vollmann engineering gmbh
>

C++OS on 2nd RP2040 Core

C++OS goes Multi-Core

emBO++

March 2023

Detlef Vollmann
vollmann engineering gmbh

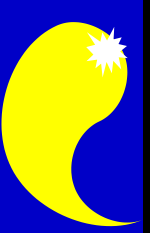
vollmann engineering gmbh
>

C++OS on 2nd RP2040 Core

C++OS goes Multi-Core

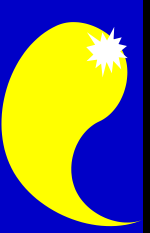
Detlef Vollmann
vollmann engineering gmbh
Luzern, Switzerland

dv@vollmann.ch
http://www.vollmann.ch/



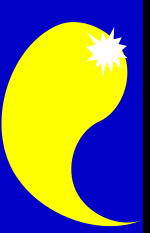
Overview

- What and Why?
- C++
- C++OS
- RP2040
- Executors
- Pico
- Dual Core
- Coroutines
- Sender/Receiver
- Why C++OS?
- References



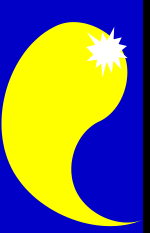
What?

- C++ Standard Interface
  - not implementation
  - C++ compiler is not an OS
- C++OS
  - a specific implementation
  - not complete



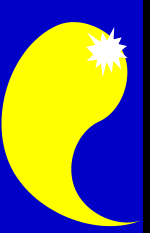
C++OS on RP2040

- Dual core
  - no real SMP
  - special synchronization
- C++OS support
  - see what's required
  - what are the challenges



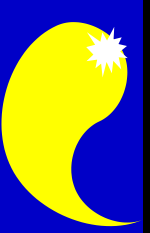
Why?

- Portability
- Testing / Simulation
  - on host
- Maintenance
  - well known API
- Control Structures
  - well known mechanisms



Why RP2040?

- Easily available
- ARM Cortex-M0+ based
- Documentation not too bad
- SDK as documentation
- It's pretty cool hardware



Overview

What and Why?

C++

`execution`

C++OS

RP2040

Executors

Pico

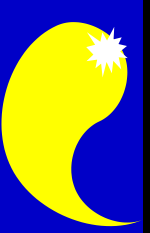
Dual Core

Coroutines

Sender/Receiver

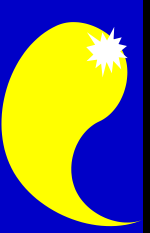
Why C++OS?

References



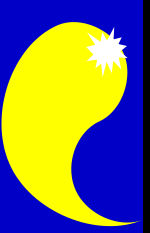
Concurrency

- C++11 provided `thread`, `async`
- Synchronization
  - `mutex`, `condition_variable`
  - `latch`, `barrier`, `semaphore`, `atomic_wait`
- Memory model
- Atomics



Time

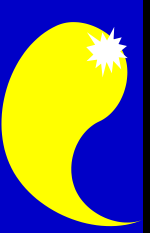
- Current time
  - `chrono`
- Durations
  - Sleep
- No Timers
  - missing event mechanism



More C++ Support

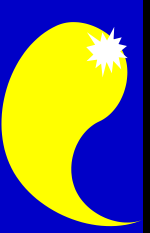
- Minimal event support
  - interrupts are signals
- Event model part of Networking TS
  - based on callbacks
  - problematic lifetime issues
- Coroutines
  - still no real library support

- Filesystem



Standard C++ Library

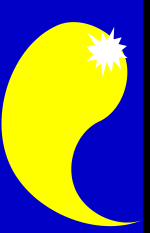
- C++ provides much for small systems beyond synchronization and tasks
  - not only freestanding!



Standard Library

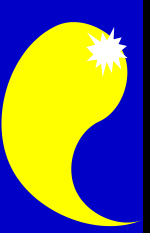
```
#include <string>
#include <vector>
#include <memory_resource>
#include <charconv>

constexpr size_t pmrSize = 1024;
std::byte pmrBuf[pmrSize];
StaticAllocator alloc(pmrBuf, pmrSize);
std::pmr::vector<int> vi({10, -24, 3456}, &alloc);
std::pmr::vector<std::pmr::string> vs(&alloc);
void fillVect()
{
    vi.push_back(-713);
    vs.push_back(std::pmr::string("Hello", &alloc));
    vs.push_back(std::pmr::string(" AVR", &alloc));
}
```

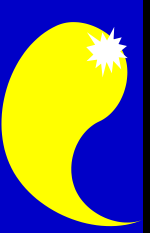
execution

- C++ Standard Proposal (P2300)
  - heterogenious computing resources
    - `std::thread` is not sufficient
  - event handling model
  - aimed for C++26



Execution Contexts

- `std::thread` is insufficient
- Execution contexts provide agents to run tasks
  - `std::async`, `std::thread`
  - thread pool
  - ISR
  - other core (GPU)
- Execution contexts not part of API
- Execution contexts provide schedulers
- Schedulers have a function `schedule()`

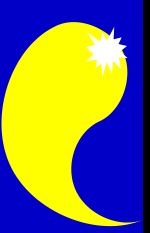


Event Interface

- Many (embedded) systems are event triggered
- Sender/Receiver provides the standard interface for this
  - not:

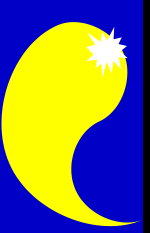
    async_event(ev, callback);
  - but:

    ev | callback;
- Some details still open
- Networking proposal expected
- Planned for C++26



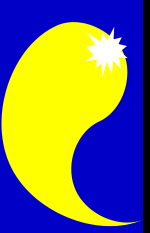
Queues

- Important communication mechanism
- Between different tasks
- Between ISRs and tasks
  - needs to be lock-free on ISR side



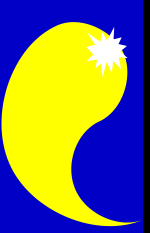
Sender/Receiver Queues

- The role of queues in the sender/receiver model is still largely open
- But the most important point of sender/receiver is customization
  - but the customization mechanism is still open
- Any queues between two stages can be customized to the involved execution agents



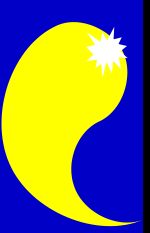
Overview

- What and Why?
- C++
- C++OS
- RP2040
- Executors
- Pico
- Dual Core
- Coroutines
- Sender/Receiver
- Why C++OS?
- References



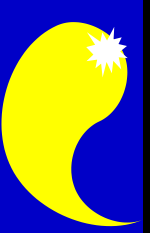
C++OS Goals

- Implement full C++ on small systems
- Some extensions specific for small systems
- Native ARM
  - Cortex-M0, Cortex-M3/4
- Native 8-bit AVR
  - AVR6
- Hosted implementation on FreeRTOS
  - ESP32



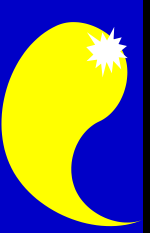
Standard Library

- Lot of C++ Standard Library is OS independent
  - "freestanding"
- Even non-freestanding
  - `<chrono>`
    - just `now()` is missing
  - `<mutex>`
    - `lock_guard`, `unique_lock`
    - just `mutex` itself is missing
  - etc...
- GCC's libstdc++ can be built e.g. for ARM, AVR, MSP430
  - C++OS simply fills the gaps



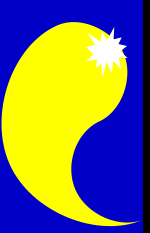
Overview

- What and Why?
- C++
- C++OS
- RP2040
- Executors
- Pico
- Dual Core
- Coroutines
- Sender/Receiver
- Why C++OS?
- References



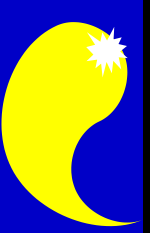
RP2040

- Based on ARM Cortex-M0+
  - not designed for multi-core (ARMv6, not ARMv7)
- Special Synchronization HW
  - CPUID
  - spinlocks
  - FIFO
- SMP / no SMP
  - Shared Memory Processing
  - not Symmetrical Multi-Processing
  - no "cache coherence"



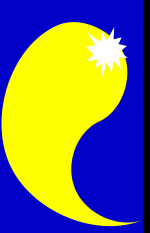
C++OS on RP2040

- Single core
- Standard Cortex-M0+ support
- No Pico SDK
  - does some resource management and synchronization
  - job of OS
  - somewhat fragmented
- Support for `std::mutex`, `std::condition_variable`
- Support for `std::atomic`
  - not complete



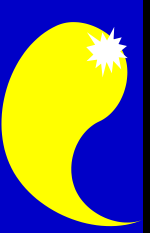
Examples

- Simple Test example
  - setup some tasks and start them
  - nice test for correct porting



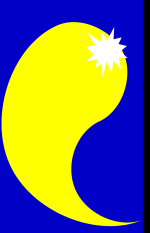
Overview

- What and Why?
- C++
- C++OS
- RP2040
- Executors
- Pico
- Dual Core
- Coroutines
- Sender/Receiver
- Why C++OS?
- References



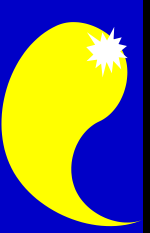
Executors in P2300

- Nearly non-existent
- No `execution_context` interface
  - not even how to get a scheduler
- No concrete execution contexts
  - not even a thread pool



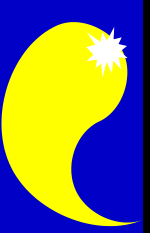
C++OS Executors

- `exec_context`
  - classic pre-emptive multi-tasking
- `single_context`
  - if you don't want preemption
  - based on hardware (not based on just a thread)
  - also LoopExecutor
- `IrqContext`
  - for interrupt service routines
  - not for RP2040 yet



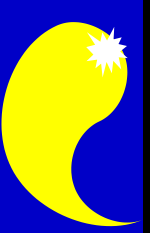
Synchronization

- (Lock-free) atomics work everywhere
  - no implementation for `atomic_notify`, `atomic_wait` yet
  - most others should work
- `std::mutex` and `std::condition_variable` only specified for standard preemption
- Basic `event` proposal P0073R2
  - notify from one execution context to another
  - possible m:n problem
  - possibly multiple others
  - somewhat superseeded by `atomic_notify`, `atomic_wait`
    - but mainly standard thread only



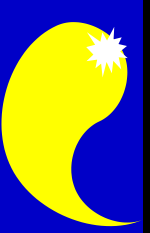
Events

- C++OS provides `event` as implementation mechanism behind e.g. `std::mutex`
- Originally for `exec_context` only
  - `block()`, `unblock_one()` and `unblock_all()`
- New support for `notify_one` templated on the sending executor
  - interface not settled
- Requires special support from executor



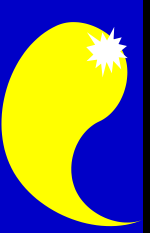
Queues

- Classic queue
  - blocking and non-blocking push and pop
  - implementation actually taken from the host
  - can be used from ISR (`try_` only)
- Event based queue
  - queue body
  - separate push and pull ends templated on executor
  - blocking and non-blocking (`try_`) push and pop
  - implementation based on `event`
  - only single-ended (yet)



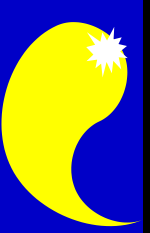
Examples

- Simple test example
  - three tasks in `exec_context`
  - first task creates numbers and pushes them into first queue
  - second task waits a bit and then transfers to second queue
  - third task consumes second queue
- Tests full and empty conditions



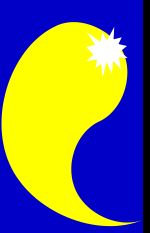
Overview

- What and Why?
- C++
- C++OS
- RP2040
- Executors
- Pico
- Dual Core
- Coroutines
- Sender/Receiver
- Why C++OS?
- References



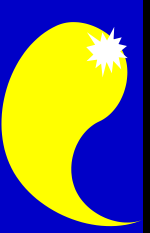
Pico W

- WiFi modem
  - connected via SPI
  - special SPI via PIO
- Idea: WiFi speaker
  - previous version based on Espressif and FreeRTOS
- I2S also via PIO



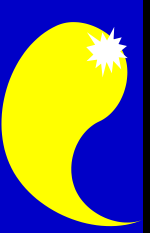
WiFi Speaker

- Using SnapCast
- Sound encoded via Opus
  - decoding on second core
- WiFi didn't work out
  - using UART connection instead
  - simple modification to `snapserver`



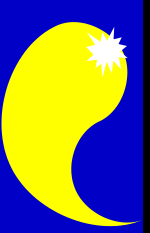
Code

- ISR for UART pushes Opus frames into first queue
- Decoder task decodes the frame and pushes PCM to second queue
- ISR for I2S pulls from second queue and puts them out



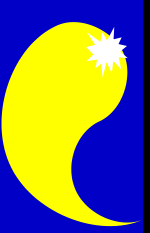
Overview

- What and Why?
- C++
- C++OS
- RP2040
- Executors
- Pico
- Dual Core
- Coroutines
- Sender/Receiver
- Why C++OS?
- References



Dual Core

- No traditional SMP
  - tasks can't migrate cores
- Currently only `single_context`
  - templated on core id
  - blocking should be `WFI`
  - startup for second core needs special code
- Atomics based on spinlock
- Event notification via FIFO

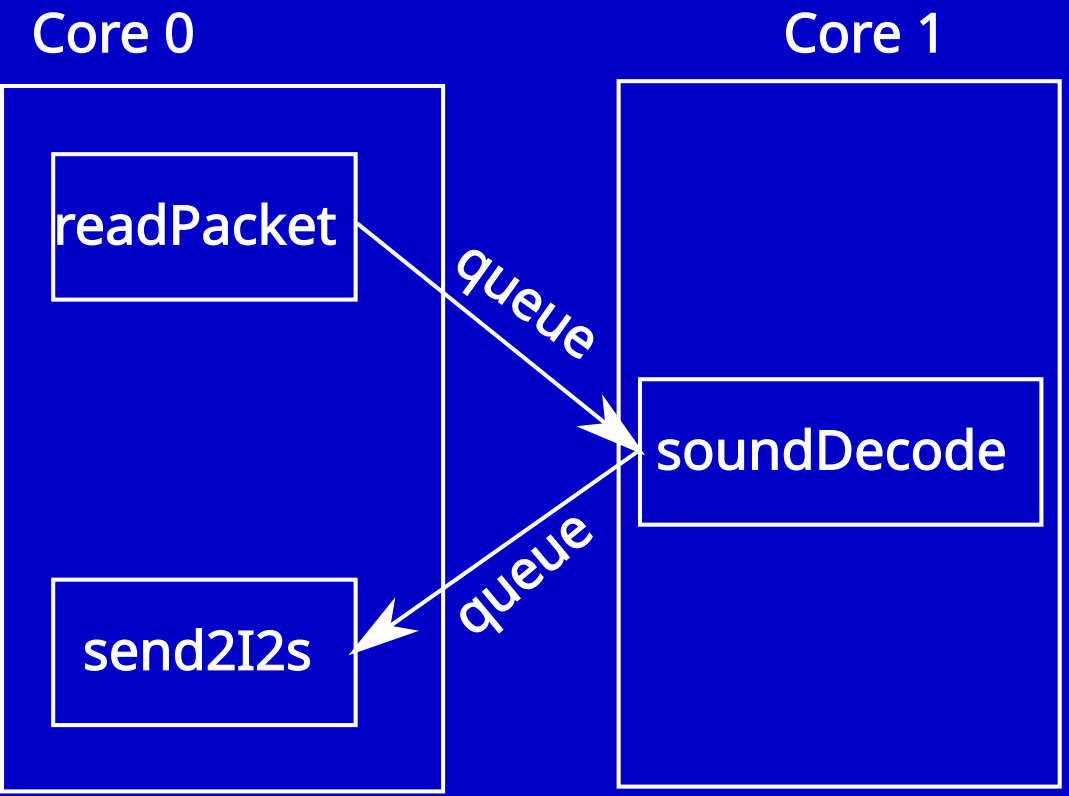


Examples

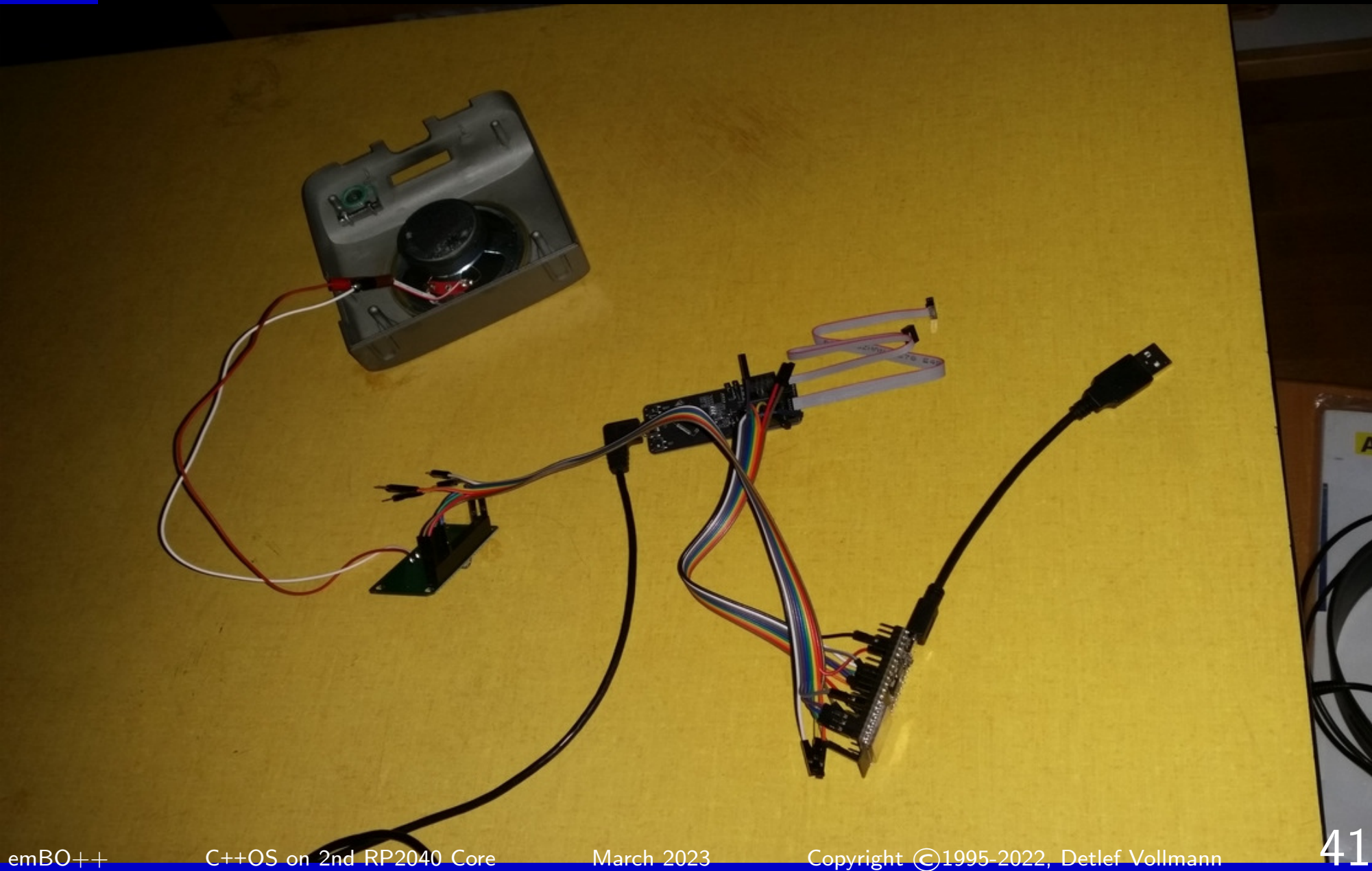
- Queue test as before
  - second task now on second core
- WiFi speaker also just puts decoder on second core

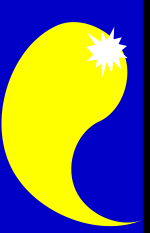
Examples

- Wireless speaker
  - receive sound stream from WiFi
  - decode it
  - send to I2S
  - dual core



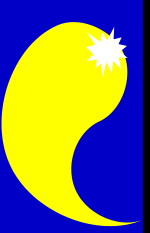
Hardware





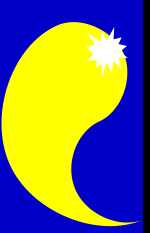
Overview

- What and Why?
- C++
- C++OS
- RP2040
- Executors
- Pico
- Dual Core
- Coroutines
- Sender/Receiver
- Why C++OS?
- References



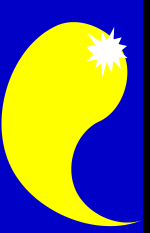
Coroutines

- Coroutines avoid much of the overhead
  - especially if no scheduling is required
  - typical for event based systems
- Still no standard `task` class



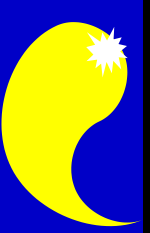
Queues for Coroutines

- Queues need coroutines for push and pop
- Coroutine promise must register for events
- Scheduler required based on select/epoll mechanism
- No implementation yet



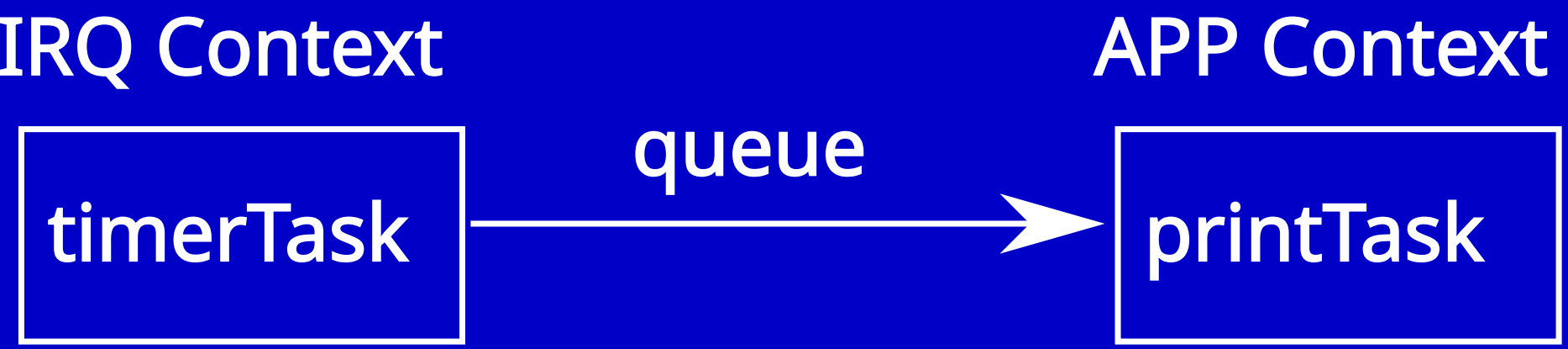
Overview

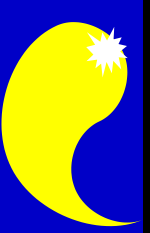
- What and Why?
- C++
- C++OS
- RP2040
- Executors
- Pico
- Dual Core
- Coroutines
- Sender/Receiver
- Why C++OS?
- References



Examples

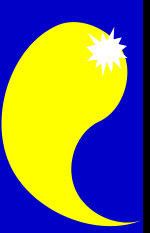
- Interrupts
  - transfer data from ISR to application





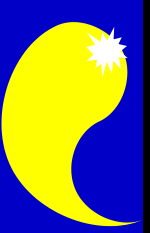
Overview

- What and Why?
- C++
- C++OS
- RP2040
- Executors
- Pico
- Dual Core
- Coroutines
- Sender/Receiver
- Why C++OS?
- References

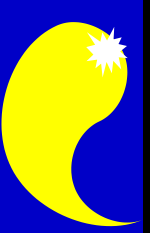


Don't Use C++OS

- C++OS is a proof of concept
- C++OS is a toy to try out new C++proposals
- Not for production use
  - buggy
  - no tests
  - incomplete
  - no interrupt policy

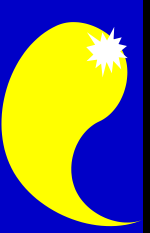


Overview



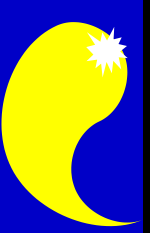
Code

- Any code shown here can be found at
  `https://gitlab.com/cppos1/embo-2023`



References

- These slides:
  `http://www.vollmann.ch/`
- Implementation:
  `https://gitlab.com/cppos1/cppos`
- Standard proposals, TSes:
  `http://www.open-std.org/JTC1/SC22/WG21/docs/papers/`
- Executors:
  `2022/p2300r4.html`
- Queues:
  `2019/p0260r3.html`



Questions

- ??????????????????????????????????

