TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM

Ngày: 05/09/2025

SV: Nguyễn Thiện Nhân

MSSV: 23134041

Lớp: 23134A

SV kí tên: ……………………

GV chấm điểm: ……………...

KHOA CƠ KHÍ CHẾ TẠO MÁY

**BỘ MÔN CƠ ĐIỆN TỬ**

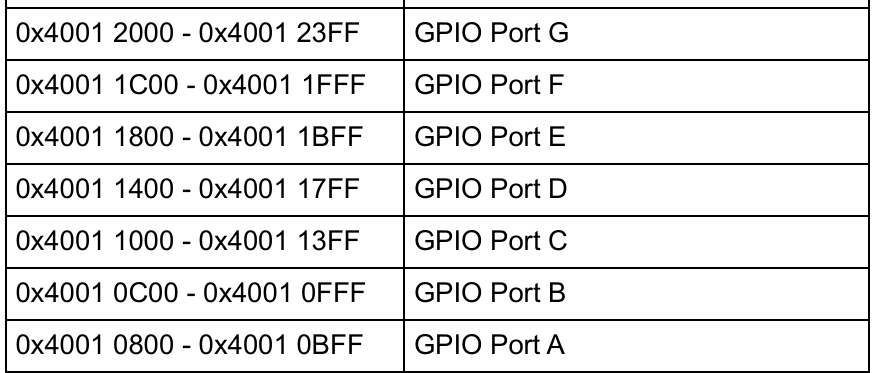
**PHIẾU THỰC HÀNH**

**LẬP TRÌNH THANH GHI CHO CHỨC NĂNG GPIO**

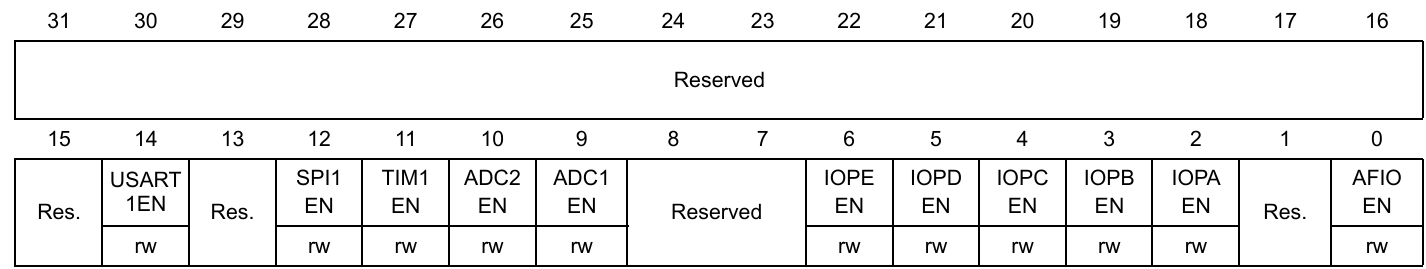
1. **Tra cứu thanh ghi:**

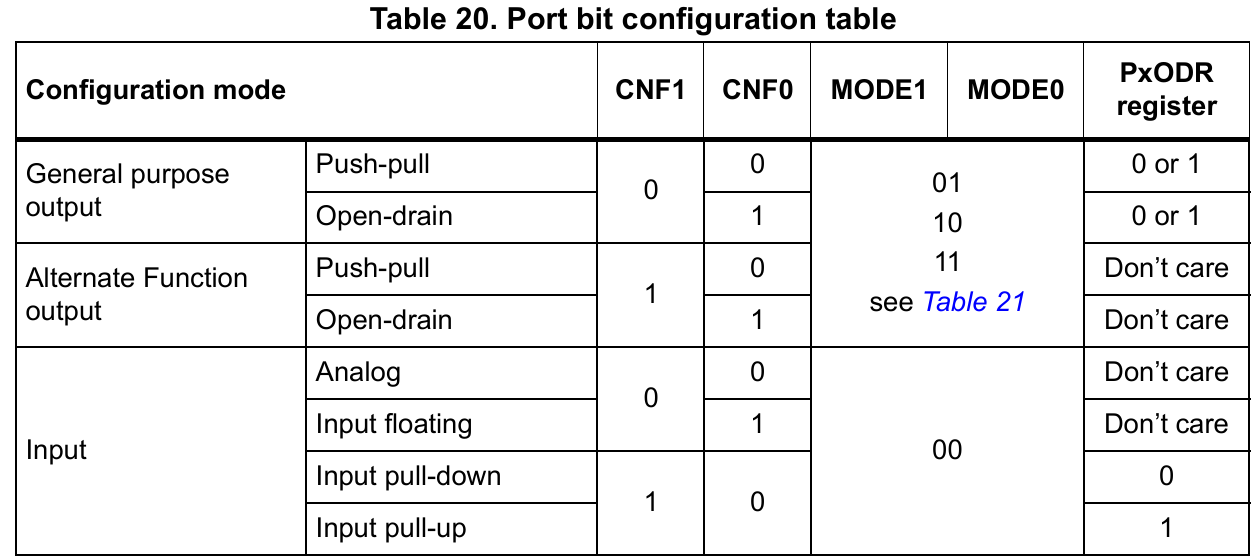
****

****

****

****

****

****

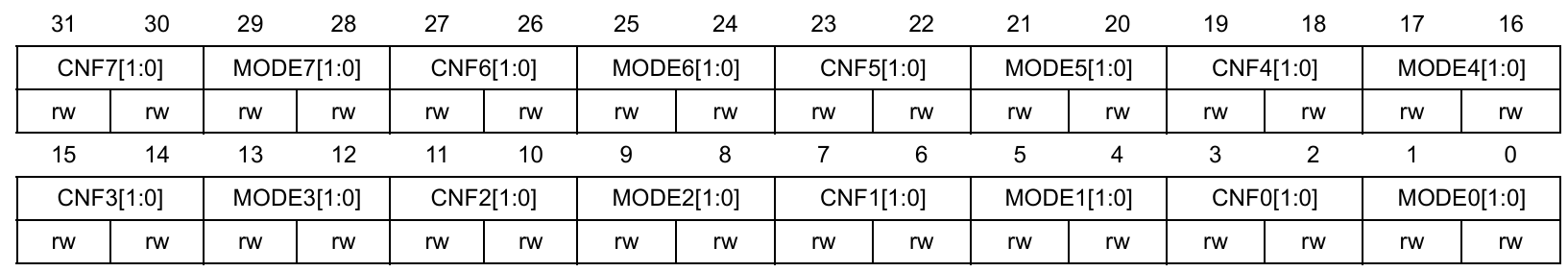
* 1. Thanh ghi **GPIOx\_CRL** (x=A..G) ; Address offset: 0x00
* Chức năng:

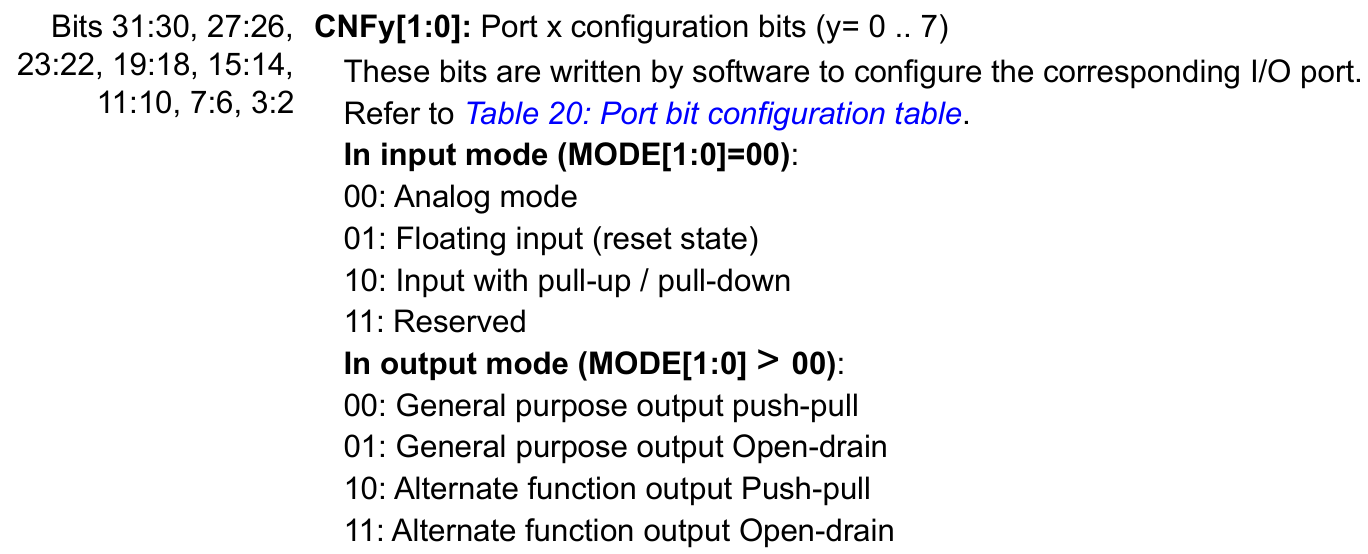
Thanh ghi **CRL** là viết tắt của **Port Configuration Register Low**. Nó dùng để cấu hình **8 chân thấp** của mỗi cổng GPIO, tức từ **Pin 0 → Pin 7**.

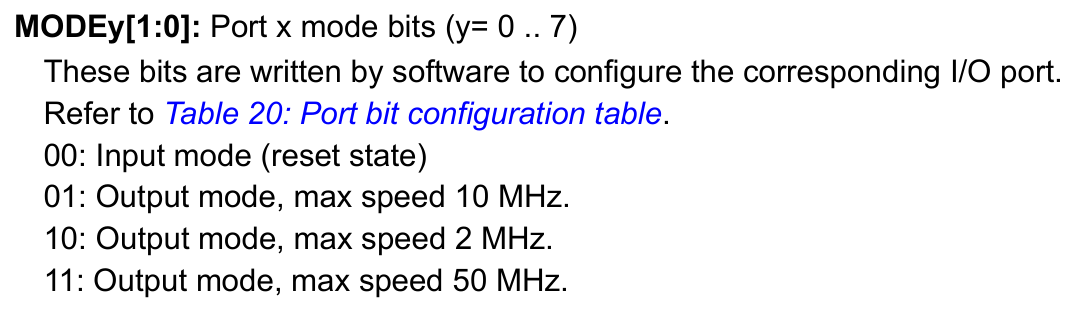
Mỗi chân sẽ được điều khiển bằng **4 bit** trong thanh ghi này, bao gồm:

**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.







* 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.  \*/  } |

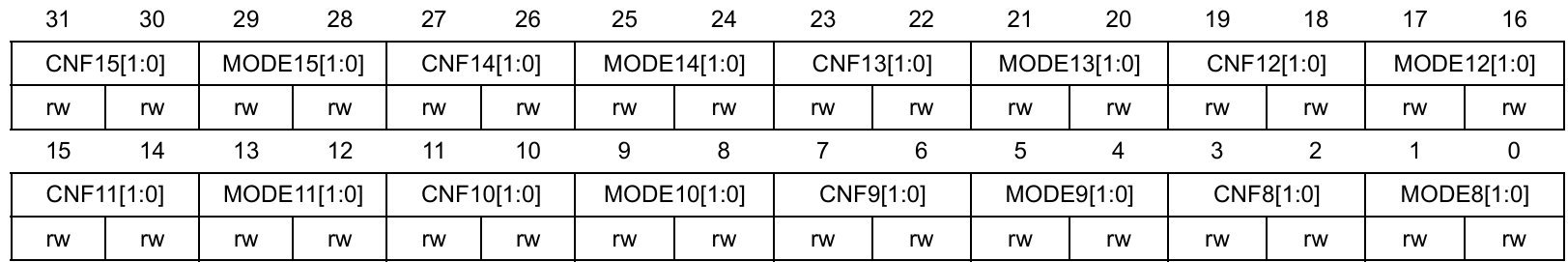
* 1. Thanh ghi **GPIOx\_CRH** (x=A..G) ; Address offset: 0x04

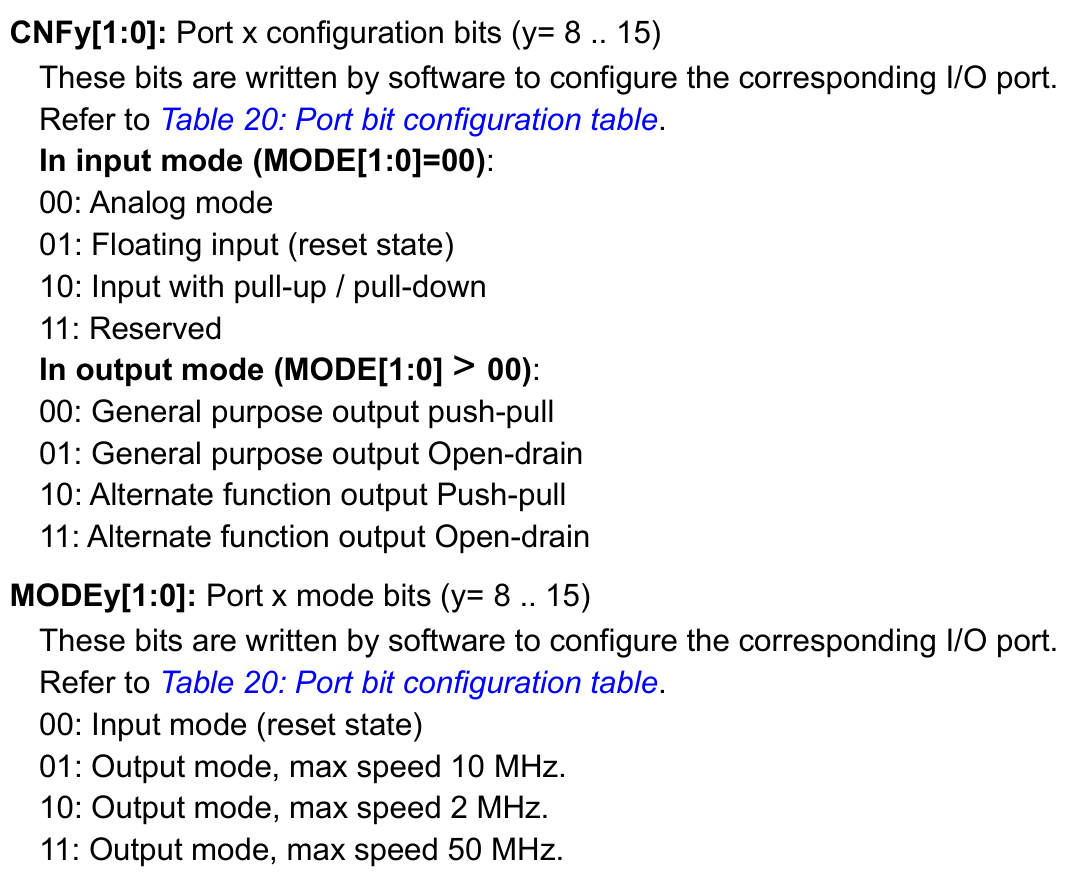
Chức năng: Thanh ghi **CRH** là viết tắt của **Port Configuration Register HIGH**. Nó dùng để cấu hình **8 chân cao** của mỗi cổng GPIO, tức từ **Pin 8 → Pin 15**.

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.





* 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.  \*/  } |

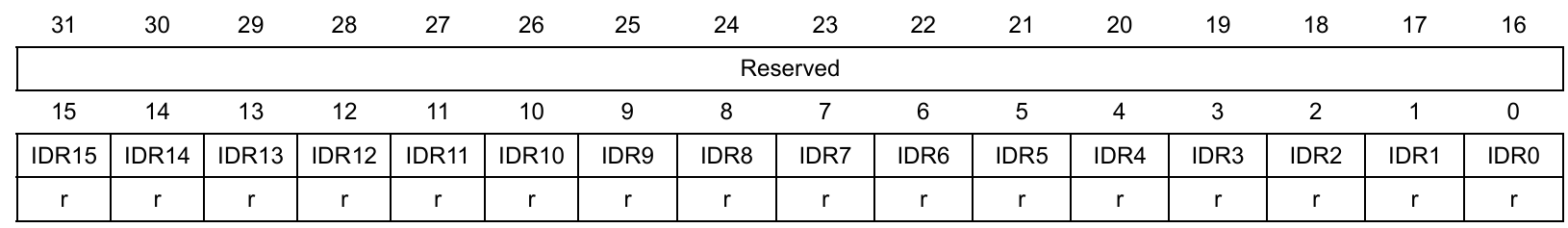
* 1. Thanh ghi **GPIOx\_IDR** (x=A..G) ; Address offset: 0x08

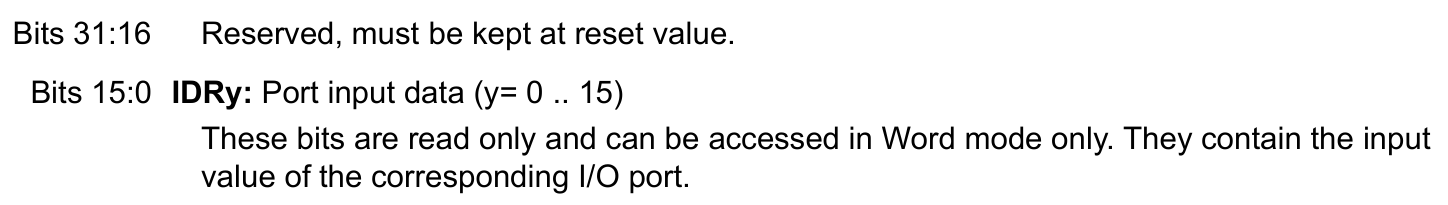
Chức năng: IDR là viết tắt của Input Data Register – tức thanh ghi dữ liệu ngõ vào.

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).



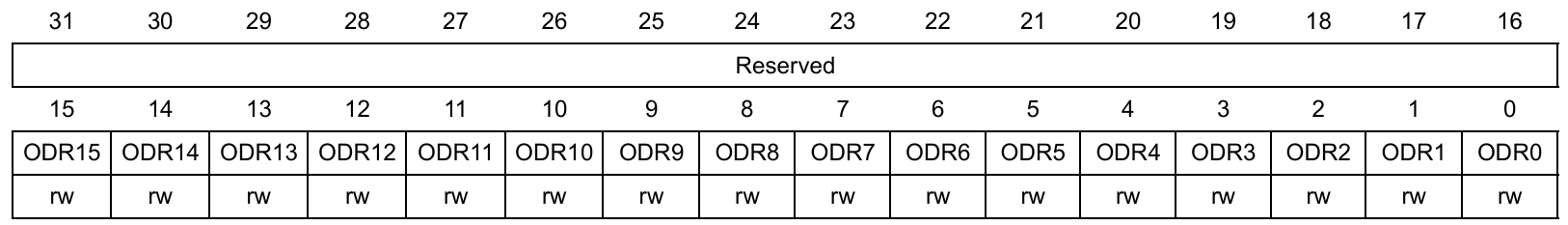


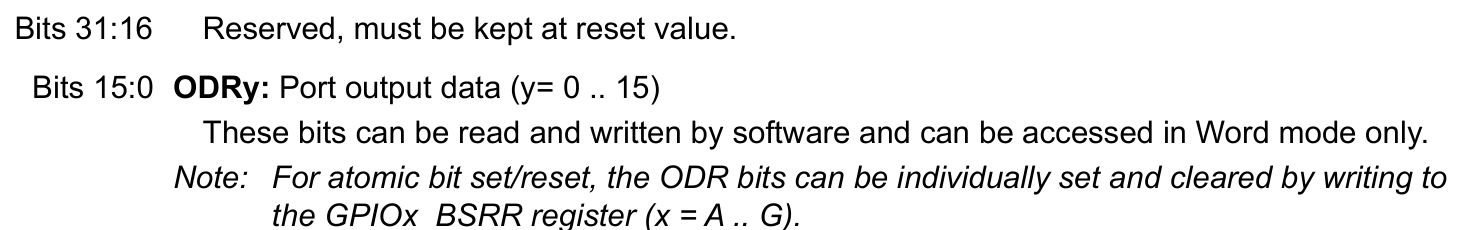
* Đoạn chương trình thao tác lên thanh ghi: (ghi lệnh và giải thích ý nghĩa)

|  |
| --- |
| #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. 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.





* 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)  /\* 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  } |

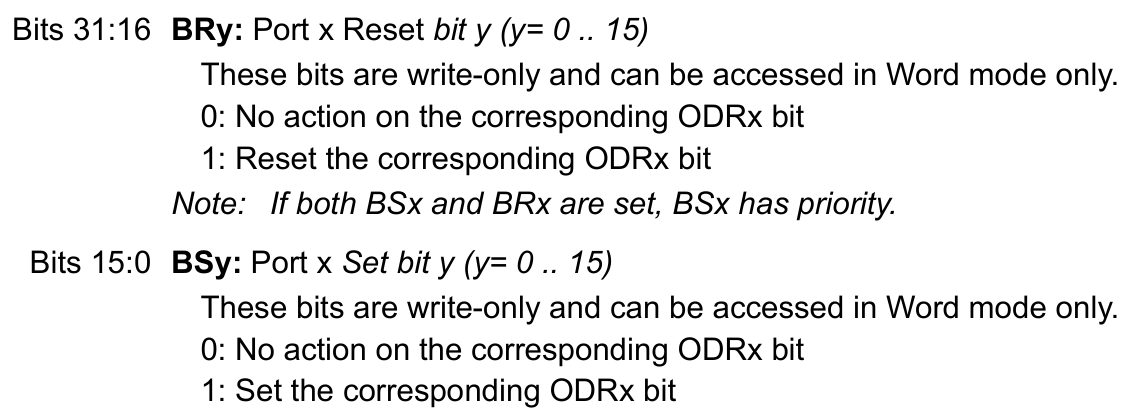
* 1. Thanh ghi **GPIOx\_BSRR** (x=A..G) ; Address offset: 0x10

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.



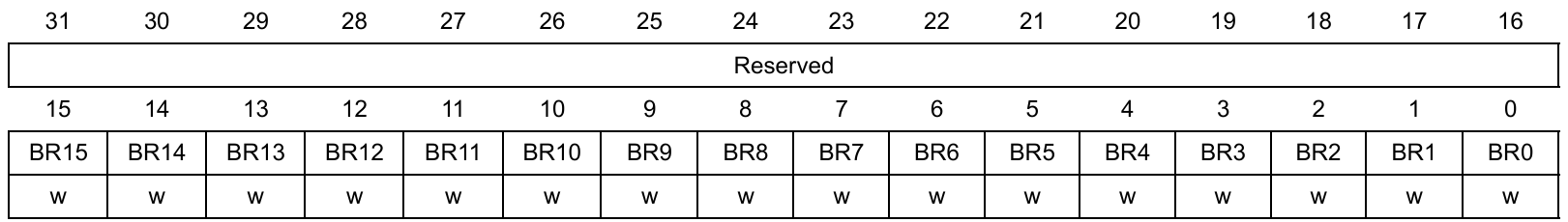


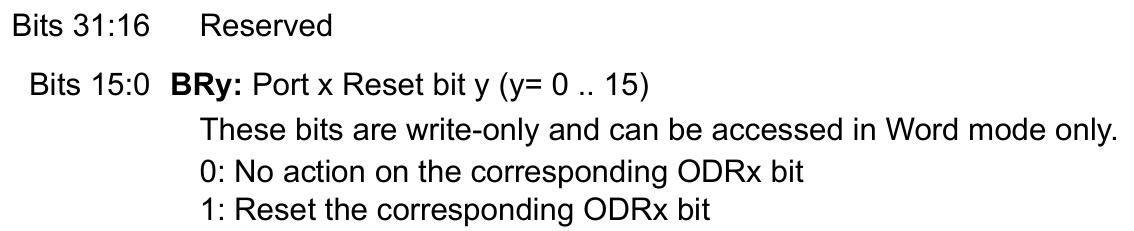
* Đoạn chương trình thao tác lên thanh ghi: (ghi lệnh và giải thích ý nghĩa)

|  |
| --- |
| #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. 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.

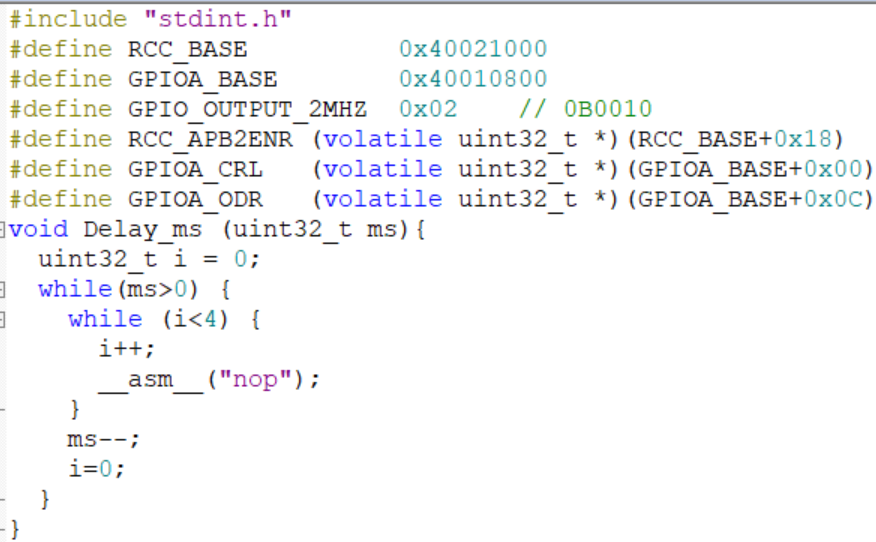


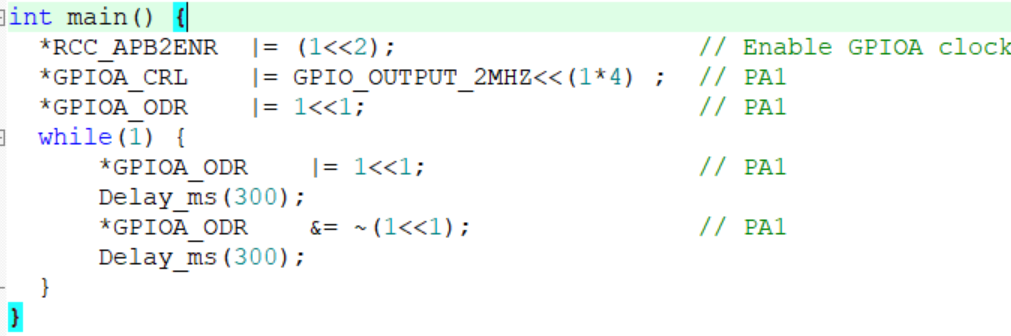


* Đoạn chương trình thao tác lên thanh ghi: (ghi lệnh và giải thích ý nghĩa)

|  |
| --- |
| #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  } |

1. Các chương trình mẫu:
   1. Chương trình bật/tắt PA1 theo chu kỳ 300ms:





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

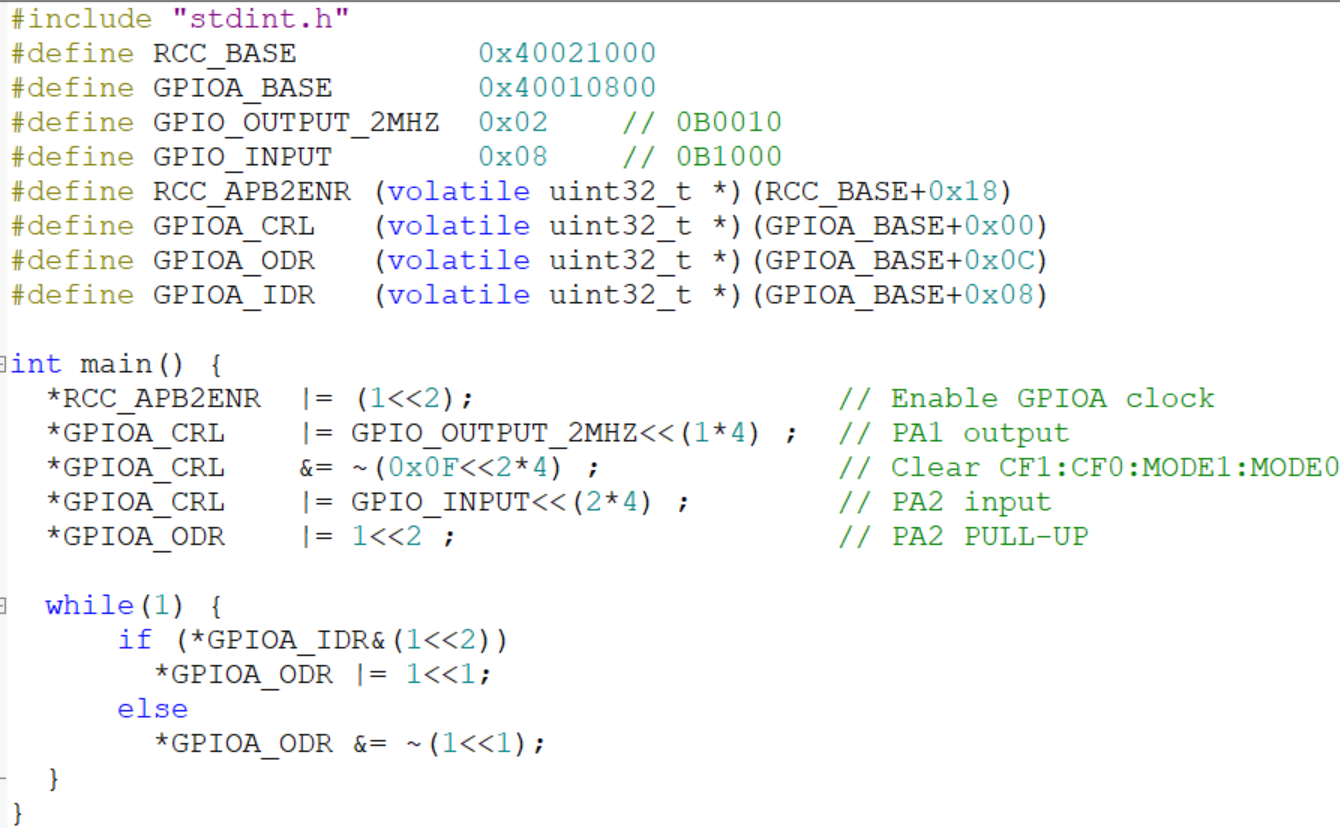
**\*GPIOA\_CRL &= ~(0x0F << (4\*1))**; Xóa 4 bit của PA1 (CNF[1:0], MODE[1:0])

**\*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

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



**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 ghiCRL (Configuration Register Low) của GPIOA. Offset 0x00 từ GPIOA\_BASE

**GPIOA\_ODR:** Con trỏ đến thanh ghiODR (Output Data Register) của GPIOA. Offset 0x08 từ GPIOA\_BASE

**GPIOA\_IDR**: Con trỏ đến thanh ghiIDR (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.

1. **Các bài thực hành:**
   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;      }  } |

* 1. 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        }  } |

* 1. 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--;      }  }  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      }  } |

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

|  |
| --- |
| #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;        }      }  } |