Спецификатор volatile – принуждает компилятор не оптимизировать ваш код. Что это значит? Допустим, по каким-то причинам, компилятор решил, что ваша переменная не изменяется, тогда он может поместить её в кэш и брать значение из него. Если ticks\_delay не будет обновляться, то программа застрянет в функции delay().

MCO – output – для тактирования чего-то другого. На наш мк приходит с программатрора, который как раз с помощью MCO и отправляет меандр.

Нужно понять, как настроить pll и еще нужно делители для разных шин настроить, они не могут на высокой частоте работать.

The AHB and the APB domains maximum frequency is 48 MHz.HCLK for AHB or PCLK for APB

PllSRC – (RCC\_CFGR) Задачет источник питания  
PLLXTPRE – HSE divider for PLL input clock.   
PLLMUL[3:0]: PLL multiplication factor

HSE – внешний, высокоскоростной  
HIS – внутренний, высокоскоростной

# Внешний генератор (HSE bypass)

     В этом режиме МК может тактироваться от внешнего источника тактового сигнала, который может иметь частоту до 32 МГц. Для выбора этого режима необходимо установить биты HSEBYP и HSEON регистра RCC\_CR. Внешний тактовый сигнал может представлять собой как меандр, так и синусоиду и треугольник и иметь скважность 40-60%.

  HSI-генератор.

**HSI**–это встроенный RC-генератор на 8 МГц, который может быть использован как напрямую в качестве источника системной тактовой частоты, так и в качестве тактового сигнала **PLL (ФАПЧ)**. Данный генератор запускается сразу же после подачи питания на микроконтроллер, однако, как и все RC-генераторы, обладает очень маленькой точностью (даже с учётом заводской калибровки – точность 1% при 25°C).

     При переходе **HSI**в установившийся режим работы устанавливается флаг (бит) HSIRDY в регистре RCC\_CR. Включение **HSI-**генератора выполняется установкой бита HSION в регистре RCC\_CR.

To modify the PLL configuration, proceed as follows:

1. Disable the PLL by setting PLLON to 0.

2. Wait until PLLRDY is cleared. The PLL is now fully stopped.

3. Change the desired parameter.

4. Enable the PLL again by setting PLLON to 1.

5. Wait until PLLRDY is set. An interrupt can be generated when the PLL is ready, if enabled in the Clock interrupt register (RCC\_CIR).

The PLL output frequency must be set in the range 16-48 MHz.

# Clock control register

PLLRDY 0,1 – PLL clock ready flag (software)

PLLON – PLL enable (by software, cleared by hardware. Can not be reset if the PLL clock is used)

# Clock config register

SWS[1:0]: System clock switch status

Set and cleared by hardware to indicate which clock source is used as system clock.

10: PLL used as system clock

11: HSI48 oscillator used as system clock (when available)

0: PLL is divided by 2 for MCO 1: PLL is not divided for MCO

MCO - 0111: PLL clock selected (divided by 1 or 2, depending on PLLNODIV)

System core cl update

Настроить pll

1 подключить

#include "stm32f0xx.h" // Device header

#define SETLEDred() (GPIOC->BSRR = GPIO\_BSRR\_BS\_6)

#define SETLEDgreen() (GPIOC->BSRR = GPIO\_BSRR\_BS\_9)

#define SETLEDblue() (GPIOC->BSRR = GPIO\_BSRR\_BS\_7)

#define SETLEDyellow() (GPIOC->BSRR = GPIO\_BSRR\_BS\_8)

#define RESETLEDred() (GPIOC->BSRR = GPIO\_BSRR\_BR\_6)

#define RESETLEDgreen() (GPIOC->BSRR = GPIO\_BSRR\_BR\_9)

#define RESETLEDblue() (GPIOC->BSRR = GPIO\_BSRR\_BR\_7)

#define RESETLEDyellow() (GPIOC->BSRR = GPIO\_BSRR\_BR\_8)

#define ResetAllLEDs() (GPIOC->BSRR = (GPIO\_BSRR\_BR\_6|GPIO\_BSRR\_BR\_7|GPIO\_BSRR\_BR\_8|GPIO\_BSRR\_BR\_9))

#define FlashLock() (FLASH->CR |= FLASH\_CR\_LOCK)

uint32\_t tick = 0;

uint8\_t flag\_yellow\_LED = 0;

void PLLConfig(void);

void LEDConfig(void);

uint8\_t Trig(uint8\_t flag);

void DefineYellow(void);

void SysTick\_Handler(void)

{

tick++;

}

int main(void)

{

SysTick\_Config(SystemCoreClock/1000); /\* 1000Hz tick \*/

uint32\_t temp = 0;

//SystemCoreClockUpdate();

PLLConfig();

LEDConfig();

SETLEDgreen();

while(1)

{

if(tick>1000)

{

tick = 0;

flag\_yellow\_LED = Trig(flag\_yellow\_LED);

DefineYellow();

}

}

}

void PLLConfig(void)

{

if ((RCC->CFGR & RCC\_CFGR\_SWS) == RCC\_CFGR\_SWS\_PLL) /\* test if pll is a system clock \*/

{

RCC->CFGR &= (uint32\_t) (~RCC\_CFGR\_SW); /\* seleck hsi is system cl \*/

while ((RCC->CFGR & RCC\_CFGR\_SWS) != RCC\_CFGR\_SWS\_HSI){} //3

}

RCC->CFGR |= RCC\_CFGR\_PLLSRC\_HSI\_DIV2;

RCC->CR &= (uint32\_t)(~RCC\_CR\_PLLON); /\* disable pll \*/

while((RCC->CR & RCC\_CR\_PLLRDY) != 0) {}

RCC->CFGR |= RCC\_CFGR\_PLLMUL2; /\* (6) \*/

RCC->CR |= RCC\_CR\_PLLON; /\* (7) \*/

while((RCC->CR & RCC\_CR\_PLLRDY) == 0){} /\* (8) \*/

RCC->CFGR |= (uint32\_t) (RCC\_CFGR\_SW\_PLL); /\* (9) \*/

while ((RCC->CFGR & RCC\_CFGR\_SWS) != RCC\_CFGR\_SWS\_PLL){} /\* (10) \*/