Chương 3

BỘ ĐIỀU KHIỂN RESET VÀ XUNG CLOCK

RESET AND CLOCK CONTROL – RCC

3.1 CÁC LOẠI RESET

Có 3 loại reset là:

- System Reset
- Power Reset
- Backup Domain Reset.

3.1.1 System Reset

System Reset sẽ reset giá trị của tất cả các thanh ghi ngoại trừ các cờ trong thanh ghi điều khiển xung clock và các thanh ghi của Backup Domain.

System Reset xảy ra khi một trong những sự kiện sau phát sinh:

- Tác động vào chân NRST bằng điện áp mức thấp(reset ngoài).
- Window Watchdog đếm hết giá trị đặt (WWDG reset).
- Independent Watchdog đếm hết giá trị đặt (IWDG reset)
- Reset bằng phần mềm(SW reset)
- Reset do việc điều khiển các chế độ tiết kiệm năng lượng.
  - Reset xảy ra khi đưa ARM vào chế độ Standby.
  - Reset xảy ra khi đưa ARM vào chế độ Stop.

Nguyên nhân của Reset có thể biết được thông qua việc kiểm tra các cờ báo Reset trong thanh ghi Control/Status (RCC_CSR ở địa chỉ 0x24 – các cờ báo reset trong thanh ghi này chỉ có thể được reset bằng Power Reset).

3.1.2 Power reset

Power reset sẽ reset giá trị của tất cả các thanh ghi về trạng thái mặc định của nó ngoại trừ các thanh ghi của Backup Domain.

Power Reset xảy ra khi một trong những sự kiện sau phát sinh:

- Mở nguồn/ Sụt áp (POR/PDR reset)
- Thoát khỏi chế độ Standby.

3.1.3 Backup domain reset

Backup Domain Reset xảy ra khi một trong những sự kiện sau phát sinh:

- Reset bằng phần mềm, kích hoạt khi đặt bit BDRST(bit thứ 16) trong thanh ghi Backup domain control(RCC_BDCR địa chỉ 0x20) là mức ‘1’.
- Khi cấp nguồn VDD hay VBAT mà cả hai nguồn này trước đó đều tắt(người ta thường dùng để kiểm tra việc hết Pin).

3.2 CÁC NGUỒN XUNG CLOCK(CLOCKS)

Có 3 loại xung clock có thể được dùng làm xung clock hệ thống(SYSCLK-xung clock cấp cho khối xử lý):

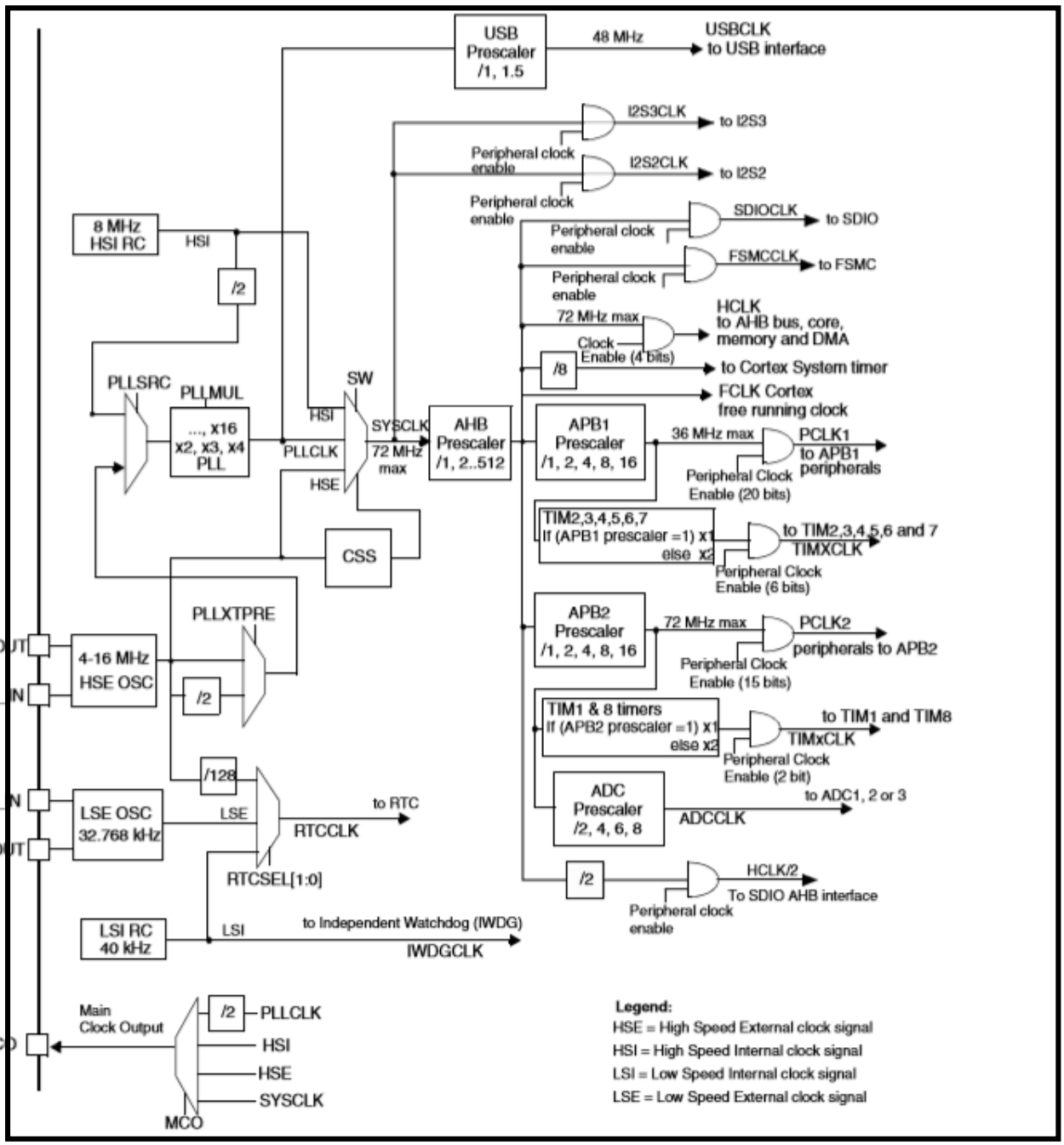
- HSI (High Speed Internal) nguồn xung clock tốc độ cao ở bên trong ARM.
- HSE(High Speed External) nguồn xung clock tốc độ cao ở bên ngoài ARM
- PLL(Phase Locked Loop) nguồn xung clock lấy từ bộ nhân tần số(hay sử dụng).

Ngoài ra ARM cũng còn có 2 nguồn xung clock phụ sau:

● LSI(Low Speed Internal) nguồn xung clock tốc độ chậm 40 kHz ở bên trong ARM nguồn xung clock này được dùng để cấp cho independent watchdog và có thể được dùng để cấp cho RTC trong việc định thời gian tự động đánh thức CPU thoát khỏi chế độ Sop hoặc Stanbly.

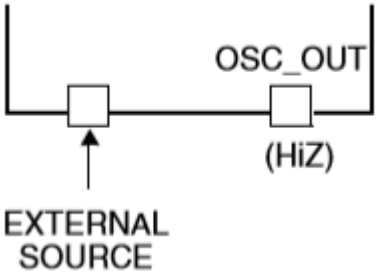
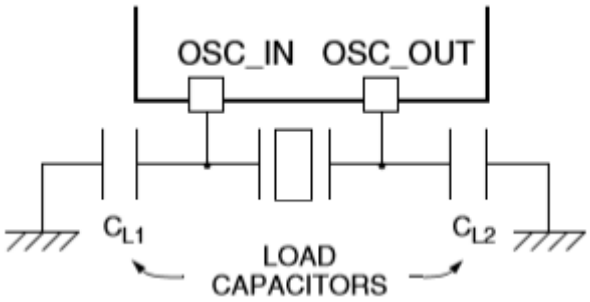
● LSE(Low Speed External) nguồn xung clock tốc độ chậm thường được nối với thạch anh 32.768 kHz từ bên ngoài, xung clock này có thể được dùng để cấp cho RTC.

Mỗi nguồn xung clock có thể được bật tắt độc lập nhằm tiết kiệm năng lượng.



Hình 3.1 Sơ đồ cây xung clock

Chú ý: Muốn hệ thống có thể hoạt động được ở tần số cao nhất(72Mhz) ta phải sử dụng HSE (4-16Mhz) kết hợp với mạch nhân tần số PLLMUL. Thông thường ta hay chọn giá trị HSE = 8Mhz, điều này có nghĩa là ta phải kết nối thạch anh hoặc một nguồn xung clock 8Mhz bên ngoài ARM theo một trong hai như **hình 3.2**:

Nguồn xung clock	Kết nối phần cứng
Sử dụng nguồn xung clock ở bên ngoài	
Sử dụng thạch anh	

Hình 3.2 Các cách kết nối cho nguồn xung ngoại HSE

Nếu không muốn dùng thạch anh hoặc clock ngoại như HSE ta có thể dùng giao động nội HSI nhưng lúc này tốc độ tối đa chỉ đạt 64Mhz.

Muốn dùng RTC định thời một cách chính xác ta phải sử dụng LSE bằng cách kết nối thạch anh hoặc bộ giao động có tần số 32.768 Khz vào 2 chân OSC32_IN và OSC32_OUT theo một trong hai cách giống với khi dùng HSE ở **hình 3.2**.

Đối với những ứng dụng không yêu cầu RTC định thời chính xác ta có thể sử dụng LSI(40 Khz) thay cho LSE giúp tiết kiệm được thạch anh gắn ngoài.

Mặc định các nguồn xung clock cấp cho ngoại vi được tắt hết để tiết kiệm năng lượng tiêu thụ. Vì vậy, khi lập trình trước khi điều khiển ngoại vi nào thì ta đều phải cho phép cấp xung clock cho ngoại vi đó thông qua khối quản lý xung clock của nó. **Bảng 3.1** liệt kê các ngoại vi và nguồn cấp xung clock của chúng.

Bảng 3.1 Ngoại vi và nguồn cấp xung clock của chúng.

NGOẠI VI	NGUỒN XUNG CLOCK	GHI CHÚ
USB	PLLCLK (cho OTG-FS) APB1	USB: Universal Serial Bus OTG-FS: On The Go Full Speed APB1(Advanced Peripheral Bus 1): Bus ngoại vi tốc độ tối đa 36 Mhz.
I2S2 I2S3	SYSCLK	I2S (Inter-IC Sound): Bus truyền dữ liệu âm thanh số. SYSCLK(System Clock): Xung clock cấp cho nhân ARM Cortex.
SDIO FSMC	AHB	SDIO (*Secure Digital Input Output*): Bus này thường được dùng để giao tiếp với thẻ nhớ SD. AHB(Advanced High-performance Bus): Bus hiệu năng cao. FSMC(Flexible Static Memory Controller) bus này được dùng để giao tiếp với bộ nhớ ngoài và kể cả LCD.

Timer 2,3,4,5,6,7 SPI 2,3 USART 2,3 UART 4,5 I2C 1,2 CAN 1,2 BKP PWR DAC WWDG	APB1	SPI: Serial Peripheral Interface USART: Universal Asynchronous Receiver /Transmitter I2C: Inter-Integrated Circuit CAN: Controller Area Network BKP: Backup PWR: Power DAC: Digital-To-Analog Converter WWDG: Window Watchdog
GPIOA,B,C, D,E,F,G AFIO ADC 1,2,3 Timer 1,8 SPI 1 USART 1	APB2	GPIO(General-purpose input/output): Các chức năng liên quân đến xuất nhập dữ liệu qua chân ARM. AFIO (Alternate Function I/O): APB2(Advanced Peripheral Bus 2): Bus ngoại vi tốc độ tối đa 72 Mhz.

.

Ví dụ: Khi muốn sử dụng SPI2 và GPIOA thì nhìn trên **bảng 3.1** ta thấy SPI2 thuộc APB1 và GPIOA thuộc APB2 do đó để cấp xung clock ta viết:

```
RCC_APB1PeriphClockCmd(RCC_APB1Periph_SPI2, ENABLE);
```
(Cấp xung clock cho SPI2 từ APB1, để tắt ta đổi "ENABLE" thành "DISABLE")

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);
```
(Cấp xung clock cho GPIOA từ APB2)

3.2.1 Các lệnh thông dụng để cài đặt xung clock.

Bảng 3.2 Các lệnh thông dụng cho việc cài đặt xung clock

SỬ DỤNG THƯ VIỆN "stm32f10x_rcc"	
Lệnh	
Thông số hay dùng	**Giải thích**
SystemInit(); (Lệnh cài đặt xung clock sử dụng HSE(8Mhz) , PLL (x9)) • SYSCLK: 72 Mhz • PCLK1: 36 Mhz • PCLK2 : 72 Mhz	
RCC_DeInit(); (Lệnh reset RCC)	
RCC_HSEConfig(A); (Lệnh cấu bật hoặc tắt HSE)	
A: RCC_HSE_ON RCC_HSE_OFF	**A:** Bật hoặc tắt Bật HSE Tắt HSE

RCC_HSICmd(A); (Lệnh bật hoặc tắt HSI)	
A: ENABLE DISABLE	**A:** Bật hoặc tắt Bật HSI Tắt HSI
RCC_LSEConfig(A); (Lệnh bật hoặc tắt LSE)	
A: RCC_LSE_ON RCC_LSE_OFF	**A:** Bật hoặc tắt Bật LSE Tắt LSE
RCC_LSICmd(A); (Lệnh bật hoặc tắt LSI)	
A: ENABLE DISABLE	**A:** Bật hoặc tắt Bật LSI Tắt LSI
RCC_PLLCmd(A); (Lệnh bật hoặc tắt PLL)	
A: ENABLE DISABLE	**A:** Bật hoặc tắt Bật PLL Tắt PLL
FlagStatus F; **F=RCC_GetFlagStatus(A);** (Lệnh kiểm tra cờ trạng thái của RCC)	
A: RCC_FLAG_HSIRDY RCC_FLAG_HSERDY RCC_FLAG_PLLRDY RCC_FLAG_LSERDY RCC_FLAG_LSIRDY …	**A:** Cờ cần kiểm tra HSI đã sẵn sàng? HSE đã sẵn sàng? PLL đã sẵn sàng? LSE đã sẵn sàng? LSI đã sẵn sàng? … **F=1**: Sẵn sang **F=0**: Chưa sẵn sàng
RCC_PLLConfig(A,B); (Lệnh cấu hình cho PLL)	
A: RCC_PLLSource_HSE_Div1 RCC_PLLSource_HSE_Div2 RCC_PLLSource_HSI_Div2 **B:** RCC_PLLMul_2 … RCC_PLLMul_16	**A:** Chọn nguồn xung đi vào PLL RCC_PLLSource_HSE_Div1 RCC_PLLSource_HSE_Div2 RCC_PLLSource_HSI_Div2 **B:** Chọn hệ số nhân tần số Nhân 2 … Nhân 16

RCC_HCLKConfig(A); (Lệnh cấu hình cho HCLK)	
A: RCC_SYSCLK_Div1 RCC_SYSCLK_Div2 RCC_SYSCLK_Div4 … RCC_SYSCLK_Div512	**A:** Chọn tần số cho HCLK Bằng System Clock chia 1 Bằng System Clock chia 2 Bằng System Clock chia 4 … Bằng System Clock chia 512
RCC_PCLK1Config(A); (Lệnh cấu hình cho PCLK1)	
A: RCC_HCLK_Div1 RCC_HCLK_Div2 RCC_HCLK_Div4 … RCC_HCLK_Div16	**A:** Chọn tần số cho PCLK1 Bằng HCLK chia 1 Bằng HCLK chia 2 Bằng HCLK chia 4 … Bằng HCLK chia 16
RCC_PCLK2Config(A); (Lệnh cấu hình cho PCLK2)	
A: RCC_HCLK_Div1 RCC_HCLK_Div2 RCC_HCLK_Div4 … RCC_HCLK_Div16	**A:** Chọn tần số cho PCLK2 Bằng HCLK chia 1 Bằng HCLK chia 2 Bằng HCLK chia 4 … Bằng HCLK chia 16
RCC_SYSCLKConfig(A); (Lệnh cấu hình System Clock)	
A: RCC_SYSCLKSource_HSI RCC_SYSCLKSource_HSE RCC_SYSCLKSource_PLLCLK	**A:** Chọn nguồn xung làm System Clock Lấy HSI làm System Clock Lấy HSE làm System Clock Lấy PLLCLK làm System Clock
unsigned char A; **A= RCC_GetSYSCLKSource();** (Lệnh đọc xem SYSCLK đang được cấp bởi nguồn nào)	
A: 0x00 0x04 0x08	**A:** mã của nguồn xung cấp cho SYSCLK System Clock được cấp từ HSI System Clock được cấp từ HSE System Clock được cấp từ PLLCLK
RCC_ClocksTypeDef A; **RCC_GetClocksFreq(&A);** (Lệnh đọc giá trị tần số)	
A: **A**.SYSCLK_Frequency **A**. HCLK_Frequency **A**. PCLK1_Frequency **A**. PCLK2_Frequency **A**. ADCCLK_Frequency	**A:** Tần số đọc được Tần số của System Clock Tần số của HCLK Tần số của PCLK1 Tần số của PCLK2 Tần số xung clock cấp cho ADC (Các giá trị tần số này đều là số 32 bit)

3.3 HƯỚNG DẪN CÀI ĐẶT, SỬ DỤNG CÁC NGUỒN XUNG CLOCK

3.3.1 Sử dụng HSE

Để sử dụng **HSE** ta gọi chương trình con **"SystemInit();"** ở đầu chương trình chính.

Ví dụ:

```
int main(void)
     {
          SystemInit();
          …
     }
```

Với thạch anh 8 Mhz thì chương trình con **"SystemInit();"** sẽ cấu hình:

- SYSCLK: 72 Mhz
- PCLK1: 36 Mhz
- PCLK2 : 72 Mhz

Nếu muốn cài đặt tần số thấp hơn mà vẫn sử dụng HSE ta mở file "**system_stm32f10x.c**" chỉnh lại từ hàng 110 đến hàng 115. Ví dụ để chọn tốc độ 48Mhz ta hiệu chỉnh như sau:

```
110      /* #define SYSCLK_FREQ_HSE    HSE_VALUE */
111      /* #define SYSCLK_FREQ_24MHz  24000000 */
112      /* #define SYSCLK_FREQ_36MHz  36000000 */
113         #define SYSCLK_FREQ_48MHz  48000000
114      /* #define SYSCLK_FREQ_56MHz  56000000 */
115      /* #define SYSCLK_FREQ_56MHz  72000000 */
```

Ta cũng có thể thay đổi hệ số nhân của PLL để được các loại tần số khác bằng cách chỉnh lại thông số "RCC_CFGR2_PLL2MUL**X**" (với X là số nguyên từ 2 đến 16) trong các chương trình con bên dưới thuộc file "**system_stm32f10x.c**":

- `static void SetSysClockTo24(void)`
- `...`
- `static void SetSysClockTo72(void)`

Tùy thuộc vào tần số mà ta đã chọn(ở bước chỉnh các dòng từ 110 đến 115) là bao nhiêu mà ta hiệu chỉnh thông số "RCC_CFGR2_PLL2MUL**X**" trong chương trình con tương ứng. Bằng việc hiệu chỉnh hệ số nhân của PLL ta có thể ép xung CPU lên đến 128 Mhz(không nên thử) khi dùng HSE 8Mhz kết hợp với PLL nhân 16.

3.3.2 Sử dụng HSI

Để bỏ thạch anh ngoài (tiết kiệm, mạch gọn, dễ thi công) ta có thể dùng HSI lúc này tần số tối đa chỉ đạt 64 Mhz.

Đoạn chương trình con sau giúp cấu hình HSI kết hợp với PLL tạo SYSCLK 64 Mhz.

```
void RCC_Configuration(void)
{
     RCC_DeInit();        // Reset RCC
     RCC_HSICmd(ENABLE); // Bật HSI
     while(RCC_GetFlagStatus(RCC_FLAG_HSIRDY) == RESET);
                          //  Chờ đến khi HSI khởi động xong
     FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);
                          // Cho phép Prefetch Buffer
        // FLASH_Latency_0, nếu  0      < HCLK < 24 MHz
        // FLASH_Latency_1, nếu  24 MHz < HCLK < 56 MHz
        // FLASH_Latency_2, nếu  56 MHz < HCLK < 72 MHz
        // Do chọn HCLK = SYSCLK = 64 Mhz nên chọn độ trễ 2
```

```
    FLASH_SetLatency(FLASH_Latency_2);
    RCC_HCLKConfig(RCC_SYSCLK_Div1);
            // Cấu hình HCLK = SYSCLK = 64 Mhz
    RCC_PCLK2Config(RCC_HCLK_Div1);
            // Cấu hình PCLK2 = HCLK = 64 Mhz
    RCC_PCLK1Config(RCC_HCLK_Div2);
            // Cấu hình PCLK1 = HCLK/2 = 32 Mhz
            // Vì PLCK1 tối đa chỉ đạt 36 Mhz
    RCC_PLLConfig(RCC_PLLSource_HSI_Div2, RCC_PLLMul_16);
            // Chọn HSI/2 là ngõ vào PLL hệ số nhân là 16:
            // 8/2*16=4*16=64 Mhz
    RCC_PLLCmd(ENABLE);  // Bật PLL
    while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET);
            // Chờ PLL khởi động xong
    RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);
           // Đặt PLLCLK làm SYSCLK
    while(RCC_GetSYSCLKSource() != 0x08);
           // Chờ cho đến khi đặt xong PLL làm SYSCLK
}
```

Khi sử dụng thay vì ở đầu chương trình chính ta gọi hàm **"SystemInit();"** thì ta gọi chương trình con"**RCC_Configuration()"** viết bên trên**.** Sau khi thực hiện xong chương trình con này CPU sẽ hoạt động ở tốc độ 64 Mhz.

Nếu muốn CPU hoạt động ở tần số thấp hơn thì ta thay đổi hệ số nhân từ **2** đến **16** (mỗi đơn vị thay đổi tương ứng với **4 Mhz**) trong lệnh:

```
RCC_PLLConfig(RCC_PLLSource_HSI_Div2, RCC_PLLMul_16);
```

3.4 ĐO TẦN SỐ CÁC NGUỒN XUNG CLOCK CƠ BẢN

Để kiểm tra xem việc cài đặt xung clock có đúng như ý muốn hay không ta tiến hành đọc tần số về rồi quan sát.

Đoạn chương trình sau hướng dẫn cách đọc tần số của các nguồn xung clock cơ bản về:

```
...
int main(void)
{
    RCC_ClocksTypeDef  CLK;       // Khai báo biến CLK
    RCC_Configuration();          // Cấu hình xung clock
    while(1)
    {
        RCC_GetClocksFreq(&CLK); // Đọc giá trị xung clock về
       /* Tiến hành hiển thị các biến sau để quan sát
        CLK. SYSCLK_Frequency   => Chứa giá trị SYSCLK
        CLK. HCLK_Frequency     => Chứa giá trị HCLK
        CLK. PCLK1_Frequency    => Chứa giá trị PCLK1
        CLK. PCLK2_Frequency  => Chứa giá trị PCLK2
        Chú ý tất cả các biến bên trên đều là kiểu số
              nguyên không dấu 32 bit*/
    }
}
```