arm

Trusted Firmware-M Documentation Proposal

11th April 2019

Goals

The main goal of this proposal is to define a consistent and scalable methodology/process for the documentation of TF-M development. This covers both existing documents and design proposal documents.

This proposal includes:

- The process of creating, maintaining and publishing the documentation (lifecycle)
- The proposed format
- The proposed content structures
- The tools needed to support the above

arm

Design proposal process

Requirements and assumptions

During the consultation and initial investigation phase the following requirements were gathered and assumptions made

- Reviews could take place on the mailing list or with smaller audience
- A design proposal may need to have restricted access before going for a public review
- The documents should ideally be in a version controllable text format
- We should aim for a consistent format across all documentation
- We should align with industry best practices – both in content and format – if possible. Python's PEP review processes were used as a guide for designing our own.

Design Proposal Process

- Write the design proposal in the format that is described in this document with the status set to *Draft* and create a pull request.
- The maintainers accept the pull request and after merge it is visible in the TF-M documentation
- Start an email thread on the TF-M mailing list for discussing the proposal.
- Build initial consensus within the community about the proposed design change, rework it according to the feedbacks and identify members who would like to participate in the detailed review.
- When the shortlist of members who are willing to participate in the detailed review is established, set the status field to *Detailed*, push it on for Gerrit review and add the interested people as reviewers.
- The detailed discussion then takes place in the gerrit review and gets recorded there.
- Additional changes are submitted as new commits to the review.
- When the proposal is accepted and merged, the status field can be set to *Accepted* and the design proposal can be moved under the accepted design documents section.
- If at any point the proposal is rejected its status field is set to *Rejected*.

arm

Formats and structure

Content Structure

Introduction
- Licensing
- Glossary

Getting Started Guide

Contribution Guidelines
- Maintainers
- Design Proposal Process

User Guides
- OS migration
- TF-M Integration Guide
- Trusted Firmware M User Guide
- Non-Secure Identity Manager
- Secure Boot
- Adding a new service

Services
- Attestation
- Audit
- Crypto
- Platform
- Secure Storage
- Adding a new service

Design Documents
- Design Proposal Process
- Accepted Design Documents
- Proposed Changes/Amendments for the design

Security
- Threat Modelling
- Secure Coding
- Code Review

Tools
- Software requirements
- Build Instructions
- Continuous Integration

API documentation

Release Notes

arm

Format and tools

Currently the design documents are in Markdown format and the proposals are in Phabricator's Remarkup. The proposal is to use **reStructured** text across the board.

- reStructured text is standardized while Markdown isn't
- Remarkup seems to be Phabricator specific

Use **Sphinx** as the documentation renderer:

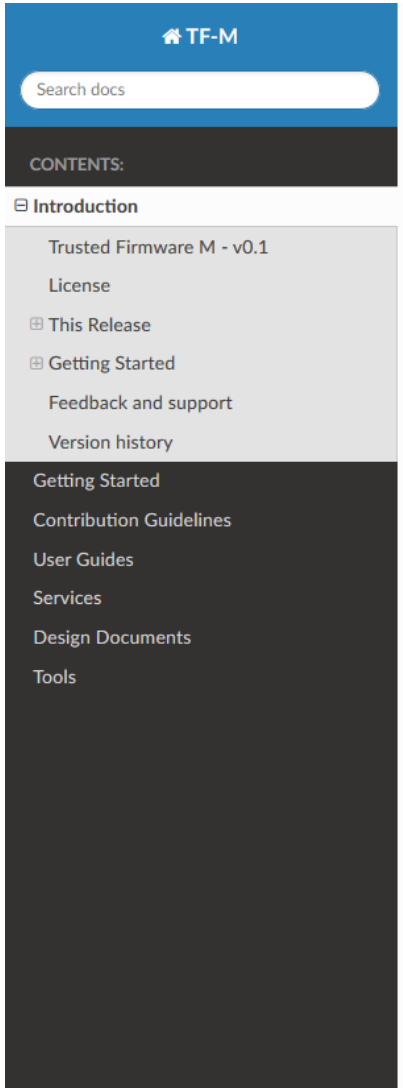
- Widespread and mature
- Comes with several HTML styles
- Can generate API docs from Doxygen via the Breathe extension
- Used by many-many open source projects including Zephyr

Current Progress

- Converted existing content from Markdown to ReStructured Text format using Pandoc
  - Generally good quality output with some minor, manual changes required
- Organised documents according to the proposed content structure
- Generated HTML output using Sphinx with the ReadTheDocs theme
- Used Gerrit to review a design proposal in the new format
  - Restricted reviews are possible by marking the change as Private.

How this looks in practice

- Integrated search tool
- Navigation toolbar / table of contents always present on the left side
- High-quality rendering of RST source as HTML, with support for content such as code blocks, raw formatting, notes ,etc.



Introduction

Trusted Firmware M - v0.1

Trusted Firmware M provides a reference implementation of secure world software for ARMv8-M.

Note: The software implementation contained in this project is designed to be a reference implementation of the Arm Platform Security Architecture (PSA). It currently does not implement all the features of that architecture, however we expect the code to evolve over 2018 along with the specifications.

```
Terms 'TFM' and 'TF-M' are commonly used in documents and code and both refer to Trusted Firmware M.
```

TF-M glossary of terms and abbreviations has the list of terms and abbreviations.

License

The software is provided under a BSD-3-Clause License. Contributions to this project are accepted under the same license with developer sign-off as described in the Contributing Guidelines.

This project contains code from other projects as listed below. The code from external projects is limited to `app` and `platform` folders. The original license text is included in those source files.

- The `platform` folder currently contains drivers imported from external project and the files have Apache 2.0 license.
- The `app` folder contains files imported from CMSIS_5 project and the files have Apache 2.0 license.
- The `bl2` folder contains files imported from MCUBoot project and the files have Apache 2.0 license.

Note Any code that has license other than BSD-3-Clause is kept in specific sub folders named `ext` so that it can isolated if required.

arm

Technicalities

A few minor things

- All documentation files should be under the designated <docs> directory. Currently the readme, glossary, maintainers, etc. files are in the root directory
- The _build directory that contains the rendered HTML should be added to the .gitignore file
- After having rendered the HTML, all the .rst files are actually copied into the _build directory and errors are flagged on that version – make sure to fix them at source
- Although .md files should be supported by Sphinx it didn’t give good results with the flavour we use

Features that are now available

Using the ReStructured Text format provides several advantages over Markdown:

- Using Sphinx linkcheck validates internal and external links
- Linking is done by page title, not path, so moving a file doesn't break links
- Intersphinx extension allows linking between other document trees
- Extlinks extension enables long, repeated URLS to be abbreviated
- Breathe extension allows Doxygen output to be integrated. Makes very nice API documentation. Example at: https://xtensor.readthedocs.io/en/latest/api/xio.html
- Integrated glossary functions (.. glossary:: and .. term::)

Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
תודה