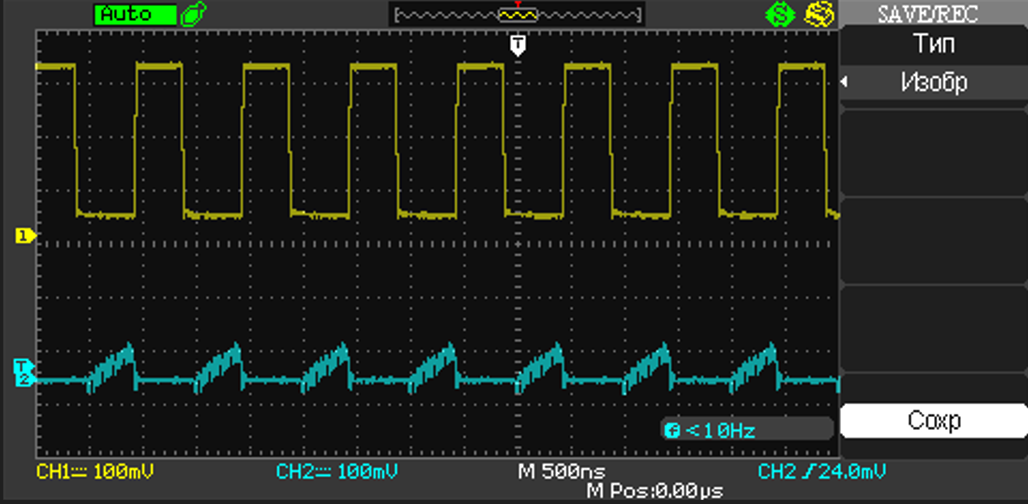
Вариант 2



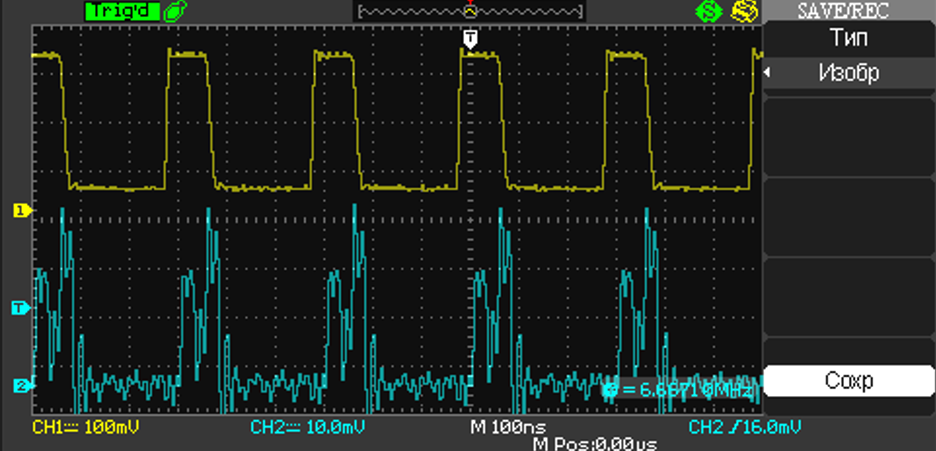
Цель работы:

Привитие практических навыков по работе с ИСР Keil, технической документацией, подсистемой тактирования и ПВВ микроконтроллера, библиотекой CMSIS.

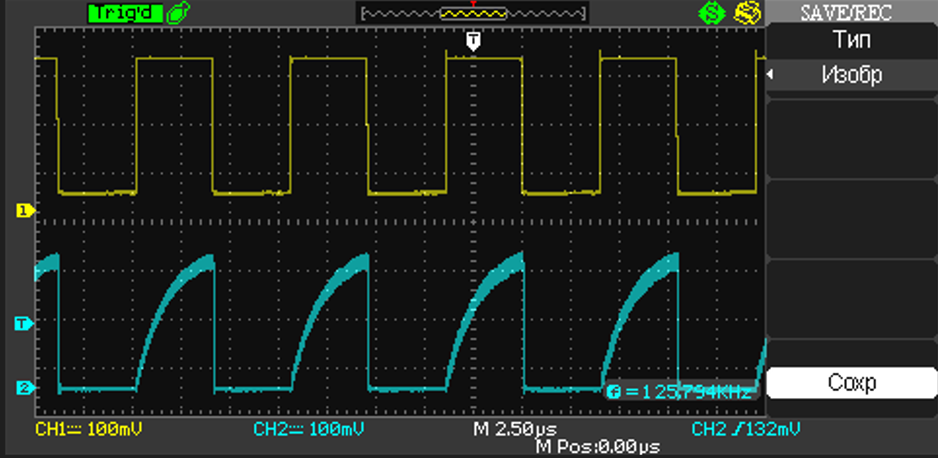
Осциллограммы на линиях PD0, PA9 при отключенной оптимизации компилятора



Осциллограммы на линиях PD0, PA1 при подключенной оптимизации компилятора



Осциллограммы на линиях PD0, PA9 при подключенной оптимизации компилятора и с задержкой



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Линия | Измеряемый параметр | | | | |
| Частота следования импульсов, КГц | Амплитуда, В | Длительность импульса, нс | Время нарастания фронта, нс | Время спада среза, нс |
| Оптимизация компилятора отключена ‘Level 0(-O0)’ | | | | | |
| PA9 | 1000 | 0,1 | 500 | 0.1 | 0.1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PD0 | 1000 | 0,1 | 500 | 500 | 0.1 |
| Оптимизация компилятора включена ‘Level 3(-O3)’ | | | | | |
| PA9 | 6666 | 0,35 | 100 | 10 | 1 |
| PD0 | 6666 | 0,04 | 100 | 50 | 50 |
| Добавлена задержка 8 тактов | | | | | |
| PA9 | 125 | 0,230 | 3000 | 0.1 | 0.1 |
| PD0 | 125 | 0,250 | 3300 | 3300 | 0.1 |

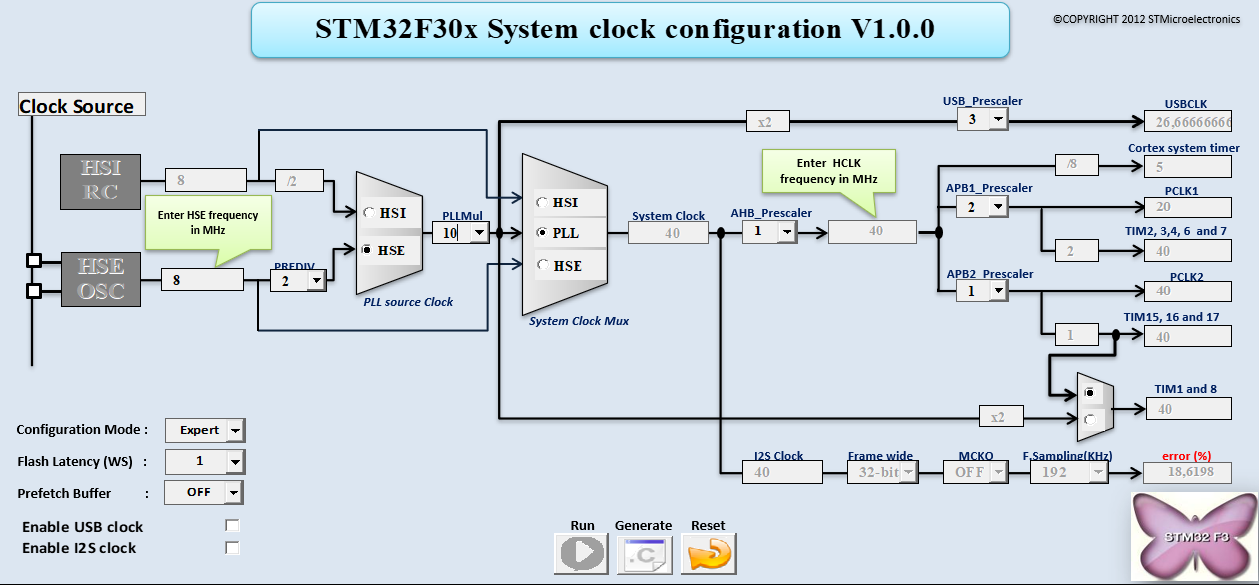
## Расчёт соотношения частоты HCLK с частотой переключения выводов.

Посчитаем соотношение частоты HCLK= 24 МГц и частоты переключения: а) без оптимизации 24000 000 / 2 340 000 = 10,25

б) с включенной оптимизацией компилятора 24 000 000 / 2 340 000 = 10,25

Снимок экрана настройки частоты согласно варианту с помощью инструмента

STM32F30x\_Clock\_Configuration\_V1.0.0.xls [31].



## Код

#include "RTE\_Components.h"

#include CMSIS\_device\_header

void delay(volatile uint32\_t count)

{

    while (count--)

        \_\_NOP();

}

int main(void)

{

    // Вспомогательные переменные

    volatile uint32\_t StartUpCounter = 0, HSEStatus = 0;

    SET\_BIT(RCC->CR, RCC\_CR\_HSEON); // включаем HSE

    do

    { // ждем вхождения в работу HSE

        HSEStatus = RCC->CR & RCC\_CR\_HSERDY;

        StartUpCounter++;

    } while ((HSEStatus == 0) && (StartUpCounter != 0x5000));

    // если за 0x5000 итераций, HSE не запустился, то проблемы в аппаратуре

    if ((RCC->CR & RCC\_CR\_HSERDY) != RESET)

    {

        // настраиваем буфер FLASH предварительной выборки команд

        FLASH->ACR = FLASH\_ACR\_LATENCY\_1;

        // настройка PLL на 24 МГц = 8 МГц(HSE)/4\*12

        // сначала выключаем чтобы изменить биты PLL, после настройки включим

        CLEAR\_BIT(RCC->CR, RCC\_CR\_PLLON);

        // PLLSRC=HSE/2

        // PLLCLK=PLLSRC \* 10

        RCC->CFGR2 |= (uint32\_t)RCC\_CFGR2\_PREDIV\_DIV2;

        RCC->CFGR |= (uint32\_t)(RCC\_CFGR\_PLLSRC\_HSE\_PREDIV | RCC\_CFGR\_PLLMUL10);

        SET\_BIT(RCC->CR, RCC\_CR\_PLLON); // включаем PLL

        // ждём запуск и стабилизацию PLL

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

        {

        }

        // выбираем выход PLL источником тактирования МК

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

        // Ожидаем установки PLL источником тактирования МК

        while ((RCC->CFGR & (uint32\_t)RCC\_CFGR\_SWS) != (uint32\_t)RCC\_CFGR\_SWS\_PLL)

        {

        }

    }

    else

    {

        while (1)

        {

        }

    } // HSE не запустился

    // контролируем частоту тактирования 4/1=4 МГц

    SystemCoreClockUpdate(); // устанавливается в глобальной переменной SystemCoreClock

    // Настройка MCO на PLLCLK/2

    SET\_BIT(RCC->CFGR, RCC\_CFGR\_MCO\_PLL);

    SET\_BIT(RCC->AHBENR, RCC\_AHBENR\_GPIOAEN); // разрешаем тактирование GPIOA

    // для PA8 устанавливаем Alternate function mode

    SET\_BIT(GPIOA->MODER, GPIO\_MODER\_MODER8\_1);

    // для восьмой линии выбираем AF0 режим работы MCO

    CLEAR\_BIT(GPIOA->AFR[1], GPIO\_AFRH\_AFRH0\_Msk);

    // разрешаем тактирование GPIOD

    SET\_BIT(RCC->AHBENR, RCC\_AHBENR\_GPIODEN);

    // устанавливаем работу линий PD0,PA9 на вывод

    SET\_BIT(GPIOD->MODER, GPIO\_MODER\_MODER0\_0);

    SET\_BIT(GPIOA->MODER, GPIO\_MODER\_MODER9\_0);

    // линию PA9 в режим вывода с открытым стоком

    SET\_BIT(GPIOA->OTYPER, GPIO\_OTYPER\_OT\_9);

    // подтягиваем PA9 к питанию Pull-up

    SET\_BIT(GPIOA->PUPDR, GPIO\_PUPDR\_PUPDR9\_0);

    while (1)

    {

        // устанавливаем (1) на выходе линий PD0,PA9

        GPIOD->BSRR = GPIO\_ODR\_0;

        GPIOA->BSRR = GPIO\_ODR\_9;

        // delay(10); // задержка

        // сбрасываем в (0) выходы линий PD0,PA9

        GPIOD->BRR = GPIO\_ODR\_0;

        GPIOA->BRR = GPIO\_ODR\_9;

        // delay(10);

    }

}