Announcing general availability of Zephyr 3.4

On behalf of the Zephyr community, it's my pleasure to announce that as of today, Zephyr 3.4.0 is generally available.
Before diving into some of the notable changes and additions to this release, let's take a look at some key numbers for this release.

Zephyr 3.4 by the numbers

- 491 contributors, including 167 new contributors.
- 500+ boards
- Commit velocity: 1.94 commits/h (5023 commits since v3.3.0).
    - 80k+ commits since day 1 (Apr. 10, 2015)
- 228K lines of code added
- 91 maintainers.

This new release is a testimony to Zephyr's increased adoption for a wide variety of applications. More and more companies are using Zephyr for building embedded controllers—microcontroller-powered applications that support a computer in handling low-level system tasks—and some of the enhancements in 3.4 are helping streamline efforts in this area. For example, Zephyr 3.4 is adding new APIs and driver implementations to interact with NVMe disks, SMBus peripherals, and real-time clocks in a uniform way.

Zephyr 3.4 also introduces several improvements to its built-in testing framework (Twister) that make it possible to write more comprehensive tests than in previous versions. Developers can now use popular third-party testing frameworks such as pyTest, GoogleTest, and RobotFramework, to write end-to-end tests running on real or emulated hardware, and potentially connecting to e.g. IoT servers.

If you'd like to see some of the highlights of this release in action, I've assembled a short video going through some cool examples and demos.

<< VIDEO >>

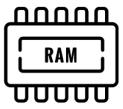
If you'd rather read through them (note: you'll find the full release notes here), here we go:

New & Noteworthy

New types of peripherals supported

Auxiliary displays

NVMe disks & controllers



Retained memory

SMBus

Real-time clocks (RTC)

Auxiliary displays

Auxiliary Displays are text-based displays that have simple interfaces for displaying textual, numeric or alphanumeric data. You usually want to interact with these by sending *characters*, as opposed to *pixels*. The newly added Auxiliary Text Display API enables you to do just that. Several drivers for commonly found auxiliary displays (from Hitachi, Noritake, Jinghua, …) are already available today. A code sample is available here.

NVMe disks

NVMe (Non-Volatile Memory Express) is a high-performance storage protocol that's designed specifically for NAND flash memory (think: Solid-State Drives (SSD), M.2 cards, …). As of 3.4, NVMe controllers and disks are supported, and fully integrated with Devicetree to make configuration/customization XXX.

Real-time clocks

Real-time clocks are low-power, often battery-powered, devices used for measuring the passage of time, including when the main system might be powered off. Zephyr 3.4 adds support for RTCs as first-class citizens, providing a consistent way to interact with them in a hardware-independent way. Beyond basic clock get/set interactions, the API also enables setting alarms or calibrating the clock, should these features be supported by the underlying hardware. Drivers for popular RTC chips such as NXP PCF8523 and Motorola MC146818 are already available.

Retained memory

The newly added retained memory API enables applications to read and write data to memory areas (ex. uninitialized RAM section) or devices that retain information while the device is powered. It's a good alternative when one does not want to rely on non-volatile storage for e.g. sharing information between different applications, or within a single application without losing state information upon device reboot.

SMBus

SMBus (System Management Bus) is a two-wire bus derived from I²C and often used fro communication with low-bandwidth devices on motherboards, ex. to get information from temperature sensors, battery fuel gauges, etc. As of Zephyr 3.4, the new SMBus subsystem allows developers to manipulate SMBus controllers and devices in their applications.

New input subsystem

The input subsystem provides an API for dispatching input events from input devices to the application. It provides a higher/common level of abstraction for handling input events corresponding to keys/buttons pressed, a touch display being pressed, etc. Among other things, his makes it easier to write graphical user interfaces independently from how "inputs" are handled at the hardware-level. This is also a great opportunity for developers to look at leveraging Zephyr's built-in state machine framework to handle more complex interaction scenarios.

New retention subsystem

Complementing the added support for retained memory, the new Retention subsystem is integrated with the Devicetree to easily configure and customize how data may be retained, including the ability to create several partitions, verify data integrity through checksums, or handling the special case of a device being rebooted, ex. to have it run a different application.

Testing framework (Twister) improvements

Twister, Zephyr's own testing framework, is widely used internally when "dogfooding" and making sure that Zephyr itself is well-tested. In fact, for every single pull request made to the Zephyr repository (assuming it has code changes), our CI jobs trigger Twister and run literally thousands of unit tests.
Zephyr 3.4 adds many improvements to Twister, making it even more suitable for complex functional testing. Developers can now use popular third-party testing frameworks such as pyTest, GoogleTest, and RobotFramework to write end-to-end tests running on real or emulated hardware, and potentially connecting to e.g. IoT servers.

Updates to Zephyr SDK

It is recommended to update to the latest version of the Zephyr SDK (0.16.1). One of the main benefits of the Zephyr SDK is that it's a one stop shop for getting all the toolchains and host tools that you may need for your Zephyr day-to-day development. As the SDK has grown in size over the year, it's worth noting that this newest version is up to 2x smaller (and hence 2x faster to download) since it is now packaged using tar.xz (Linux/macOS) and 7zip (Windows) as opposed to .tar.gz and .zip before.

Snippets

The newly added "snippets" help streamline all the common configuration settings (ex. configuration files, Devicetree overlays) that one may need to re-use across various projects. A typical use case would be to pack all your favorite debugging options (ex. enabling the shell, custom log levels, etc.) into a snippet so that you can easily instrument an application that needs troubleshooting, including changes that may be needed at the hardware definition level (ex. enabling a Zephyr shell over a USB interface).

Other notable APIs introduced

Memory Barriers

A new API has been introduced for data memory barriers. Data barriers are essentially a way to nicely tell your processor: "Hey, I know you like to rearrange tasks for efficiency, but these particular memory operations need to happen in the exact order I've given them!". This is

particularly useful in Symmetric Multi-Processing (SMP) scenarios, but can also be needed in multi-threaded applications or when hardware is accessed asynchronously.
The new barrier API enables a more consistent way to implement synchronization fences, regardless of the processor architecture.

Bluetooth 5.4 (but not only!) additions

Bluetooth Core Specification version 5.4 was released earlier this year on Feb. 7, 2023, and this new version of Zephyr already supports pretty much every feature added to the standard, namely:

- Encrypted Advertising Data (EAD), which enables secure broadcasting of data in Bluetooth LE advertising packets ;
- Periodic Advertising with Responses (PAwR), a feature that allows Bluetooth Low Energy devices to perform energy-efficient, bi-directional, communication in a large-scale one-to-many topology. Combined with EAD, this can prove very useful for applications such as electronic shelf labels ;

Other notable changes on the Bluetooth front include support for:

- Common Audio Profile (CAP) Unicast ;
- Telephony and Media Audio Profile (TMAP) – Bluetooth is particularly popular for all things telephony, so it's great to see that initial support for Bluetooth LE Telephony and Media Audio Profile (TMAP) was added ;
- Mesh – Experimental support was added for recent working drafts of Mesh Protocol 1.1, Mesh Binary Large Object Transfer Mode 1.0, and Mesh Device Firmware Update Model 1.0.

New boards and sensor drivers

- Over 30 additional boards are supported compared to the previous release, including the Arduino GIGA R1 WiFi, Seeed Studio's Wio Terminal and XIAO BLE, ESP32-S3 dev kit, and more..
- Drivers for dozens of sensors (environmental sensors, IMUs, current sensors, etc.) have been added, and it's now over 150 sensors that are not only supported out-of-the-box in Zephyr, but often tightly integrated with the Zephyr stack, for example to leverage power management features.

Meet our release engineers for Zephyr 3.4

It takes a village to deliver an open-source project such as Zephyr on time every quarter, and our release engineers are instrumental in making this happen. I've asked our two release engineers for 3.4, Anas Nashif from Intel and Joshua Lilly from Meta to tell us what's their highlight for this new release.

- Anas: “The amount of changes, improvements, addition of new features and innovations to the ecosystem we manage to deliver between two Zephyr releases and a development time spanning 4 month is amazing and shows how the community is marching toward one goal and such milestone is an attestation of great things to come out of the Zephyr project as we see more contributors and members join the project. This release is just a milestone and we are just getting started.”
- Joshua: “My favorite thing about 3.4 is the sheer amount of bug fixes and stability improvements that were made to Zephyr, for example all the work that has been done to make pthreads more robust by having them rely on Zephyr primitives.”