Основные библиотеки для микроконтроллеров stm32fxxx

Библиотека CMSIS

В микроконтроллере все управляющие регистры расположены по определённым адресам. Чтобы не держать в голове, по какому адресу, какой регистр расположен, была написана библиотека CMSIS. Эта библиотека сопоставляет адреса регистров с определёнными названиями, говорящими о назначении того или иного регистра.

Библиотека StdPeriph

Библиотека StdPeriph создана для упрощения работы с периферийными устройствами МК. Эта библиотека позволяет производить настройку нужного периферийного устройства, не обращаясь к управляющим регистрам напрямую. Всё, что требуется от программиста, это заполнить соответствующую структуру параметров и вызвать функцию инициализации с входным параметром в виде этой структуры. Это позволяет производить настройку системы, не обращаясь к технической документации на используемый микроконтроллер.

Ещё одним плюсом использования данной библиотеки является то, что она уменьшает зависимость написанного программистом кода от используемого микроконтроллера, т.к. выходные функции библиотеки по возможности одинаковы для любого микроконтроллера.

Стоит отметить, что функции библиотеки StdPeriph основаны на библиотеке CMSIS и без неё библиотека не работает. Отсюда следует один недостаток этой библиотеки. Программа, написанная с её помощью, работает зачастую существенно медленнее, чем написанная только с использованием CMSIS.

Обычно имеет смысл сделать начальную инициализацию на StdPeriph, а в случае необходимости срочной перенастройки, использовать CMSIS. Т.е. необходимо свободное владение обоими библиотеками.

Создание проекта в keil uVission

После установки и запуска программы, мы получаем окно следующего вида, как показано на рис 1.

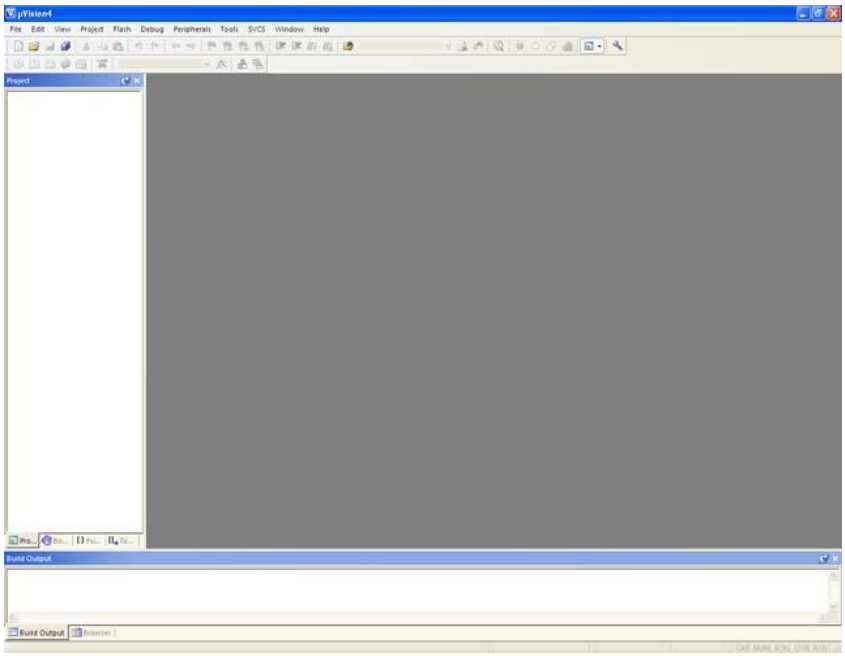

Рис 1

Для начала создания проекта жмём *Project -> New uVision project*. Появится окно для сохранения созданного проекта. Придумайте ему осмысленное название и поместите в отдельную папку. Жмите «Сохранить». Появится окно выбора микроконтроллера как на рис 2:

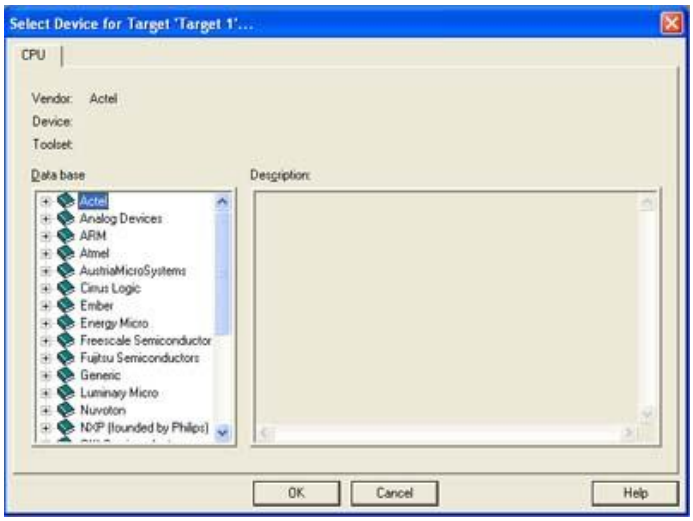

Рис 2

Выбираем требуемый микроконтроллер (название микроконтроллера можно прочитать на нём). Нажимаем «Ок». Появится окно с вопросом, создать ли стартовый файл (Рис 3). Нажимаем «Да».

Рис 3

В итоге проект должен принять вид, похожий на то, что представлено на рис 4.

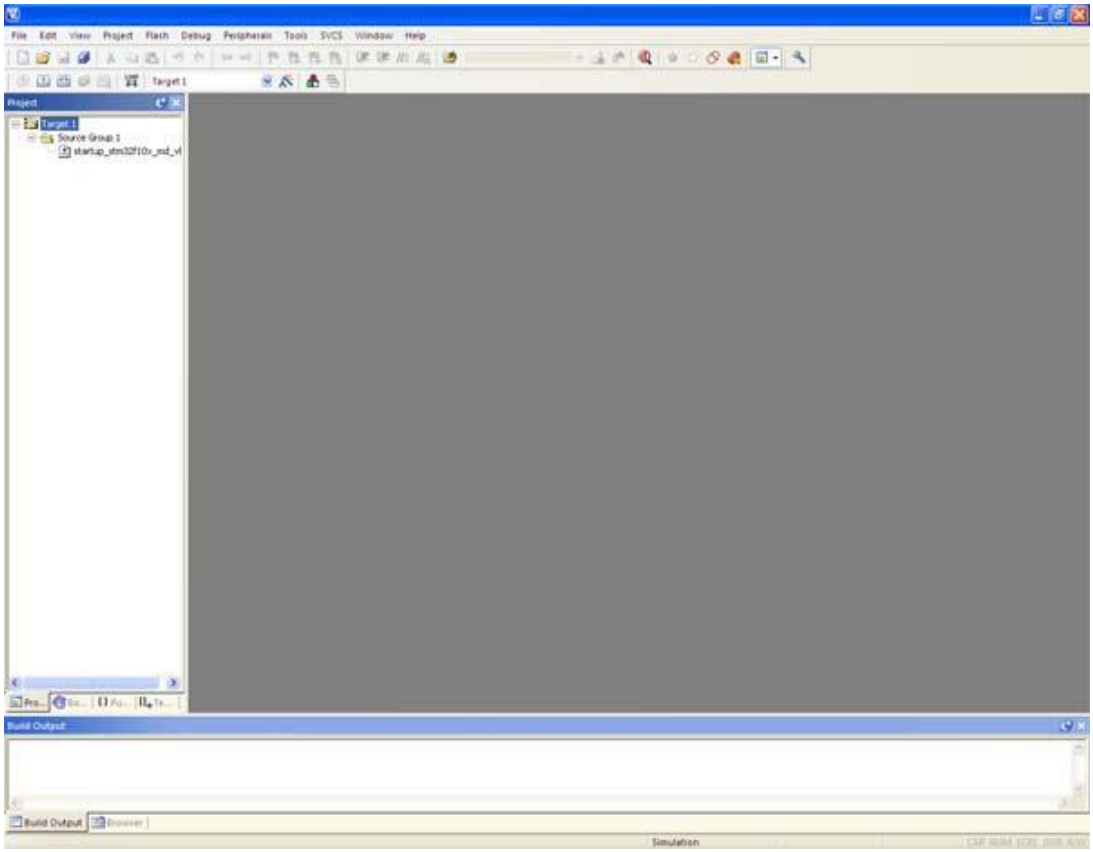

Рис 4

Обратите внимание на вкладку «Target 1» слева. Там хранятся файлы проекта. Здесь же вы будете подключать все необходимые библиотеки. Прежде всего подключим пользовательский файл main.c. Для этого в keil слева (окно project) щелкаем по «Target 1», выбираем «Add Group». Появится новая группа. Переименовываем её в User. В этой группе вы будете хранить свои файлы.

Далее нажимаем File -> New. Создастся новый документ. Сохраните его в заранее созданной папке User в папке вашего проекта, назвав файл main.c. Добавьте только что созданный файл в группу User.

Теперь щёлкните правой кнопкой мыши по Target1 -> Options for Target 'Target 1'. Во вкладке C/C++ пропишите в строке Include paths путь к вашему файлу main.c. По этому пути программа будет искать все упомянутые файлы. Если у вас будут лежать файлы в других местах, то пути к ним так же придётся прописать тут.

Подключение библиотеки CMSIS

Библиотека CMSIS входит в состав библиотеки StdPeriph о которой будет рассказано ниже. Скачать обе библиотеки можно с официального сайта фирмы производителя микроконтроллеров www.st.com.

Для подключения этой библиотеки скопируйте в папку с проектом всю папку с библиотекой CMSIS из архива. Папка CMSIS лежит в папке Libraries.

Для уменьшения занимаемого места на диске все папки, кроме Include и Device, можно удалить.

В программе создайте группу CMSIS и добавьте туда файл system_stm32f4xx.c, расположенный по адресу:

CMSIS\Device\ST\STM32F4xx\Source\Templates.

Во вкладке C/C++ в строке Include paths пропишите пути:

\CMSIS\Device\ST\STM32F4xx\Source

\CMSIS\Device\ST\STM32F4xx\Include

Примечание: В относительном пути к файлу не должно быть пробелов. Программа игнорирует всё, что написано после пробела и не сможет найти путь к нужным файлам.

В файле main.c пропишите следующие строки:

```
#include “stm32f4xx.h”

int main(void)
{
        while(1){}
}
```

В первой строке вы подключаете файл из библиотеки CMSIS.

Но перед компиляцией проект требуется настроить.

Настройка библиотеки CMSIS

Прежде всего, надо библиотеке указать тип используемого микроконтроллера.

Но прежде чем изменять библиотечные файлы, надо с них снять галочки только для чтения. Для этого найдите в папке CMSIS по адресу CMSIS\Device\ST\STM32F4xx\Include файлы stm32f4xx.h и system_stm32f4xx.h. И, зайдя в свойства, снимите эти галочки. Теперь файлы можно изменять из-под keil.

Откройте файл stm32f4xx.h. Это можно сделать из-под keil, нажав правую кнопку мыши в строке #include “stm32f4xx.h” на названии файла и в выпавшем контекстном меню выбрав open stm32f4xx.h.

Чтобы указать тип микроконтроллера, найдите в открытом файле строку:

/* #define STM32F40_41xxx */

Требуется раскомментировать эту строку, чтобы указать, что именно этот тип микроконтроллера используется в лабораторной работе. Получится следующее:

#define STM32F40_41xxx

Найдите чуть ниже строку:

#define HSE_VALUE ((uint32_t)xxxxxxxx),

где xxxxxxxx это значение тактовой частоты используемого кварцевого резонатора в Гц. Установите её равной частоте используемого тактового резонатора (см. отладочную плату). Значение тактовой частоты указано на кварцевом резонаторе на крышке.

Теперь щёлкните правой кнопкой мыши по Target1 -> Options for Target 'Target 1'. В разделе Target установите значение Xtal равным HSE.

Нажмите F7. Проект должен завершить компиляцию успешно.

Подключение библиотеки StdPeriph

Для подключения этой библиотеки, скопируйте в папку с проектом папку STM32F4xx_StdPeriph_Driver, находящуюся в той же папке Libraries. В keil создайте группу StdPeriph и добавьте в неё все файлы из папки src.

В итоге у вас должно получиться что-то похожее на то, что изображено на рис 5:

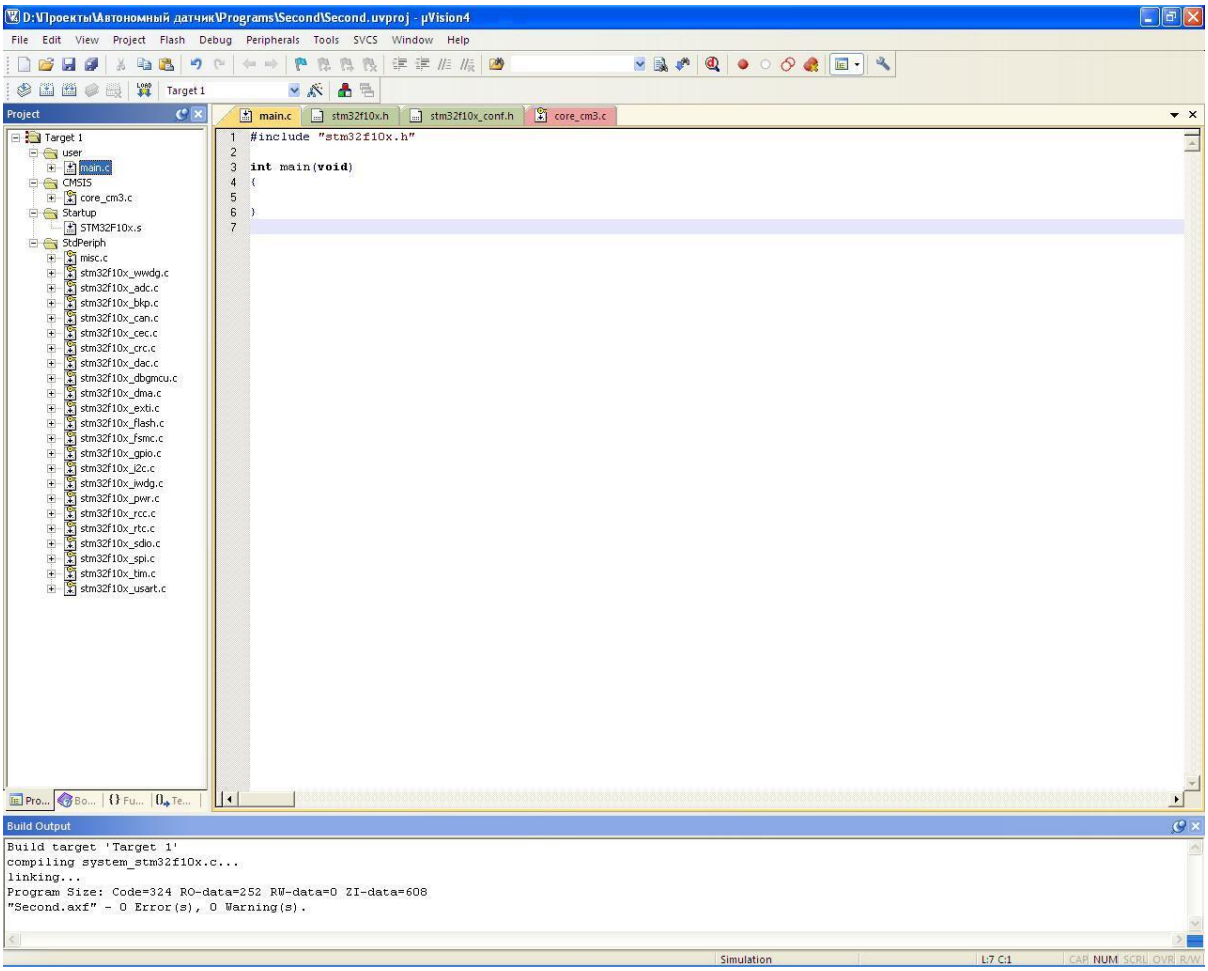

Рис 5

Во вкладке C/C++ в строке Include paths прописываем пути к папкам inc и src и к самой папке stdPeriph. Например:

\STM32F4xx_StdPeriph_Driver\inc

\STM32F4xx_StdPeriph_Driver\src

У вас должно получиться как на рис 6:

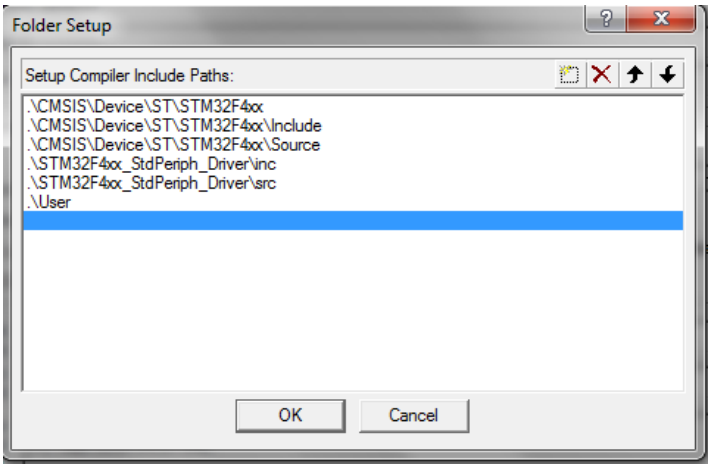

Рис 6

В файле stm32f4xx.h найдите строки:

```
#if !defined  USE_STDPERIPH_DRIVER
/**
* @brief Comment the line below if you will not use the peripherals drivers.
  In this case, these drivers will not be included and the application code will
  be based on direct access to peripherals registers
*/
 /*#define USE_STDPERIPH_DRIVER*/
#endif
```

Раскомментируйте строку /*#define USE_STDPERIPH_DRIVER*/. Это значит, что вы собираетесь использовать библиотеку STDPERIPH.

Теперь добавьте в проект файл stm32f4xx_conf.h. Этот файл можно найти в папке Project\examples в любом проекте. Например, GPIO. Бросьте его в папку STM32F4xx_StdPeriph_Driver. В ней его будет просто найти в случае необходимости. Пропишите путь к папке, где лежит этот файл. В этом файле подключаются все файлы библиотеки. Неиспользуемые модули можно комментировать.

Примечание: В случае, если функции какой-то периферии не работают, имеет смысл проверить не закомментирована ли строчка с её подключением.

На этом подключение библиотеки закончено. Если нажать на F7, то проект должен собраться без ошибок.

В случае возникновения ошибок удалите файл stm32f4xx_fmc.c. Микроконтроллер stm32f407 не поддерживает эту периферию.

Программирование платы stm32f4-discovery

После написания кода программу необходимо собрать нажав либо на F7, либо на кнопку 2 или 3, показанные на рис 7.

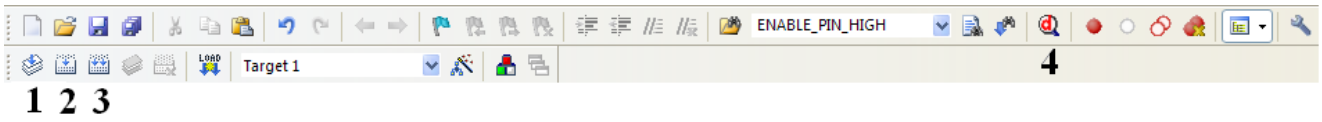

Рис 7

Кнопка 1 (рис 2.7) проверяет синтаксис одной страницы, кнопка 2 добавляет в собранный проект изменения. Кнопка 3 – собирает проект целиком с начала. Т.о. при частом изменении проекта стоит использовать кнопку 2, т.к. она собирает проект значительно быстрее.

Программирование платы производится кнопкой 4. Но до использования этой кнопки требуется задать тип используемого программатора. По-умолчанию стоит симулятор.

Для изменения программатора зайдите в настройки target1. Выберите вкладку debug. Переставьте точку с Use simulator на Use … В выпадающем окне выберете St-Link и нажмите настройки. Там надо выбрать SWD вместо JTAG (см. рис 8).

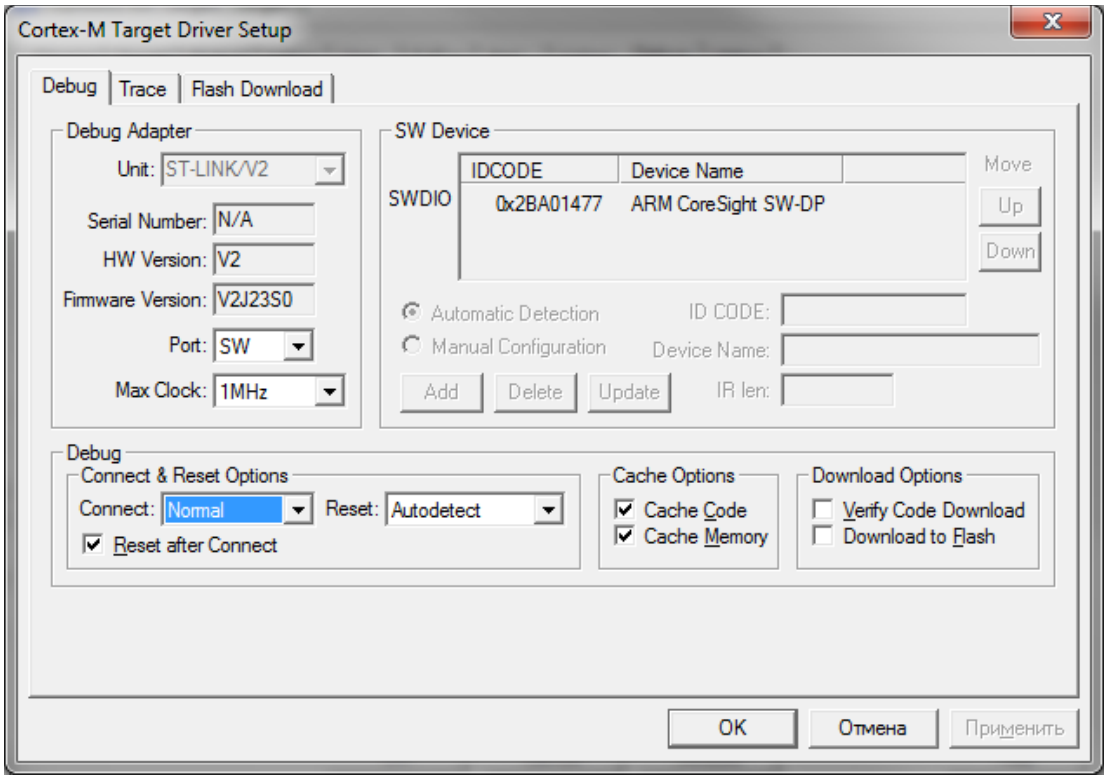

Рис 8

Во вкладке Flash Download нажмите add и выберете stm32f4 Flash (рис 9). Это действие укажет алгоритм, по которому требуется зашивать данный микроконтроллер. Если используется МК отличный от stm32f4, то нужно выбрать соответствующий ему алгоритм. Выбрав нужный алгоритм, нажмите add.

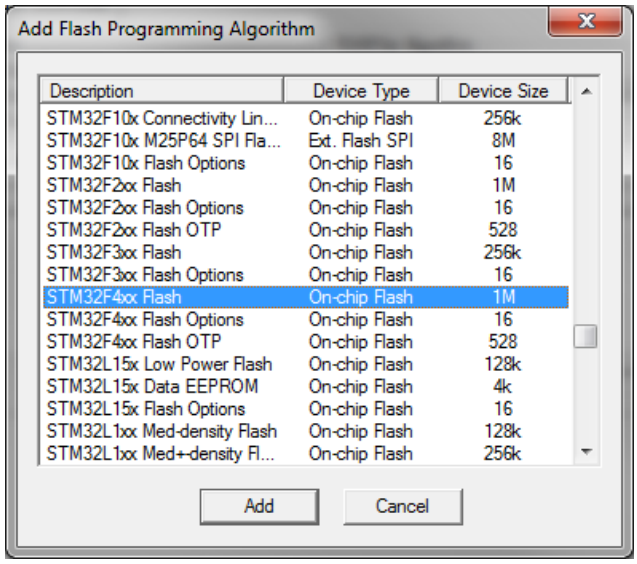

Рис. 9

Далее во вкладке Utilities выберете точно такой же программатор. Нажмите Ок. Теперь плата stm32f4-discovery должна запрограммироваться.

При программировании может выскочить окно, предупреждающее об ограничении кода в 32кБ. Это ограничения бесплатной версии программы. Для продолжения необходимо просто нажать Ок.

Настройка тактовой частоты ядра микроконтроллера

Система тактирования микроконтроллера представляет собой сложное устройство, разветвляющее сигналы с разными тактовыми частотами на разные модули микроконтроллера. Блок-схема модуля настройки частоты выглядит как на рис. 10.

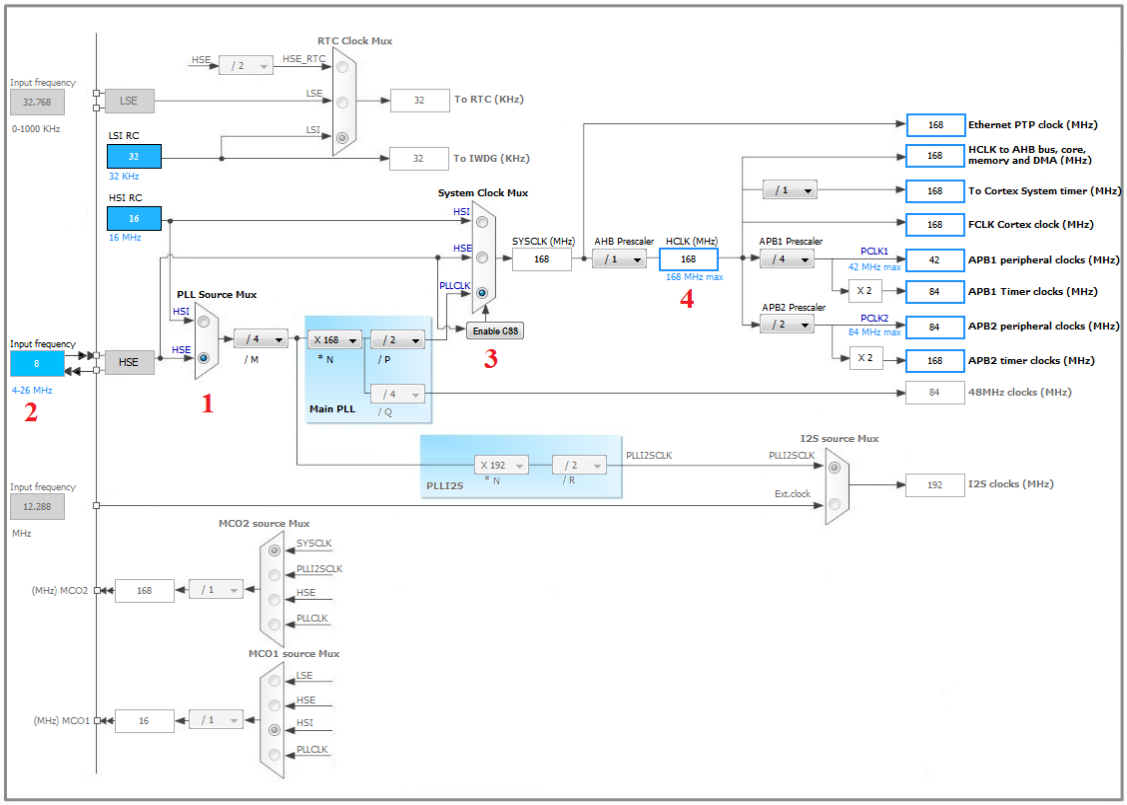

Рис. 10

Для настройки тактирования микроконтроллера необходимо в файле system_stm32f4xx.c необходимо найти строку #define PLL_M. На это значение делится входной тактовый сигнал с кварцевого резонатора. На выходе должно получиться число больше 1 МГц. Т.е., если частота кварцевого резонатора равна 9 МГц, а делитель – 25, это неправильно. В этом случае он не может превышать значение, равное 8.

Следующее число – это умножитель частоты, расположенный в строке:

#define PLL_N

На выходе этого блока частота тактового сигнала не может превышать 400МГц. Её следует устанавливать с учётом того, что далее следует ещё 1 делитель не менее чем на 2. Этот делитель представлен в строке.

#define PLL_P

Итоговый сигнал не может превышать 168 МГц.

Совет: В файле system_stm32f4xx.c может быть несколько строк определения делителей и умножителей. Чтобы выяснить какой именно рабочий, щёлкните по нему правой кнопкой мыши и выберите go to definition. Курсор перейдёт к тому элементу, который участвует в работе программы.

Как узнать частоты тактирования

Для того, чтобы узнать частоты тактирующих сигналов различных шин микроконтроллера возможно воспользоваться специальной функцией:

void RCC_GetClocksFreq(RCC_ClocksTypeDef* RCC_Clocks);

В качестве её аргумента используется структура, в которую записываются значения тактовых сигналов различных шин. Их значения приведены в таблице.

Таблица

Название	Описание
SYSCLK_Frequency	Системная частота тактирования
HCLK_Frequency	Частота тактирования ядра
PCLK1_Frequency	Частота тактирования шины PCLK1, отвечающей за тактирование таймеров 2-7, 12-14.
PCLK2_Frequency	Частота тактирования шины PCLK1, отвечающей за тактирование таймеров 1, 8-11.

Частота тактирования таймера может отличаться от частоты тактирования шины в 1 или 2 раза. Если HCLK_Frequency и PCLKx_Frequency совпадают, то частота тактирования соответствующих таймеров равна PCLK1_Frequency. Если PCLK1_Frequency меньше HCLK_Frequency, то частота таймера в 2 раза больше частоты PCLKx_Frequency.