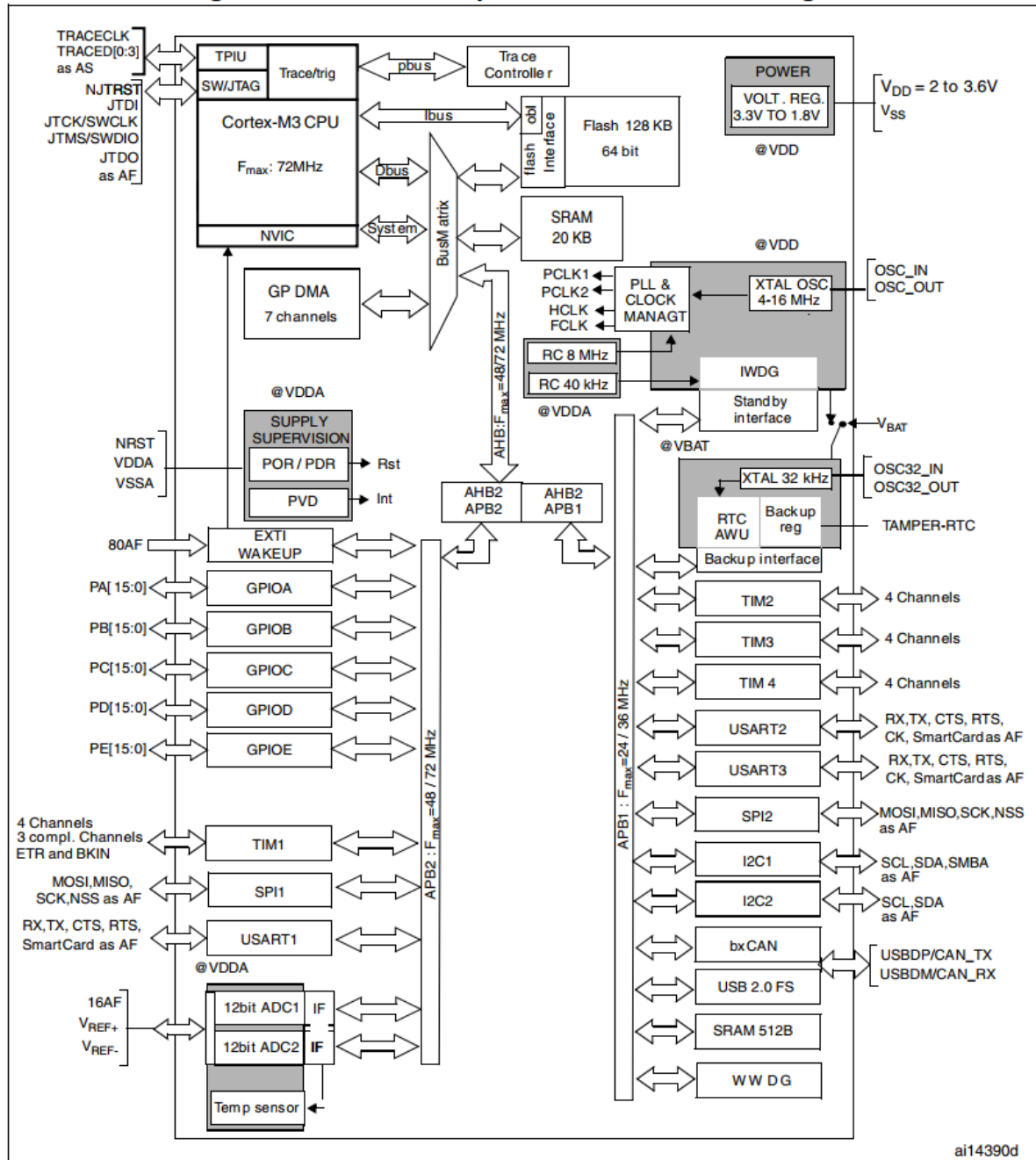


2.1. Thư viện STM32F10x Standard Peripherals Firmware Library

Figure 1. STM32F103xx performance line block diagram



2.3 Cấp clock cho ngoại vi

Module RCC cung cấp các hàm để cấu hình xung clock.

RCC_APB1PeriphClockCmd

RCC_APB2PeriphClockCmd

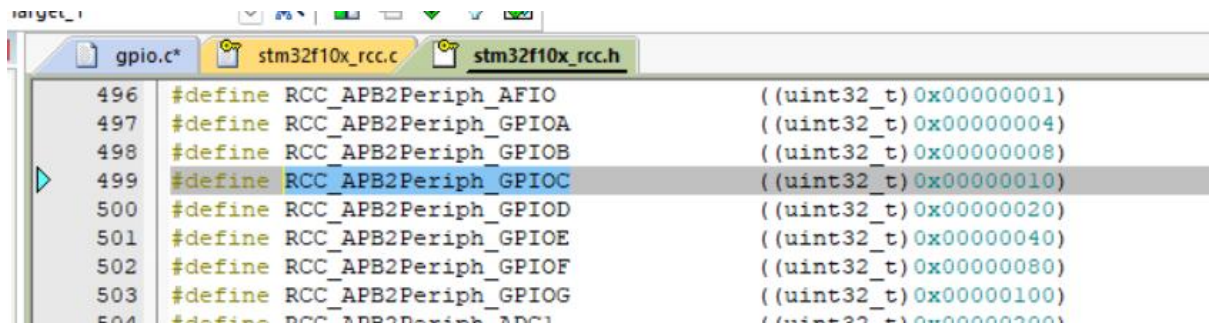
RCC_AHBPeriphClockCmd

Hàm này nhận 2 tham số:

- Ngoại vi muốn cấp clock
- Cho phép (ENABLE) hoặc không cho phép (DISABLE) cấp clock cho ngoại vi

```
brief Enables or disables the High Speed APB (APB2) peripheral clock.
* @param RCC_APB2Periph: specifies the APB2 peripheral to gates its clock.
* This parameter can be any combination of the following values:
* @arg RCC_APB2Periph_AFIO, RCC_APB2Periph_GPIOA,
RCC_APB2Periph_GPIOB,
* RCC_APB2Periph_GPIOC, RCC_APB2Periph_GPIOD,
RCC_APB2Periph_GPIOE,
* RCC_APB2Periph_GPIOF, RCC_APB2Periph_GPIOG, RCC_APB2Periph_ADC1,
* RCC_APB2Periph_ADC2, RCC_APB2Periph_TIM1, RCC_APB2Periph_SPI1,
* RCC_APB2Periph_TIM8, RCC_APB2Periph_USART1,
RCC_APB2Periph_ADC3,
* RCC_APB2Periph_TIM15, RCC_APB2Periph_TIM16, RCC_APB2Periph_TIM17,
* RCC_APB2Periph_TIM9, RCC_APB2Periph_TIM10,
RCC_APB2Periph_TIM11
* @param NewState: new state of the specified peripheral clock.
* This parameter can be: ENABLE or DISABLE.
* @retval None
*/
void RCC_APB2PeriphClockCmd(uint32_t RCC_APB2Periph, FunctionalState NewState)
{
    /* Check the parameters */
    assert_param(IS_RCC_APB2_PERIPH(RCC_APB2Periph)); // Hàm assert trong C để
    kiểm tra điều kiện, nếu không thỏa mãn dừng chương trình
    assert_param(IS_FUNCTIONAL_STATE(NewState));
    if (NewState != DISABLE)
    {
        RCC->APB2ENR |= RCC_APB2Periph; // RCC_APB2Periph là 1 mặt nạ bit, quy ra nhị
        phân chỉ 1 bit = 1 ở vị trí gpio tương ứng, Phép Or chính là ghi vào thanh ghi RCC-
        >APB2ENR bit 1, cho phép nó hoạt động
    }
    else
    {
        RCC->APB2ENR &= ~RCC_APB2Periph; // Dùng phép đảo bit xóa thanh ghi RCC-
        >APB2ENR cho về 0 hết
    }
}
```

Các mặt nạ sau khi quy ra nhị phân thì chỉ có một bit ở vị trí gpio tương ứng = 1, dùng phép OR bit và ~ bit để ghi giá trị 1 vào thanh ghi hoặc xóa giá trị này đi.



```
496 #define RCC_APB2Periph_AFIO          ((uint32_t)0x00000001)
497 #define RCC_APB2Periph_GPIOA        ((uint32_t)0x00000004)
498 #define RCC_APB2Periph_GPIOB        ((uint32_t)0x00000008)
499 #define RCC_APB2Periph_GPIOC        ((uint32_t)0x00000010)
500 #define RCC_APB2Periph_GPIOD        ((uint32_t)0x00000020)
501 #define RCC_APB2Periph_GPIOE        ((uint32_t)0x00000040)
502 #define RCC_APB2Periph_GPIOF        ((uint32_t)0x00000080)
503 #define RCC_APB2Periph_GPIOG        ((uint32_t)0x00000100)
```

2.4 Cấu hình ngoại vi

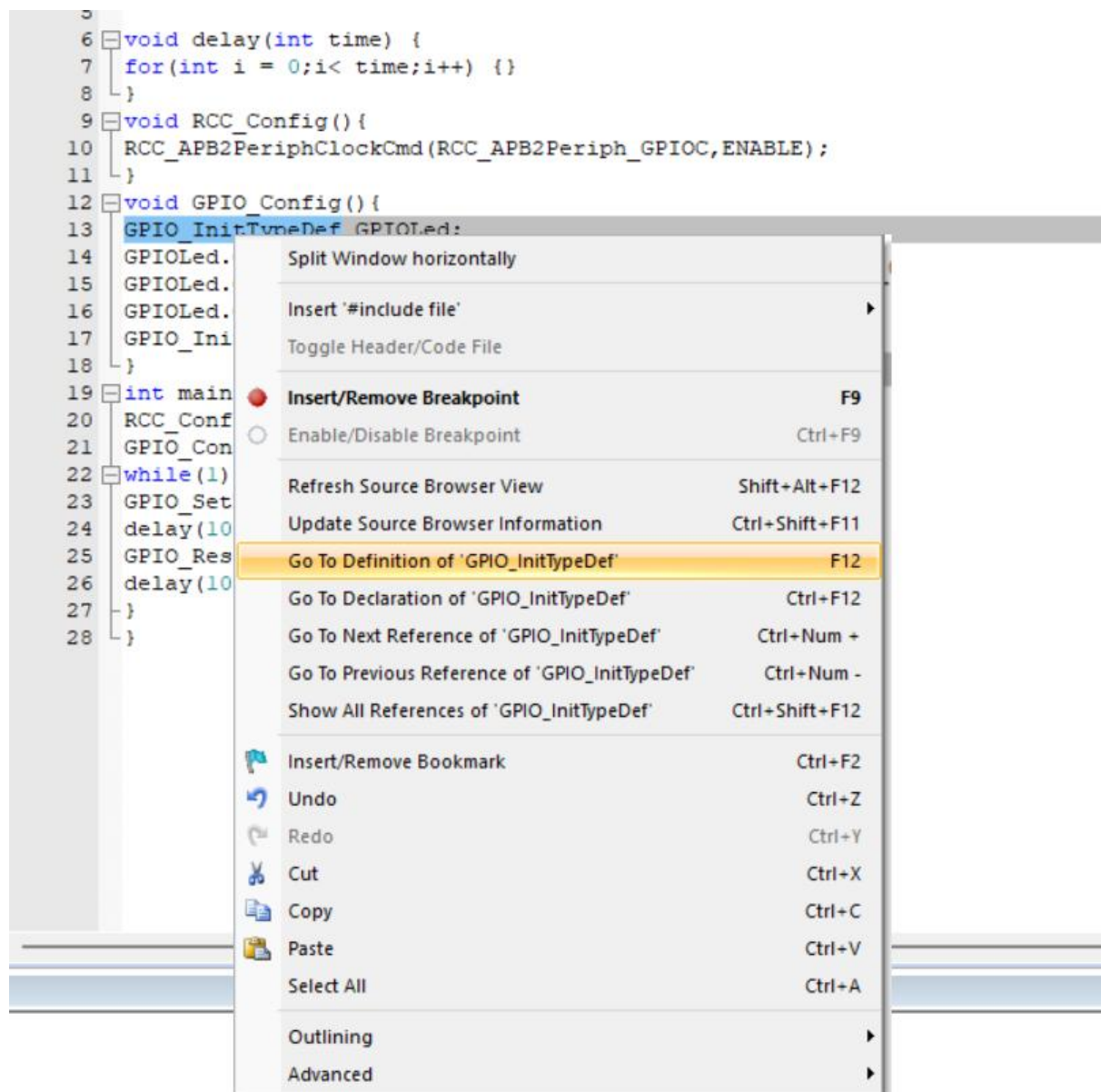
- GPIO được định nghĩa trong 1 struct trong thư viện gpio.h, gpio.c
- Việc cấu hình là khai báo struct đó ra và truyền tham số cho từng thành viên để thực hiện cấu hình.

Chú ý : File head .h chỉ khai báo tên các struct, tên hàm, tên biến toàn cục cho gọn.

Còn việc triển khai code có các hàm sẽ nằm cụ thể trong file .c

Bạn có thể search hàm bằng từ khóa go to Definition.

Trong file .h search go definition không thấy thì khả năng nó nằm trong file .c cùng tên.



```

90 typedef struct
91 {
92     uint16_t GPIO_Pin;          /*!< Specifies the GPIO pins to be configured.
93                                 This parameter can be any value of @ref GPIO_pins_defin
94
95     GPIOSpeed_TypeDef GPIO_Speed; /*!< Specifies the speed for the selected pins.
96                                 This parameter can be a value of @ref GPIOSpeed_TypeDef
97
98     GPIOMode_TypeDef GPIO_Mode; /*!< Specifies the operating mode for the selected pins.
99                                 This parameter can be a value of @ref GPIOMode_TypeDef
100 }GPIO_InitTypeDef;
101
102
103 /**
104  * @brief Bit_SET and Bit_RESET enumeration
105  */

```

GPIO_Pin:

Pin nó cũng là 1 cái mask, quy đổi ra nhị phân thì chỉ có 1 bit = 1 => đó chính là vị trí pin.

Mask dùng để thực hiện phép Or bit, ~ bit,... với biến thanh ghi để ghi dữ liệu vào thanh ghi.

```

121 |
122 | /** @defgroup GPIO_pins_define
123 | * @{
124 | */
125 |
126 | #define GPIO_Pin_0          ((uint16_t)0x0001)  /*!<< Pin 0 selected */
127 | #define GPIO_Pin_1          ((uint16_t)0x0002)  /*!<< Pin 1 selected */
128 | #define GPIO_Pin_2          ((uint16_t)0x0004)  /*!<< Pin 2 selected */
129 | #define GPIO_Pin_3          ((uint16_t)0x0008)  /*!<< Pin 3 selected */
130 | #define GPIO_Pin_4          ((uint16_t)0x0010)  /*!<< Pin 4 selected */
131 | #define GPIO_Pin_5          ((uint16_t)0x0020)  /*!<< Pin 5 selected */
132 | #define GPIO_Pin_6          ((uint16_t)0x0040)  /*!<< Pin 6 selected */
133 | #define GPIO_Pin_7          ((uint16_t)0x0080)  /*!<< Pin 7 selected */
134 | #define GPIO_Pin_8          ((uint16_t)0x0100)  /*!<< Pin 8 selected */
135 | #define GPIO_Pin_9          ((uint16_t)0x0200)  /*!<< Pin 9 selected */
136 | #define GPIO_Pin_10         ((uint16_t)0x0400)  /*!<< Pin 10 selected */
137 | #define GPIO_Pin_11         ((uint16_t)0x0800)  /*!<< Pin 11 selected */
138 | #define GPIO_Pin_12         ((uint16_t)0x1000)  /*!<< Pin 12 selected */
139 | #define GPIO_Pin_13         ((uint16_t)0x2000)  /*!<< Pin 13 selected */
140 | #define GPIO_Pin_14         ((uint16_t)0x4000)  /*!<< Pin 14 selected */
141 | #define GPIO_Pin_15         ((uint16_t)0x8000)  /*!<< Pin 15 selected */
142 | #define GPIO_Pin_All        ((uint16_t)0xFFFF)  /*!<< All pins selected */
143 |
144 | #define IS_GPIO_PIN(PIN) (((PIN) & (uint16_t)0x00) == 0x00) && ((PIN) != (uint16_t)0x00)

```

GPIO_Speed,

Nó là 1 struct, chọn tốc độ tương ứng 10MHz, 2 MHz, 50 MHz.

```

16 |
17 | typedef enum
18 | {
19 |     GPIO_Speed_10MHz = 1,
20 |     GPIO_Speed_2MHz,
21 |     GPIO_Speed_50MHz
22 | }GPIO_Speed_TypeDef;
23 | #define IS_GPIO_SPEED(SPEED) (((SPEED) == GPIO_Speed_10MHz) || ((SPEED) == GPIO_Speed_2MHz) || \
24 |                               ((SPEED) == GPIO_Speed_50MHz))
25 |
26 | /**

```

GPIO_Mode

```

69 |
70 | typedef enum
71 | {
72 |     GPIO_Mode_AIN = 0x0,
73 |     GPIO_Mode_IN_FLOATING = 0x04,
74 |     GPIO_Mode_IPD = 0x28,
75 |     GPIO_Mode_IPU = 0x48,
76 |     GPIO_Mode_Out_OD = 0x14,
77 |     GPIO_Mode_Out_PP = 0x10,
78 |     GPIO_Mode_AF_OD = 0x1C,
79 |     GPIO_Mode_AF_PP = 0x18
80 | }GPIO_Mode_TypeDef;
81 | #define IS_GPIO_MODE(MODE) (((MODE) == GPIO_Mode_AIN) || ((MODE) == GPIO_Mode_IN_FLOATING) || \
82 |                             ((MODE) == GPIO_Mode_IPD) || ((MODE) == GPIO_Mode_IPU) || \
83 |                             ((MODE) == GPIO_Mode_Out_OD) || ((MODE) == GPIO_Mode_Out_PP) || \
84 |                             ((MODE) == GPIO_Mode_AF_OD) || ((MODE) == GPIO_Mode_AF_PP))
85 |

```

Các chế độ chân GPIO:

Chế độ đầu vào (Input Modes):

1. Input floating:

- **Đặc điểm:** Chân GPIO được cấu hình để nhận tín hiệu đầu vào mà không có điện trở kéo lên hoặc kéo xuống.

- **Ứng dụng:** Sử dụng khi cần nhận tín hiệu từ nguồn bên ngoài mà không muốn có sự can thiệp từ điện trở nội bộ.
 - **Lưu ý:** Tín hiệu có thể không ổn định do dễ bị nhiễu.
2. **Input pull-up:**
- **Đặc điểm:** Chân GPIO được cấu hình với điện trở kéo lên (được kết nối với nguồn điện dương).
 - **Ứng dụng:** Sử dụng khi cần đảm bảo chân GPIO luôn ở mức cao trừ khi có tín hiệu kéo xuống.
 - **Lợi ích:** Giảm thiểu nhiễu và dao động tín hiệu.
3. **Input pull-down:**
- **Đặc điểm:** Chân GPIO được cấu hình với điện trở kéo xuống (được kết nối với nguồn điện âm).
 - **Ứng dụng:** Sử dụng khi cần đảm bảo chân GPIO luôn ở mức thấp trừ khi có tín hiệu kéo lên.
 - **Lợi ích:** Giảm thiểu nhiễu và dao động tín hiệu.
4. **Analog:**
- **Đặc điểm:** Chân GPIO hoạt động ở chế độ tương tự, không số hóa tín hiệu.
 - **Ứng dụng:** Sử dụng khi cần đọc hoặc ghi tín hiệu tương tự, như trong các ứng dụng cảm biến hoặc DAC (Digital-to-Analog Converter).

Chế độ đầu ra (Output Modes):

5. **Output open-drain:**
- **Đặc điểm:** Chân GPIO chỉ có thể kéo tín hiệu xuống mức thấp hoặc để nổi (hở mạch), cần điện trở kéo lên bên ngoài.
 - **Ứng dụng:** Sử dụng trong các mạch nối nhiều thiết bị với nhau hoặc trong các ứng dụng I2C.
 - **Lợi ích:** Phù hợp cho việc chia sẻ tín hiệu giữa nhiều thiết bị.
6. **Output push-pull:**
- **Đặc điểm:** Chân GPIO có thể kéo tín hiệu lên mức cao hoặc xuống mức thấp.
 - **Ứng dụng:** Sử dụng trong các ứng dụng yêu cầu tín hiệu số rõ ràng và mạnh mẽ.
 - **Lợi ích:** Đảm bảo tín hiệu ổn định và rõ ràng.

Chế độ chức năng thay thế (Alternate Function Modes):

7. **Alternate function push-pull:**
- **Đặc điểm:** Chân GPIO được sử dụng để điều khiển các chức năng thay thế như UART, SPI, I2C, với khả năng kéo tín hiệu lên hoặc xuống.
 - **Ứng dụng:** Sử dụng khi cần kết nối

2.5 Các hàm thông dụng

```
uint8_t GPIO_ReadInputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
\\Đọc giá trị 1 chân trong GPIO được cấu hình là INPUT
uint16_t GPIO_ReadInputData(GPIO_TypeDef* GPIOx);
\\Đọc giá trị nguyên GPIO được cấu hình là INPUT, ghi vào biến con trỏ hay mảng dữ liệu
uint8_t GPIO_ReadOutputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
\\Đọc giá trị 1 chân trong GPIO được cấu hình là OUTPUT
uint16_t GPIO_ReadOutputData(GPIO_TypeDef* GPIOx);
\\Đọc giá trị nguyên GPIO được cấu hình là OUTPUT
void GPIO_SetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
\\Cho giá trị điện áp của 1 chân trong GPIO = 1
void GPIO_ResetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
\\Cho giá trị điện áp của 1 chân trong GPIO = 0
void GPIO_WriteBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, BitAction BitVal);
\\Ghi giá trị "BitVal" vào 1 chân trong GPIO
void GPIO_Write(GPIO_TypeDef* GPIOx, uint16_t PortVal);
\\Ghi giá trị "PortVal" vào nguyên GPIO
```


uint8_t **GPIO_ReadInputDataBit**(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);

\\Đọc giá trị 1 chân trong GPIO được cấu hình là INPUT

```
/**
 * @/**
 * @brief Reads the specified input port pin.
 * @param GPIOx: where x can be (A..G) to select the GPIO peripheral.
 * @param GPIO_Pin: specifies the port bit to read.
 * This parameter can be GPIO_Pin_x where x can be (0..15).
 * @retval The input port pin value.
 */
uint8_t GPIO_ReadInputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
{
    uint8_t bitstatus = 0x00; // tạo biến chứa giá trị từ chân gpio

    /* Check the parameters */
    assert_param(IS_GPIO_ALL_PERIPH(GPIOx)); // kiểm tra điều kiện đúng port
    GPIO, sai dừng chương trình
    assert_param(IS_GET_GPIO_PIN(GPIO_Pin)); // kiểm tra điều kiện đúng chân
    GPIO_Pin, sai dừng chương trình

    if ((GPIOx->IDR & GPIO_Pin) != (uint32_t)Bit_RESET) // nếu Giá trị pin trong
    thanh ghi IDR khác 0 thì ghi bitstatus lên 1
    {
        bitstatus = (uint8_t)Bit_SET;
    }
    else
    {
        bitstatus = (uint8_t)Bit_RESET;
    }
    return bitstatus;
}
```



```

1413 #define CEC ((CEC_TypeDef *) CEC_BASE)
1414 #define AFIO ((AFIO_TypeDef *) AFIO_BASE)
1415 #define EXTI ((EXTI_TypeDef *) EXTI_BASE)
1416 #define GPIOA ((GPIO_TypeDef *) GPIOA_BASE)
1417 #define GPIOB ((GPIO_TypeDef *) GPIOB_BASE)
1418 #define GPIOC ((GPIO_TypeDef *) GPIOC_BASE)
1419 #define GPIOD ((GPIO_TypeDef *) GPIOD_BASE)
1420 #define GPIOE ((GPIO_TypeDef *) GPIOE_BASE)
1421 #define GPIOF ((GPIO_TypeDef *) GPIOF_BASE)

```

```

1320
1321 #define AFIO_BASE (APB2PERIPH_BASE + 0x0000)
1322 #define EXTI_BASE (APB2PERIPH_BASE + 0x0400)
1323 #define GPIOA_BASE (APB2PERIPH_BASE + 0x0800)
1324 #define GPIOB_BASE (APB2PERIPH_BASE + 0x0C00)
1325 #define GPIOC_BASE (APB2PERIPH_BASE + 0x1000)
1326 #define GPIOD_BASE (APB2PERIPH_BASE + 0x1400)
1327 #define GPIOE_BASE (APB2PERIPH_BASE + 0x1800)
1328 #define GPIOF_BASE (APB2PERIPH_BASE + 0x1C00)
1329 #define GPIOG_BASE (APB2PERIPH_BASE + 0x2000)
1330 #define ADC1_BASE (APB2PERIPH_BASE + 0x2400)
1331 #define ADC2_BASE (APB2PERIPH_BASE + 0x2800)
1332 #define TIM1_BASE (APB2PERIPH_BASE + 0x2C00)
1333 #define SPI1_BASE (APB2PERIPH_BASE + 0x3000)
1334 #define TIM8_BASE (APB2PERIPH_BASE + 0x3400)
1335 #define USART1_BASE (APB2PERIPH_BASE + 0x3800)
1336 #define ADC3_BASE (APB2PERIPH_BASE + 0x3C00)
1337 #define TIM15_BASE (APB2PERIPH_BASE + 0x4000)
1338 #define TIM16_BASE (APB2PERIPH_BASE + 0x4400)
1339 #define TIM17_BASE (APB2PERIPH_BASE + 0x4800)
1340 #define TIM9_BASE (APB2PERIPH_BASE + 0x4C00)
1341 #define TIM10_BASE (APB2PERIPH_BASE + 0x5000)
1342 #define TIM11_BASE (APB2PERIPH_BASE + 0x5400)

```

```

//Code nhay led, chay led
#include "stm32f10x.h" // Device header
#include "stm32f10x_rcc.h" // Keil::Device:StdPeriph Drivers:RCC
#include "stm32f10x_gpio.h" // Keil::Device:StdPeriph Drivers:GPIO

void delay(int time) {
for(int i = 0;i< time;i++) {}
}

void RCC_Config(){
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC,ENABLE);
}

void GPIO_Config(){
GPIO_InitTypeDef GPIOLed;
GPIOLed.GPIO_Pin = GPIO_Pin_13 ;
GPIOLed.GPIO_Mode = GPIO_Mode_Out_PP;
GPIOLed.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOC,&GPIOLed);
}

void chaseLed(uint8_t loop){ // Nhay duoi led
uint16_t Ledval;
for(int j = 0; j < loop; j++)
{
Ledval = 0x0010; //0b0 0001 0000
for(int i = 0; i < 4; i++)
{
Ledval = Ledval << 1;
GPIO_Write(GPIOC, Ledval);
delay(1000000);
}
}
}

int main(){
RCC_Config();
GPIO_Config();
while(1){
GPIO_SetBits(GPIOC,GPIO_Pin_13);
delay(10000000);
GPIO_ResetBits(GPIOC,GPIO_Pin_13);
delay(10000000);
}
}

```