STM32 ???? ? 2 ????????

? ????

1. 理解 **STM32F446** 的時脈架構 - 掌握振盪器、倍頻器、分頻器的概念
2. 學會配置 **Clock Tree** - 透過 CubeMX 設定各組態的時脈分配
3. 實現多時脈配置實驗 - 透過改變時脈速度觀察 LED 閃爍頻率變化

? ????????

????????

STM32 微控制器內部的每個元件（ CPU
、外設、計時器）都需要時脈信號來同步運作。時脈越快，處理速度越快，但功耗也越高。

STM32F446 ????

STM32F446 提供 **3** 種時脈源 ：

時脈源	頻率	精度	功耗	用途
HSI (內部高速振盪器 **)**	16 MHz	±2%	低	備用時脈、低功耗模式
HSE (外部高速振盪器 **)**	8 MHz	±1%	中等	主系統時脈（需晶振）
LSI (內部低速振盪器 **)**	32 kHz	±2%	很低	看門狗、 RTC

PLL ????

PLL (Phase-Locked Loop) 是一個電路，能將低頻時脈信號倍頻至高頻。例如：

- 輸入: 8 MHz（ HSE）
- 倍數: 21
- 輸出: 8 MHz × 21 = **168 MHz**

STM32F446 的最大系統時脈為 **180 MHz**，但實驗中常用 **168 MHz**（更穩定）。

?????

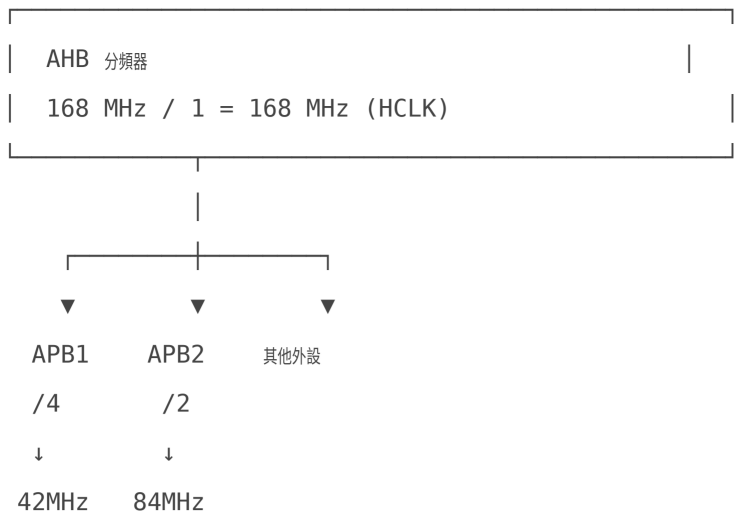
□□□□□ 168 MHz□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□	□□	□□□□
AHB (HCLK)	CPU □□□□□	168 MHz / 1 = 168 MHz
APB1 (PCLK1)	□□□□□ UART□ I2C□	168 MHz / 4 = 42 MHz
APB2 (PCLK2)	□□□□□ SPI□ ADC□	168 MHz / 2 = 84 MHz

?? Clock Tree ????

Clock Tree ????

```
┌─────────────────────────────────────────────┐
│  □□□□□                                       │
│  ├─ HSI (16 MHz) - □□□□□                    │
│  ├─ HSE (8 MHz)  - □□□□                      │
│  └─ LSI (32 kHz) - □□□□□□□               │
└─────────────────────────────────────────────┘
             │
             ▼
┌─────────────────────────────────────────────┐
│  PLL □□□□□□□                              │
│  □□□□□PLLM / PLLN / PLLP / PLLQ           │
│  □□□8 MHz → 168 MHz                          │
└─────────────────────────────────────────────┘
             │
             ▼
┌─────────────────────────────────────────────┐
│  □□□□□□□SYSCLK□                         │
│  ├─ HSI (16 MHz)                                │
│  ├─ HSE (8 MHz)                                 │
│  └─ PLL (168 MHz)    ← □□                     │
└─────────────────────────────────────────────┘
             │
             ▼
```



PLL ????

STM32F446 PLL 配置公式:

```
f_VCO = (f_HSE / PLLM) × PLLN
f_PLLCLK = f_VCO / PLLP
f_USB = f_VCO / PLLQ
```

常用配置（ **HSE = 8 MHz**）：

```
PLLM = 8      (預分頻: 8 MHz / 8 = 1 MHz)
PLLN = 168    (倍頻: 1 MHz × 168 = 168 MHz)
PLLP = 2      (主時脈分頻: 168 MHz / 2 = 84 MHz) □ □□□ 84 MHz
PLLP = 2      (若要 168 MHz, 應設定 PLLN = 336, PLLP = 2)
```

更精確的配置（目標 **168 MHz**）：

```
PLLM = 8      (預分頻: 8 MHz / 8 = 1 MHz)
PLLN = 336    (倍頻: 1 MHz × 336 = 336 MHz)
PLLP = 2      (主時脈分頻: 336 MHz / 2 = 168 MHz) □
```

?? CubeMX ??????

?? 1?????????

1. 開啟之前的專案或建立新專案
2. 雙擊 `.ioc` 檔案開啟 CubeMX
3. 點擊 **Clock Configuration** 標籤

?? 2??? HSE?8 MHz ???

1. 在「System Clock Mux」右側，點擊 **HSE** 下拉選單
2. 選擇 **Crystal/Ceramic Resonator**
3. 確認 HSE 頻率顯示為 **8 MHz**

?? 3??? PLL

1. 找到 **PLL** 配置區塊
2. 設定以下參數：
   - **PLLM**: 8
   - **PLLN**: 336
   - **PLLP**: 2（對應頻率應顯示 168 MHz）
   - **PLLQ**: 7（用於 USB，通常保持預設）
3. 若 PLLN 改為 336 後系統時脈仍未達 168 MHz，檢查 PLLP 是否為 2

?? 4??? PLL ???????

1. 在「System Clock Mux」中，選擇 **PLLCLK**
2. 確認 **SYSCLK** 顯示 **168 MHz**

?? 5??? AHB ???

1. 在「System Clock Mux」下方，找到 **AHB Prescaler**
2. 設定為 **1**（不分頻，HCLK = 168 MHz）

?? 6??? APB ???

1. **APB1 Prescaler** 設為 **4**
   - 結果：168 MHz / 4 = **42 MHz**（適合 UART、I2C）
2. **APB2 Prescaler** 設為 **2**
   - 結果：168 MHz / 2 = **84 MHz**（適合 SPI、ADC）

?? 7?????

- □ □□ CubeMX 下方的總結，應顯示：

```
SYSCLK:  168 MHz
HCLK:    168 MHz
PCLK1:    42 MHz
PCLK2:    84 MHz
```

? ????? - ??????

main.c - ???????

```c
/* STM32 Lesson 02 - Clock Configuration Verification
 * 功能: 測試不同的時脈配置，透過 LED 閃爍頻率變化來驗證
 * 難度: 中級
 */

#include "main.h"
#include "gpio.h"

/* 私有變數和函數 */
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
void LED_Blink(uint16_t delay_ms, uint8_t count);

/* 全域變數: 存儲當前系統時脈速度 */
uint32_t SystemCoreClock = 168000000;  /* 168 MHz */

int main(void)
{
  HAL_Init();

  /* 配置系統時脈為 168 MHz */
  SystemClock_Config();

  /* 初始化 GPIO */
  MX_GPIO_Init();

  /* 驗證系統時脈 */
```

```
  while (1)
  {
    /* 第一階段: 快速閃爍 (5 次, 100ms 間隔) - 表示 168 MHz */
    LED_Blink(100, 5);
    HAL_Delay(1000);  /* 暫停 1 秒 */

    /* 第二階段: 中速閃爍 (3 次, 200ms 間隔) - 表示配置完成 */
    LED_Blink(200, 3);
    HAL_Delay(1000);

    /* 第三階段: 慢速閃爍 (2 次, 500ms 間隔) - 表示系統運行 */
    LED_Blink(500, 2);
    HAL_Delay(2000);
  }
}

/**
  * @brief LED 間隔閃爍函數
  * @param delay_ms: 每次亮/暗的延遲時間 (毫秒)
  * @param count: 閃爍次數
  */
void LED_Blink(uint16_t delay_ms, uint8_t count)
{
  for (uint8_t i = 0; i < count; i++)
  {
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);    /* LED 亮 */
    HAL_Delay(delay_ms);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);  /* LED 暗 */
    HAL_Delay(delay_ms);
  }
}

/**
  * @brief 系統時脈配置函數
  * 配置 HSE (8 MHz) → PLL 倍頻至 168 MHz
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
```

```
  /* 配置電源調節器 */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

  /* 配置 RCC 振盪器 */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 8;      /* 預分頻: 8 MHz / 8 = 1 MHz */
  RCC_OscInitStruct.PLL.PLLN = 336;    /* 倍頻: 1 MHz × 336 = 336 MHz */
  RCC_OscInitStruct.PLL.PLLP = 2;      /* 主輸出分頻: 336 MHz / 2 = 168 MHz */
  RCC_OscInitStruct.PLL.PLLQ = 7;      /* USB 時脈: 336 MHz / 7 = 48 MHz */

  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /* 配置時脈分頻器 */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLK
                              | RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;      /* HCLK = 168 MHz */
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;       /* PCLK1 = 42 MHz */
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;       /* PCLK2 = 84 MHz */

  /* Flash 延遲設定（高速時脈需要更多等待週期） */
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
  {
    Error_Handler();
  }

  /* 配置 SysTick 用於系統計時 */
  HAL_SYSTICK_Config(SystemCoreClock / 1000);  /* 1ms 中斷一次 */
}

/**
  * @brief GPIO 初始化函數
```

```
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  __HAL_RCC_GPIOA_CLK_ENABLE();

  GPIO_InitStruct.Pin = GPIO_PIN_5;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
}

/**
  * @brief 系統錯誤處理
  */
void Error_Handler(void)
{
  while (1)
  {
    HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);  /* 快速閃爍表示錯誤 */
    HAL_Delay(100);
  }
}
```

? ?????

????

 LED 閃爍模式序列 :

1. **5** 次快速閃爍 （100ms） → 系統時脈 168 MHz 正常
2. 暫停 **1** 秒
3. **3** 次中速閃爍 （200ms） → 時脈配置完成
4. 暫停 **1** 秒
5. **2** 次慢速閃爍 （500ms） → 系統穩定運行
6. 暫停 **2** 秒後重複

???????

問題	原因	解決方案
LED 不閃爍	PLL 配置失敗	檢查 CubeMX Clock Configuration，確認 PLLN 為 336，PLLP 為 2
閃爍節奏不對	HAL_Delay 計算錯誤	確認 SysTick 配置正確，系統時脈應為 168 MHz
編譯警告 「implicit conversion」	時脈值類型不匹配	確認 `SystemCoreClock` 變數型別為 `uint32_t`
程式掛起	Flash 延遲不足	檢查 `FLASH_LATENCY_5` 是否設定，高速時脈需更多等待週期

??????

方法 **1**：**UART** 輸出驗證（後續課程） 透過串口列印 `SystemCoreClock` 值

方法 **2**：邏輯分析儀 測量 PA5 的方波頻率，應為固定值

方法 **3**：計算驗證

- 若 `HAL_Delay(100)` 實際延遲 100ms，時脈正確

? ????

Flash ?????FLASH_LATENCY?

快閃記憶體存取速度是有限的。當系統時脈增加時，需要增加等待週期（ Latency）。

系統時脈	建議 **LATENCY**
0-30 MHz	0
31-60 MHz	1
61-90 MHz	2
91-120 MHz	3
121-150 MHz	4
151-168 MHz	5

? ????

?? 1????????

修改代碼，使用 **HSI (16 MHz)** 而非 PLL:

```
/* 使用 HSI 代替 PLL（低功耗模式） */
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
RCC_OscInitStruct.HSIState = RCC_HSI_ON;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_OFF;  /* 關閉 PLL */

RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
```

觀察 LED 閃爍速度是否變慢（因為系統時脈從 168 MHz 降至 16 MHz）

?? 2????????

實現在運行時在不同時脈間切換的函數

? ????

???? - ? 3 ??GPIO ??????????????

第 3 節將學習:

- **GPIO** 輸入配置 - 按鍵檢測原理
- 中斷系統（**EXTI**） - 事件驅動編程
- 軟體去彈跳 - 解決按鍵抖動問題

????

- □□ STM32F446 Reference Manual - RCC章節__
- □□ PLL 倍頻計算器（線上工具）_____

□ □□□□□□□□□□□□ **GPIO** 進階應用 □□

Revision #2
Created 2026-04-01 02:06:24 UTC by TaipeiTechRacing

Updated 2026-04-06 06:24:02 UTC