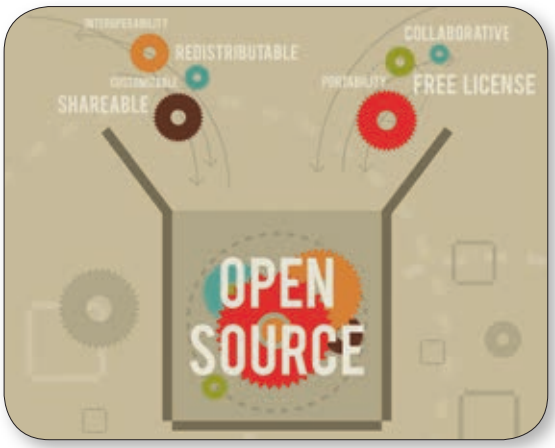
Open source vs. proprietary: an embedded hardware issue

By Michael Parks, P.E., and Lynnette Reese, Mouser Electronics

Fewer things ignite more passion in software geeks than the debate of open source versus proprietary code, but when going to market with an embedded system, sometimes you don't get a choice.



Source code has a lot to do with the concerns that businesses have concerning control of their destiny and future costs. Source code is a set of human-readable (mostly) instructions that tell a processor what to do. Before it can be used by a computer it must be compiled or interpreted into binary object code. If you have the source code, you can modify it and recompile it to meet your specific needs. You can also do an independent review to ensure there are no bugs or hidden backdoors. Operating systems can also be open source and therefore transparent, but work at a deeper level on the processor to provide interrupt handling, peripheral management, and more.

With open source, you are not beholden to another party to keep your product working. Proprietary-based systems, if they are not home-grown and thus belonging to you are open to the possibility of getting "orphaned" if the proprietary software is abandoned. For example, an operating system (OS) might be purchased from and maintained by another company, but what happens if that company gets bought by your competitor? Updates and bug fixes would be done to their schedule. Worse, you might be ordered to cease and desist using the software immediately. On the other hand, if you go with an open source compiler or operating system, you need to know what you are doing; open source is rarely turnkey in the embedded world. You also need to have reasonable expectations of what the open source maintenance community can do with respect to your need-for-speed in going to market, since open source may adopt a specific technology at a comparatively glacial pace. You can always roll up your sleeves and create and contribute to the open source community for what you need, but even then the acceptance into the formal repository (e.g. the maintained Linux "tree" for a specific embedded processor) may not proceed as you would like. If you do not contribute your changes back to the community, which is entirely legal, no one will help you maintain it but you, no matter how good you think it is (this is known as "forking" the code). Sometimes you don't have a choice in whether you use proprietary or open source, however.

A processor may not have an open source operating system, software, or tools. The immediate choice is to use a proprietary version, or if you have the time, start an open source tree or library on your own. It would be remiss not to include a mention about open source hardware (OSHW). OSHW is similarly "open," especially if source files for the schematics are provided, but the openness may stop at the doorstep of the SoC, processor, or a support chip on the board. OSHW can also be supported by proprietary development tools, as well, depending upon the processor. Nevertheless, an open source movement is afoot, inspired perhaps by Linux.

Both open source and proprietary software have equal shares of misconception. Further complicating matters for open source is the vast number of licenses that can confuse even the savviest technologist. The Open Source Initiative states, "Open source software is software that can be freely used, changed, and shared (in modified or unmodified form) by anyone. Open source software is made by many people, and distributed under licenses that comply with the Open Source Definition."

Some of the most popular open source licenses include the Creative Commons, Apache License, BSD, GNU General Public License (GPL), MIT License, and the Mozilla Public License. For a complete listing, you can head over here. Open source licensing varies, but a typical license ensures that if you build or expand on an open source project, you are given attribution and those who build upon your work must also release their code under the same license. This prevents someone else from unfairly profiting from your hard work, or at the very least, making it much harder to do so. Open source software is fundamentally about sharing and gaining wisdom from the crowd. Many larger,

Figure 1. Latest Texas Instrument (OSHW) LaunchPad features the MSP432 processor and is aimed at the portables, wearables, smart grid, medical, and automation control markets.

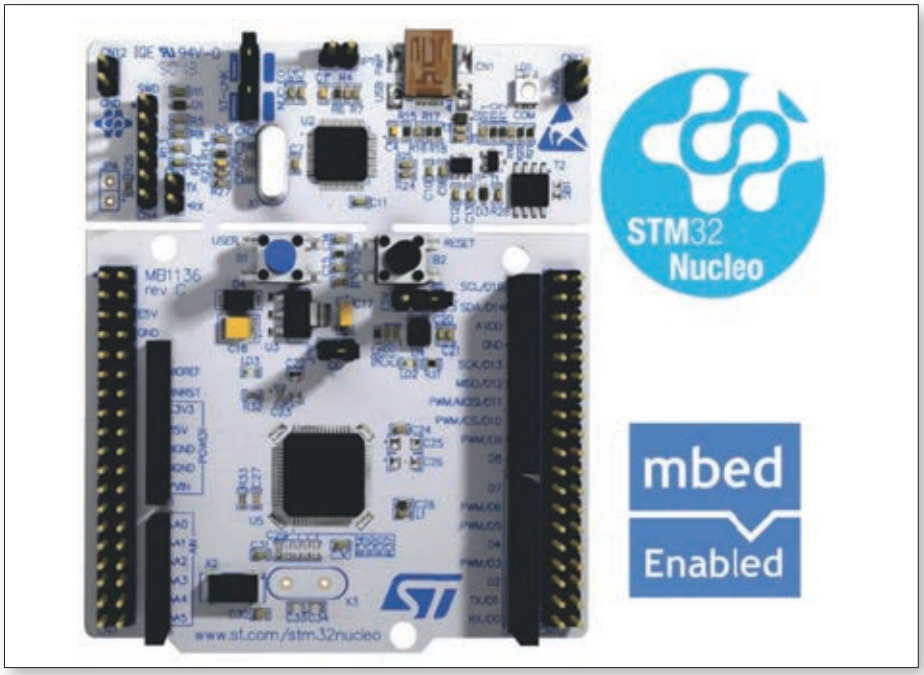


Figure 2. The OSHW Nucleo is about $10, includes free mbed.org software tools and is also supported by KEIL tools. STM32 Nucleo is compatible with most Arduino shields.

non-corporately backed open source projects rely on a small army of volunteers to develop, debug, and document source code. Some projects gather a significantly large enough community for developers to receive some modest amount of funding via donations solicited from end users. The collaborative nature of open source projects between the user community and developers can become quite a remarkable resource to entice new users and coders alike. Recognition for participating in an open source project is another alluring appeal for independent developers.

In contrast, with proprietary software the proprietor retains significantly more in legal rights by enforcing copyright restrictions. Typically, proprietary software is closed source with no access to the source code even when the application is purchased. The complete list of limitations is typically contained in an End User License Agreement or other legally binding document, and tends to involve restrictions on modification, sharing, redistribution, and reverse engineering. At the end of the day, proprietary software is about protecting Intellectual Property (IP).

It is important to note that open source does not necessarily mean cost-free, and proprietary does not necessarily mean that it costs money to use. Nor is it true that open source is only for amateur projects while professional products are all proprietary. However, should you select a proprietary platform, be sure to understand the licensing fee structure, if one exists, so that you aren't shocked with a bill when you are ready to go into production. Otherwise known as royalties, the fees associated with acquiring a license allow the developers to dedicate their time to refining and debugging their application-specific code. From a developer perspective, this means that proprietary OEM support should be prompt and the support will ideally be there for a long time (in some cases, a maintenance and support agreement must be paid on an annual basis).

Since commercial and industrial embedded systems tend to have lifespans that last many years, if not decades, the longevity of reach-back support can be crucial. Support for open source software might in some cases be more unreliable in comparison to proprietary. But at least you have access to the source code and can make changes if you have enough expertise, time, or money to hire someone to do it for you. With proprietary code, there is no legal basis to access source code or make changes without the originator's consent.

Just to make matters even more complex, it is also possible to have software that contains a mix of both proprietary and open source code. For example, Apple OS X operating system builds upon the open UNIX operating system. However, the windowing system that drives the OS X GUI is not open source. There are many embedded systems that have open source operating systems with custom adjustments. Google Android operating system source code is taken and customized by carriers to install on their authorized phones, which is legal but is not necessarily supported or maintained by Google. (This customization often results in bloatware.) Historically the embedded electronics world has been dominated by proprietary software, including things like the integrated development environment (IDE), real-time operating systems (RTOS), and firmware libraries. Platforms such as Arduino are attempting to change that, but many professional platforms remain locked behind proprietary license agreements. It's not just open source communities that are pushing for change, the U.S Department of Defense and NASA are also pushing for open software architectures to enable collaboration and data sharing and prevent vendor lock.

Some established embedded-platform OEM companies are beginning to respond to developer desire for more open source solutions. Others are at least rethinking the licensing. From IDE (a development tool) perspective, the open source Arduino IDE is rather spartan when compared to other platform IDEs such as Texas Instrument's Code Composer Studio, which includes many significant professional development features such as power consumption monitoring. TI is also supporting an open source product called Energia as an alternative, for various TI microcontroller products. STMicroelectronics STM32 Open Development Environment (STM32ODE) is another example of a company that is adapting to changing licensing models.

The STM32 Open Development Environment includes their OSHW Nucleo boards.In the end game, TI and STM are chip makers focused on providing cutting-edge integrated chips, but they also understand that silicon is not the whole embedded picture. Hardware is also undergoing an open source revolu-

tion. The discussion surrounding hardware is perhaps unsurprisingly more complicated. Software source code is granted copyright protection as soon as it's written down. Furthermore, nothing tangible comes from source code. In the hardware world, that isn't true. CAD design files or 3D printer STL files eventually get turned into a physical product. Does copyright apply to the design files only, or does it also cover the eventual tangible product? Even if conventional wisdom holds that copyright doesn't apply to hardware, these are uncharted waters that haven't been tested in the courts yet. There are specialized open source hardware licenses, with two of the most popular being the CERN Open Hardware License and the TAPR Open Hardware License.

With respect to the embedded development world, the choice of using open source or proprietary software is often driven by the hardware selection. If a microcontroller or FPGA that meets project specifications only comes from a vendor that has locked their code behind a proprietary license, then you really have little choice but to accept it. You might try to negotiate additional data rights, but not without adding significant cost to your budget. With that said, as hardware becomes more commoditized and multiple vendors offer similarly capable hardware solutions, the more software licensing may affect the balance of overall offerings and create differentiation. Therefore, it's reasonable to assume that as products become more commoditized, more vendors will offer choices on the software side of things. The licensing structure of the IDE, (RT)OS and libraries may be enough to sway developers to pick one platform over another. ■

For further information, visit the section Application & Technologies of www.mouser.com