STM32F4 family – Практические занятия

Atollic TrueSTUDIO STM32
+ STM32F4 discovery kit
modified by ***www.emcu.it***

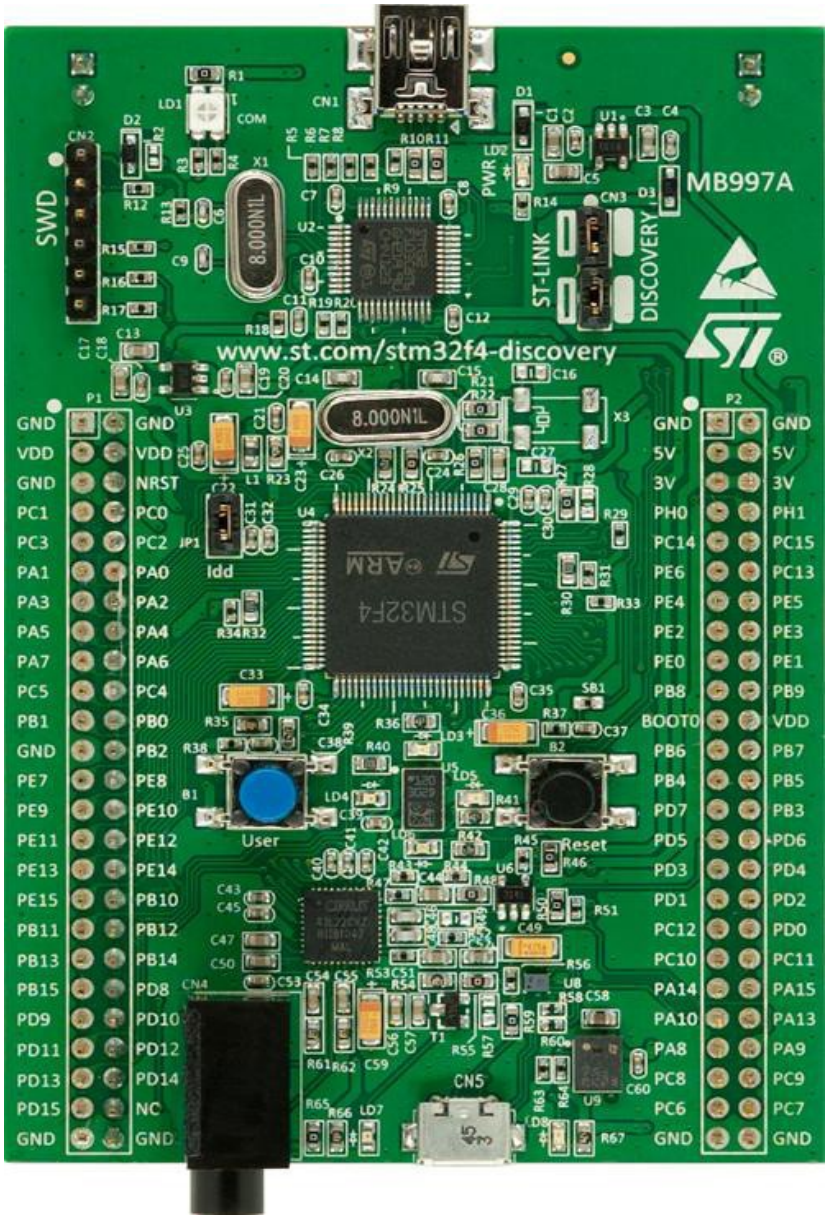
Thanks to the collaboration of:

Alexander A. Shchitnikov

Прежде чем начать

- Потребуется ПК с Windows 2000/XP/Vista/7 и правами администратора (требуется для работы ST-Link программатора/отладчика)
- Установите последнюю версию Atollic TrueStudio STM32
- Подготовить кабель USB cable type A to mini-B
- Подготовить отладочную плату STM32F4-DISCOVERY
- Проверить наличие обновлений для ST-Link на сайте www.st.com/stm32f4-discovery

STM32F4-DISCOVERY знакомство

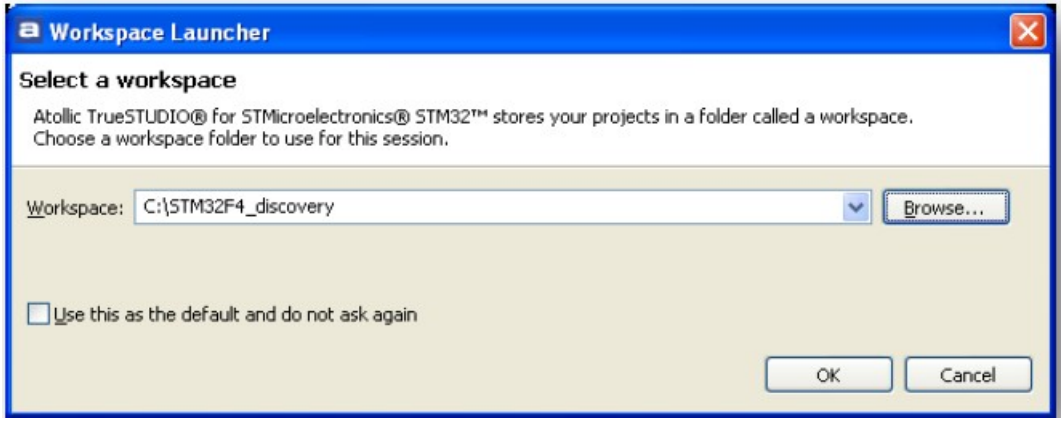


- Плата STM32F4-DISCOVERY состоит из двух частей: программатора/отладчика ST-Link и отладочной платы с МК семейства STM32F4**(STM32 на базе Cortex-M4)
- ST-Link может использоваться в качестве внутрисхемного программатора других STM32 микроконтроллеров (переключатель SEL отключен).Работает исключительно в режиме SW
- Отладочная часть содержит МК STM32F407VGT6 (1MB Flash,192kB RAM, 100 pin package) со встроенной тактируемой системой(основная частота HSE генератора 8Mhz)
- Также плата включает:

  MEMS акселлерометр (LIS302DL) подключенный через SPI1
  Простой пользовательский интерфейс(кнопка + 4 светодиода)
  Audio codec с аудиовыходом
  MEMS микрофон (MP45DT02)
  USB OTG разъем с 2 светодиодами

Atollic TrueStudio/STM32 Введение

- Запустите TrueStudio STM32 Pro
- Введите путь к новой или уже существующей рабочей области (workspace)
- Лучше не ставить метку "использовать по умолчанию и никогда больше не спрашивать" (Use this as the default and do not ask again)
- При создании нового проекта появляется окно wellcom window -> выберите Start Using TrueStudio

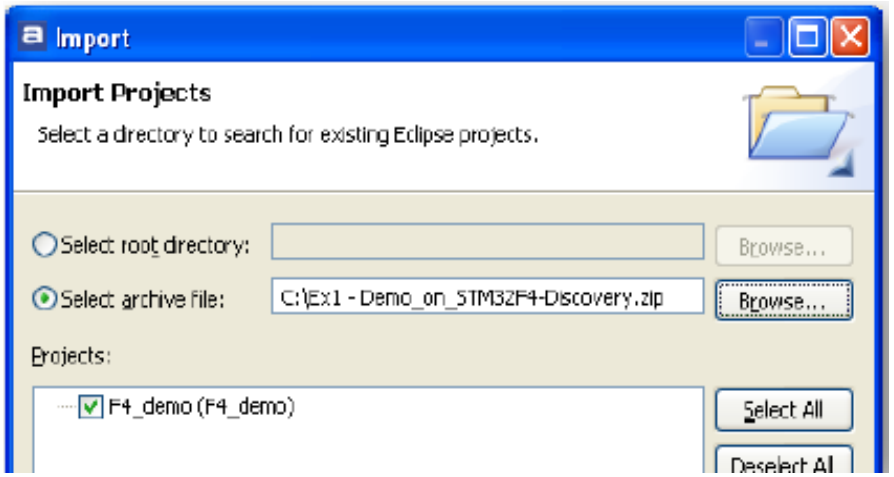


Atollic TrueStudio/STM32
Импорт существующего проекта (.ZIP) в рабочую область

- Выберите File->Import
- В Import window выберите General а затем Existing Project into Workspace
- Нажмите Next

В Import project window:

- Добавте архивированный файл
- Выберите проект из файла
- Нажмите Finish



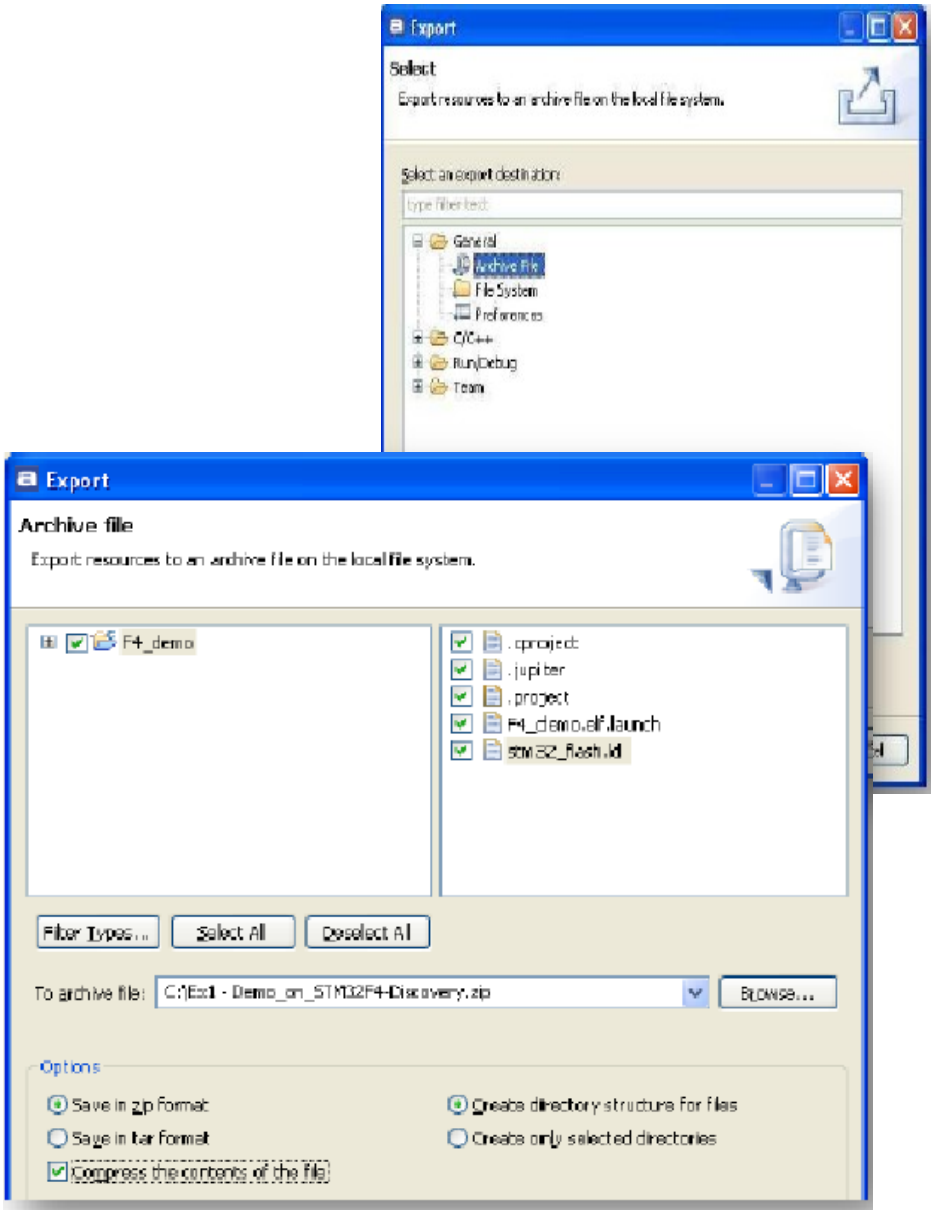
- В результате весь проект копируется в workspace и автоматически компилируется

Atollic TrueStudio/STM32
Экспорт существующего проекта в архив (.ZIP)

- Выберите File-Export
- In Export window выберите вкладку General
- Нажмите Next

В окне Archive file:

- Выберите проект в текущей workspace
- Выделите настройки проекта(.cproject, .project и связующий файл *.ld)
- Выберите выходной формат (.zip,.tar)
- Если необходимо выберите сжатие содержимого
- Выделите Create directory structure for files
- Finish



Atollic TrueStudio/STM32 Проект – структура файлов

- После создание проекта, следующая файловая структура появляется в project explorer:
  - Binaries
  - Includes
  - Utilities -> процедуры для управления ГУИ на отладочной плате
  - Src – главная директория исходников (main.c main.h)
  - Libraries -> ST и CMSIS библиотеки
    - CMSIS
    - STM32Fxx_StdPeriph_Driver
  - Debug
- Физическое расположение файлов проекта на HDD включают следующие сабдиректории:
  - .metadata – конфигурация workspace;
  - <Prj_name> -> главная директория проекта, включающая исходники, компановщики(*.ld), object files и executables
  - .coverege -> используется для анализа плотности кода (не доступно в бесплатной версии)
  - Settings ->настройки hardware выбранного МК, отладочной платы и программатора/отладчика
  - Debug -> объектные файлы испольняемого проекта
  - Libraries
  - Utitlities
  - src
- При исользовании внешних исходников имеются лишь папки .settings и debug

Atollic TrueStudio/STM32 Проект – работа с файлами

- Самый простой способо добавить new.c файл в проект – скопировать его в одну из директорий. Файл будет определен автоматически и добавлен в компилируемые и связывающие файлы. Другий способ – импортировать или установить внешнюю ссылку
- Важная опция Project Explorer – **действительны для файлов и каталогов**(правый щелчок мыши):
  - **Exclude from built –** для того, чтобы не компилировать часть кода
  - **Delete** – физическое удаление всех выделенных файлов с HDD
  - **Import** – физическое копирование выделенного файла в каталог проекта
- Открыть/создать новый проект в той же рабочей области приведет к копированию библиотек, избежать этого можно используя настройки удаленного доступа.

Atollic TrueStudio/STM32 Tasks editor - возможности

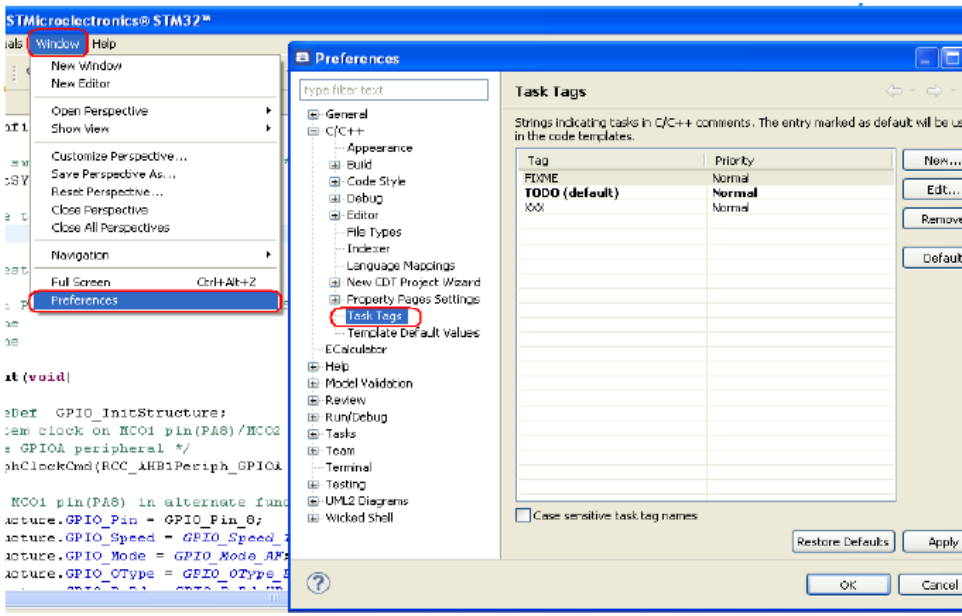
Вернуться позже...задачи

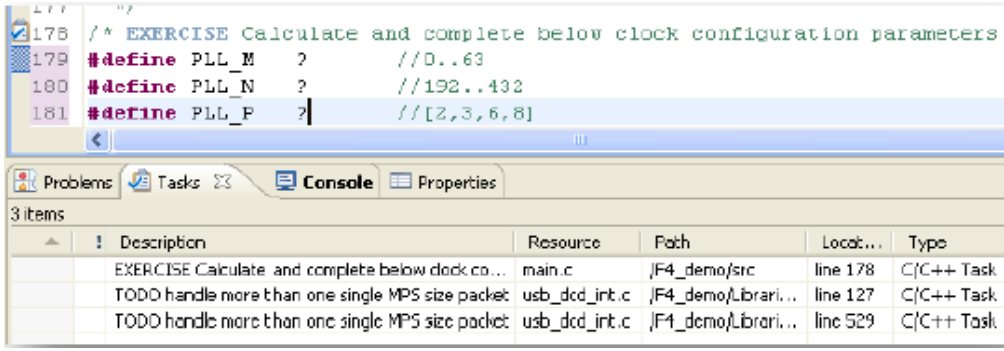
- Иногда возникает необходимость вернуться к определенному участку кода позже.

  Чтобы сделать это:
- Правый клик мыши на номер строки к которой потребуется вернуться.
- Вставте комментарий в стиле C (/**/) с первым словом соответствующем определенному ключевуму слову (task tags)
- Сделать task tag видимыми

  window->preference->C/C++ section-> Task Tags
- **Tasks list** будет виден в нижней части экрана, двойной щелчек на задаче откроет определенный файл в месте где была поставлена задача
- Чтобы удалить задачу достаточно щелкнуть правой кнопкой мыши на задаче и выбрать delete или удалить ключевое слово

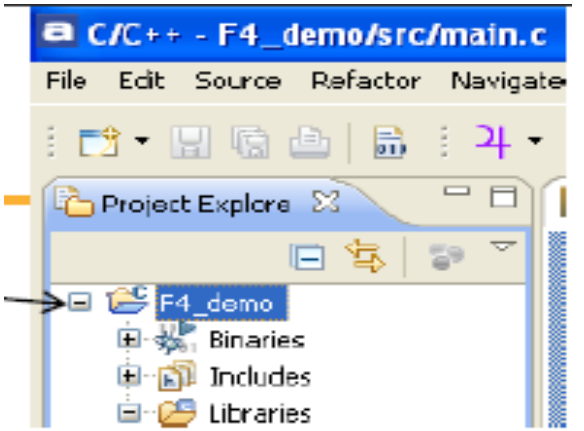
  ВНИМАНИЕ- определенные пользователем task tag не будут экспортированы вместе с проектом в новую рабочую область. Они должны быть заданы заново.

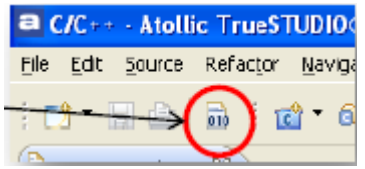


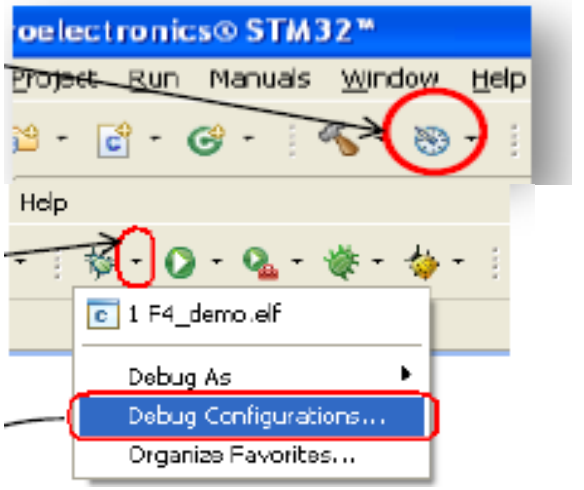


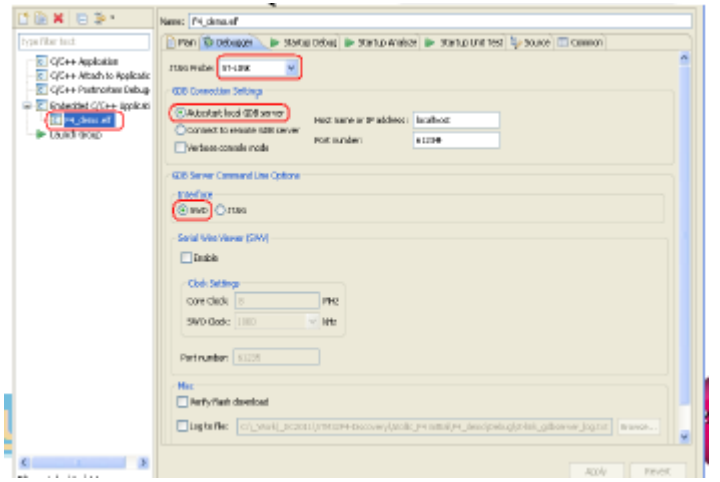
Atollic TrueStudio/STM32 Компановка проекта и конфигурация режима отладки

- Выберите проект в project explorer
- Выберите конефигурации отладки
  - Project -> Build Configuration-> Set Active
- Собрать проект
  - Project-> Build All или Ctrl+B
- Конфигурация отладки
  - Run->Debug Configurations
    - Выделить "Autostart local GDB server"
    - Выделить SWD
- Запустить отладку
  - Run-> Debug или F11



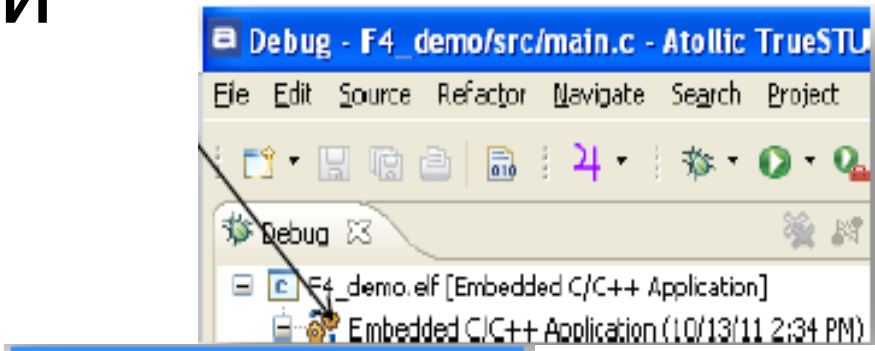




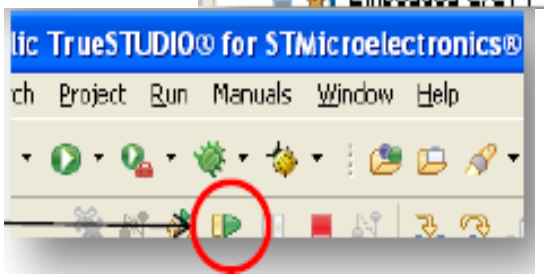


Atollic TrueStudio/STM32
Запуск режима отладки

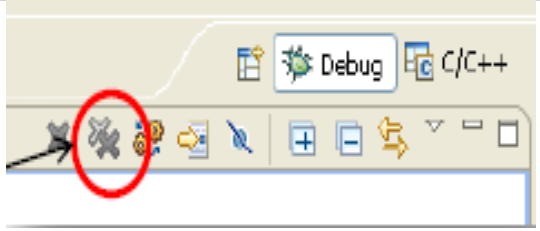
- Выделить GDB аппаратный отладчик



- Запустить программу
  - Run->Resume или F8 или
- Завршить отладку
  - Run -> Terminate or F12 or

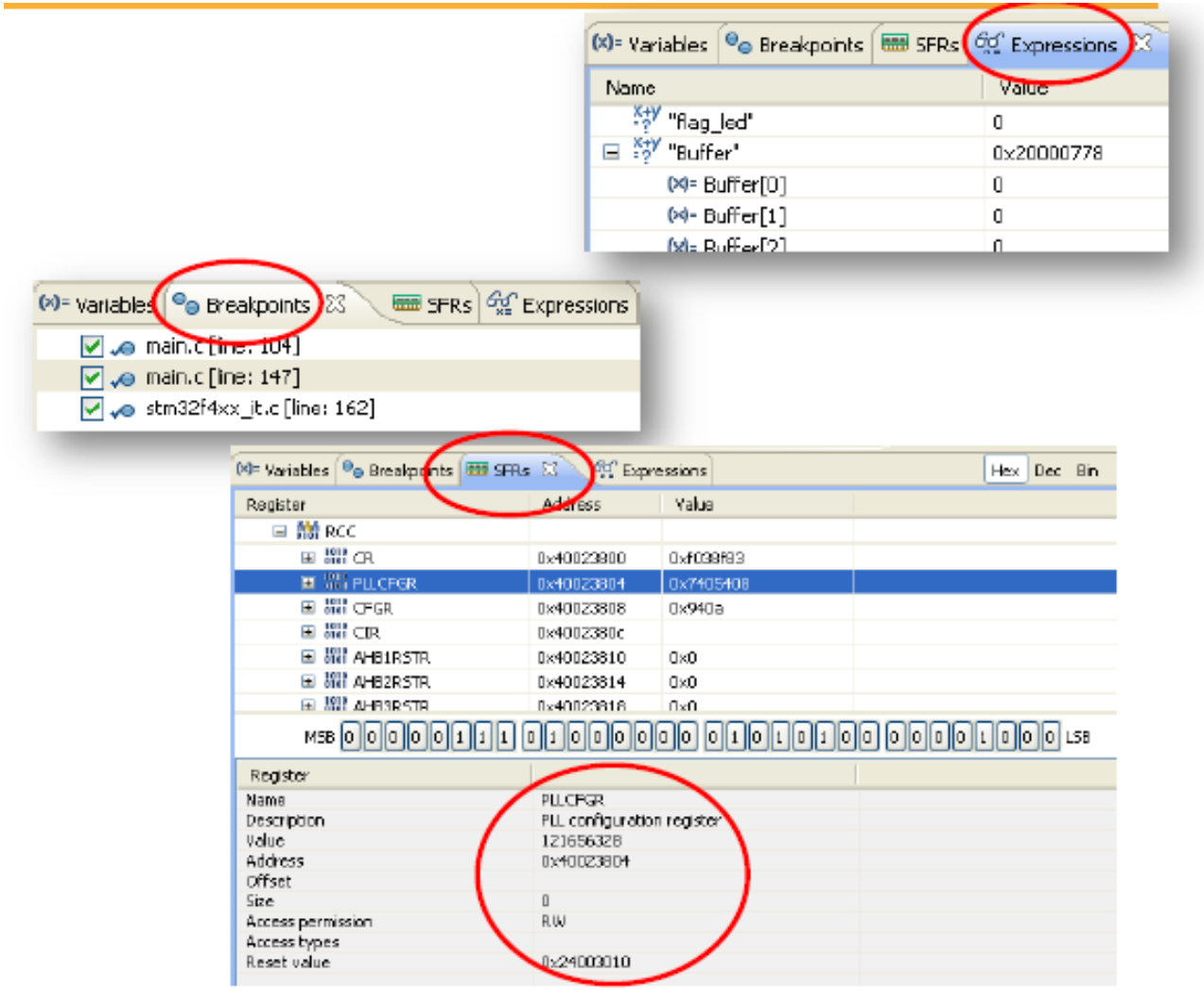


- Добавить/удалить breakpoint
  - Двойной щелчек на номере строки
- Удалить все breakpoints
  - Run->Remove all breakpoints or



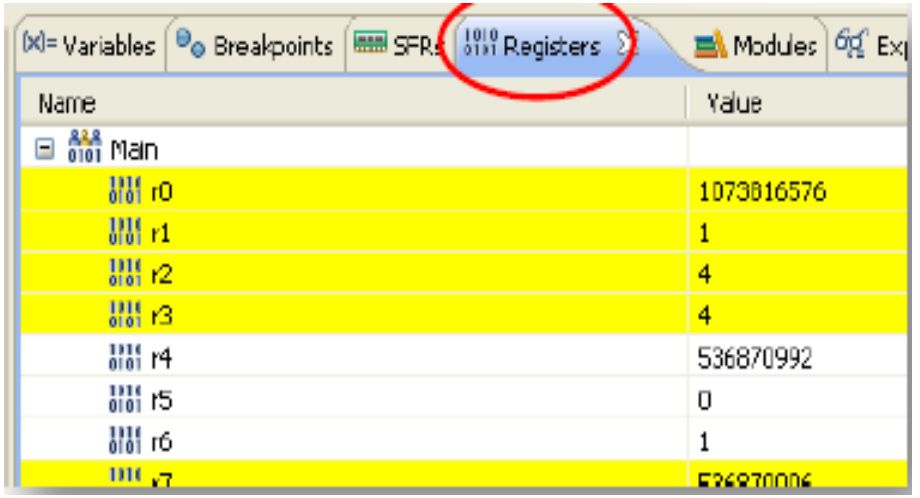
Atollic TrueStudio/STM32
Режим отладки

- Чтобы наблюдать за состоянием переменных -> перетащите их названия во вкладку Expression или правый щелчк мыши и выберите "Добавить наблюдаемые переменные" (Add watch expression).
- Чтобы установить Breakpoint -> двойной щелчек по строке. Breakpoint можно вкл/выкл во вкладке Breakpoints.
- Вся важная информация о состоянии регистров переферийных устройств также доступна в режиме отладки (состояние сброса, адресс, текущее состояние и описание).

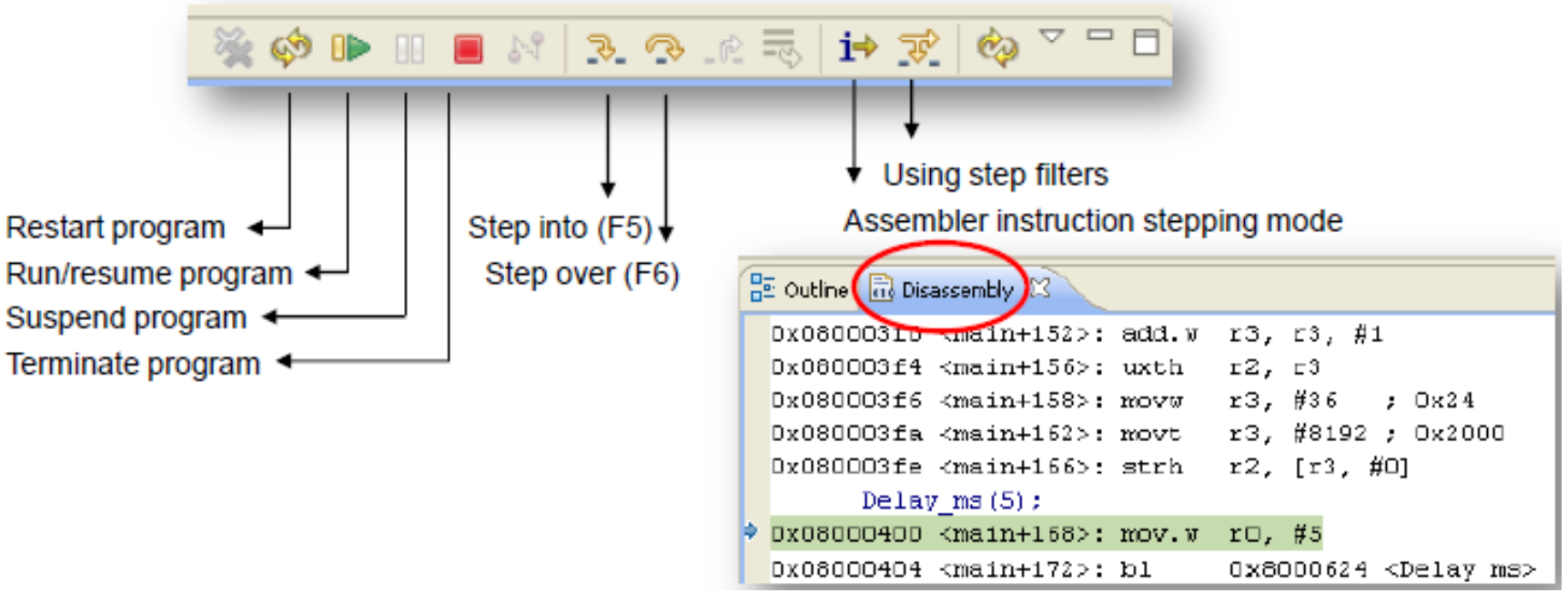


Atollic TrueStudio/STM32
Режим отладки - управление

- Возможно отслеживать изменение в регистрах общего назначения -> вкладка Registers(любые изменения сигнализируются желтым выделением)

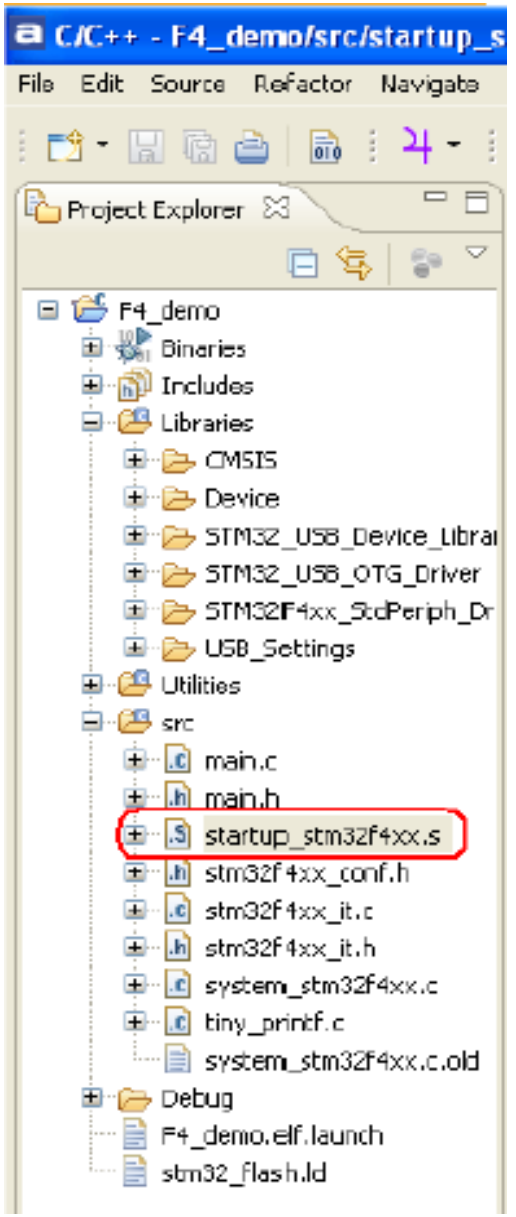


- Отладочная панель

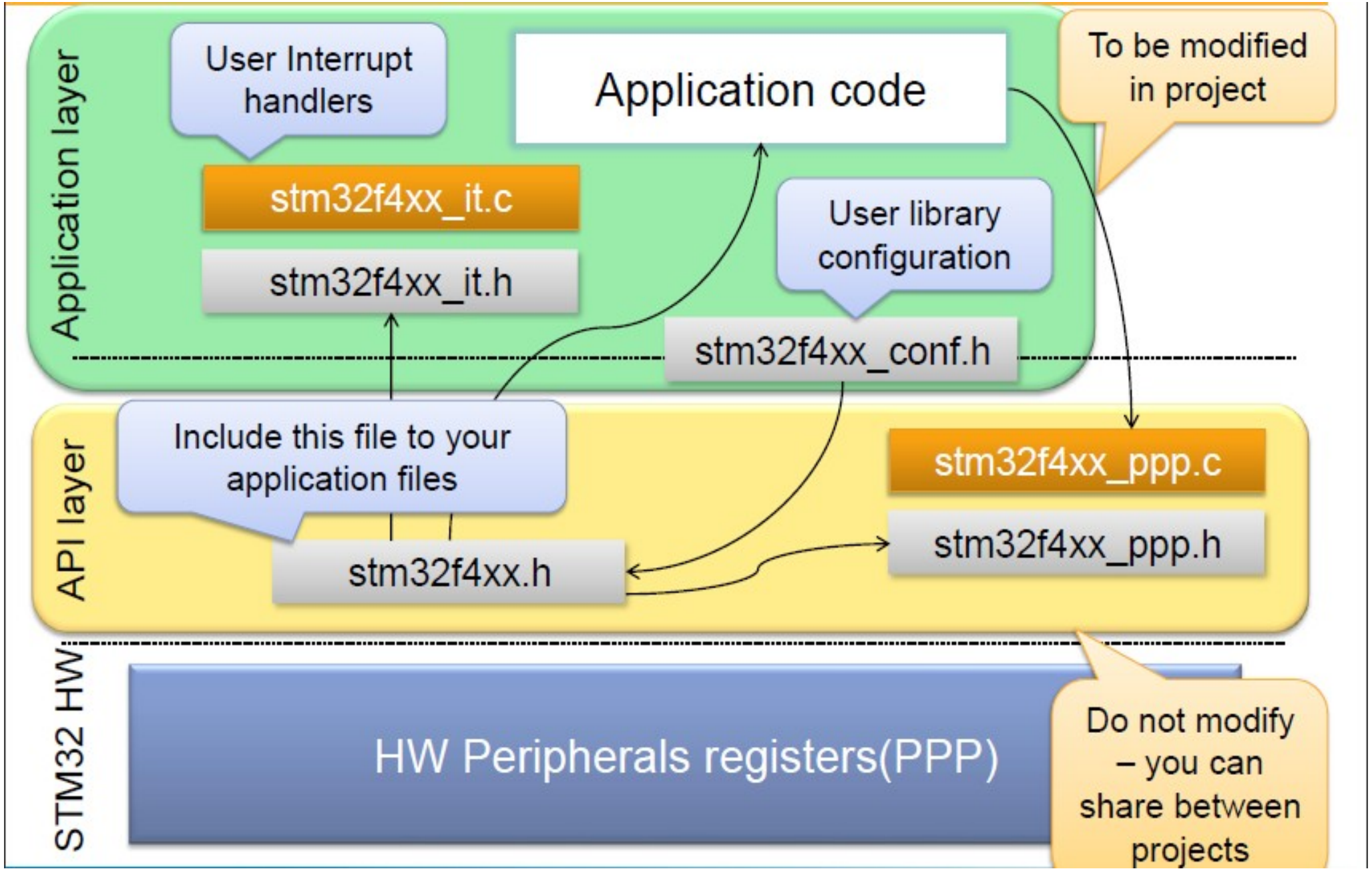


STM32 – начальные процедуры

- Минимальные требования для запуска STM32 – записать два первых слова таблицу векторов:
  - Первое слово инициализирует указатель стека (**Stack Pointer Value**)
  - Второе слово адресс процедуры сброса (**adress of reset procedure**)
- Рекомендуется применять вектора обработки ошибок (хотя бы векор аппаратного сбоя – HardFault)
- При использовании стандартной библиотеки стандарта CMSIS, таблица векторов определена в стартовом файле (**startup**), который определен для всех семейств и сборок.
- В случае STM32f407VGT6 и Atollic, стартовый файл называтся startup_stm32fxx.s и расположен в папке /src внутри проекта.
- В ST library предусмотрены дополнительные операции выполняющиеся непосредственно перед **main()**. Наиболее важная **SystemInit()**, расположенная в **system_stm32f4xx.c**. Данная функция настраивает систему тактирования и порты общего назначения в соответствии с требованиями внешних условиях. Это не очень важно при использовании стандартных настроек.
- Для отключения этой функции, стока **"bl SystemInit" в startup_stm32fxx.s должна быть закомментирована.** (строка 104)

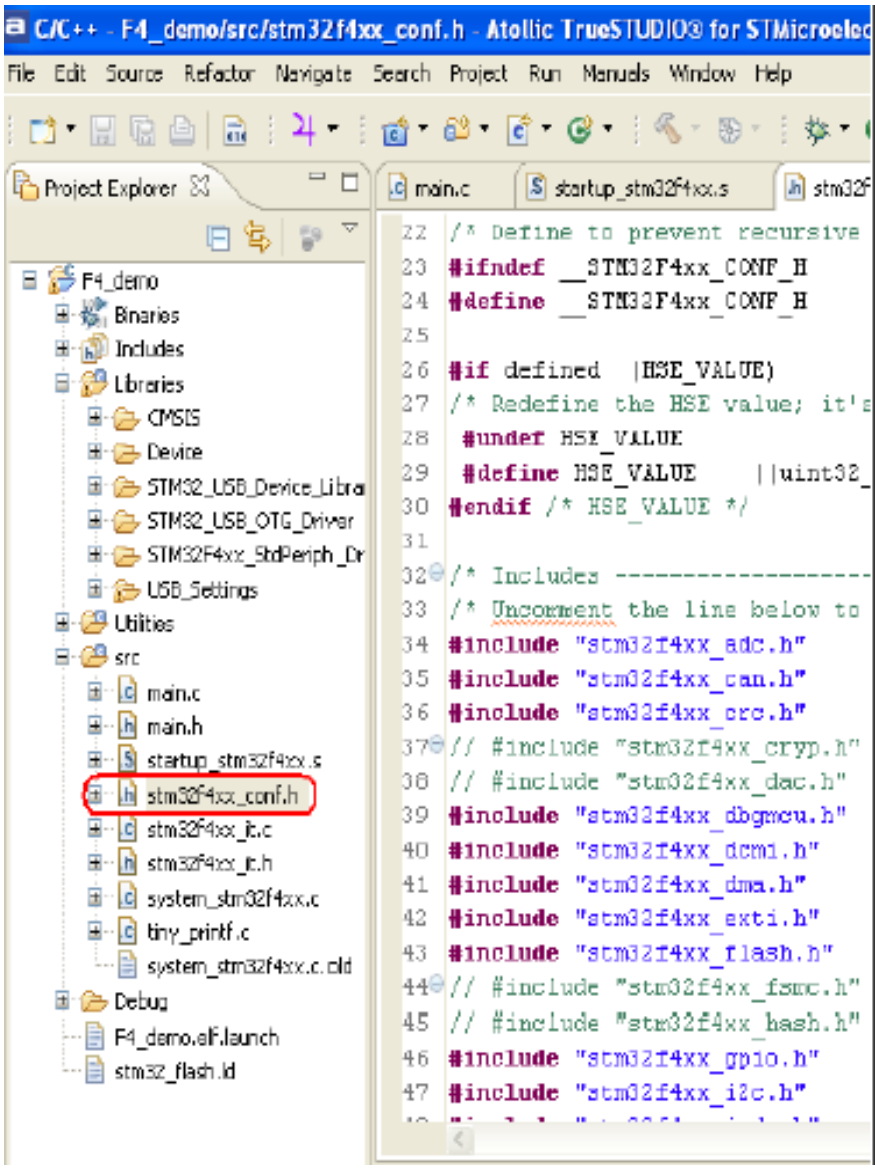


STM32 стандартные библиотеки - интерфейс прикладного программирования (API)



STM32 – стандартная библиотека периферии

- Для STM32 стандартная периферийная библиотека написана на **ANSI C**
- Для каждого устройства предусмотренно два раздельных файла, например:
  - stm32f4xx_gpio.c
  - stm32f4xx_gpio.h
- Для этого необходимо:
  - **#include "stm32f4xx.h"**
  - Добавить в исходные файлы используемую периферию, i.e. "stm32f4xx_gpio.c" для портов общего назначения
- В "stm32f4xx_conf.h" снимите комментарий с используемых переферийных устройств:
  - #include "stm32f4xx_gpio.h"
- Пустые процедуры прерываний представлены в stm32f4xx_it.c. Все прерывания должны быть описаны здесь.
- Функции прерываний не требуют специального синтаксиса и описываются в виде: **void funct(void)**
- Вся документация о библиотеке доступна в html формате: **stm32f4xx_dsp_stdperiph_lib_um.chm**
- В основном периферий ные устройства определены одой или двума структурами данных, которые используются при конфигурации. После заполнения структуры используются в **PPP_Init()** функциях, для конфигурации регистров переф. устройств



STM32 – библиотека – как ее использовать?

- Функции и переменные для каждого периферийного модуля, имеют приставку в названии: GPIO, TIM1-> **GPIO_Init(), ADC_Channel_0, USART_TX_TXE**
- Большинство настроек имеют обозначения от 1 до N и к ним можно применять конкатенацию: GPIO_Pin_0|GPIO_Pin_1, это значит, что pin0 и pin1 будут настроены в одно время.
- В stm32f4xx.h определены типы данных:
  - u8 – unsigned char
  - u16 – unsigned short
  - RESET/SET
  - FALSE/TRUE
  - DISABLE/ENABLE
- Большинство периферийных уст-в имеет следующий набор инструкций
  - PPP_DeInit(...) - перевести устройство в начальное состояние
  - PPP_Init(...) - подтверждение конфигурации устройства
  - PPP_Cmd(ENABLE/DISABLE) – вкл/выкл устройство
  - PPP_ITConfig(...) - настройка прерывания устройства
  - PPP_GetFlagStatus(...) - прочитать значение флага устройства
  - PPP_ClearFlag(...) - очистить флаг в устройстве
  - PPP_ClearITPendingBit(...) - очистить флаг запроса прерывания

STM32 – библиотека – как ее использовать?

- Компилятор сообщает о множестве ошибок, таких как:

  Missing prototype

  GPIO_Pin_0 undefined

  Решение:

  Убедитесь, что в файле stm32f4xx_conf.h используемые модули библиотеки не закомментированы

  Убедитесь, что константа USE_STDPERIPH_DRIVER определена в вашей среде

- Линкер сообщает о множестве ошибок, таких как:

  Lab_library.lkf:1 symbol_GPIO_WriteHigh not defined (Debug/main.o)

  Решение:

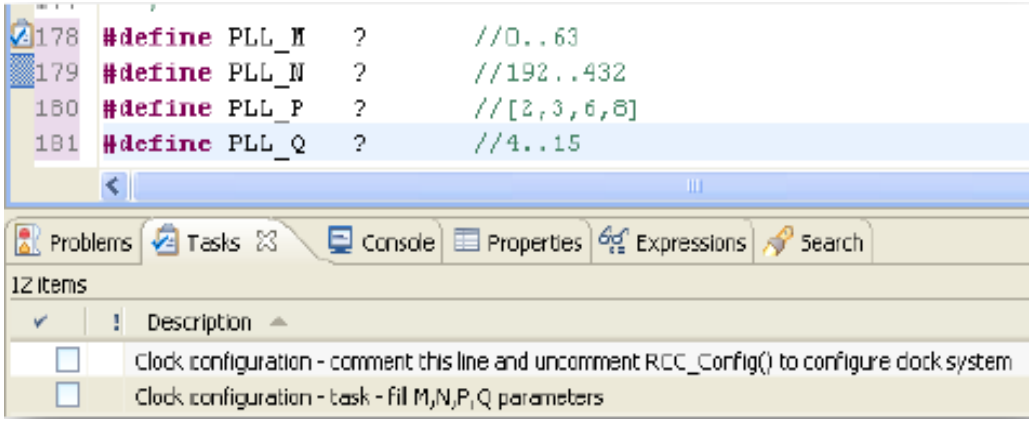
  Убедитесь, что исходные файлы библиотеки добавлены, в данном случае **stm32f4xx_gpio.c**

Упражнение №1

Базовые настройки. Упражнение 1/2

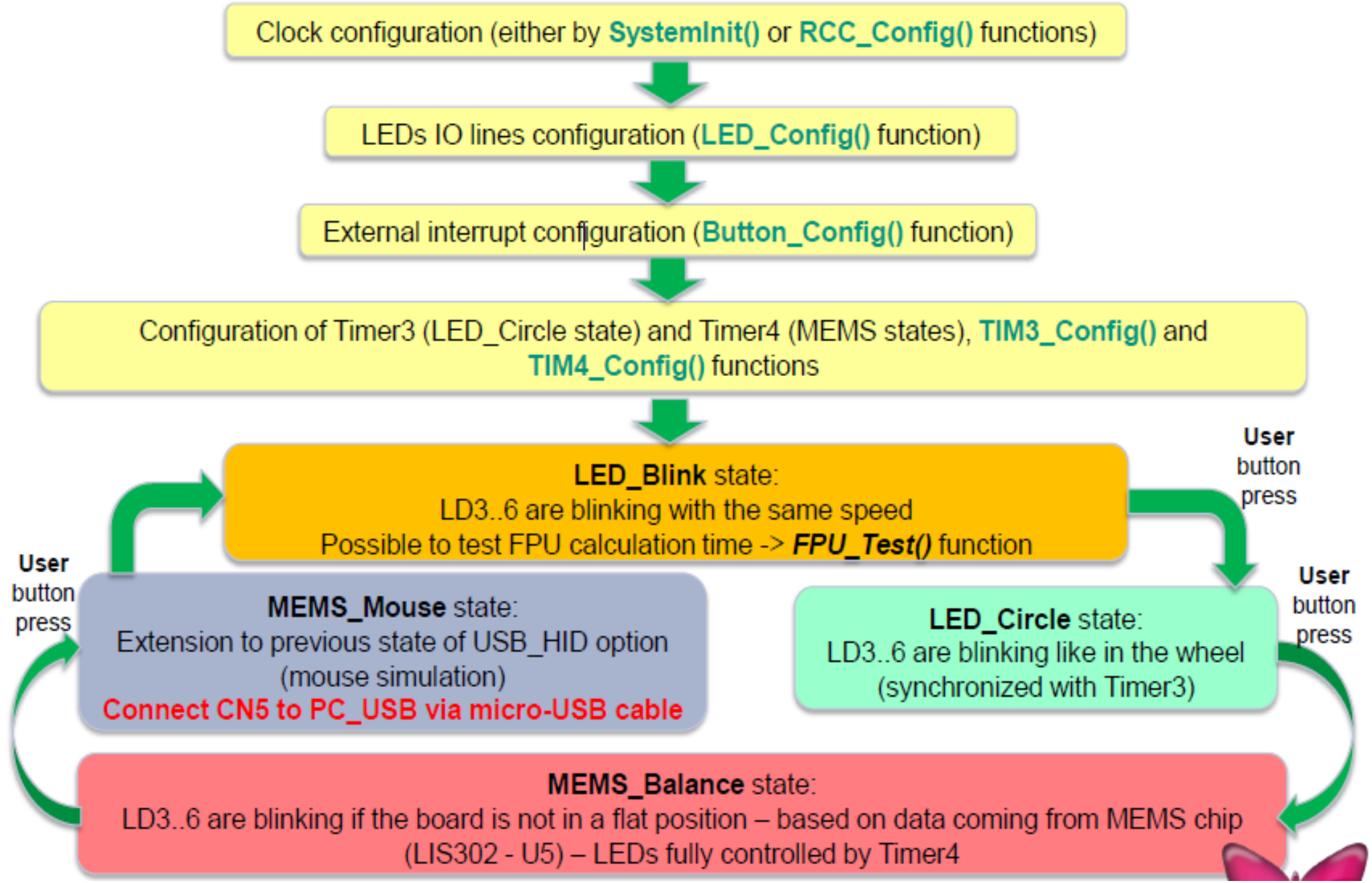
- Создайте новую рабочую область(workspace) в среде Atollic и импортируйте заархивированный проект:

  Ex1-Demo_on_STM32F4-Discovery.zip
- Исходный код поврежден в нескольких местах (пара файлов из раздела /src)
- Изменения должны быть выполены только в:
  - main.c ->основные процедуры
  - stm32f4xx_it.c-> прерывания
  - stm32f4xx_conf.h-> выбор библиотечных модулей
- Места которые необходимо изменить помечены символом "?" и сгруппированны в 5 мини-задач, полностью описанных в следующей секции
- Задачи можно увидеть во вкладке задач(Tasks Tab), с приставкой EXERCISE
- **Чтобы сделать их видимыми добавте метку EXERCISE в Tasks Tag window**



Задачи можно увидеть во вкладке задач(Tasks Tab), с приставкой EXERCISE

Базовые настройки. Упражнение 2/2



Настройка портов ввода/вывода. Теория

- После сброса все ножки находяться в состоянии input floating(неподтянутый вход).
- Ножки сгруппированны в 16 битные порты (GPIOA,GPIOB,...,GPIOI)
- Большинство пинов толерантны к входному сигналу 5В
- Порты настраиваются несколькими регистрами, которые определяются функцией **GPIO_Init()**, в соответствии со значениями структуры **GPIO_InitTypeDef**:

- **GPIO_Pin** -> GPIO_Pin_0 .... 15, GPIO_Pin_All, GPIO_Pin_None
- **GPIO_Mode**:
  - GPIO_Mode_AN //analog mode
  - GPIO_Mode_IN //input mode
  - GPIO_Mode_OUT //output mode
  - GPIO_Mode_AF //alternate function mode
- **GPIO_OType**:
  - *GPIO_OType_PP*
  - *GPIO_OType_OD*
- **GPIO_Speed**:
  - GPIO_Speed_2MHz //lowest EMI -> softer edges
  - GPIO_Speed_25MHz
  - GPIO_Speed_50MHz
  - GPIO_Speed_100MHz //highest EMI -> sharper edges
- **GPIO_PuPd**:
  - *GPIO_PuPd_NOPULL*
  - *GPIO_PuPd_UP*
  - *GPIO_PuPd_DOWN*

Настройка портов ввода/вывода. Задача

Исправте LED_Config() функцию (main.c), чтобы настроить выводы 12..15 порта GPIOD:

- В случаях GPIOD (используется в LED_Blink и LED_Circle):
  Как выходные выводы общего назначения, скоростью 25МГц, без подтягижки к единице

- В других случаях (используется в MEMS_Balance и MEMS_Mouse):
  Выводы подключены к Timer4(как выходы) со скоростью 25MHz (альтернативное использование)

Не забудьте подключить тактирование ПЕРЕД конфигурацией!

Настройка портов ввода/вывода. Решение

Исправте LED_Config() функцию (main.c), чтобы настроить выводы 12..15 порта GPIOD:

- В случаях GPIOD (используется в LED_Blink и LED_Circle):

  Как выходные выводы общего назначения, скоростью 25МГц, без подтягижки к единице

```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12 | GPIO_Pin_13| GPIO_Pin_14| GPIO_Pin_15;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_25MHz;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
```

- В других случаях (используется в MEMS_Balance и MEMS_Mouse):

  Выводы подключены к Timer4(как выходы) со скоростью 25MHz (альтернативное использование)

```
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;

GPIO_PinAFConfig(GPIOD, GPIO_PinSource12, GPIO_AF_TIM4);
GPIO_PinAFConfig(GPIOD, GPIO_PinSource13, GPIO_AF_TIM4);
GPIO_PinAFConfig(GPIOD, GPIO_PinSource14, GPIO_AF_TIM4);
GPIO_PinAFConfig(GPIOD, GPIO_PinSource15, GPIO_AF_TIM4);
```

Не забудьте подключить тактирование ПЕРЕД конфигурацией!

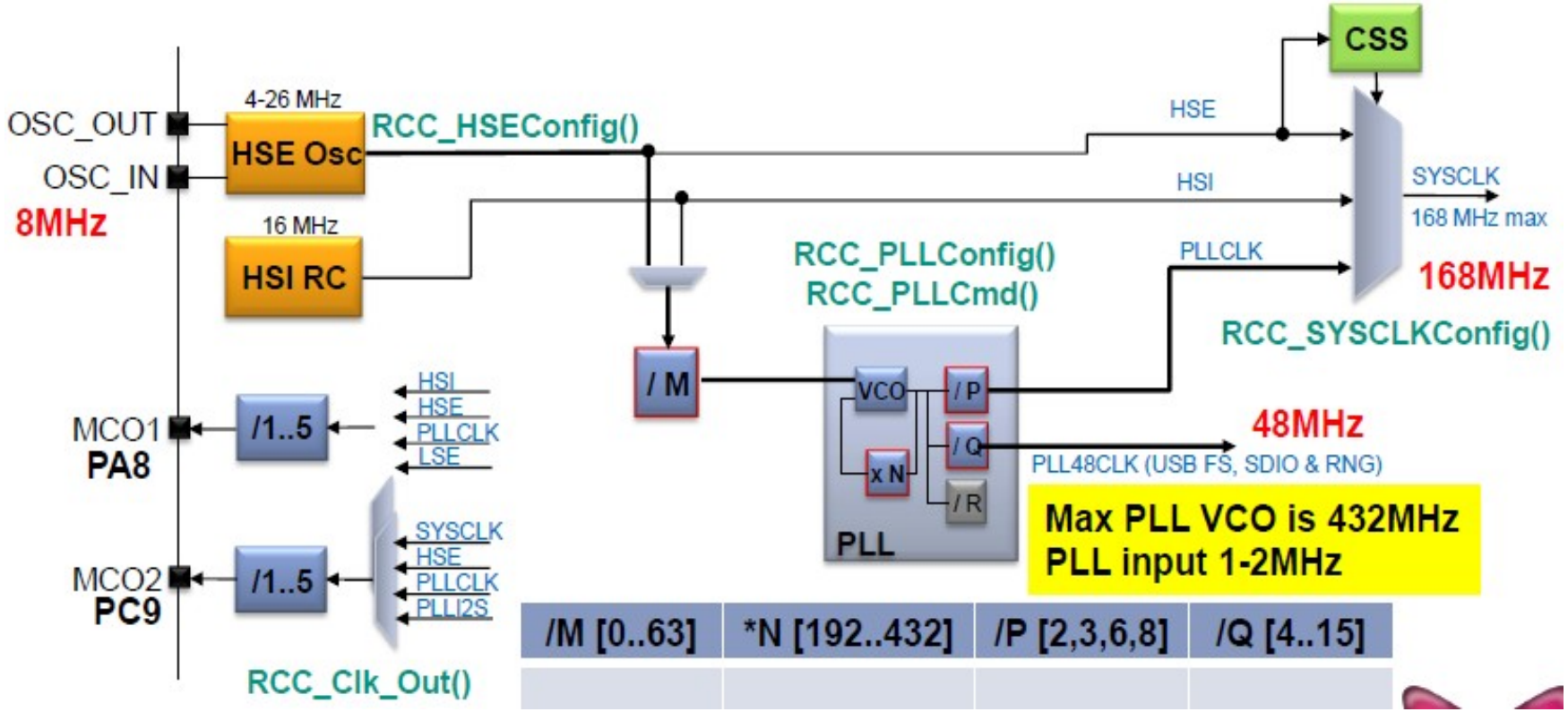
```
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);
```

Настройка тактовой частоты - теория

- После сброса частота системы берется от HSI = 16МГц
- После сброса вся периферия (GPIO в том числе), отключена от тактирования
- Аварийная система контроля HSE, позволяет в случае возникновения проблемы с HSE автоматически переключиться на HSI(сбросовое состояние)
- Существует возможность вывести тактовую частоту через выводы МК (MCO1 MCO2) -> до 100 Мгц
- Невозможно тактировать ядро и периферию низкочастными генераторами
- **При использовании стандартной библиотеки STM32 тактовая частота настраивается автоматически, перед выполнением основного кода, на максимальную частоту (168МГц) от HSE.** Выполняется из функции SystemInit().

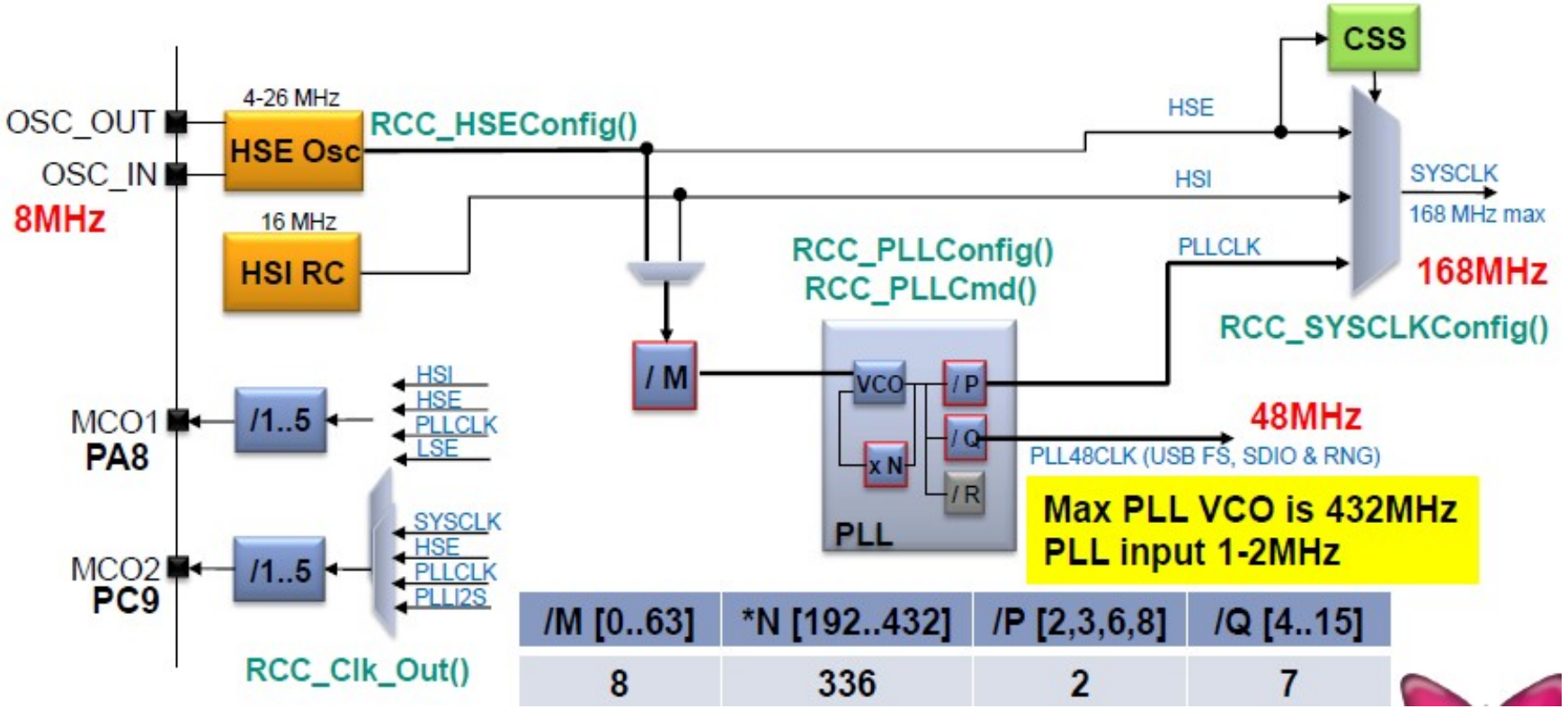
Настройка тактовой частоты - задача

- Расчитайте значение параметров M,N,P,Q и подставте их в функцию RCC_Config() (файл main.c). Требуемые частоты выделены красным.
- Закомментируйте строку 104 (/**/ или //) в startup_stm32f4xx.s – отключите автоматическую конфигурацию функцией SystemInit()
- Разкомментируйте строку 110 в main.c (запустите фукнцию конфигурации тактирования RCC Config()



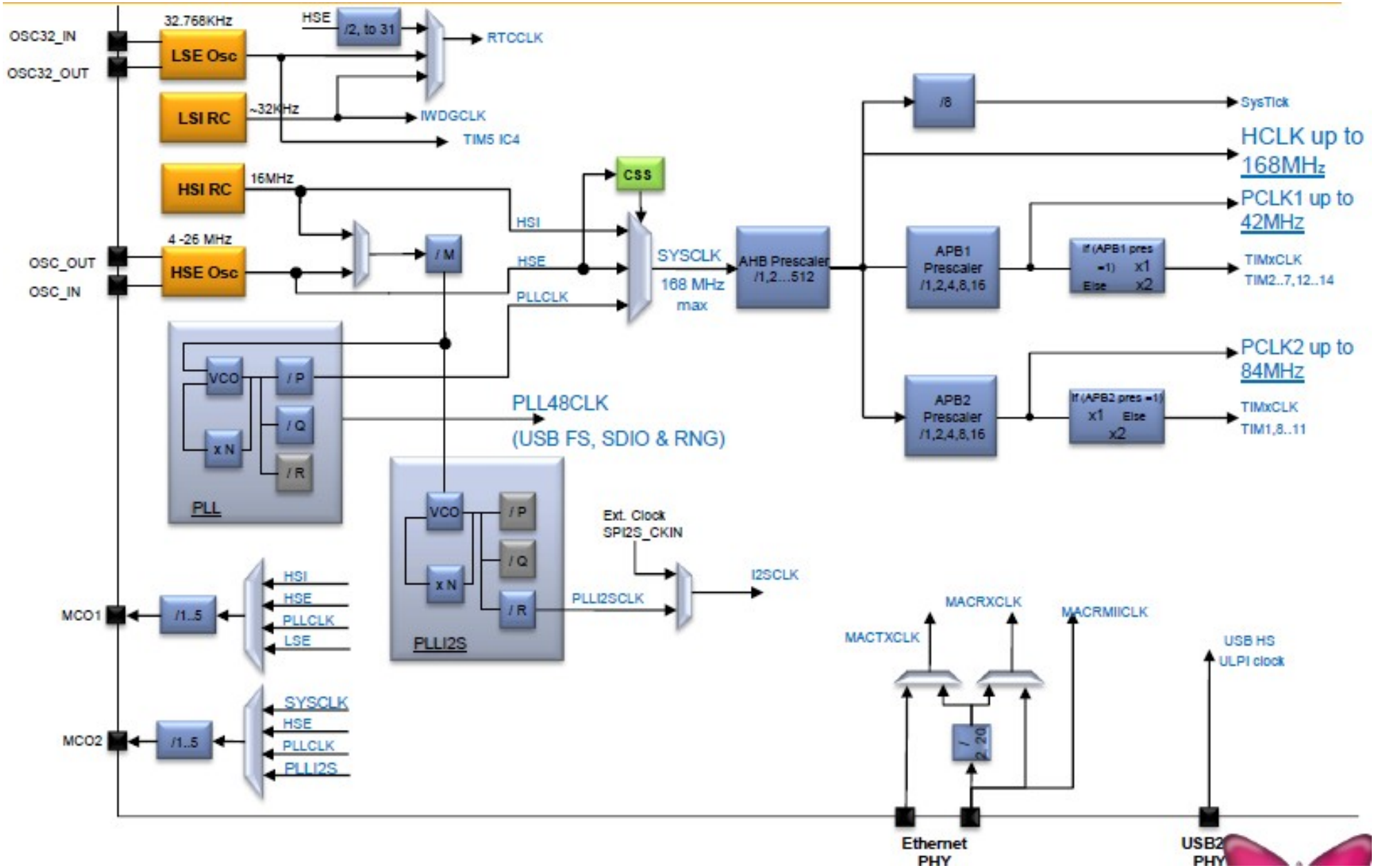
Настройка тактовой частоты - решение

- Расчитайте значение параметров M,N,P,Q и подставте их в функцию RCC_Config() (файл main.c). Требуемые частоты выделены красным.
- Закомментируйте строку 104 (/**/ или //) в startup_stm32f4xx.s – отключите автоматическую конфигурацию функцией SystemInit()
- Разкомментируйте строку 110 в main.c (запустите фукнцию конфигурации тактирования RCC Config()



/M [0..63]	*N [192..432]	/P [2,3,6,8]	/Q [4..15]
8	336	2	7

Схема тактирования – общий вид

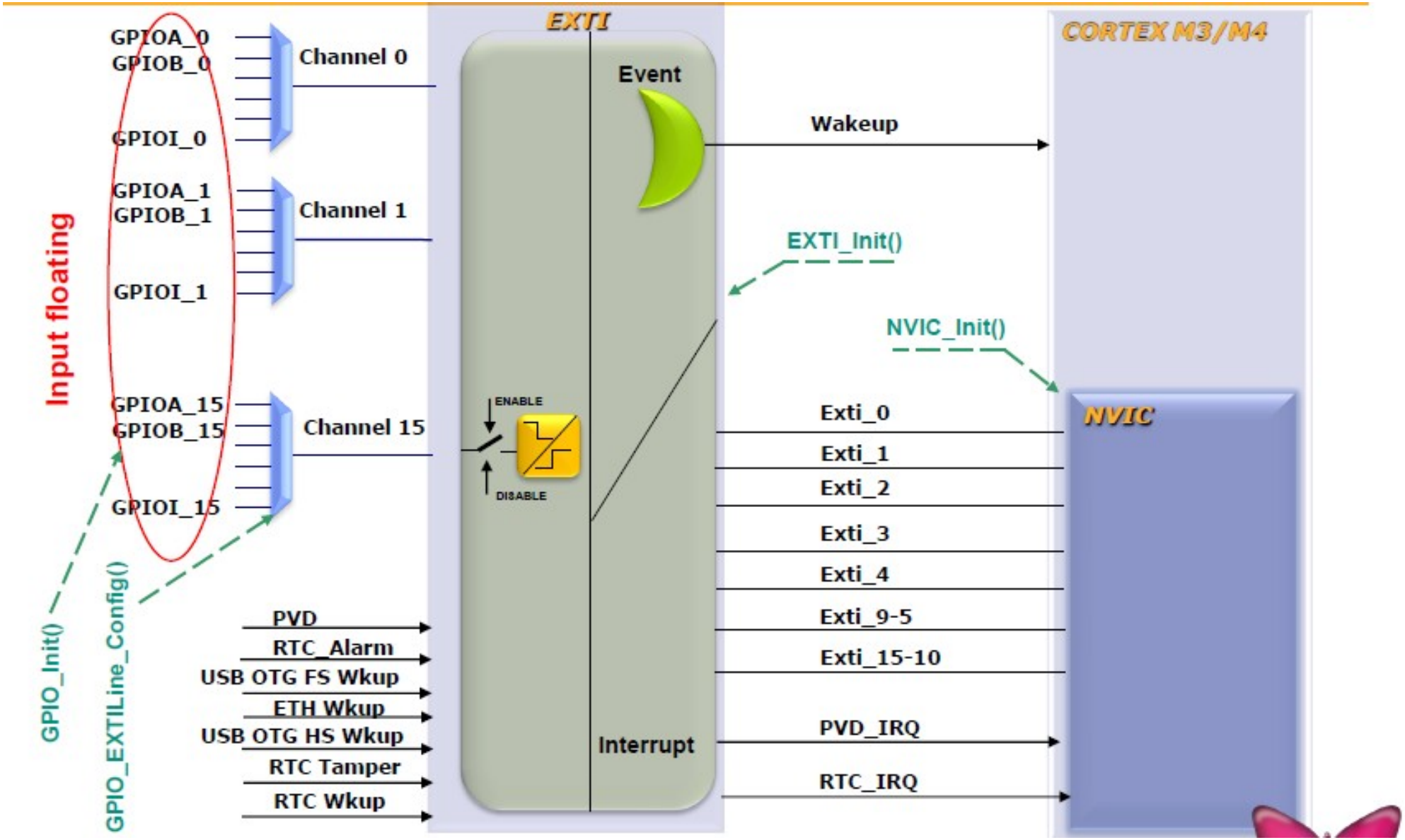


Прерывания – теория

- После **сброса** все прерывания от переверии запрещены, таблица векторов прерываний расположена в начале Flash памяти.
- Прерывания должны быть:
  - Разрешены на периферии-> точный источник прерывания
  - Настроить NVIC (контроллер прерываний)-> приоритеты, расположение в памяти
  - Запрограммированы в stm32fxx_it.c-> тело процедуры
- Дополнительно внешние прерывания требуют:
  - Настройка приложенных линий как вход (GPIO module)
  - Указать режим – **событие или прерывание** (EXTI module)
  - Выберите порт,подходящий источнику прерывания выбранного канала.(тоесть Канал1 может ссылаться на Pin1 любого порта) (SYSCFG module)
  - Разрешить канал внешного прерывания (EXTI module)
  - Выберите **синхронизацию** канала(по фронту или по спаду) (EXTI module)

EXTI module: form pin to NVIC

Модуль внешних прерываний: от вывода до контроллера прерываний



Настройка EXTI и NVIC

- **GPIO_InitTypeDef** -> выберите входной вывод и настройте в режиме входа.
- **GPIO_EXTILineConfig**-> настройте входной мультиплексор.
- **EXTI_InitTypeDef:**
  - **EXTI_Line->**EXTI_Line0..15
  - **EXTI_Mode:**
    - EXTI_Mode_Event (обращение к ядру без вызова прерывания)
    - EXTI_Mode_Interrupt
  - **EXTI_Trigger:**
    - EXTI_Trigger_Rising
    - EXTI_Trigger_Falling
  - **EXTI_LineCmd:**
    - ENABLE (переключиться на канал)
    - DISABLE
- **NVIC_InitTypeDef:**
  - **NVIC_IRQChannel: PPP_IRQn *)**
  - **NVIC_IRQChannelPreemptionPriority:0..15** (ниже номер – выше приоритет)
  - **NVIC_IRQChannelSubPriority**
  - **NVIC_IRQChannelCmd -> ENABLE/DISABLE**

*) - PPP – имя вектора прерывания определено в stm32f4xx.h

Прерывания - задача

Исправте функцию **Button_Config()**, чтобы заставить кнопку User работать:

- Настройте GPIOA, pin0 как вход (GPIO module)
- Настройте pin на работу с портом (периферия отключена) (SYSCFG module)
- Настройте режим (прерывания) и синхронизацию (по фронту) (EXTI module)
- Настройте вектор прерывания и его приоритетность (NVIC module)

Исправте функцию обработки прерывания EXTI0-IRQHandler() в stm32f4xx_it.c

- Очистить флаг прерывания

Прерывания - решения

Исправте функцию **Button_Config()**, чтобы заставить кнопку User работать:

- Настройте GPIOA, pin0 как вход (GPIO module)

  после сброса можно оставить без изменений

- Настройте pin на работу с портом (периферия отключена) (SYSCFG module)

  SYSCFG_EXTILineConfig(EXTI_PortSourseGPIOA, EXTI_PinSource0);

- Настройте режим (прерывания) и синхронизацию (по фронту) (EXTI module)

  EXTI_InitStucture.EXTI_Line = EXTI_Line0;
  EXTI_InitStucture.EXTI_Mode = EXTI_Mode_Interrupt;
  EXTI_InitStucture.EXTI_Trigger = EXTI_Trigger_Rising;
  EXTI_InitStucture.EXTI_LineCmd = ENABLE;
  EXTI_Init(&EXTI_InitStructure);

- Настройте вектор прерывания и его приоритетность (NVIC module)

  NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;

Исправте функцию обработки прерывания EXTI0-IRQHandler() в stm32f4xx_it.c

- Очистить флаг прерывания

EXTI_ClearITPendingBit(EXTI_Line0);

Настройка Таймера - функции

- Запустить тактирование RCC module (APBx bus) используя функцию:

  RCC_APBxPeriphClockCmd()

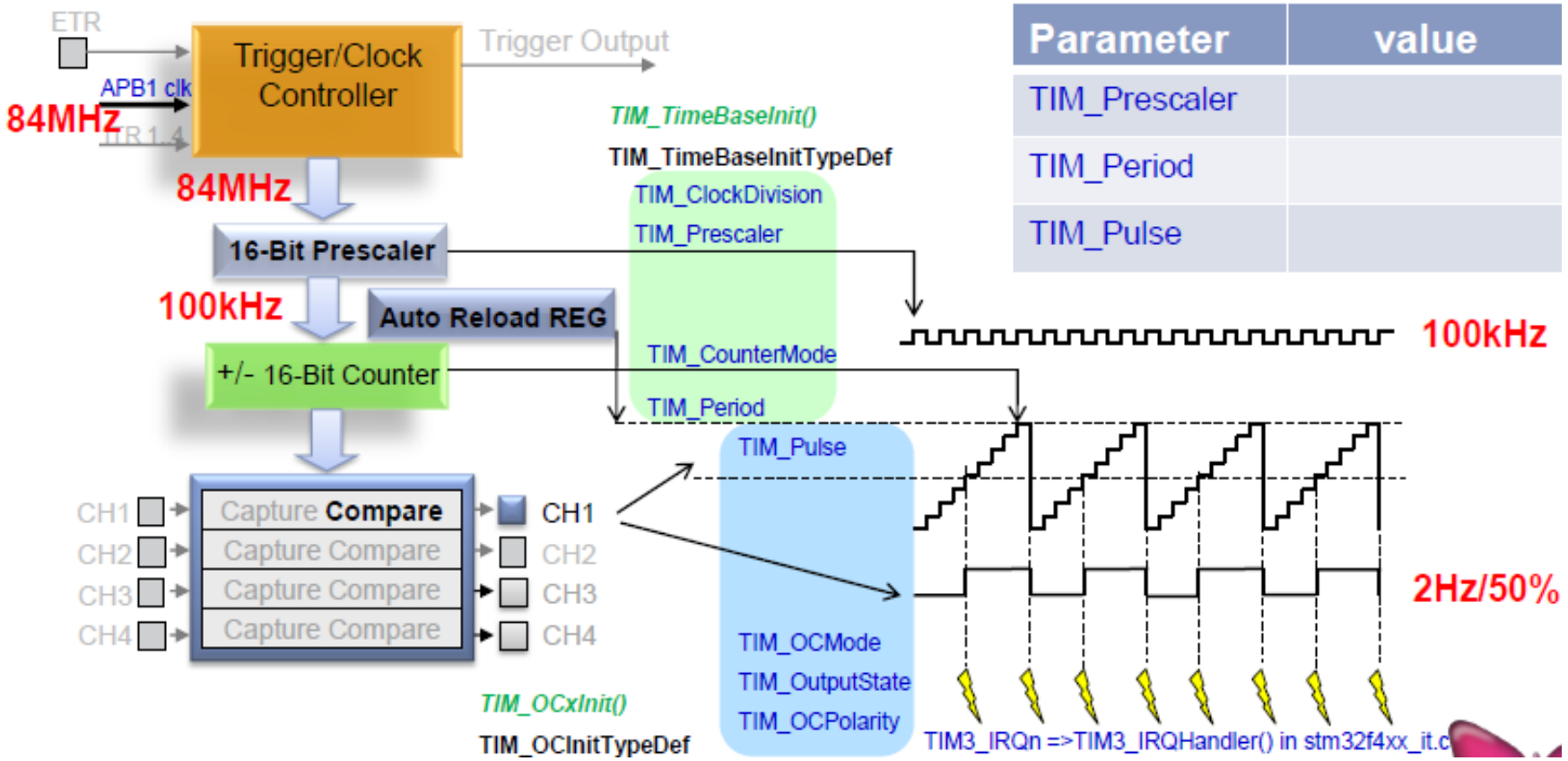
- Настроить модуль таймера используя **TIM_TimerBaseInitTypeDef** структуру в функции **TIM_TimerBaseInit()**
- Настроить **TIM_OCInitTypeDef** структуру и функцию **TIM_OCxInit()** для выбранных каналов таймера
- Инициализировать предзагрузку автозагружающего регистра (preload autoreload register) используя **TIM_OCxPreloadConfig()** и предзагрузку регистра захвата и сравнения **TIM_ARRPreloadConfig**() для каждого канала
- Включить таймер **TIM_Cmd()**

Настройка Таймера - структуры

- Для настройки канала ШИМ генератора нужно заполнить две структуры:
  - Настройка времени -> **TIM_TimeBaseInitTypeDef**
    - **TIM_Period –** автоматически перезагружается
      - F = TIM_counter_clk/(Period + 1)
    - **TIM_Prescaler** – [0->2^16-1] – значение предделителя таймера 3
      -
    - **TIM_ClockDivision** – используется для входного цифрового фильтра, можно оставить 0
      - 0
    - **TIM_CounterMode –**TIM_CounterMode_[Up/Down/CenterAligned1..3]
      - TIM_Counter_Mode_Up
  - Модуль захвата и сравнения для канала X -> TIM_OCInitTypeDef
    - **TIM_OCMode** – различные выходные настройки модуля сравнения
      - TIM_OCMode_PWM1
    - **TIM_OutputState** – входной режим:захват разрешен, выходной: выход разрешен
      - TIM_OutputState_Enable
    - **TIM_Pulse** - [0->2^16-1] – значение регистра захвата для канала X
      - Duty_cycle = (TIM_Pulse/TIM_Period)*100%
    - **TIM_OCPolarity –** выходной сигнал, высокий или низкий
      - TIM_OCPolarity_High

Таймер3 – Режим сравнения - задача

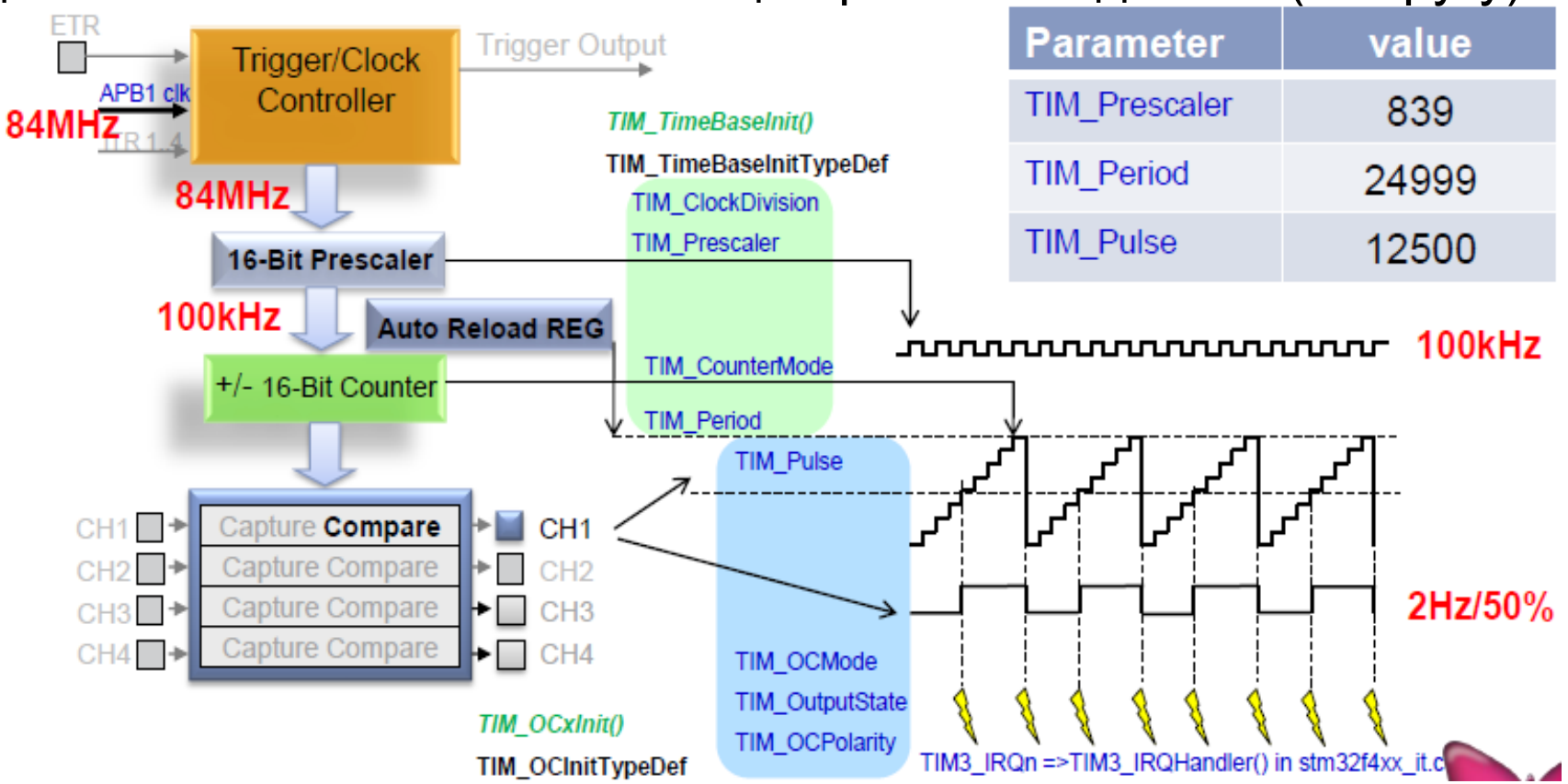
- Исправте TIM3_Config() в файле main.c для того чтобы настройть таймер3 на работу в режиме сравнения для генерации прерываний по переполнению с частотой 2Гц с выходом на канал1. Прерывания таймера3 контроллируют светодиоды LD3..6.
- Результат: в LED_Circle state в главном цикле светодиоды LD3..6 должны светить с частотой 1Гц и фазовым сдвигом (по кругу)



Таймер3 – Режим сравнения - решение

- Исправте TIM3_Config() в файле main.c для того чтобы настройть таймер3 на работу в режиме сравнения для генерации прерываний по переполнению с частотой 2Гц с выходом на канал1. Прерывания таймера3 контроллируют светодиоды LD3..6.

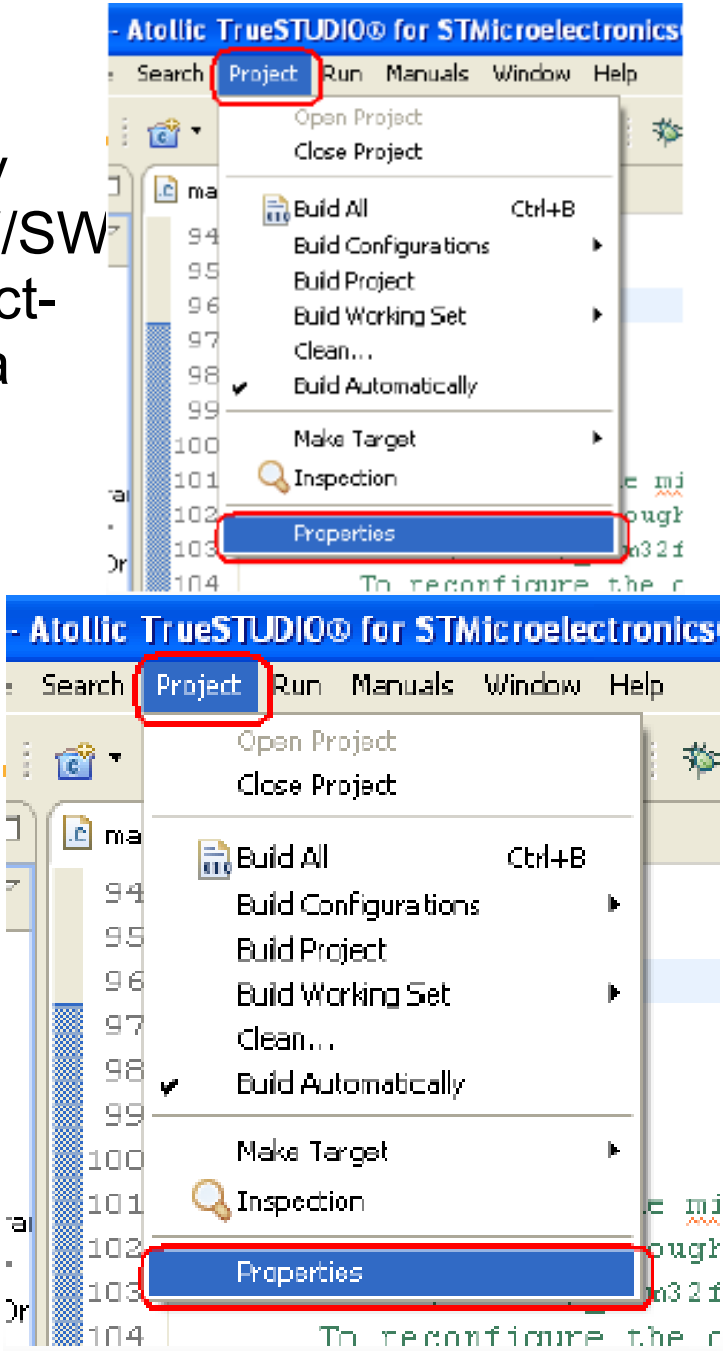
- Результат: в LED_Circle state в главном цикле светодиоды LD3..6 должны светить с частотой 1Гц и фазовым сдвигом (по кругу)



Parameter	value
TIM_Prescaler	839
TIM_Period	24999
TIM_Pulse	12500

Настройка математического сопроцессора (FPU) в STM32 TrueStudio

- Чтобы использовать аппаратную поддержку вычислений с плавающей запятой: "Mix HW/SW implementation" в настройках проекта: Project->Properties->C/C++ Build->Settings->вкладка Tool Setting, во всех окнах: Assembler, C Compiler и C Linker.
- При выходе из режима отладки, программа продолжает работать. Перезагрузка МК вызовет перезагрузку программы
- На плате STM32F4_Discovery находиться встроенный отладчик STLink, работающий исключительно в режиме SWD.



FPU - задача

- Функция FPU_Test() внутри раздела LED_Blink, основного цикла
- Внутри функции реализовано 2 генератора случайных числе с плавающей точкой, а также основные арифметические операции
- Используя System Timer(SysTick) – можно узнать сколько системных тактов необходимо для решения каждой операции:
  - **time_add** - сложение
  - **time_sub** - вычитание
  - **time_mul** - умножение
  - **time_div** - деление

No FPU	FPU

- Используя настройки Atollic STM32 TrueStudio, можно вкл/откл аппаратную поддержку операций с плавающей запятой -> пожалуйста, обратите внимание на предыдущий слайд

  Задача: проверте сколько тактов занимает решение с вкл и откл математическим сопроцессором

FPU - решение

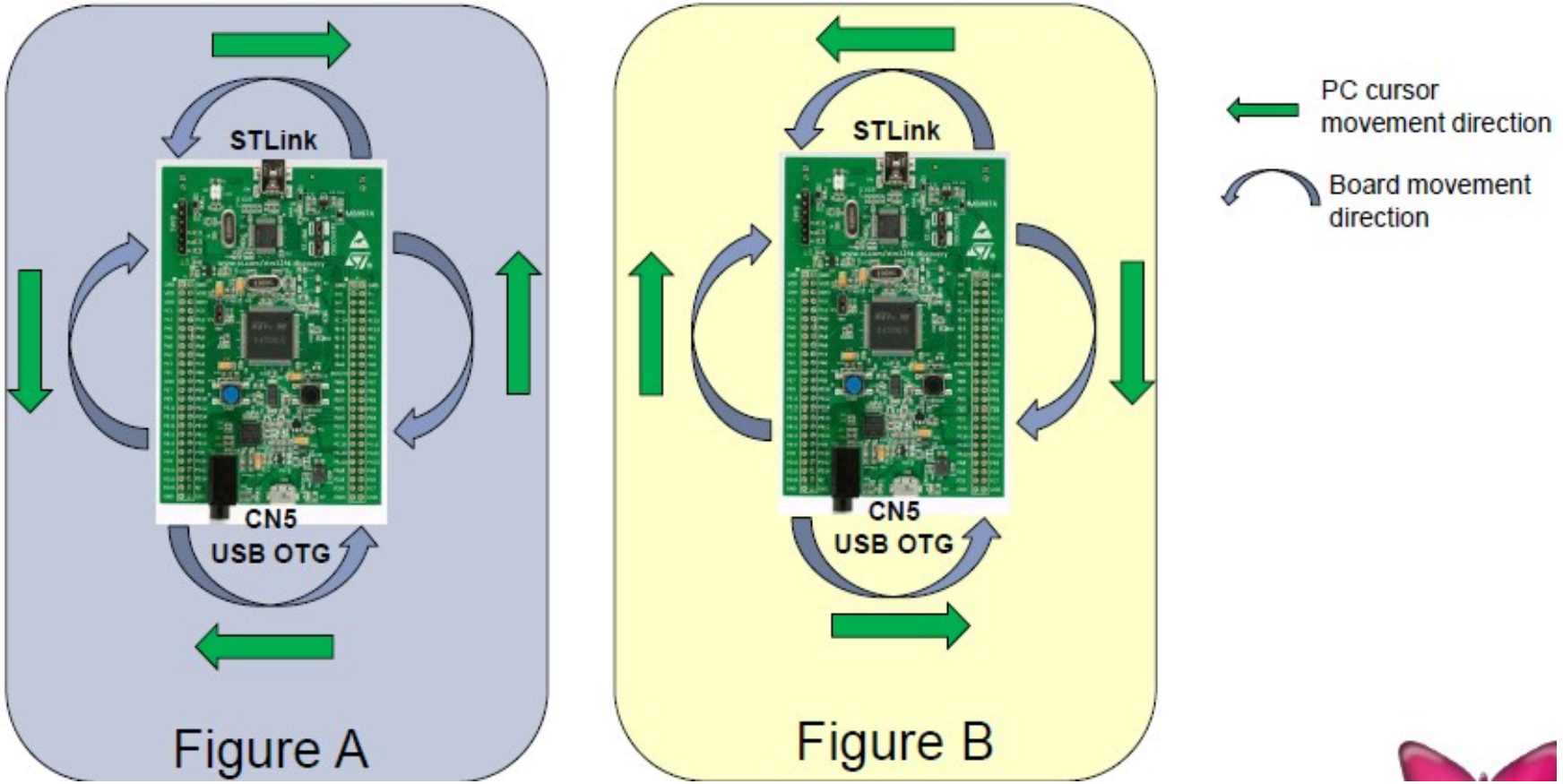
- Функция FPU_Test() внутри раздела LED_Blink, основного цикла
- Внутри функции реализовано 2 генератора случайных числе с плавающей точкой, а также основные арифметические операции
- Используя System Timer(SysTick) – можно узнать сколько системных тактов необходимо для решения каждой операции:
  - **time_add** - сложение
  - **time_sub** - вычитание
  - **time_mul** - умножение
  - **time_div** - деление

No FPU	FPU
160	19
165	19
120	19
247	30

- Используя настройки Atollic STM32 TrueStudio, можно вкл/откл аппаратную поддержку операций с плавающей запятой -> пожалуйста, обратите внимание на предыдущий слайд

Точная настройка приложения - задача

- Подпрограмма MEMS_Mouse работает в режиме "реверсивная ориентация", как в самолете(рисунок A)
- Задача заключается в изменении ориентации как показано на рисунке B. Ответ на этот вопрос находится в stm32f4xx_it.c, смотри строки 330..360.



Дополнительная информация

STM32F4xx are here
http://www.emcu.it/STM32F4xx/STM32F4xx.html

STM32 motor control is here
http://www.emcu.it/STM32/STM32-MotorControl/STM32-MotorControl.html

STM32W is here
http://www.emcu.it/STM32/STM32Wxx/STM32Wxx.html

SPEAr is here
http://www.emcu.it/SPEAr/SPEAr.html

ST-Link-v2 is here
http://www.emcu.it/ST-LINKv2/ST-LINKv2.html

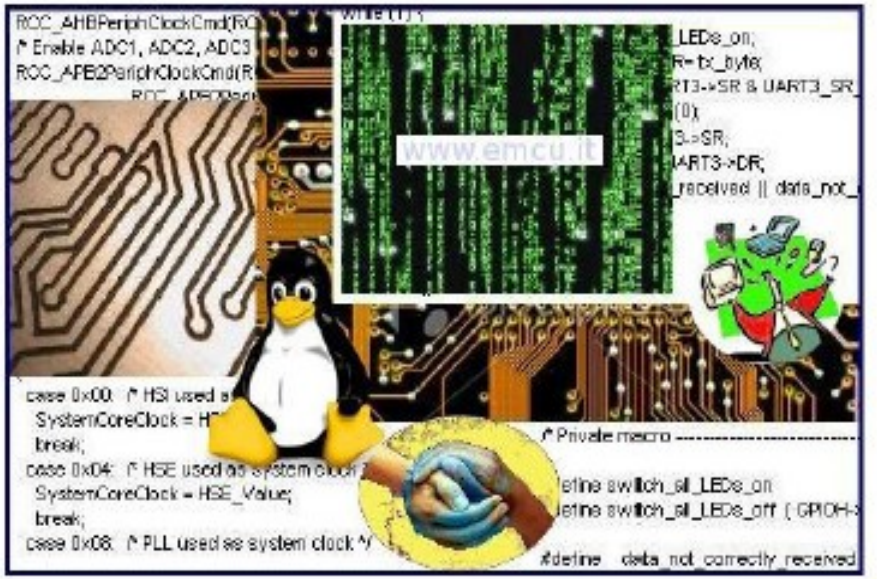
M24LRxx memory + RFID is here
http://www.emcu.it/MEMeRFID/MemAndRFID.html

Power Line Module is here
http://www.emcu.it/PLM/PLM.html

MEMS is here
http://www.emcu.it/MEMS/MEMS.html

ATOLLIC tips and tricks are here
http://www.emcu.it/ARM_Compiler/ATOLLIC/ATOLLIC_tips_and_tricks.html

WWW.EMCU.IT



In general information concerning STM32xx, STM8xx, MEMS, RFID, SMART Meter, ZigBee, Blue Tooth, etc are here