Electronic Design®

By BRENDON SLADE, Director of the General Purpose MCU Ecosystem,
NXP Semiconductors, www.nxp.com

Community-Backed Open-Source Initiatives Help Unlock MCU Capabilities

In today's rapidly evolving landscape, open-source and community-backed initiatives are playing a crucial role in realizing the potential and versatility of MCUs to meet the ever-growing demands of modern applications.

Developers throughout the world know the value of microcontrollers (MCUs). From their first introduction to the simplest of applications, MCUs have followed a never-ending course of feature advancement. Today, such devices underpin all aspects of our lives, from industrial applications such as automation and the industrial Internet of Things (IIoT) to personal applications like health trackers and smart home devices.

Engineers developing solutions utilizing MCUs face several technical barriers slowing the deployment rate—variations in development platforms, the sheer breadth of hardware devices, and the constant demand for increasing levels of functionality are all constant challenges.

The current market only amplifies these issues. With many hardware solutions evolving incredibly quickly, new standards emerging constantly, and novel applications appearing every day, bringing a product to market quickly is vital for its success. But what can we do to mitigate these issues and help engineers deliver innovative solutions at the speed the market demands?

The answer is often a more open approach, with collaboration being essential, utilizing open-source solutions like the Zephyr OS. But with so many options out there, what solutions are proving popular and delivering the gains needed?

The Challenges of Developing Embedded Software

Many factors affect the challenges associated with embedded software when it comes to MCU-based designs. Issues such as limited code portability, lack of open or easily accessible support systems, proprietary platforms, and fragmentation of development flows can all hinder the design process.

For a singular project, those factors slow teams down, with engineers having to dedicate time getting to grips with new environments and hitting walls when the limited support systems fail. But the issues are compounded further when developing derivative or follow-on products, where engineers essentially must repeat tasks due to variations in the generations of underlying MCU platforms.

Perhaps the most easily recognizable challenge for many MCU software developers is being locked into one platform. Development is often trapped with a single vendor. And even when porting is available, issues can occur, resulting in being reliant on that single vendor's support for assistance.

With embedded hardware naturally limited in capacity, there's also no place for largely un-optimized code. Even in instances where enough resource headroom allows code bases to be ported across products, nuances created by hardware dissimilarities can often mean large swathes of the original code are no longer fit for purpose.

Proprietary or closed software solutions rely on support from the originating source, be that a software provider or a semiconductor company. With the wealth of MCU applications continuously expanding, silicon vendors might struggle to supply enough reference code and documentation to support every project. For smaller customers, there might not be anywhere to get reference code to meet their needs at all.

While hardware processing power and memory have expanded, the demand for greater functionality means developers must not only produce more code than ever before, but it must also be optimized and with the capability to be updated and maintained. For many developers, open, community-backed solutions with widespread vendor support and good engagement can have a significant positive impact on their projects, providing a greater pool of collective resources and ongoing enhancements to take advantage of the latest hardware.

Open-Source and Real-Time Operating Systems

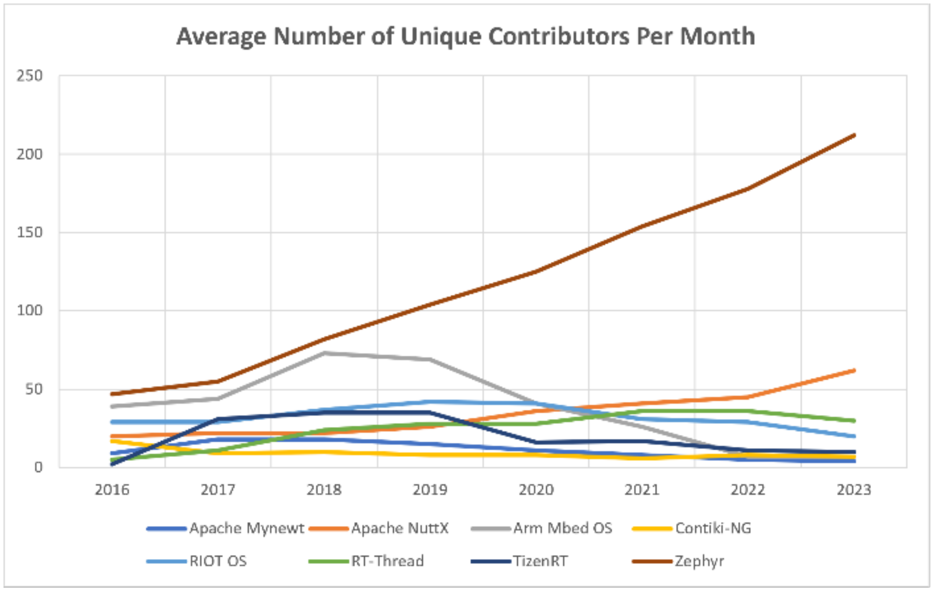
As the complexity of many MCU implementations keeps evolving to add increased levels of functionality, the embedded software has advanced. Historically, MCU developers implemented software applications around state machines and sequential processing loops. However, as the number of parallel tasks increases, this approach needs to change.

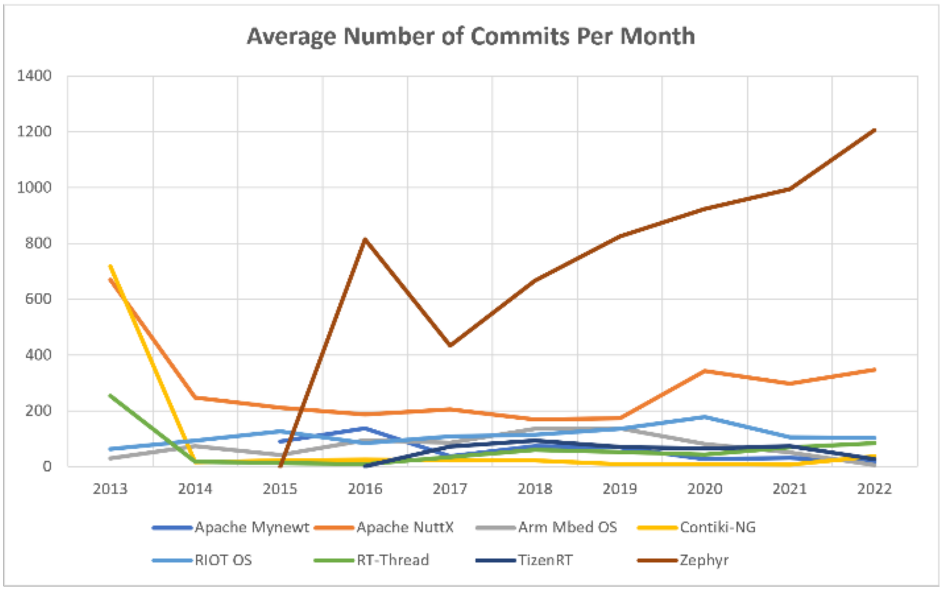
More MCUs now utilize real-time operating systems (RTOSes), allowing for greater complexity, more interrupt sources, and expansive communication interface options. Use of an RTOS is becoming increasingly essential as time-critical tasks from different elements of a system are more likely to conflict with each other. One example is a controller application with a GUI that's simultaneously managing a wireless stack and running control algorithms on a single MCU.

With the complexity of code on the rise, open-source, community-backed solutions like Zephyr are coming to the fore. Built from the ground up to meet the latest demands of MCU deployments such as edge solutions and IoT, Zephyr is headed by a panel comprising not only leading silicon vendors, e.g., Intel and NXP, but also powerhouse OEMs like Google and Meta. The project has already seen major application contributions from these OEMs, such as Google's submission of a complete compute platform code base.

Zephyr's functionality is designed to meet the many prominent challenges head-on with high flexibility, modular design and hardware abstraction, making it easy for developers to get started and share support across its growing community. It features the highest number of contributors and upstream commits per month of any other RTOS *(Fig. 1)*, demonstrating the platform's ability to solve and adapt to changes.

Zephyr's rapid development and extensive support network are delivered largely by the collaboration of this ex-





1. Zephyr has a higher average number of unique contributors and commits per month. (NXP, Source data: drawn from GitHub data)

pansive community of developers and manufacturers. This same community provides an extensive pool of reference code, working examples, and platform support.

In terms of the architecture, the small kernel and system scalability allow for deployment on a wide range of hardware devices with over 450 boards currently supported. Security is also central to the platform, with the working group heavily focused on this aspect. Once again, the Zephyr community acts as a self-checking mechanism for any potential security concerns.

While many RTOS platforms are available, the open nature and size of the community that supports Zephyr is leading to quickly increasing popularity among embedded developers. Its ability to rapidly progress and support innovations across a range of disparate platforms, while reducing the workload of software engineers, is incredibly appealing. But Zephyr is just one aspect of MCU software development, with other key components needed.

Trapped Inside IDE Platforms

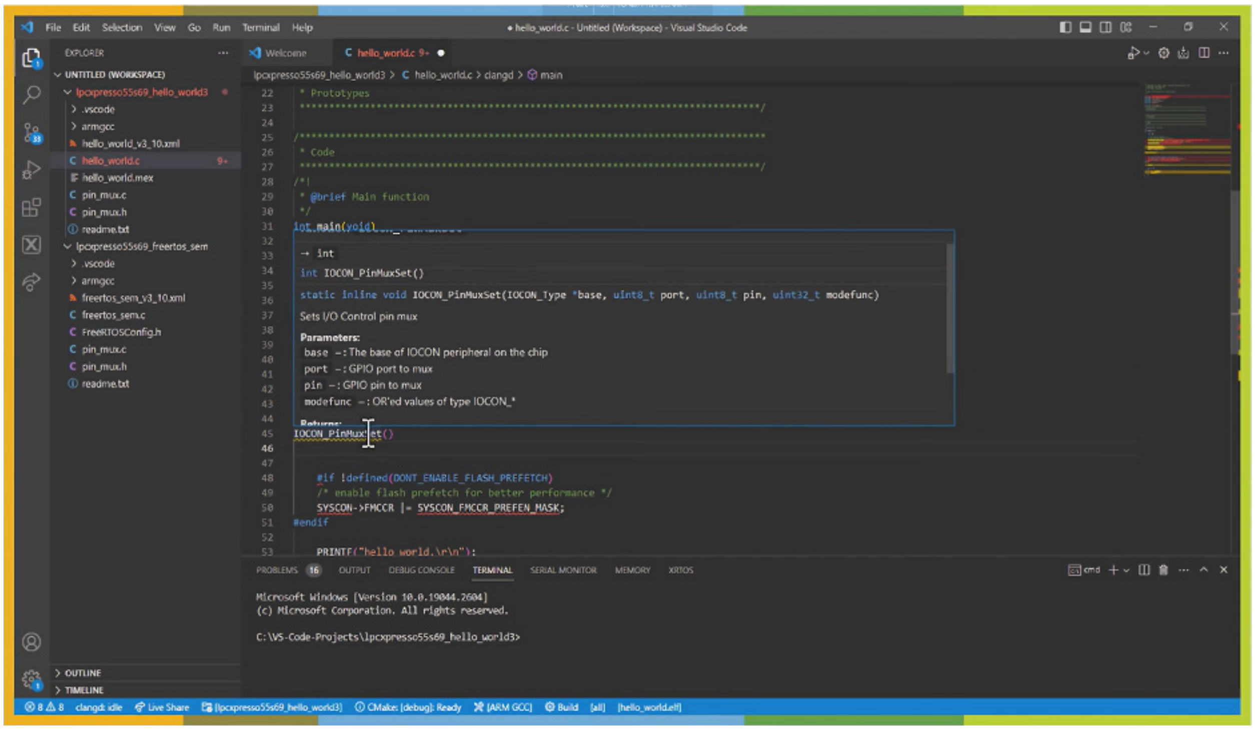
Many MCU vendors utilize Eclipse to deliver free tools to their customers. As a versatile IDE platform, it can be customized to suit their needs accordingly, but its Java-based core can consume lots of resources on its host platform, making it frustrating to use for some.

Although Eclipse is based on open source, the customizations usually need to be controlled by silicon vendors, inherently leading to a quite different experience from one MCU to another. In these instances, Microsoft's lightweight and fast IDE Visual Studio Code (VS Code) is a popular alternative. Since VS Code is basically a highly extensible, very powerful editor, it better lends itself to being customized by the developer and can be set up to easily jump from command line to GUI-based operations.

This also makes it inherently suitable for custom-build systems, like that used by Zephyr. Such flexibility makes it possible for developers to work with more than one MCU vendor's products in the same environment, rather than jumping between two very different flavors of Eclipse.

Despite all of the power and benefits of these modern IDE options, some customers have additional needs for safety certification, specialist compilers, or have long-term investment in premium development tools. Seeing the need for a broader solution and the value of collaboration, and working closely with key technical partners, NXP developed a novel approach.

Under the umbrella of its MCUXpresso ecosystem, NXP enables not only its own branded IDE interfaces with a software development kit (SDK) and a configuration toolset, but also industry-leading tool partners. IAR and Arm Keil are two of the most prominent names in development tools, each with their own premium IDEs that provide debugging, compilers, and safety certification. Both are also key technical partners for NXP, working to ensure their own tools can



```
 * Prototypes
 ******************************************************************************/

/*******************************************************************************
 * Code
 ******************************************************************************/
/*!
 * @brief Main function
 */
int main(void)
```

```
→ int
int IOCON_PinMuxSet()
static inline void IOCON_PinMuxSet(IOCON_Type *base, uint8_t port, uint8_t pin, uint32_t modefunc)
Sets I/O Control pin mux
Parameters:
 base –: The base of IOCON peripheral on the chip
 port –: GPIO port to mux
 pin –: GPIO pin to mux
 modefunc –: OR'ed values of type IOCON_*
Returns:
```

```
IOCON_PinMuxSet()


    #if !defined(DONT_ENABLE_FLASH_PREFETCH)
    /* enable flash prefetch for better performance */
    SYSCON->FMCCR |= SYSCON_FMCCR_PREFEN_MASK;
#endif

    PRINTF("hello world.\r\n");
```

```
Microsoft Windows [Version 10.0.19044.2604]
(c) Microsoft Corporation. All rights reserved.

C:\VS-Code-Projects\lpcxpresso55s69_hello_world3>
```

2. This is an example of an MCUXpresso workspace within Microsoft's Visual Studio Code. (NXP)

be used right out of the box with MCUXpresso SDK software packages.

MCUXpresso works closely with these qualified partners to develop a multilayered functional development platform that utilizes some of the most advanced and open methodologies available to MCU development today. For Eclipse users, it delivers an easy-to-use Eclipse-based IDE for NXP MCUs based on Arm Cortex-M cores, providing advanced editing, compiling, and debugging features.

Recognizing the potential of Visual Studio Code, NXP has worked closely with Microsoft, and after a highly successful beta, the newly released MCUXpresso for Visual Studio Code was launched in July 2023 as a free-to-use extension for Microsoft's popular tool *(Fig. 2)*.

The extension features complete support for the MCUXpresso software drivers and middleware, providing fast and responsive coding using the MCUXpresso SDK. It's complemented by an easy-to-use installation manager to remove the complexities of preparing VS Code for use with NXP devices. Collaboration between NXP and its community of partners and beta testers has created an optimized experience for Zephyr-based development flows, helping to make the powerful Zephyr ecosystem more accessible to new users.

Middleware Challenges

As complexity and demand for functionality have increased, so too has the importance of middleware in MCU solutions. These software libraries consist of ready-to-run code for tasks such as graphics processing, network communication, USB device enumeration, audio capabilities, and even advanced features like machine learning and AI.

While some of these elements are available from open-source projects, other specialist software technologies are too new or specialized to be available or mature enough to use in products. Trying to incorporate third-party middleware into a project can be extremely challenging due to integration issues. In these cases, engineers historically had to undertake manual intervention to restructure the code base and modify build flows, wasting valuable time both during third-party code evaluation and in actual product development.

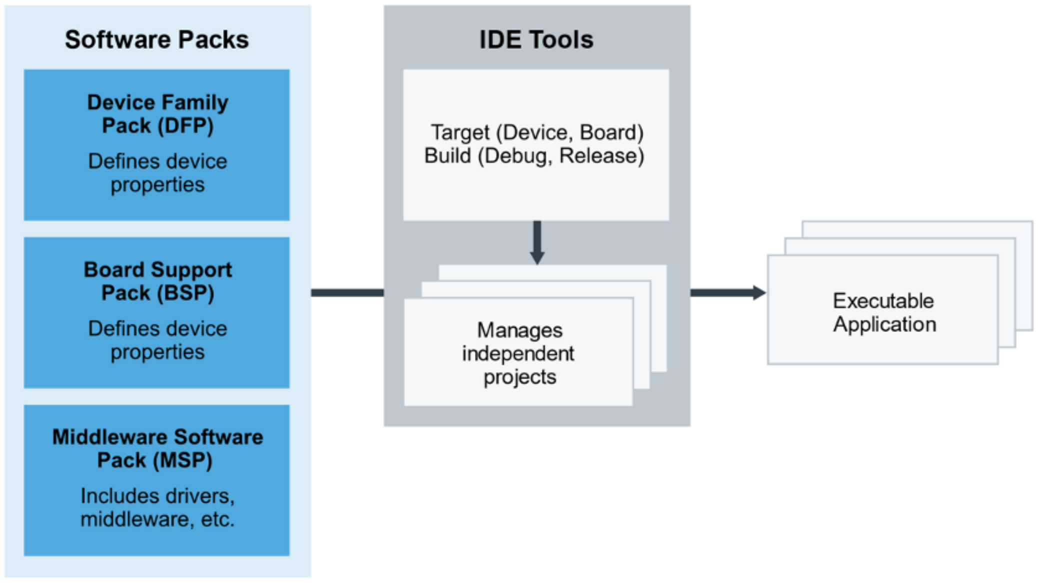
A new solution to this barrier is Open-CMSIS-Packs (OCP). This open standard defines a mechanism for packaging software such that compliant IDEs can automatically identify the content type, add the necessary files to a project, configure supporting tools, and update build parameters *(Fig. 3)*. OCP is supported by the entire NXP MCUXpresso ecosystem in keeping with the ethos of supporting open-source solutions, with backing from platinum IDE partners IAR and Arm.

This support enables the simple integration of software components from specialist middleware providers such as Ametek Crank, Embedded Systems Academy (EmSA), Memfault, Segger, TARA, and WolfSSL, covering a broad range of applications from graphics and communications and security stacks to device management and RTOSes.

The Open Approach

By creating a solution that's designed whereby developers can easily move between MCU platforms in a way that fits the best price and performance for their end products, projects like Zephyr and MCUXpresso are removing many longstanding barriers.

The inherent hardware abstraction layer that comes as part of Zephyr provides standardization of most APIs across the portfolio of MCUs from NXP, and into some aspects of i.Mx processors. This allows for code to be rapidly ported



Software Packs
Device Family Pack (DFP)
Defines device properties
Board Support Pack (BSP)
Defines device properties
Middleware Software Pack (MSP)
Includes drivers, middleware, etc.
IDE Tools
Target (Device, Board)
Build (Debug, Release)
Manages independent projects
Executable Application

3. Open-CMSIS-Packs are a key component of Middleware Software Packs (MSP) and vital to MCU embedded software development. (Source: NXP)

across the entire portfolio, opening up a vast range of optimal hardware choices across different architectures without having to worry about rewriting code.

Whether we're talking about RTOS, IDE, or middleware, providing open choices encourages broad-based momentum, further propagating their success by expanding the pool of reference code and expertise within the community support network. There's a reason more community-based solutions are becoming popular, and it's largely due to the level of ongoing investment needed to handle the complexity and rate of development.

When having to cover many MCU solutions, that can easily far outstrip what one company can supply. Furthermore, OEMs are taking greater risk in their software investment by relying on a single MCU vendor with proprietary code base.

An example of the successful use of open technologies comes from Vestas Wind Systems A/S, a Danish manufacturer of wind turbines. In their installations, real-time MCU performance and reliability are paramount, as is the ease of integration given the number and complexity of the systems at play.

The company chose to deploy Zephyr in its wind turbines, thanks to its open-source nature and strong community backing, which makes effective technical solutions readily available. Vestas' engineers contributed to the Zephyr community, using a combination of MCUXpresso SDK and existing Zephyr drivers from NXP. This combination provided the Vestas engineers with quick development support for the wide set of NXP Kinetis peripherals, further accelerating their prototype and hardware validation. Vestas is now well positioned to leverage NXP's newer devices, with the ability to rapidly update its platform for higher performance and extended features.

Collaboration and an Open-Source Future

In the rapidly evolving digital world, functionality is key, without compromising reliability and security. For MCU developers, meeting this target at the speed needed to match market expectations in time is quite the task. While, typically, MCU ecosystems have been closed or vendor-specific, it's clear that just as with the microprocessor market before it, the future will move toward open-source methods.

By looking at solutions like Zephyr, complemented by NXP's initiatives for MCUXpresso and its flexibility in IDE choice, the benefits are already proving popular and will effectively self-perpetuate. As the adoption of open source grows, so does the community. And with that comes more shareable code and greater collective support, further setting it apart from more closed solutions.

Collaboration is a powerful tool that many in the industry already value, as can be seen with the success of clean-energy innovators Vestas and major code contributions to consumer platforms from the likes of Google. Such teamwork in software delivery and open-source platforms provides the freedom to design in rapidly innovative and complex ways.

By enabling customers to leverage both open-source and specialist middleware with modern and flexible tool choices, NXP is establishing a simplified, quick, yet accurate and reliable, method of creating innovative production-grade devices.

Brendon Slade is the director of the general purpose MCU Ecosystem team. He has more than 25 years of experience in the DSP and microcontroller industry, where he has held design, applications, and technical marketing roles serving industrial, mobile, automotive, voice communications, and audio processing markets. His team focuses on enablement of NXP's Arm Cortex-M based MCUs, working with partners and NXP's internal software teams to define and deliver complementary development tool and software solutions. A graduate of the University of Plymouth, UK, he lives in Sunnyvale, California and holds patents in debug technology.