mbed OS
Adding Bootloader Support

ARM

Martin Kojtal

Senior Software Engineer
ARM mbed

Silicon Partner Workshop - Wyboston Lakes
March 2017

Agneda

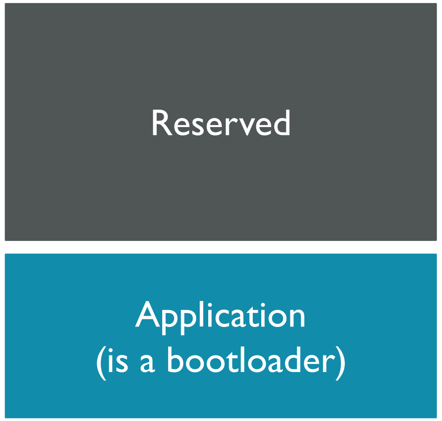
- Overview
- Flash **I**n-**A**pplication-**P**rogramming
- Flash HAL
- Workshop

Overview

Bootloader

- A bootloader is an application
- mbed_app.json target member **restrict_size**
  - Restricts the bootloader code from growing larger than the specified size.
  - Pads the output image to exactly the size specified.

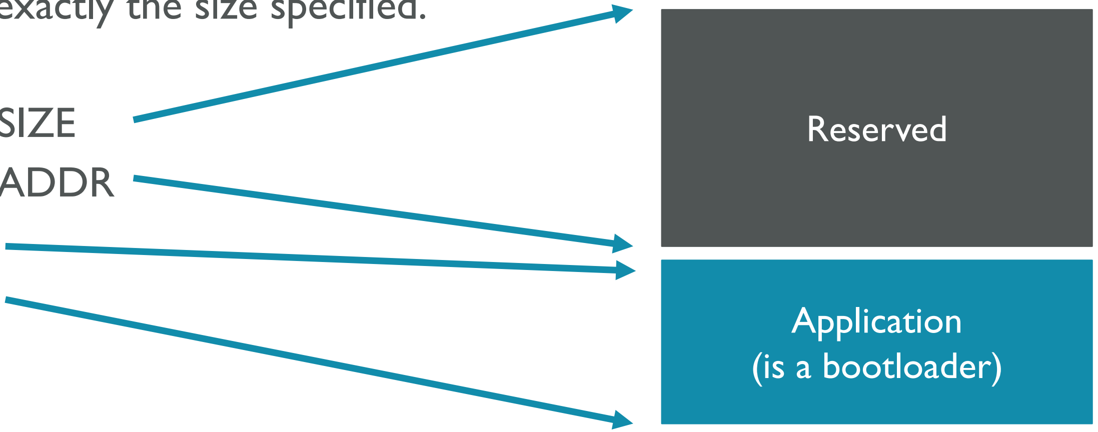
```
{
    "target_overrides": {
        "K64F": {
            "target.restrict_size": "0x20000"
        },
        "NUCLEO_F429ZI": {
            "target.restrict_size": "0x20000"
        },
        "UBLOX_EVK_ODIN_W2": {
            "target.restrict_size": "0x20000"
        }
    }
}
```



https://docs.mbed.com/docs/mbed-os-handbook/en/latest/advanced/bootloader/

Bootloader

- A bootloader is an application
- mbed_app.json target member **restrict_size**
  - Restricts the bootloader code from growing larger than the specified size.
  - Pads the output image to exactly the size specified.
  - Defines the symbols:
    - POST_APPLICATION_SIZE
    - POST_APPLICATION_ADDR
    - APPLICATION_SIZE
    - APPLICATION_ADDR

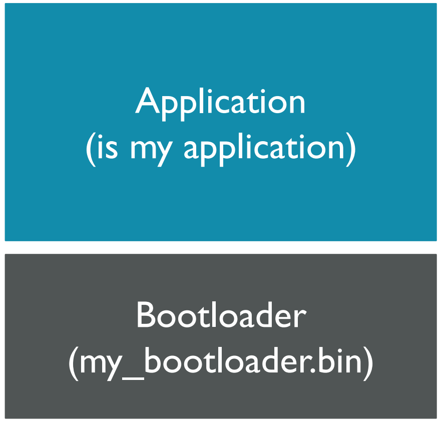


https://docs.mbed.com/docs/mbed-os-handbook/en/latest/advanced/bootloader/

Applications

- An application is also an application
- mbed_app.json target member **bootloader_img**
  - The application is automatically combined with the bootloader to create the final image
  - Start of application determined by size of bootloader_img

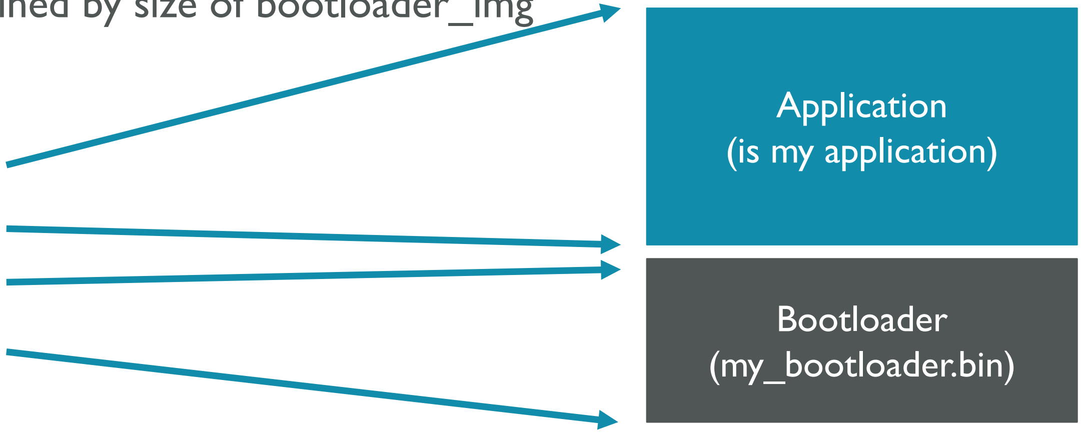
```
{
    "target_overrides": {
        "K64F": {
            "target.bootloader_img": "bootloader/K64F.bin"
        },
        "NUCLEO_F429ZI": {
            "target.bootloader_img": "bootloader/NUCLEO_F429ZI.bin"
        },
        "UBLOX_EVK_ODIN_W2": {
            "target.bootloader_img": "bootloader/UBLOX_EVK_ODIN_W2.bin"
        }
    }
}
```



https://docs.mbed.com/docs/mbed-os-handbook/en/latest/advanced/bootloader/

Applications

- An application is an application
- mbed_app.json target member **bootloader_img**
  - The application is automatically combined with the bootloader to create the final image
  - Start of application determined by size of bootloader_img
  - Defines the symbols:
    - APPLICATION_SIZE
    - APPLICATION_ADDR
    - BOOTLOADER_SIZE
    - BOOTLOADER_ADDR



https://docs.mbed.com/docs/mbed-os-handbook/en/latest/advanced/bootloader/

How to enable this?

- Enabling bootloader capability in mbed OS
  - Implement flash HAL API – FlashIAP class invokes flash HAL
    - Direct SDK implementation
    - CMSIS flash algo blob implementation
  - Modify linker script
  - bootloader_supported flag in targets.json

```
{
    "Target": {
        "supported_toolchains": null,
        "extra_labels": [],
        "macros": [],
        "device_has": [],
        "features": [],
        "bootloader_supported": false
    },
}
```

FlashIAP

FlashIAP

- Thread safety level
  - Protected by mutex
- Gotchas
  - Latency
  - No protection of the flash device
  - Other device specific considerations
    - Max clock speed
    - Sector erase to page write ratio
    - Etc…

mbed::FlashIAP::init

```
namespace mbed {

  class FlashIAP {
  public:

     /** Initialize a flash IAP device
      *
      *  Should be called once per lifetime of the object.
      *  @return 0 on success or a negative error code on failure
      */
      int init();

     /** Deinitialize a flash IAP device
      *
      *  @return 0 on success or a negative error code on failure
      */
      int deinit();
  };
}
```

ARM

mbed::FlashIAP::read

```
namespace mbed {

  class FlashIAP {
  public:

     /** Read data from a flash device.
      *
      *  This method invokes memcpy - reads number of bytes from the address
      *
      *  @param buffer Buffer to write to
      *  @param addr   Flash address to begin reading from
      *  @param size   Size to read in bytes
      *  @return       0 on success, negative error code on failure
      */
      int read(void *buffer, uint32_t addr, uint32_t size);
  };
}
```

mbed::FlashIAP::program

```
namespace mbed {

  class FlashIAP {
  public:

      /** Program data to pages
       *
       *  The sectors must have been erased prior to being programmed
       *
       *  @param buffer Buffer of data to be written
       *  @param addr   Address of page to begin writing to must be a multiple of program and sector sizes
       *  @param size   Size to write in bytes, must be a multiple of program and sector sizes
       *  @return       0 on success, negative error code on failure
       */
      int program(const void *buffer, uint32_t addr, uint32_t size);
  };
}
```

mbed::FlashIAP::erase

```
namespace mbed {

  class FlashIAP {
  public:

     /** Erase sectors
      *
      *  The state of an erased sector is undefined until it has been programmed
      *
      *  @param addr Address of a sector to begin erasing, must be a multiple of the sector size
      *  @param size Size to erase in bytes, must be a multiple of the sector size
      *  @return     0 on success, negative error code on failure
      */
      int erase(uint32_t addr, uint32_t size);
  };
}
```

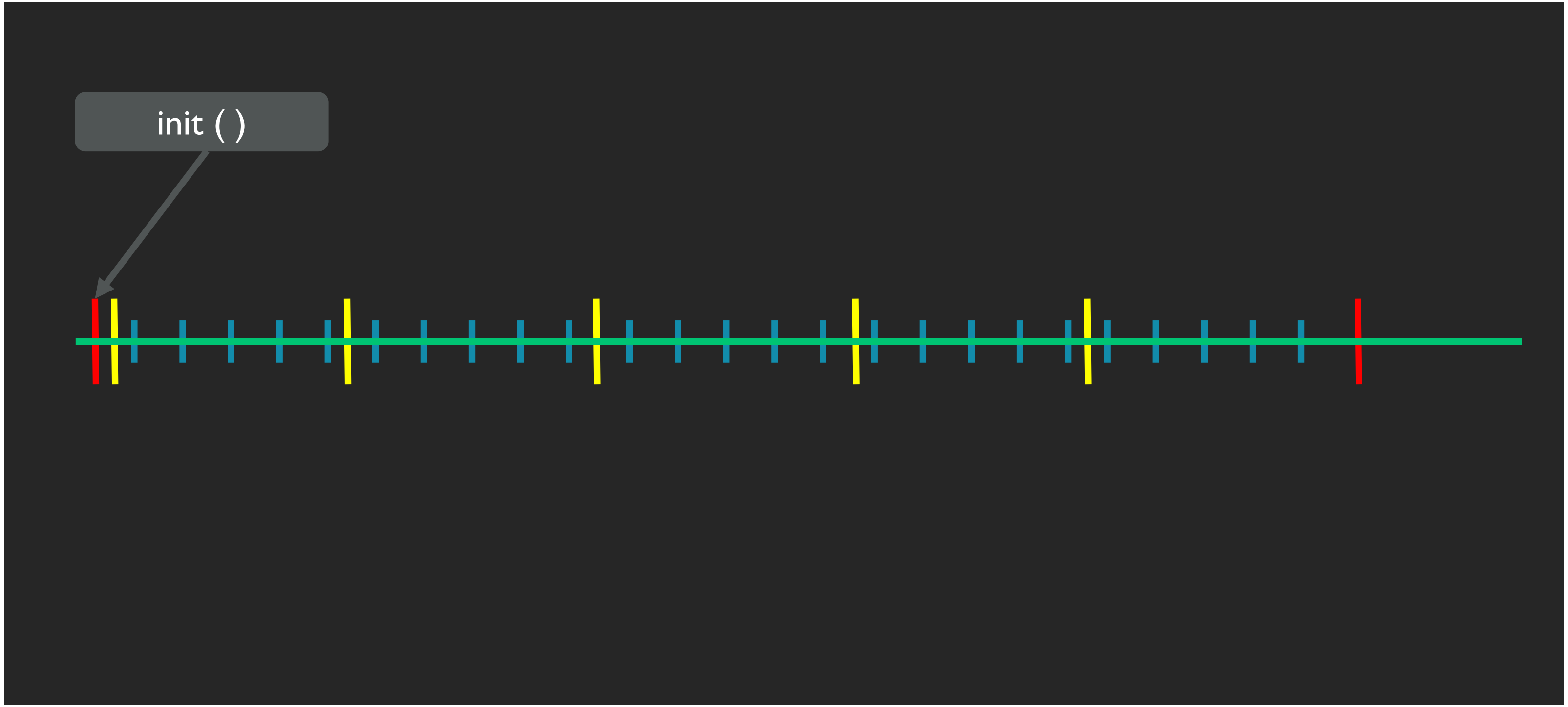
mbed::FlashIAP::get_sector_size

```
namespace mbed {

  class FlashIAP {
  public:

     /** Get the sector size at the defined address
      *
      *  Sector size might differ at address ranges.
      *  An example <0-0x1000, sector size=1024; 0x10000-0x20000, size=2048>
      *
      *  @param addr Address of or inside the sector to query
      *  @return Size of a sector in bytes or MBED_FLASH_INVALID_SIZE if not mapped
      */
      uint32_t get_sector_size(uint32_t addr) const;
  };
}
```

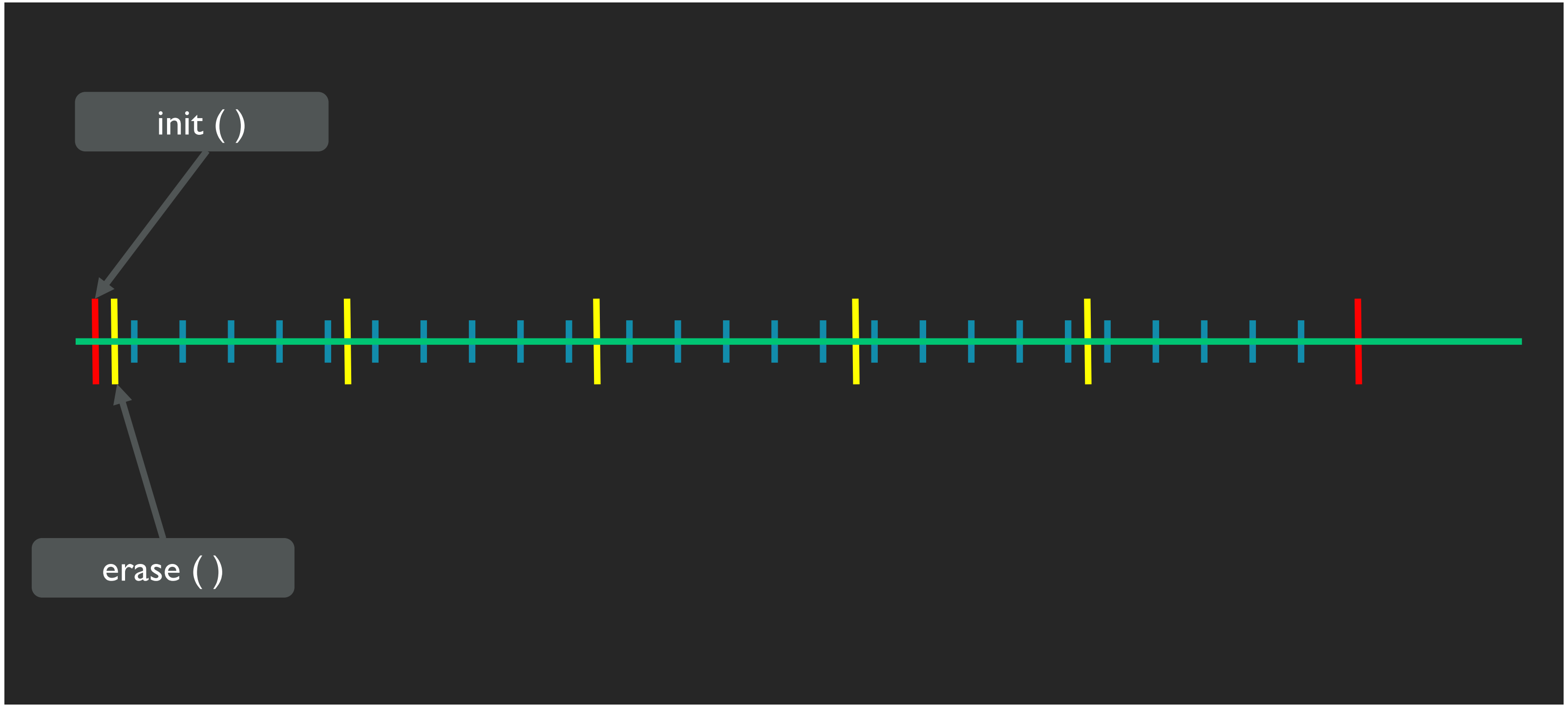
mbed::FlashIAP::accessors

```
namespace mbed {

  class FlashIAP {
  public:

     /** Get the flash start address
      *  @return Flash start address
      */
      uint32_t get_flash_start() const;

     /** Get the flash size
      *  @return Flash size
      */
      uint32_t get_flash_size() const;

     /** Get the program page size
      *  @return Size of a program page in bytes
      */
      uint32_t get_page_size() const;
  };
}
```

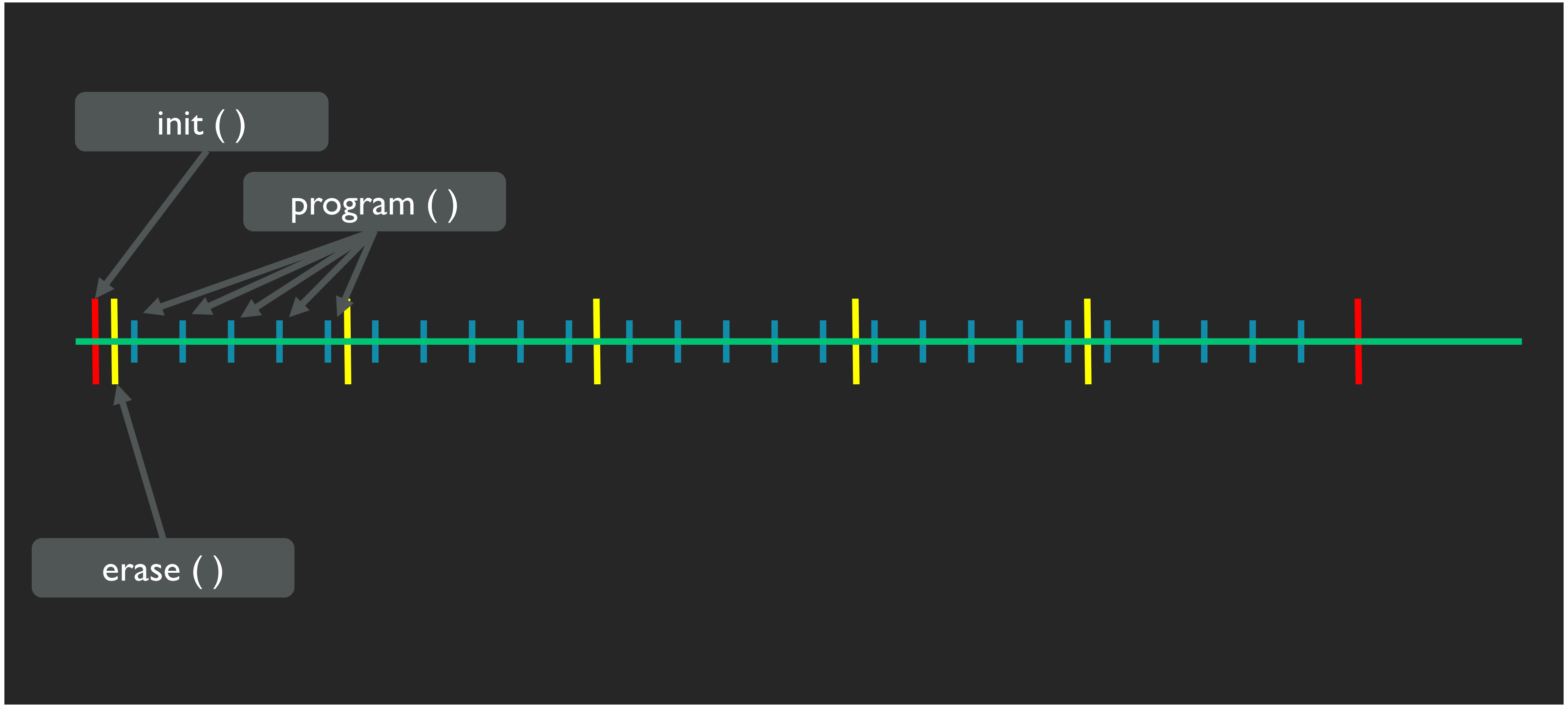
FlashIAP example usage



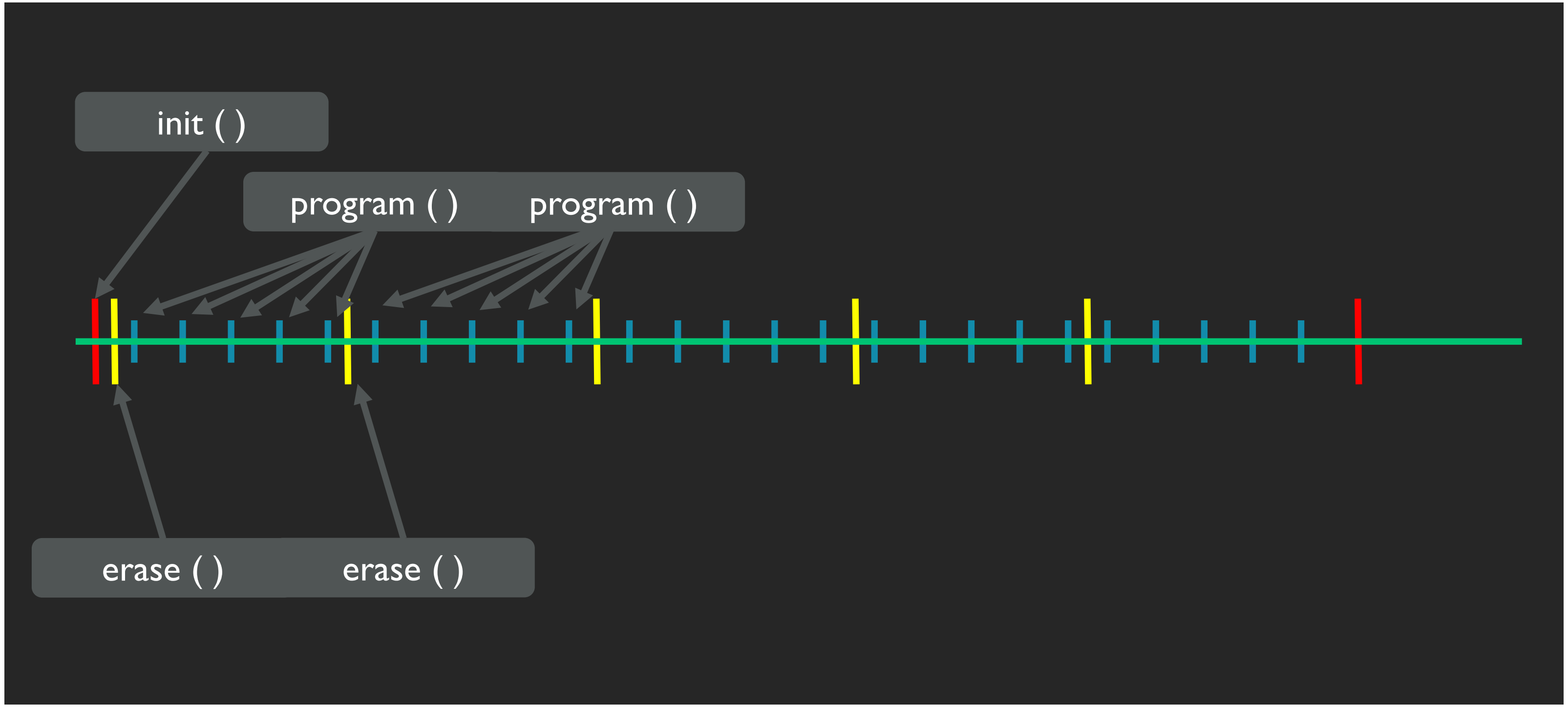
FlashIAP example usage



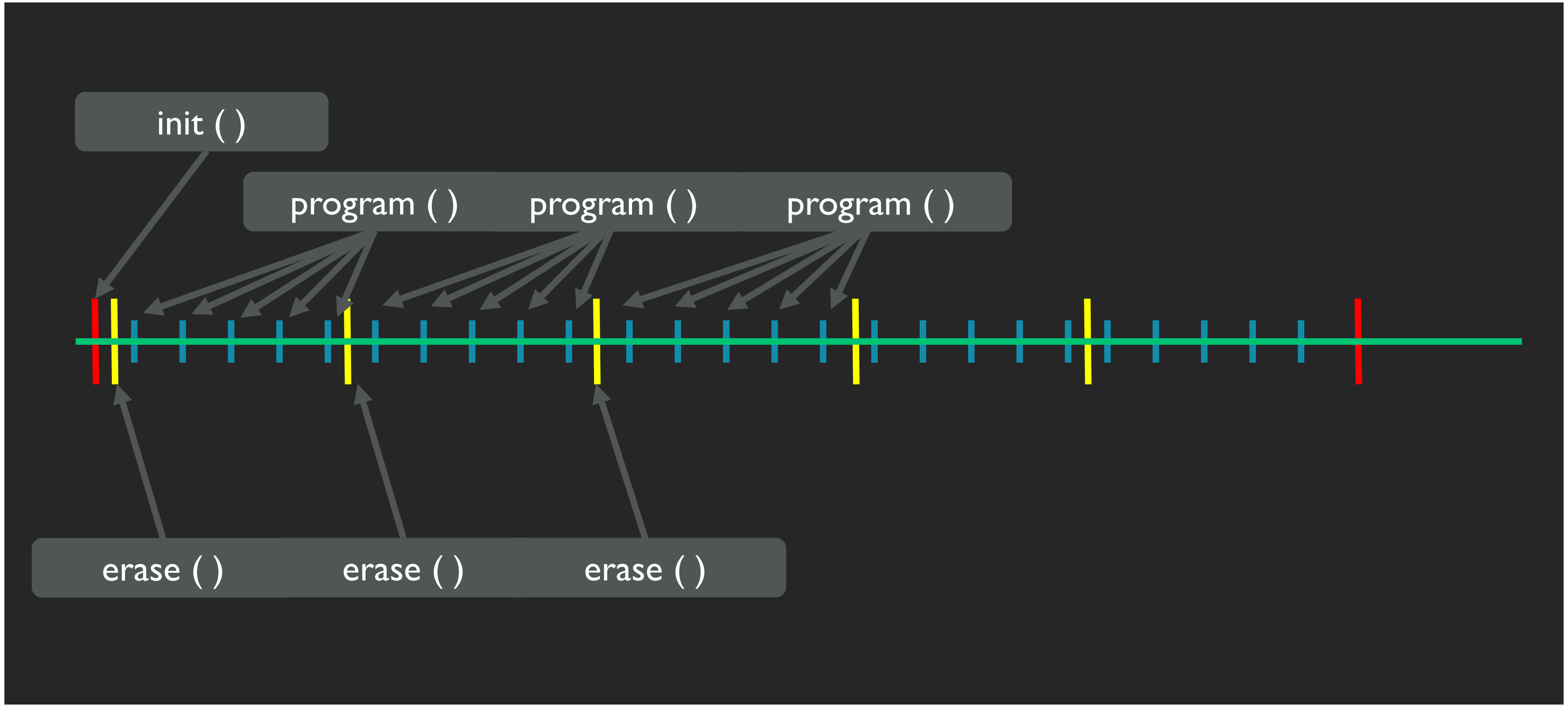
FlashIAP example usage



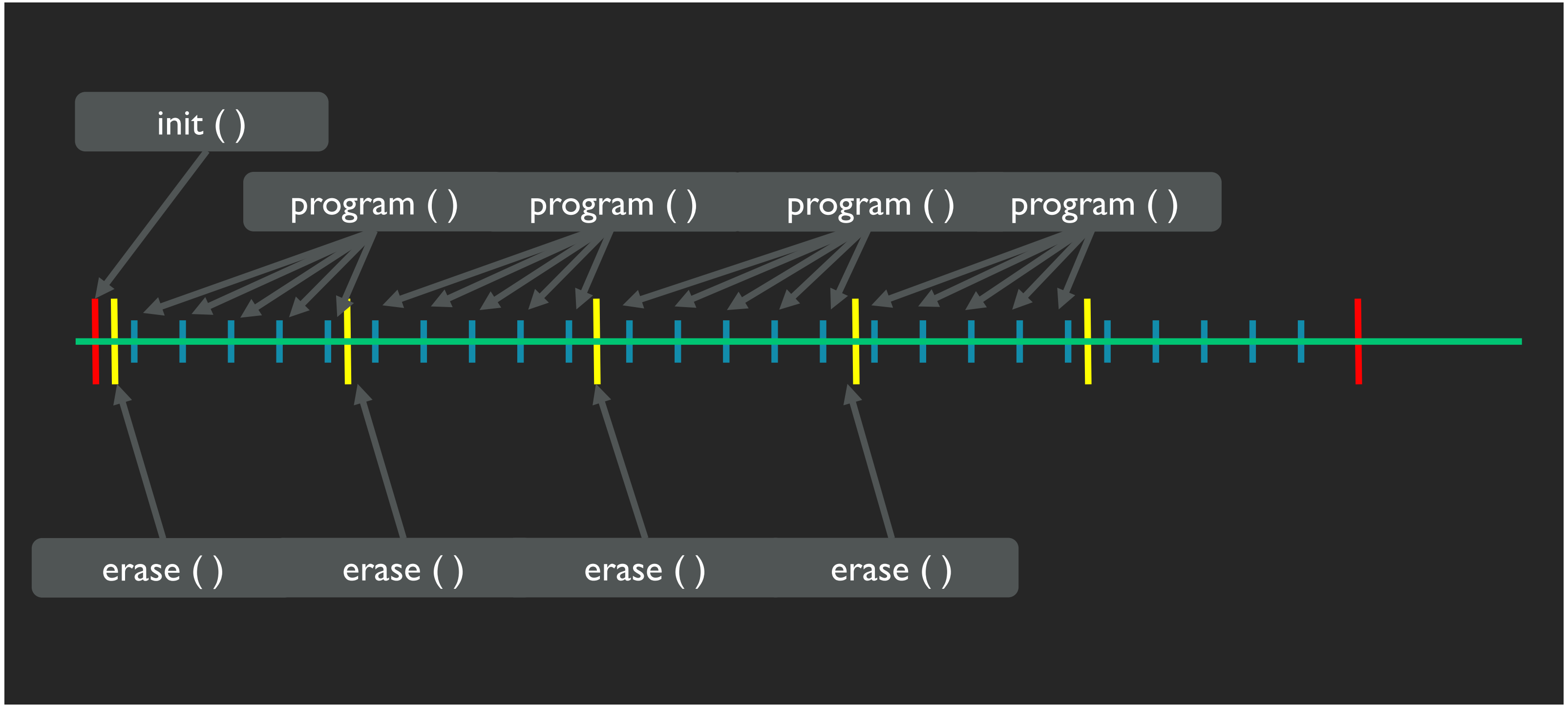
FlashIAP example usage



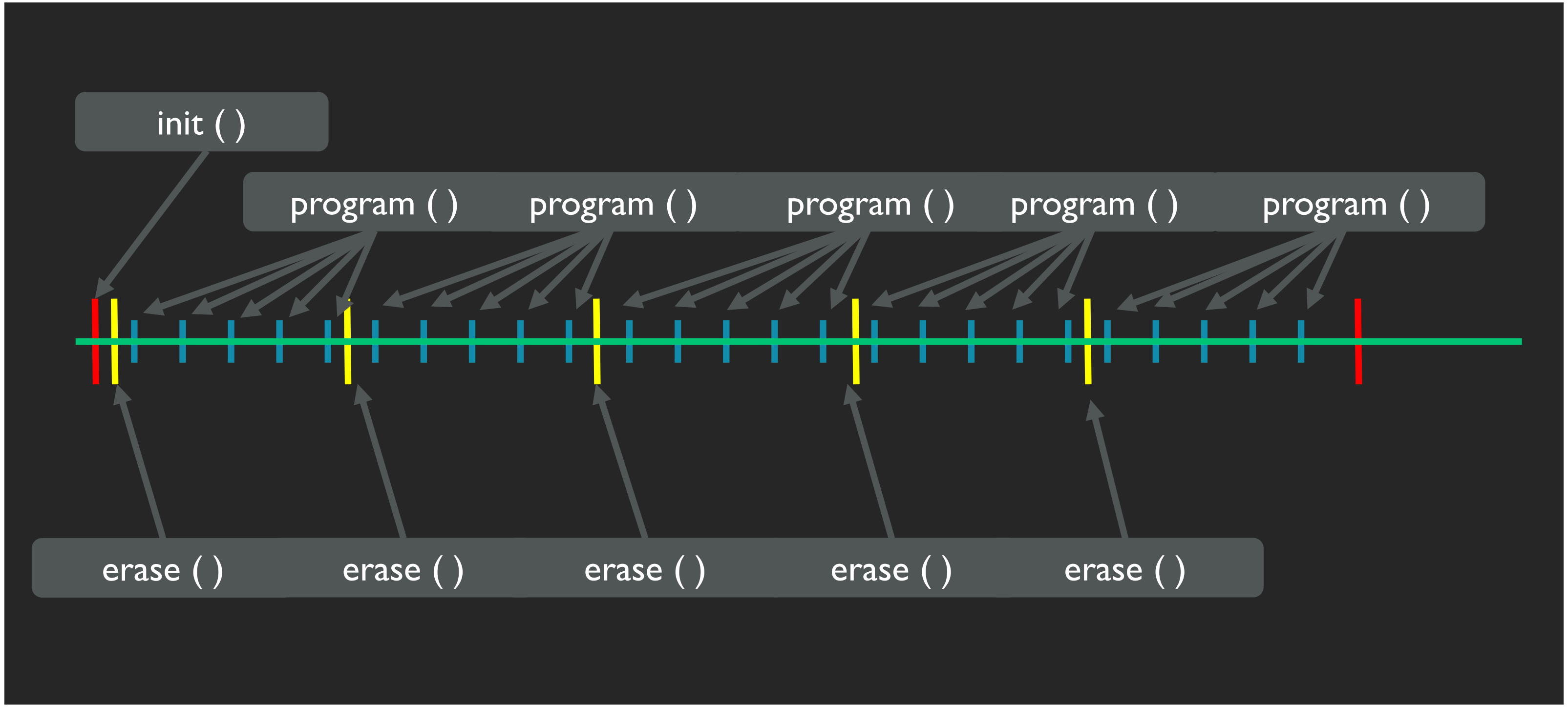
FlashIAP example usage



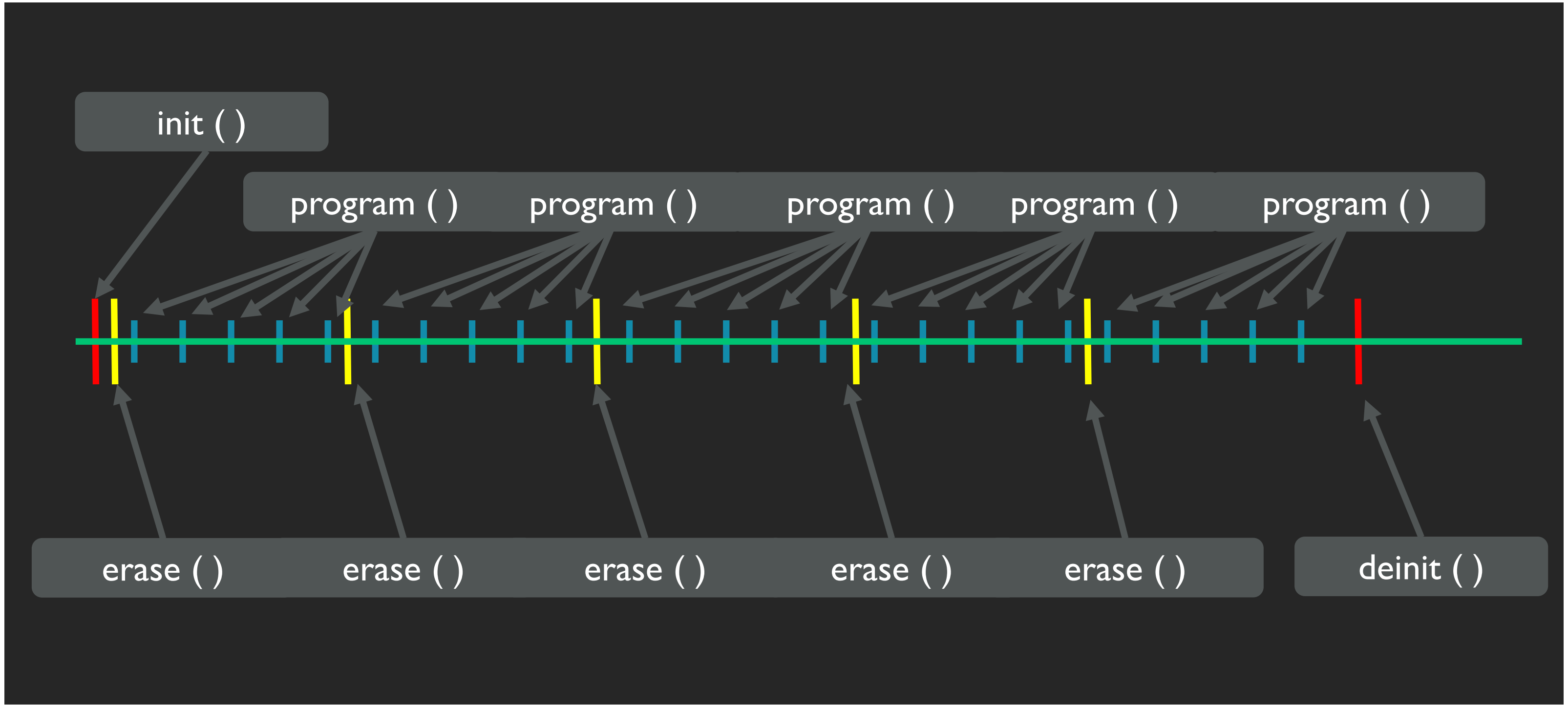
FlashIAP example usage



FlashIAP example usage



FlashIAP example usage



Flash HAL

SDK implementation for flash HAL

FlashIAP	Flash HAL
FlashIAP::init	flash_init
FlashIAP::deinit	flash_free
FlashIAP::read	n/a
FlashIAP::program	flash_program_page
FlashIAP::erase	flash_erase_sector
FlashIAP::get_sector_size	flash_get_sector_size
FlashIAP::get_flash_start	flash_get_start_address
FlashIAP::get_flash_size	flash_get_size
FlashIAP::get_page_size	flash_get_page_size

ARM

Enabling flash HAL

- Enable flash HAL by defining **FLASH** in the device_has list
  - Intended for SDK based implementation with flash peripheral drivers
  - Implements the previously discussed flash HAL API
  - Be considerate of gotchas from FlashIAP section

```
{
    "Target": {
        "core": "Cortex-M4F",
        "supported_toolchains": ["ARM", "GCC_ARM", "IAR"],
        "extra_labels": [],
        "device_has": ["ANALOGIN", "ANALOGOUT", "TRNG", "FLASH"]
    }
}
```

  - https://docs.mbed.com/docs/mbed-os-handbook/en/latest/advanced/flash

A FLASH_CMSIS_ALGO implementation

- Where do CMSIS flash algos come from?

- What is extracted?
  - pic code for program_page, erase_sector, get_sector_size
  - Converted into byte array and stored in RAM

- Protected operation due to mixing pic and non-pic code

Enable FLASH_CMSIS_ALGO

- Uses cmsis-packs to create a binary blob in a c file
- Common CMSIS Flash Algo HAL implementation
  - located at **hal/TARGET_FLASH_CMSIS_ALGO**

```
$ cd tools/flash_algo

$ python extract.py

$ mv ./output/<device_name>_0.c ./output/flash_api.c

$ mv ./output/flash_api.c ../../../targets/<PATH_TO_YOUR_TARGET_HAL>
```

- Add **FLASH_CMSIS_ALGO** in extra_labels
- And enable flash HAL by defining **FLASH** in the device_has list

```
{
    "Target": {
        "supported_toolchains": ["ARM", "GCC_ARM", "IAR"],
        "extra_labels": ["FLASH_CMSIS_ALGO"],
        "device_has": ["ANALOGIN", "ANALOGOUT", "TRNG", "FLASH"]
    }
}
```

Linker scripts update

- The start of device flash, MBED_APP_START
- The size of device flash, MBED_APP_SIZE

```
#! armcc -E

#if !defined(MBED_APP_START)
  #define MBED_APP_START 0x08000000
#endif

#if !defined(MBED_APP_SIZE)
  #define MBED_APP_SIZE 0x200000
#endif

LR_IROM1 MBED_APP_START MBED_APP_SIZE  {    ; load region size_region

  ER_IROM1 MBED_APP_START MBED_APP_SIZE  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
  }

  ; Total: 256 vectors = 1024 bytes (0x400) to be used
  RW_IRAM1 (0x20000000 + (256*4)) (0x30000 - (256*4))  {  ; RW data
   .ANY (+RW +ZI)
  }
}
```

And enable the bootloader_supported flag

- Enable bootloader for a target in the targets.json file
  - "**bootloader_supported**": true

```
{
    "Target": {
        "supported_toolchains": null,
        "extra_labels": [],
        "macros": [],
        "device_has": [],
        "features": [],
        "bootloader_supported": true
    },
}
```

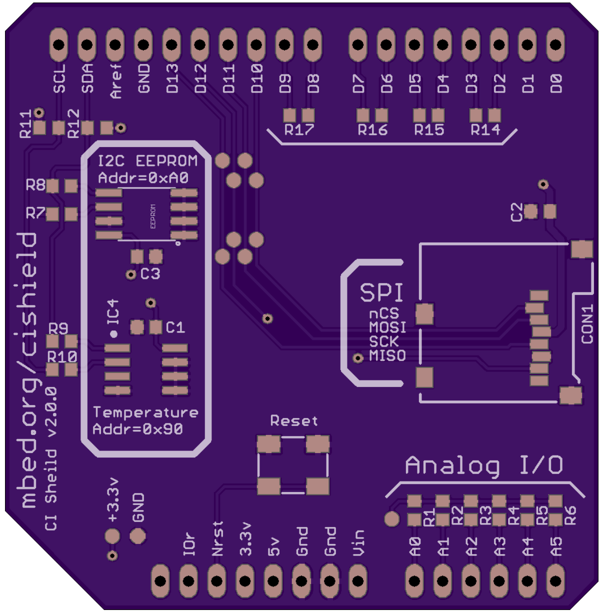
Workshop

ARM

Agenda

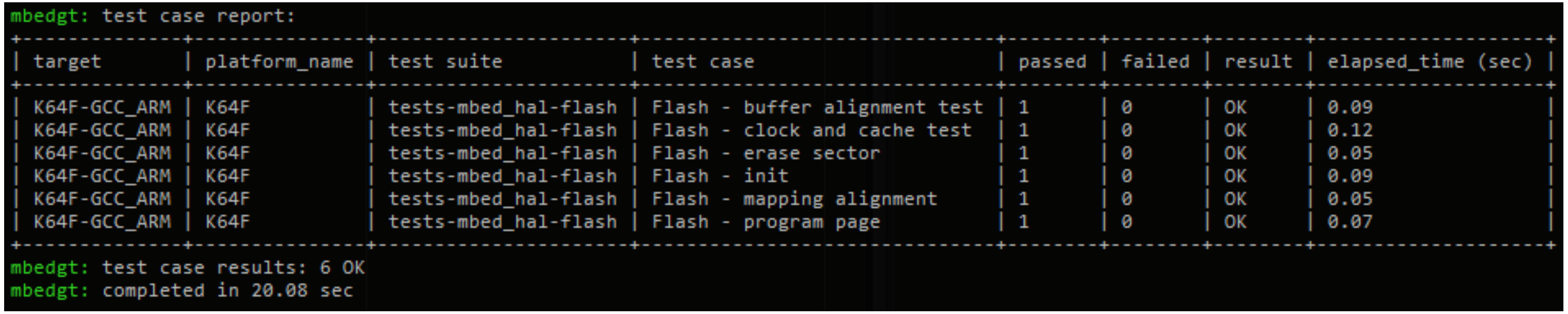
- Requirements:
  - CI Test Shield

    or
  - SD card on dev board
- Using k64f as a showcase platform
  - Run flash HAL and FlashIAP tests
  - Create a bootloader
  - Build blinky example for loading with the bootloader



Flash HAL tests

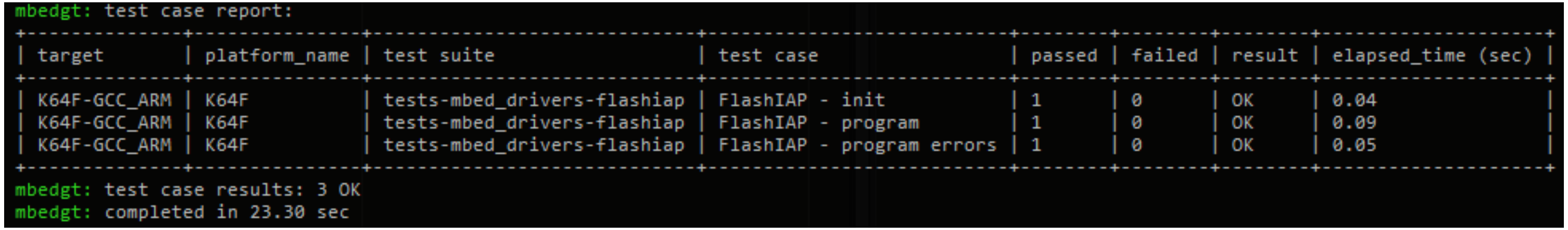
- Flash HAL
  - unit tests: tests-mbed_hal-flash
- Flash HAL tests:
  - Implemented in mbed-os/TESTS/mbed_hal/flash/functional_tests/main.cpp
  - mbed test -n tests-mbed_hal-flash -m K64F -t GCC_ARM



```
mbedgt: test case report:
+---------------+---------------+---------------------+------------------------------------+---------+---------+---------+----------------------+
| target        | platform_name | test suite          | test case                          | passed  | failed  | result  | elapsed_time (sec) |
+---------------+---------------+---------------------+------------------------------------+---------+---------+---------+----------------------+
| K64F-GCC_ARM | K64F          | tests-mbed_hal-flash | Flash - buffer alignment test      | 1       | 0       | OK      | 0.09               |
| K64F-GCC_ARM | K64F          | tests-mbed_hal-flash | Flash - clock and cache test       | 1       | 0       | OK      | 0.12               |
| K64F-GCC_ARM | K64F          | tests-mbed_hal-flash | Flash - erase sector               | 1       | 0       | OK      | 0.05               |
| K64F-GCC_ARM | K64F          | tests-mbed_hal-flash | Flash - init                       | 1       | 0       | OK      | 0.09               |
| K64F-GCC_ARM | K64F          | tests-mbed_hal-flash | Flash - mapping alignment          | 1       | 0       | OK      | 0.05               |
| K64F-GCC_ARM | K64F          | tests-mbed_hal-flash | Flash - program page               | 1       | 0       | OK      | 0.07               |
+---------------+---------------+---------------------+------------------------------------+---------+---------+---------+----------------------+
mbedgt: test case results: 6 OK
mbedgt: completed in 20.08 sec
```

FlashIAP tests

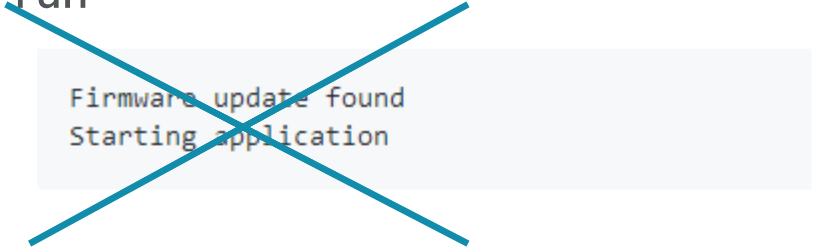
- FlashIAP
  - unit tests: tests-mbed_drivers-flashiap
- FlashIAP tests:
  - Implemented in mbed-os/TESTS/mbed_drivers/flashiap/main.cpp
  - mbed test -n tests-mbed_drivers-flashiap -m K64F -t GCC_ARM



```
mbedgt: test case report:
+--------------+---------------+-----------------------------+----------------------------+--------+--------+--------+--------------------+
| target       | platform_name | test suite                  | test case                  | passed | failed | result | elapsed_time (sec) |
+--------------+---------------+-----------------------------+----------------------------+--------+--------+--------+--------------------+
| K64F-GCC_ARM | K64F          | tests-mbed_drivers-flashiap | FlashIAP - init            | 1      | 0      | OK     | 0.04               |
| K64F-GCC_ARM | K64F          | tests-mbed_drivers-flashiap | FlashIAP - program         | 1      | 0      | OK     | 0.09               |
| K64F-GCC_ARM | K64F          | tests-mbed_drivers-flashiap | FlashIAP - program errors  | 1      | 0      | OK     | 0.05               |
+--------------+---------------+-----------------------------+----------------------------+--------+--------+--------+--------------------+
mbedgt: test case results: 3 OK
mbedgt: completed in 23.30 sec
```

Create a bootloader

- https://github.com/ARMmbed/mbed-os-example-bootloader
- Build a bootloader image
  - Import the example: mbed import mbed-os-example-bootloader
  - Compile: mbed compile –m <target>
  - Restrict size determined by compile size and next sector boundary
  - Bootloader binary should be created at .\BUILD\<TARGET_NAME>\<TOOLCHAIN>\mbed-os-example-bootloader.bin
- Copy to board and run



```
Firmware update found
Starting application
```

Blinky built for a bootloader

- Link: https://github.com/ARMmbed/mbed-os-example-blinky
- We need to have a bootloader image for our target (from previous exercise)
  - /bootloader folder is a good storage place
  - Set the path in the mbed_app.json file

```
"target_overrides": {
    ...
    "<TARGET_NAME>": {
        "target.bootloader_img": "bootloader/<TARGET_NAME>.bin"
    },
    ...
```

  - Compile and run the example, LED should be blinking. The console should display:
  - Copy file to the sd card
  - Plug in sd card and run the example

```
Firmware update found
Starting application
```

Questions?

ARM

Thank you for your attention!