Chương 4

GPIO VÀ AFIO

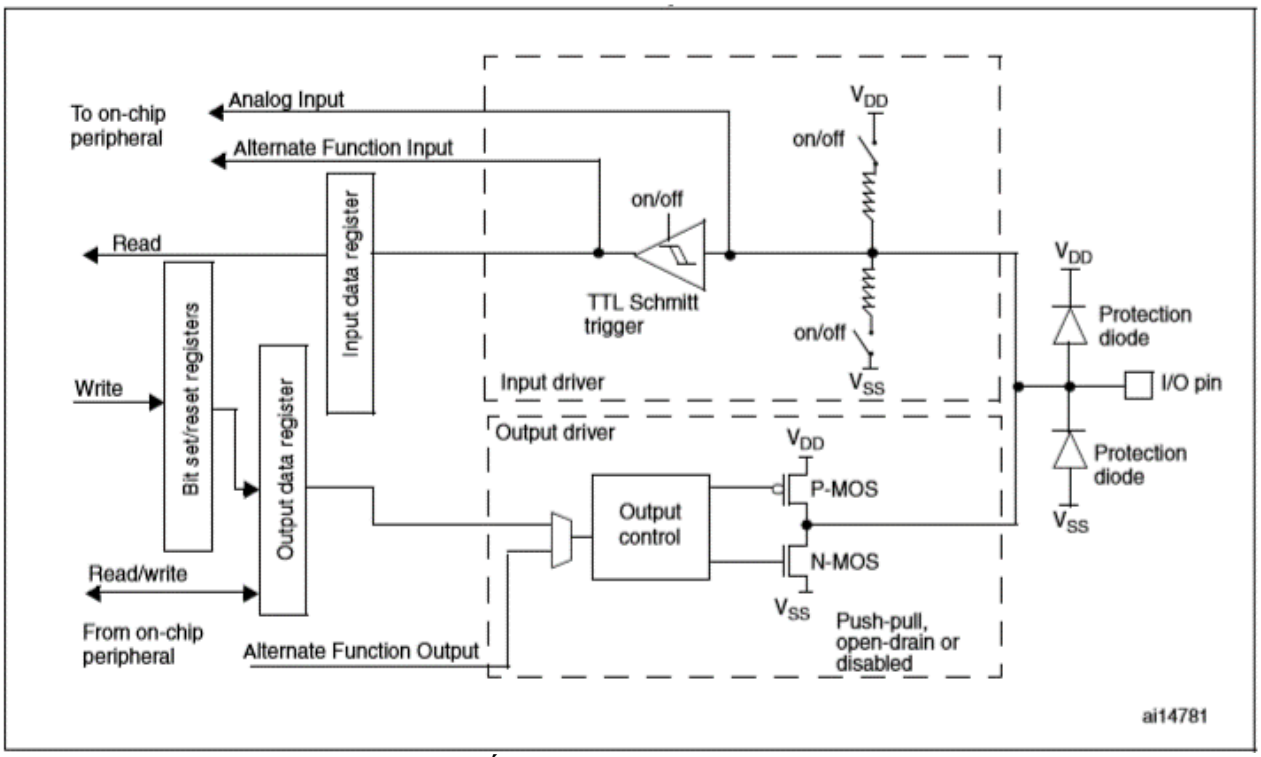
GENERAL-PURPOSE AND ALTERNATE-FUNCTION I/O

4.1 GIỚI THIỆU GPIO

GPIO (General-purpose input/output) : ngõ vào -ngõ ra sử dụng chung.

- Đối với các dòng STM32 thì mỗi PORT có 16 chân I/O.
- Ngoài chức năng là I/O nếu muốn sử dụng các chân này làm ngõ vào hoặc ngõ ra của ngoại vi ta phải thiết lập chân này là chức năng thay thế (Alternate Function) hoặc Analog.
- Để tiện cho thiết kế phần cứng nhà sản xuất xây dựng chức năng remap cho phép người dùng có thể thay đổi vị trí I/O của ngoại vi trong một phạm vi nhất định .
- Mỗi chân I/O bên trong chip đều được gắn điện trở nội kéo lên và kéo xuống.

4.2 CẤU TRÚC CƠ BẢN CỦA 1 CHÂN I/O



Hình 4.1 Cấu trúc cơ bản của 1 chân I/O

4.3 CÁC THANH GHI VÀ LỆNH LIÊN QUAN ĐẾN GPIO

Mỗi GPIO có các thanh ghi sau:

4.3.1 Hai thanh ghi 32 bit GPIOx_CRL và GPIOx_CRH

a. GPIOx_CRL (**G**eneral **P**urpose **I/O** **C**onfiguration **R**egister **L**ow)

(Địa chỉ **0x00h**, giá trị mặc định sau khi reset **0x4444 4444h**)

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

Hình 4.2 Thanh ghi GPIOx_CRL

Thanh ghi GPIOx_CRL dùng để cấu hình các chân từ 0 đến 7 của PORTx (4 bit cấu hình cho 1 chân). Chân thứ **"n"** có thể được cấu hình bằng cách thay đổi giá trị 2 bit CNF**n**[1:0] (chỉnh chế độ) và 2 bit MODE**n**[1:0] (chỉnh tốc độ) thành các dạng theo **bảng 4.1**

Bảng 4.1 Các chế độ hoạt động của GPIO và cách cấu hình tương ứng

CONFIGURATION MODE		CNF1	CNF0	MODE1	MODE0	PxODR Register
General purpose output	Push-pull	0	0	01 : 10 MHz 10 : 2 MHz 11 : 50 MHz		0 or 1
	Open-drain		1			0 or 1
Alternate Function output	Push-pull	1	0			X
	Open-drain		1			X
Input	Analog input	0	0	0		X
	Input floating		1			X
	Input pull-down	1	0			0
	Input pull-up					1

b. GPIOx_CRH (**G**eneral **P**urpose **I/O C**onfiguration **R**egister **H**igh)

(Địa chỉ **0x04h**, giá trị mặc định sau khi reset **0x4444 4444h**)

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

Hình 4.3 Thanh ghi GPIOx_CRH

Thanh ghi GPIOx_CRH dùng để cấu hình các chân từ **8** đến **15** của PORTx và cách cấu hình tương tự như GPIOx_CRL được trình bày ở **bảng 4.1.**

c. Các lệnh thông dụng liên quan đến GPIOx_CRL và GPIOx_CRH

Bảng 4.2 Các lệnh thông dụng để cấu hình GPIO

SỬ DỤNG THƯ VIỆN "stm32f10x_gpio"	
Lệnh	
Thông số hay dùng	**Giải thích**
GPIO_InitTypeDef A; (Khai báo biết A thuộc kiểu GPIO_InitTypeDef)	
A.GPIO_Pin = B; (Lệnh chọn chân cần cấu hình)	
B: GPIO_Pin_0 GPIO_Pin_1 … GPIO_Pin_15 PIO_Pin_All	**B: Chọn chân cần cấu hình** Chọn chân 0 Chọn chân 1 … Chọn chân 15 Chọn tất cả các chân 0-15 **Chú ý:** Nếu muốn chọn nhiều chân thì ta sử dụng lệnh OR "|" giữa các lựa chọn **Ví dụ:** A.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_3;

A.GPIO_Mode =C; (Lệnh cấu hình chế độ hoạt động)	
C:	**C: Cấu hình chế độ hoạt động**
GPIO_Mode_AIN	Ngõ vào tương tự
GPIO_Mode_IN_FLOATING	Ngõ vào thả nổi
GPIO_Mode_IPD	Ngõ vào kéo xuống
GPIO_Mode_IPU	Ngõ vào kéo lên
GPIO_Mode_Out_OD	Ngõ ra cực thu hở
GPIO_Mode_Out_PP	Ngõ ra đẩy kéo
GPIO_Mode_AF_OD	Chức năng thay thế cực thu hở
GPIO_Mode_AF_PP	Chức năng thay thế đẩy kéo
A.GPIO_Speed =D; (Lệnh cấu hình tốc độ)	
D:	**D: Tốc độ ngõ ra**
GPIO_Speed_10MHz	Chọn tốc độ 10 Mhz
GPIO_Speed_2MHz	Chọn tốc độ 2 Mhz
GPIO_Speed_50MHz	Chọn tốc độ 50 Mhz
GPIO_Init(GPIOX,&A); (Lệnh cấu hình cho GPIOX theo các thông số được lưu trong biến A)	

d. Ví dụ về cách cấu hình GPIO bằng cách sử dụng thư viện "stm32f10x_gpio"

Ví dụ 4.1: Cấu hình chân A8 và chân A1 là ngõ ra đẩy kéo tốc độ 50 Mhz

Chương trình

```
GPIO_InitTypeDef    GPIO_Structure;
GPIO_Structure.GPIO_Pin = GPIO_Pin_8|GPIO_Pin_1;
 // Chọn chân số 8 và chân số 1
GPIO_Structure.GPIO_Mode = GPIO_Mode_Out_PP;
  // Chọn chế độ ngõ ra đẩy kéo
GPIO_Structure.GPIO_Speed = GPIO_Speed_50MHz;
  // Chọn tốc độ 50 Mhz
GPIO_Init(GPIOA,&GPIO_Structure );
  // Cài đặt các cấu hình trên cho PORTA
```

e. Các ví dụ về cách cấu hình GPIO khi không sử dụng thư viện "stm32f10x_gpio"

Ví dụ 4.2: Cấu hình chân A8 và chân A1 là ngõ ra đẩy kéo tốc độ 50 Mhz

Chương trình

```
    /*  Chân A8 được cấu hình bởi GPIOA_CRH
       Tốc độ 50 Mhz => MODE8[1:0]=11
       Ngõ ra đẩy kéo => CNF8[1:0]=00
       CNF8[1:0]MODE8[1:0] = 0011= 3
       Chân A1 được cấu hình bởi GPIOA_CRL
       Tốc độ 50 Mhz => MODE1[1:0]=11
       Ngõ ra đẩy kéo => CNF1[1:0]=00
       CNF1[1:0]MODE1[1:0] = 0011= 3 */
    GPIOA->CRL  = 0x00000030;     // Cấu hình cho chân A1
    GPIOA->CRH  = 0x00000003;     // Cấu hình cho chân A8
```

Ví dụ 4.3: Cấu hình tất cả các chân của PORTB là ngõ ra đẩy kéo tốc độ 50 Mhz

Chương trình

```
GPIOB->CRL = 0x33333333;
GPIOB->CRH = 0x33333333;
```

Ví dụ 4.4: Cấu hình tất cả các chân PORTB là ngõ vào thả nổi.

Chương trình

```
GPIOB->CRL = 0x44444444;
GPIOB->CRH = 0x44444444;
```

Chú ý : Ví dụ 4.3 và 4.4 được sử dụng rất nhiều trong lập trình để chuyển nhanh một PORT từ ngõ ra thành ngõ vào và ngược lại để tránh viết lệnh dài như thư viện "stm32f10x_gpio"

4.3.2 Hai thanh ghi 32 bit GPIOx_IDR và GPIOx_ODR

a. GPIOx_IDR (**G**eneral **P**urpose **I/O** **I**nput **D**ata **R**egister)

(Địa chỉ **0x08h**, giá trị mặc định sau khi reset **0x0000 XXXXh**)

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**

Hình 4.4 Thanh ghi GPIOx_IDR

Thanh ghi GPIOx_IDR chứa trạng thái logic đọc về từ các chân của PORTx.

b. GPIOx_ODR (**G**eneral **P**urpose **I/O** **O**utput **D**ata **R**egister)

(Địa chỉ **0x0Ch**, giá trị mặc định sau khi reset **0x0000 0000h**)

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
0DR15	**0DR14**	**0DR13**	**0DR12**	**0DR11**	**0DR10**	**0DR9**	**0DR8**	**0DR7**	**0DR6**	**0DR5**	**0DR4**	**0DR3**	**0DR2**	**0DR1**	**0DR0**
rw	**rw**	**rw**	**rw**	**rw**	**rw**	**rw**	**rw**	**rw**	**rw**	**rw**	**rw**	**rw**	**rw**	**rw**	**rw**

Hình 4.5 Thanh ghi GPIOx_IDR

Thanh ghi GPIOx_ODR chứa trạng thái logic xuất ra các chân của PORTx.

c. Các lệnh thường dùng liên quan đến GPIOx_IDR và GPIOx_ODR

Bảng 4.3 Các lệnh trong thư viện "stm32f10x_gpio" dùng để truy xuất PORT

SỬ DỤNG THƯ VIỆN "stm32f10x_gpio"	
Lệnh	
Thông số hay dùng	**Giải thích**
unsigned short A; **A=GPIO_ReadInputData(B);**	(Lệnh đọc trạng thái logic 16 chân của 1 PORT rồi lưu vào biến A)

B: GPIOA GPIOB … GPIOG	**B:** PORT cần đọc dữ liệu Đọc PORTA Đọc PORTB … Đọc PORTG **A:** Biến 16 bit để lưu trạng thái logic 16 chân của PORT được đọc.
GPIO_Write(A, B); (Lệnh xuất dữ liệu ra 16 chân của PORT)	
A: GPIOA GPIOB … GPIOG	**A:** PORT cần xuất dữ liệu Xuất ra PORTA Xuất ra PORTB … Xuất ra PORTG **B:** Số nguyên 16 bit ứng với trạng thái 16 chân của PORT cần xuất.

Bảng 4.4 Các lệnh dùng để truy xuất PORT khi không dùng thư viện"stm32f10x_gpio"

KHÔNG SỬ DỤNG THƯ VIỆN "stm32f10x_gpio"	
Lệnh	
Thông số hay dùng	**Giải thích**
unsigned short A; **A= P->IDR** (Lệnh đọc trạng thái logic 16 chân của 1 PORT rồi lưu vào biến A)	
P: GPIOA GPIOB … GPIOG	**P:** PORT cần đọc dữ liệu Đọc PORTA Đọc PORTB … Đọc PORTG **A:** Biến 16 bit để lưu trạng thái logic 16 chân của PORT được đọc.
P->ODR =A (Lệnh xuất dữ liệu ra 16 chân của PORT)	
P: GPIOA GPIOB … GPIOG	**P:** PORT cần xuất dữ liệu Xuất ra PORTA Xuất ra PORTB … Xuất ra PORTG **A:** Số nguyên 16 bit ứng với trạng thái 16 chân của PORT cần xuất.

d. Các ví dụ về truy xuất dữ liệu qua PORT khi không dùng thư viện" stm32f10x_gpio"

Ví dụ 4.4: Viết chương trình đọc dữ liệu về từ **PORTD** theo 2 cách là sử dụng thư viện" stm32f10x_gpio" và không sử dụng thư viện.

Chương trình

```
unsigned short   A;
// Khai báo biến A kiểu số nguyên 16 bit để lưu giá trị đọc về
```

```
//Dùng thư viện của ST :
A=GPIO_ReadInputData(GPIOD);
//Không dùng thư  viện của ST:
A= GPIOD->IDR
```

Ví dụ 4.5: Viết chương trình điều khiển 8 chân từ 0-7 của **PORTC** ở mức '1' và 8 chân còn lại mức '0' theo 2 cách là sử dụng thư viện" stm32f10x_gpio" và không sử dụng thư viện.

Chương trình

```
unsigned short   A;
// Khai báo biến A kiểu số nguyên 16 bit để lưu giá trị đọc về
//Dùng thư viện của ST :
GPIO_Write(GPIOC, 0x00FF);
// 0x00FFH = 0000 0000 1111 1111B
//Không dùng thư  viện của ST:
GPIOC->ODR =0x00FF;
```

4.3.3 Hai thanh ghi GPIOx_BSRR và GPIOx_BRR

a. GPIOx_BSRR (**G**eneral **P**urpose **I/O B**it **S**et **R**eset **R**egister)

(Địa chỉ **0x10h**, giá trị mặc định sau khi reset **0x0000 0000h**)

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

Hình 4.6 Thanh ghi GPIOx_BSRR

Thanh ghi GPIOx_BSRR được dùng để xuất mức '1'(điều khiển các bit từ 0 đến 15) hoặc xuất mức "0-reset" (điều khiển các bit từ 16 đến 31) ra các chân của PORTx.

Ví dụ điều khiển chân số 2 và chân số 5 của PORTx xuất mức '1', chân số 0 và chân số 12 của PORTx xuất mức '0'. Ta phân tích yêu cầu như sau:

Do yêu cầu chân số 2 và chân số 5 xuất mức '1'nên tại vị trí bit số 2 (ứng với BS2 – set chân 2) và bit số 5 (ứng với BS5 – set chân 5) của thanh ghi GPIOx_BSRR ta điền số 1, mặt khác chân số 0 và chân 12 xuất mức '0'nên tại vị trí bit số 16 (ứng với BR0 – reset chân 0) và bit thứ 28 (ứng với BR12 – reset chân 12) ta cũng điền số 1(số 1 là cho phép chứ không phải là mức '1'). Các vị trí còn lại ta điền số 0.

b. GPIOx_BRR (**G**eneral **P**urpose **I/O B**it **R**eset **R**egister)

(Địa chỉ **0x14h**, giá trị mặc định sau khi reset **0x0000 0000h**)

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

Hình 4.7 Thanh ghi GPIOx_BRR

Thanh ghi GPIOx_BRR được dùng để xuất mức '0' ra các chân của PORTx . Trong lập trình để xuất mức '0' người ta thường hay dùng thanh ghi này ít dùng thanh ghi GPIOx_BSRR.

Cách sử dụng thanh ghi GPIOx_BRR cũng tương tự như GPIOx_BSRR.

c. Các lệnh thường dùng liên quan đến GPIOx_BRR và GPIOx_BSRR

Bảng 4.5 Các lệnh trong thư viện“ stm32f10x_gpio”dùng để truy xuất từng chân của PORT

SỬ DỤNG THƯ VIỆN “stm32f10x_gpio”	
Lệnh	
Thông số hay dùng	**Giải thích**
GPIO_SetBits(A, B); (Lệnh xuất mức ‘1’ ra 1 chân vi điều khiển)	
A: GPIOA … GPIOG **B:** GPIO_Pin_0 … GPIO_Pin_15 PIO_Pin_All	**A:** PORT cần xuất Xuất ra PORTA … Xuất ra PORTG **B:** Chân cần xuất Chân số 0 … Chân số 15 Tất cả các chân của PORT
GPIO_ResetBits(A, B); (Lệnh xuất mức ‘0’ ra 1 chân vi điều khiển)	
A: GPIOA … GPIOG **B:** GPIO_Pin_0 … GPIO_Pin_15 GPIO_Pin_All	**A:** PORT cần xuất Xuất ra PORTA … Xuất ra PORTG **B:** Chân cần xuất Chân số 0 … Chân số 15 Tất cả các chân của PORT
BitAction C; **GPIO_WriteBit(A, B, C);** (Lệnh xuất mức logic của biến C ra 1 chân vi điều khiển)	
A: GPIOA … GPIOG **B:** GPIO_Pin_0 … GPIO_Pin_15 GPIO_Pin_All **C:** Bit_RESET Bit_SET	**A:** PORT cần xuất Xuất ra PORTA … Xuất ra PORTG **B:** Chân cần xuất Chân số 0 … Chân số 15 Tất cả các chân của PORT **C**: Mức logic cần xuất Mức ‘0’ Mức ‘1’

Bảng 4.6 Các lệnh dùng để truy xuất từng chân của PORT mà không dùng thư viện GPIO

KHÔNG SỬ DỤNG THƯ VIỆN “stm32f10x_gpio”
Lệnh

Thông số hay dùng	Giải thích
P->BSRR =A ; (Lệnh xuất mức ‘1’ hoặc ‘0’ ra chân vi điều khiển)	
P: GPIOA … GPIOG **A:** Số 32 bit	**P:** PORT cần xuất Xuất ra PORTA … Xuất ra PORTG **A:** Chân cần xuất **Ví dụ:** chân số 5 và 10 của PORTC là mức ‘1’ **A** = 0000 0**1**00 00**1**0 0000B **A** = 0x00000420 ⇨ GPIOC->BSRR =0x00000420 ;
P->BRR =A ; (Lệnh xuất mức ‘0’ ra chân vi điều khiển)	
P: GPIOA … GPIOG **A:** [0-65535]	**P:** PORT cần xuất Xuất ra PORTA … Xuất ra PORTG **A:** Chân cần xuất **Ví dụ:** chân số 5 và 10 của PORTC là mức ‘0’ **A** = 0000 0**1**00 00**1**0 0000B = 0x0420 GPIOC->BRR =0x0420 ;

d. Các ví dụ liên quan đến GPIOx_BRR và GPIOx_BSRR

Ví dụ 4.6: Viết chương trình xuất mức ‘1’ ra chân số 5 và chân số 10 PORTD theo 2 cách là sử dụng thư viện“ stm32f10x_gpio” và không sử dụng thư viện.

Chương trình

```
//Dùng thư viện của ST :
GPIO_SetBits(GPIOD, GPIO_Pin_5| GPIO_Pin_10);
//Không dùng thư viện ST :
GPIOD-> BSRR = 0x0420 ;
```

Ví dụ 4.7: Viết chương trình xuất mức ‘0’ ra chân số 5 và chân số 10 PORTD theo 2 cách là sử dụng thư viện“ stm32f10x_gpio” và không sử dụng thư viện.

Chương trình

```
//Dùng thư viện của ST :
GPIO_ResetBits(GPIOD, GPIO_Pin_5| GPIO_Pin_10);
//Không dùng thư viện ST :
GPIOD-> BRR = 0x0420 ;
//Sử dụng kết hợp cả thư viện cả trực tiếp:
GPIOC->BRR = GPIO_Pin_5|GPIO_Pin_10;
```

4.3.4 Thanh ghi GPIOx_LCKR(General Purpose I/O LoCK Register)

(Địa chỉ **0x18h**, giá trị mặc định sau khi reset **0x0000 0000h**)

Thanh ghi GPIOx_LCKR dùng để khóa cấu hình các chân của PORTx (đóng băng 4 bit cấu hình tương ứng của thanh ghi GPIOx_CRL và GPIOx_CRH) cho đến khi MCU bị reset. Bit

số 16 của thanh ghi GPIOx_LCKR điều khiển việc cho phép hoặc không cho phép khóa cấu hình. Các bit từ 0 đến 15 điều khiển việc cho phép hoặc không cho phép khóa cấu hình tương ứng với các chân từ 0 đến 16 của PORTX.

4.4 CHỨC NĂNG THAY THẾ VÀ THAY ĐỔI VỊ TRÍ CÁC I/O (Alternate function and remap I/O)

Do mỗi chân vi điều khiển có nhiều chức năng nên khi muốn sử dụng chức năng khác với mặc định thì ta phải cấu hình cho chân đó là chức năng thay thế(Alternate Function) hoặc Analog.

Khi được thiết lập là là chức năng thay thế, các chân này được gọi là AFIO(**A**lternate **F**unction **I/O**). Để các AFIO có thể hoạt động được ta cần phải cấp xung clock bằng lệnh sau:

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE);
```

Để tiện cho thiết kế phần cứng nhà thiết kế cho phép ta có thể thay đổi vị trí các ngõ vào ra của ngoại vi trong một phạm vi nhất định, thao tác này gọi là "**Remap**".

4.4.1 Sử dụng các chân đặc biệt làm GPIO.

a. Sử dụng hai chân OSC32_IN(PC14)/OSC32_OUT(PC15)

(Hai chân này được dùng để nối thạch anh ngoài 32.768 Khz cho LSE)

Hai chân OSC32_IN và OSC32_OUT có thể được dùng như là các chân GPIO thông thường nếu ta tắt LSE. Khi bật LSE thì hai chân này trở thành chân để kết nối thạch anh 32.768 Khz.

b.Sử dụng hai chân OSC_IN(PD0)/OSC_OUT(PD1)

(Hai chân này được dùng để kết nối thạch anh ngoài – thường chọn 8 Mhz- cho HSE)

Việc tận dụng hai chân OSC_IN(PD0)/OSC_OUT(PD1) làm GPIO chỉ có thể thực hiện được trên các dòng ARM ít chân (36, 48, 64 chân) còn các dòng nhiều chân hơn thì hai chân PD0 và PD1 nằm tách biệt so với OSC_IN và OSC_OUT nên không cần remap.

Chú ý: Khi remap hai chân OSC_IN và OSC_OUT thành GPIO thì chúng không có khả năng tạo ngắt ngoài như các GPIO khác.

Đối với các dòng ARM ít chân hai chân PD0 và PD1 mặc định là ngõ kết nối thạch anh để remap chúng thành GPIO ta dùng lệnh sau:

```
GPIO_PinRemapConfig(GPIO_Remap_PD01, ENABLE);
```

c. Sử dụng các chân JTDI(PA15), JTDO(PB3) và JNTRST(PB4)

(Các chân này kết nối với mạch nạp JTAG để nạp và debug chương trình)

Mặc định các chân này được dùng làm chân để nạp chương trình và debug qua cổng JTAG nhưng nếu như ta nạp và debug theo chuẩn SWD(Serial Wire Debug) thì ta có thể tận dụng các chân này làm GPIO bằng lệnh sau:

```
GPIO_PinRemapConfig(GPIO_Remap_SWJ_JTAGDisable, ENABLE);
```

Chú ý : Sau khi remap các chân này thành GPIO thì khi nạp chương trình ta cần phải chuyển chế độ nạp sang SWD vì lúc này chuẩn JTAG không còn nạp được nữa.

4.4.2 Remap các ngoại vi thông dụng

a. CAN1

TÊN CHÂN	VỊ TRÍ MẶC ĐỊNH	REMAP: GPIO_PinRemapConfig(A , ENABLE);	
		A=GPIO_Remap1_CAN1	A = GPIO_Remap2_CAN1
CAN_RX	PA11	PB8	PD0
CAN_TX	PA12	PB9	PD1

Chú ý: Dòng ARM 36 chân không **Remap1_CAN1** được. Chỉ có dòng ARM 100 chân và 144 chân mới có **Remap2_CAN1** khi mà các chân PD0 và PD1 không dùng để kết nối thạch anh.

b. Timer 5

TÊN CHÂN	VỊ TRÍ MẶC ĐỊNH	REMAP: GPIO_PinRemapConfig(A , ENABLE);
		A= GPIO_Remap_TIM5CH4_LSI
TIM5_CH4	A3	TIM5_CH4 nối với LSI 40 Khz

Chú ý: Remap_TIM5CH4_LSI chỉ hỗ trợ dòng high-density

c. Timer 4

TÊN CHÂN	VỊ TRÍ MẶC ĐỊNH	REMAP: GPIO_PinRemapConfig(A , ENABLE);
		A= GPIO_Remap_TIM4
TIM4_CH1	PB6	PD12
TIM4_CH2	PB7	PD13
TIM4_CH3	PB8	PD13
TIM4_CH4	PB9	PD13

Chú ý : Chỉ những dòng ARM 100,144 chân mới hỗ trợ **Remap_TIM4**

d. Timer 3

TÊN CHÂN	VỊ TRÍ MẶC ĐỊNH	REMAP: GPIO_PinRemapConfig(A , ENABLE);	
		A=GPIO_PartialRemap_TIM3	A= GPIO_FullRemap_TIM3
TIM3_CH1	PA6	PB4	PC6
TIM3_CH2	PA7	PB5	PC7
TIM3_CH3	PB0		PC8
TIM3_CH4	PB1		PC9

Chú ý : Chỉ những dòng ARM 64 chân trở lên mới hỗ trợ **FullRemap_TIM3**

e. Timer 2

TÊN CHÂN	VỊ TRÍ MẶC ĐỊNH	REMAP: GPIO_PinRemapConfig(A , ENABLE);		
		A= GPIO_PartialRemap1_TIM2	A= GPIO_PartialRemap2_TIM2	A= GPIO_FullRemap _TIM2
TIM2_CH1	PA0	PA15	PA0	PA15
TIM2_CH2	PA1	PB3	PA1	PB3
TIM2_CH3	PA2		PB10	
TIM2_CH4	PA3		PB11	

Chú ý: Dòng ARM 36 chân không Remap này.

TIM2_CH1 và TIM2_ETR dùng chung một chân nên không thể sài cả hai cùng lúc.

f. Timer 1

TÊN CHÂN	VỊ TRÍ MẶC ĐỊNH	REMAP: GPIO_PinRemapConfig(A , ENABLE);	
		A=GPIO_PartialRemap_TIM1	A=GPIO_FullRemap_TIM1
TIM1_ETR	PA12		PE7
TIM1_CH1	PA8		PE9
TIM1_CH2	PA9		PE11
TIM1_CH3	PA10		PE13
TIM1_CH4	PA11		PE14
TIM1_BKIN	PB12	PA6	PE15
TIM1_CH1N	PB13	PA7	PE8
TIM1_CH2N	PB14	PB0	PE10
TIM1_CH3N	PB15	PB1	PE12

Chú ý: Dòng ARM 36 chân không hỗ trợ remap này.
Chỉ dòng ARM 100, 144 chân mới hỗ trợ **FullRemap_TIM1**

g. USART3

TÊN CHÂN	VỊ TRÍ MẶC ĐỊNH	REMAP: GPIO_PinRemapConfig(A , ENABLE);	
		A= GPIO_PartialRemap_USART3	A= GPIO_FullRemap_USART3
USART3_TX	PB10	PC10	PD8
USART3_RX	PB11	PC11	PD9
USART3_CK	PB12	PC12	PD10
USART3_CTS	PB13		PD11
USART3_RTS	PB14		PD12

Chú ý: Remap này chỉ hỗ trợ ARM từ 64 chân trở lên.
FullRemap_USART3 chỉ hỗ trợ ARM 100, 144 chân.

h. USART2

TÊN CHÂN	VỊ TRÍ MẶC ĐỊNH	REMAP: GPIO_PinRemapConfig(A , ENABLE);
		A= GPIO_Remap_USART2
USART2_CTS	PA0	PD3
USART2_RTS	PA1	PD4
USART2_TX	PA2	PD5
USART2_RX	PA3	PD6
USART2_CK	PA4	PD7

Chú ý: Chỉ những dòng ARM 100, 144 chân mới hỗ trợ **Remap_USART2**

i. USART1

TÊN CHÂN	VỊ TRÍ MẶC ĐỊNH	REMAP: GPIO_PinRemapConfig(A , ENABLE);
		A= GPIO_Remap_USART1
USART1_TX	PA9	PB6
USART1_RX	PA10	PB7

j. I2C 1

TÊN CHÂN	VỊ TRÍ MẶC ĐỊNH	REMAP: GPIO_PinRemapConfig(A , ENABLE);
		A= GPIO_Remap_I2C1
I2C1_SCL	PB6	PB8
I2C1_SDA	PB7	PB9

Chú ý: ARM 36 chân không hỗ trợ Remap này

k. SPI 1

TÊN CHÂN	VỊ TRÍ MẶC ĐỊNH	REMAP: GPIO_PinRemapConfig(A , ENABLE);
		A= GPIO_Remap_SPI1
SPI1_NSS	PA4	PA15
SPI1_SCK	PA5	PB3
SPI1_MISO	PA6	PB4
SPI1_MOSI	PA7	PB5

4.5 Các ví dụ về GPIO và AFIO

4.5.1 Ví dụ xuất dữ liệu ra 1 chân GPIO

Ví dụ 4.8: Viết chương trình điều khiển LED đơn kết nối với chân D9 nhấp nháy.

Chương trình

```
#include<stm32f10x.h>
void delay(unsigned long t){while(t--);}
     // Chương trình con delay
void CauHinhChanLED()
{    // Chương trình con cấu hình chân D9 điều khiển LED là  //
    ngõ ra đẩy kéo tốc độ 50Mhz
     GPIO_InitTypeDef    GPIO_Structure;
     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);
     // Vì chân D9 thuộc PORTD nên cần phải cho phép cấp
     // xung PORTD ( lý thuyết bài RCC)
     GPIO_Structure.GPIO_Pin     = GPIO_Pin_9;
     GPIO_Structure.GPIO_Mode    = GPIO_Mode_Out_PP;
     GPIO_Structure.GPIO_Speed   = GPIO_Speed_50MHz;
     GPIO_Init(GPIOD,&GPIO_Structure );
}
int main(void)
{
    SystemInit();
    // Cấu hình xung Clock
    CauHinhChanLED();
    while(1)
        {
         GPIO_SetBits(GPIOD,GPIO_Pin_9);    // Chân D9 = 1
         delay(1000000);
         GPIO_ResetBits(GPIOD,GPIO_Pin_9); // Chân D9 = 0
         delay(1000000);
        }
}
```

Chú ý: Do trong chương trình có sử dụng các hàm của thư viện **"stm32f10x_gpio.c"** và **"stm32f10x_rcc.c"** nên ta cần thêm hai thư viện này vào Project.

4.5.2 Ví dụ xuất, nhập dữ liệu qua chân GPIO

Ví dụ 4.9: Viết chương trình điều khiển LED đơn kết nối với chân D9 sáng hoặc tắt nhờ hai nút nhấn ON và OFF. Biết nút ON được nối với chân C13 và nút OFF nối với chân A8.

Chương trình

```
#include<stm32f10x.h>
void delay(unsigned long t){while(t--);}
void CauHinh_Button_LED()
{     GPIO_InitTypeDef      GPIO_Structure;
      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD
                             |RCC_APB2Periph_GPIOA
                             |RCC_APB2Periph_GPIOC, ENABLE);
      // Các chân điều khiển nằm ở 3 PORT A,C,D nên cho phép
       //cấp xung cả 3 PORT này
      GPIO_Structure.GPIO_Pin   = GPIO_Pin_9;
      GPIO_Structure.GPIO_Mode  = GPIO_Mode_Out_PP;
      GPIO_Structure.GPIO_Speed = GPIO_Speed_50MHz;
      GPIO_Init(GPIOD,&GPIO_Structure );
      // Chân D9 là ngõ ra đẩy kéo tốc độ 50Mhz
      GPIO_Structure.GPIO_Pin  = GPIO_Pin_8;
      GPIO_Structure.GPIO_Mode = GPIO_Mode_IPU;
      GPIO_Init(GPIOA,&GPIO_Structure );
      // Chân A8 là ngõ vào kéo lên
      GPIO_Structure.GPIO_Pin = GPIO_Pin_13;
      GPIO_Init(GPIOC,&GPIO_Structure );
      // Chân C13 là ngõ vào kéo lên
}
int main(void)
{    SystemInit();              // Cấu hình xung clock
     CauHinh_Button_LED();
     while(1)
           {
             if( (GPIO_ReadInputData(GPIOA)&GPIO_Pin_8) ==0)
                  GPIO_ResetBits(GPIOD,GPIO_Pin_9);
             if( (GPIO_ReadInputData(GPIOC)&GPIO_Pin_13) ==0)
                  GPIO_SetBits(GPIOD,GPIO_Pin_9);
           }
}
```

Chú ý:

Để cấu hình 3 chân A8, C13 và D9 ta không nên tách ra làm 3 chương trình con riêng biệt vì nếu vậy chương trình sẽ rất dài. Trong chương trình trên ta đã gom chúng lại thành 1 chương trình duy nhất tên là **"CauHinh_Button_LED()"** và nhờ thao tác này mà chương trình sẽ gọn đi rất nhiều do kế thừa được các khai báo trước đó. Ví dụ quan sát chương trình con **"CauHinh_Button_LED()"** ta thấy việc cấu hình cho chân C13 thiếu lệnh chọn chế độ do trước đó khi cấu hình chân A8 ta đã chọn rồi. Vậy khi cấu hình GPIO ta thấy cái gì thay đổi thì ta cấu hình lại còn không thay đổi thì bỏ qua.

Giải thích chương trình:

Để kiểm tra có nhấn nút hay không ta dùng lệnh:

```
if((GPIO_ReadInputData(GPIOA)&GPIO_Pin_8) ==0)
```

Lệnh trên có nghĩa là đọc PORTA về rồi **"AND"** với GPIO_Pin_8 để xóa hết trạng thái các chân khác chỉ để lại chân A8 nếu có nhấn nút thì chân A8 = '0' làm cho điều kiện lệnh if đúng (true) và lúc này ta mở LED bằng lệnh:

```
GPIO_ResetBits(GPIOD,GPIO_Pin_9);
```

Tương tự ta kiểm tra chân C13 nếu phát hiện có nhấn nút thì tắt LED bằng lệnh:

```
GPIO_SetBits(GPIOD,GPIO_Pin_9);
```

4.5.3 Ví dụ về remap

Ví dụ 4.10: Một board ARM STM32F103VET6 thi công sẵn sử dụng hết các chân để giao tiếp với các thiết bị bên ngoài chỉ còn dư lại các chân từ B0 đến B7 đưa ra một hàng Pin. Vậy làm cách nào để sử dụng board này giao tiếp với Module Sim900 qua chuẩn UART biết các chân từ B0 đến B7 mặc định không phải là chân của UART nào hết?

a. Giải pháp

Khi thiết kế cũng như khi sử dụng board sẵn có nếu gặp khó khăn về việc sử dụng đụng chân đã sài rồi thì ta nên nghĩ tới việc **"remap" .**

Trong trường hợp này Module Sim đang cần được giao tiếp qua UART nên ta kiếm trong bảng remap UART xem có UART nào sau khi remap có chân nằm trong phạm vi từ B0 đến B7 không?

⇨ Nhận thấy UART1 sau khi remap có chân TX là B6 và RX là B7 thỏa mãn yêu cầu. Vậy ta sẽ sử dụng UART1 remap để giao tiếp với Module Sim900. Sau đây là đoạn chương trình con hướng dẫn cách remap và cấu hình chân cho UART1.

b. Chương trình

```
void  CauhinhchanUART1_Remap()
  {
   GPIO_InitTypeDef     GPIO_InitStructure;
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB
                          |RCC_APB2Periph_AFIO
                          |RCC_APB2Periph_USART1, ENABLE);
   // Cho phép cấp xung clock cho PORTB, AFIO và UART1
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;
   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
   GPIO_Init(GPIOB, &GPIO_InitStructure);
   //TX chức năng thay thế đẩy kéo, 50M
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_7;
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
   GPIO_Init(GPIOB, &GPIO_InitStructure);
   // RX ngõ vào thả nổi
   GPIO_PinRemapConfig(GPIO_Remap_USART1, ENABLE);
   // Remap UART1
  }
```

4.5.4 Ví dụ xuất cách sử dụng kỹ thuật bitband

Ví dụ 4.11: Viết chương trình sử dụng kỹ thuật bitband điều khiển LED đơn kết nối với chân D8 sáng hoặc tắt nhờ nút nhấn ONOFF kết nối với chân A8. Khi LED đang tắt nhấn nút thì LED sáng và ngược lại.

Chương trình

```
#include<stm32f10x.h>
#define     Bitband(diachi,bitso)  *(unsigned long*)\
((diachi&0xF0000000)+0x2000000+((diachi&0xFFFFF)*32)+ bitso*4)
#define   LED           Bitband((unsigned long)&GPIOD->ODR,8)
#define   BT            Bitband((unsigned long)&GPIOA->IDR,8)
// Định nghĩa sử dụng Bitband- xem bài giới thiệu STM32F1
void delay(unsigned long t){while(t--);}
void CauHinh_Button_LED()
{
     GPIO_InitTypeDef     GPIO_Structure;
     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD
                            |RCC_APB2Periph_GPIOA, ENABLE);
    // Cho phép cấp xung clock cho PortA, D (Xem bài RCC)
     GPIO_Structure.GPIO_Pin = GPIO_Pin_8;
     GPIO_Structure.GPIO_Mode = GPIO_Mode_Out_PP;
     GPIO_Structure.GPIO_Speed = GPIO_Speed_50MHz;
     GPIO_Init(GPIOD,&GPIO_Structure );
    // Cấu hình cho chân D8 điều khiển LED
     GPIO_Structure.GPIO_Mode = GPIO_Mode_IPU;
     GPIO_Init(GPIOA,&GPIO_Structure );
    // Cấu hình cho chân A8 giao tiếp với nút nhấn
}
int main(void)
{
    SystemInit();
    CauHinh_Button_LED();
    while(1)
    {
             if(!BT) // Kiểm tra chống dội nút nhấn
             {
                   delay(500000); // delay chống đội
                   if(!BT)
                     {
                      LED=!LED;  // Đảo trạng thái led
                      while(!BT);// Chờ nhả phím
                     }
             }
      }
}
```

4.6 CÁC CÂU HỎI GIÚP SINH VIÊN CỦNG CỐ NỘI DUNG CỦA BÀI HỌC

Câu hỏi 1: Một bạn SV khi thiết kế mạch đã sử dụng các chân từ B0 đến B7 của ARM STM32F103VET6 làm 8 đường dữ liệu cho LCD 20x4. Khi viết chương trình bạn SV đó đã cấu hình 8 chân trên như sau:

```
void  CauhinhchanLCD()
{
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
   GPIOB->CRL = 0x33333333;
}
```

Hỏi việc cấu hình như vậy có đúng chưa ? Nếu sai thì bạn hãy cấu hình lại sao cho đúng.

Câu hỏi 2: Tại sao trong **ví dụ 4.10** khi cấu hình GPIO cho chân RX của USART tác giả lại cấu hình chân này là ngõ vào thả nổi mà không cấu hình là chức năng thay thế?

```
void  CauhinhchanUART1_Remap()
   {
    ...
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    ...
  }
```

Câu hỏi 3: Viết chương trình điều khiển 4 led đơn kết nối với các chân từ D8 đến D11 sáng dần, tắt dần, sáng dịch. Các kiểu sáng này được chọn bởi nút nhấn MODE kết nối với chân A8. Khi nhấn nút MODE sẽ chuyển sang trạng thái kế ngay lập tức. SV hãy thực hiện các yêu cầu trên với điều kiện không dùng thư viện "stm32f10x_gpio" và nút nhấn MODE sử dụng kỹ thuật bitband.

Câu hỏi 4: Trong **ví dụ 4.11** tác giả đã chống đội phím nhấn đơn bằng phương pháp delay và chờ nhả phím điều này giây trì hoãn hệ thống khi người dung nhấn phím lâu. SV hãy làm lại ví dụ trên mà không sử dụng delay cũng như chờ nhả phím.