Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра безопасности информационных систем (БИС)

Кафедра микроэлектроники, информационных технологий и управляющих систем (МИТиУС)

RST\_CLK

Отчет по лабораторной работе №1

по дисциплине «Аппаратные Средства Телекоммуникационных Систем»

Студенты гр. 735:

\_\_\_\_\_\_\_\_ Д. А. Осипов

\_\_\_\_\_\_\_\_ Е. Ю. Борисова

\_\_\_\_\_\_\_\_ И. С. Забатурина

\_\_\_\_\_\_\_\_

Принял:

Ст. преподаватель кафедры МИТУС

\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ С. П. Недяк

(оценка)

Инженер кафедры МИТУС

\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ Ю. Б. Шаропин

(оценка)

\_\_\_\_\_\_\_\_

(дата)

1 Введение

1.1 Техническое задание

Цель лабораторной работы:

Изучить модуль сброса и тактовых частота по технической документации на МК 1986ВЕ9х.

Задачи:

1. Изучить код примера из библиотеки SPL lib\MDR32F9\_1986ВЕ4\_2015\Examples\MDR1986VE9x\MDR32F9Q3\_EVAL\RST\_CLK

2. Запустить программу примера на отладочной плате, объяснить результаты.

3. Поменять частоту работу процессора, объяснить результаты:

3.1 Установить тактовую частоту 32МГц.

3.2 Экспериментально ее измерить.

1.2 О плате LDM-BB-K1986BE92QI

LDM-BB-K1986BE92QI – отладочная плата на основе 32-разрядного ARM Cortex-M3 микроконтроллера K1986BE92QI от компании Миландр (Россия). Микроконтроллер выполнен в корпусе LQFP-64. Флэш-память программ составляет 128 Кб, SRAM 32 Кб. Тактовая частота – до 80 МГц, количество линий I/O – 43. Микроконтроллер имеет ряд популярных интерфейсов, среди которых USB Device и Host (до 12 Мбит/сек), 2 хUART, 2 xCAN, 2 xSPI, I2C. В структуре МК имеются также 12-разрядный АЦП (8 каналов) и один 12-разрядный одноканальный ЦАП. (Рисунок 1.1)

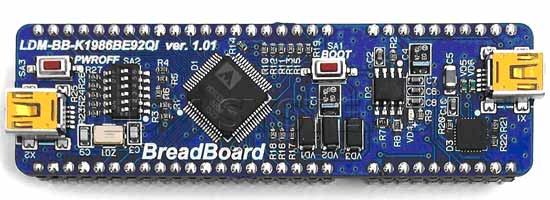


Рисунок 1.1 – Отладочная плата LDM-BB-K1986BE92QI. Вид сверху

1.3 Спецификация на МК 1986ВЕ9х: Изучение модуля сброса

При включении питания вырабатывается **внутренний сигнал сброса POR** (power-on reset) для цифровой части, питание UCC нарастает и, пока оно не превысило уровень 2,0 В, сигнал сброса POR удерживается; после превышения данного уровня сигнал POR выдается еще на протяжении ~ 4 мс для того, чтобы гарантировано установилось напряжение питания, после чего сигнал POR снимается, и схема может начать работать. (Рисунок 1.2)

Т.е. есть некоторый сигнал который не дает работать микроконтроллеру, пока нарастающее напряжение питания не установится в заданном уровне, после чего применяется сигнал еще 4мс (необходимо для того чтобы удостовериться, что напряжение не начнет уменьшаться (при включении питания могут быть скачки напряжения (называется дребезг контактов))).

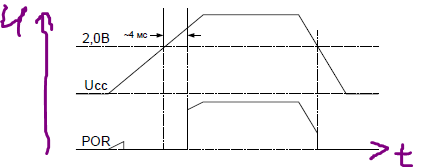


Рисунок 1.2 – Сигнал сброса при включении и выключении основного напряжения питания

При снижении напряжения питания UCC ниже уровня 2,0 В сигнал POR вырабатывается без задержки.

Сигнал POR также служит для переключения питания батарейного домена между BUCC и UCC.

При включении основного напряжения питания UCC автоматически включается встроенный регулятор напряжения для формирования напряжения DUCC питания цифрового ядра. В ходе работы микроконтроллера встроенный регулятор может быть отключен.

Микроконтроллер также может быть установлен в начальное состояние внешним сигналом сброса **RESET**, внутренними сигналами сброса сторожевых таймеров (**IWDG и WWDG**) или программным сбросом. При этом сигнал сброса формируется специальной схемой сброса, содержащий фильтр «иголок» по сигналу сброса и одновибратор для увеличения длительности сигнала сброса. (Рисунок 1.3)

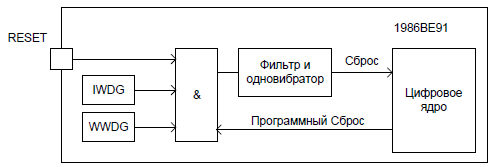


Рисунок 1.3 – Структурная блок-схема сброса

При приходе импульсов сброса длительностью менее 10 нс эти импульсы отфильтровываются и не приводят к сбросу процессора. Если длительность импульса больше 200 нс, вырабатывается сигнал сброса. При этом длительность сформированного сигнала сброса будет не менее 20 мкс.

Импульс < 10 нс – НЕ ЧИТАТЬ;

Импульс > 200 нс – СИГНАЛ СБРОСА (одновибратор увеличивает длительность сигнала сброса)

Текст. (Рисунок 1.4)

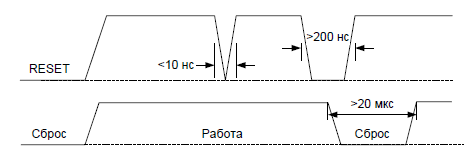


Рисунок 1.4 – Формирование сигнала сброса

1.4 Спецификация на МК 1986ВЕ9х: Изучение модуля тактовых частот RST\_CLK

Микроконтроллер имеет два встроенных (HIS (8 МГц) и LSI (40 кГц)) и два внешних (HSE (2…16 МГц) и LSE (32 кГц)) генератора, а также специализированный блок формирования тактовой синхронизации микроконтроллера (две схемы умножения тактовой частоты PLL для ядра и USB интерфейса).

Управление тактовыми частотами ведется через периферийный блок RST\_CLK.

(При включении питания микроконтроллер запускается на частоте HSI генератора.)

Выдача тактовых сигналов синхронизации для всех периферийных блоков, кроме RST\_CLK, отключена.

Для начала работы с нужным периферийным блоком необходимо включить его тактовую частоту в регистре PER\_CLOCK.

Некоторые контроллеры интерфейсов (UART, CAN, USB, Таймеры) могут работать на частотах, отличных от частоты процессорного ядра, поэтому в соответствующих регистрах (UART\_CLOCK, CAN\_CLOCK, USB\_CLOCK, TIM\_CLOCK) могут быть заданы их скорости работы.

Для изменения тактовой частоты ядра можно перейти на другой генератор и/или воспользоваться блоком умножения тактовой частоты. Для корректной смены тактовой частоты сначала должны быть сформированы необходимые тактовые частоты и затем осуществлено переключение на них на соответствующих мультиплексорах, управляемых регистрами CPU\_CLOCK и USB\_CLOCK. (Рисунок 1.5)

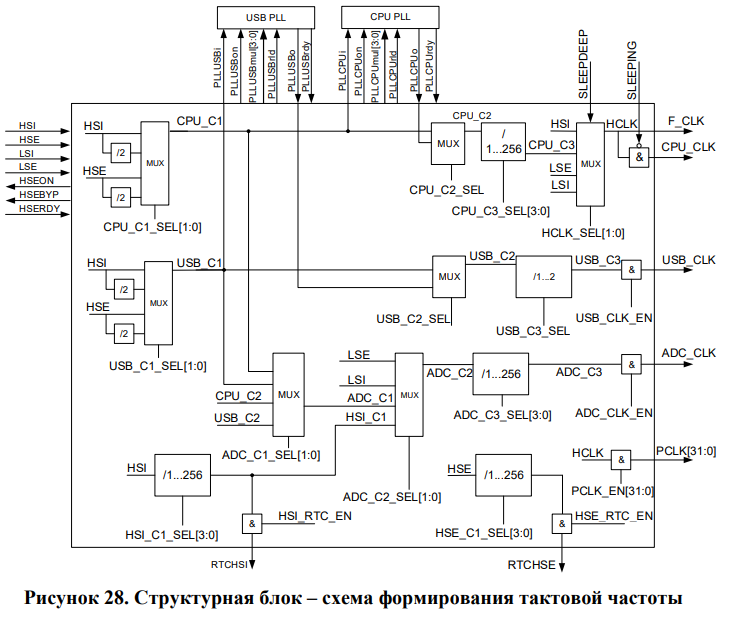


Рисунок 1.5 – Структурная блок-схема формирования тактовой частоты

Где,

MUX – мультиплексор, обеспечивает выбор линии тактирования,

ADC – АЦП,

PCLK – Pereferio CLK – 31 периферийных устройства и для них идет подстройка частоты тактовой,

RTC – таймер реального времени,

Сигнал HCLK является тактовой частотой процессора от него тактируется периферия PCLK.

Остальное мы рассмотрим подробнее. Источником тактовой частоты может быть HSI и HSE (с возможностью умножения, если настроить блок CPU\_PLL) или LSI и LSE.

HIS – higth speed **Internal** - Источник высокой частоты **внутренней**

HSE – higth speed **external** - Источник высокой частоты **внешний**

LSI – low speed **internal** – Источник низкой частоты **внутренней**

LSE – low speed **external** – Источник низкой частоты **внешний**

**Встроенный RC генератор HSI**

Генератор HSI вырабатывает тактовую частоту fO\_HSI с типовым значением 8 МГц. Генератор автоматически запускается при появлении питания UCC и при выходе в нормальный режим работы вырабатывает сигнал HSIRDY в регистре батарейного домена BKP\_REG\_0F. Первоначально процессорное ядро запускается на тактовой частоте HSI. При дальнейшей работе генератор HSI может быть отключен при помощи сигнала HSION в регистре BKP\_REG\_0F. Так же генератор может быть подстроен при помощи сигнала HSITRIM в регистре BKP\_REG\_0F.

**Встроенный RC генератор LSI**

Генератор LSI вырабатывает тактовую частоту fO\_LSI с типовым значением 40 кГц. Генератор автоматически запускается при появлении питания UCC и при выходе в нормальный режим работы вырабатывает сигнал LSIRDY в регистре BKP\_REG\_0F. Первоначально тактовая частота генератор LSI используется для формирования дополнительной задержки tpor. При дальнейшей работе генератор LSI может быть отключен при помощи сигнала LSION в регистре BKP\_REG\_0F.

**Внешний генератор HSE**

Генератор HSE предназначен для выработки тактовой частоты 2..16 МГц с помощью внешнего резонатора. Генератор запускается при появлении питания UCC и сигнала разрешения HSEON в регистре HS\_CONTROL. При выходе в нормальный режим работы вырабатывает сигнал HSERDY в регистре CLOCK\_STATUS. Также этот генератор может работать в режиме HSEBYP, когда входная тактовая частота с входа OSC\_IN проходит напрямую на выход HSE. Выход OSC\_OUT находится в этом режиме в третьем состоянии.

**Внешний генератор LSE**

Генератор LSE предназначен для выработки тактовой частоты 32 кГц с помощью внешнего резонатора. Генератор запускается при появлении питания BDUCC и сигнала разрешения LSEON в регистре BKP\_REG\_0F. При выходе в нормальный режим работы вырабатывает сигнал LSERDY в регистре BKP\_REG\_0F. Также осциллятор может работать в режиме LSEBYP, когда входная тактовая частота с входа OSC\_IN32 проходит напрямую на выход LSE. Выход OSC\_OUT32 находится в этом режиме третьем состоянии. Так как генератор LSE питается от напряжения питания BDUCC и его регистр управления BKP\_REG\_0F расположен в батарейном домене, то генератор может продолжать работать при пропадании основного питания UCC. Генератор LSE используется для работы часов реального времени.

**Встроенный блок умножения системной тактовой частоты**

Блок умножения позволяет провести умножение входной тактовой частоты на коэффициент от 2 до 16, задаваемый на входе PLLCPUMUL[3:0] в регистре PLL\_CONTROL. Входная частота блока умножителя должна быть в диапазоне 2…16 МГц выходная до 100 МГц. При выходе блока умножителя тактовой частоты в расчетный режим вырабатывается сигнал PLLCPURDY в регистре CLOCK\_STATUS. Блок включается с помощью сигнала PLLCPUON в регистре PLL\_CONTROL. Выходная частота используется как основная частота процессора и периферии.

**Встроенный блок умножения USB тактовой частоты**

Блок умножения позволяет провести умножение входной тактовой частоты на коэффициент от 2 до 16, задаваемый на входе PLLUSBMUL[3:0] в регистре PLL\_CONTROL. Входная частота блока умножителя должна быть в диапазоне 2…16 МГц, выходная должна составлять 48 МГц. При выходе блока умножителя тактовой частоты в расчетный режим вырабатывается сигнал PLLUSBRDY в регистре CLOCK\_STATUS. Блок включается с помощью сигнала PLLUSBON в регистре PLL\_CONTROL. Выходная частота используется как основная частота протокольной части USB интерфейса.

**Делители частоты и мультиплексоры** - эти элементы позволяют коммутировать внутренние цепи схемы тактирования. Коммутация позволяет создать гибкую конфигурацию тактирования всех устройств микроконтроллера.

По умолчанию модуль выбора источника частоты (MUX) настроен на прием с HSI. С частотой ничего не происходит и она, через HCLK (линия тактирования, совмещенная в том числе и с SysTick таймером) попадает в CPU\_CLK без изменений.

Отобразим основные линии тактирования для процессора. (Рисунок 1.6)

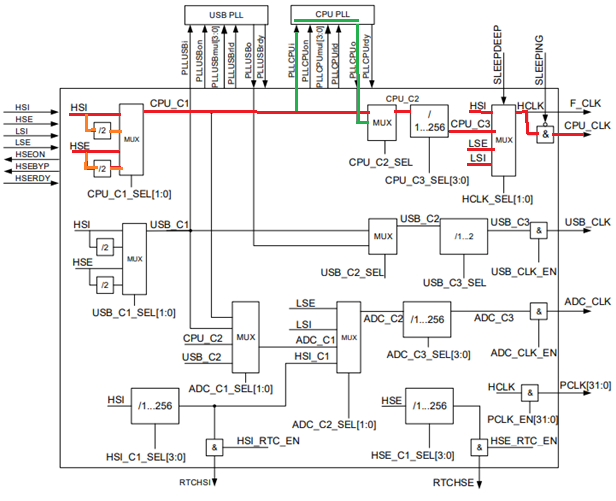


Рисунок 1.6– Линии подачи тактовой частоты на процессор

На рисунке 1.6 изображены линии тактирования процессора. С помощью мультиплексоров можно выбирать ту или иную линию, использовать напрямую источник тактовых сигналов HIS, LSE, LSI или же пропустить HIS или HSE через PLL и умножить частоту (отображено зеленым цветом) до нужного нам уровня, а потом если необходимо поделить ее на делителе.

CPU\_C1\_SEL – центральный процессор С1 SEL(выбор) – выбираем частоту тактирования процессора, которую можно будет послать на CPU\_PLL. (4 входа: HIS и HSE, поделенная на 2 или не поделенная). (Рисунок 1.7 – 1.8)

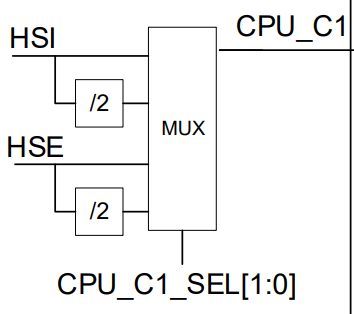


Рисунок 1.7 – Делители частоты

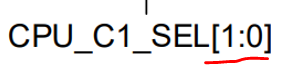


Рисунок 1.8 – Количество управляющих сигналов

CPU\_PLL – Фазовая автоподстройка частоты. На ее основе строятся блоки или схемы умножения частоты (они нужны чтобы повысить частоту до нужной).

Встроенный блок умножения системной тактовой частоты позволяет провести умножение входной тактовой частоты на коэффициент от 2 до 16, задаваемый на входе PLLCPUMUL[3:0] в регистре PLL\_CONTROL. Входная частота блока умножителя должна быть в диапазоне 2…16 МГц выходная до 100 МГц. При выходе блока умножителя тактовой частоты в расчетный режим вырабатывается сигнал PLLCPURDY в регистре CLOCK\_STATUS. Блок включается с помощью сигнала PLLCPUON в регистре PLL\_CONTROL. Выходная частота используется как основная частота процессора и периферии. (Рисунок 1.9)

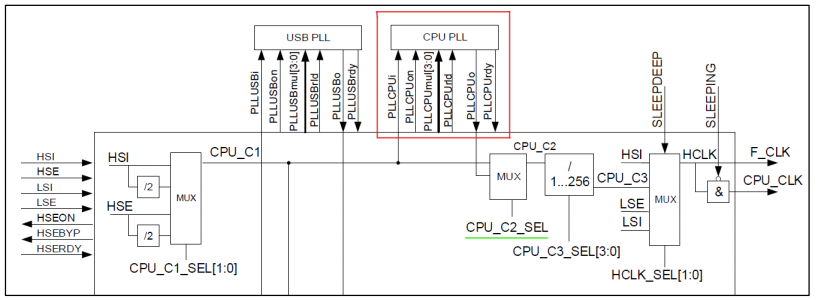


Рисунок 1.9 – Схема формирования тактовой частоты – Блок умножения частоты

Делитель частоты - электронное устройство, уменьшающее в целое число раз частоту подводимых к нему периодических колебаний. Как видно из схемы, делитель способен выполнять деление в диапазоне от 1 до 256. (Рисунок 1.10)

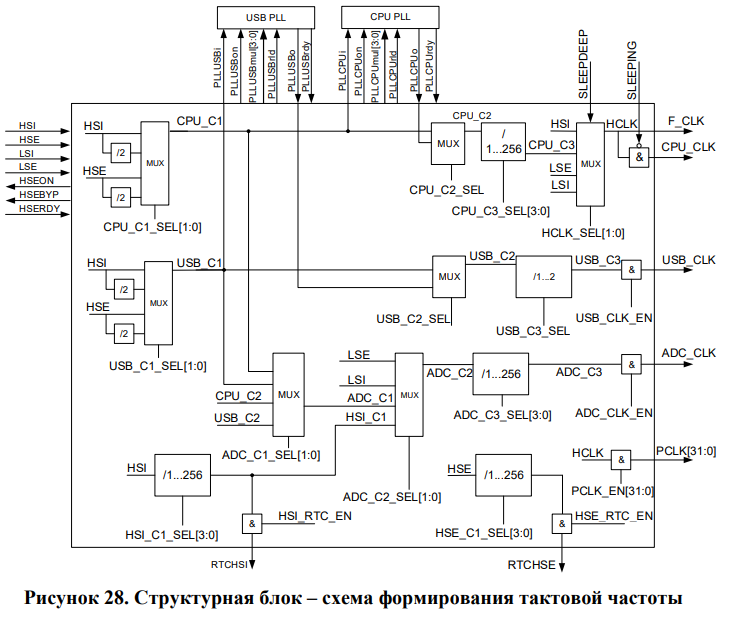


Рисунок 1.10 - Делитель

CPU\_CLK – CPU Clock –формирует частоту идущую на процессор.

Мультиплексор с управляющим сигналом HCLK\_SEL выбирает линию тактирования для CPU\_CLK и F\_CLK. (Рисунок 1.11)

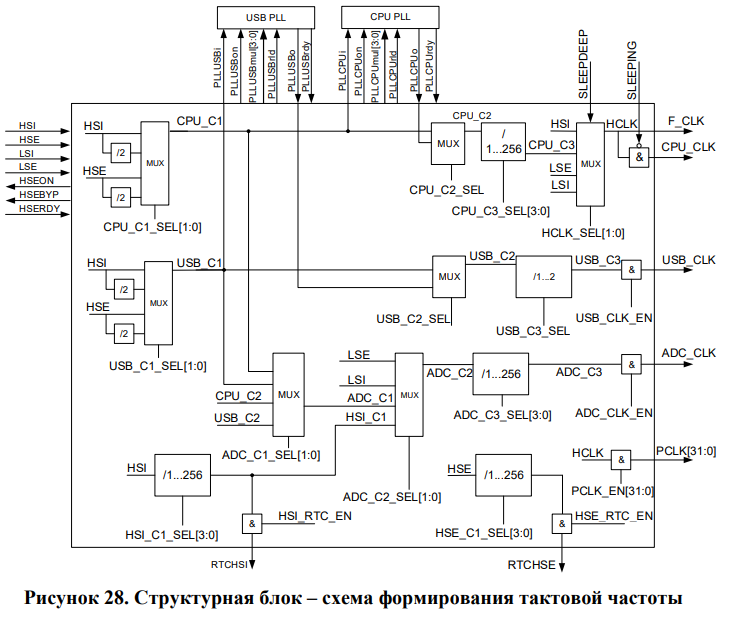


Рисунок 1.11 – Мультиплексор HCLK – Выбор сигнала тактирования процессора

За настройку тактирования отвечает блок контроллера тактовой частоты MDR\_RST\_CLK. (Рисунок 1.12)

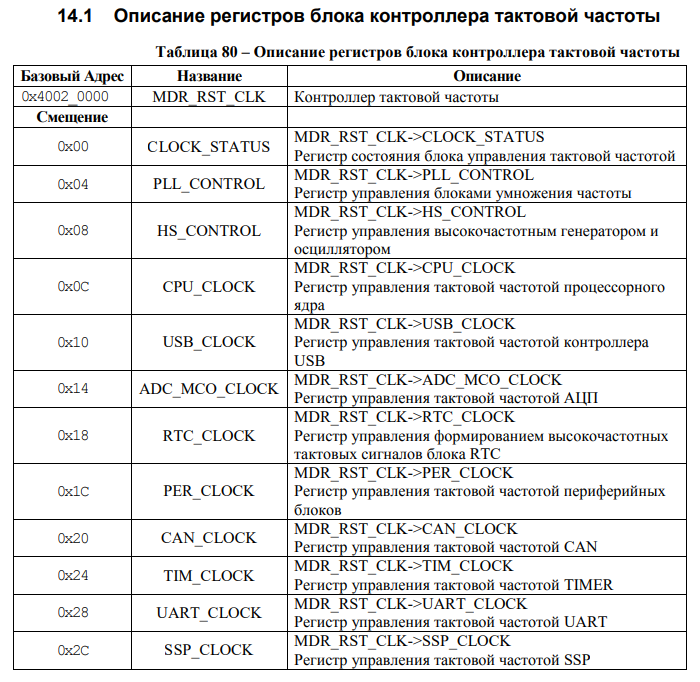


Рисунок 1.12 – Описание регистров блока контроллера тактовой частоты

Для настройки тактирования непосредственно процессора понадобится 3 регистра:

1) MDR\_RST\_CLK->PER\_CLOCK - Регистр управления тактовой частотой периферийных блоков;

2) MDR\_RST\_CLK->CLOCK\_STATUS - Регистр состояния блока управления тактовой частотой;

3) MDR\_RST\_CLK->PLL\_CONTROL - Регистр управления блоком умножения частоты;

4) MDR\_RST\_CLK->HS\_CONTROL - Регистр управления высокочастотным генератором и осциллятором;

5) MDR\_RST\_CLK->CPU\_CLOCK - Регистр управления тактовой частотой процессорного ядра.

В регистре PER\_CLOCK необходимо включить тактирование контроллера тактовой частоты RST\_CLK в блоке PCLK\_EN[4] (1<<0). На рисунке 1.13 в формате таблицы приведены регистры настройки тактирования периферийных блоков.

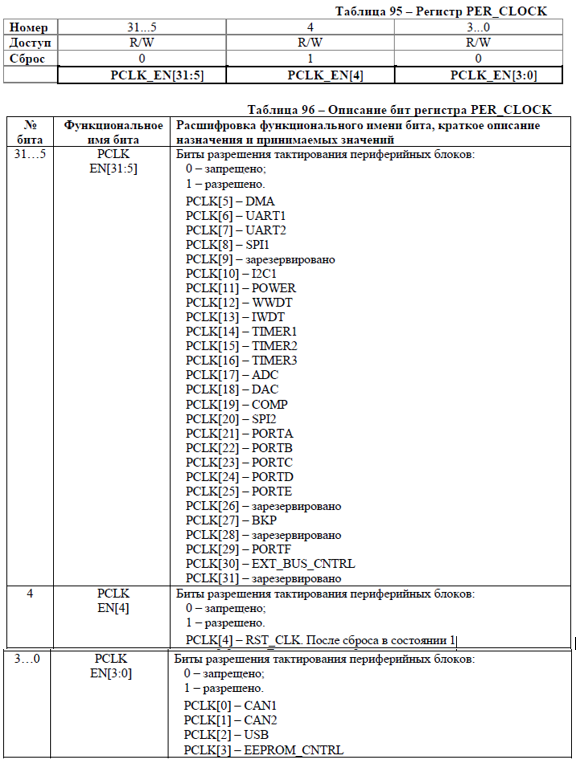


Рисунок 1.13 - Описание бит регистра управления тактовой частотой периферийных блоков (PER\_CLOCK)

Генератор HSE предназначен для выработки тактовой частоты 2..16 МГц с помощью внешнего резонатора. Генератор запускается при появлении питания UCC и сигнала разрешения HSE\_ON в регистре HS\_CONTROL. На рисунке 1.14 в формате таблицы приведены регистры настройки блока HS\_CONTROL.

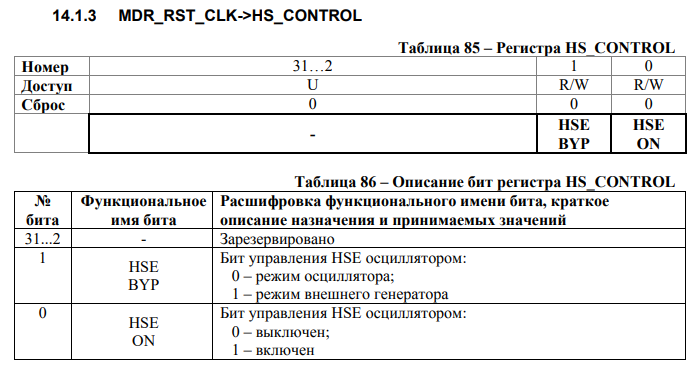


Рисунок 1.14 – Описание бит регистра управления высокочастотным генератором и осциллятором (HS\_CONTROL)

В регистре PLL\_CONTROL необходимо будет установить нужный коэффициент умножения в PLL\_CPU\_MUL (c 8 по 11 биты), включить PLL (PLL\_CPU\_ON - 1<<2), после чего перезагрузить PLL (если он был запущен до настройки коэффициента умножения) в PLL\_CPU\_RLD (1<<3). На рисунке 1.15 в формате таблицы приведены регистры настройки блока умножения частоты PLL.

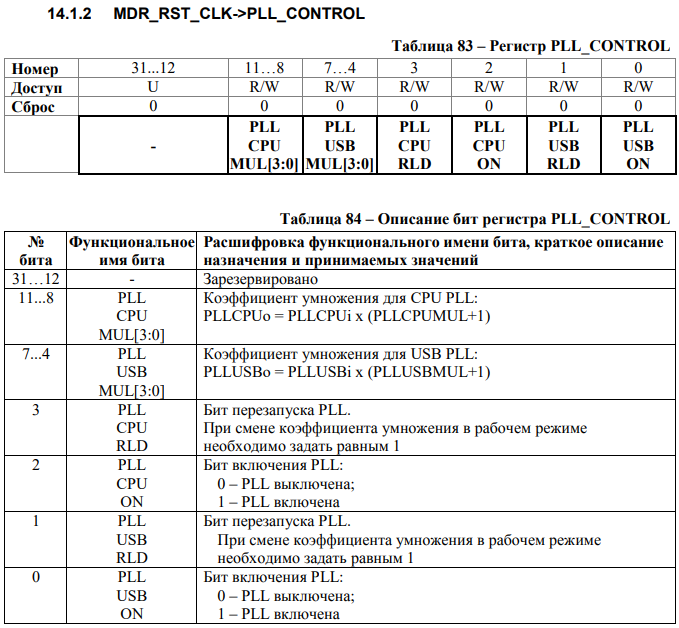


Рисунок 1.15 – Описание бит регистра управления блоком умножения частоты (PLL\_CONTROL)

В регистре CLOCK\_STATUS можно будет проверить флаги выхода в рабочий режим HSE (1<<2) и PLL (1<<1). (Рисунок 1.16)

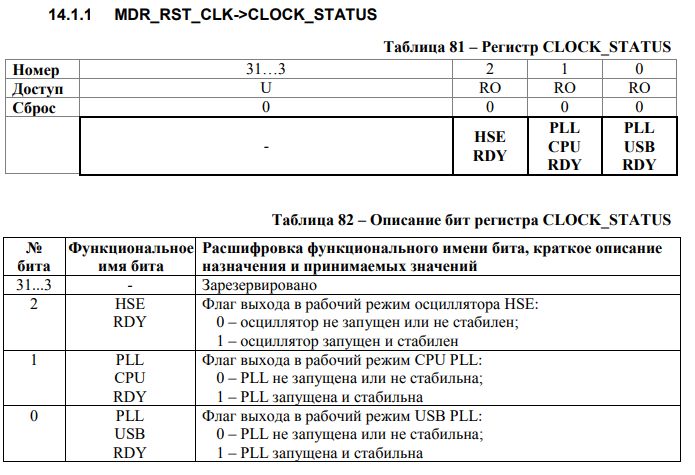


Рисунок 1.16 – Описание бит регистра состояния блока управления тактовой частотой (CLOCK\_STATUS)

В регистре CPU\_CLOCK выбирается путь тактирования и на рисунке 1.17 в формате таблицы приведены регистры настройки пути тактирования для HCLK. (Рисунок 2.17 – 2.21)

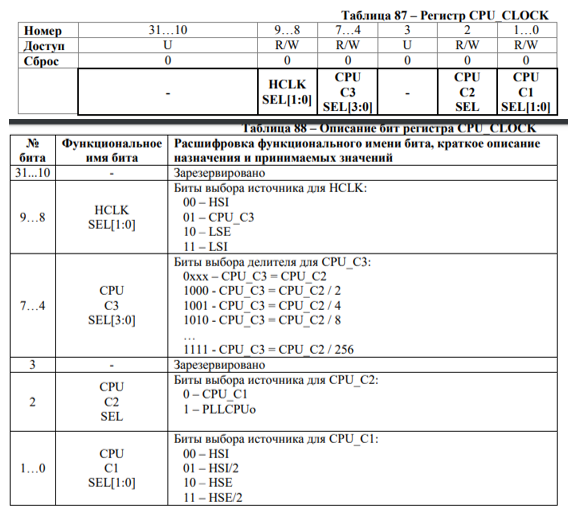


Рисунок 1.17 – Описание бит регистра управления тактовой частотой процессорного ядра (CLOCK\_STATUS)

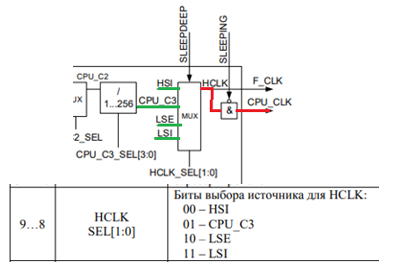


Рисунок 1.18 – Отображение на схеме линий тактирования (зеленый цвет) и связь с битами выбора источника для HCLK

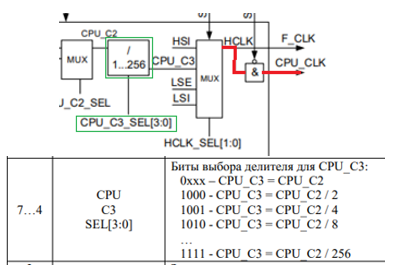


Рисунок 1.19 - Отображение на схеме блока делителя частоты (зеленый цвет) и связь с битами выбора делителя для CPU\_C3

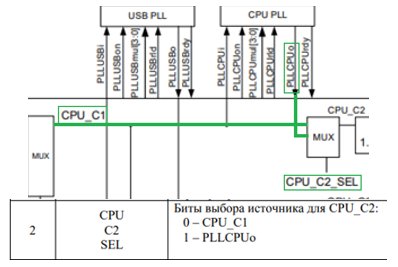


Рисунок 1.20 - Отображение на схеме линий тактирования (зеленый цвет) и связь с битами выбора источника для CPU\_C2

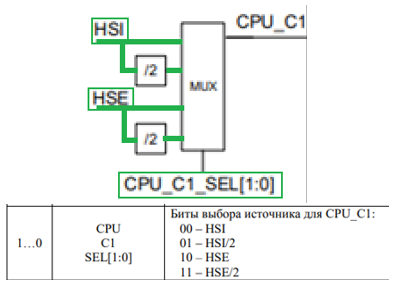


Рисунок 1.21 - Отображение на схеме линий тактирования (зеленый цвет) и связь с битами выбора источника для CPU\_C1

2 Ход работы

2.1 Прошивка

Для загрузки загрузчика на плату LDM потребуется программа «1986UARTWSD» и сам загрузчик. HEX-файл загрузчика находится в каталоге «..\Intec\MSTN\M100\boot\», а именно файл «MSTN\_Bootloader.hex»

Подключаем микроконтроллер как на картинке в порт «X1». (Рисунок 2.1)

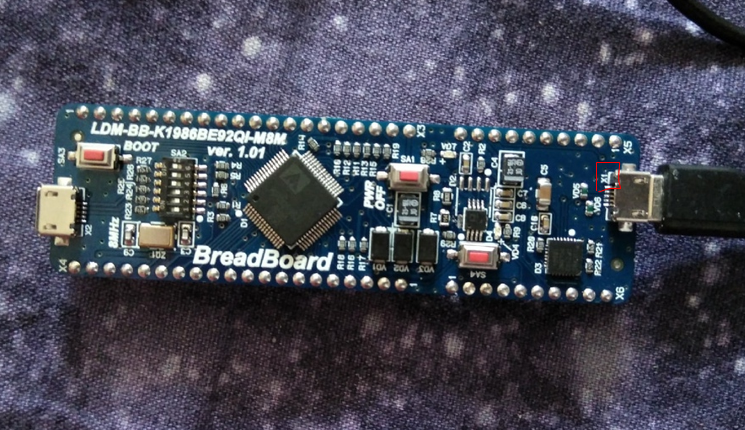


Рисунок 2.1 – Подключение платы для загрузки прошивки (Х1)

Затем переходим в «Управление компьютером»→«Диспетчер устройств». Находим новое, подключенное, устройство и запоминаем его COM-порт. (Рисунок 2.2)



Рисунок 2.2 – Просмотр к какому порту подключено устройство

Далее в программе необходимо указать данный порт. (Рисунок 2.3)

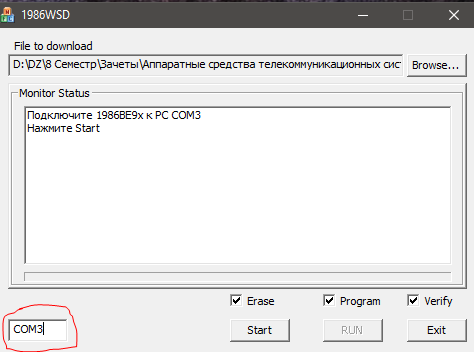


Рисунок 2.3 – Изменение ком порта в программе

Если каким либо образом не удается изменить порт в программе, переходим в файл «1986WSD.cfg» корневого каталога программы «1986WSD». В этом файле в конце, или воспользовавшись поиском по ключевому слову «COM», можно найти установленный в программе порт по умолчанию и заменить его на нужный порт. (Рисунок 2.4)

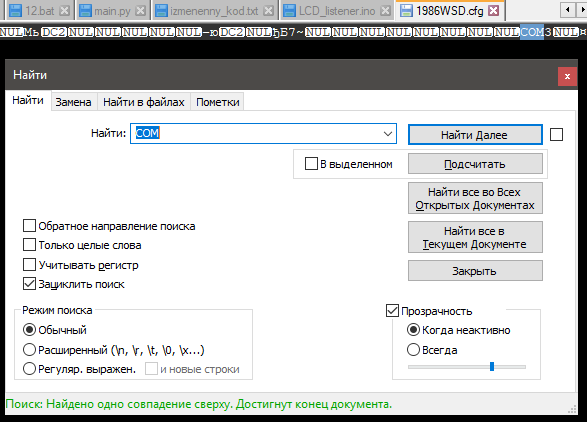


Рисунок 2.4 – Изменение ком порта в файле «1986WSD.cfg»

Далее в программе, нажав на кнопку «Browse» выбираем файл загрузчика, найденный в папке «intec». (Рисунок 2.5)

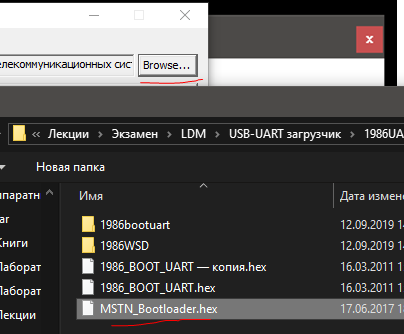


Рисунок 2.5 – Выбор файла загрузчика

Ставим галочки как тут. (Рисунок 2.6)

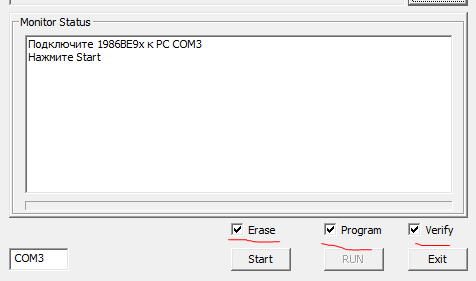


Рисунок 2.6 – Настройка программы

Для загрузки на плату необходимо перейти в специальный режим, зажав на пол секунды (секунду) кнопку SA1, по истечению времени нажать на кнопку SA3, после этого отпустить их по очереди, начиная с SA1. С первого раза может не получиться (об этом свидетельствует отмеченная строка «1»). Нажимаем «Start». (Рисунок 2.7 - 8)

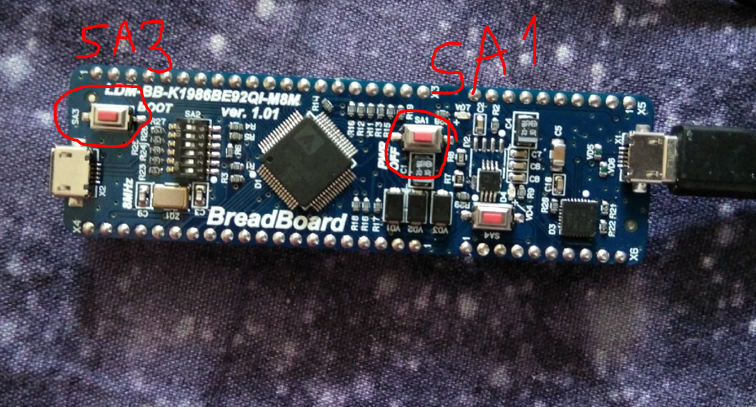


Рисунок 2.7 – Обозначение кнопок на плате для перевода микроконтроллера в специальный режим

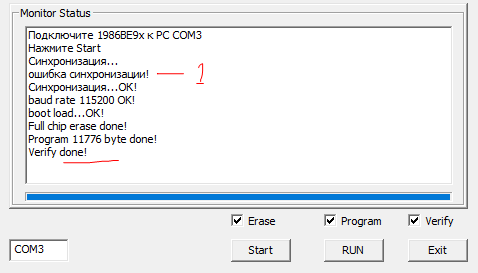


Рисунок 2.8 – Вывод программы об успешной загрузки прошивки на плату

После этого можно подключаться к плате через порт X2 для загрузки исполняемого кода программы. (Рисунок 2.9)

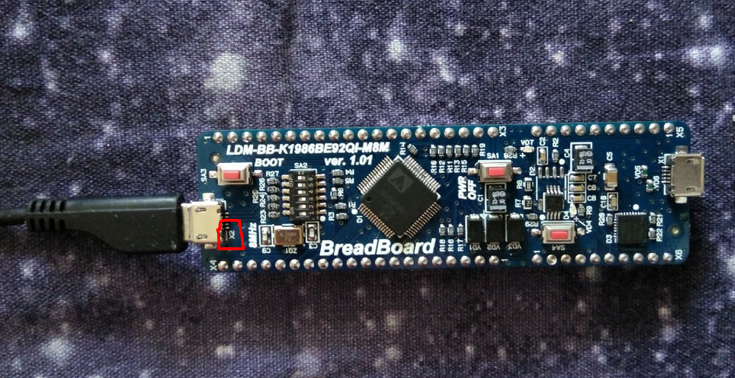


Рисунок 2.9 – Подключение платы для загрузки исполняемого кода (Х2)

Код скомпилированный в «.hex» формат можно загружать через «1986WSD» и порт «X1».

2.2 Обзор примера MDR32F9Q3 для RST\_CLK

В коде примера «Examples/MDR32F9Q3\_EVAL/RST\_CLK/CPU\_Clock/main.c» описываются функции инициализации портов на вывод для двух светодиодов (тема GPIO будет исследована в следующей лабораторной работе), заданы функции включения и выключения светодиодов, задана функция задержки циклом, на базе этих функций реализовано мигание одним светодиодом в случае ошибки «IndicateError» и мигание светодиодом с заданным количество раз и задержкой «BlinkLED1(uint32\_t num, uint32\_t del)».

В главной функции описывается установка тактовой частоты микроконтроллера с выходом в рабочий режим от источника HIS и миганием светодиодом заданное количество раз после выхода в рабочий режим, иначе мигает сигнал ошибки. (Рисунок 2.10)

После этого устанавливается тактирование процессора с частотой в 2 раза меньше от HSI с помощью предделителя, и происходит мигание светодиодом. (Рисунок 2.11)

После этого устанавливается тактирование от 7\*HSE/2 при помощи предделителя и умножителя, и происходит мигание светодиодом. (Рисунок 2.12)

В последней части кода устанавливается тактирование от LSI, и происходит мигание светодиодом. (Рисунок 2.13)

Частот работы микроконтроллера устанавливается путем умножения и/или деления частоты CPU\_PLL:

1. PLLscrHSEdiv1 - Деление.

2. PLLscrHSEmul1 - Умножение.

Тут вместо HSE может быть: HIS, LSE, LSI; а коэффициент деления/умножения меняется от 0 до 8.

|  |
| --- |
| /\* Set RST\_CLK to default \*/  RST\_CLK\_DeInit();  RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTF, ENABLE);  RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_BKP,ENABLE);  /\* 1. CPU\_CLK = HSI clock \*/  /\* Enable HSI clock source \*/  RST\_CLK\_HSIcmd(ENABLE);  if (RST\_CLK\_HSIstatus() == SUCCESS) /\* Good HSI clock \*/  {  /\* Select HSI clock on the CPU clock MUX \*/  RST\_CLK\_CPUclkSelection(RST\_CLK\_CPUclkHSI);  /\* LED1 blinking with HSI clock as input clock source \*/  BlinkLED1(BLINK\_NUM, BLINK\_DELAY);  }  else  { /\* HSI timeout \*/  IndicateError();  } |

Рисунок 2.10 – установка тактовой частоты микроконтроллера с выходом в рабочий режим от источника HIS

|  |
| --- |
| /\* 2. CPU\_CLK = HSI/2 clock \*/  /\* Enable HSI clock source \*/  RST\_CLK\_HSIcmd(ENABLE);  /\* Disable CPU\_PLL \*/  RST\_CLK\_CPU\_PLLcmd(DISABLE);  /\* Select HSI/2 clock as CPU\_PLL input clock source \*/  RST\_CLK\_CPU\_PLLconfig(RST\_CLK\_CPU\_PLLsrcHSIdiv2, 1);  if (RST\_CLK\_HSIstatus() == SUCCESS) /\* Good HSI clock \*/  {  /\* Set CPU\_C3\_prescaler to 1 \*/  RST\_CLK\_CPUclkPrescaler(RST\_CLK\_CPUclkDIV1);  /\* Switch CPU\_C2\_SEL to CPU\_C1 clock instead of CPU\_PLL output \*/  RST\_CLK\_CPU\_PLLuse(DISABLE);  /\* Select CPU\_C3 clock on the CPU clock MUX \*/  RST\_CLK\_CPUclkSelection(RST\_CLK\_CPUclkCPU\_C3);  /\* LED1 blinking with HSI/2 clock as input clock source \*/  BlinkLED1(BLINK\_NUM, BLINK\_DELAY);  }  else /\* HSI timeout \*/  {  IndicateError();  } |

Рисунок 2.11 – Установка тактирования процессора с частотой в 2 раза меньше от HSI с помощью предделителя

|  |
| --- |
| /\* 3. CPU\_CLK = 7\*HSE/2 clock \*/  /\* Включение HSE осциллятора (внешнего кварцевого резонатора) \*/  RST\_CLK\_HSEconfig(RST\_CLK\_HSE\_ON);  if (RST\_CLK\_HSEstatus() == SUCCESS) /\* Если HSE включился \*/  {  /\* Выбор HSE осциллятора в качестве источника тактирования импульсов для CPU\_PLL \*/  /\* Установка умножения тактовой частоты CPU\_PLL равного 7 \*/  /\* RST\_CLK\_CPU\_PLLconfig( Источник тактирования PLL, Коэффициент умножения=9+1 \*/  /\* Коэффициент умножения=0 соответствует 1 \*/  RST\_CLK\_CPU\_PLLconfig(RST\_CLK\_CPU\_PLLsrcHSEdiv1, 7);  /\* Включение схемы PLL \*/  RST\_CLK\_CPU\_PLLcmd(ENABLE);  if (RST\_CLK\_HSEstatus() == SUCCESS) /\* Если включение CPU\_PLL прошло успешно \*/  {  /\* Установить CPU\_C3\_prescaler = 2 \*/  RST\_CLK\_CPUclkPrescaler(RST\_CLK\_CPUclkDIV2);  /\* Уытановить CPU\_C2\_SEL от CPU\_PLL выхода вместо CPU\_C1 такта \*/  RST\_CLK\_CPU\_PLLuse(ENABLE);  /\* Выбор CPU\_C3 ткта на мультиплексоре тактовых импульсов микропроцессора CPU clock MUX \*/  RST\_CLK\_CPUclkSelection(RST\_CLK\_CPUclkCPU\_C3);  /\* Мигание светодиодом \*/  BlinkLED1(BLINK\_NUM, BLINK\_DELAY);  }  else /\* CPU\_PLL не включается \*/  {  IndicateError();  }  }  else /\* HSE не включается \*/  {  IndicateError();  } |

Рисунок 2.12 – Установка тактирования процессора с частотой F\*7/2 от HSE при помощи предделителя и умножителя

|  |
| --- |
| /\* 4. CPU\_CLK = LSI clock \*/  /\* Enable LSI clock source \*/  RST\_CLK\_LSIcmd(ENABLE);  if (RST\_CLK\_LSIstatus() == SUCCESS) /\* Good LSI clock \*/  {  /\* Select LSI clock on the CPU clock MUX \*/  RST\_CLK\_CPUclkSelection(RST\_CLK\_CPUclkLSI);  /\* LED1 blinking with LSI clock as input clock source \*/  BlinkLED1(BLINK\_NUM, BLINK\_DELAY);  }  else /\* LSI timeout \*/  {  IndicateError();  } |

Рисунок 2.13 – Установка тактирование процессора от LSI

2.3 Выполнение задания по установке частоты 32 МГц

Идея состоит в том, чтобы измерить время мигания светодиодом микроконтроллера в двух случаях: установить максимальную частоту работы процессора (80 МГц), установить частоту процессора 32 МГц.

В качестве источника тактирования выбираем внешний источник HSE с частотой 8МГц. Необходимо произвести тактирование от HSE и умножить частоту с помощью PLL до нужного значения (8\*10 и 8\*4), делитель оставим без изменения (коэффициент деления единица), включить тактирование процессора от установленного сигнала.

Ниже предоставляем схему прохождения импульсов. (Рисунок 2.14)

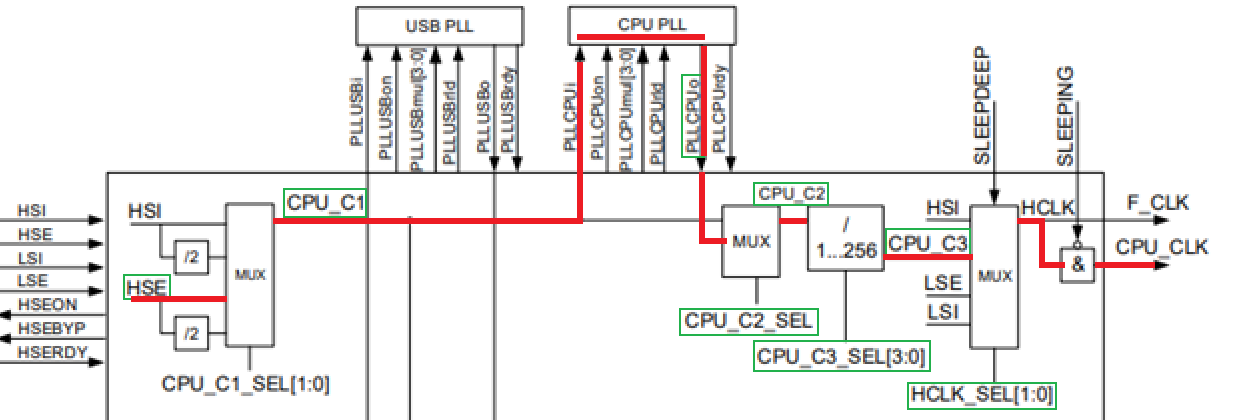


Рисунок 2.14 – Схема прохождения импульсов

Настройка тактирования производится с помощью библиотеки MDR32F9Qx\_rst\_clk.

**RST\_CLK\_PCLKcmd**. С ее помощью мы можем подать сигнал тактирования на любой блок периферии.

У функции есть два параметра:

* **RST\_CLK\_PCLK** — имя блока периферии, на который нужно подать или с которого нужно снять тактовый сигнал (сигнал тактирования).
* **New\_state** – Может быть ENABLE или DISABLE.

Возможные значения параметра RST\_CLK\_PCLK:

#define RST\_CLK\_PCLK\_CAN1 PCLK\_BIT(MDR\_CAN1\_BASE)

#define RST\_CLK\_PCLK\_CAN2 PCLK\_BIT(MDR\_CAN2\_BASE)

#define RST\_CLK\_PCLK\_USB PCLK\_BIT(MDR\_USB\_BASE)

#define RST\_CLK\_PCLK\_EEPROM PCLK\_BIT(MDR\_EEPROM\_BASE)

#define RST\_CLK\_PCLK\_RST\_CLK PCLK\_BIT(MDR\_RST\_CLK\_BASE)

#define RST\_CLK\_PCLK\_DMA PCLK\_BIT(MDR\_DMA\_BASE)

#define RST\_CLK\_PCLK\_UART1 PCLK\_BIT(MDR\_UART1\_BASE)

#define RST\_CLK\_PCLK\_UART2 PCLK\_BIT(MDR\_UART2\_BASE)

#define RST\_CLK\_PCLK\_SSP1 PCLK\_BIT(MDR\_SSP1\_BASE)

#define RST\_CLK\_PCLK\_09 PCLK\_BIT(0x40048000)

#define RST\_CLK\_PCLK\_I2C PCLK\_BIT(MDR\_I2C\_BASE)

#define RST\_CLK\_PCLK\_POWER PCLK\_BIT(MDR\_POWER\_BASE)

#define RST\_CLK\_PCLK\_WWDG PCLK\_BIT(MDR\_WWDG\_BASE)

#define RST\_CLK\_PCLK\_IWDG PCLK\_BIT(MDR\_IWDG\_BASE)

#define RST\_CLK\_PCLK\_TIMER1 PCLK\_BIT(MDR\_TIMER1\_BASE)

#define RST\_CLK\_PCLK\_TIMER2 PCLK\_BIT(MDR\_TIMER2\_BASE)

#define RST\_CLK\_PCLK\_TIMER3 PCLK\_BIT(MDR\_TIMER3\_BASE)

#define RST\_CLK\_PCLK\_ADC PCLK\_BIT(MDR\_ADC\_BASE)

#define RST\_CLK\_PCLK\_DAC PCLK\_BIT(MDR\_DAC\_BASE)

#define RST\_CLK\_PCLK\_COMP PCLK\_BIT(MDR\_COMP\_BASE)

#define RST\_CLK\_PCLK\_SSP2 PCLK\_BIT(MDR\_SSP2\_BASE)

#define RST\_CLK\_PCLK\_PORTA PCLK\_BIT(MDR\_PORTA\_BASE)

#define RST\_CLK\_PCLK\_PORTB PCLK\_BIT(MDR\_PORTB\_BASE)

#define RST\_CLK\_PCLK\_PORTC PCLK\_BIT(MDR\_PORTC\_BASE)

#define RST\_CLK\_PCLK\_PORTD PCLK\_BIT(MDR\_PORTD\_BASE)

#define RST\_CLK\_PCLK\_PORTE PCLK\_BIT(MDR\_PORTE\_BASE)

#define RST\_CLK\_PCLK\_26 PCLK\_BIT(0x400D0000)

#define RST\_CLK\_PCLK\_BKP PCLK\_BIT(MDR\_BKP\_BASE)

#define RST\_CLK\_PCLK\_28 PCLK\_BIT(0x400E0000)

#define RST\_CLK\_PCLK\_PORTF PCLK\_BIT(MDR\_PORTF\_BASE)

#define RST\_CLK\_PCLK\_EBC PCLK\_BIT(MDR\_EBC\_BASE)

#define RST\_CLK\_PCLK\_31 PCLK\_BIT(0x400F8000)

Вспомним, что наш светодиод подключен к PC2. Не сложно догадаться, что в нашем случае функция включения тактирования порта С будет выглядеть так:

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTC, ENABLE); // Включаем тактирование порта C.

Для начала включаем тактирование RST\_CLK:

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_RST\_CLK, ENABLE); // Включаем тактирование порта RST\_CLK

Далее нам нужно включить генератор HSE и обязательно дождаться его запуска. Генератор «разгоняется» постепенно, на это уйдет некоторое время:

// Включаем генератор на внешнем кварце

RST\_CLK\_HSEconfig (RST\_CLK\_HSE\_ON);

while (RST\_CLK\_HSEstatus () != SUCCESS);

Сначала мы берем исходную частоту HSE, равную 8 МГц, и, минуя делитель на 2, подаем ее на вход мультиплексора, на выходе которого получаем частоту CPU\_C1, равную тем же 8 МГц. Частота CPU\_C1 подается на вход CPU PLL (умножителя частоты). Здесь мы умножаем частоту на 10 и получаем 80 МГц. Все это записывается в программе одной строчкой – вызовом специальной функции для настройки CPU PLL:

// Настраиваем источник и коэффициент умножения PLL

// (CPU\_C1\_SEL = HSE / 1 \* 10 = 8 МГц / 1 \* 10 = 80 МГц)

// Первый параметр позволяет выбрать источник тактирования CPU PLL: HSE/1 (как у нас), HSE/2, HSI/1 или HSI/2

// Второй параметр задает коэффициент умножения частоты из предопределенного ряда: 1, 2, …16

RST\_CLK\_CPU\_PLLconfig (RST\_CLK\_CPU\_PLLsrcHSEdiv1,

RST\_CLK\_CPU\_PLLmul10);

// Включаем PLL, но еще не подключаем к кристаллу

RST\_CLK\_CPU\_PLLcmd (ENABLE);

while (RST\_CLK\_CPU\_PLLstatus () != SUCCESS);

На выходе CPU PLL имеем частоту CPU\_C2 = 80 МГц. Эту частоту подаем на вход мультиплексора, а с его выхода – на делитель. Мы берем коэффициент деления равный 1, т.е. по сути не делим. В результате получаем частоту CPU\_C3 = 80 МГц:

// можно поделить на любой коэффициент из ряда: 1, 2, 4, 8, 16, 32, 64, 128 или 256

// Делитель С3 ( CPU\_C3\_SEL = CPU\_C2\_SEL )

RST\_CLK\_CPUclkPrescaler (RST\_CLK\_CPUclkDIV1);

// На С2 идет с PLL, а не напрямую с С1 (CPU\_C2\_SEL = PLL)

RST\_CLK\_CPU\_PLLuse (ENABLE);

Теперь частота CPU\_C3 = 80 МГц через очередной мультиплексор попадет на выход схемы тактирования и пойдет к ядру микроконтроллера под именем CPU\_CLK, а также к периферийным устройствам под именем HCLK или FCLK:

// CPU берет тактирование с выхода С3

// (HCLK\_SEL = CPU\_C3\_SEL = 80 МГц)

RST\_CLK\_CPUclkSelection (RST\_CLK\_CPUclkCPU\_C3);

Именно с этого момента ядро микроконтроллера будет работать на 80 МГц, поскольку CPU\_C3 заменит собой HSI, подведенное к мультиплексору, которое до сих пор использовалось по умолчанию.

Ниже приведен код настройки тактовой частоты. (Рисунок 2.15)

|  |
| --- |
| //Функция настройки тактовой частоты МК  void clk\_CoreConfig(void) {  //Реинициализация настроек тактирования  // Включить тактирование батарейного блока  //и внутренние генераторы, все остальное сбросить  RST\_CLK\_DeInit();  //Включение тактирования от внешнего источника HSE (High Speed External)  RST\_CLK\_HSEconfig(RST\_CLK\_HSE\_ON);  //Проверка статуса HSE  //if (RST\_CLK\_HSEstatus() == SUCCESS) /\* Если HSE осциллятор включился  //if (RST\_CLK\_HSEstatus() == ERROR) while (1);  while (RST\_CLK\_HSEstatus () != SUCCESS);  //Настройка делителя/умножителя частоты CPU\_PLL(фазовая подстройка частоты)  /\* Указываем PLL от куда брать частоту (RCC\_PLLSource\_HSE\_Div1) и на сколько ее умножать (RCC\_PLLMul\_9) \*/  /\* PLL может брать частоту с кварца как есть (RCC\_PLLSource\_HSE\_Div1) или поделенную на 2 (RCC\_PLLSource\_HSE\_Div2). Смотри схему \*/  RST\_CLK\_CPU\_PLLconfig(RST\_CLK\_CPU\_PLLsrcHSEdiv1, RST\_CLK\_CPU\_PLLmul10);  // RST\_CLK\_CPU\_PLLconfig(div, mul);  //Включение CPU\_PLL  //, но еще не подключать к кристаллу  RST\_CLK\_CPU\_PLLcmd(ENABLE);  //Проверка статуса CPU\_PLL  //if (RST\_CLK\_CPU\_PLLstatus() == SUCCESS) //Если включение CPU\_PLL прошло успешно  //if (RST\_CLK\_CPU\_PLLstatus() == ERROR) while (1);  while (RST\_CLK\_CPU\_PLLstatus() != SUCCESS);    /\* Установка CPU\_C3\_prescaler = 2 \*/  // Делитель CPU\_C3\_SEL ( CPU\_C3\_SEL = CPU\_C2\_SEL/2 )  RST\_CLK\_CPUclkPrescaler(RST\_CLK\_CPUclkDIV2);  //Коммутация выхода CPU\_PLL на вход CPU\_C3  //На С2 идет с PLL, а не напрямую с С1 (CPU\_C2\_SEL = PLL)  RST\_CLK\_CPU\_PLLuse(ENABLE);  //Выбор источника тактирования ядра процессора  //CPU берет с выхода С3 (а может с выхода HSI,LSI,LSE) (HCLK\_SEL = CPU\_C3\_SEL )  RST\_CLK\_CPUclkSelection(RST\_CLK\_CPUclkCPU\_C3);  //Тактирование перифирии  //Подача тактовой частоты на PORTC, PORTD  RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTC, ENABLE);  RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTD, ENABLE);  } |

Рисунок 2.15 – Код настройки тактовой частоты

Бал написан код для тестирования, сначала он запускается на частоте тактирования 80 МГц и несколько раз мигает светодиодом с периодом 1 сек, а потом меняет частоту тактирования на 32 МГц и снова должен мигает с установленным периодом 1 сек. (Рисунок 2.16)

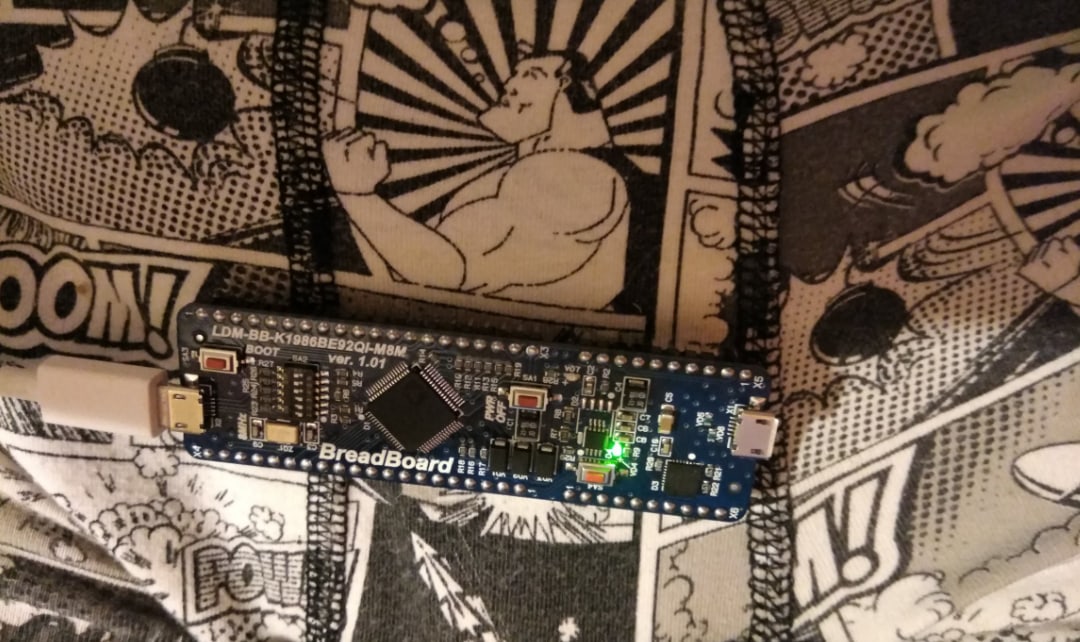


Рисунок 2.16 – Пример работы программы

Измеряем время между миганием светодиода при 80 МГц, а потом при 32 МГц и измеряем тот же интервал времени ожидания между миганием светодиода. Интервал ожидания увеличился с 1 до 2,4, с 10 до 24, с 30 до 74 секунд. (Рисунок 2.17)

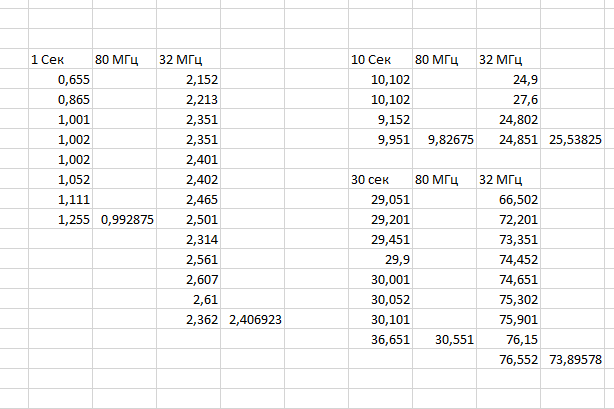


Рисунок 2.17 – Результаты эксперимента

По формуле 1 вычисляем частоту и она оказывается равное примерно 32 МГц. (Рисунок 2.18)

*(1)*

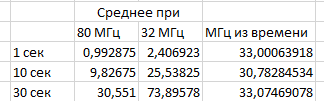


Рисунок 2.18 – Вычисленное значение частоты из усредненного времени из всех опытов

Код программы приведен в приложении А и находится на GitHub по ссылке «https://github.com/OzziOsbarn/MSTN» в папке «Lab1».

3 Заключение

В ходе работы были изучены модуль сброса и тактовых частота по технической документации на МК 1986ВЕ9х. А также запущен и проанализирован код примера, выполнено изменение значения тактовой частоты на 32 МГц и экспериментально измерено.

Был написан отчёт согласно требованиям ОС ТУСУР 01-2013.

Приложение А

(обязательное)

Код программы

/\*+--- <INCLUDES> ---------------------------------\*/

#include "main.h"

#include <stdlib.h>

#include <stdio.h>

//#include "stdint.h"

#include <inttypes.h>

#include "MDR32F9Qx\_board.h"

#include "MDR32F9Qx\_config.h"

#include "MDR32Fx.h"

#include "MDR32F9Qx\_port.h"

#include "MDR32F9Qx\_rst\_clk.h"

#include "mstn\_led.h"

#include "mstn\_clk.h"

#include "mstn\_usb.h"

/\*+==================================================\*/

/\*+--- <DEFINES> ---------------------------------\*/

#define LED1 PORT\_Pin\_0

/\*+==================================================\*/

/\*+--- <FUNCTIONS DECLARATION> ---------------------\*/

void BlinkLED(uint32\_t num, uint32\_t del);

void IndicateError(void);

//Функция инициализации кнопки SA4

void button\_Init(void);

//Функция считывания текущего состояния кнопки SA4

uint8\_t button\_State(void);

//Функция инициализации светодиода VD7

void led\_Init(void);

//Функция записи состояния (1:0) светодиода VD7

void led\_Write(bool on\_off);

void clk\_CoreConfig80(void);

void clk\_CoreConfig32(void);

//int main(int argc, char \*argv[])

int main()

{

printf("Usb cycle\n");

while(USB\_GetStatus() != PERMITTED);

printf("USB ready\n");

while (1){

for(int i=0;i<100;i++){

printf("%d",i);

printf(" ");

}

uint8\_t state = 0;

uint32\_t waitTime = 5;

clk\_CoreConfig80();

printf("Start Init 80\n");

led\_Init();

button\_Init();

printf("Buffer save\n");

BlinkLED(4,10000);

printf("end\n");

//printf("Input waitTime (ms):\n");

//scanf("%lu", &waitTime);

printf("Start Init 32\n");

clk\_CoreConfig32();

printf("Init 32\n");

led\_Init();

button\_Init();

printf("Buffer save\n");

BlinkLED(4,10000);

printf("end\n");

}

return EXIT\_SUCCESS;

}

/\*+--- <FUNCTIONS DESCRIPTION> ---------------------\*/

//

void BlinkLED(uint32\_t num, uint32\_t del){

uint32\_t cnt;

for ( cnt = 0; cnt < num; cnt++)

{

Delay(del);

led\_Write(1);

Delay(del);

led\_Write(0);

}

}

//

void IndicateError(void){

/\*<<<>>> Switch LED3 on and off in case of error \*/

BlinkLED(3,5000);

}

//Функция инициализации кнопки SA4

void button\_Init(void){

//Создание структуры для инициализации порта

PORT\_InitTypeDef PORT\_InitStructure;

//Настройки порта: ввод, функция ввода/вывода, цифровой режим,

//минимальная скорость, Pin5

PORT\_InitStructure.PORT\_OE = PORT\_OE\_IN;

PORT\_InitStructure.PORT\_FUNC = PORT\_FUNC\_PORT;

PORT\_InitStructure.PORT\_MODE = PORT\_MODE\_DIGITAL;

PORT\_InitStructure.PORT\_SPEED = PORT\_SPEED\_SLOW;

PORT\_InitStructure.PORT\_Pin = (PORT\_Pin\_5);

PORT\_Init(MDR\_PORTD, &PORT\_InitStructure);

}

//Функция считывания текущего состояния кнопки SA4

uint8\_t button\_State(void){

return PORT\_ReadInputDataBit(MDR\_PORTD, PORT\_Pin\_5);

}

//Функция инициализации светодиода VD7

void led\_Init(void){

//Создание структуры для инициализации порта

PORT\_InitTypeDef PORT\_InitStructure;

//Настройки порта: вывод, функция ввода/вывода, цифровой режим,

//максимальная скорость, Pin2

PORT\_InitStructure.PORT\_OE = PORT\_OE\_OUT;

PORT\_InitStructure.PORT\_FUNC = PORT\_FUNC\_PORT;

PORT\_InitStructure.PORT\_MODE = PORT\_MODE\_DIGITAL;

PORT\_InitStructure.PORT\_SPEED = PORT\_SPEED\_MAXFAST;

PORT\_InitStructure.PORT\_Pin = (PORT\_Pin\_2);

PORT\_Init(MDR\_PORTC, &PORT\_InitStructure);

}

//Функция записи состояния (1:0) светодиода VD7

void led\_Write(bool on\_off){

PORT\_WriteBit(MDR\_PORTC, PORT\_Pin\_2, on\_off ? Bit\_SET : Bit\_RESET);

}

//Функция настройки тактовой частоты МК

void clk\_CoreConfig80(void) {

//Реинициализация настроек тактирования

// Включить тактирование батарейного блока

//и внутренние генераторы, все остальное сбросить

RST\_CLK\_DeInit();

//Включение тактирования от внешнего источника HSE (High Speed External)

RST\_CLK\_HSEconfig(RST\_CLK\_HSE\_ON);

//Проверка статуса HSE

//if (RST\_CLK\_HSEstatus() == SUCCESS) /\* Если HSE осциллятор включился

//if (RST\_CLK\_HSEstatus() == ERROR) while (1);

while (RST\_CLK\_HSEstatus () != SUCCESS);

//Настройка делителя/умножителя частоты CPU\_PLL(фазовая подстройка частоты)

/\* Указываем PLL от куда брать частоту (RCC\_PLLSource\_HSE\_Div1) и на сколько ее умножать (RCC\_PLLMul\_9) \*/

/\* PLL может брать частоту с кварца как есть (RCC\_PLLSource\_HSE\_Div1) или поделенную на 2 (RCC\_PLLSource\_HSE\_Div2). Смотри схему \*/

RST\_CLK\_CPU\_PLLconfig(RST\_CLK\_CPU\_PLLsrcHSEdiv1, RST\_CLK\_CPU\_PLLmul10);

// RST\_CLK\_CPU\_PLLconfig(div, mul);

//Включение CPU\_PLL

//, но еще не подключать к кристаллу

RST\_CLK\_CPU\_PLLcmd(ENABLE);

//Проверка статуса CPU\_PLL

//if (RST\_CLK\_CPU\_PLLstatus() == SUCCESS) //Если включение CPU\_PLL прошло успешно

//if (RST\_CLK\_CPU\_PLLstatus() == ERROR) while (1);

while (RST\_CLK\_CPU\_PLLstatus() != SUCCESS);

/\* Установка CPU\_C3\_prescaler = 2 \*/

// Делитель CPU\_C3\_SEL ( CPU\_C3\_SEL = CPU\_C2\_SEL/2 )

RST\_CLK\_CPUclkPrescaler(RST\_CLK\_CPUclkDIV1);

//Коммутация выхода CPU\_PLL на вход CPU\_C3

//На С2 идет с PLL, а не напрямую с С1 (CPU\_C2\_SEL = PLL)

RST\_CLK\_CPU\_PLLuse(ENABLE);

//Выбор источника тактирования ядра процессора

//CPU берет с выхода С3 (а может с выхода HSI,LSI,LSE) (HCLK\_SEL = CPU\_C3\_SEL )

RST\_CLK\_CPUclkSelection(RST\_CLK\_CPUclkCPU\_C3);

//Тактирование перифирии

//Подача тактовой частоты на PORTC, PORTD

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTC, ENABLE);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTD, ENABLE);

}

void clk\_CoreConfig32(void) {

//Реинициализация настроек тактирования

// Включить тактирование батарейного блока

//и внутренние генераторы, все остальное сбросить

RST\_CLK\_DeInit();

//Включение тактирования от внешнего источника HSE (High Speed External)

RST\_CLK\_HSEconfig(RST\_CLK\_HSE\_ON);

//Проверка статуса HSE

//if (RST\_CLK\_HSEstatus() == SUCCESS) /\* Если HSE осциллятор включился

//if (RST\_CLK\_HSEstatus() == ERROR) while (1);

while (RST\_CLK\_HSEstatus () != SUCCESS);

//Настройка делителя/умножителя частоты CPU\_PLL(фазовая подстройка частоты)

/\* Указываем PLL от куда брать частоту (RCC\_PLLSource\_HSE\_Div1) и на сколько ее умножать (RCC\_PLLMul\_9) \*/

/\* PLL может брать частоту с кварца как есть (RCC\_PLLSource\_HSE\_Div1) или поделенную на 2 (RCC\_PLLSource\_HSE\_Div2). Смотри схему \*/

RST\_CLK\_CPU\_PLLconfig(RST\_CLK\_CPU\_PLLsrcHSEdiv1, RST\_CLK\_CPU\_PLLmul4);

// RST\_CLK\_CPU\_PLLconfig(div, mul);

//Включение CPU\_PLL

//, но еще не подключать к кристаллу

RST\_CLK\_CPU\_PLLcmd(ENABLE);

//Проверка статуса CPU\_PLL

//if (RST\_CLK\_CPU\_PLLstatus() == SUCCESS) //Если включение CPU\_PLL прошло успешно

//if (RST\_CLK\_CPU\_PLLstatus() == ERROR) while (1);

while (RST\_CLK\_CPU\_PLLstatus() != SUCCESS);

/\* Установка CPU\_C3\_prescaler = 2 \*/

// Делитель CPU\_C3\_SEL ( CPU\_C3\_SEL = CPU\_C2\_SEL/2 )

RST\_CLK\_CPUclkPrescaler(RST\_CLK\_CPUclkDIV1);

//Коммутация выхода CPU\_PLL на вход CPU\_C3

//На С2 идет с PLL, а не напрямую с С1 (CPU\_C2\_SEL = PLL)

RST\_CLK\_CPU\_PLLuse(ENABLE);

//Выбор источника тактирования ядра процессора

//CPU берет с выхода С3 (а может с выхода HSI,LSI,LSE) (HCLK\_SEL = CPU\_C3\_SEL )

RST\_CLK\_CPUclkSelection(RST\_CLK\_CPUclkCPU\_C3);

//Тактирование перифирии

//Подача тактовой частоты на PORTC, PORTD

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTC, ENABLE);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTD, ENABLE);

}