Electronic Design

By ERNEST WORTHMAN, Contributing Editor

Reap the Rewards of Open-Source Software

Sponsored by Texas Instruments: Prioritizing software has become essential in design as embedded systems dominate the electronics ecosystem. And many designers are turning to open software to fulfill their needs.

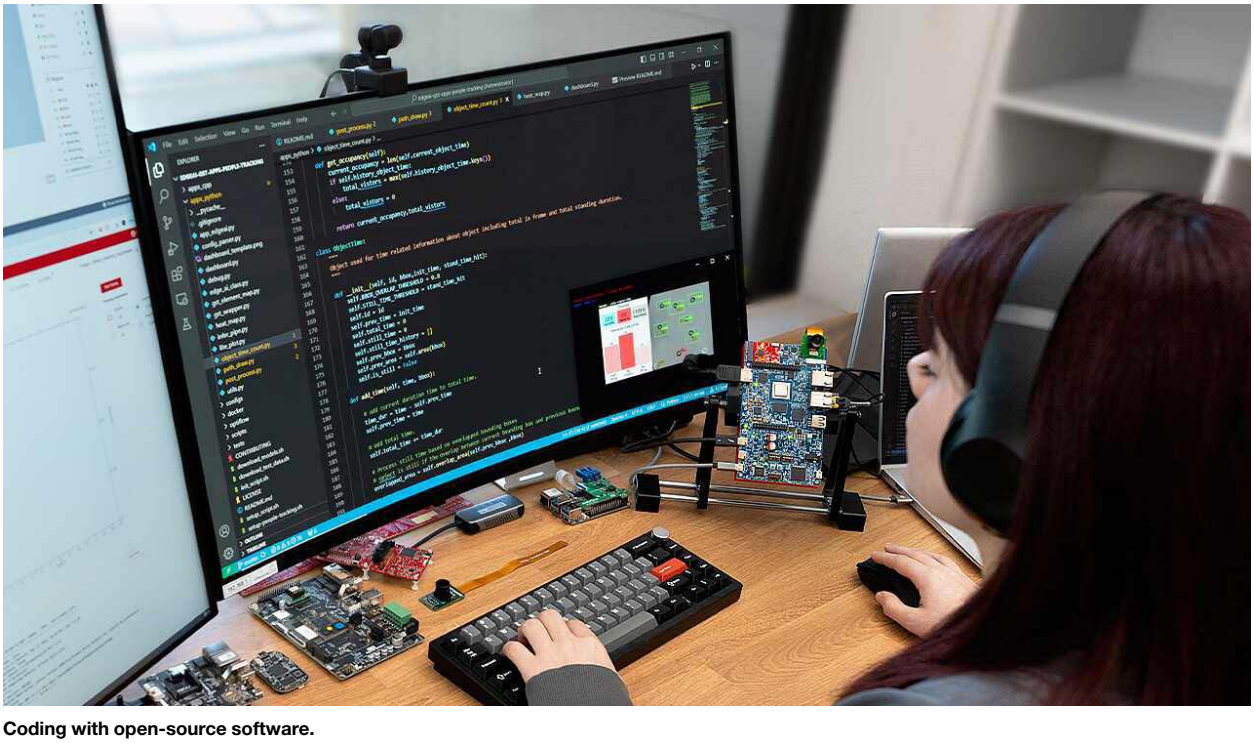
Embedded systems are emerging as a go-to solution for many projects. This rapid uptake in embedded interest is due, in large part, to advances in semiconductor technology and the advancement of open software. With such coalescence, many in the design community seek core software that's not proprietary and/or with a complex structure. The answer comes in the form of open-source software (OSS).

Today, there's a flurry of vendors offering embedded technology. Nearly all have their own proprietary software. However, this bounty of software doesn't bode well for embedded designs. A better solution is needed, one that's flexible, interoperable, and standard (and non-proprietary).

Open-source software has been around for decades. Today's evolving embedded sector is revisiting it in the face of the booming embedded device market as a simplified solution. This article discusses why OSS is rapidly becoming the go-to solution for many embedded designs.

Familiarizing Oneself with Open Source

Using open source in embedded systems benefits many stages of the manufacturing process. But the journey is not free of entanglements. For many, the open-system environment is unfamiliar territory.



Coding with open-source software.

Open-source code ensures interoperability among diverse applications.

Porting to OSS can be a challenge. Open-source software requires knowledge of, and working with, standardized code. It also requires a deep understanding of its structure, primarily because it can be modified and that can spell disaster if not properly understood.

Yet, even if one has some familiarity with it, there's much to know about implementing it. It is essential to work with a knowledgeable vendor, one that has a long history of development and a deep well of knowledge. As software pivots to becoming the first component of the design chain, finding experienced OSS vendors is imperative—especially as designers are increasingly pressed to use vendor-neutral options.

What is OSS?

There are three tenets of OSS. First, it has to be adaptable. Next, it needs to have standards. Finally, it must be interoperable. Let's drill down on these a bit.

Open-source software is source code developed and maintained through an open alliance of open-systems software developers. It's a community camp that collaborates to ensure accuracy, interoperability, applicability, rapid development, frequent bug fixes, and continuous updates. It's available for anyone to use, examine, alter, and redistribute as they see fit within its coding criterion. This gives it many advantages.

However, open-source code isn't a total panacea. The fact that its a collaborative work means that many hands are in the pot, which tends to limit the development of very advanced features and functions. This is the major disadvantage for developers who require advanced features beyond open-source code capabilities.

Open-source code can also have compatibility issues. Integrating open-source code into existing software structures may require additional effort to ensure seamless integration and interoperability. And it can't offer the same level of security as proprietary software.

And, of course, additional issues may arise in certain situations. However, if open source is plugged in at the beginning of the development cycle, disadvantages can be minimized.

Overall, however, the advantages offered by open-source code—cost-effectiveness, transparency, community collaboration, and customization flexibility—are of considerable value, particularly when working with embedded hardware.

Software Adaptability

Adaptability in software is of monumental importance. This enables designers to create or add new features and functions by modifying the code. As long as designers adhere to the code's parameters, there will be no errors. This works at any stage of design or production, even the final product.

A prime example of this is the software-defined vehicle (SDV). Today's vehicles have a diverse manufacturing base with thousands of vendors. This ecosystem relies on complex and expansive proprietary software to operate. Open-source software has the potential to seriously reduce soft-

ware complexity, often by orders of magnitude.

And, as the electric-vehicle (EV) environment expands, especially with autonomous iterations, software will manage even more of its functions. And this isn't relegated to just driver assistance; it will also manage all of the vehicle's directed operation (user input), hardware, and maintenance, as well as interconnect with external devices and systems. The less complex the software, the fewer chances there are for errors. Other use cases include Bluetooth, Wi-Fi, smart homes, and many low-power wireless platforms.

The Need for Standardization

Without standardization, the open-source ecosystem is nothing more than the wild west of code development. Standards form the bounds for what developers can do with it.

It's important to understand how standards interact with open source. Open-source camps and standards camps are critically intertwined. Such an alignment produces stable, open-source solutions, tested, and as bug-free as possible. That makes it important for developers to work with companies deeply entrenched in the open-source ecosystem.

Because of the power of advanced AI- and ML-driven computing, complex designs can be incorporated into the virtual realm before any piece of hardware is touched. However, initial software costs can make the upfront investment a bit pricey. Therefore, manufacturers must first understand the value of investing in OSS. However, payback will come in the form of a multitude of benefits.

Moreover, this isn't a time to cut corners or go with vendors that don't have a proven track record, as that can be a recipe for disaster. Vetting vendors is critical; once the transition is made, projects become optimized in cost, scalability, and intelligent design. In essence, OSSs can become the go-to tool for product differentiation and efficiency.

Conclusion

While open-source software offers many advantages in the design ecosystem, one must be cautious and think it through. And one must be prepared to refocus on the software-first philosophy. Sometimes it can be the complete solution, but in many projects, the solution lies with integrating both open and proprietary software.

However, if the choice is made to go with open systems, the results can be very rewarding. As has been mentioned often, the simplification of implementing open systems will reduce manufacturing complexity and improve time-to-market, realizing significant cost savings across the many parts of the design and production cycle.