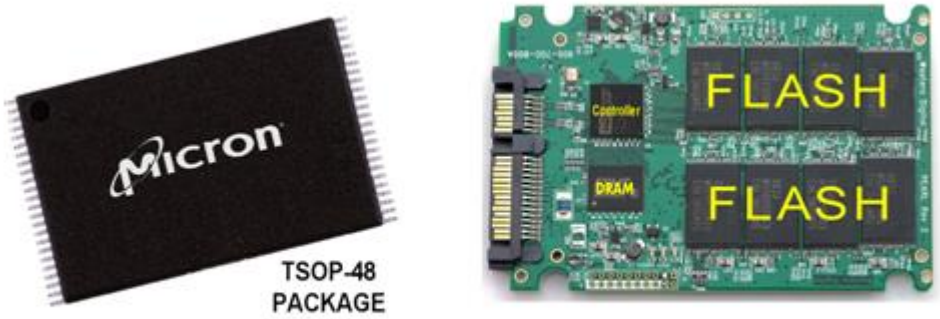
4.EEPROM (Non-volatile memory)

DR.SOMSIN THONGKRAIRAT



NAND flash is available in TSOP-48, WSOP-48, LGA-52, and BGA-63 packages

EEPROM

ROM -> read-only memory

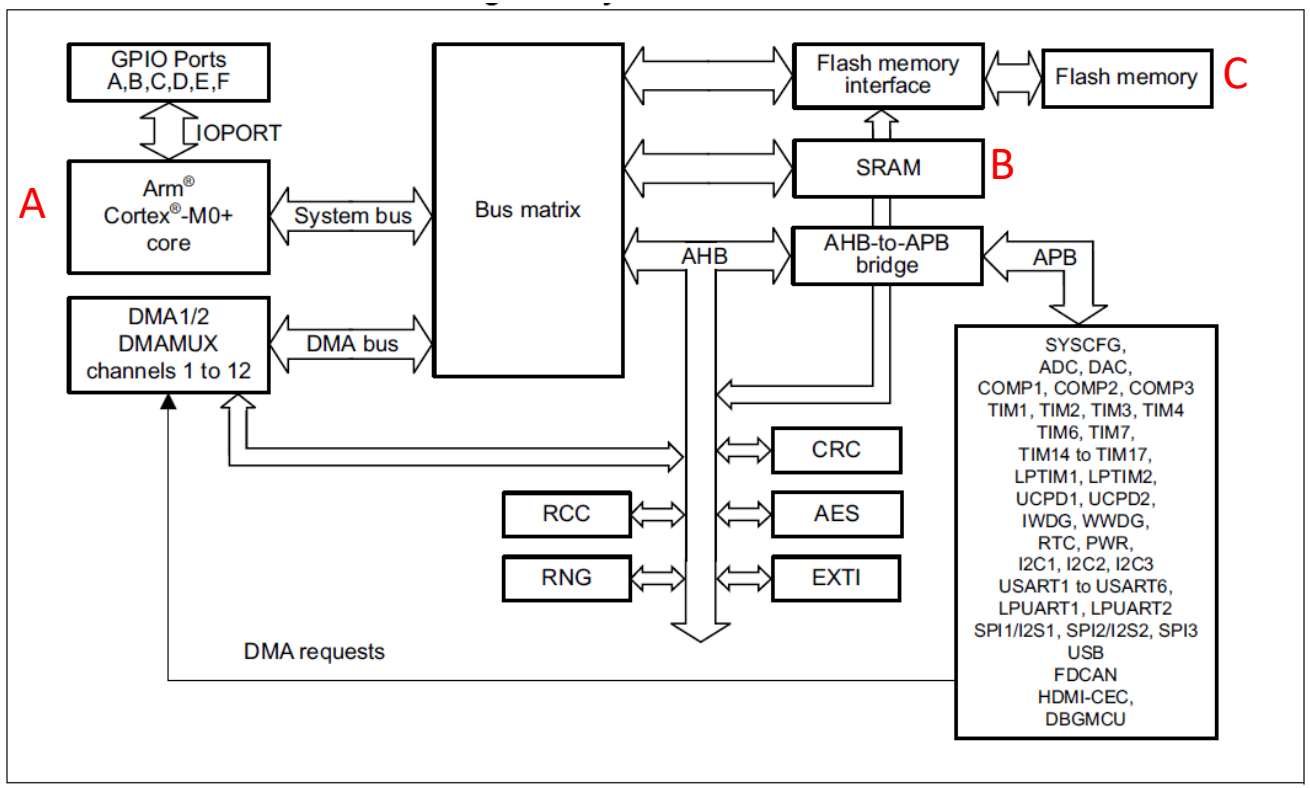
PROM-> programmable read-only memory

EPROM-> erasable programmable read-only memory

EEPROM-> electrically erasable programmable read-only memory

Where are code stored in MCU

Yes memory , Which one?



A. CPU

B. RAM

C. FLASH

D. All above

*From rm0444 page 56

MEMORY

Register (in CPU) -> define peripheral behavior and process

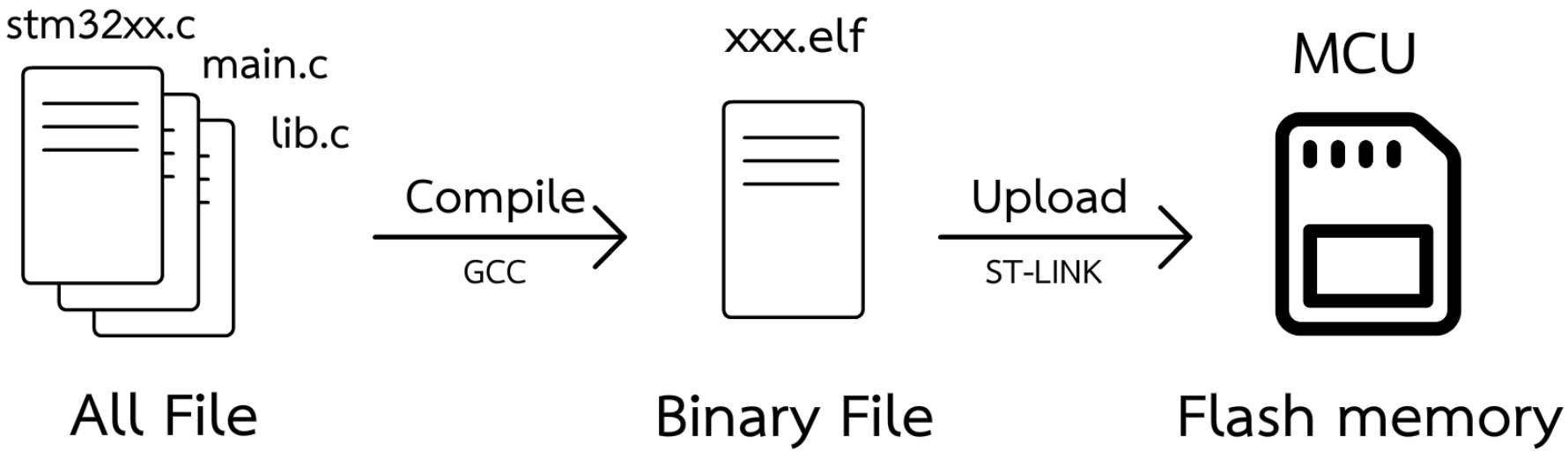
RAM -> store variable and HEAP

FLASH -> store program (code) and <u>persistence memory</u>

What is persistence memory

- อะไรก็ตามที่คงสถานะ (state) ไว้ได้ถึงแม้ว่าจะไม่มีแหล่งจ่ายไฟ (aka. non-volatile memory , ROM)

Uploading code



That why our program are persistence (we can plug off and plug in USB without reuploading code)

เป็นเหตุผลว่าทำไมโปรแกรมเรายังอยู่ใน MCU ในเมื่อเราหยุดจ่ายไฟให้ MCU

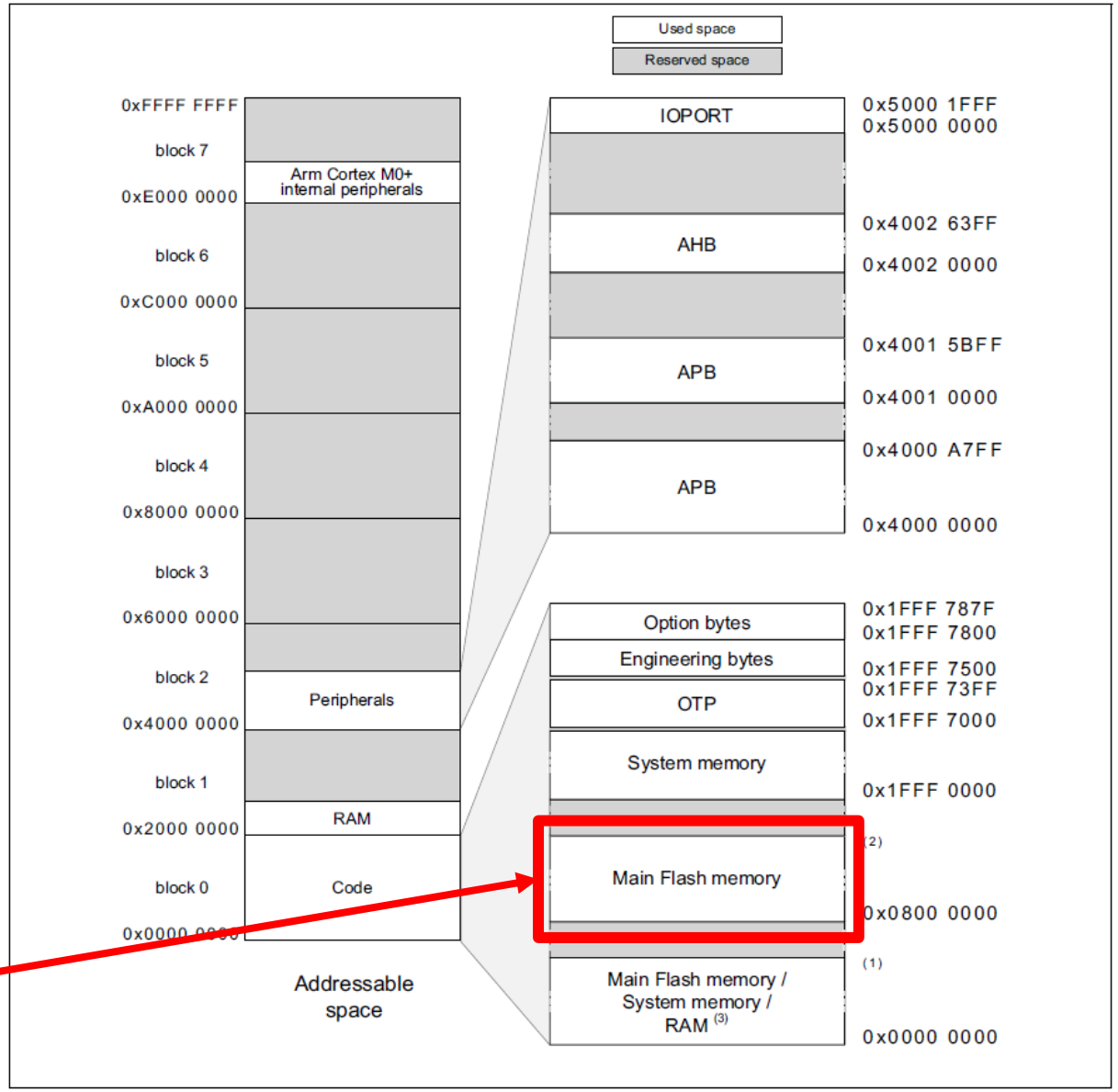
STM32G071RB ACTIVE

Save to MyST

Mainstream Arm Cortex-M0+ MCU with 128 Kbytes of Flash memory, 36 Kbytes RAM, 64 MHz CPU, 4x USART, timers, ADC, DAC, comm. I/F, 1.7-3.6V

Type	Boundary address	Size	Memory Area	Register description
SRAM	0x2000 9000 - 0x3FFF FFFF	~512 MB	Reserved	-
	0x2000 0000 - 0x2000 8FFF	36 KB	SRAM	*Section 2.3 on page 65*
Code	0x1FFF 7880- 0x1FFF FFFF	~34 KB	Reserved	-
	0x1FFF 7800 - 0x1FFF 787F	128 B	Option bytes	*Section 3.4 on page 81*
	0x1FFF 7500 - 0x1FFF 77FF	768 B	Engineering bytes	-
	0x1FFF 7400- 0x1FFF 74FF	256 B	Reserved	-
	0x1FFF 7000 - 0x1FFF 73FF	1 KB	OTP	-
	0x1FFF 0000 - 0x1FFF 6FFF	28 KB	System memory	-
	0x0802 0000 - 0x1FFF D7FF	~384 MB	Reserved	-
	0x0800 0000 - 0x0801 FFFF	128 KB	Main Flash memory	*Section 3.3.1 on page 70*
	0x0002 0000 - 0x07FF FFFF	~8 MB	Reserved	-
	0x0000 0000 - 0x0001 FFFF	128 KB	Main Flash memory, system memory or SRAM depending on BOOT configuration	-

Our Code (in binary)



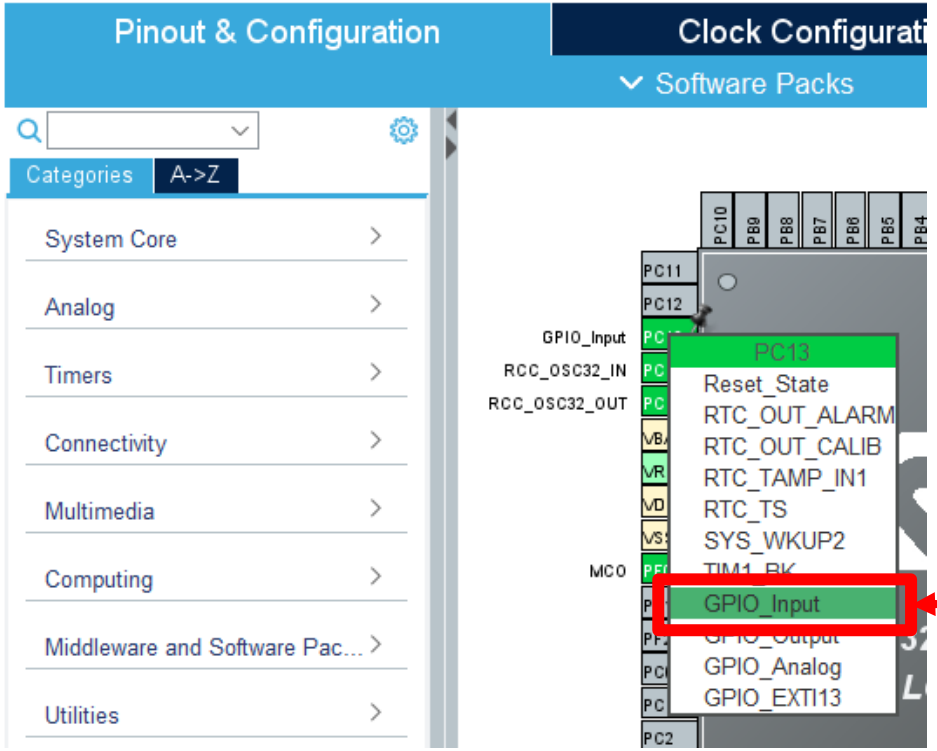
Compare RAM and FLASH(ROM)

comparation	RAM	FLASH
Speed	Super fast	fast
Persistence	NO	YES
Write Cycle	unlimited	Limited (10k – 1M cycle)
Price	More Expansive	Expansive

Size usually vary with price (want more pay more!)

Using RAM

- count number of released button and store in variable



Set PC13 to input

Coding

```c
/* USER CODE BEGIN 2 */
long but_release_count;
unsigned char string_buffer[50];
int string_buffer_size = -1;
int last_button_state = HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13);
int button_state;
/* USER CODE END 2 */


/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
button_state = HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13);
if(last_button_state == 0 && button_state == 1){ // release
but_release_count++;
string_buffer_size = sprintf(string_buffer,"haruhi %ld\r\n",but_release_count);
HAL_UART_Transmit(&huart2, string_buffer, string_buffer_size, 1000);
HAL_Delay (200); // debounce
}
last_button_state = button_state;
HAL_Delay (100);
/* USER CODE END WHILE */


/* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

```
haruhi 3
haruhi 4
haruhi 5
haruhi 6
haruhi 7
haruhi 8
haruhi 9
haruhi 10
haruhi 11
haruhi 0
haruhi 1
haruhi 2
haruhi 3
haruhi 0
haruhi 1
haruhi 2
haruhi 3
```

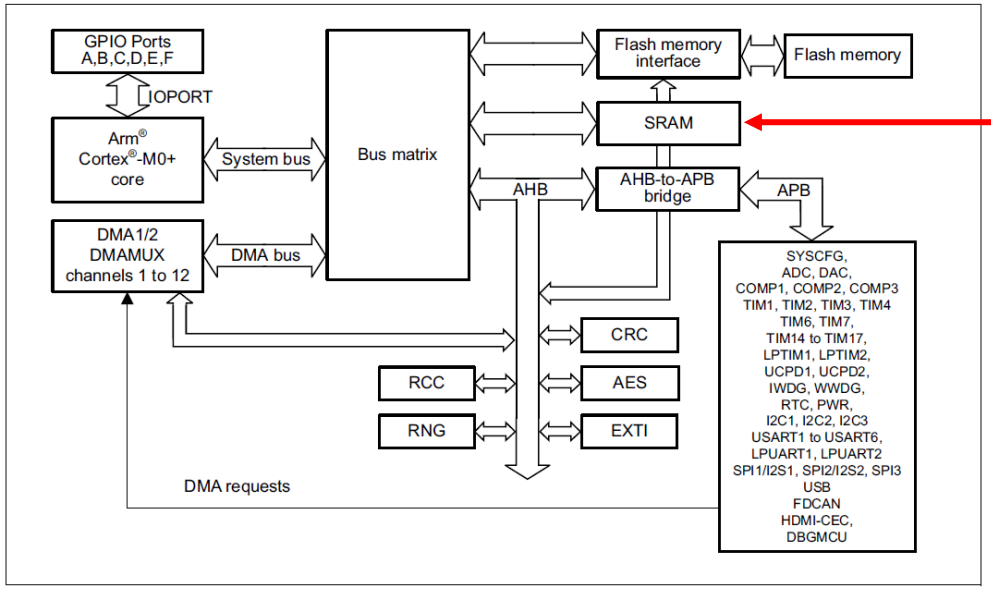
Reset

Reset

Why variable reset after MCU reset?

It on RAM, not persistence memory.

ตัวแปรอยู่บน RAM ไม่ได้อยู่บน persistence memory



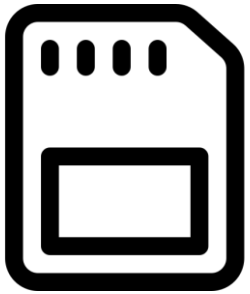
`but_release_count`

How to make it count continuedly (after Reset) ?

ทำอย่างไรให้ MCU นับต่อไป (หลังจาก Reset) ?

How to store data persistency

- Write data to file system (SD card , flash drive)
- Send data to network and load
- Use external memory (external FLASH)
- Use internal memory (FLASH)



EEPROM

electrically erasable programmable read-only memory

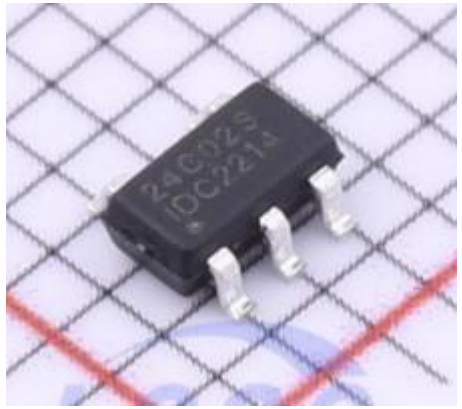
A type of Device or memory the provide non-volatile memory user can read and reprogram each byte on memory repeatedly.

คือ ประเภทของอุปกรณ์ หรือ หน่วยความจำ (memory) ที่สามารถอ่านและเปลี่ยนแปลงข้อมูลได้ เรื่อย ๆ โดยสามารถคงสถานะได้แม่จะไม่มีแหล่งจ่ายไฟ (non-volatile)

EEPROM chip

AD24C02 (2K bits[256*8])

Qty.	Unit Price	Ext. Price
10+	$0.0407	$ 0.41



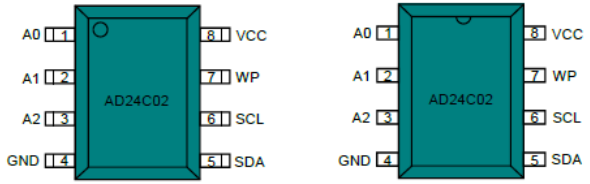


图2 AD24C02引脚定义图

产品特点

- 宽电压工作范围：1.8V～5.5V
- 存储器结构为：2K bits(256x8)
- 时钟频率为：1MHz(5V)；400kHz(1.8V,2.5V,2.7V)
- 自定时编程周期(最大5ms)
- 页写(8字节/页)
- 施密特触发器，抑制输入噪声
- 两线串行接口
- 硬件数据写保护
- 高可靠性：擦写次数——100万次
  保存时间——100年

Memory

Write cycle 1M cycle

Store time 100 year?

24LC01B/02B (2Kbits [256*8])

Quantity	Unit Price THB
1 - 9	18.11
10 - 49	14.92

MICROCHIP

- Page-write buffer for up to 8 bytes
- 2 ms typical write cycle time for page-write
- Hardware write protect for entire memory
- Can be operated as a serial ROM
- ESD protection > 3,000V
- 1,000,000 E/W cycles guaranteed
- Data retention > 200 years
- 8 pin DIP, SOIC, TSSOP* or SOT-23* package
- Available for temperature ranges
  - Commercial (C): 0°C to +70°C
  - Industrial (I): -40°C to +85°C

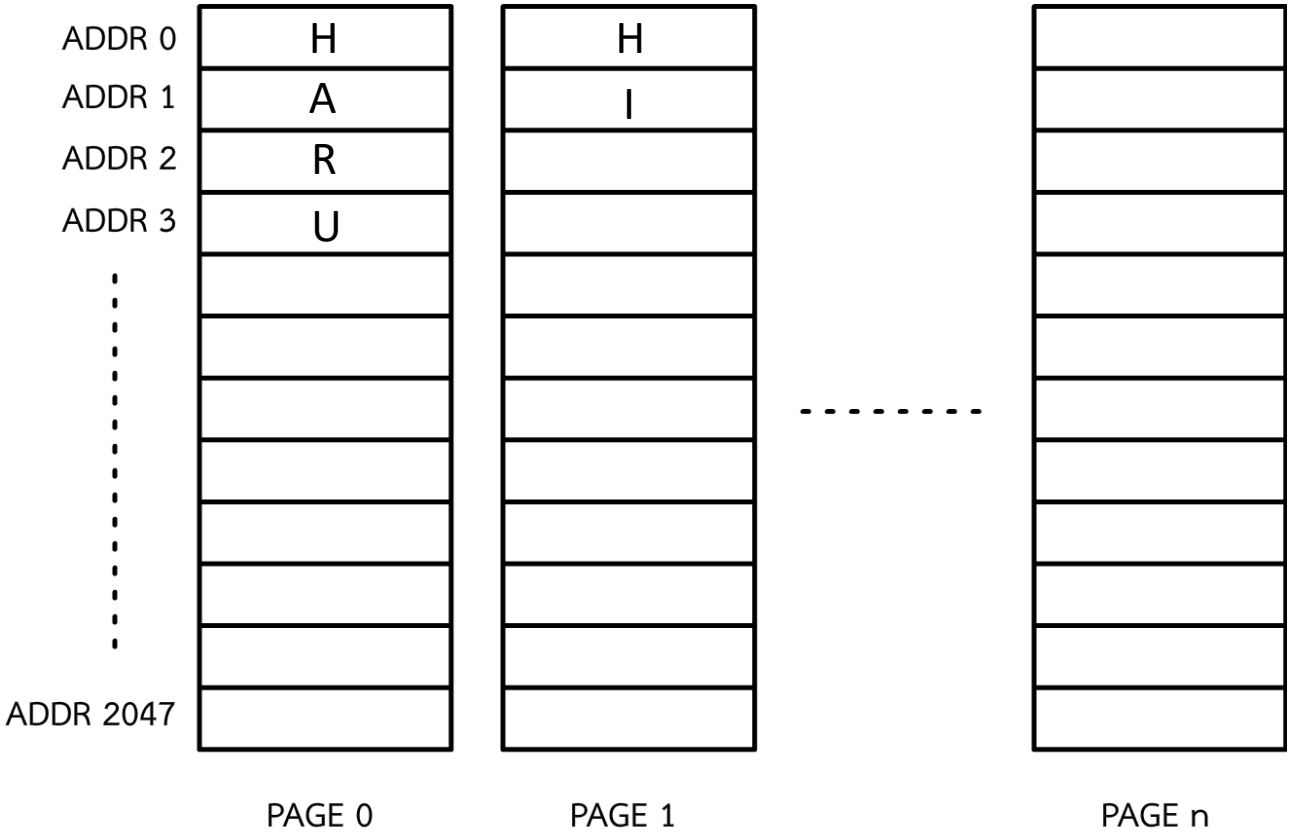
EEPROM Emulator

- using FLASH as EEPROM , ใช้พื้นที่บน FLASH memory แต่ใช้แบบ EEPROM

Area	Addresses	Size (bytes)	16 Kbyte devices	32 Kbyte devices	64 Kbyte devices	128 Kbyte devices
Main memory	0x0801 F800 - 0x0801 FFFF	2 K				Page 63
	...	...				...
	0x0801 0000 - 0x0801 07FF	2 K				Page 32
	0x0800 F800 - 0x0800 FFFF	2 K			Page 31	
	...	...				
	0x0800 8000 - 0x0800 87FF	2 K			Page 16	
	0x0800 7800 - 0x0800 7FFF	2 K		Page 15	Page 15	
	...	...		...	...	
	0x0800 4000 - 0x0800 47FF	2 K		Page 8	Page 8	
	0x0800 3800 - 0x0800 3FFF	2 K	Page 7		Page 7	
	...	...	...		...	
	0x0800 1000 - 0x0800 17FF	2 K	Page 2		Page 2	
	0x0800 0800 - 0x0800 0FFF	2 K	Page 1		Page 1	
	0x0800 0000 - 0x0800 07FF	2 K	Page 0		Page 0	

Emulate to EEPROM

How to use EEPROM (write)



Write(PAGE,ADDR,DATA)

Write(0,0,'H')

Write(0,1,'A')

Write(0,2,'R')

Write(0,3,'U')

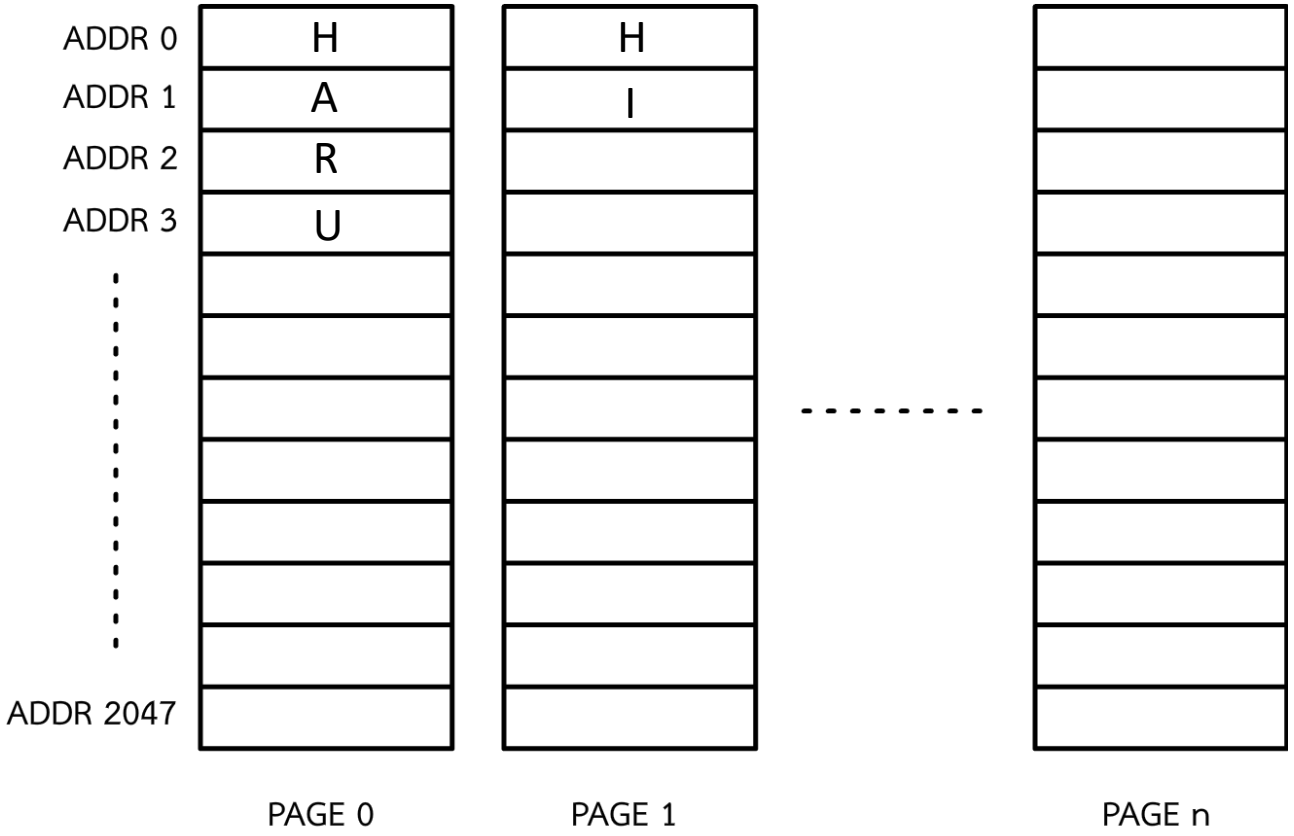
Write(1,0,'H')

Write(1,1,'I')

Writable memory indexing by address

สามารเขียนข้อมูลไปยัง address ได้

How to use EEPROM (Read)



Read(PAGE,ADDR)

Read(0,0) -> ‘H’

Read(0,1) -> ‘A’

Read(0,2) -> ‘R’

Read(0,3) -> ‘U’

Read(1,0) -> ‘H’

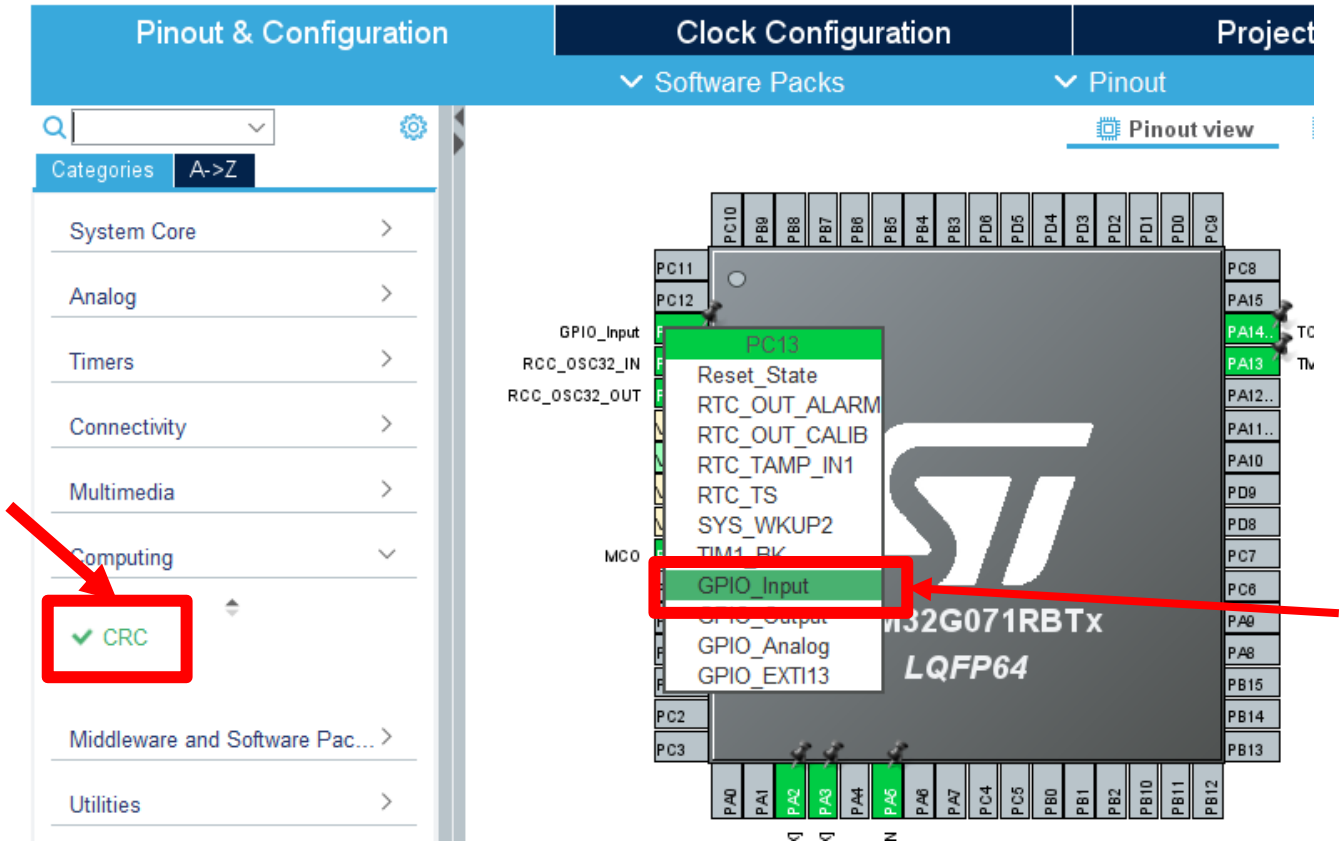
Read(1,1) -> ‘I’

Readable memory using address

สามารอ่านข้อมูลอ้างอิงโดย address

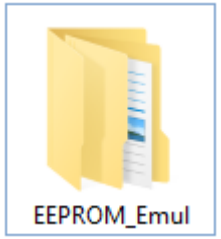
EEPROM on CUBE ide

Enable CRC for EEPROM

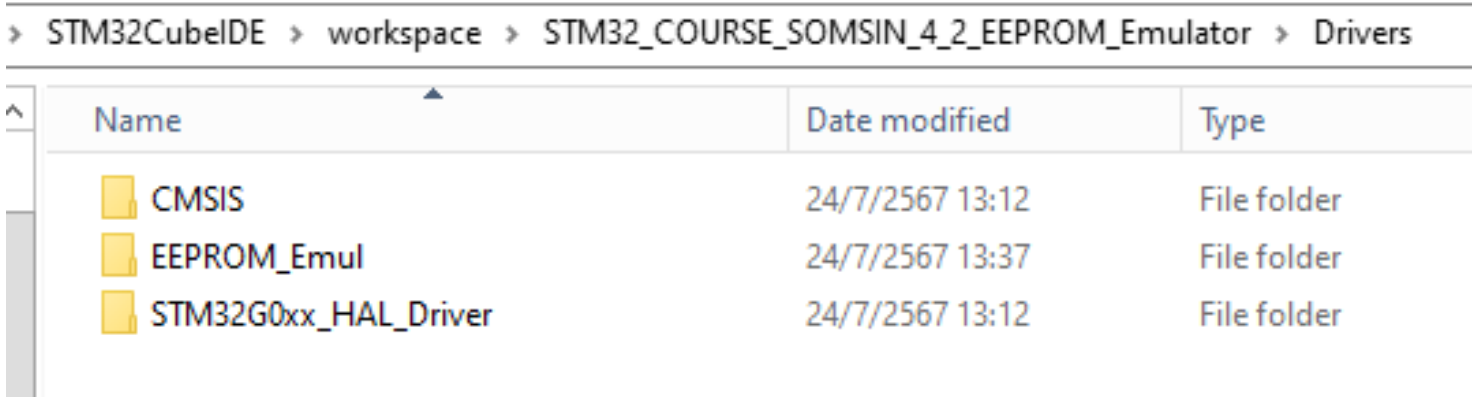


Set PC13 to input
Just for button
(not relate to EEPROM)

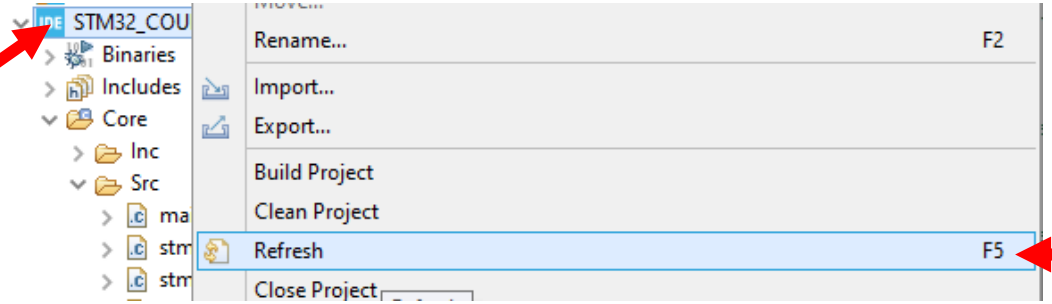
EEPROM on CUBE ide



Copy into Driver Folder



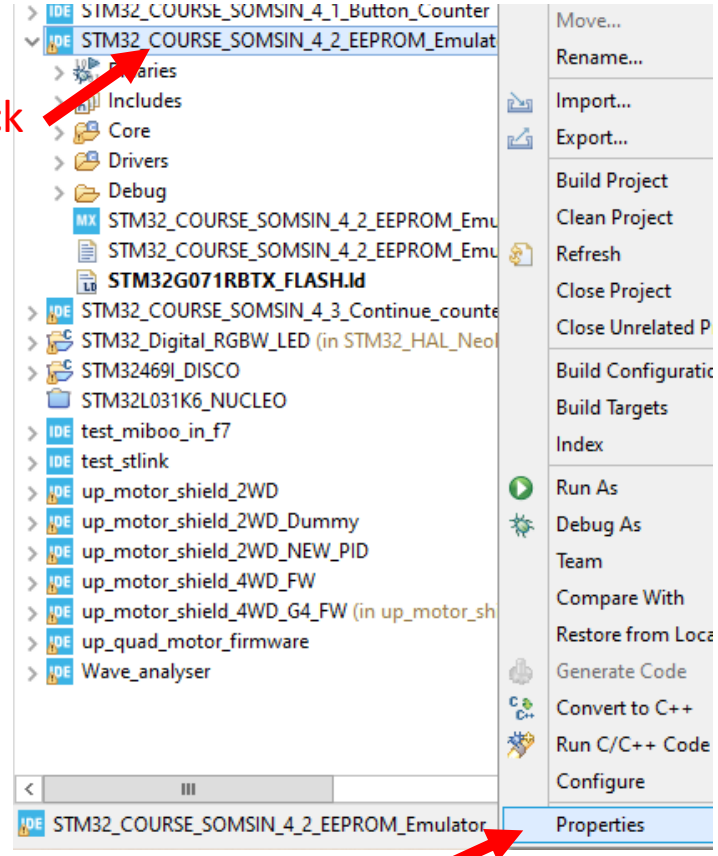
Right Click



Click Refresh to refresh library and reindexing

Config Environment

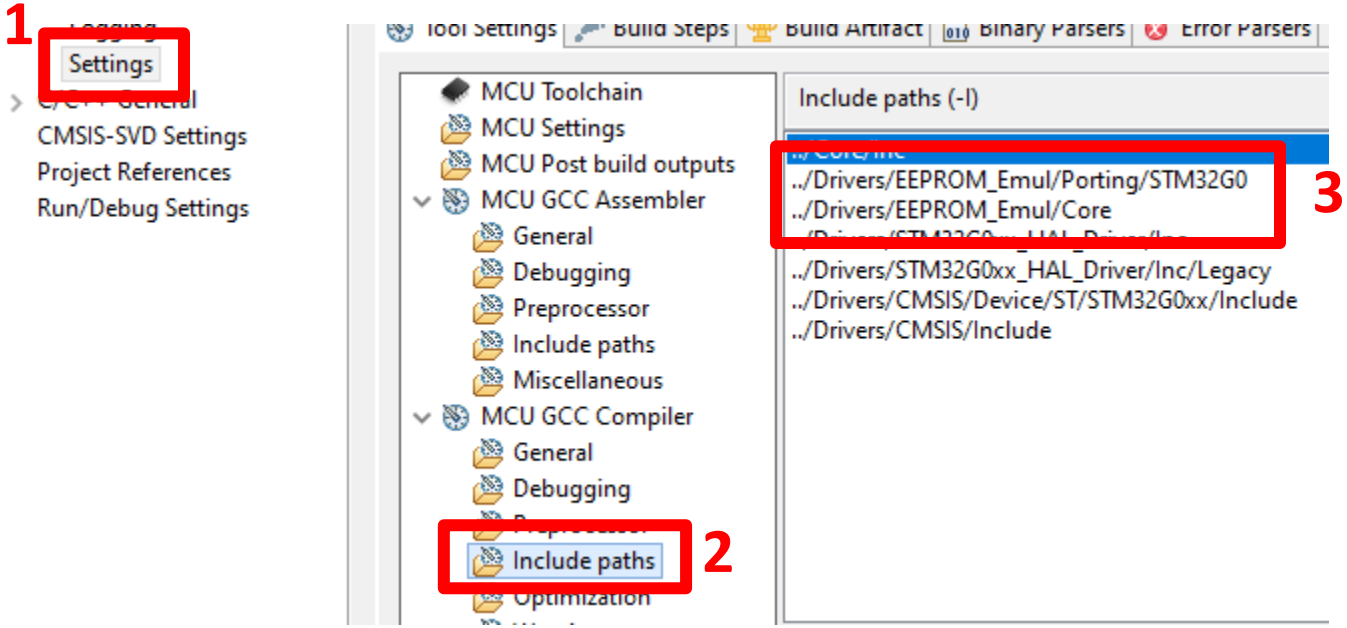
Right Click



Select Properties

Add include Path

1



2

3

coding

```c
/* Private includes ----------
/* USER CODE BEGIN Includes */
#include "eeprom_emul.h"
/* USER CODE END Includes */
```

```c
/* USER CODE BEGIN Includes */
#include "eeprom_emul.h"
/* USER CODE END Includes */
```

```c
MX_CRC_Init();
/* USER CODE BEGIN 2 */

HAL_FLASH_Unlock();
/* EEPROM Init */
HAL_Delay(2000);
EE_Init(EE_FORCED_ERASE);
```

Unlock Flash to Writable Mode

Init EEPROM Emulator (select PAGE 16)

```c
if(HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_13) == 0){ // If User But
    char write_word[20] = "hatsune miku\r\nr";
    for(int i=0;i<15;i++){
        EE_WriteVariable8bits(20 + i, write_word[i]);
    }
}

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  // read data from eeprom
  EE_ReadVariable8bits(20, &string_buffer[0]);
  EE_ReadVariable8bits(21, &string_buffer[1]);
  EE_ReadVariable8bits(22, &string_buffer[2]);
  EE_ReadVariable8bits(23, &string_buffer[3]);
  EE_ReadVariable8bits(24, &string_buffer[4]);
```

Write EEPROM

Read EEPROM

```c
/* USER CODE BEGIN 2 */


HAL_FLASH_Unlock();
/* EEPROM Init */
HAL_Delay(2000);
EE_Init(EE_FORCED_ERASE);


unsigned char string_buffer[50];
int string_buffer_size = -1;


HAL_UART_Transmit(&huart2, (uint8_t *)"haruhi\r\n", 8, 1000);


if(HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_13) == 0){ // If User Button is Pressed the Write EEPROM
char write_word[20] = "hatsune miku\r\nr";
for(int i=0;i<15;i++){
EE_WriteVariable8bits(20 + i, write_word[i]);
}
}


/* USER CODE END 2 */


/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
// read data from eeprom
EE_ReadVariable8bits(20, &string_buffer[0]);
EE_ReadVariable8bits(21, &string_buffer[1]);
EE_ReadVariable8bits(22, &string_buffer[2]);
EE_ReadVariable8bits(23, &string_buffer[3]);
EE_ReadVariable8bits(24, &string_buffer[4]);
EE_ReadVariable8bits(25, &string_buffer[5]);
EE_ReadVariable8bits(26, &string_buffer[6]);
EE_ReadVariable8bits(32, &string_buffer[7]);
EE_ReadVariable8bits(33, &string_buffer[8]);


//string_buffer_size = sprintf(string_buffer,"%s",string_buffer);
string_buffer_size =
sprintf(string_buffer,"%c%c%c%c%c%c%c",string_buffer[0],string_buffer[1],string_buffer[2],string_buffer[3],string_buffer[4],string_buffer[5],string_buffer[6],string_buffer[7
]);
HAL_UART_Transmit(&huart2, string_buffer, string_buffer_size, 1000);


HAL_Delay (1000);
/* USER CODE END WHILE */


/* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

Warning

```
    EE_ReadVariable8bits(32, &string_buffer[7]);
    EE_ReadVariable8bits(33, &string_buffer[8]);

    //string_buffer_size = sprintf(string_buffer,"%s",st
    string_buffer_size = sprintf(string_buffer,"%c%c%c%c
    HAL_UART_Transmit(&huart2, string_buffer, string_buf

    HAL_Delay (1000);
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

Always use delay 500 – 1000 ms

ให้ใส่ delay 500 – 1000 ms เสมอ

Because it is limited cycle to write (prevent write in same address rapidly)

เพื่อป้องกันการเขียนซ้ำที่เดิมเป็นจำนวนมากครั้ง เนื่องจากจะทำให้ Flash address นั้นพัง

Result

```
Asanina
Asahina
Asahina
Asahina
Asahina
Asahina
haruhi
hatsune
hatsune
hatsune
hatsune
hatsune
Serial por
```

Reset

Write to EEPROM

Function

```
EE_Status EE_WriteVariable8bits(uint16_t VirtAddress, uint8_t Data)
```

Write [Data] at memory address [VirtAddress]

บันทึก [Data] ไปยัง [VirtAddress]

```
EE_Status EE_ReadVariable8bits(uint16_t VirtAddress, uint8_t* pData)
```

Read memory at [VirtAddress] and store in [pData]

อ่านข้อมูลจาก address [VirtAddress] และนำไปเก็บไวที่ pData

LAB

ทำให้ button counter นับต่อไปถึงแม้ว่าจะถอด USB และ เสียบใหม่ก็ตาม

ทำต่อจากไฟล์ LAB3 ได้

```
haruhi RESET
Miku 2
Miku 3
Miku 4
Miku 5
Miku 6
haruhi RESET
Miku 7
Miku 8
Miku 9
Miku 10
Miku 11
Miku 12
haruhi RESET
Miku 13
Miku 14
Miku 15
```

Reset

Remove and Plugin USB

Reset