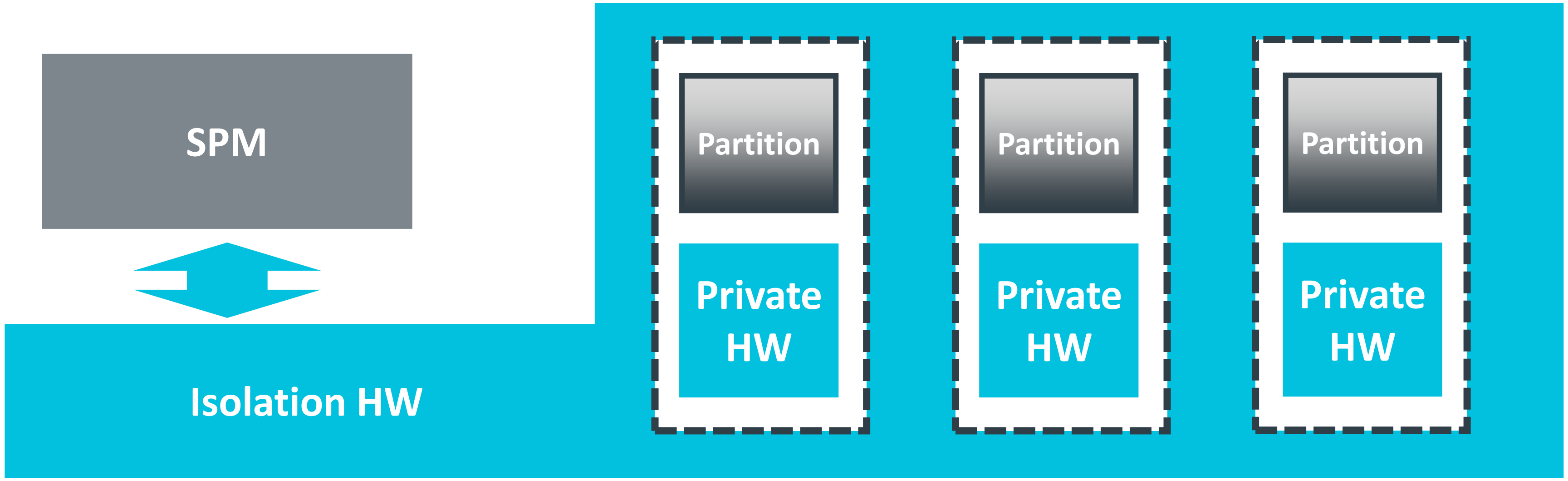
Secure Partition MMIO and Interrupt Binding

Ken Liu
Sep 2nd

Background

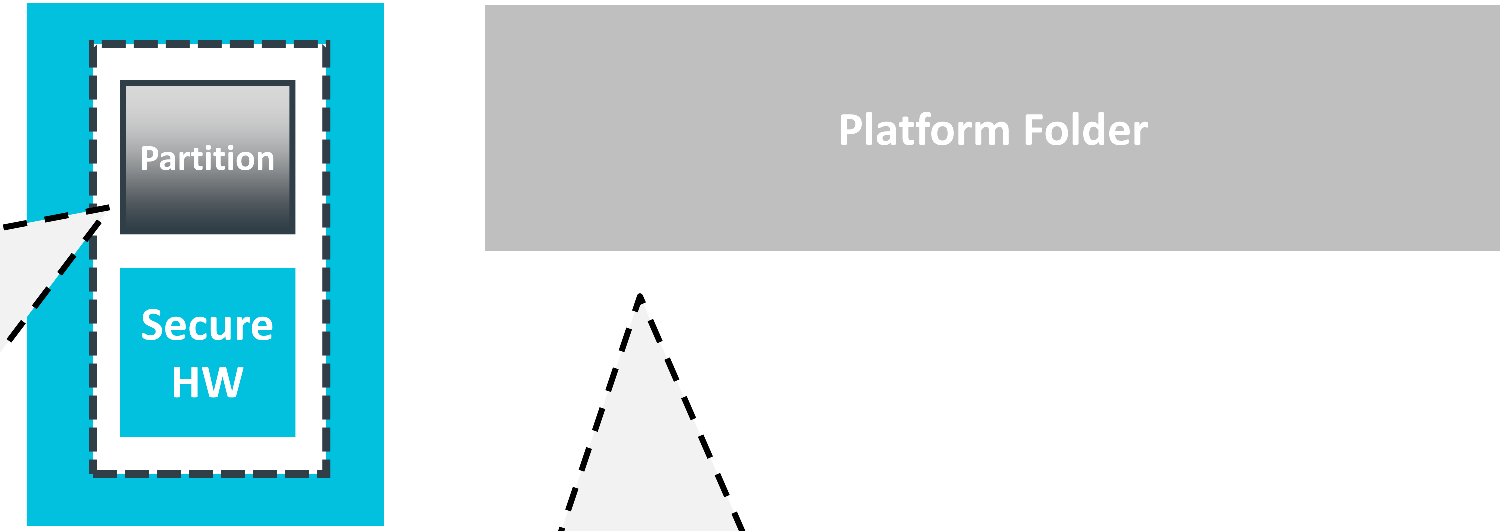
- FF-M requirements vs Practical Implementation
  - Based on the FF-M examples, partitions manipulate their own peripherals after claimed the required register address map.
  - While most of the peripheral drivers are provided as libraries already.
  - Interrupt is the similar case, and one more thing: IRQ vector needs to call SPM API to handle interrupt to follow FF-M handling process.



Background

- Code examples

```
"mmio_regions": [
    {
      "name": "TFM_PERIPHERAL_STD_UART",
      "permission": "READ-WRITE"
    },
    {
      "base": 0xE000D000,
      "size": 0x200
      "permission": "READ-ONLY"
    }
  ]
```

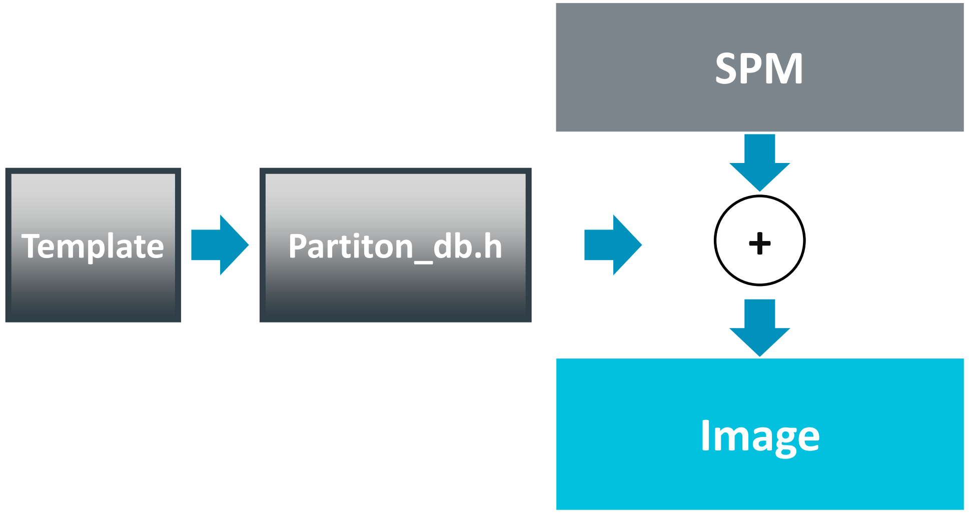


```
static const struct arm_uart_dev_cfg_t ARM_UART2_DEV_CFG_S = {
    .base = UART2_BASE_S,
    .default_baudrate = DEFAULT_UART_BAUDRATE};
};
```

How to link the partition with its peripherals?

- Concept: SPM does not want to get involved with peripherals code if possible.
- Situation: Partitions are selectable – do not involve the peripherals when owner is not included.
- Solution 1: Using template
  - What we were using.
  - **Hard to be maintained** – The template base needs to be updated every time new peripherals get involved because the intermedia data structure is put inside the template.
  - Still need platform code modification – HAL is there as the bridge.
  - SPM needs to be complied when configuration changed as the template output is a big header file.

```
{% for partition in partitions %}
   {% if partition.manifest.mmio_regions %}
      {% if partition.attr.conditional %}
#ifdef {{partition.attr.conditional}}
      {% endif %}
const struct platform_data_t *
   platform_data_list_{{partition.manifest.name}}[] =
{
      {% for region in partition.manifest.mmio_regions %}
         {% if region.conditional %}
```

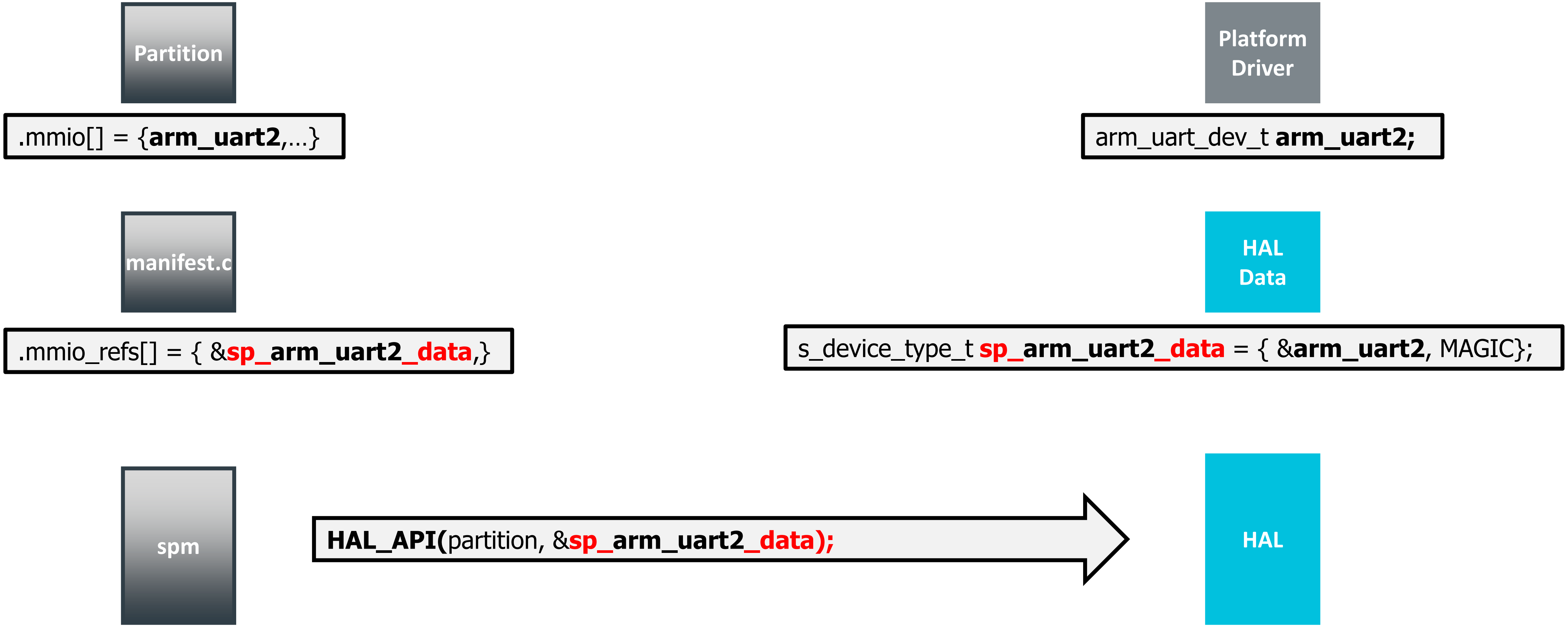


How to link the partition with its peripherals?

- **Solution 2 (Under upstreaming): More abstracted HAL**
- Assumption: The system designer need to decide the resource allocation for your system.
  - The driver code is already available, just need an allocation.
  - Define those secure drivers into a HAL required structure in C Source:
    - C source, no further learning is needed compared to the template solution.
    - If the symbol is not referenced, it is stripped by linker.
- The manifest tooling references platform symbols by name pattern.
  - This pattern is passed to platform to confirm and associate.
  - This process is called as ‘Binding’.
  - Don’t like the pattern? The pattern is also changeable for platform owner.

The solution diagram

- **Solution 2 (Under upstreaming): Advanced HAL**



Partition

.mmio[] = {**arm_uart2**,…}

Platform Driver

arm_uart_dev_t **arm_uart2;**

manifest.c

.mmio_refs[] = { &**sp_arm_uart2_data**,}

HAL Data

s_device_type_t **sp_arm_uart2_data** = { &**arm_uart2**, MAGIC};

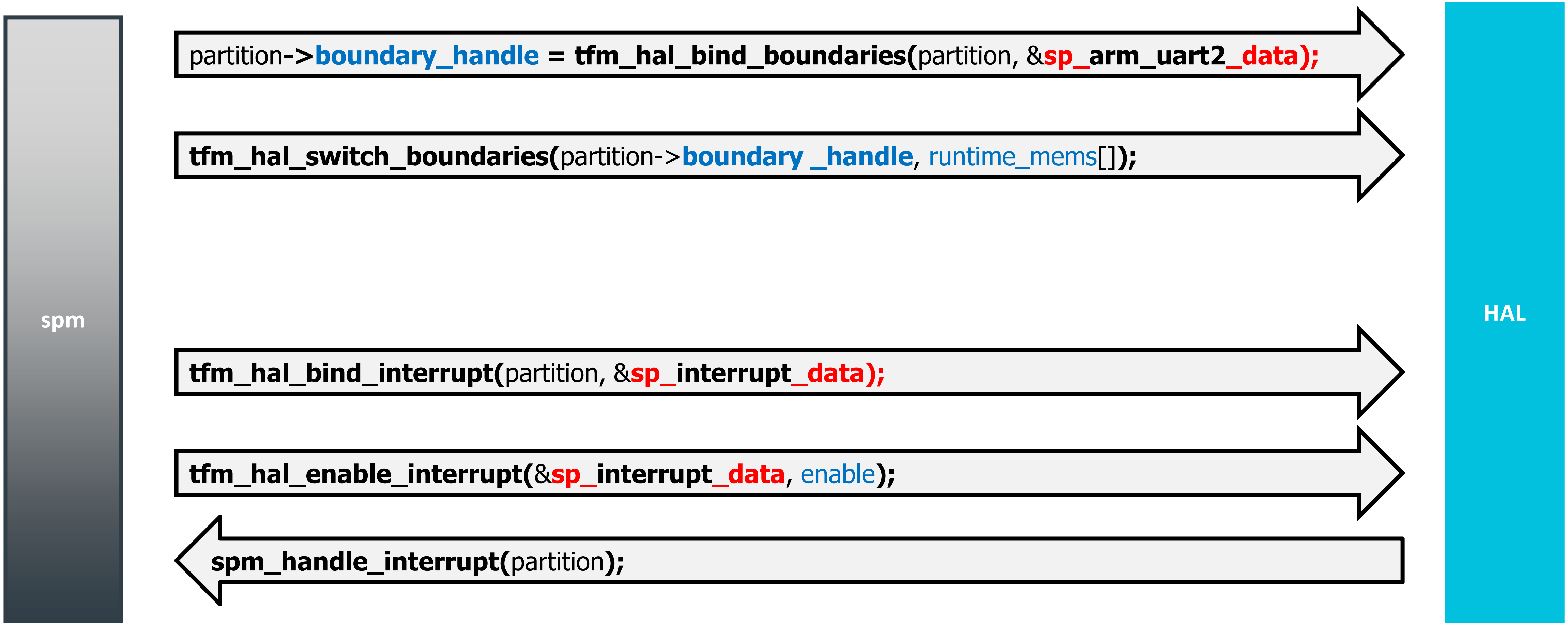
spm

HAL_API(partition, &**sp_arm_uart2_data);**

HAL

The solution process

- **Solution 2 (Under upstreaming): Advanced HAL**



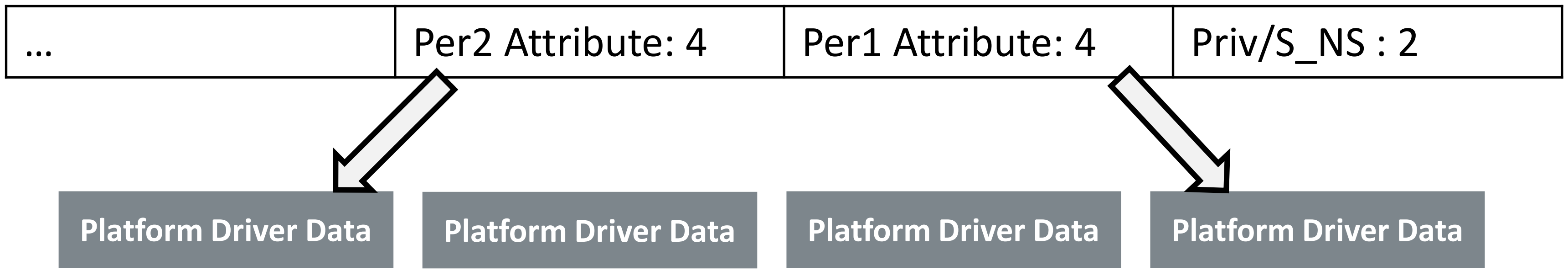
```
partition->boundary_handle = tfm_hal_bind_boundaries(partition, &sp_arm_uart2_data);
tfm_hal_switch_boundaries(partition->boundary _handle, runtime_mems[]);
tfm_hal_bind_interrupt(partition, &sp_interrupt_data);
tfm_hal_enable_interrupt(&sp_interrupt_data, enable);
spm_handle_interrupt(partition);
```

Challenges

- Platform drivers are put in the same sources.
  - Hard for putting them into separate regions, unless use tricky __attribute__.

- Leave more implementation decisions to platform.
  - Platform need to decide how to encode the handle for various purposes.
  - We provide examples.

partition->boundary_handle:



…	Per2 Attribute: 4	Per1 Attribute: 4	Priv/S_NS : 2

Patches

- Binding
  - https://review.trustedfirmware.org/c/TF-M/trusted-firmware-m/+/11036
- Correction
  - Remove ARM_LIB_STACK_MSP
- Upcoming changes
  - Init would be two HALs only: one before SPM runtime setup, one after that.
  - A default example is provided.

arm

Thank You

Danke

Gracias

谢谢

ありがとう

Asante

Merci

감사합니다

धन्यवाद

Kiitos

شكرًا

ধন্যবাদ

תודה