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)

Томск 2020

1 Введение

1.1 Техническое задание

Цель лабораторной работы:

Изучить модуль сброса и тактовых частота по технической документации на МК 1986BE9x.

Задачи:

1. Изучить код примера из библиотеки SPL lib\MDR32F9_1986BE4_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 xUART, 2 xCAN, 2 xSPI, I2C. В структуре МК имеются также 12-разрядный АЦП (8 каналов) и один 12-разрядный одноканальный ЦАП. (Рисунок 1.1)

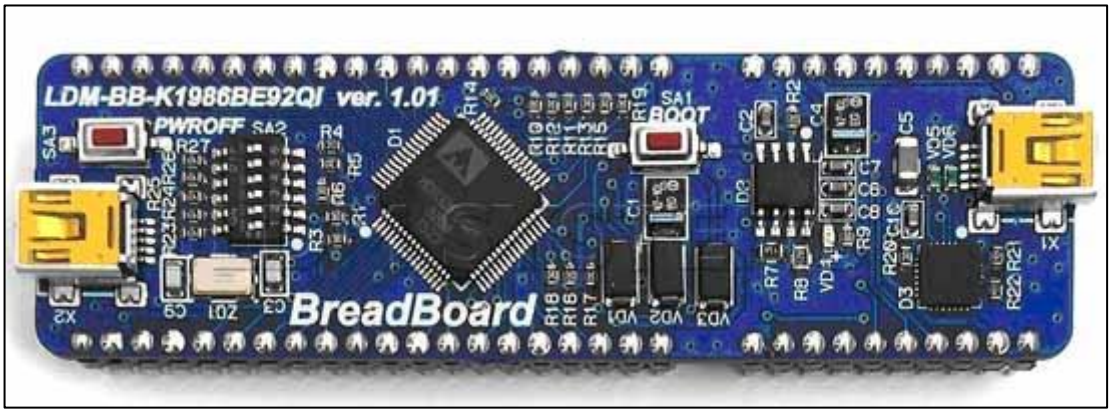


Рисунок 1.1 – Отладочная плата LDM-BB-K1986BE92QI. Вид сверху

1.3 Спецификация на МК 1986BE9x: 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))).

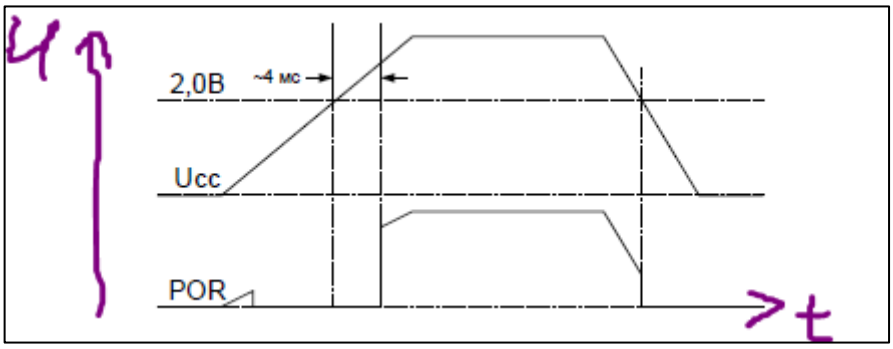


Рисунок 1.2 – Сигнал сброса при включении и выключении основного напряжения питания

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)

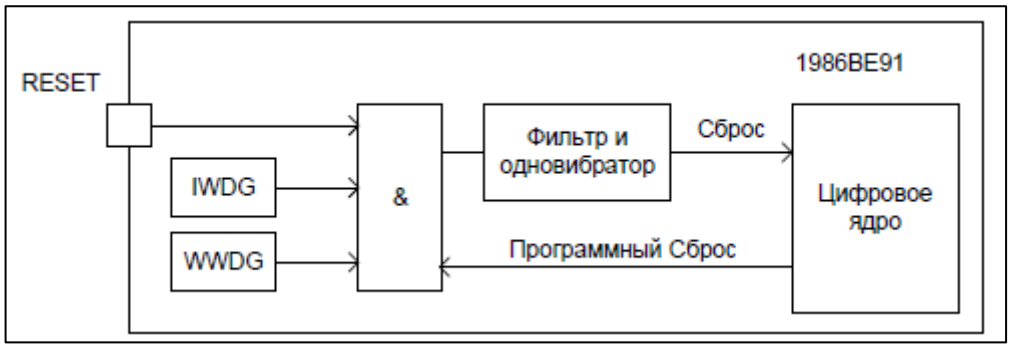


Рисунок 1.3 – Структурная блок-схема сброса

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)

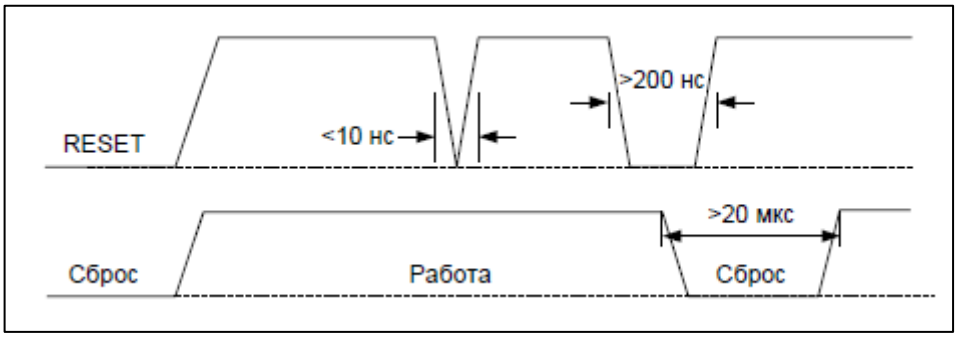


Рисунок 1.4 – Формирование сигнала сброса

1.4 Спецификация на МК 1986BE9x: 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)

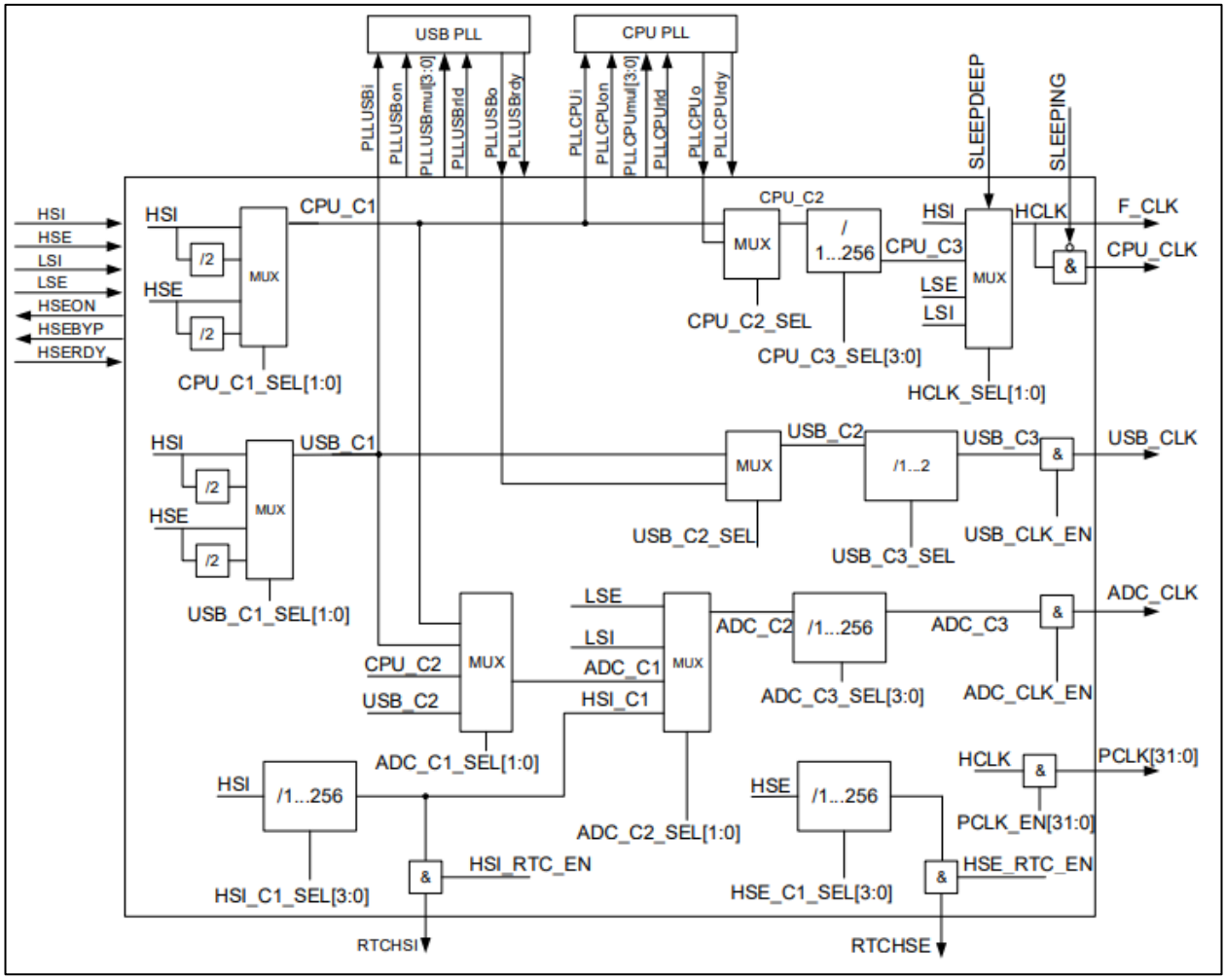


Рисунок 1.5 – Структурная блок-схема формирования тактовой частоты

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)

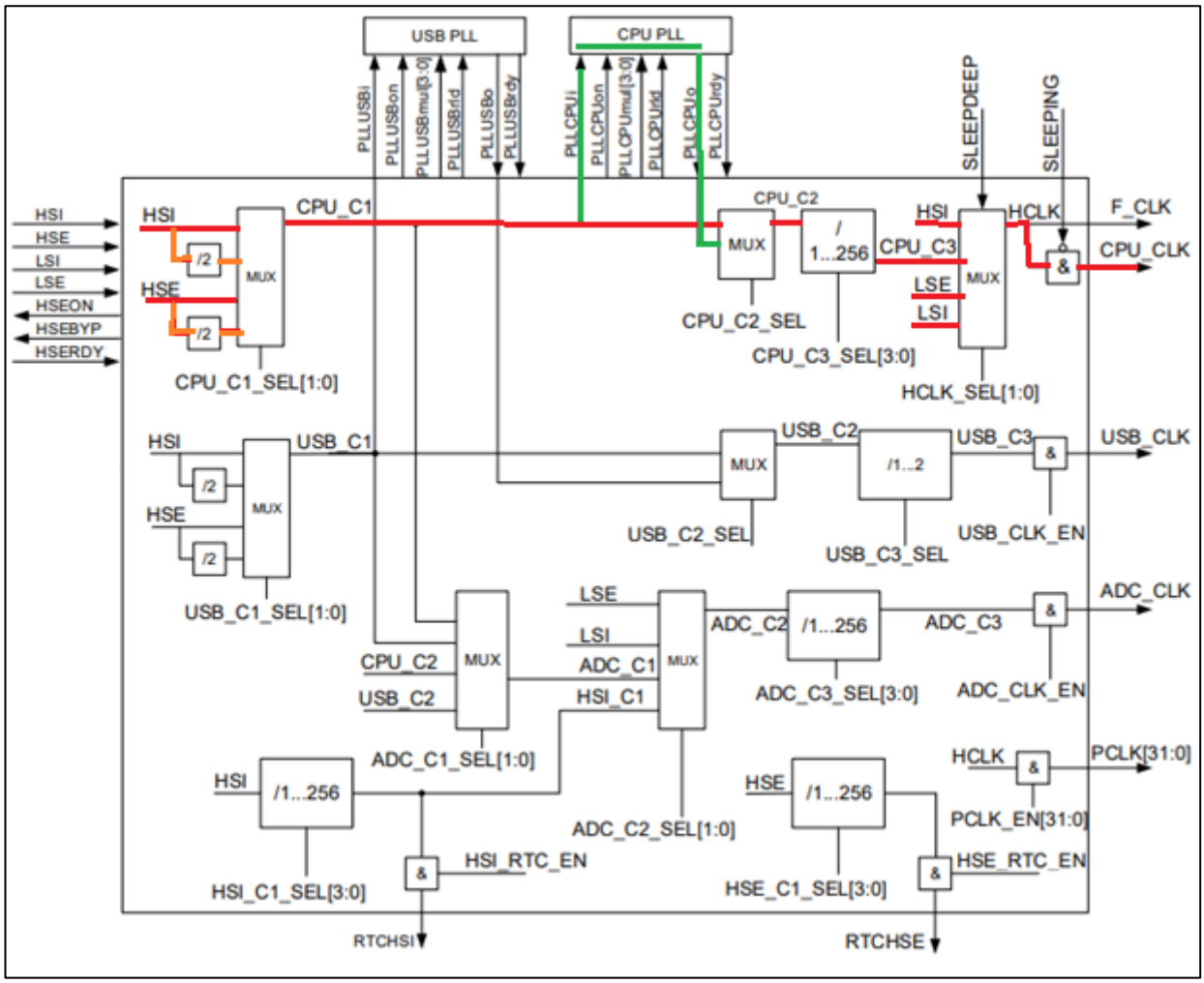


Рисунок 1.6– Линии подачи тактовой частоты на процессор

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 C1 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)

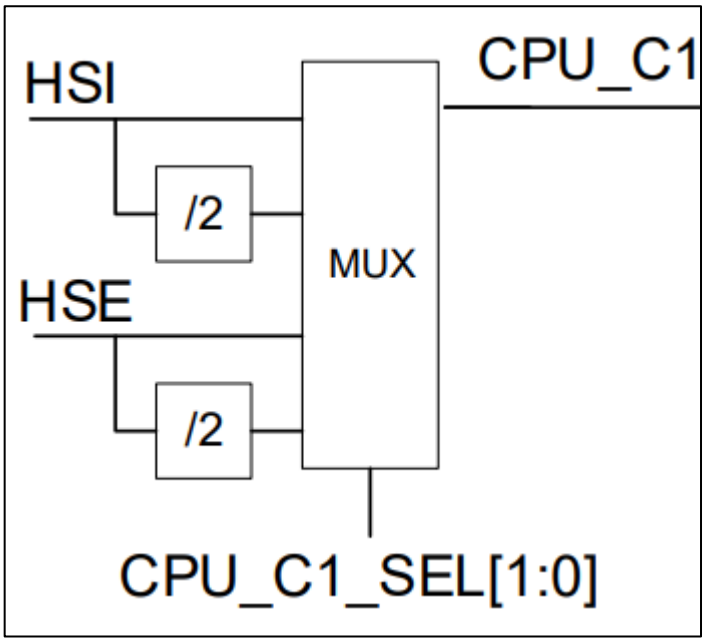


Рисунок 1.7 – Делители частоты

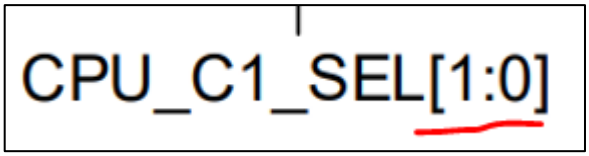


Рисунок 1.8 – Количество управляющих сигналов

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)

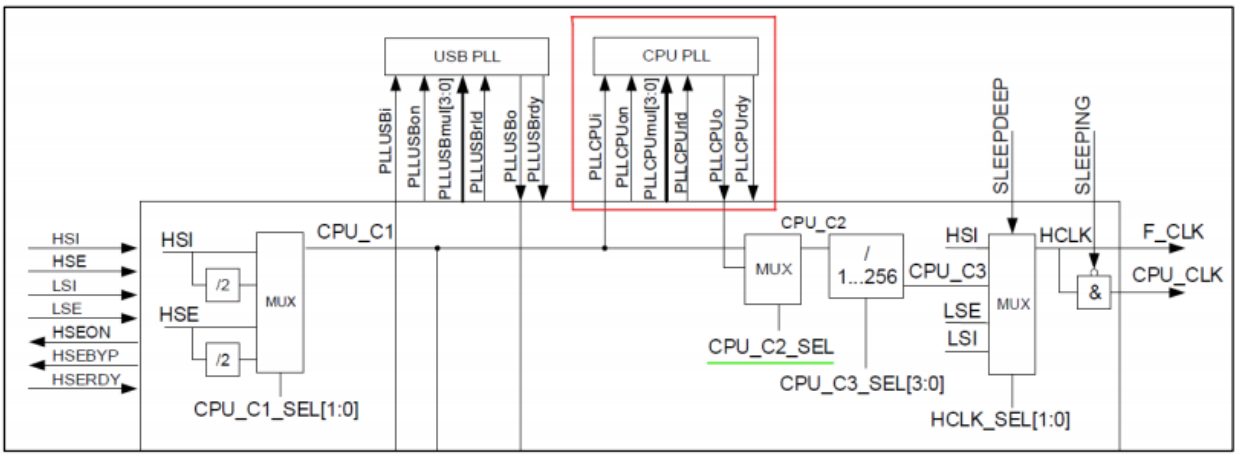


Рисунок 1.9 – Схема формирования тактовой частоты – Блок умножения частоты

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)

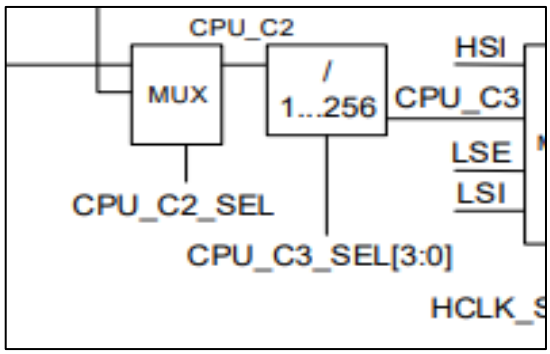


Рисунок 1.10 - Делитель

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)

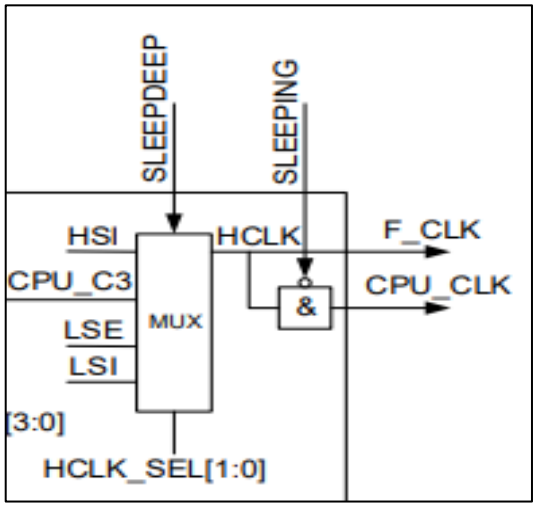


Рисунок 1.11 – Мультиплексор HCLK – Выбор сигнала тактирования процессора

За настройку тактирования отвечает блок контроллера тактовой частоты MDR_RST_CLK. (Рисунок 1.12)

Базовый Адрес	Название	Описание
0x4002_0000	MDR_RST_CLK	Контроллер тактовой частоты
Смещение		
0x00	CLOCK_STATUS	MDR_RST_CLK->CLOCK_STATUS Регистр состояния блока управления тактовой частотой
0x04	PLL_CONTROL	MDR_RST_CLK->PLL_CONTROL Регистр управления блоками умножения частоты
0x08	HS_CONTROL	MDR_RST_CLK->HS_CONTROL Регистр управления высокочастотным генератором и осциллятором
0x0C	CPU_CLOCK	MDR_RST_CLK->CPU_CLOCK Регистр управления тактовой частотой процессорного ядра
0x10	USB_CLOCK	MDR_RST_CLK->USB_CLOCK Регистр управления тактовой частотой контроллера USB
0x14	ADC_MCO_CLOCK	MDR_RST_CLK->ADC_MCO_CLOCK Регистр управления тактовой частотой АЦП
0x18	RTC_CLOCK	MDR_RST_CLK->RTC_CLOCK Регистр управления формированием высокочастотных тактовых сигналов блока RTC
0x1C	PER_CLOCK	MDR_RST_CLK->PER_CLOCK Регистр управления тактовой частотой периферийных блоков
0x20	CAN_CLOCK	MDR_RST_CLK->CAN_CLOCK Регистр управления тактовой частотой CAN
0x24	TIM_CLOCK	MDR_RST_CLK->TIM_CLOCK Регистр управления тактовой частотой TIMER
0x28	UART_CLOCK	MDR_RST_CLK->UART_CLOCK Регистр управления тактовой частотой UART
0x2C	SSP_CLOCK	MDR_RST_CLK->SSP_CLOCK Регистр управления тактовой частотой SSP

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.

Таблица 95 – Регистр PER_CLOCK

Номер	31...5	4	3...0
Доступ	R/W	R/W	R/W
Сброс	0	1	0
	PCLK_EN[31:5]	PCLK_EN[4]	PCLK_EN[3:0]

Таблица 96 – Описание бит регистра PER_CLOCK

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31…5	PCLK EN[31:5]	Биты разрешения тактирования периферийных блоков: 0 – запрещено; 1 – разрешено. PCLK[5] – DMA PCLK[6] – UART1 PCLK[7] – UART2 PCLK[8] – SPI1 PCLK[9] – зарезервировано PCLK[10] – I2C1 PCLK[11] – POWER PCLK[12] – WWDT PCLK[13] – IWDT PCLK[14] – TIMER1 PCLK[15] – TIMER2 PCLK[16] – TIMER3 PCLK[17] – ADC PCLK[18] – DAC PCLK[19] – COMP PCLK[20] – SPI2 PCLK[21] – PORTA PCLK[22] – PORTB PCLK[23] – PORTC PCLK[24] – PORTD PCLK[25] – PORTE PCLK[26] – зарезервировано PCLK[27] – BKP PCLK[28] – зарезервировано PCLK[29] – PORTF PCLK[30] – EXT_BUS_CNTRL PCLK[31] – зарезервировано
4	PCLK EN[4]	Биты разрешения тактирования периферийных блоков: 0 – запрещено; 1 – разрешено. PCLK[4] – RST_CLK. После сброса в состоянии 1
3…0	PCLK EN[3:0]	Биты разрешения тактирования периферийных блоков: 0 – запрещено; 1 – разрешено. PCLK[0] – CAN1 PCLK[1] – CAN2 PCLK[2] – USB PCLK[3] – EEPROM_CNTRL

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.

Таблица 85 – Регистра HS_CONTROL

Номер	31…2	1	0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	**HSE BYP**	**HSE ON**

Таблица 86 – Описание бит регистра HS_CONTROL

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...2	-	Зарезервировано
1	HSE BYP	Бит управления HSE осциллятором: 0 – режим осциллятора; 1 – режим внешнего генератора
0	HSE ON	Бит управления HSE осциллятором: 0 – выключен; 1 – включен

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 (с 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.

Таблица 83 – Регистр PLL_CONTROL

Номер	31...12	11…8	7…4	3	2	1	0
Доступ	U	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0
	-	**PLL CPU MUL[3:0]**	**PLL USB MUL[3:0]**	**PLL CPU RLD**	**PLL CPU ON**	**PLL USB RLD**	**PLL USB ON**

Таблица 84 – Описание бит регистра PLL_CONTROL

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…12	-	Зарезервировано
11...8	PLL CPU MUL[3:0]	Коэффициент умножения для CPU PLL: PLLCPUo = PLLCPUi x (PLLCPUMUL+1)
7...4	PLL USB MUL[3:0]	Коэффициент умножения для USB PLL: PLLUSBo = PLLUSBi x (PLLUSBMUL+1)
3	PLL CPU RLD	Бит перезапуска PLL. При смене коэффициента умножения в рабочем режиме необходимо задать равным 1
2	PLL CPU ON	Бит включения PLL: 0 – PLL выключена; 1 – PLL включена
1	PLL USB RLD	Бит перезапуска PLL. При смене коэффициента умножения в рабочем режиме необходимо задать равным 1
0	PLL USB ON	Бит включения PLL: 0 – PLL выключена; 1 – PLL включена

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)

Таблица 81 – Регистр CLOCK_STATUS

Номер	31…3	2	1	0
Доступ	U	RO	RO	RO
Сброс	0	0	0	0
	-	**HSE RDY**	**PLL CPU RDY**	**PLL USB RDY**

Таблица 82 – Описание бит регистра CLOCK_STATUS

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...3	-	Зарезервировано
2	HSE RDY	Флаг выхода в рабочий режим осциллятора HSE: 0 – осциллятор не запущен или не стабилен; 1 – осциллятор запущен и стабилен
1	PLL CPU RDY	Флаг выхода в рабочий режим CPU PLL: 0 – PLL не запущена или не стабильна; 1 – PLL запущена и стабильна
0	PLL USB RDY	Флаг выхода в рабочий режим USB PLL: 0 – PLL не запущена или не стабильна; 1 – PLL запущена и стабильна

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)

Таблица 87 – Регистр CPU_CLOCK

Номер	31…10	9…8	7…4	3	2	1…0
Доступ	U	R/W	R/W	U	R/W	R/W
Сброс	0	0	0	0	0	0
	-	HCLK SEL[1:0]	CPU C3 SEL[3:0]	-	CPU C2 SEL	CPU C1 SEL[1:0]

Таблица 88 – Описание бит регистра CPU_CLOCK

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...10	-	Зарезервировано
9…8	HCLK SEL[1:0]	Биты выбора источника для HCLK: 00 – HSI 01 – CPU_C3 10 – LSE 11 – LSI
7…4	CPU C3 SEL[3:0]	Биты выбора делителя для CPU_C3: 0xxx – CPU_C3 = CPU_C2 1000 - CPU_C3 = CPU_C2 / 2 1001 - CPU_C3 = CPU_C2 / 4 1010 - CPU_C3 = CPU_C2 / 8 … 1111 - CPU_C3 = CPU_C2 / 256
3	-	Зарезервировано
2	CPU C2 SEL	Биты выбора источника для CPU_C2: 0 – CPU_C1 1 – PLLCPUo
1…0	CPU C1 SEL[1:0]	Биты выбора источника для CPU_C1: 00 – HSI 01 – HSI/2 10 – HSE 11 – HSE/2

Рисунок 1.17 – Описание бит регистра управления тактовой частотой процессорного ядра (CLOCK_STATUS)

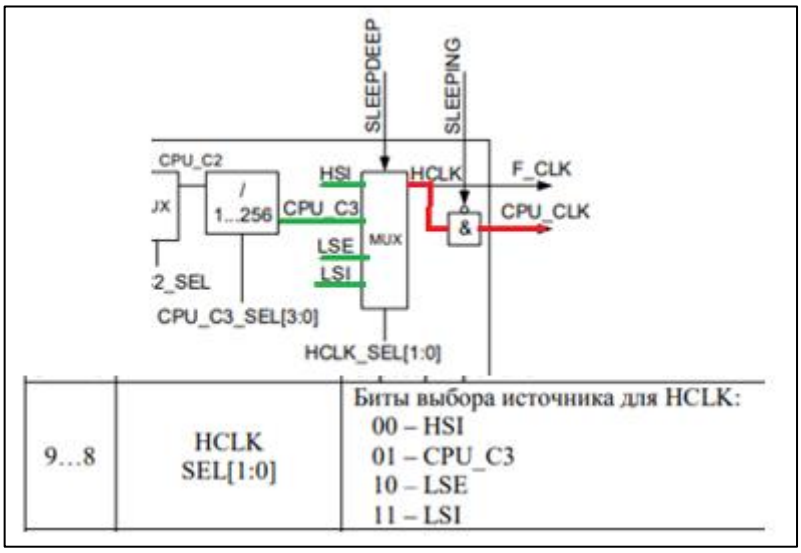
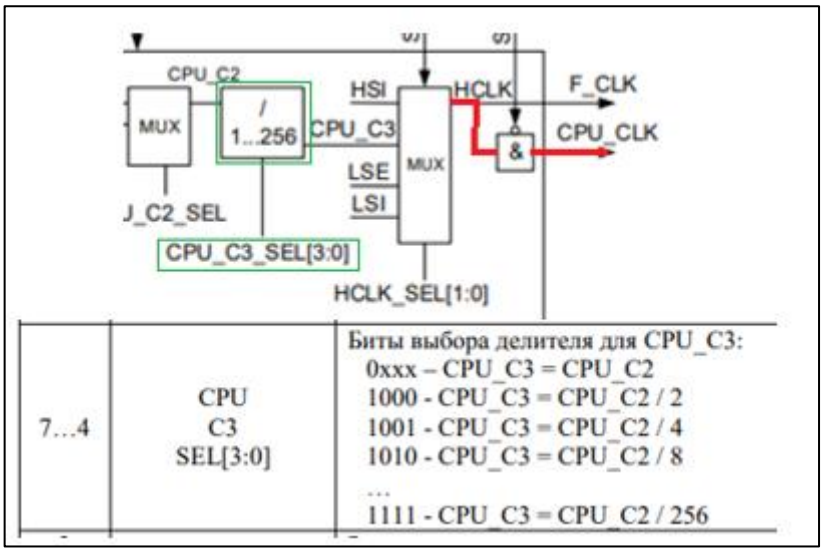
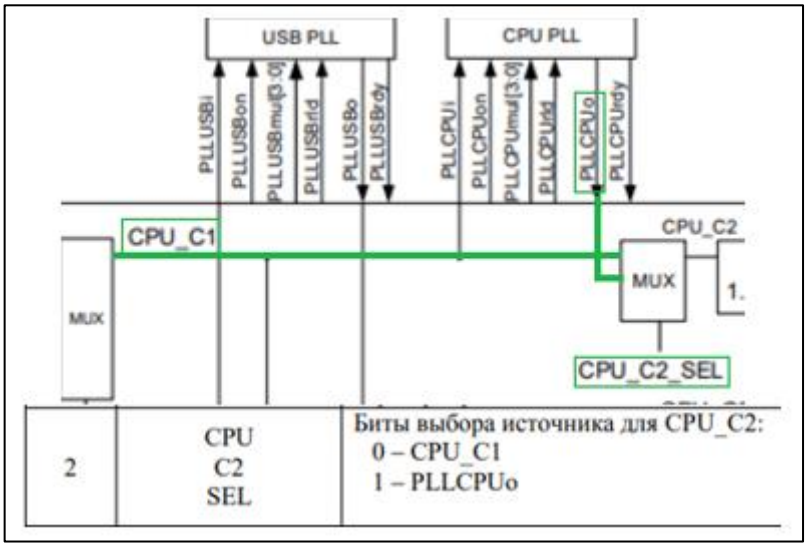


Рисунок 1.18 – Отображение на схеме линий тактирования (зеленый цвет) и связь с битами выбора источника для HCLK



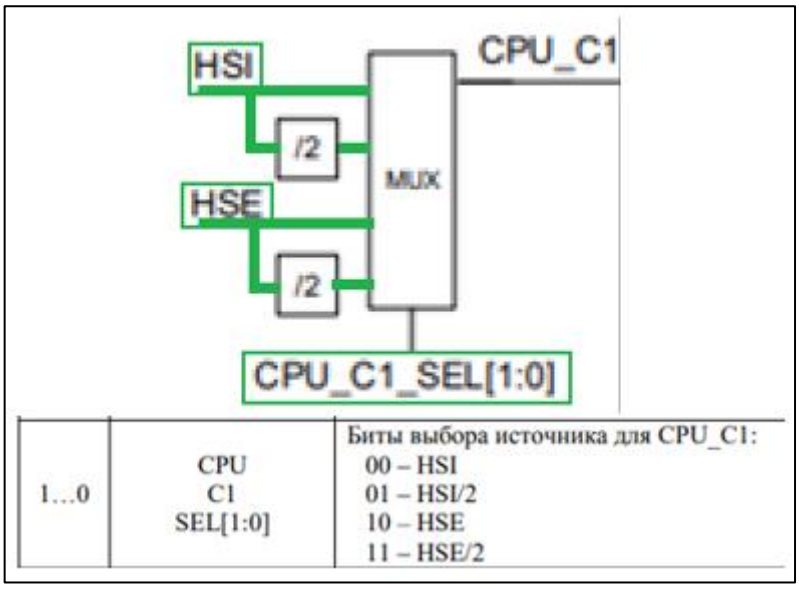
7…4	CPU C3 SEL[3:0]	Биты выбора делителя для CPU_C3: 0xxx – CPU_C3 = CPU_C2 1000 - CPU_C3 = CPU_C2 / 2 1001 - CPU_C3 = CPU_C2 / 4 1010 - CPU_C3 = CPU_C2 / 8 … 1111 - CPU_C3 = CPU_C2 / 256

Рисунок 1.19 - Отображение на схеме блока делителя частоты (зеленый цвет) и связь с битами выбора делителя для CPU_C3



2	CPU C2 SEL	Биты выбора источника для CPU_C2: 0 – CPU_C1 1 – PLLCPUo

Рисунок 1.20 - Отображение на схеме линий тактирования (зеленый цвет) и связь с битами выбора источника для CPU_C2



1…0	CPU C1 SEL[1:0]	Биты выбора источника для CPU_C1: 00 – HSI 01 – HSI/2 10 – HSE 11 – HSE/2

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)

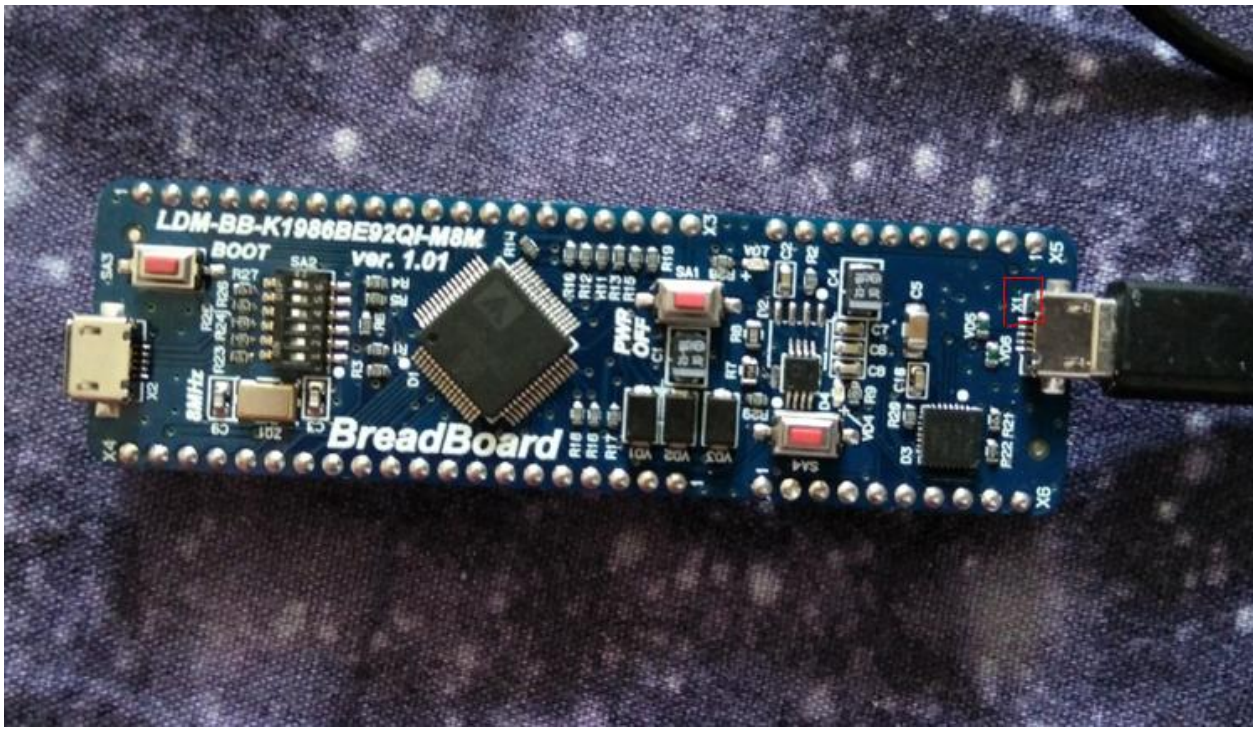


Рисунок 2.1 – Подключение платы для загрузки прошивки (X1)

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

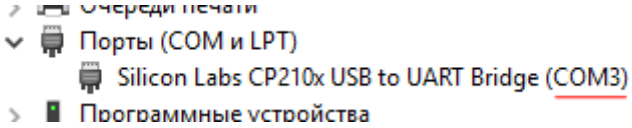


Рисунок 2.2 – Просмотр к какому порту подключено устройство

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

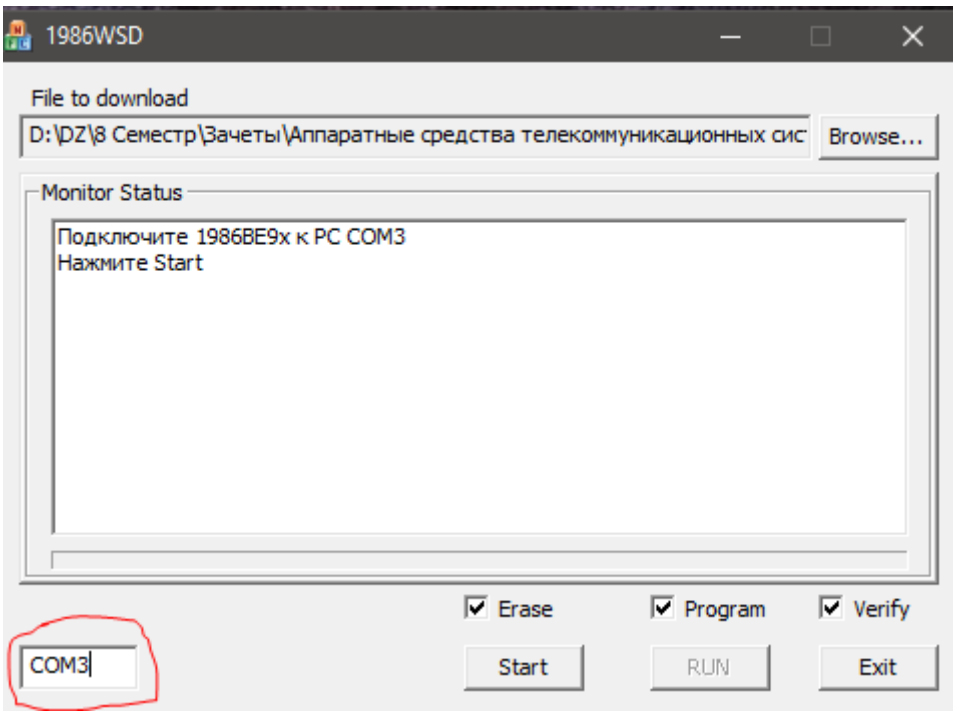


Рисунок 2.3 – Изменение ком порта в программе

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)

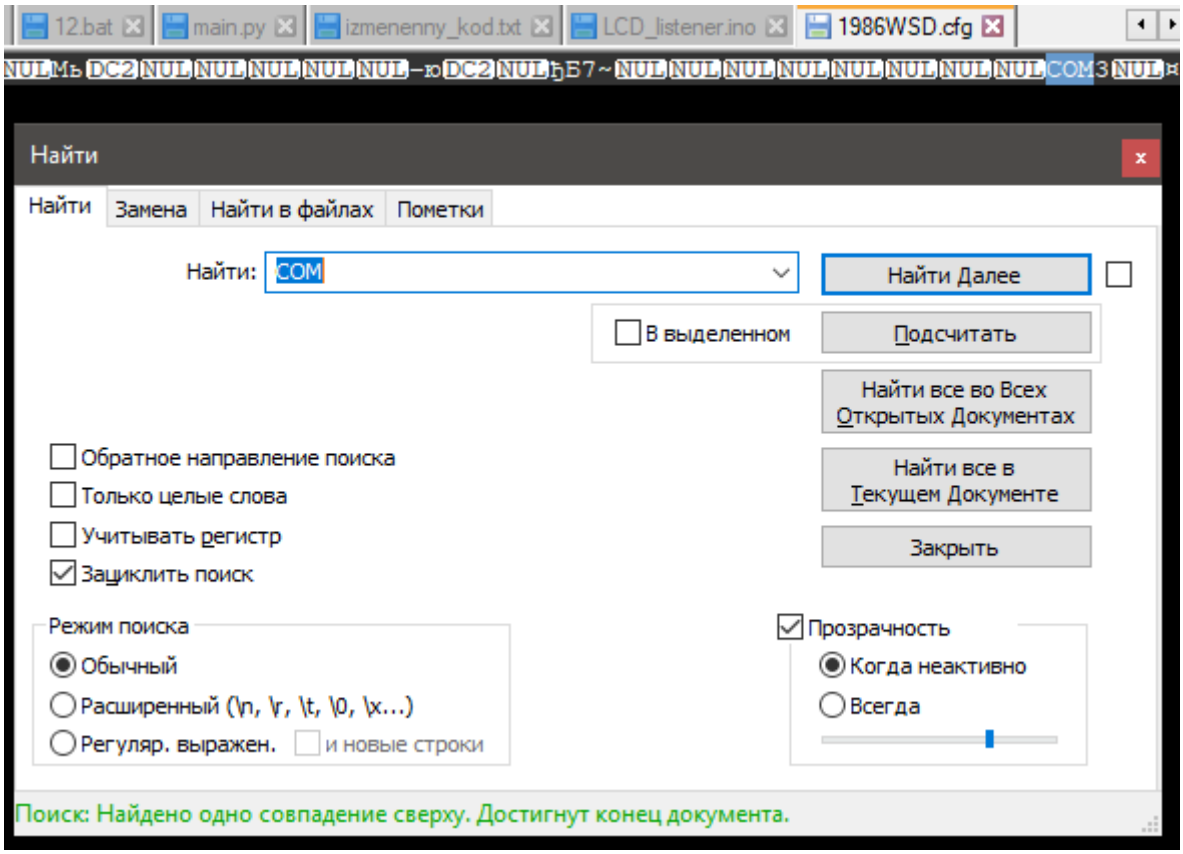


Рисунок 2.4 – Изменение ком порта в файле «1986WSD.cfg»

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

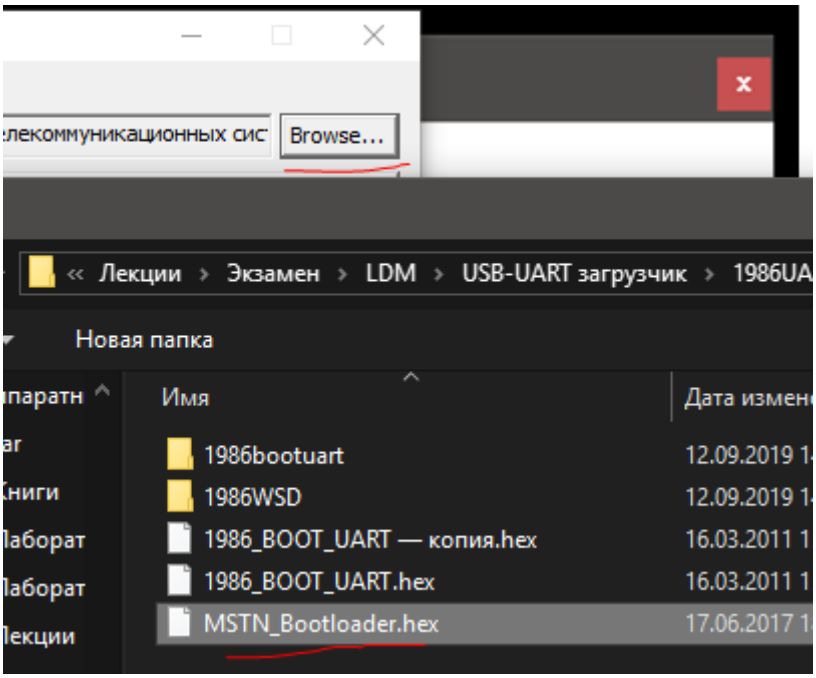


Рисунок 2.5 – Выбор файла загрузчика

Ставим галочки как тут. (Рисунок 2.6)

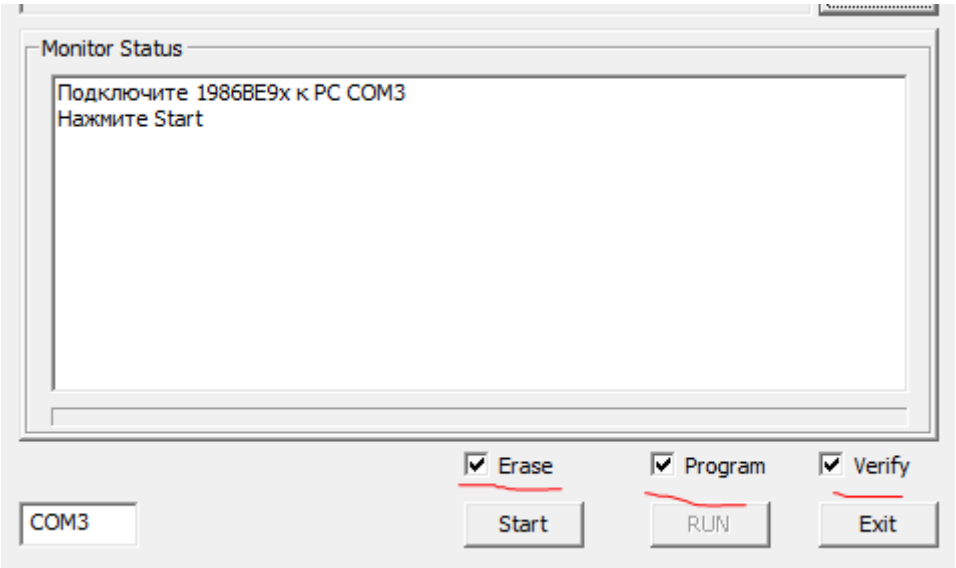


Рисунок 2.6 – Настройка программы

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)

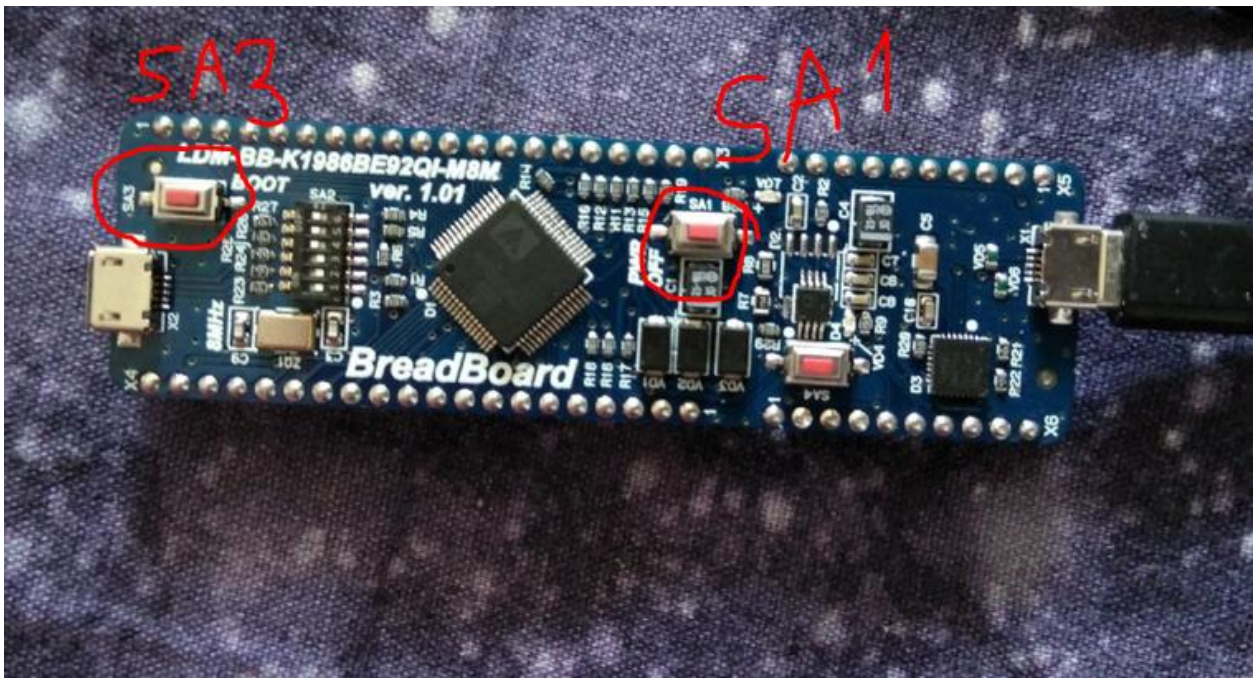


Рисунок 2.7 – Обозначение кнопок на плате для перевода микроконтроллера в специальный режим

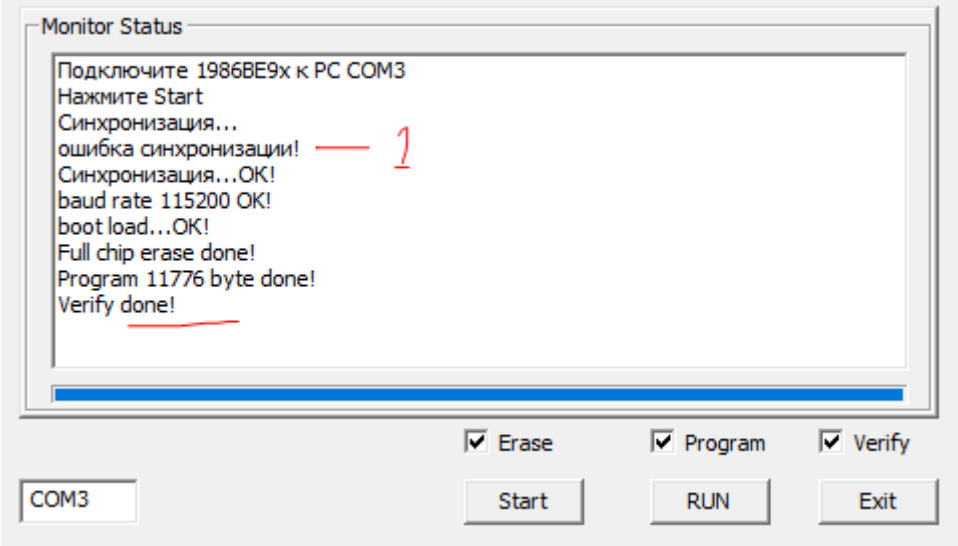


Рисунок 2.8 – Вывод программы об успешной загрузки прошивки на плату

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

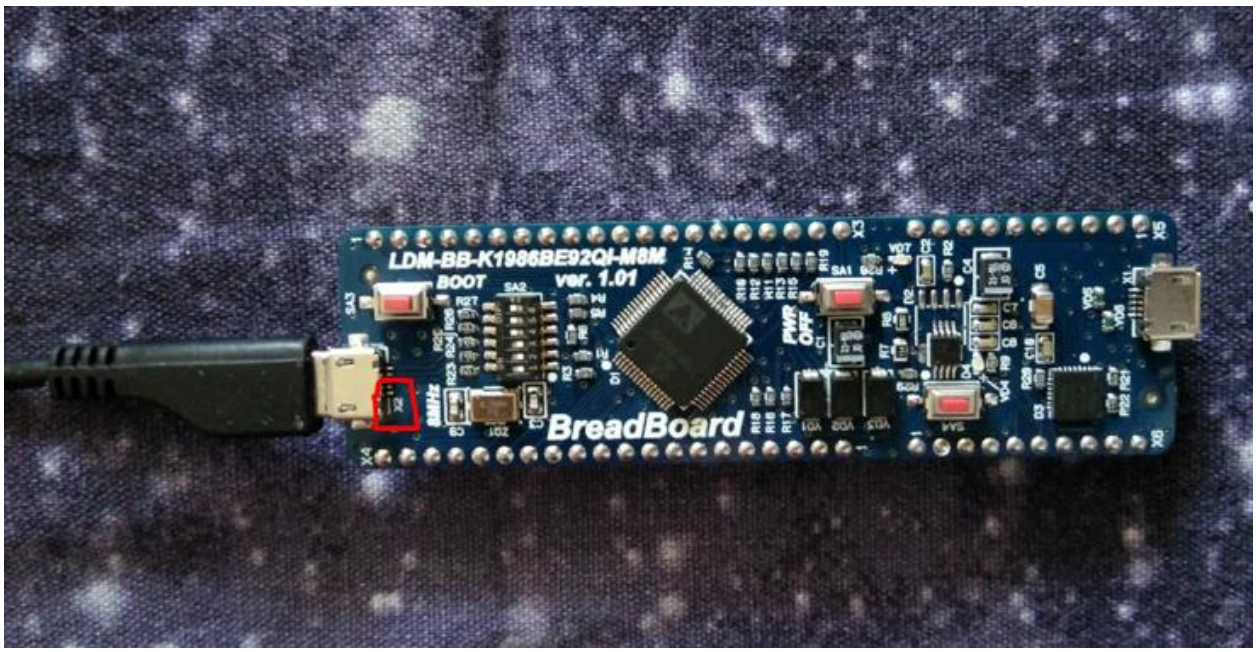


Рисунок 2.9 – Подключение платы для загрузки исполняемого кода (X2)

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)

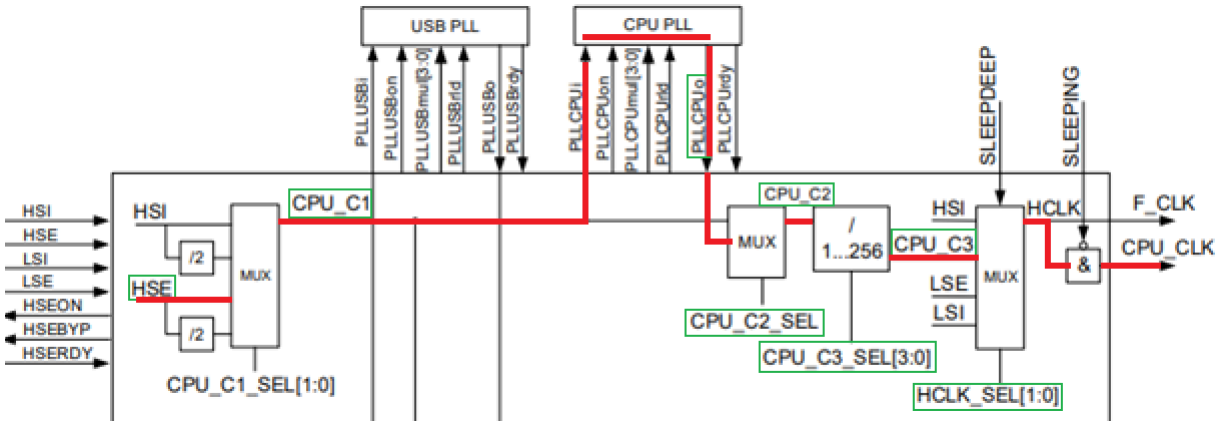


Рисунок 2.14 – Схема прохождения импульсов

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 C 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 C3 ( CPU_C3_SEL = CPU_C2_SEL )
RST_CLK_CPUclkPrescaler (RST_CLK_CPUclkDIV1);
// На C2 идет с PLL, а не напрямую с C1 (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 C3
// (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)

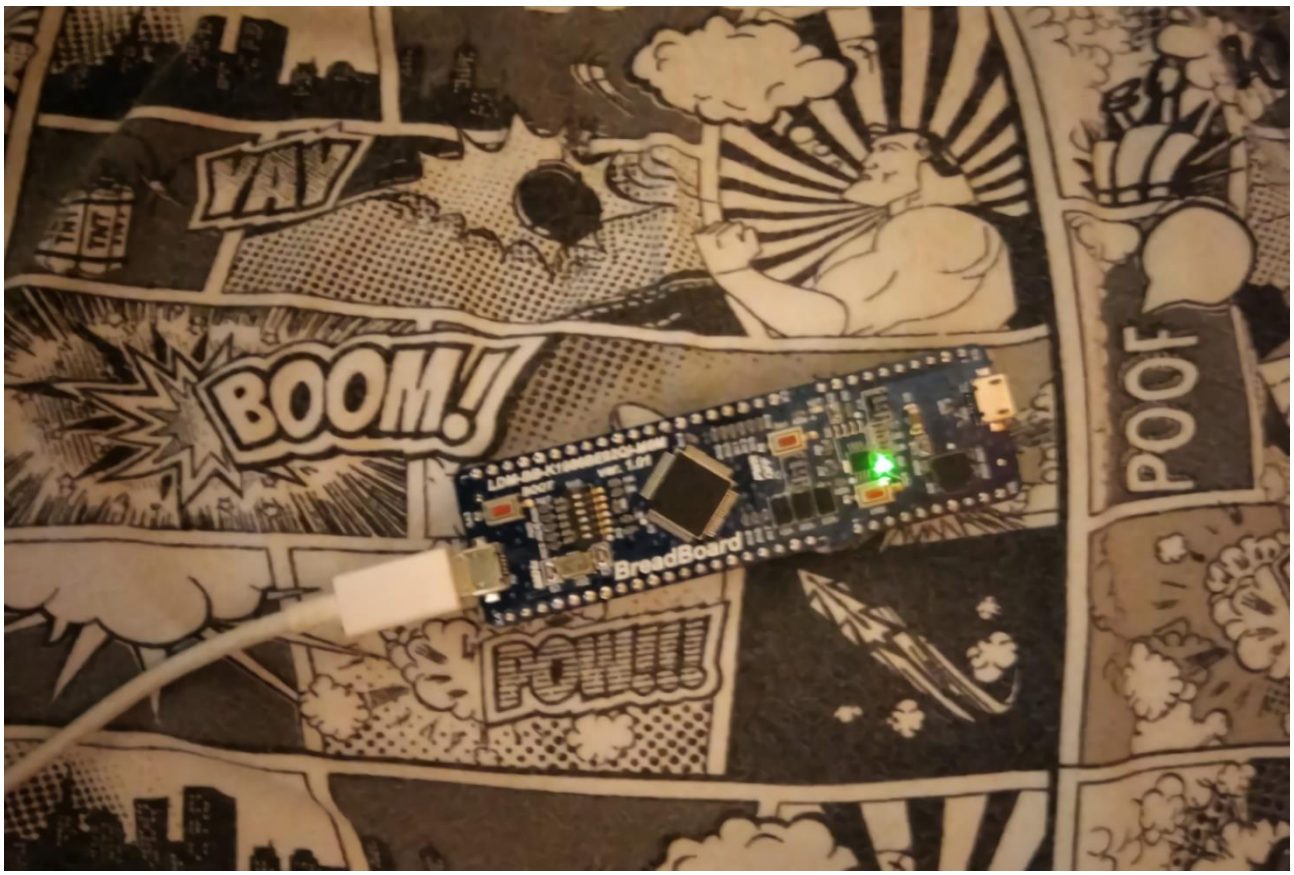


Рисунок 2.16 – Пример работы программы

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)

1 Сек	80 МГц	32 МГц			10 Сек	80 МГц	32 МГц	
0,655		2,152			10,102		24,9	
0,865		2,213			10,102		27,6	
1,001		2,351			9,152		24,802	
1,002		2,351			9,951	9,82675	24,851	25,53825
1,002		2,401						
1,052		2,402			30 сек	80 МГц	32 МГц	
1,111		2,465			29,051		66,502	
1,255	0,992875	2,501			29,201		72,201	
		2,314			29,451		73,351	
		2,561			29,9		74,452	
		2,607			30,001		74,651	
		2,61			30,052		75,302	
		2,362	2,406923		30,101		75,901	
					36,651	30,551	76,15	
							76,552	73,89578

Рисунок 2.17 – Результаты эксперимента

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

$$v2 = \frac{v1 * t1}{t2} \quad (1)$$

	Среднее при		
	80 МГц	32 МГц	МГц из времени
1 сек	0,992875	2,406923	33,00063918
10 сек	9,82675	25,53825	30,78284534
30 сек	30,551	73,89578	33,07469078

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 C2 идет с PLL, а не напрямую с C1 (CPU_C2_SEL = PLL)
RST_CLK_CPU_PLLuse(ENABLE);
//Выбор источника тактирования ядра процессора
```

```
	//CPU берет с выхода C3 (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 C2 идет с PLL, а не напрямую с C1 (CPU_C2_SEL = PLL)
        RST_CLK_CPU_PLLuse(ENABLE);
        //Выбор источника тактирования ядра процессора
        //CPU берет с выхода C3 (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);
}
```