# C++ Modules in a Microcontroller

Paulo Martinez, Siemens

C++ Under the Sea, Lightning Talk, 15. October 2026

# Previously last year...

- pauloacmartinez/cpp_binary_size
- STM32F4 (Cortex-M4, 1MB FLASH)
- Hello world with `<iostream>` ≈ 174KB
- Hello world with `<print>`: ≈ 325KB
- Since then:
  - → GCC 15 for ARM (12/2025), CMake 4.x
  - → **`import std`**
  - → better module support

# Part 1: `import std` on a microcontroller

# Experiment with modules

```
export module experiment_module;

import std;

export namespace experiment_module {
void __attribute__((noinline)) experiment() {
    constexpr auto str = "Hello World!";
    std::cout << str << std::endl;
    // alternatively: std::println(str);
}
}  // namespace experiment_module
```

# gcc-arm-none-eabi 15.3: libstdc++.modules.json

```
{
  "version": 1,
  "revision": 1,
  "modules": [
    {
      "logical-name": "std",
      "source-path": "../include/c++/15.3.1/bits/std.cc",
      "is-std-library": true
    },
    {
      "logical-name": "std.compat",
      "source-path": "../include/c++/15.3.1/bits/std.compat.cc",
      "is-std-library": true
    }
  ]
}
```

# First Roadblock

- ”CXX_MODULE_STD” on target ”experiment” requires CMAKE_CXX_STDLIB_MODULES_JSON

  → Embedded toolchains require `CMAKE_SYSTEM_NAME==Generic` and\or `CMAKE_CXX_COMPILER_FORCED==TRUE`.

  → Also skips module manifest detection.

  → Find JSON manually with `g++ -print-file-name=libstdc++.modules.json`.

# gcc-arm-none-eabi 15.3: std.cc

```
module;
...
export module std;
...
// 33.7 <condition_variable>
export namespace std
{
  using std::condition_variable;
  using std::condition_variable_any;
  using std::cv_status;
  using std::notify_all_at_thread_exit;
}
...
```

# gcc-arm-none-eabi 15.3: std.cc

```
...
// 33.5.2 <atomic>
export namespace std
{
  ...
  using std::atomic_flag_wait;
  using std::atomic_flag_wait_explicit;
  ...
}
...
```

# Second Roadblock

- `std.cc`: error: 'mutex' has not been declared in 'std'
- same for `jthread`, `condition_variable`, `latch`, `semaphore`, ...

  → `std.cc` assumes a fully featured target.

  → `arm-none-eabi` toolchain has no threading support.

  → No feature guards (`__cpp_lib_atomic_wait`, `_GLIBCXX_HAS_GTHREADS`, ...) or stubs.

# Solution (for the next two months): own `std` module

```
#ifdef __cpp_lib_atomic_wait
  using std::atomic_flag_notify_all;
  using std::atomic_flag_wait_explicit;
#endif
...
#if _GLIBCXX_HAS_GTHREADS
// 33.7 <condition_variable>
export namespace std
{
  ...
}
#endif
```

Already fixed in GCC 16!

# Results

| Experiment | ROM | Link | Compile |
|---|---|---|---|
| baseline | 3396 B | 0.21 s | 0.04 s |
| `std::cout` with `#include <iostream>` | 178580 B | 0.29 s | 0.70 s |
| `std::cout` with `import std` | 178588 B | 0.28 s | 0.36 s |
| `std::println` with `#include <print>` | 332648 B | 0.97 s | 0.95 s |
| `std::println` with `import std` | 332648 B | 0.98 s | 0.70 s |

# Thank you!

Paulo Martinez, Siemens

paulo.martinez@siemens.com

pauloacmartinez/cpp_binary_size

paulo-martinez