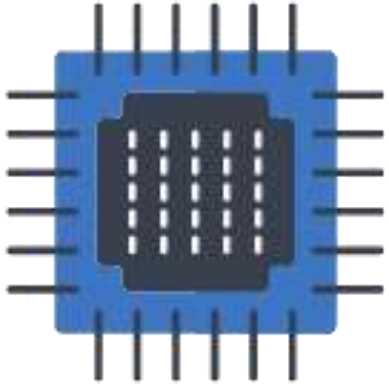


MICROPROCESSOR -LAB

Lab3: MBED Library

Summary

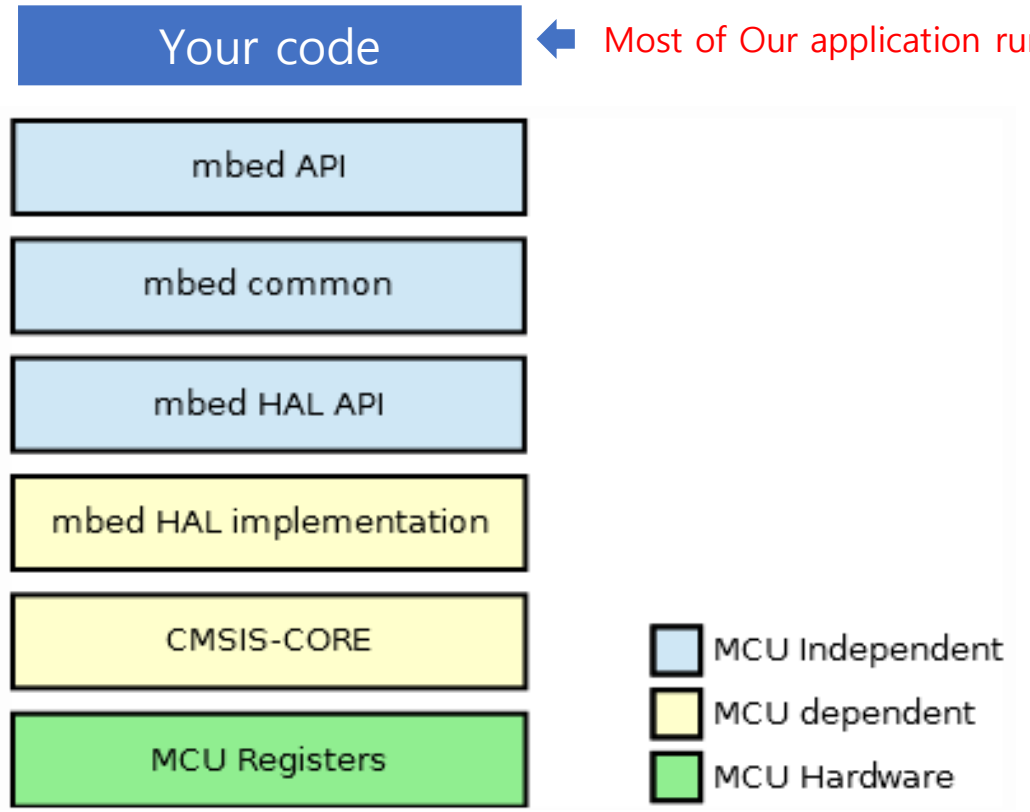
- Research Objectives
  - Learn how to use a MBED API.
  - Learn Why we use a MBED API instead of hardware dependent code.

- Methodology
  - Lab 1. LED ON/OFF using MBED API
  - Lab 2. LED ON/OFF using CMSIS-CORE

- Assignment
  - Lab 3. (Assignment) LED ON/OFF by Directly accessing the register

MBED Library Internals

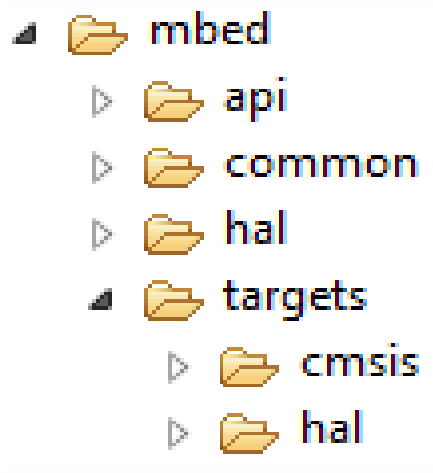
- MBED library and its usage
  - Porting library to a new processor
  - Adding new peripheral devices
  - Adding support for new toolchain
- MBED library Hierarchy



Library source and its directory tree

- MBED library source file
  - https://developer.mbed.org/users/mbed_official/code/mbed-src/

- Directory tree
  - Target Independent (Microprocessor independent code)
    - mbed/api: Headers that define the actual MBED lib API
    - mbed/common: MBED common source files
    - mbed/hal: HPL API implemented for all targets
  - Target dependent (Microprocessor dependent code)
    - mbed/targets/hal: Actual implementation of Hardware Abstraction Layer
    - mbed/targets/cmsis: CMSIS-CORE source files

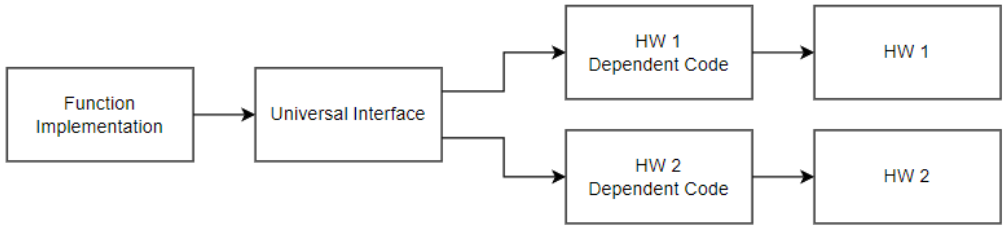


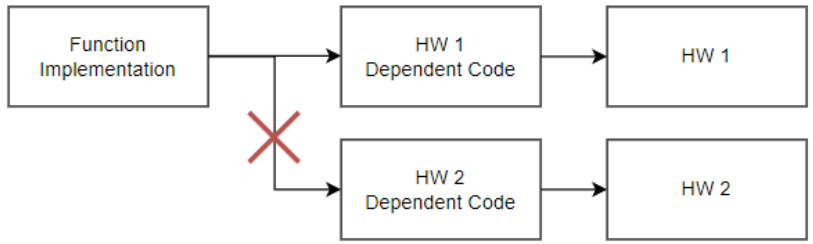
HAL: Hardware Abstraction Layer

- Hardware Abstraction Layer

- If you want to port some code that runs on current microprocessor(STM32F401RE) to another microprocessor.
  - If you use hardware dependent code, you must modify it.

- Instead, by using APIs, it's possible to write code that is not dependent on specific devices or hardware, making it easier to reuse the code.
  - Usually, Hardware provider gives you a standardized HAL implementation





MCU Register

- To control hardware, read/write operations are performed on a specific address

- For example, If you want a set GPIOA port0 to HIGH (1)
  - GPIOA base addr. (0x40020000) + GPIO port set/reset addr. Offset (0x18) = 0x40020018
  - **Write value 0x00000001 on address 0x40020018**

Table 1. STM32F401xB/C and STM32F401xD/E register boundary addresses

Boundary address	Peripheral	Bus	Register map
0x5000 0000 - 0x5003 FFFF	USB OTG FS	AHB2	*Section 22.16.6: OTG_FS register map on page 755*
0x4002 6400 - 0x4002 67FF	DMA2	AHB1	*Section 9.5.11: DMA register map on page 198*
0x4002 6000 - 0x4002 63FF	DMA1		
0x4002 3C00 - 0x4002 3FFF	Flash interface register		*Section 3.8: Flash interface registers on page 60*
0x4002 3800 - 0x4002 3BFF	RCC		*Section 6.3.22: RCC register map on page 137*
0x4002 3000 - 0x4002 33FF	CRC		*Section 4.4.4: CRC register map on page 70*
0x4002 1C00 - 0x4002 1FFF	GPIOH		*Section 8.4.11: GPIO register map on page 164*
0x4002 1000 - 0x4002 13FF	GPIOE		
0x4002 0C00 - 0x4002 0FFF	GPIOD		
0x4002 0800 - 0x4002 0BFF	GPIOC		
0x4002 0400 - 0x4002 07FF	GPIOB		
0x4002 0000 - 0x4002 03FF	GPIOA		

8.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..E and H)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x reset bit y (y = 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Resets the corresponding ODRx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BSy:** Port x set bit y (y= 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Sets the corresponding ODRx bit

CMSIS-CORE

- Provides data structures for accessing low-level registers
  - Hardware dependent code

```
typedef struct
{
  __IO uint32_t MODER;    /*!< GPIO port mode register,               Address offset: 0x00      */
  __IO uint32_t OTYPER;   /*!< GPIO port output type register,        Address offset: 0x04      */
  __IO uint32_t OSPEEDR;  /*!< GPIO port output speed register,       Address offset: 0x08      */
  __IO uint32_t PUPDR;    /*!< GPIO port pull-up/pull-down register,  Address offset: 0x0C      */
  __IO uint32_t IDR;      /*!< GPIO port input data register,         Address offset: 0x10      */
  __IO uint32_t ODR;      /*!< GPIO port output data register,        Address offset: 0x14      */
  __IO uint16_t BSRRL;    /*!< GPIO port bit set/reset low register,  Address offset: 0x18      */
  __IO uint16_t BSRRH;    /*!< GPIO port bit set/reset high register, Address offset: 0x1A      */
  __IO uint32_t LCKR;     /*!< GPIO port configuration lock register, Address offset: 0x1C      */
  __IO uint32_t AFR[2];   /*!< GPIO alternate function registers,     Address offset: 0x20-0x24 */
} GPIO_TypeDef;

#define PERIPH_BASE           ((uint32_t)0x40000000)
#define APB1PERIPH_BASE       PERIPH_BASE
#define AHB1PERIPH_BASE       (PERIPH_BASE + 0x00020000)

/*!< AHB1 peripherals */
#define GPIOA_BASE            (AHB1PERIPH_BASE + 0x0000)
#define GPIOB_BASE            (AHB1PERIPH_BASE + 0x0400)
#define GPIOC_BASE            (AHB1PERIPH_BASE + 0x0800)
#define GPIOD_BASE            (AHB1PERIPH_BASE + 0x0C00)
#define GPIOE_BASE            (AHB1PERIPH_BASE + 0x1000)
#define GPIOF_BASE            (AHB1PERIPH_BASE + 0x1400)
#define GPIOG_BASE            (AHB1PERIPH_BASE + 0x1800)
#define GPIOH_BASE            (AHB1PERIPH_BASE + 0x1C00)
#define GPIOI_BASE            (AHB1PERIPH_BASE + 0x2000)

```

MBED API

- Provide a user-friendly object-oriented API.
- Define intuitive basic operators for primitive data types and assignment statements.
- Used by most programs developed on the MBED platform.

```
1  class DigitalInOut {
2
3  public:
4      DigitalInOut(PinName pin) : gpio() {
5          gpio_init_in(&gpio, pin);
6      }
7
8      DigitalInOut(PinName pin, PinDirection direction, PinMode mode, int value) : gpio() {
9          gpio_init_inout(&gpio, pin, direction, mode, value);
10     }
11
12     void write(int value) {
13         gpio_write(&gpio, value);
14     }
15
16     int read() {
17         return gpio_read(&gpio);
18     }
19
20     void output() {
21         gpio_dir(&gpio, PIN_OUTPUT);
22     }
23
24     void input() {
25         gpio_dir(&gpio, PIN_INPUT);
26     }
27
28     void mode(PinMode pull) {
```

MBED API - DigitalOut

```
#include <DigitalOut.h>
```

Public Member Functions

	DigitalOut (PinName pin) Create a **DigitalOut** connected to the specified pin. More...
	DigitalOut (PinName pin, int value) Create a **DigitalOut** connected to the specified pin. More...
void	**write** (int value) Set the output, specified as 0 or 1 (int) More...
int	**read** () Return the output setting, represented as 0 or 1 (int) More...
int	**is_connected** () Return the output setting, represented as 0 or 1 (int) More...
DigitalOut &	**operator=** (int value) A shorthand for **write()** More...
DigitalOut &	**operator=** (**DigitalOut** &rhs) A shorthand for **write()** using the assignment operator which copies the state from the **DigitalOut** argument. More...
	operator int () A shorthand for **read()** More...

- DigitalOut(PinName Pin, int value)
- Write
  - Sets output value (int)
- Read
  - Returns read value (int)
- Is_connected()
  - Returns connection status (int)
- DigitalOut&
  - Operator overloading

Lab 1. LED ON/OFF using MBED API

```
#include "mbed.h"

DigitalOut led1(LED1);

int main(){
    while (true){
        led1 = 1; // same as led1.write(1);
        wait(0.5);

        led1 = 0; // same as led1.write(0);
        wait(0.5);
    }
}
```

- DigitalOut led1(LED1);
  - LED1 is a constant defined in PinNames.h.
  - Using the DigitalOut class, an instance led1 is created
  - During instantiation, constructor initializes the ouput pin as a GPIO.
- led1 = 1;
  - This turns the LED on.
  - led1.write() function is overloaded with the assignment operator to handle the setting of the pin value.
- wait(0.5);
  - This makes the processor wait for 0.5 seconds, effectively doing nothing (NOP) during this period.

Lab 2. LED ON/OFF using CMSIS-CORE

```c
#include "mbed.h"
DigitalOut led1(LED1); //PA_5 = 0x5; - PinNames.h

int main(){
    unsigned int pin = 0x20;

    while(1){
        GPIOA->BSRR = pin;
        wait(0.5);

        GPIOA->BSRR = pin << 16;
        wait(0.5);
    }
}
```

- CMSIS-CORE contains Register information
  - Defined as c struct
- Code explain
  - PA_05 = 5th pin = 0x20 (0b10_0000)
  - BSRR: 32bit set/reset register
    - Low 16 bit = set corresponding pin
    - High 16 bit = reset corresponding pin

```c
typedef struct
{
  __IO uint32_t MODER;    /*!< GPIO port mode register,               Address offset: 0x00      */
  __IO uint32_t OTYPER;   /*!< GPIO port output type register,        Address offset: 0x04      */
  __IO uint32_t OSPEEDR;  /*!< GPIO port output speed register,       Address offset: 0x08      */
  __IO uint32_t PUPDR;    /*!< GPIO port pull-up/pull-down register,  Address offset: 0x0C      */
  __IO uint32_t IDR;      /*!< GPIO port input data register,         Address offset: 0x10      */
  __IO uint32_t ODR;      /*!< GPIO port output data register,        Address offset: 0x14      */
  __IO uint32_t BSRR;     /*!< GPIO port bit set/reset register,      Address offset: 0x18      */
  __IO uint32_t LCKR;     /*!< GPIO port configuration lock register, Address offset: 0x1C      */
  __IO uint32_t AFR[2];   /*!< GPIO alternate function registers,     Address offset: 0x20-0x24 */
} GPIO_TypeDef;
```

8.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..E and H)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x reset bit y (y = 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Resets the corresponding ODRx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BSy:** Port x set bit y (y= 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Sets the corresponding ODRx bit

Lab 3. (Assignment) LED ON/OFF by Directly accessing the register

- Fill in the blanks to ensure the code works properly
  - 1. Pin
  - 2. GPIOA_BSRR address
  - 3. shift amount

```
#include "mbed.h"
DigitalOut led1(LED1); //PA_5 = 0x05; - PinNames.h

int main(){
    unsigned int pin = ____;

    volatile unsigned int *portA = (unsigned int *) ____________; //GPIOA_BSRR


    while(1){
        *portA = pin;
        wait(0.5);

        *portA = pin << ____;
        wait(0.5);
    }
}
```

Useful Links

- ARM MBED - DigitalOut class Reference
  - https://os.mbed.com/docs/mbed-os/v6.16/apis/digitalout.html

- STM32F401xD/E advanced Arm-based 32bit MCUs – Reference Manual
  (Automatic download on click)
  - https://www.st.com/resource/en/reference_manual/rm0368-stm32f401xbc-and-stm32f401xde-advanced-armbased-32bit-mcus-stmicroelectronics.pdf