable engineers to quickly build designs whose development tools would have just a few years ago required days or weeks of manual learning. These tools also speed development of testbench software and interfaces. Additionally, code developed for a testbench often teaches a device's capabilities to engineers and provides some code that they can use in designs.

You can use these same tools, which facilitate fast evaluation, to create in a matter of hours a basic testbench, using high-level system functions that include all of the control and routing firmware necessary to implement the features you desire. You can also use demo or application code to provide core function blocks, such as stitching inputs and outputs together or creating a custom set of data registers that a USB virtual-communications-port driver can access using HyperTerm on a PC.

Figure 2 shows the complete design as a screen capture and divides it into three parts. On the left is the catalog of high-level functions, or drivers. On the right is the data sheet associated with the pushbutton driver. The middle shows the testbench with six voltage-rail inputs; three stimulus-voltage outputs; three stimulus-voltage inputs; the enable digital input; LCD line 1, which selects a label; and LCD line 2, voltage, which selects an actual reading to display. The USB interface is a drop-in configuration and protocol that works with a Microsoft (www.microsoft.com) Windows-standard driver.

➤ Go to www.edn.com/ms4299 and click on Feedback Loop to post a comment on this article.

➤ For more technical articles, go to www.edn.com/features.

Figure 3 shows more detail on the USB-UART driver. **Figure 4** shows the register-map ordering and more details of the USB-UART-command protocol.

Again, the benefits to your project of using microcontroller testbenches aren't just those you gain

by the testing, but also those you gain by using the microcontroller in a way other than the project requires. Getting started becomes easier, and deeper learning comes from continuing from this starting point to branch out into more specialized testbenches. With quick-start development tools, you can continue to dig deeper and customize the project or simply learn more about the microcontroller and the development tools by looking beneath the surface of the high-level abstraction to all of the automatically generated code available to designers.

Every new project has risks, but a microcontroller-testbench strategy is an effective way to reduce the risk of late hardware and speed hardware and software integration. Hardware cost is low, and the benefits extend far beyond simply providing better testing; they enable formal verification early in the design cycle, which increases the quality of the end product without negatively impacting the design schedule. The team will need to learn all of these capabilities at some point anyway. The more each team member knows about how the microcontroller works, about the development environment, and about each implementation, the better the design—now and in the future. **EDN**

REFERENCE

■ Buterbaugh, Ernie, "Power Management: Voltage Monitoring and Sequencing with PSoC and I²C," Cypress Semiconductor Application Note AN2379, Dec 13, 2007, www.cypress.com/design/AN2379.

AUTHOR'S BIOGRAPHY

Jon Pearson is the development-tools-marketing director for Cypress Semiconductor Corp. He has developed embedded-systems firmware for various microcontrollers in avionics and telecommunications equipment. You may contact him at jpx@cypress.com.