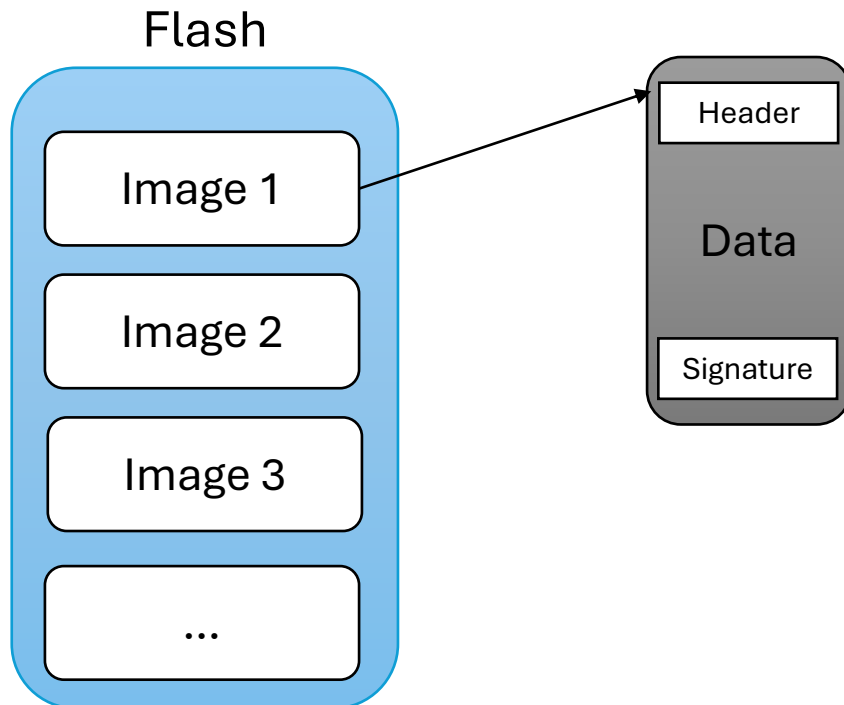


Single Signature Check for Multi Image

Sadik Ozer

MCUboot Multi Image Feature

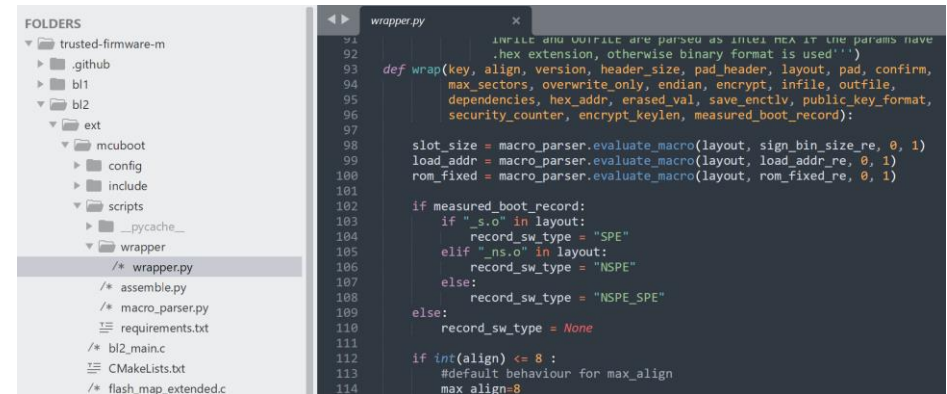
- MCUboot designed to support multiple images
 - Each image process independently and individually.
 - Each image has its header & footer sections and able to be updated independently.
 - **MCUBOOT_IMAGE_NUMBER** macro is used to set number of image.



```
243  /** Private state maintained during boot. */
244  struct boot_loader_state {
245      struct {
246          struct image_header hdr;
247          const struct flash_area *area;
248          boot_sector_t *sectors;
249          uint32_t num_sectors;
250      } imgs[BOOT_IMAGE_NUMBER][BOOT_NUM_SLOTS];
251  }
```

Multi Image with TFM

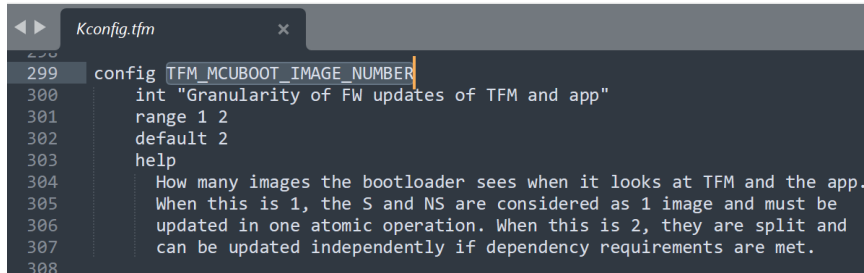
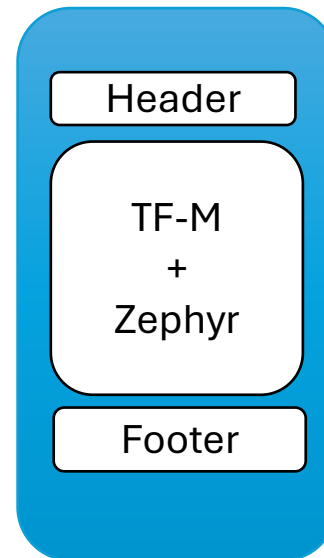
- TFM allow concatenate zephyr and tfm images and process them as a single image.
- Requires to set **CONFIG_TFM_MCUBOOT_IMAGE_NUMBER=1**
- Single signature check.
- There is wrap.py script that generate signature for single image



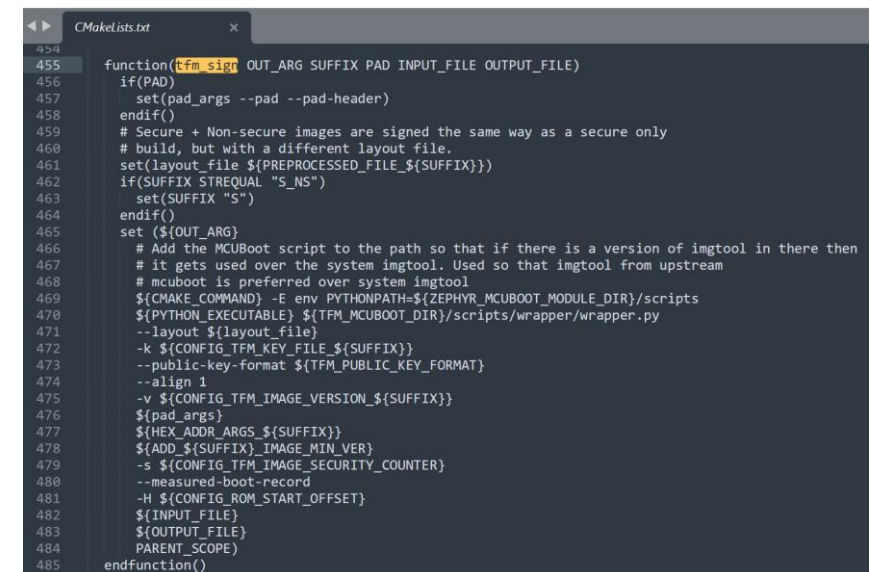
```
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
```

```
def wrap(key, align, version, header_size, pad_header, layout, pad, confirm,
max_sectors, overwrite_only, endian, encrypt, infile, outfile,
dependencies, hex_addr, erased_val, save_encrypt, public_key_format,
security_counter, encrypt_keylen, measured_boot_record):
    slot_size = macro_parser.evaluate_macro(layout, sign_bin_size_re, 0, 1)
    load_addr = macro_parser.evaluate_macro(layout, load_addr_re, 0, 1)
    rom_fixed = macro_parser.evaluate_macro(layout, rom_fixed_re, 0, 1)
    if measured_boot_record:
        if "_s.o" in layout:
            record_sw_type = "SPE"
        elif "_ns.o" in layout:
            record_sw_type = "NSPE"
        else:
            record_sw_type = "NSPE_SPE"
    else:
        record_sw_type = None
    if int(align) <= 8 :
        #default behaviour for max_align
        max_align=8
```

Single Image



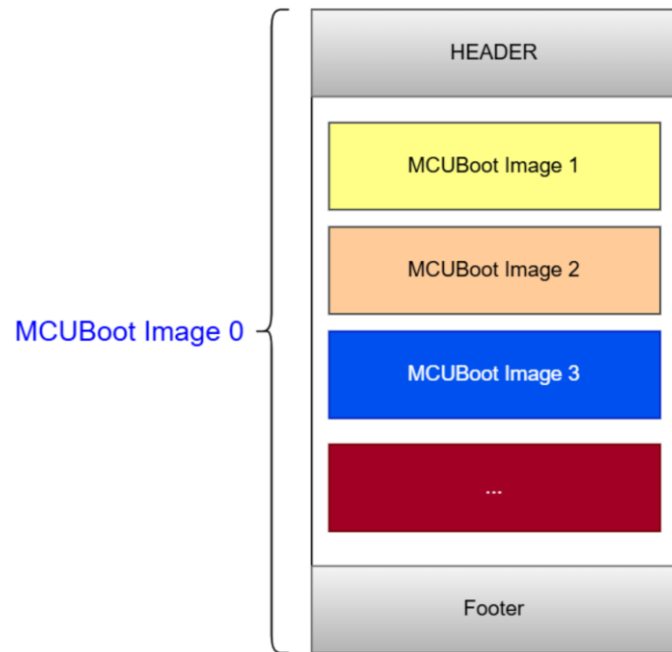
```
299 config TFM_MCUBOOT_IMAGE_NUMBER
300     int "Granularity of FW updates of TFM and app"
301     range 1 2
302     default 2
303     help
304         How many images the bootloader sees when it looks at TFM and the app.
305         When this is 1, the S and NS are considered as 1 image and must be
306         updated in one atomic operation. When this is 2, they are split and
307         can be updated independently if dependency requirements are met.
308
```



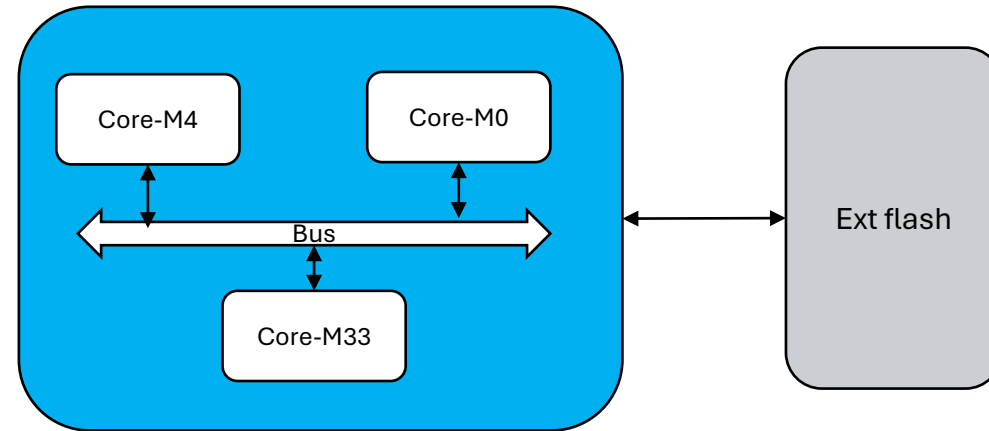
```
454
455 function(tfm_sign OUT_ARG SUFFIX PAD INPUT_FILE OUTPUT_FILE)
456     if(PAD)
457         set(pad_args --pad --pad-header)
458     endif()
459     # Secure + Non-secure images are signed the same way as a secure only
460     # build, but with a different layout file.
461     set(layout_file ${PREPROCESSED_FILE_${SUFFIX}})
462     if(SUFFIX STREQUAL "S_NS")
463         set(SUFFIX "S")
464     endif()
465     set(${OUT_ARG}
466         # Add the MCUBoot script to the path so that if there is a version of imgtool in there then
467         # it gets used over the system imgtool. Used so that imgtool from upstream
468         # mcuboot is preferred over system imgtool
469         ${CMAKE_COMMAND} -E env PYTHONPATH=${ZEPHYR_MCUBOOT_MODULE_DIR}/scripts
470         ${PYTHON_EXECUTABLE} ${TFM_MCUBOOT_DIR}/scripts/wrapper/wrapper.py
471         --layout ${layout_file}
472         -k ${CONFIG_TFM_KEY_FILE_${SUFFIX}}
473         --public-key-format ${TFM_PUBLIC_KEY_FORMAT}
474         --align 1
475         -v ${CONFIG_TFM_IMAGE_VERSION_${SUFFIX}}
476         ${pad_args}
477         ${HEX_ADDR_ARGS_${SUFFIX}}
478         ${ADD_${SUFFIX}_IMAGE_MIN_VER}
479         -s ${CONFIG_TFM_IMAGE_SECURITY_COUNTER}
480         --measured-boot-record
481         -H ${CONFIG_ROM_START_OFFSET}
482         ${INPUT_FILE}
483         ${OUTPUT_FILE}
484         PARENT_SCOPE)
485 endfunction()
```

Proposal

- Reason: Multi core system may not need multiple signature check due to some boot time restrictions.
- Add a python script to generate image individually. Similar to TF-M solution.
- Then concatenate them and generate main image.
- MCUboot will validate main image and copy sub images to the target address.
- <https://github.com/mcu-tools/mcuboot/pull/2465>



A multi core device



Script to combine images

- As TF-M solution (wrapper.py) **combine_images.py** script added to combines images.
- Added **combine_images.yaml** file which defines each images input parameter
- <https://github.com/mcu-tools/mcuboot/pull/2465>
- `python scripts/combine_images.py --config <USER_YAML_FILE> --imgtool imgtool --output <OUT_FOLDER>`

```
53 scripts/combine_images.yaml
18 +
19 + combined_app_pack:
20 +   outputfile: combined
21 +
22 +   header:
23 +     private_signing_key: ../root-rsa-2048.pem
24 +     header-size: 0x400
25 +     align: 4
26 +     load-addr: 0x20000000
27 +     pad-header: yes
28 +     version: 1.0.0
29 +     slot-size: 0x40000
30 +
31 +   image1_pack:
32 +     outputfile: image1
33 +     header:
34 +       private_signing_key: ../root-rsa-2048.pem
35 +       header-size: 0x400
36 +       align: 4
37 +       load-addr: 0x20010000
38 +       pad-header: yes
39 +       version: 1.0.0
40 +       slot-size: 0x10000
41 +     image: image1.bin
42 +
43 +   image2_pack:
44 +     outputfile: image2
45 +     header:
46 +       private_signing_key: ../root-rsa-2048.pem
47 +       header-size: 0x400
48 +       align: 4
49 +       load-addr: 0x20020000
50 +       pad-header: yes
51 +       version: 1.0.0
52 +       slot-size: 0x10000
53 +     image: image2.bin
```

```
178 scripts/combine_images.py
32 + def main():
33 +     global img_tool
34 +     global output_dir
35 +     global config_path
36 +
37 +     parser = argparse.ArgumentParser(description="Create application package", allow_abbrev=False)
38 +     parser.add_argument('--config', help="The path to config yaml file", required=True)
39 +     parser.add_argument('--imgtool', help="The path to ImgTool", required=True)
40 +     parser.add_argument('--output', help="Output directory", required=True)
41 +
42 +     args = parser.parse_args()
```

Thank You!