Register တွေကိုအသုံးပြုပြီး clock speed ပြောင်းခြင်း

Clock speed ဆိုတာ Microcontroller တစ်ခုကိုမောင်းနှင်နေတဲ့ clock ရဲ့ speed ကို ဆိုလိုခြင်း ဖြစ်ပါတယ်။ Default clock source နဲ့ speed ကတော့ board တစ်ခုနဲ့တစ်ခု မတူပါဘူး။ STM32F446 မှာဆိုရင် default ကတော့ HSI (High Speed Internal) 16 MHz ပဲဖြစ်ပါတယ်။ HSI ရဲ့မူလတန်ဖိုးကတော့ 16 MHz ပဲဖြစ်ပါတယ်။ ဒါပေမယ့် STM32 တွေမှာပါတဲ့ PLL ကိုအသုံးပြုပြီး 180 MHz အထိ တင်လို့ရပါတယ်။ ဒီ article မှာတော့ HSI ကိုအသုံးပြုပြီး clock speed 80 MHz ရအောင် လုပ်ပြပေးသွားမှာ ဖြစ်ပါတယ်။ STM32 Microcontroller တွေကို clock speed ပြောင်းရာတွင် အဓိကအချက် (၈) ချက် လိုအပ်ပါတယ်။ ။

(1) Enable clock source and wait for ready

အရင်ဦးဆုံး ကိုယ်သုံးမယ့် clock soure ကို enable လုပ်ပြီး ready ဖြစ်တာကို စောင့်ရမှာ ဖြစ်ပါတယ်။ ဒီ article မှာတော့ HSI ကို သုံးမှာဖြစ်လို့ HSI ကို enable လုပ်ပါတယ်။

```
// Enable HSI and wait until it is ready
RCC->CR |= RCC_CR_HSION;
while(!(RCC->CR & RCC_CR_HSIRDY));
```

(2) Set PWREN and VOS

Power interface နဲ့ voltage regulator ကိုလည်း on ပေးရမှာ ဖြစ်ပါတယ်။

```
// Set Power Enable Clock and Voltage Regulator
RCC->APB1ENR |= RCC_APB1ENR_PWREN;
PWR->CR |= PWR_CR_VOS;               // Scale 1 mode (reset value)
```

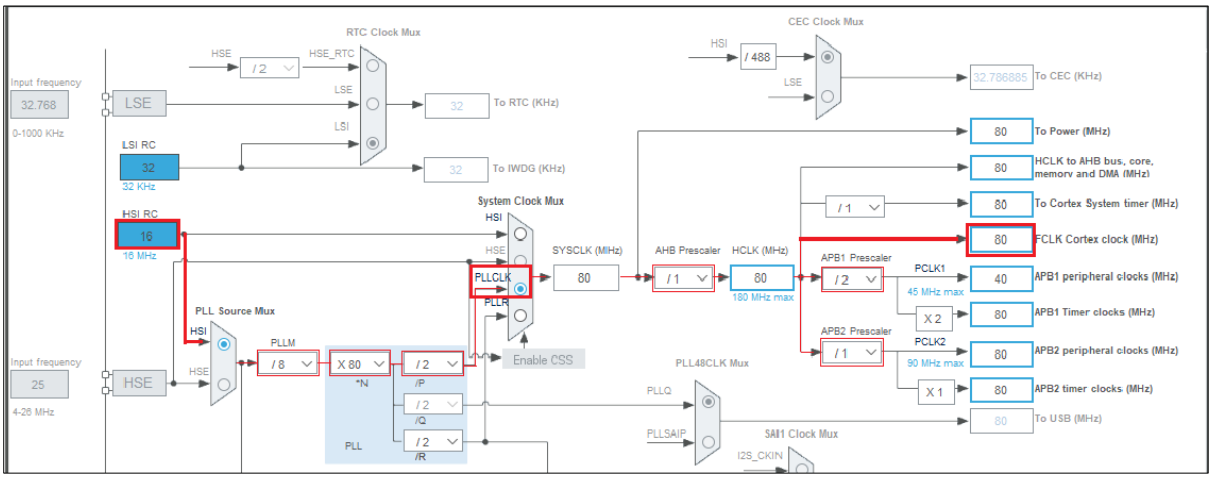
(3) Configure Flash

Flash prefetch buffer ဆိုတာ instruction တစ်ခု လုပ်ဆောင်နေစဉ် နောက် instruction တစ်ခုကို ကြိုပြီးတော့ fetch လုပ်တာ ဖြစ်ပါတယ်။ CPU အလုပ်လုပ်တာ ပိုမြန်စေပါတယ်။ Power စစချင်းမှာ flash latency ဆိုတာရှိပါတယ်။ ကိုယ်သုံးမဲ့ clock speed အပေါ် wait state တွေထားရပါတယ်။ ဒီနေရာမှာ 80 MHz ကိုသုံးမှာဖြစ်လို့ 3 wait state ထားရပါမယ်။ Wait sate အကြောင်းကို Reference Manual မှာ Embedded Flash memory ဆိုတဲ့ ခေါင်းစဉ်ကြီး အောက်မှာ ပြထားပါတယ်။

```
FLASH->ACR |= FLASH_ACR_PRFTEN;      // Activate prefetch buffer
// 3 Wait State (72 < HCLK = 96)
FLASH->ACR &= ~(FLASH_ACR_LATENCY);
FLASH->ACR |= 0x3U;
```

(4) Configure Main PLL

အဆင့် (၄) နဲ့ (၅) အတွက် ပိုမိုလွယ်ကူအောင် CubeMX ကထုတ်ပေးတဲ့ Clock Diagram ကို ကြည့်ရအောင်။ CubeMX က ကိုယ်ကြိုက်တဲ့ speed ကို ထည့်လိုက်ရင် သူ့ဟာသူ သင့်တော်မယ့် prescalar တွေကို ရွေးပေးပါတယ်။ (CubeMX မရှိရင် Reference Manual ကိုကြည့်လို့ရပါတယ်။ Reset and clock control (RCC) ဆိုတဲ့ခေါင်းစဉ်ကြီးအောက်က Clocks ဆိုတဲ့ခေါင်းစဉ်ငယ် အောက်မှာ Clock tree ဆိုတဲ့ ပုံရှိပါတယ်။)



ပြီးရင်တော့ ပုံမှာပြထားတဲ့အတိုင်း အဆင့်ဆင့်လုပ်ဆောင်ပါတယ်။ အရင်ဦးဆုံး PLL ကို မောင်းမယ့် clock source ရွေးမယ်၊ PLLM, PLLN, PLLP တို့ကို အောက်မှာပြထားတဲ့အတိုင်း သတ်မှတ်ပေးရမှာဖြစ်ပါတယ်။

```
// Select HSI as PLL entry clock source
RCC->PLLCFGR |= RCC_PLLCFGR_PLLSRC_HSI;
// VCO input frequency = PLL input frequency / PLLM[5:0] (VCO = 8/8)
RCC->PLLCFGR &= ~(0x3FU);              // clear RCC_PLLCFGR [5:0]
RCC->PLLCFGR |= 8;                     // VCO input = 16/8 = 2 MHz
// VCO output frequency = VCO input frequency × PLLN[8:0]
RCC->PLLCFGR &= ~(0x7FC0U);            // Clear RCC_PLLCFGR [14:6]
RCC->PLLCFGR |= 80 << 6;               // VCO output = 2*80 = 160 MHz
// PLL output clock frequency = VCO frequency / PLLP[1:0]
RCC->PLLCFGR &= ~(3U << 16);           // PLLCLK output = 160/2 = 80 MHz
```

ဒီအထိကတော့ ပုံအလယ်လောက်မှာရှိတဲ့ SYSCLK (MHz) ဆိုတဲ့အထိပဲ ဖြစ်ပါတယ်။

(5) Configure prescalars

ပုံရဲ့ညာဘက်ဆုံးမှာ အနီဝိုင်းထားတဲ့ထိ သွားရမှာဖြစ်ပါတယ်။ FCLK Core clock (MHz) ဆိုတာ microcontroller ရဲ့ main (သို့) core clock ဖြစ်ပါတယ်။ Core clock ရဲ့ maximum frequency က 180 MHz ဖြစ်ပြီး၊ APB1 နဲ့ APB2 တို့ရဲ့ maximum frequency ကတော့ 45 MHz နှင့် 90 MHz တို့ပဲ ဖြစ်ပါတယ်။ အဲဒါကြောင့် APB1 ကို maximum မကျော်အောင် 2 နဲ့ ပြန်စားရပါတယ်။

```
RCC->CFGR &= ~(0xFU << 4);             // system clock not divided (AHB/1)
```

```
RCC->CFGR &= ~(0x7U << 10);
RCC->CFGR |= RCC_CFGR_PPRE1_2;        // APB1 = AHB/2
RCC->CFGR &= ~(0x7U << 13);           // APB2 = AHB/1
```

(6) Enable PLL and wait for ready

Initialization တွေအကုန်ပြီးသွားတော့မှ PLL ကို on ပြီး ready ဖြစ်တာကို စောင့်ရပါမယ်။

```
RCC->CR |= RCC_CR_PLLON;
while(!(RCC->CR & RCC_CR_PLLRDY));
```

(7) Select the clock source for system clock

နံပါတ် (၄) မှာရွေးထားတဲ့ clock source က PLL ကိုမောင်းနှင်မယ့် clock ဖြစ်ပြီး အခုရွေးမှာကတော့ system ကိုမောင်းနှင်မယ့် clock source ဖြစ်ပါတယ်။ ပုံမှာပြထားတဲ့အတိုင်း PLLCLK (or) PLL_P ကိုရွေးပါမယ်။ ready ဖြစ်တာကို စောင့်ပါမယ်။

```
RCC->CFGR |= RCC_CFGR_SW_1;           // PLL_P selected as system clock
while(!(RCC->CFGR & RCC_CFGR_SWS_PLL));
```

(8) Enable system clock update

နောက်ဆုံးမှာ အောက်မှာရေးထားတဲ့ function ကိုတော့ တစ်ချို့ board တွေမှာ ခေါ်ဖို့မလို ပေမယ့် တချို့တွေမှာ ခေါ်ဖို့လိုပါတယ်။ မဟုတ်ရင် ကိုယ်ရေးထားတဲ့ clock initialization တွေကို update မလုပ်ပဲ default value အတိုင်းပဲ ဖြစ်နေတတ်ပါတယ်။

```
SystemCoreClockUpdate();
```

ဒီလိုအဆင့်ဆင့်လုပ်ဆောင်ခြင်းအားဖြင့် နဂိုမူလ 16 MHz ကနေ 80 MHz သို့ ကိုယ်တိုင်ကိုယ်ကျ ပြောင်းနိုင်ပြီ ဖြစ်ပါတယ်။ တကယ် 80 MHz ဖြစ်မဖြစ် board ပေါ်ပါပြီးသား Led ကိုသုံးပြီး systick delay နဲ့ စမ်းကြည့်ရအောင်။ အရင်ဆုံး board ပေါ်မှာရှိတဲ့ Led ရဲ့ pin နံပါတ်ကို အရင်ရှာပါ။ အခုကျနော်သုံးနေတဲ့ F446RE မှာ ဆိုရင် Led က PA5 မှာရှိပါတယ်။ အဲဒါကို output အနေနဲ့ initialize လုပ်ကြရအောင်။

```
void LED_Init(void)
{
	// LED is in PA5 for STM32F446RE //
	RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN; /* MAX 180 MHz */
	GPIOA->MODER &= ~(3U<<10);           // Clear bit 11 and 10
	GPIOA->MODER |= (1U<<10);            // Set PA5 as Output Function
	GPIOA->OTYPER &= ~(1U<<5);           // PA5: Output push-pull
	GPIOA->OSPEEDR &= ~(3U<<10);         // Clear bit 11 and 10
	GPIOA->OSPEEDR |= (1U<<11);          // PA5: Fast speed
	GPIOA->PUPDR &= ~(3U<<10);           // PA5: No pull-up/pull-down
}
```

Clock speed 80 MHz ဆိုတာ တစ်စက္ကန့်ကို 80,000,000 ticks ကို ဆိုလိုတာပါ။ အဲဒါကြောင့် delay ကို millisecond နှုန်းနဲ့ ထားမှာမို့လို့ Systick Load မှာ 80,000 ထားပါမယ်။

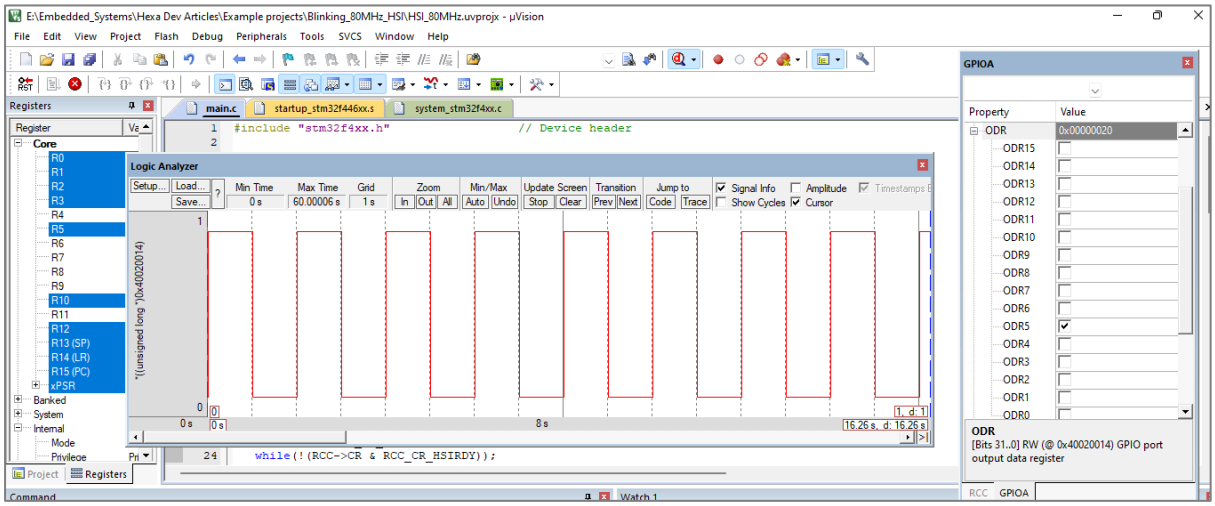
ဒါဆိုရင် delay function မှာ 1,000 ထားလိုက်ရင် 80,000,000 ပြည့်သွားပြီး ၁ စက္ကန့်နှုန်း delay ရရှိမှာဖြစ်ပါတယ်။

```
void systickDelayMs(int delay)              // see User Guide about systick
{
	// Any value between 1 - 16,777,215 (2^24 - 1)
	SysTick->LOAD = 80000-1;                // 80 Mhz means 80,000 clock per Ms
	SysTick->VAL = 0;                       // Clear Current Value Register
	SysTick->CTRL |= 0x05;                  // systick counter enabled

	for (int i = 0; i < delay; i++) {
		// Wait until the COUNT flag is set
		while ((SysTick->CTRL & 0x10000) == 0);
	}
	SysTick->CTRL = 0;                      // stop the timer
}
```

ပြီးရင်တော့ main မှာ ခေါ်သုံးလိုက်ရုံပါပဲ။

```
int main(void)
{
	System_Clock_Init();
	LED_Init();

	while(1)
	{
		GPIOA->ODR ^= 1 << 5;
		systickDelayMs(1000);
	}
}
```

တစ်စက္ကန့်နှုန်းနဲ့ Led က blink နေမှာ ဖြစ်ပါတယ်။ တစ်စက္ကန့် တိကျ၊ မတိကျ သိချင်ရင်တော့ Keil uVision မှာပါတဲ့ Logic Analyzer ကိုကြည့်ပြီး စမ်းကြည့်နိုင်ပါတယ်။



၁ စက္ကန့်နှုန်းနဲ့ on လိုက် off လိုက် ဖြစ်နေတာ ဖြစ်ပါတယ်။ Keil uVision ရဲ့ logic analyzer သုံးနည်းကို https://youtu.be/KUZS8xWWI2Y မှာလေ့လာနိုင်ပါတယ်။ ။