Zephyr® Project
Developer Summit

Device Power Management

Journey to 5uA

Jordan Yates, Embeint

Quick Disclaimers

- I am no longer a CSIRO employee
- I am presenting this work in a personal capacity with the permission of CSIRO
- All work in this presentation was funded by CSIRO

Terms and Abreviations

- A = Ampere
- mA = milliAmpere = 0.001 A
- uA = microAmpere = 0.000001 A
- ~ = Approximately
- PM = Power Management (Zephyr subsystem)
- Power consumption = Current (In mA or uA) consumed by hardware, measured at the power input.

Why do we care about power consumption?

- Power needs to come from somewhere
- For mains powered devices:
  - Cost to run (1A = $9 AUD over a year)
- For battery powered devices:
  - Smaller batteries
  - Less frequent battery swaps/recharging
  - Size and weight restrictions
  - Hitting an “energy in == energy out” budget for intermittent charging

Ballpark battery lifetimes

CR2450 Coin Cell Battery

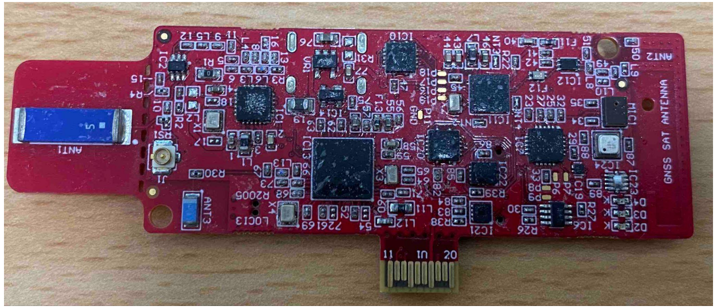
- 24mm diameter, 5mm height
- ~650 mAHr rated capacity

Average Consumption	Hours	Days	Years
5 mA	130	5	0
1 mA	650	27	0
100uA	6500	270	0.7
50 uA	13000	540	1.4
10 uA	65000	2600	7.1

Hardware Case Study

CSIRO Loci3

- General development platform
  - nRF52840 SoC
- Multiple Radios
  - Bluetooth
  - LoRa/LoRaWAN
  - GNSS
- Multiple sensors
  - 3-Axis Accelerometer
  - 9-Axis IMU
  - Environmental
    - Temperature
    - Pressure
    - Humidity
  - Light
- Multiple Storage Options
  - 128 Mbit onboard flash
  - External SD card
- Audio Codec
- Battery Management
  - Voltage Measurement
  - Solar Charging



Power Consumption Part 1

Part Number	Description	Boot (uA)	Minimum (uA)
nRF52840	Bluetooth SoC	2.35	1.5
SX1262	LoRa modem	800	1.2
ZOE-M8	GNSS modem	30000	15
BMA400	3-Axis Accelerometer	0.1	0.1
BMX160	9-Axis IMU	4	4
BME680	Environmental sensor	0.1	0.1
SI1133	Light sensor	0.1	0.1

Power Consumption Part 2

Part Number	Description	Boot (uA)	Minimum (uA)
Voltage Divider	Battery measurement	60	60
W25Q128JV	128Mbit SPI NOR Flash	10	1
Micro SD		>1000	200 - 1000
NAU8810	Audio Codec	10	10*
	Total	~32mA	~300uA

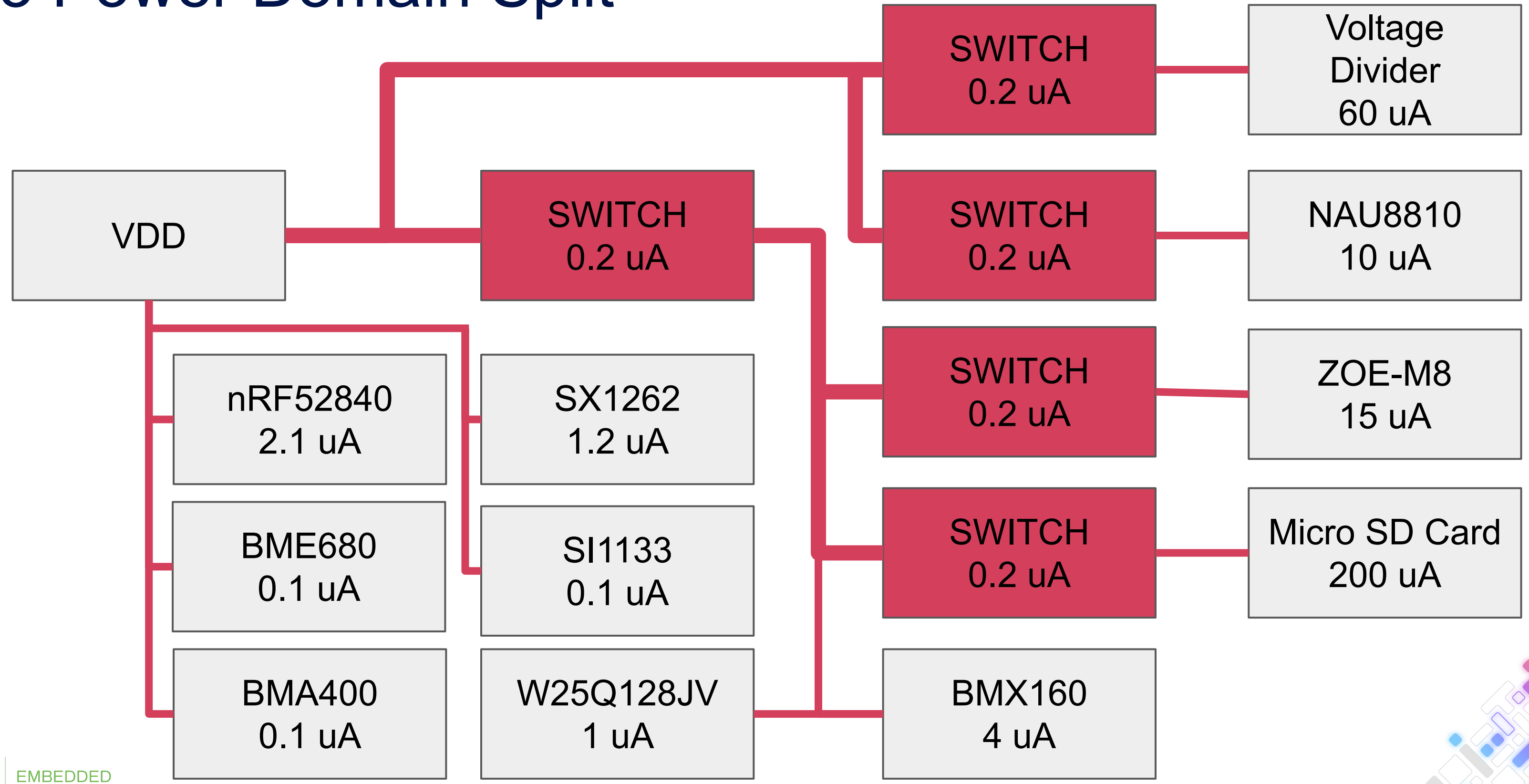
We're already in trouble!

* Minimal effort expended on validating this

Cutting the power

- There is no rule that says hardware always needs to be powered
- For peripherals that draw too much power, switch the power supply
  - Adds switch quiscent current (0.02uA)
  - Removes peripheral quiscent current
- Adds hardware & software complexity
- Need to be aware of “back-powering”
  - VDD present on a non-power pin (bus, control line, etc)

Loci 3 Power Domain Split

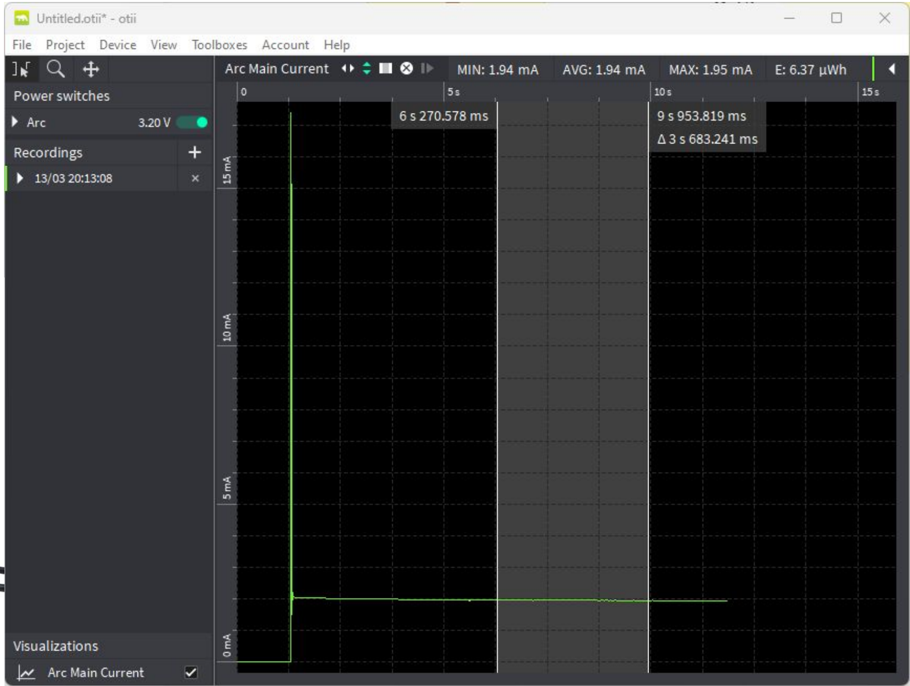


EMBEDDED
OPEN SOURCE
SUMMIT

Software Baseline

zephyr/samples/hello_world

- 2mA starting point
- Looks better than our assumed worst case, but:
  - Power domain drivers not enabled
  - Control lines have pull down resistors

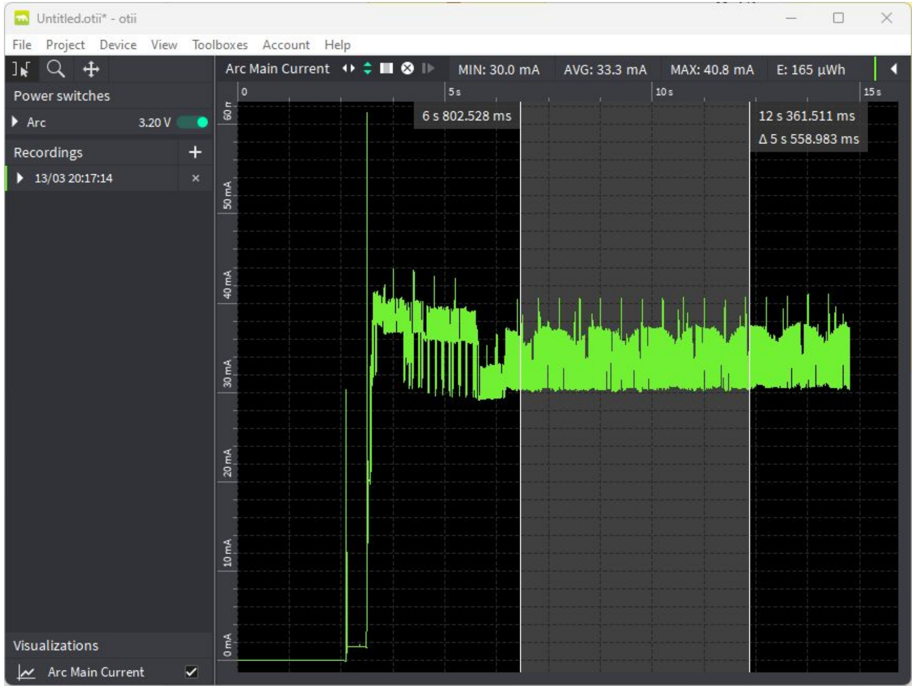


EMBEDDED OPEN SOURCE SUMMIT

Enabling the GPIO power domain driver

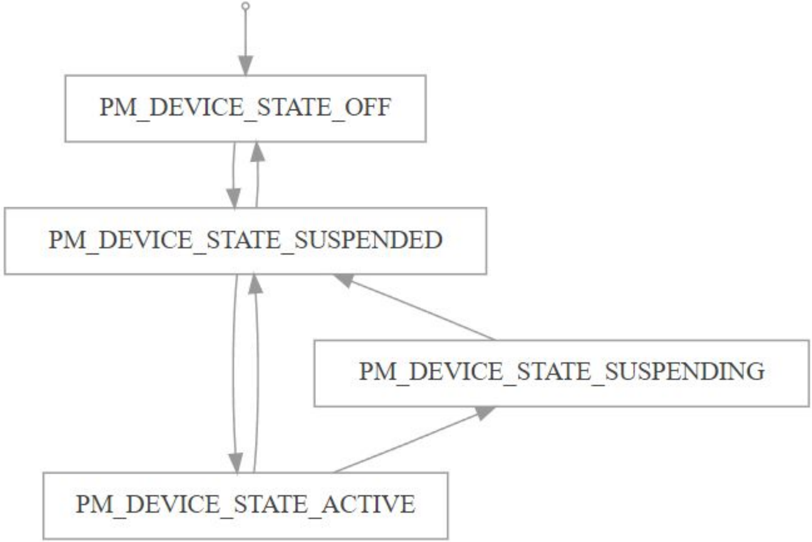
- zephyr/samples/hello_world
  - CONFIG_PM_DEVICE=y
  - CONFIG_POWER_DOMAIN=y
  - CONFIG_POWER_DOMAIN_GPIO=y

- This roughly matches our expected datasheets baseline of 32mA



EMBEDDED OPEN SOURCE SUMMIT

Device default state



Device states:

- OFF = No power applied to device
- SUSPENDED = Powered and “low power”
- ACTIVE = Powered and operational

Devices default to STATE_ACTIVE!

- Even when CONFIG_PM_DEVICE=y
- Even when CONFIG_PM_DEVICE_RUNTIME=y

Unless explicit action is taken, drivers will always be running, and therefore will never be “low power”. How not “low power” they are depends on the device.

This also applies to SoC hardware blocks (UART, SPI, etc).

Software Guidelines

BEWARE: Opinions inside

Generic low power operation guidelines

All drivers for all devices on a board should be enabled.

- Devices typically require some configuration to be low power
- Required configuration implies drivers

All instances of all devices should have “Device Runtime PM” enabled

- STATE_ACTIVE is the default, not the low power mode

All drivers in a low power application need to implement PM

- Smart hardware and frameworks are useless without driver support

The preceding points are true for ALL applications, even if only a subset of hardware is used.

All drivers should be enabled

Enabling Individual drivers is relatively simple with devicetree defaults:

```
config SHT4X
	bool "SHT4x Temperature and Humidity Sensor"
	default y
	depends on DT_HAS_SENSIRION_SHT4X_ENABLED
	select I2C
	select CRC
	help
	  Enable driver for SHT4x temperature and humidity sensors.
```

Beware higher level dependencies:

```
menuconfig SENSOR
	bool "Sensor drivers"
	help
	  Include sensor drivers in system config

...
if SENSOR

comment "Device Drivers"

source "drivers/sensor/a01nyub/Kconfig"
source "drivers/sensor/adltc2990/Kconfig"
source "drivers/sensor/adt7310/Kconfig"
source "drivers/sensor/adt7420/Kconfig"
```

EMBEDDED OPEN SOURCE SUMMIT

All instances should have PM Device Runtime

Can be manually enabled per instance in an app with: `pm_device_runtime_enable(dev);`

- Fidly, ties application to particular platform

Can be done at the board devicetree level with: `zephyr,pm-device-runtime-auto;`

- Automatically calls the above function after running the init function*
- Automatic low power application*, regardless of platform
- Using devicetree to configure software, may annoy purists

```
bma400: bma400@2 {
        compatible = "bosch,bma400";
        reg = <2>;
        spi-max-frequency = <17000000>;
        int1-gpios =  <&gpio0 9 (GPIO_ACTIVE_HIGH)>;
        zephyr,pm-device-runtime-auto;
};
```

* When CONFIG_PM_DEVICE_RUNTIME=y

All drivers need to implement PM

Implement pm_control callback:

```
static int bma400_pm_control(const struct device *dev, enum pm_device_action action) { … }
```

Update init function to be PM friendly:

1. Setup software constructs (k_sem_init, device_is_ready, etc)
2. Configure hardware (GPIO, etc) as if the device has no power applied
3. Finish init function with `return pm_device_driver_init(dev, bma400_pm_control);`

Assuming a conforming implementation of pm_control, automatically does the “right” thing:

- If device is powered off
    - Sets initial PM state to `PM_DEVICE_STATE_OFF`
- If device is powered
    - Runs `PM_DEVICE_ACTION_TURN_ON`
    - Sets initial PM state to `PM_DEVICE_STATE_SUSPENDED`
- If PM is not enabled:
    - Runs `PM_DEVICE_ACTION_TURN_ON` and `PM_DEVICE_ACTION_RESUME`

Cheap Wins

Potential cheap wins

UART is commonly used for logging and shell commands:

- Receiving is a power hog
- 1.1 mA (nRF52840)

Many Nordic peripherals depend on the HF Clock when enabled:

- Peripherals should be requested/released through PM device runtime.
- Savings depend on the crystal, 30 - 400 uA.

Internal oscillators are a bad time for everyone:

- Poor accuracy and stability
- Increased power consumption

DC/DC convertors should be used where they makes sense

Limit LED use, bright LEDs can easily hit 20mA

Iterative Improvements

Iteration Process

Create your own power budget table for the platform, then:

1. Measure the idle power consumption
2. Calculate difference from expected (Measured - Expected)
3. Iterate over potential causes
4. Once found, resolve the issue

Repeat until the measured current matches the expected current.
This approach is designed to get biggest wins first.
It ensures no regressions as changes are made.

Causes: Power budget table

Strong candidate if measured difference is roughly equal to the “boot - minimum” consumption. Keep in mind it may be 2 drivers wrong at the same time.

If the driver does not implement PM, implement it.

If the driver already implements PM:

1. pm_device_runtime_get/put() while measuring current
2. If difference does not match expectation, implementation is likely incorrect

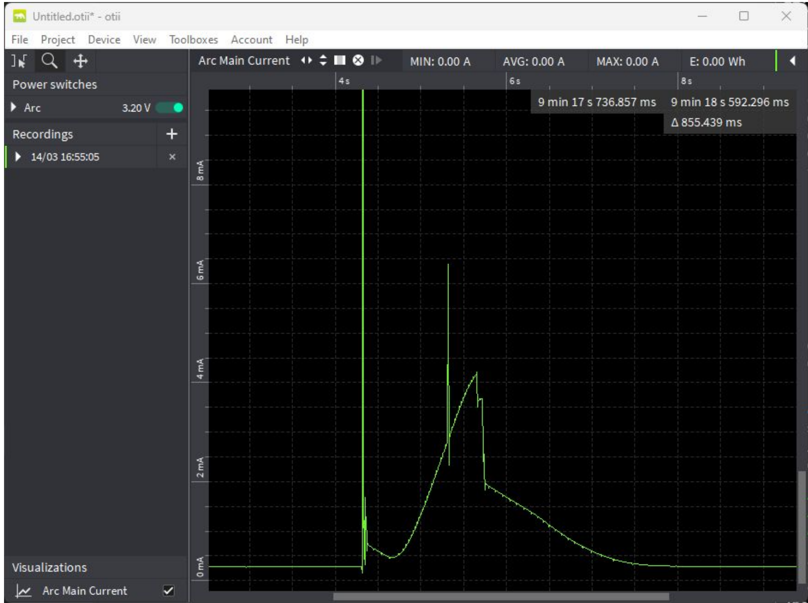
Causes: Pin Configuration

Floating pins are a common cause of leakage current.

Many peripherals have odd consumption profiles if pins are floating.

PINCTL can drive low power states for comms bus pins.

Also commonly observed with “back-powering”.



EMBEDDED OPEN SOURCE SUMMIT

Causes: Hardware

Convert the deviation to a resistance at the system voltage (V = IR).
Measure voltage across any resistors that match that value.

If using switched power domains, ensure expected domains are powered off.
If they are on, check for unbalanced get/put PM usage.

Poke every exposed pin on the board with a multimeter, find “weird” voltages:

- Often exposed as transient currents
- e.g. finding 1.4V on a 3.3V board warrants investigation

Causes: Errata

Perform a cursary review of all relevant errata documents.

Investigate anything that contains “Increased power consumption” or similar:

- Does it apply to your usage?
- Is a workaround already applied?

[78] TIMER: High current consumption when using timer STOP task only

Last Updated Mar 04, 2024 | 1 minute read | nRF52840 Errata

This anomaly applies to Revision 3, build codes QFAA-Fx0, QIAA-Fx0, CKAA-Fx0.

It was inherited from the previous IC revision Revision 2.

Symptoms

Increased current consumption when the timer has been running and the STOP task is used to stop it.

Conditions

The timer has been running (after triggering a START task) and then it is stopped using a STOP task only.

Consequences

Increased current consumption.

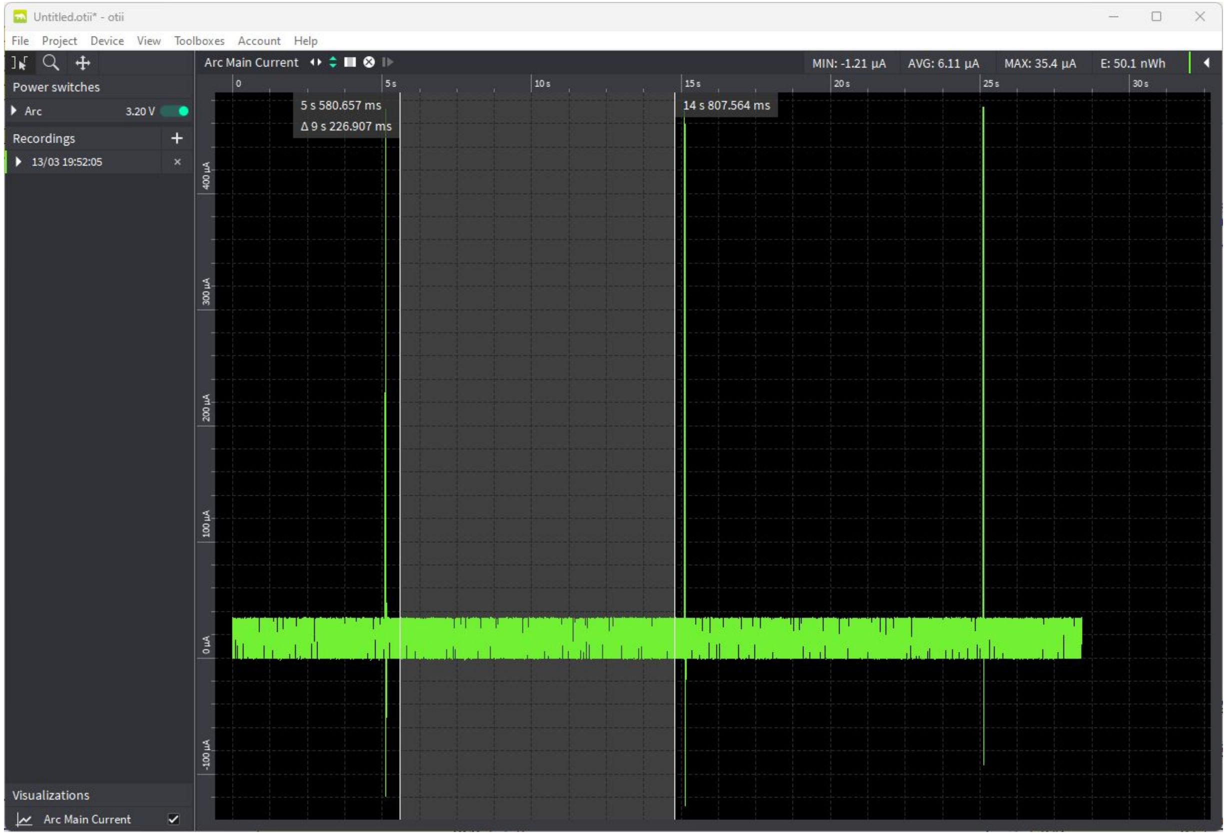
Workaround

Use the SHUTDOWN task after the STOP task or instead of the STOP task.

Causes: Measurement Setup

- Is there a LED on the programming adapter?
- Is your JTAG still logically connected?
  - Open connection (GDB, RTT) adds >3mA
- Is your JTAG still physically connected?
  - 30uA overhead on VDD_TARGET
- Is your test board just broken?
  - More likely in prototype runs, try multiple boards
- Environmental temperature plays a role at <10uA

Measured Power Consumption

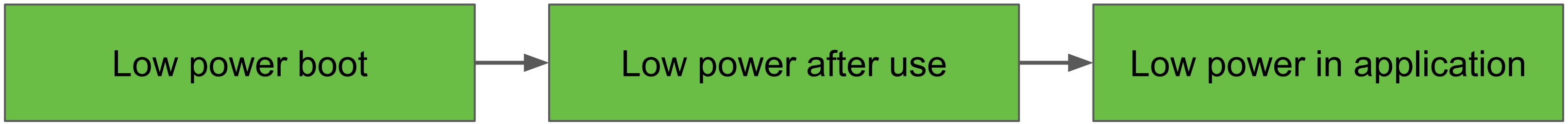


EOSS EMBEDDED OPEN SOURCE SUMMIT

Are we done?

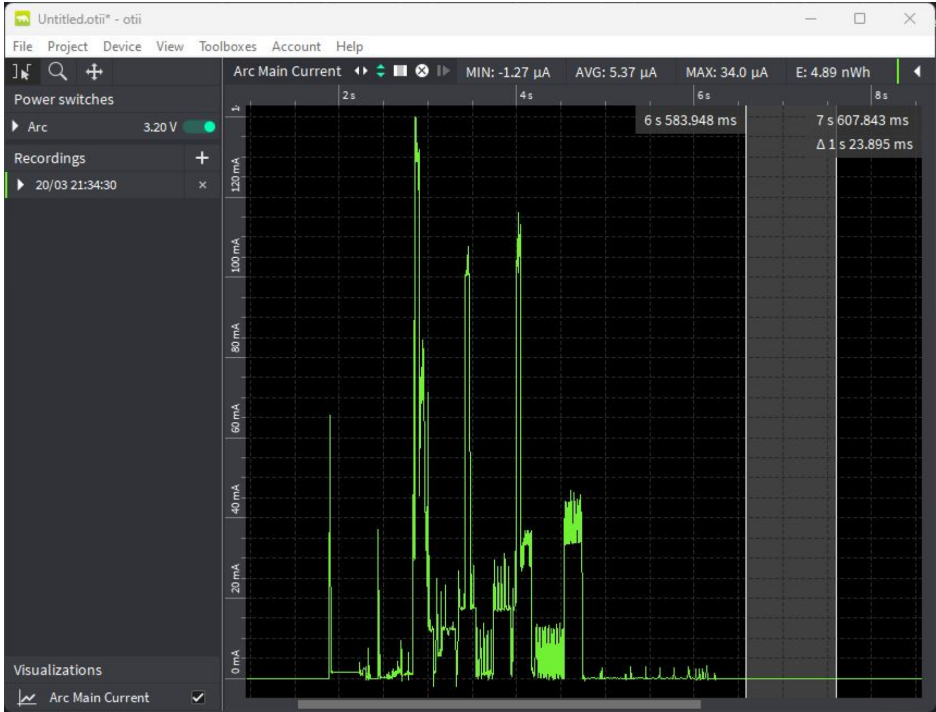
Not quite yet

- Board boots into a low power state
- Can it return to that low power state?
- Can your application return it to the low power state?



Validation Application

- Exercise every device & driver on the board
- Ensure board returns to low power
- Great opportunity to test drivers
    - Expected values, sample rates, etc
- Manufacturing test application?



EMBEDDED OPEN SOURCE SUMMIT

Validation Application Debugging

- Working baseline
    - Disable tests until you find the misbehaving driver

- Transitions towards low power don’t de-initialise resources
    - Device put in low power mode in `init`, not PM callback
    - GPIO pins not returned to `GPIO_DISCONNECTED`
    - Communications bus not released

Application Time

- Hardware proven to be low power
- Drivers proven to be low power
- Time to write application firmware
    - Think critically about when data is needed
    - Request when needed
    - Release when finished
- POWER PROFILE APPLICATION & COMPARE TO BASELINES

Zephyr® Project
Developer Summit