IRQ handling

Mate Toth-Pal

2019.10.31

IPC Model

- A secure partition can wait for an IRQ signal with psa_wait()
- On IRQ trigger a common handler runs, that sets a signal for the partition subscribed on this IRQ, and triggers scheduling
- The IRQ line gets disabled
- Scheduler decides based on priorities whether it should schedule the signaled partition
- IRQ handing partition uses `psa_eoi()` to signal that it finished handling the IRQ. TF-M core then re-enables the IRQ

Library Model

- A secure partition can register a handler that runs when a specific IRQ is triggered (name of the handler function is derived from the signal name specified in the manifest)
- IRQ is disabled by the TF-M core when the IRQ is triggered
- The partition has to call `psa_eoi()` to re-enable the IRQ (can be called from the handler)

Add Interrupt to a secure partition

- Add IRQ details to the secure partition's manifest
- Add privileged handler to the vector table (`irq_{{handler.source}}_Handler` or `{{handler.source}}_Handler` depending on source type)

IPC model

- Add `psa_wait()` call with the signal for the IRQ as parameter
- Call `psa_eoi()` once the IRQ is handled

Library model

- Create the handler function called `{{handler.signal}}_isr`
- Call `psa_eoi()` either from the handler, or from secure service.

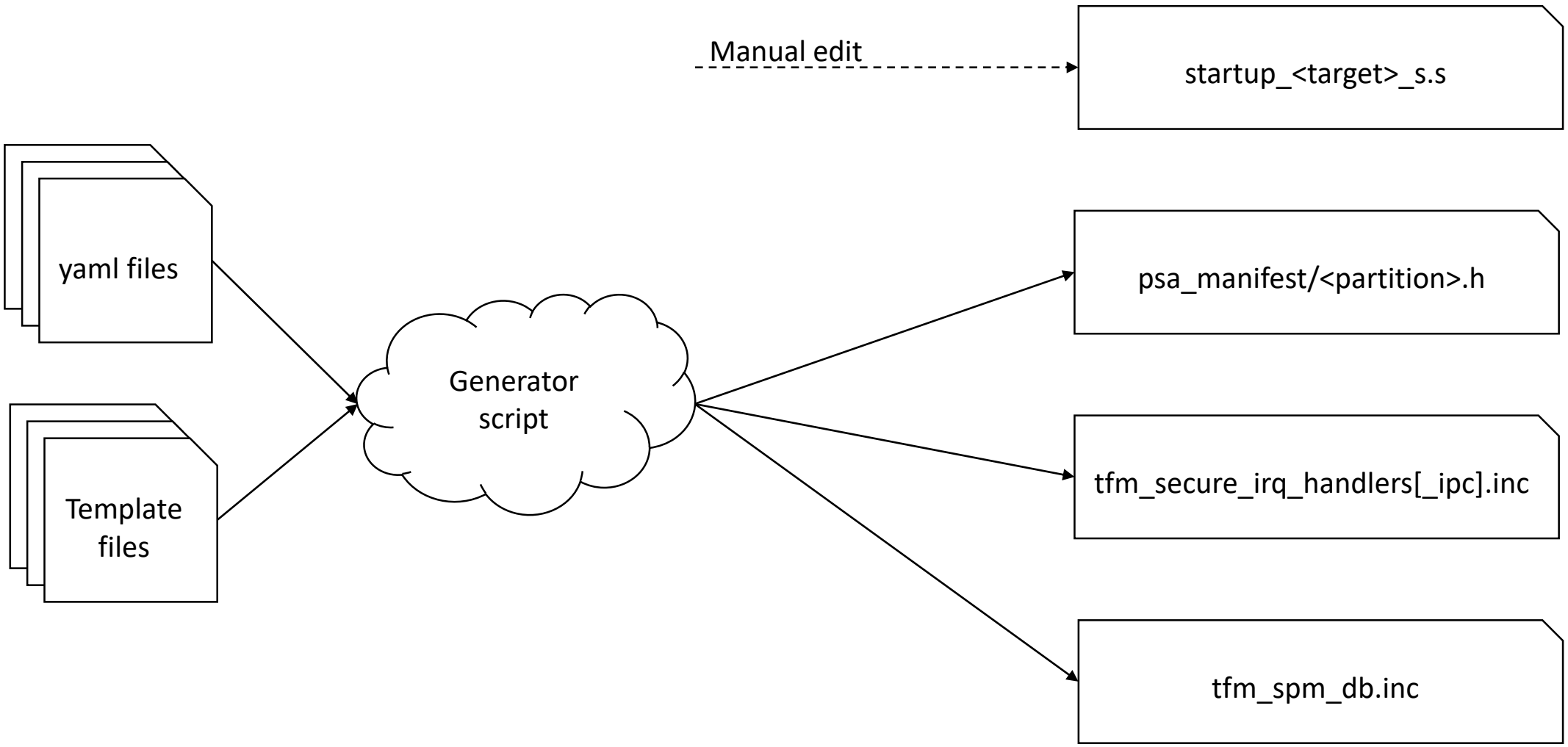
Possible improvement

- Generate startup files automatically, so there is no need to manually add the IRQ handler to the vector table
- Make Library model IRQ handling more lightweight (eliminate SVC calls)
- `psa_wait()` in library model is not useful, as currently there are no scheduling on the secure side

arm

Backup slides

Files to consider when adding assigning IRQ to a partition



arm

Yaml files

```
…
"irqs": [
    {
        "source": "TFM_TIMER0_IRQ",
        "signal": "SPM_CORE_IRQ_TEST_1_SIGNAL_TIMER_0_IRQ",
        "tfm_irq_priority": 64,
    }
],
…
```

tfm_secure_irq_handlers[_ipc].inc

```
…
/* Definitions of privileged IRQ handlers */
void TFM_TIMER0_IRQ_Handler(void)
{
    priv_irq_handler_main(TFM_IRQ_TEST_1_ID,
        (uint32_t)SPM_CORE_IRQ_TEST_1_SIGNAL_TIMER_0_IRQ_isr,
        SPM_CORE_IRQ_TEST_1_SIGNAL_TIMER_0_IRQ,
        TFM_TIMER0_IRQ);
}
…
```

tfm_secure_irq_handlers[_ipc].inc

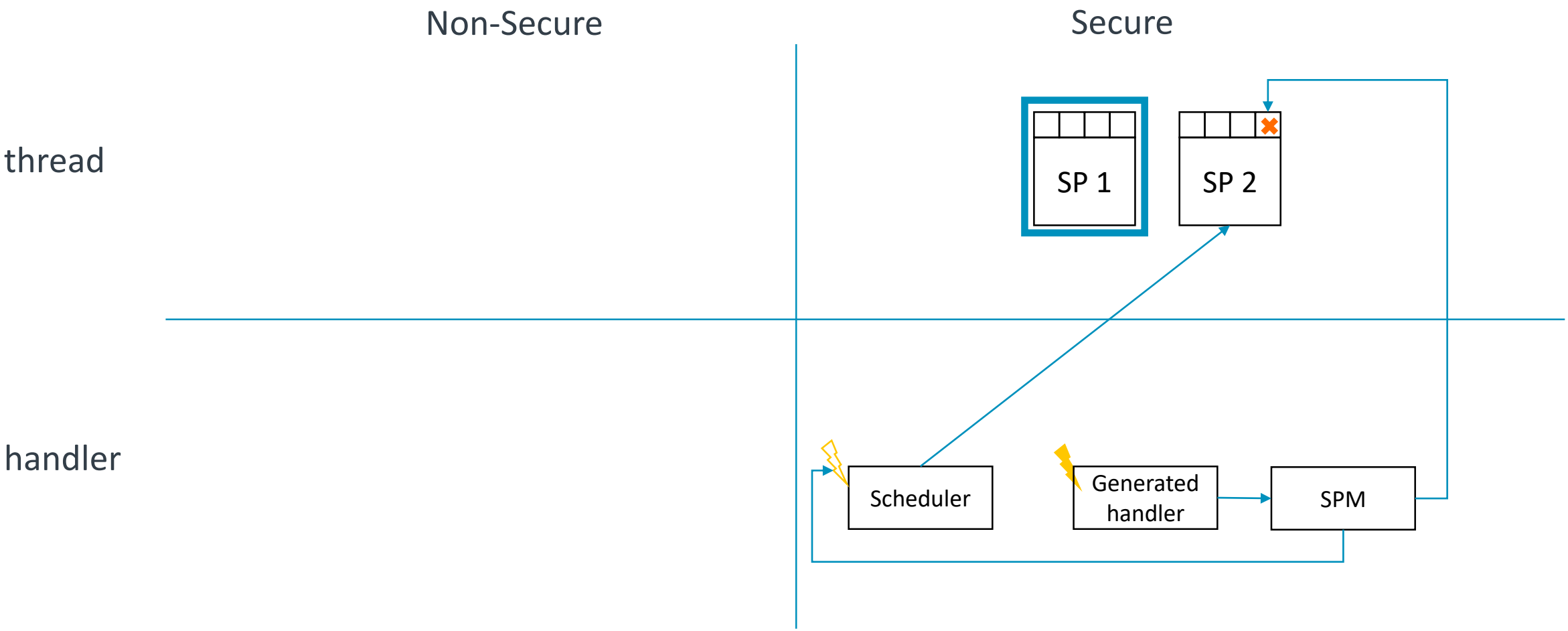
…

```
#define SPM_CORE_IRQ_TEST_1_PREPARE_TEST_SCENARIO_SIGNAL (1U << ( 0 + 4))
#define SPM_CORE_IRQ_TEST_1_EXECUTE_TEST_SCENARIO_SIGNAL (1U << ( 1 + 4))

#define SPM_CORE_IRQ_TEST_1_SIGNAL_TIMER_0_IRQ (1U << (27 + 4))
```

…

IRQ handling in IPC model



IRQ handling in IPC model