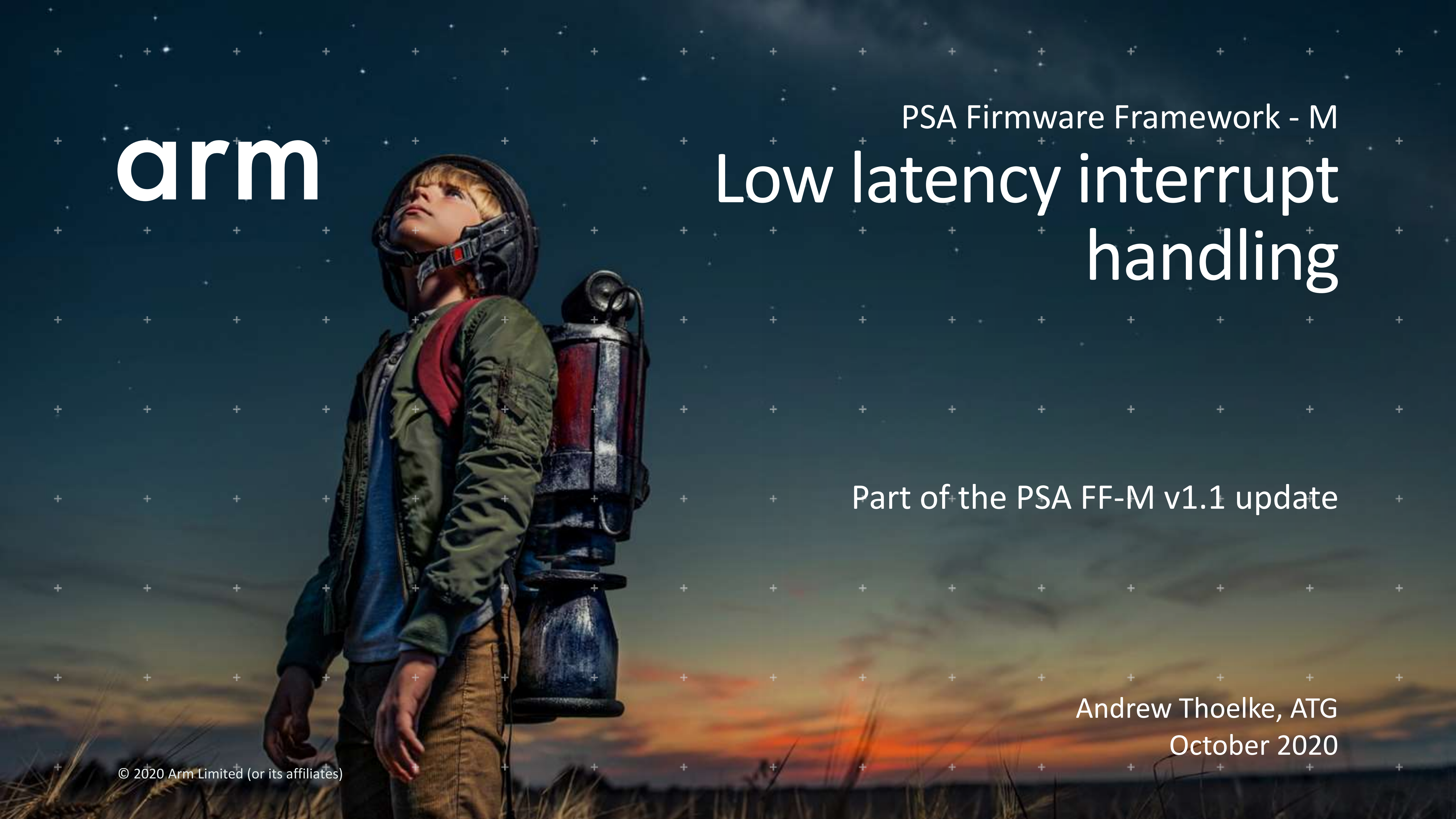

PSA Firmware Framework - M

Low latency interrupt handling

Part of the PSA FF-M v1.1 update

Andrew Thoelke, ATG
October 2020

Low latency interrupt handling

Part of the roadmap to *PSA Firmware Framework - M* v1.1

- Preamble: roadmap update

- Context
- Analysis
- Proposal

PSA FF-M v1.1 Roadmap update

- This is a roadmap proposal
  - We have proposals for all of the steps, which have evolved since the previous presentation.
  - An extension document to the PSA FF-M v1.0 specification will be published for review before finalising the new functionality

1. Stateless RoT Services
   - This was previously titled ‘Default handles (step 1)’, but that proposal is now merged with ‘Stateless services’ (step 6)
   - RoT Services that do not make use of connection features can be defined as *stateless*, which gives them a special build-time handle value that allows a client to request one-shot services without making an explicit connection. Stateless RoT Services do not receive *connection* or *disconnection messages*
2. Secure Functions
   - This introduces the SFN model as a per-SP option. Services are functions called by the framework, and use the IPC model APIs to read and write request parameters

Roadmap – continued

3. Memory mapped iovecs
   - This optional API introduces the ability for a service to directly read and write the client parameter memory. This will not work on all implementations, but is necessary for efficiency in simple systems
4. Low latency interrupt handling
   - This adds a deprivileged, low-latency, interrupt handling capability to SPs. FLIH functions cannot use normal Secure Partition APIs, but can signal the Secure Partition for later in-thread processing
5. Miscellaneous
   - Ensure alignment of functionality between SFN model and IPC model.
   - Clarifications and minor improvements to other aspects of the specification.

Context

PSA FF-M v1.0 provides limited support for handling interrupts

- The challenges presented by version 1.0 cover two issues:

1. The signal-based mechanism in PSA-FF-M v1.0 makes it difficult to write drivers that need interrupts to be handled in a bounded time
   - Interrupt handling code runs within a Secure Partition thread, which is subject to delays due to scheduling of other threads and due to completion of current activity within the Secure Partition thread

2. The simple interface in v1.0 assumes that dynamic interrupt control (outside of the peripheral MMIO interface) is not needed. This leaves a gap that is currently filled by implementation specific APIs
   - For example, during initialization it is better for interrupts to be disabled until the Secure Partition is able to finish initialising the peripheral device

Analysis

- The v1.0 API was designed to be simple and easy to use securely. It avoided concurrent execution within the Secure Partition by requiring the interrupt handler to run within the execution thread
  - Working around the limitations of this API for lower-latency interrupt requirements is painful, complex and error-prone
  - Framework support for immediate, asynchronous interrupt handling is needed in these use cases
- The v1.0 API assumed that peripheral interrupts are all ‘behaved well’:
  - The interrupt can be disabled at-source using a peripheral control register
  - The peripheral resets with all interrupts disabled
- In reality, not all peripherals meet these requirements either due to design decisions or implementation defects. PSA-FF-M needs to accommodate this reality
  - This requires changes to the interrupt model, and the addition of interfaces to support managing the interrupts at the CPU/Interrupt Controller

Terminology

- Many operating systems have the concept of two levels of interrupt handling
  - Initial handling that occurs within the context of the interrupt exception, blocking all normal scheduled execution. Any interrupt response activity with bounded response time requirements is done here. The initial handling may run in a privileged or deprivileged context
  - Deferred handling that occurs within a thread/task context, subject to normal scheduling. This is often triggered from the initial handler, and can use the full set of OS functionality

- In PSA FF-M we use the term *First-level interrupt handling* (FLIH) for the initial handling, and *Second-level interrupt handling* (SLIH) for the deferred activity
- Using these definitions:
  - PSA FF-M v1.0 only permitted a Secure Partition developer to use SLIH. The FLIH was part of the framework, and this only set the interrupt signal for the Secure Partition

Proposal – support for FLIH and SLIH

***Note:** this is an initial proposal, and open for review, feedback and update*

- An interrupt in a Secure Partition can choose between FLIH and SLIH handling modes
- SLIH mode works the same way as interrupt handling in v1.0
  - Sets the interrupt signal, which is detected in a call to `psa_wait()`
  - Requires the Secure Partition thread (or SFN) to call `psa_eoi()` when interrupt processing is complete
- FLIH mode requires the developer to provide a callback function for the interrupt
  - The FLIH function is executed asynchronously when the interrupt occurs
  - Execution occurs within the Secure Partition memory context – it is deprivileged
  - Functionality is limited
    - FLIH functions cannot block, and most of the Secure Partition API is not available
  - Execution can pre-empt other code in the same Secure Partition
    - Care is required to avoid race conditions
  - End-of-interrupt processing occurs when the FLIH function returns
  - The FLIH function can optionally set the interrupt signal in the Secure Partition
    - Enables the SP thread/SFN to process interrupt results, for example by replying to a message

Proposal – interrupt control

***Note:** this is an initial proposal, and open for review, feedback and update*

- Provide a flexible IRQ control API
  - Allows Secure Partition code to enable and disable interrupts belonging to the Secure Partition
  - Also allows for Implementation specific extensions for managing other interrupt features
- Interrupts are disabled before first entry to any Secure Partition code
  - This is a change from the v1.0 interrupt model, but was typically required in implementations anyway, which provided an implementation-specific API to manage the interrupt
  - The Secure Partition must enable the interrupt explicitly to start receiving interrupt FLIH callbacks or interrupt signals, after initializing the peripheral

Next steps

- Compile all of the proposals into a PSA FF-M v1.1-alpha update specification
- This will be published for review and feedback before finalising the new features.

- Please provide feedback on this proposal, or the roadmap in the TF-M mailing list, or to arm.psa-feedback@arm.com

Thank You

Danke

Merci

谢谢

ありがとう

Gracias

Kiitos

감사합니다

धन्यवाद

شكرًا

ধন্যবাদ

תודה