TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA CƠ KHÍ CHẾ TẠO MÁY
BỘ MÔN CƠ ĐIỆN TỬ

Ngày: 05/09/2025
SV: Nguyễn Thiện Nhân
MSSV: 23134041
Lớp: 23134A

SV kí tên: ……

GV chấm điểm: ……...

PHIẾU THỰC HÀNH
LẬP TRÌNH THANH GHI CHO CHỨC NĂNG GPIO

1. Tra cứu thanh ghi:

Boundary address	Peripheral
0x4002 1000 - 0x4002 13FF	Reset and clock control RCC
0x4001 2000 - 0x4001 23FF	GPIO Port G
0x4001 1C00 - 0x4001 1FFF	GPIO Port F
0x4001 1800 - 0x4001 1BFF	GPIO Port E
0x4001 1400 - 0x4001 17FF	GPIO Port D
0x4001 1000 - 0x4001 13FF	GPIO Port C
0x4001 0C00 - 0x4001 0FFF	GPIO Port B
0x4001 0800 - 0x4001 0BFF	GPIO Port A

APB2 peripheral clock enable register (RCC_APB2ENR)

Address: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART 1EN	Res.	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	Reserved		IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.	AFIO EN
	rw		rw	rw	rw	rw			rw	rw	rw	rw	rw		rw

Table 20. Port bit configuration table

Configuration mode		CNF1	CNF0	MODE1	MODE0	PxODR register
General purpose output	Push-pull	0	0	01 10 11 see *Table 21*		0 or 1
	Open-drain		1			0 or 1
Alternate Function output	Push-pull	1	0			Don't care
	Open-drain		1			Don't care
Input	Analog	0	0	00		Don't care
	Input floating		1			Don't care
	Input pull-down	1	0			0
	Input pull-up					1

1.1. Thanh ghi **GPIOx_CRL** (x=A..G) ; Address offset: 0x00

- Chức năng:

Thanh ghi **CRL** là viết tắt của **Port Configuration Register Low**. Nó dùng để cấu hình **8 chân thấp** của mỗi cổng GPIO, tức từ **Pin 0 → Pin 7**.
Mỗi chân sẽ được điều khiển bằng **4 bit** trong thanh ghi này, bao gồm:

MODE[1:0]: chọn chế độ tốc độ Output (nếu cấu hình là Output).

CNF[1:0]: chọn chức năng đặc biệt hoặc kiểu Input/Output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

CNFy[1:0]: Port x configuration bits (y= 0 .. 7)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 20: Port bit configuration table*.
In input mode (MODE[1:0]=00):
00: Analog mode
01: Floating input (reset state)
10: Input with pull-up / pull-down
11: Reserved
In output mode (MODE[1:0] > 00):
00: General purpose output push-pull
01: General purpose output Open-drain
10: Alternate function output Push-pull
11: Alternate function output Open-drain

MODEy[1:0]: Port x mode bits (y= 0 .. 7)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 20: Port bit configuration table*.
00: Input mode (reset state)
01: Output mode, max speed 10 MHz.
10: Output mode, max speed 2 MHz.
11: Output mode, max speed 50 MHz.

- Đoạn chương trình thao tác lên thanh ghi: (ghi lệnh và giải thích ý nghĩa)

```
#include “stdint.h”
#define GPIOA_CRL                    (volatile uint32_t*) (0x40010800+0x00)
/* 0x40010800 là base address của GPIOA, +0x00
Offset (khoảng dịch) đến thanh ghi CRL, dòng này để define một con trỏ trỏ tới
thanh ghi CRL */
#define GPIO_OUTPUT_PP_2MHZ 0x02
/* định nghĩa một hằng số tên GPIO_OUTPUT_PP_2MHZ với giá trị 0x02 */
int main(void) {
    *GPIOA_CRL &= ~(0x0F<<(4*3)); // PA3 xxxxxx0000xxxxx
/* Mỗi chân cũng có 4 bit cấu hình:
MODEy[1:0]: chọn chế độ ngõ ra hoặc input.
CNFy[1:0]: chọn kiểu ngõ ra hoặc kiểu ngõ vào
Dùng để tạo mặt nạ (mask) bao phủ đủ 4 bit cấu hình của 1 pin.
Dịch trái 0x0F sang vị trí của PA3 → 0x0F << 12. */
    *GPIOA_CRL |= (GPIO_OUTPUT_PP_2MHZ << (4*3)); // PA3
```

```
/*
Cấu hình Output Push-Pull 2 MHz có giá trị:
MODE = 10 (Output, tốc độ 2 MHz)
CNF = 00 (General Purpose Output, Push-Pull)
Gộp lại 4 bit: 0010 (binary) = 0x2 (GPIO_OUTPUT_PP_2MHZ).
Dịch trái 0x2 sang vị trí của PA3 → GPIO_OUTPUT_PP_2MHZ  << 12.
*/
}
```

1.2. Thanh ghi **GPIOx_CRH** (x=A..G) ; Address offset: 0x04

Chức năng: Thanh ghi **CRH** là viết tắt của **Port Configuration Register HIGH**. Nó dùng để cấu hình **8 chân cao** của mỗi cổng GPIO, tức từ **Pin 8 → Pin 15**. Mỗi chân sẽ được điều khiển bằng **4 bit** trong thanh ghi này, bao gồm:

MODE[1:0]: chọn chế độ tốc độ Output (nếu cấu hình là Output).

CNF[1:0]: chọn chức năng đặc biệt hoặc kiểu Input/Output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

CNFy[1:0]: Port x configuration bits (y= 8 .. 15)

These bits are written by software to configure the corresponding I/O port.

Refer to *Table 20: Port bit configuration table*.

In input mode (MODE[1:0]=00):

00: Analog mode

01: Floating input (reset state)

10: Input with pull-up / pull-down

11: Reserved

In output mode (MODE[1:0] > 00):

00: General purpose output push-pull

01: General purpose output Open-drain

10: Alternate function output Push-pull

11: Alternate function output Open-drain

MODEy[1:0]: Port x mode bits (y= 8 .. 15)

These bits are written by software to configure the corresponding I/O port.

Refer to *Table 20: Port bit configuration table*.

00: Input mode (reset state)

01: Output mode, max speed 10 MHz.

10: Output mode, max speed 2 MHz.

11: Output mode, max speed 50 MHz.

- Đoạn chương trình thao tác lên thanh ghi: (ghi lệnh và giải thích ý nghĩa)

```
#include “stdint.h”
#define GPIOA_CRH                  (volatile uint32_t*) (0x40010800+0x04)
/* 0x40010800 là base address của GPIOA, +0x04
Offset (khoảng dịch) đến thanh ghi CRH, dòng này để define một con trỏ trỏ
tới thanh ghi CRH */
#define GPIO_OUTPUT_PP_2MHZ 0x02
```

```
/* định nghĩa một hằng số tên GPIO_OUTPUT_PP_2MHZ với giá trị 0x02 */
int main(void) {
    *GPIOA_CRH &= ~(0x0F<<(4*3));
/* Mỗi chân cũng có 4 bit cấu hình:
MODEy[1:0]: chọn chế độ ngõ ra hoặc input.
CNFy[1:0]: chọn kiểu ngõ ra hoặc kiểu ngõ vào
Dùng để tạo mặt nạ (mask) bao phủ đủ 4 bit cấu hình của 1 pin.
Dịch trái 0x0F sang vị trí của PA11 → 0x0F << 12. */
    *GPIOA_CRH |= (GPIO_OUTPUT_PP_2MHZ << (4*3)); // PA11
/*
Cấu hình Output Push-Pull 2 MHz có giá trị:
MODE = 10 (Output, tốc độ 2 MHz)
CNF = 00 (General Purpose Output, Push-Pull)
Gộp lại 4 bit: 0010 (binary) = 0x2 (GPIO_OUTPUT_PP_2MHZ).
Dịch trái 0x2 sang vị trí của PA3 → GPIO_OUTPUT_PP_2MHZ  << 12.
*/
}
```

1.3. Thanh ghi **GPIOx_IDR** (x=A..G) ; Address offset: 0x08

Chức năng: IDR là viết tắt của Input Data Register – tức thanh ghi dữ liệu ngõ vào.

Mỗi cổng GPIO (A, B, C, …) đều có một thanh ghi IDR riêng.

Độ rộng: 16 bit (nhưng được đặt trong thanh ghi 32 bit, IDR 16 bit thấp) .

Chỉ đọc trạng thái điện áp thực tế tại pin (dù pin đó là input hay output).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDRy:** Port input data (y= 0 .. 15)

These bits are read only and can be accessed in Word mode only. They contain the input value of the corresponding I/O port.

.....

.....

.....

.....

- Đoạn chương trình thao tác lên thanh ghi: (ghi lệnh và giải thích ý nghĩa)

```
#include “stdint.h”
#define GPIOA_IDR (volatile uint32_t*) (0x40010800+0x08)
// Khai bao con trỏ thanh ghi IDR của GPIOA
#define GPIOA_CRL (volatile uint32_t*) (0x40010800+0x00)
/* 0x40010800 là base address của GPIOA, +0x00
Offset (khoảng dịch) đến thanh ghi CRL, dòng này để define một con trỏ trỏ
tới thanh ghi CRL */
```

```
#define GPIO_OUTPUT_PP_2MHZ 0x02
/* định nghĩa một hằng số tên GPIO_OUTPUT_PP_2MHZ với giá trị 0x02 */
int main(void) {
    *GPIOA_CRL &= ~(0x0F<<(4*3)); // PA3 xxxxxx0000xxxxx
/* Mỗi chân cũng có 4 bit cấu hình:
MODEy[1:0]: chọn chế độ ngõ ra hoặc input.
CNFy[1:0]: chọn kiểu ngõ ra hoặc kiểu ngõ vào
Dùng để tạo mặt nạ (mask) bao phủ đủ 4 bit cấu hình của 1 pin.
Dịch trái 0x0F sang vị trí của PA3 → 0x0F << 12. */
    *GPIOA_CRL |= (GPIO_OUTPUT_PP_2MHZ << (4*3)); // PA3
/*
Cấu hình Output Push-Pull 2 MHz có giá trị:
MODE = 10 (Output, tốc độ 2 MHz)
CNF = 00 (General Purpose Output, Push-Pull)
Gộp lại 4 bit: 0010 (binary) = 0x2 (GPIO_OUTPUT_PP_2MHZ).
Dịch trái 0x2 sang vị trí của PA3 → GPIO_OUTPUT_PP_2MHZ  << 12.
*/
    uint32_t pinState = (GPIOA_IDR >>  3)&0x01;
    /* Đọc trạng thái của chân PA3, bit thứ 3 trong thanh ghi IDR, dùng
bitwise and trả về giá trị logic 0, 1*/
}
```

1.4. Thanh ghi **GPIOx_ODR** (x=A..G) Address offset: 0x0C

Chức năng: ODR (Output Data Register) là thanh ghi dùng để lưu và điều khiển trạng thái xuất (output) của các chân GPIO. Mỗi cổng GPIO (A, B, C, …) đều có một thanh ghi ODR riêng. Độ rộng16 bit (nhưng được đặt trong thanh ghi 32 bit, dùng 16 bit thấp) .Khi ghi giá trị 1 hoặc 0 vào ODR, chân GPIO sẽ xuất mức cao (VDD) hoặc mức thấp (GND), nếu chân đó được cấu hình là output. Có thể đọc lại trạng thái lưu trong thanh ghi ODR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODRy:** Port output data (y= 0 .. 15)

These bits can be read and written by software and can be accessed in Word mode only.

Note: For atomic bit set/reset, the ODR bits can be individually set and cleared by writing to the GPIOx BSRR register (x = A .. G).

.....

.....

.....

- Đoạn chương trình thao tác lên thanh ghi: (ghi lệnh và giải thích ý nghĩa)

```
#include “stdint.h”
#define GPIOA_ODR (volatile uint32_t*) (0x40010800+0x0C)
// Khai bao con trỏ thanh ghi ODR của GPIOA
#define GPIOA_CRL                  (volatile uint32_t*) (0x40010800+0x00)
```

```c
/* 0x40010800 là base address của GPIOA, +0x00
Offset (khoảng dịch) đến thanh ghi CRL, dòng này để define một con trỏ trỏ tới
thanh ghi CRL */
#define GPIO_OUTPUT_PP_2MHZ 0x02
/* định nghĩa một hằng số tên GPIO_OUTPUT_PP_2MHZ với giá trị 0x02 */
int main(void) {
    *GPIOA_CRL &= ~(0x0F<<(4*3)); // PA3 xxxxxx0000xxxxx
/* Mỗi chân cũng có 4 bit cấu hình:
MODEy[1:0]: chọn chế độ ngõ ra hoặc input.
CNFy[1:0]: chọn kiểu ngõ ra hoặc kiểu ngõ vào
Dùng để tạo mặt nạ (mask) bao phủ đủ 4 bit cấu hình của 1 pin.
Dịch trái 0x0F sang vị trí của PA3 → 0x0F << 12. */
    *GPIOA_CRL |= (GPIO_OUTPUT_PP_2MHZ << (4*3)); // PA3
/*
Cấu hình Output Push-Pull 2 MHz có giá trị:
MODE = 10 (Output, tốc độ 2 MHz)
CNF = 00 (General Purpose Output, Push-Pull)
Gộp lại 4 bit: 0010 (binary) = 0x2 (GPIO_OUTPUT_PP_2MHZ).
Dịch trái 0x2 sang vị trí của PA3 → GPIO_OUTPUT_PP_2MHZ  << 12.
*/
    *GPIOA_ODR |= (1U << 3);     // Set chân PA3
    *GPIOA_ODR &= ~(1U << 3); // Reset chân PA3
    uint32_t pinState = (*GPIOA_ODR >> 3 )&0x01;    // Đọc lại trạng thái đã
ghi chân PA3 (bit thứ 3 trong thanh ghi ODR), dùng bitwise and để trả về 0 1
}
```

1.5. Thanh ghi **GPIOx_BSRR** (x=A..G) ; Address offset: 0x10

Chức năng: BSRR (Bit Set/Reset Register) là thanh ghi đặc biệt dùng để set hoặc reset các chân GPIO mà không cần thao tác trực tiếp với ODR. Mỗi cổng GPIO có 1 thanh ghi 32-bit BSRR. BSRR cho phép set (đặt 1) hoặc reset (đặt 0) từng chân GPIO mà không ảnh hưởng các chân khác, rất an toàn khi thao tác nhiều chân hoặc trong ngắt.

Cấu trúc:

- Bit 0–15: Dùng để set (đặt 1) cho các chân tương ứng.
- Bit 16–31: Dùng để reset (đặt 0) cho các chân tương ứng.

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 mode only.
0: No action on the corresponding ODRx bit
1: Reset 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 mode only.
0: No action on the corresponding ODRx bit
1: Set the corresponding ODRx bit

.....

.....

.....

- Đoạn chương trình thao tác lên thanh ghi: (ghi lệnh và giải thích ý nghĩa)

```
#include “stdint.h”
#define GPIOA_BSRR (volatile uint32_t*) (0x40010800 + 0x10)
// Khai báo con trỏ thanh ghi BSRR của GPIOA
#define GPIOA_CRL                  (volatile uint32_t*) (0x40010800+0x00)
/* 0x40010800 là base address của GPIOA, +0x00
Offset (khoảng dịch) đến thanh ghi CRL, dòng này để define một con trỏ trỏ tới
thanh ghi CRL */
#define GPIO_OUTPUT_PP_2MHZ 0x02
/* định nghĩa một hằng số tên GPIO_OUTPUT_PP_2MHZ với giá trị 0x02 */
int main(void) {
    *GPIOA_CRL &= ~(0x0F<<(4*3)); // PA3 xxxxxx0000xxxxx
/* Mỗi chân cũng có 4 bit cấu hình:
MODEy[1:0]: chọn chế độ ngõ ra hoặc input.
CNFy[1:0]: chọn kiểu ngõ ra hoặc kiểu ngõ vào
Dùng để tạo mặt nạ (mask) bao phủ đủ 4 bit cấu hình của 1 pin.
Dịch trái 0x0F sang vị trí của PA3 → 0x0F << 12. */
    *GPIOA_CRL |= (GPIO_OUTPUT_PP_2MHZ << (4*3)); // PA3
/*
Cấu hình Output Push-Pull 2 MHz có giá trị:
MODE = 10 (Output, tốc độ 2 MHz)
CNF = 00 (General Purpose Output, Push-Pull)
Gộp lại 4 bit: 0010 (binary) = 0x2 (GPIO_OUTPUT_PP_2MHZ).
Dịch trái 0x2 sang vị trí của PA3 → GPIO_OUTPUT_PP_2MHZ  << 12.
*/
    *GPIOA_BSRR = (1U << 3); // Set chân PA3
    *GPIOA_BSRR = (1U << (3+16)); // RESET chân PA3
    // bit 0-15 là set, bit 16-31 là reset
}
```

1.6. Thanh ghi **GPIOx_BRR** (x=A..G) Address offset: 0x14

Chức năng: BRR (Bit Reset Register) là thanh ghi dùng để reset (đặt 0) các chân GPIO, tức là đưa chân xuống mức thấp logic (GND). Mỗi bit trong BRR tương ứng với một chân GPIO. Khi ghi 1 vào bit i → chân GPIO tương ứng bị reset (0). Ghi 0 → không ảnh hưởng.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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

Bits 31:16 Reserved

Bits 15:0 **BRy:** Port x Reset bit y (y= 0 .. 15)
These bits are write-only and can be accessed in Word mode only.
0: No action on the corresponding ODRx bit
1: Reset the corresponding ODRx bit

.....

.....

.....

- Đoạn chương trình thao tác lên thanh ghi: (ghi lệnh và giải thích ý nghĩa)

```
#include “stdint.h”
#define GPIOA_BSRR (volatile uint32_t*) (0x40010800 + 0x14)
// Khai báo con trỏ thanh ghi BRR của GPIOA
#include “stdint.h”
#define GPIOA_CRL                    (volatile uint32_t*) (0x40010800+0x00)
/* 0x40010800 là base address của GPIOA, +0x00
Offset (khoảng dịch) đến thanh ghi CRL, dòng này để define một con trỏ trỏ tới
thanh ghi CRL */
#define GPIO_OUTPUT_PP_2MHZ 0x02
/* định nghĩa một hằng số tên GPIO_OUTPUT_PP_2MHZ với giá trị 0x02 */
int main(void) {
    *GPIOA_CRL &= ~(0x0F<<(4*3)); // PA3 xxxxxx0000xxxxx
/* Mỗi chân cũng có 4 bit cấu hình:
MODEy[1:0]: chọn chế độ ngõ ra hoặc input.
CNFy[1:0]: chọn kiểu ngõ ra hoặc kiểu ngõ vào
Dùng để tạo mặt nạ (mask) bao phủ đủ 4 bit cấu hình của 1 pin.
Dịch trái 0x0F sang vị trí của PA3 → 0x0F << 12. */
    *GPIOA_CRL |= (GPIO_OUTPUT_PP_2MHZ << (4*3)); // PA3
/*
Cấu hình Output Push-Pull 2 MHz có giá trị:
MODE = 10 (Output, tốc độ 2 MHz)
CNF = 00 (General Purpose Output, Push-Pull)
Gộp lại 4 bit: 0010 (binary) = 0x2 (GPIO_OUTPUT_PP_2MHZ).
Dịch trái 0x2 sang vị trí của PA3 → GPIO_OUTPUT_PP_2MHZ  << 12.
*/
    *GPIOA_BRR = (1U << 3); // Reset chân PA3
}
```

2. Các chương trình mẫu:

2.1. Chương trình bật/tắt PA1 theo chu kỳ 300ms:

```c
#include "stdint.h"
#define RCC_BASE          0x40021000
#define GPIOA_BASE        0x40010800
#define GPIO_OUTPUT_2MHZ  0x02     // 0B0010
#define RCC_APB2ENR (volatile uint32_t *)(RCC_BASE+0x18)
#define GPIOA_CRL   (volatile uint32_t *)(GPIOA_BASE+0x00)
#define GPIOA_ODR   (volatile uint32_t *)(GPIOA_BASE+0x0C)
void Delay_ms (uint32_t ms){
  uint32_t i = 0;
  while(ms>0) {
    while (i<4) {
      i++;
      __asm__("nop");
    }
    ms--;
    i=0;
  }
}
int main() {
  *RCC_APB2ENR  |= (1<<2);                    // Enable GPIOA clock
  *GPIOA_CRL    |= GPIO_OUTPUT_2MHZ<<(1*4) ;  // PA1
  *GPIOA_ODR    |= 1<<1;                      // PA1
  while(1) {
      *GPIOA_ODR    |= 1<<1;                  // PA1
      Delay_ms(300);
      *GPIOA_ODR    &= ~(1<<1);               // PA1
      Delay_ms(300);
  }
}
```

Giải thích các giá trị thanh ghi:

RCC_BASE: Địa chỉ cơ sở (base address) của khối Reset and Clock Control (RCC) trong vi điều khiển STM32

GPIOA_BASE: Địa chỉ cơ sở của GPIOA

RCC_APB2ENR: Thanh ghi APB2 Peripheral Clock Enable của RCC. Offset 0x18 từ RCC_BASE

GPIOA_CRL: CRL (Configuration Register Low) của GPIOA. Offset 0x00 từ GPIOA_BASE

GPIOA_ODR: ODR (Output Data Register) của GPIOA. Offset 0x0C từ GPIOA_BASE

***RCC_APB2ENR |= (1 << 2);** Cấu hình bit 2 trong thanh ghi APB2ENR (IOPAEN) = 1

***GPIOA_CRL &= ~(0x0F << (4*1))**; Xóa 4 bit của PA1 (CNF[1:0], MODE[1:0])

***GPIOA_CRL |= (GPIO_OUTPUT_2MHZ << (4*1));** Cấu hình PA1 là OUTPUT PUSH PULL 2MHz (0b0010, CNF 00, MODE 10)

***GPIOA_ODR |= (1 << 1);** Set PA1, (1<<1) = 0b10 ứng với PA1 là bit 1

***GPIOA_ODR &= ~(1 << 1);** // Clear PA1, ~(1<<1) = 0b01 ứng với PA1 là bit 0

2.2. Chương trình đọc nút nhấn tại ngõ vào PA2 và điều khiển ngõ ra PA1 bật/tắt tương ứng:

```
#include "stdint.h"
#define RCC_BASE          0x40021000
#define GPIOA_BASE        0x40010800
#define GPIO_OUTPUT_2MHZ  0x02    // 0B0010
#define GPIO_INPUT        0x08    // 0B1000
#define RCC_APB2ENR (volatile uint32_t *)(RCC_BASE+0x18)
#define GPIOA_CRL   (volatile uint32_t *)(GPIOA_BASE+0x00)
#define GPIOA_ODR   (volatile uint32_t *)(GPIOA_BASE+0x0C)
#define GPIOA_IDR   (volatile uint32_t *)(GPIOA_BASE+0x08)

int main() {
  *RCC_APB2ENR  |= (1<<2);                    // Enable GPIOA clock
  *GPIOA_CRL    |= GPIO_OUTPUT_2MHZ<<(1*4) ;  // PA1 output
  *GPIOA_CRL    &= ~(0x0F<<2*4) ;             // Clear CF1:CF0:MODE1:MODE0
  *GPIOA_CRL    |= GPIO_INPUT<<(2*4) ;        // PA2 input
  *GPIOA_ODR    |= 1<<2 ;                     // PA2 PULL-UP

  while(1) {
      if (*GPIOA_IDR&(1<<2))
        *GPIOA_ODR |= 1<<1;
      else
        *GPIOA_ODR &= ~(1<<1);
  }
}
```

RCC_BASE: Địa chỉ cơ sở (base address) của khối Reset and Clock Control (RCC) trong vi điều khiển STM32

GPIOA_BASE: Địa chỉ cơ sở của GPIOA

GPIO_INPUT: Input, chế độ pull-up/pull-down.

RCC_APB2ENR: Con trỏ đến thanh ghi APB2 Peripheral Clock Enable của RCC. Offset 0x18 từ RCC_BASE

GPIOA_CRL: Con trỏ đến thanh ghi CRL (Configuration Register Low) của GPIOA. Offset 0x00 từ GPIOA_BASE

GPIOA_ODR: Con trỏ đến thanh ghi ODR (Output Data Register) của GPIOA. Offset 0x08 từ GPIOA_BASE

GPIOA_IDR: Con trỏ đến thanh ghi IDR (Input Data Register) của GPIOA. Offset 0x0C từ GPIOA_BASE

***RCC_APB2ENR |= (1 << 2);** Cấu hình bit 2 trong thanh ghi APB2ENR (IOPAEN) = 1

***GPIOA_CRL |= GPIO_OUTPUT_2MHZ<<(1*4);** Cấu hình chân PA1 làm Output, tốc độ 2MHz.

***GPIOA_CRL &= ~(0x0F << (4*1))**; Xóa 4 bit của PA2 (CNF[1:0], MODE[1:0])

***GPIOA_CRL |= (GPIO_INPUT << (2*4));** Cấu hình PA2 Input mode, pull-up/pull-down.

***GPIOA_ODR |= (1 << 2);** Kích hoạt pull-up cho PA2 (bằng cách set bit ODR tương ứng).

if (*GPIOA_IDR & (1<<2))

***GPIOA_ODR |= 1<<1;**

else

***GPIOA_ODR &= ~(1<<1);**

Đọc giá trị từ chân PA2.

Nếu PA2 ở mức HIGH (bit IDR2 = 1) → set PA1 lên mức HIGH.

Nếu PA2 ở mức LOW (bit IDR2 = 0) → reset PA1 về LOW.

3. Các bài thực hành:

3.1. Viết chương trình sáng tắt so le PA1 và PA2

```
#include "stdint.h"

#define GPIOA_BASE  0x40010800
#define RCC_BASE        0x40021000
#define GPIOA_CRL               ((volatile uint32_t*) (GPIOA_BASE + 0x00))
#define GPIOA_BSRR          ((volatile uint32_t*) (GPIOA_BASE + 0x10))
#define RCC_APB2ENR         ((volatile uint32_t*) (RCC_BASE+0x18))
typedef enum {
    Pin_1 = 1,
    Pin_2 = 2
} PinNum;

#define OUTPUT_PP_2MHZ  0x02

void Delay_ms(uint32_t ms) {
    while(ms) {
        for(uint8_t i = 0; i < 4; i++) {
            __asm("nop");
        }
        ms--;
    }
}


int main(void) {
    *RCC_APB2ENR |= (1U << 2);  // Enable GPIOA Clock
    *GPIOA_CRL &= ~(0x0F << Pin_1*4); // Clear CNF[1:0], MODE[1:0] PA1
    *GPIOA_CRL &= ~(0x0F << Pin_2*4); // Clear CNF[1:0], MODE[1:0] PA2
    *GPIOA_CRL |= (OUTPUT_PP_2MHZ << 1*4); // PA1 Output pushpull 2MHz
    *GPIOA_CRL |= (OUTPUT_PP_2MHZ << 2*4); // PA2 Output pushpull 2MHz


    uint32_t led_state = 0x01; // PA1 ON, PA2 OFF, bit 0 - PA1, bit 1 - PA2
    while(1) {
        *GPIOA_BSRR = (1U << (16+Pin_1)); // RESET PA1
        *GPIOA_BSRR = (1U << (16+Pin_2)); // RESET PA2
```

```
        *GPIOA_BSRR = (led_state &0x01)?(1U << Pin_1):(1U <<
(16+Pin_1));               // Output state to PA1
        *GPIOA_BSRR = ((led_state >> 1) &0x01)?(1U << Pin_2):(1U <<
(16+Pin_2));    // Output state to PA2

        Delay_ms(600);

        // Toggle led_state 01 and 10
        led_state = (~led_state) & 0x03;
    }
}
```

3.2. Viết chương trình một điểm sáng chạy từ PA0 đến PA7; trì hoãn 0,5s

```
#include "stdint.h"

#define GPIOA_BASE  0x40010800
#define RCC_BASE    0x40021000
#define GPIOA_CRL           ((volatile uint32_t*) (GPIOA_BASE + 0x00))
#define GPIOA_BSRR          ((volatile uint32_t*) (GPIOA_BASE + 0x10))
#define RCC_APB2ENR         ((volatile uint32_t*) (RCC_BASE+0x18))
typedef enum {
    Pin_0,
    Pin_1,
    Pin_2,
    Pin_3,
    Pin_4,
    Pin_5,
    Pin_6,
    Pin_7
} PinNum;

#define OUTPUT_PP_2MHZ  0x02

void Delay_ms(uint32_t ms) {
    while(ms) {
        for(uint8_t i = 0; i < 4; i++) {
            __asm("nop");
        }
        ms--;
    }
}


int main(void) {
    *RCC_APB2ENR |= (1U << 2);  // Enable GPIOA Clock
    for(uint8_t pin = Pin_0; pin <= Pin_7; pin++) {
        *GPIOA_CRL &= ~(0x0F << pin*4); // Clear CNF[1:0], MODE[1:0] PAx
        *GPIOA_CRL |= (OUTPUT_PP_2MHZ << pin*4); // PAx Output pushpull 2MHz
```

```
        *GPIOA_BSRR = (1U << (16+pin)); // RESET PAx
    }
    uint32_t led_state = 0x00;
    while(1) {
        if(led_state == 0xFF) {
            *GPIOA_BSRR = (led_state << 16);    // RESET ALL LED
            led_state = 0x00;
        } else {
            led_state = (led_state << 1) | 1; // Dich bit bien led_state và |
LSB len 1, sang dan led (tu chan PA0 den chan PA7)
        }
        *GPIOA_BSRR = led_state;    // Set cac pin theo bien led_state
        Delay_ms(500);                      // Delay 0.5s

    }
}
```

3.3. Viết chương trình điều khiển sự sáng tắt của PB0 theo nút nhấn gắn tại PA0.

```
#include "stdint.h"

#define GPIOA_BASE  0x40010800
#define GPIOB_BASE  0x40010C00
#define RCC_BASE    0x40021000


#define RCC_APB2ENR ((volatile uint32_t*) (RCC_BASE+0x18))

#define GPIOA_CRL   ((volatile uint32_t*) (GPIOA_BASE+0x00))
#define GPIOB_CRL   ((volatile uint32_t*) (GPIOB_BASE+0x00))

#define GPIOA_IDR   ((volatile uint32_t*) (GPIOA_BASE+0x08))
#define GPIOB_IDR   ((volatile uint32_t*) (GPIOB_BASE+0x08))
#define GPIOB_BSRR  ((volatile uint32_t*) (GPIOB_BASE+0x10))
#define GPIOA_ODR   ((volatile uint32_t*) (GPIOA_BASE+0x0C))

#define OUTPUT_PP_2MHZ  0x02    // CNF 00, MODE 10
#define INPUT_PD_PU     0x08    // CNF 10, MODE 00

void Delay_ms(uint32_t ms) {
    while(ms) {
        for(uint8_t i = 0; i < 4; i++) {
            __asm("nop");
        }
        ms--;
    }
}
```

```c
int main(void) {

  *RCC_APB2ENR |= (3U << 2);  //  Enable GPIOA, GPIOB Clock


  *GPIOB_CRL &= ~0x0F;            // Reset CNF,MODE Pin0 GPIOB
  *GPIOB_CRL |= OUTPUT_PP_2MHZ;   // Output pp 2MHZ
  *GPIOA_CRL &= ~0x0F;            // Reset CNF,MODE Pin0 GPIOA
  *GPIOA_CRL |= INPUT_PD_PU;      // INPUT Pull Down, Pull Up
  *GPIOA_ODR &= ~1U;              // Input Pull Down

  *GPIOB_BSRR = 1U << 16;               // Output state LOW on PB0

  while(1) {
    if((*GPIOA_IDR & 0x01)) {
      Delay_ms(5);
      if((*GPIOA_IDR & 0x01)) {
        *GPIOB_BSRR = (*GPIOB_IDR & 0x01)?(1U << 16):(1U);    // Toggle LED
        while((*GPIOA_IDR & 0x01));
      }
    } // Debouncing

  }
}
```

3.4. Viết chương trình một điểm sáng di chuyển từ PA0 tới PA7 theo sự nhấn nút tại PB0.

```c
#include "stdint.h"

#define GPIOA_BASE  0x40010800
#define GPIOB_BASE  0x40010C00


#define RCC_BASE    0x40021000
#define GPIOA_CRL       ((volatile uint32_t*) (GPIOA_BASE + 0x00))
#define GPIOA_ODR       ((volatile uint32_t*) (GPIOA_BASE + 0x0C))
#define RCC_APB2ENR     ((volatile uint32_t*) (RCC_BASE+0x18))

#define GPIOB_CRL   ((volatile uint32_t*) (GPIOB_BASE+0x00))
#define GPIOB_IDR   ((volatile uint32_t*) (GPIOB_BASE+0x08))
#define GPIOB_ODR   ((volatile uint32_t*) (GPIOB_BASE+0x0C))


typedef enum {
  Pin_0,
  Pin_1,
  Pin_2,
  Pin_3,
  Pin_4,
  Pin_5,
```

```
  Pin_6,
  Pin_7
} PinNum;

typedef enum {
  BTN_OFF,
  BTN_ON
} ButtonState_t;
#define OUTPUT_PP_2MHZ  0x02
#define INPUT_PD_PU     0x08    // CNF 10, MODE 00


void Delay_ms(uint32_t ms) {
  while(ms) {
    for(uint8_t i = 0; i < 4; i++) {
      __asm("nop");
    }
    ms--;
  }
}


int main(void) {
  ButtonState_t ButtonState = BTN_OFF;
  *RCC_APB2ENR |= (3U << 2);  // Enable GPIOA, GPIOB Clock
  for(uint8_t pin = Pin_0; pin <= Pin_7; pin++) {
    *GPIOA_CRL &= ~(0x0F << pin*4); // Clear CNF[1:0], MODE[1:0] PAx
    *GPIOA_CRL |= (OUTPUT_PP_2MHZ << pin*4); // PAx Output pushpull 2MHz
  }
  uint32_t led_state = 0x00; // LedState - OFF ALL LED
  *GPIOA_ODR &= (~0xFF);    // OFF ALL LED


  *GPIOB_CRL &= ~0x0F;            // Clr CNF, MODE
  *GPIOB_CRL |= INPUT_PD_PU;      // INPUT Pull Down, Pull Up
  *GPIOB_ODR &= ~1U;              // Input Pull Down




  while(1) {
    if((*GPIOB_IDR & 0x01)) {
      Delay_ms(5);
      if((*GPIOB_IDR & 0x01)) {
        ButtonState ^= BTN_ON;  // Toggle led state
        while((*GPIOB_IDR & 0x01));
      }
    } // Debouncing
```

```
    if(ButtonState == BTN_ON) {
      if(led_state == 0xFF) {
        *GPIOA_ODR &= (~0xFF);  // OFF ALL LED
        led_state = 0x00;
      } else {
        led_state = (led_state << 1) | 1; // Dich bit bien led_state và | LSB
len 1, sang dan led (tu chan PA0 den chan PA7)
        *GPIOA_ODR = led_state; // Set cac pin theo bien led_state
      }
      Delay_ms(500);            // Delay 0.5s

    } else {
      led_state = 0xFF;
      *GPIOA_ODR &= (~0xFF);  // OFF ALL LED
      led_state = 0x00;

    }

  }
}
```