

PIGWEED

Q Search

Home

Getting Started

Source Code ↗

Code Reviews ↗

Mailing List ↗

Chat Room ↗

Issue Tracker ↗

Contributing

Code of Conduct

Embedded C++ Guide

Style Guide

Automated Analysis

OS Abstraction Layers

Targets ▾

Build System ▾

Size Optimizations

FAQ

Module Structure

Module Guides ▾

Third Party Support ▾

OS Abstraction Layers

Pigweed's operating system abstraction layers are portable and configurable building blocks, giving users full control while maintaining high performance and low overhead.

Although we primarily target smaller-footprint MMU-less 32-bit microcontrollers, the OS abstraction layers are written to work on everything from single-core bare metal low end microcontrollers to asymmetric multiprocessing (AMP) and symmetric multiprocessing (SMP) embedded systems using Real Time Operating Systems (RTOS). They even fully work on your developer workstation on Linux, Windows, or MacOS!

Pigweed has ports for the following systems:

Environment	Status
STL (Mac, Window, & Linux)	✓ Supported
FreeRTOS	✓ Supported
Azure RTOS (formerly ThreadX)	✓ Supported
SEGGER embOS	✓ Supported
Baremetal	<i>In Progress</i>
Zephyr	Planned
CMSIS-RTOS API v2 & RTX5	Planned

Pigweed's OS abstraction layers are divided by the **functional grouping of the primitives**. Many of our APIs are similar or **nearly identical to C++'s Standard Template Library (STL)** with the notable exception that we do not support exceptions. We opted to follow the STL's APIs partially because they are relatively well thought out and many developers are already familiar with them, but also because this means they are compatible with existing helpers in the STL; for example, `std::lock_guard`.

Time Primitives

The `pw_chrono` module provides the building blocks for expressing durations, timestamps, and acquiring the current time. This in turn is used by other modules, including `pw_sync` and `pw_thread` as the basis for any time bound APIs (i.e. with timeouts and/or deadlines). Note that this module is optional and bare metal targets may opt not to use this.

Supported On	SystemClock
FreeRTOS	<code>pw_chrono_freertos</code>
ThreadX	<code>pw_chrono_threadx</code>
embOS	<code>pw_chrono_embos</code>
STL	<code>pw_chrono_stl</code>
Zephyr	Planned
CMSIS-RTOS API v2 & RTX5	Planned
Baremetal	Planned

CONTENTS

Time Primitives

System Clock

Synchronization Primitives

Critical Section Lock Primitives

Thread Safe Mutex

Interrupt Safe

InterruptSpinLock

Signaling Primitives

Thread Notification

Counting Semaphore

Binary Semaphore

Threading Primitives

Thread Creation

Controlling the current thread

Execution Contexts

On naming

Construction & Initialization

Kernel / Platform Initialization

Before C++ Global Static

Constructors

Roadmap