Qualcomm

From Pre-Silicon to Production: Firmware Development on Zephyr

Dev Bhaveshbhai Joshi

Qualcomm Technologies Inc.

Open-Source Summit North America 2026

- Terminology
- Why Pre-silicon Testing Matters
- Problem
- Solution: Zephyr
- HIL Twister testing
- Results & Impact
- Q&A

Key Terms

- **PMIC**: Power Management Integrated-Circuit
- **Pre-silicon**: the development phase before final chip silicon is ready
- **EVK**: Evaluation Kit, reference hardware that exposes a chip's interfaces
- **USB PD**: USB Power Delivery, negotiates voltage and current over USB-C
- **HIL**: Hardware-in-the-Loop, test method that uses real hardware in the loop
- **Twister**: Zephyr's built-in test runner for hardware and simulation targets
- **Pytest**: Python's standard test framework for writing and running automated tests.
- **Production firmware**: Battery Management and USB Type-C PD application

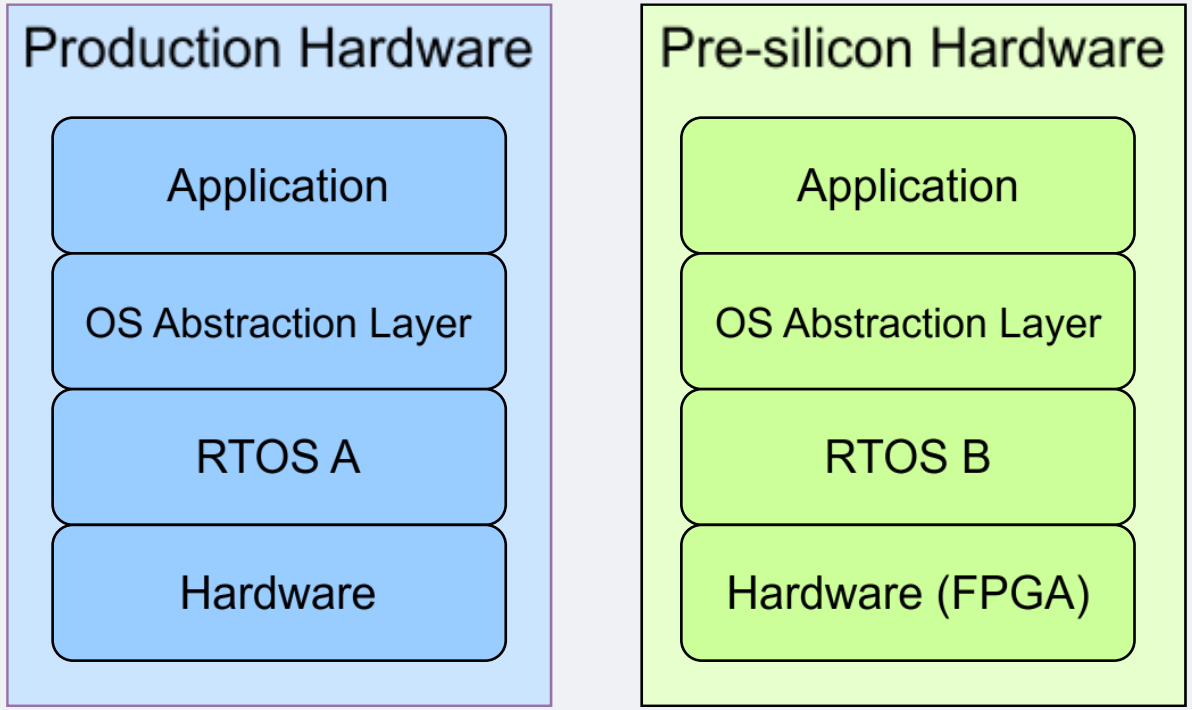
Why Pre-Silicon Testing Matters

- Production silicon often arrives late, delaying hands-on bring-up
- Running real firmware ahead of silicon delivery validates the design early
- Catching bugs at this stage is far cheaper than fixing them during bring-up
- Bugs discovered during bring-up stall dependent teams and push project timelines
- Tests become more meaningful the closer pre-silicon code mirrors production
- Affordable, off-the-shelf hardware like the Raspberry Pi Pico makes pre-silicon testing accessible without custom tooling.

The Problem

One App, Two Separate Repositories

- Production firmware ran on a proprietary RTOS tied to the production hardware
- Pre-silicon platform ran different RTOS on a FPGA board
- Separate repo, separate build system, separate team workflows
- The same application logic existed in both codebases independently
- Neither RTOS supported the other platform's hardware out of the box



The Cost of Maintaining Two Stacks

- Every new feature meant writing and testing the same logic twice
- Bug fixes had to be ported manually from one repository to the other
- The two codebases drifted apart as work continued on each side
- Synchronizing them became a recurring overhead
- Pre-silicon testing was only as valuable as the code parity between the two stacks

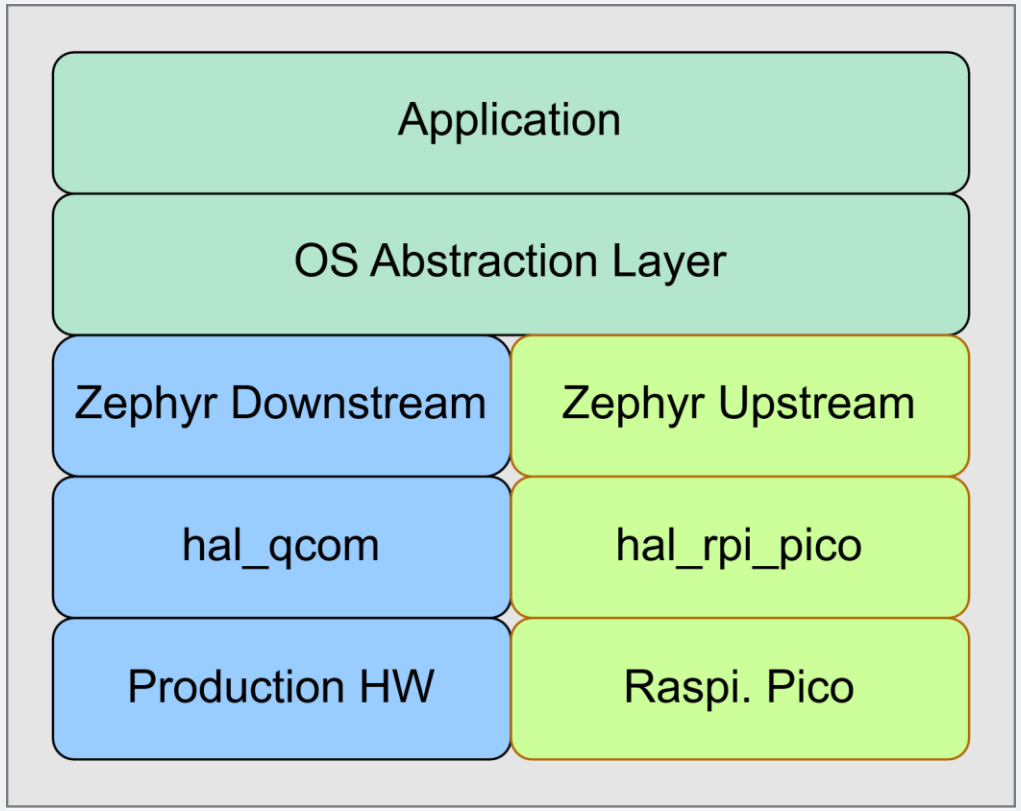
The Solution: Zephyr

Why Zephyr?

- Migration of production firmware to Zephyr meant we could merge both stacks into one
- Supports hundreds of MCUs and shields out of the box
- Strong community, good docs, and active upstream development
- Device Tree overlays and Kconfig fragments isolate board differences, keeping application code portable across targets
- west makes multi-repo projects manageable with a single manifest file
- Apache 2.0 license makes upstream contributions and open-sourcing straightforward

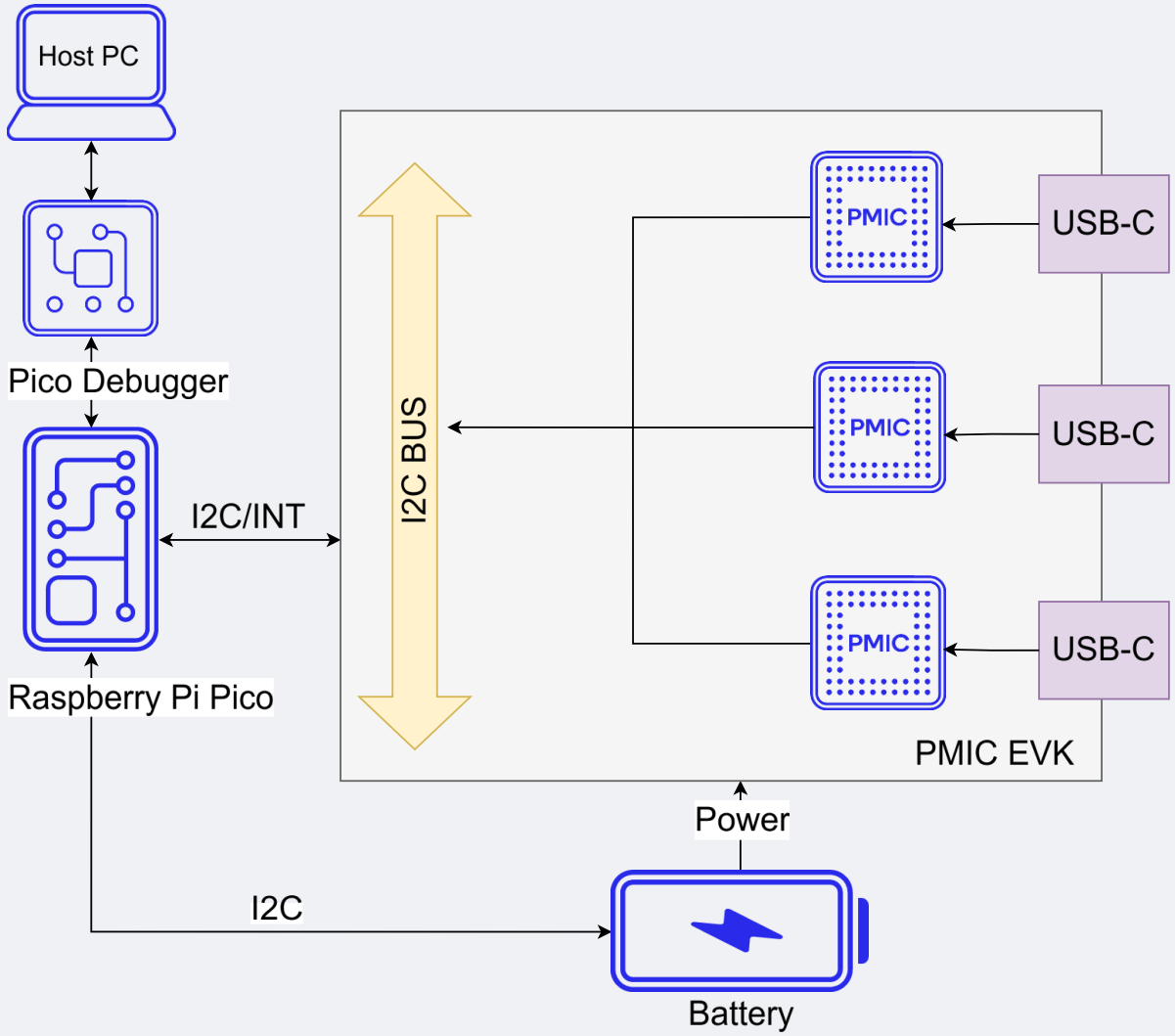
Software Architecture

- Application layer holds all business logic with no direct OS calls
- OS abstraction layer wraps platform and OS specific APIs behind a common interface
- Device Tree, Kconfig and OSAL keep board differences out of the application code
- Pre-silicon platform can also be used to test latest Zephyr kernel updates
- Removing codebase delta gave us high confidence going into silicon bring-up



Pre-Silicon Hardware Setup

- Raspberry Pi Pico runs the Zephyr application
  - Cheap, widely available, and upstream-supported in Zephyr
- Pico connects to the PMIC evaluation kit over I2C
  - I2C carries commands and status between the MCU and the PMIC
- The eval kit exposes the PMIC's charging circuitry and PD PHY to software
- Any Zephyr-supported board could replace the Pico with only a board config change

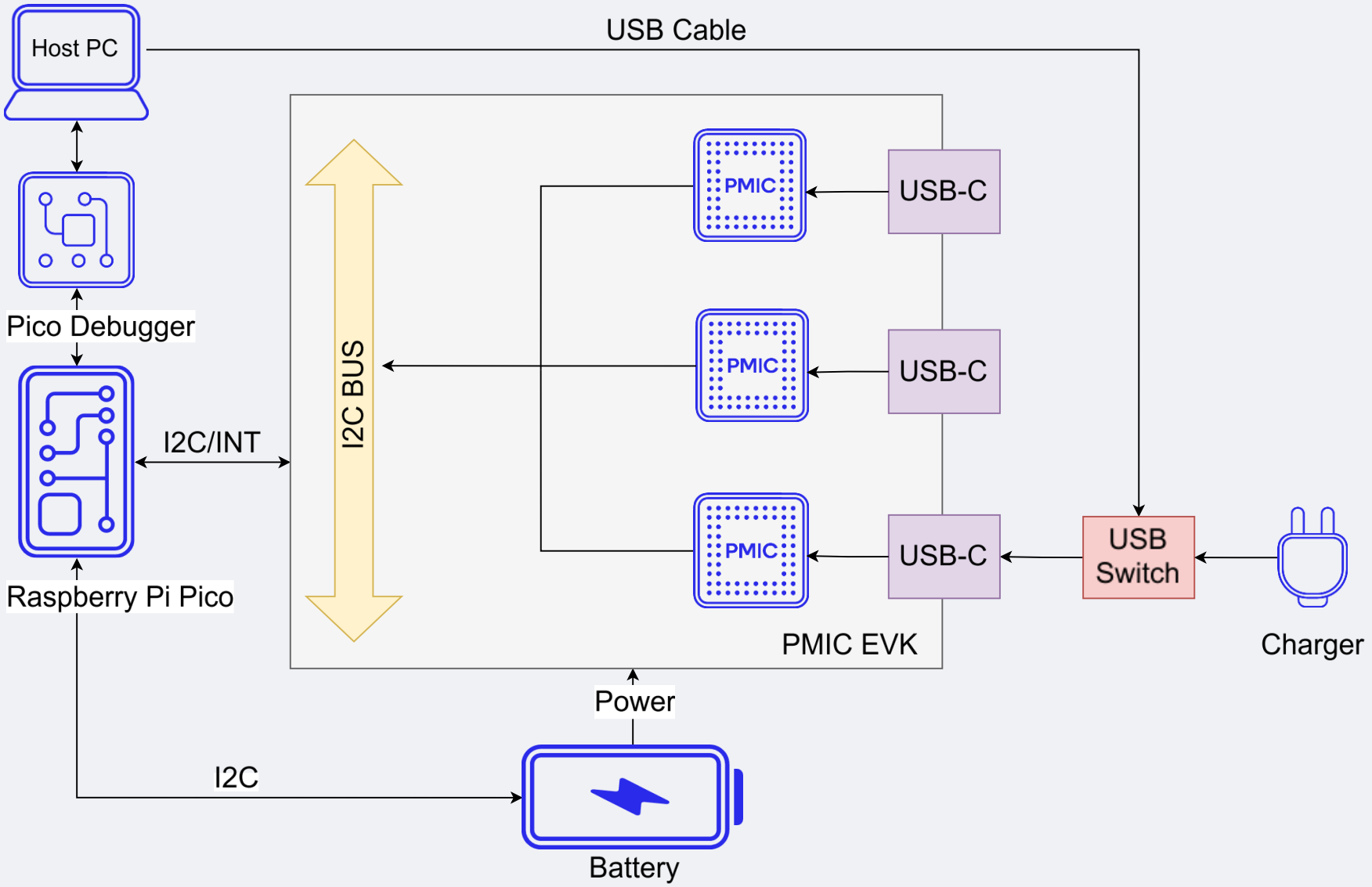


Validation with Twister

Twister: Zephyr's Built-In Test Runner

- Our pre-silicon validation was entirely manual, with no CI integration.
- Twister bridged the gap; enabled automated pre-silicon validation
- Test plan is declared in a Zephyr sample.yaml/testcase.yaml
- Twister compiles and flash firmware to the board
- We use Twister’s pytest harness:
  - Twister hands control to a Pytest test suite on the host. The firmware runs on the Pico, while Pytest orchestrates external equipment.
- Twister aggregates results and shows pass/fail from Pytest test suite

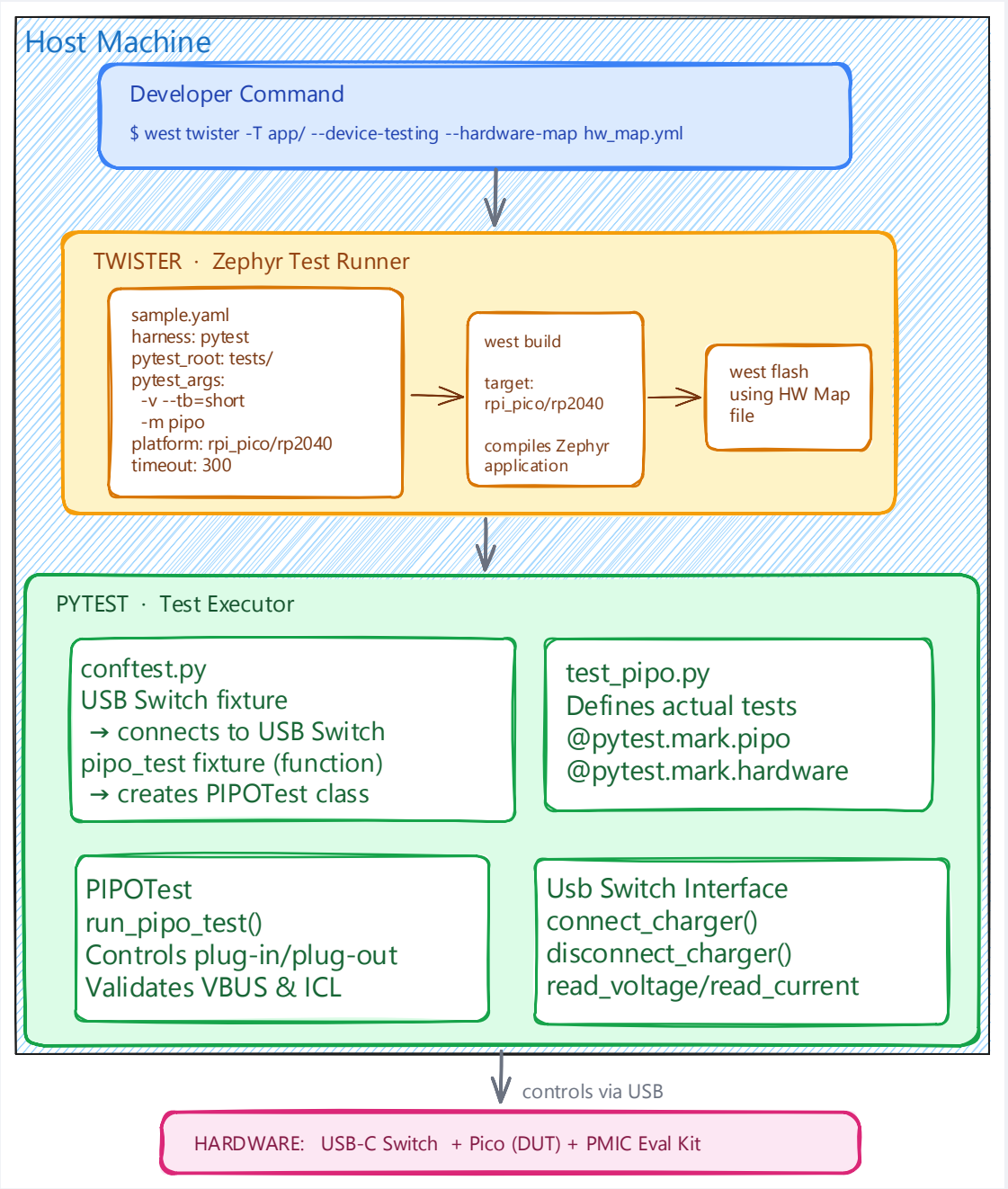
HIL Hardware Connections



HIL Test Architecture

Host controls hardware; firmware runs on its own

- Host PC runs Twister to build, flash, and then invoke pytest
- pytest controls USB-C switch hardware connected over USB from the host
  - Switch connects and disconnects a USB PD capable charger to the PMIC eval kit
  - Voltage and current measured at the switch's common port
- Firmware on the Pico negotiates USB PD completely on its own throughout



PIPO: Plug-In/Plug-Out Charging Test

- Charger connect and disconnect test
- Each cycle:
  - Connect the charger via the USB-C switch
  - Wait for USB PD to negotiate
  - Read VBUS, assert >= 0 mV
  - Read current, assert >= 0 mA
  - Disconnect and wait for cleanup
- Session fixture sets up the hardware once per run
- All cycles must pass for the test to pass

conftest.py

```
@pytest.fixture(scope='session')
def usb_switch():
  hw = UsbCSwitchInterface()
  yield hw
  hw.disconnect()
@pytest.fixture(scope='function')
def pipo_test(hw):
  test = PIPOTest(hw)
  yield test
  hw.disconnect()
```

test_pipo.py

```
def test_pipo_5_iterations(pipo_test):
  result = pipo_test.run_pipo_test(
      iterations=5)
  assert result['status'] == 'PASS'
  assert all(v >= 7500
             for v in result['vbus'])
  assert all(i >= 0
             for i in result['icl'])
```

Every Commit Gets Tested on Real Hardware

- Every pull request triggers a Twister run via GitHub Actions
- A self-hosted runner is physically connected to the test hardware
- PIPO is a gating test: the PR cannot merge unless all PIPO cycles pass
- Developers get hardware test results without touching the bench themselves
- Regressions show up in PR checks before code reaches the main branch

.github/workflows/hil.yml

```yaml
name: HIL Tests

on:
  push:
    branches: [main]
  pull_request:
    branches: [main]

jobs:
  hil-test:
    runs-on: [self-hosted, hil-rig]

    steps:
      - uses: actions/checkout@v4

      - name: Install dependencies
        run: pip install west && west update

      - name: Run Twister HIL tests
        run: |
          west twister -T . -W \
            --device-testing \
            --hardware-map  rpi_pico_hw_map.yml

      - name: Upload results
        uses: actions/upload-artifact@v4
        with:
          path: twister-out/
```

Results and Impact

What We Got Out of This

Single repository for the core application firmware	No more manual sync work between two separate stacks
Testing starts before silicon arrives	Bugs are found and fixed and new features are developed before production silicon
Faster bring-up after silicon delivery	Most software bugs are already gone by the time silicon shows up
Platform to perform easier demos	Can showcase PMIC features without a full production reference design

It Worked When It Mattered

1

THE CHALLENGE

Switching hardware and software at the same time

Our most recent project changed both layers in one go; we expected it to be the toughest bring-up the team had ever attempted.

2

WHAT WE DID

Months of pre-silicon runtime ahead of delivery

The same code ran on the pre-silicon platform for months, so by the time silicon arrived, most software bugs were already found and fixed.

3

THE RESULT

Production-ready, on schedule

Despite the full scope of change, we hit production readiness on time, the clearest proof this approach was worth the investment.

What to Take Away

- A broadly-supported RTOS like Zephyr aligns different production and pre-silicon platforms
- Zephyr's Device Tree and Kconfig are the real enablers, not just the RTOS itself
  - Hardware differences belong in config files, not in application code
- Pre-silicon testing is only useful if it actually runs the same code production will run
- Investing in test infrastructure early pays back quickly when silicon comes in

What Comes Next

- Qualcomm is already adopting Zephyr across a wide range of platforms, from MCUs to apps processors and beyond
- This PMIC work is one example of a larger company-wide shift toward Zephyr as a unified RTOS strategy
- Upstream USB/Type-C, FG, Charger drivers and more features to the Zephyr RTOS
- New shield board for Raspberry Pi Pico targeting the next generation of PMIC EVK boards
  - Designed as a developer kit for developers to evaluate PMIC hardware and software
- Working on an open-sourcing Zephyr application, modules, drivers, sub-system, etc.

Questions/Answers?

Thank you

Follow us on:

For more information, visit us at qualcomm.com & qualcomm.com/blog