# Claude Hardware Testing Infrastructure

## Automated Board Identification, UART Discovery, and Test Execution

Andrișan Andreea-Daniela

Analog Devices, Inc.

February 1st, 2026

# Outline

# Hardware Testing Infrastructure

## Purpose

Automated end-to-end hardware validation for ADI SC5XX development boards

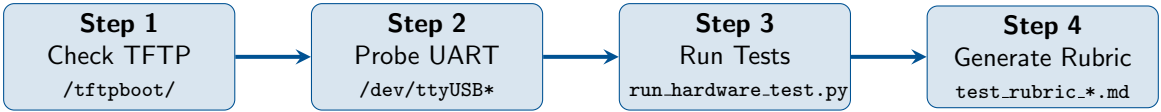
**Testing Pipeline Steps:**

1. **Identify Flash Project** – Check TFTP directory for board configuration
2. **Detect UART Connection** – Probe serial ports to find connected boards
3. **Run Hardware Tests** – Execute configuration-driven test suite
4. **Generate Test Rubric** – Produce detailed markdown results

## Key Benefit

Fully automated board discovery and validation with no manual configuration

# Testing Pipeline Architecture




**Inputs**

- TFTP boot directory
- Available serial ports
- Board configuration JSON

**Outputs**

- Board identification
- UART port mapping
- Test results with pass/fail status
- Markdown test rubric

# Step 1: Identify Flash Project (TFTP Check)

**Check `/tftpboot/` to determine which board is configured**

```
$ ls -la /tftpboot/
-rwxr-xr-x  68940 stage1-boot.ldr
-rwxr-xr-x 557044 stage2-boot.ldr
-rwxr-xr-x 1304076 u-boot-spl-sc594-som-ezkit.elf
-rwxr-xr-x 4914388 u-boot-proper-sc594-som-ezkit.elf
-rw-r--r-- 5497752 zImage
-rw-r--r--   24666 sc594-som-ezkit.dtb
-rw-r--r-- 7865090 fitImage
-rw-r--r-- 236191744 adsp-sc5xx-full-adsp-sc594-som-ezkit.rootfs.jffs2
```

## Identified Configuration: SC594-SOM-EZKIT

- **U-Boot**: SPL + proper ELF binaries
- **Kernel**: Linux 6.12 zImage + DTB
- **Root FS**: Full JFFS2 image

# TFTP Directory Key Files

| File | Purpose | Board Indicator |
|---|---|---|
| `u-boot-spl-sc*.elf` | U-Boot SPL binary | Board name in filename |
| `u-boot-proper-sc*.elf` | U-Boot proper binary | Board name in filename |
| `sc*-som-ezkit.dtb` | Device tree blob | Board name in filename |
| `zImage` / `fitImage` | Linux kernel image | Linked to board-specific version |
| `adsp-sc5xx-*.jffs2` | Root filesystem | Board name in filename |
| `stage1-boot.ldr` | Stage 1 bootloader | Generic for all boards |
| `stage2-boot.ldr` | Stage 2 bootloader | Generic for all boards |

## Board Identification Method

Parse filenames containing `sc594`, `sc598`, `sc589`, etc. to determine the target board configuration

# Step 2: Detect UART Connection

**Probe serial ports to find and identify connected boards**

```
# List available UART devices
$ ls -la /dev/ttyUSB* /dev/ttyACM*
crw-rw----+ 1 root plugdev 188, 0 /dev/ttyUSB0
crw-rw----+ 1 root plugdev 188, 1 /dev/ttyUSB1
```

```
# Direct serial probe
import serial
ser = serial.Serial('/dev/ttyUSB0', 115200, timeout=2)
ser.write(b'hostname\n')
response = ser.read(1000).decode()
# Response: 'adsp-sc594-som-ezkit'
```

**Detection Result**

**SC594-SOM-EZKIT** detected on /dev/ttyUSB0 – matches TFTP configuration

# UART Detection Results

| Port | Status | Hostname | Board Type |
|---|---|---|---|
| /dev/ttyUSB0 | **Connected** | adsp-sc594-som-ezkit | SC594-SOM-EZKIT |
| /dev/ttyUSB1 | No Response | – | – |

**Board Identification Patterns:**

- *sc594* → SC594-SOM-EZKIT
- *sc598* → SC598-SOM-EZKIT
- *sc589*mini* → SC589-MINI
- *sc589*ezkit* → SC589-EZKIT

**Verification**

TFTP configuration (SC594-SOM-EZKIT) matches detected board on UART

# Step 3: Run Hardware Tests

**Execute configuration-driven test suite via UART**

```
$ python3 run_hardware_tests.py \
    --config configs/sc594-som-ezkit.json \
    --port /dev/ttyUSB0

Attempting to connect to SC594-SOM-EZKIT on /dev/ttyUSB0...
Connecting to /dev/ttyUSB0 at 115200 baud...
Already authenticated!

============================================================
STARTING HARDWARE TESTS FOR SC594-SOM-EZKIT
============================================================
```

## Test Configuration

- Serial: `/dev/ttyUSB0` @ 115200
- I2C Channels: 3
- Crypto: Disabled
- SRAM: `/dev/sram_mmap`
- Audio: sc5xxasoccard / ADAU1962
- Network: iperf3 server configured

# Test Categories Executed

## Core Hardware Tests

- **Board Detection**
  - Hostname verification
  - Kernel version check
- **I2C Communication**
  - Channel count validation
  - Bus scan per channel
- **Network Interface**
  - Ethernet/MAC detection
  - IP address validation
  - Ping connectivity
  - iperf3 performance

## Peripheral Tests

- **SRAM Allocation**
  - Device presence
  - Memory mapping
- **GP Timer Counters**
  - Counter detection
  - Increment validation
- **ALSA Audio**
  - Sound card detection
  - Codec identification
- **RPMsg / Clock**
  - Inter-core communication
  - Clock tree access

# Test Results: SC594-SOM-EZKIT

## Score: 80%

16 Passed / 4 Failed (20 Total Tests)

| Category | Tests | Passed | Failed |
|---|---|---|---|
| Board Detection | Hostname, Kernel Info | 2 | 0 |
| I2C Communication | Channel Count, Ch0/Ch1/Ch2 Scans | 3 | 1 |
| Network Interface | Ethernet, IP, Ping, iperf3 | 3 | 1 |
| SRAM Allocation | Device Presence, Memory Mapping | 1 | 1 |
| GP Timer Counters | Detection, Function | 2 | 0 |
| ALSA Audio | Card, Codec Detection | 2 | 0 |
| RPMsg Communication | Device, Bind, Echo | 2 | 1 |
| Clock Configuration | Debug Access | 1 | 0 |
| **Total** | | **16** | **4** |

# Detailed Test Output

```
=== Board Detection Tests ===
  [PASS] Hostname: adsp-sc594-som-ezkit
  [PASS] Kernel Info: Linux adsp-sc594-som-ezkit 6.12.0-yocto-standard-00085-g27fd...

=== I2C Communication Tests ===
  [FAIL] Channel Count: 1 channels (expected 3)
  [PASS] Channel 0 Scan: Scan completed, devices: No
  [PASS] Channel 1 Scan: Scan completed, devices: No
  [PASS] Channel 2 Scan: Scan completed, devices: Yes

=== Network Interface Tests ===
  [PASS] Ethernet Detection: MAC: xx:xx:xx:xx:xx:xx
  [PASS] IP Address: IP: xx.xx.xx.xx
  [PASS] Ping Test: Ping successful
  [FAIL] iperf3 Performance: iperf3 failed or timeout

=== GP Timer Counter Tests ===
  [PASS] Counter Detection: 8 counters found
  [PASS] Counter Function: Diff: 111720833 (175922355 -> 287643188)

=== ALSA Audio Tests ===
  [PASS] Card Detection: Card: sc5xxasoccard
  [PASS] Codec Detection: Codec: ADAU1962
```

# Failed Tests Analysis

| Test | Expected | Actual |
|---|---|---|
| I2C Channel Count | 3 channels | 1 channel |
| iperf3 Performance | Throughput data | Timeout/connection failed |
| SRAM Memory Mapping | mmap success | mmap failed |
| RPMsg Device Presence | Devices found | 0 devices found |

**Possible Causes:**

- **I2C**: Device tree configuration or driver loading issue
- **iperf3**: Network server not running or firewall blocking
- **SRAM**: Kernel module not loaded or permission issue
- **RPMsg**: SHARC cores not loaded or remoteproc not started

# Step 4: Generated Test Rubric

**Output: Markdown test rubric for documentation**

```
# Hardware Test Rubric: SC594-SOM-EZKIT
**Date:** 2026-01-26 16:08:12
**Serial Port:** /dev/ttyUSB0
**Kernel:** Linux 6.12.0-yocto-standard

## Test Results Summary
| Category             | Passed | Failed | Total |
|----------------------|--------|--------|-------|
| Board Detection      | 2      | 0      | 2     |
| I2C Communication    | 3      | 1      | 4     |
| Network Interface    | 3      | 1      | 4     |
| ...                  | ...    | ...    | ...   |

## Overall Score: 80% (16/20)

## Issues Identified
1. I2C Channel Count: Expected 3, found 1
2. iperf3: Connection timeout
...
```

## Output Location

`docs/hardware-testing/test_rubric_SC594-SOM-EZKIT_2026-01-26_160812.md`

# Supported ADI SC5XX Boards

| Board | Processor | I2C Ch | Crypto | Config File |
|---|---|---|---|---|
| SC598-SOM-EZKIT | Cortex-A55 + SHARC+ | 3 (skip ch0) | Yes | `sc598-som-ezkit.json` |
| SC594-SOM-EZKIT | Cortex-A5 + SHARC+ | 3 | No | `sc594-som-ezkit.json` |
| SC589-MINI | Cortex-A5 + SHARC+ | 2 | No | `sc589-mini.json` |
| SC589-EZKIT | Cortex-A5 + SHARC+ | 3 | No | `sc589-ezkit.json` |
| SC584-EZKIT | Cortex-A5 + SHARC+ | 3 | No | `sc584-ezkit.json` |
| SC573-EZKIT | Cortex-A5 + SHARC+ | 3 | No | `sc573-ezkit.json` |

**Serial Communication Settings:**

- Baud rate: 115200
- Credentials: root / adi
- Flow control: None (GPIO-controlled on SOM boards)

# GitHub Actions Workflow Overview

## Workflow: `build-linux-images.yml`

Automated CI/CD pipeline for building, flashing, and testing ADI SC5XX boards

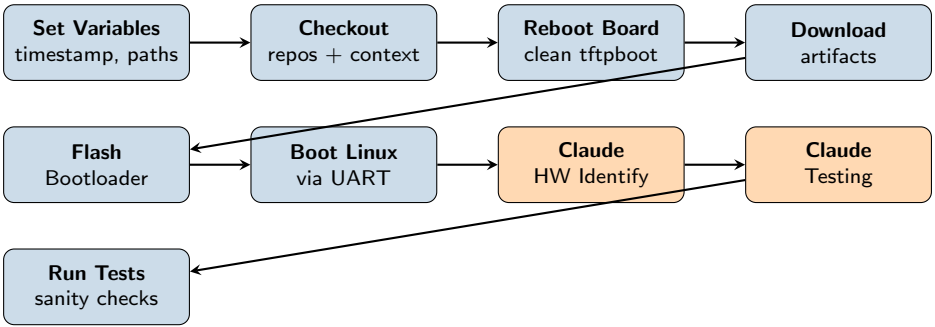
**Workflow Triggers:**

- `workflow_dispatch` – Manual trigger with version/manifest inputs
- `repository_dispatch` – External API triggers
- `pull_request` – PR events (opened, edited, synchronize)
- `push` – Push to `claude_testing` branch

## Key Feature: Claude AI Integration

The workflow includes stages that invoke Claude Code CLI for automated hardware identification and test execution

# Workflow Architecture




**Standard Stages** (blue)

- Variable setup and checkout
- Board reboot and artifact download
- Bootloader flash and Linux boot

**Claude AI Stages** (orange)

- Hardware identification
- Automated test execution
- Rubric generation

# Claude Stage 1: Hardware Identification

**Step: "Hardware identification and testing via Claude"**

```
- name: Hardware identification and testing via Claude
  run: |
    source /etc/sc5xx-utils/${{ matrix.platform }}.sh bootmode1
    source /variables/claude_vars.sh
    cd claude_context_1
    claude -p "Identify board type from /tftpboot/ filenames"
    claude -p "Find board on /dev/ttyUSB* at 115200 baud, verify hostname"
    claude -p "Run hardware tests with matching config, save rubric"
```

**Claude Prompts Executed:**

1. **TFTP Check** – Identify board from `/tftpboot/` contents
2. **UART Detection** – Probe serial ports to find connected board
3. **Test Execution** – Run hardware test infrastructure

## Context Repository

Uses `claude_context_1` (branch: `hardware_identification`)

# Claude Stage 2: Automated Testing

**Step: "Claude Testing"**

```
- name: Claude Testing
  run: |
    source /etc/sc5xx-utils/${{ matrix.platform }}.sh bootmode1
    source /variables/claude_vars.sh
    cd claude_context_2
    claude -p "scan the files from /etc/sc5xx-utils and run the specific
               hardware tests for the ${{ matrix.platform }} board connected
               to this machine and generate a rubric"
```

**Single Comprehensive Prompt:**

- Scans `/etc/sc5xx-utils/` for board configuration scripts
- Identifies the specific platform (`sc594-som-ezkit`, `sc598-som-ezkit`)
- Executes hardware tests via UART
- Generates markdown test rubric

**Context Repository**

Uses `claude_context_2` (branch: `hardware_run`)

# Claude Context Repositories

**claude_context_1**
Branch: `hardware_identification`

- TFTP directory analysis
- UART port discovery
- Board hostname detection

**Key Files:**

- `CLAUDE.md`
- `uart_discovery.py`

**claude_context_2**
Branch: `hardware_run`

- Board configuration parsing
- Test suite execution
- Rubric generation

**Key Files:**

- `CLAUDE.md`
- `run_hardware_tests.py`

**Separation of Concerns**

Two separate context repositories allow specialized Claude prompts for each phase

# Workflow Run Example: SC594-SOM-EZKIT

**From workflow run**

```
Job: Flash and Boot (sc594-som-ezkit, full)
Runner: LNX-RO-1 (analog-Precision-5520)
Branch: claude_testing
Status: Completed (with test failures)

Test Results:
  GPTIMER-COUNTERS: PASS (8 timers detected)
  SRAM: PASS
  I2C: FAIL (1 of 3 channels working)
  RPMsg: FAIL (Echo test 1 failed)
  Networking: PASS (ping OK, iperf3 ~94 Mbits/sec)
  ALSA Devices: PASS (adau1962-hifi-0 detected)
  Clock: PASS

Final Score: 3 failures / 7 test categories
```

## Workflow Outcome

Claude successfully identified the board, executed tests via UART, and the results were captured in the workflow logs

# Hardware Testing Infrastructure Summary

## 4-Step Testing Pipeline

1. **Identify Flash Project** – Parse `/tftpboot/` for board configuration
2. **Detect UART Connection** – Probe serial ports, match hostname patterns
3. **Run Hardware Tests** – Execute JSON-configured test suite via UART
4. **Generate Test Rubric** – Produce markdown documentation

### SC594-SOM-EZKIT Results

- Detected on `/dev/ttyUSB0`
- Kernel: Linux 6.12.0
- Score: **80%** (16/20 tests)
- Key issues: I2C, iperf3, SRAM, RPMsg

### Framework Capabilities

- 9 test categories
- 20+ individual tests
- 6 supported board types
- Automated rubric generation

# CI/CD Integration Summary

## GitHub Actions Workflow

Fully automated pipeline from build to test with Claude AI integration

**Claude-Powered Stages:**

1. **Hardware Identification** (3 prompts)
   - Check TFTP for flashed configuration
   - Detect board via UART connections
   - Execute hardware test infrastructure
2. **Automated Testing** (1 comprehensive prompt)
   - Parse `/etc/sc5xx-utils/` configurations
   - Run platform-specific hardware tests

## Key Benefits

No manual intervention required – intelligent test execution and automated documentation

# Quick Reference Commands

```
# Step 1: Check TFTP configuration
ls -la /tftpboot/

# Step 2: Detect UART connections
python3 -c "
import serial
ser = serial.Serial('/dev/ttyUSB0', 115200, timeout=2)
ser.write(b'hostname\n')
print(ser.read(1000).decode())
"

# Step 3: Run hardware tests
python3 run_hardware_tests.py \
    --config configs/sc594-som-ezkit.json \
    --port /dev/ttyUSB0

# Output: test_rubric_SC594-SOM-EZKIT_<timestamp>.md
```

# Thank you!