Министерство образования и науки РФ
Рязанский Государственный Радиотехнический университет им В.Ф. Уткина

Кафедра ЭВМ

СПЕЦИАЛИЗИРОВАННЫЕ ЭВМ

"Программирование микроконтроллеров с ядром Cortex-M3 на примере микросхем Миландр 1986BE92QI и 1986BE93У"

Рязань 2021г.

Работа №1

"Программирование микроконтроллеров. Настройка среды Keil uVision. Выводы общего назначения. Параллельные порты" (Часть 1)

Цель работы: закрепить общие представления о выводах общего назначения, как интерфейсе периферийных устройств микропроцессорных систем, а также получить навыки настройки и создания проектов в среде Keil uVision для разработки программ на языке С.

Задачи:

- познакомиться с процессом установки и настройки программного обеспечения для программирования микроконтроллеров Milandr с ядром Cortex M3;

- научиться выполнять настройку проекта для программирования микроконтроллера 1986BE9x;

- закрепить теоретические знания о понятиях «выводы общего назначения» и «параллельный порт»;

- изучить особенности выводов общего назначения микроконтроллеров семейства 1986BE9x;

- получить навыки работы с библиотеками Common Microcontroller Software Interface Standard (CMSIS);

- получить навыки программирования выводов общего назначения микроконтроллеров 1986BE9x в режиме «порт».

Используемое оборудование:

1. Отладочная плата с микроконтроллером Миландр MDR1986BE92QI/MDR1986BE93У.
2. Комплект Программатора
   2.1 программатор JLINK (USB-JTAG);
   2.2 Кабель USB 2.0 A – B;
   2.3 Шина (20 проводников).
3. Источник питания 5В (или дополнительный USB кабель)
4. Комплект Логического анализатора
   4.1 Логический анализатор
   4.2. Кабель USB 2.0A – mini-B
   4.3 Шина 10 проводников.

Используемая документация:

1. Техническое описание на ядро Миландр MDR1986BE9x (файл «1_Тех_описание_ядро_1986BE9X.pdf или по ссылке с сайта разработчика https://ic.milandr.ru/upload/iblock/a33/ioaf9ygfmq1lbxfhd5aad0mukg3dc93s/1986%D0%92%D0%959X.pdf)

2. Выводы отладочной платы микроконтроллера 1986BE92QI (файл «2_Выводы_платы_1986BE92QI.pdf или по ссылке* с сайта разработчика https://ic.milandr.ru/upload/iblock/8f6/8f67b8b736b3ec94edbbeb4777a9c4db.zip)

3. Выводы отладочной платы микроконтроллера 1986BE93 (файл «2_Выводы_платы_1986BE93У.pdf или по ссылке* с сайта разработчика https://ic.milandr.ru/upload/iblock/782/782c4c3b486d6f8d92995e9a44a94401.zip)

При загрузке схемотехнической документации с сайта Milandr (информация о выводах платы), загружается zip архив, который содержит 1 или 2 файла со схемой размещения выводов микроконтроллера на плате и несколько схемотехнических файлов для разводки и печати платы.

Keil uVision. Установка и настройка среды

Среда Keil uVision представляет собой набор средств по написанию программного обеспечения для микроконтроллеров. Она позволяет работать с проектами разной степени сложности и поддерживает разработку программ на языках C и Assembler, позволяет работать с семействами контроллеров ARM, MSC51, C166 и многих других, кроме того возможно использование компиляторов других производителей.

Дистрибутив программы можно получить по ссылке https://www.keil.com/download/product/ выбрав MDK-ARM, и заполнив форму. После загрузки файла – установщика, его следует распаковать, причем конечный путь до программы не должен содержать русских букв и специальных символов помимо "_".

Для разработки программного обеспечения на микроконтроллеры Миландр необходимо дополнительно с сайта производителя http://ic.milandr.ru/soft/ загрузить файлы: "Software pack для Keil MDK 5" и "Standard Peripherals Library".

Файл *Software pack* (обычно имеет имя mdr_sql_v<версия>.pack) можно распаковать автоматически, запустив его, или в случае возникновения проблем воспользоваться программой Pack Installer, которую можно запустить из окна Keil или найти в папке ..\Keil\UV4 (в главном меню выбрать File/Import/ выбрать файл Software Pack).

Для нормальной работы системного отладчика (system viewer) необходимо распаковать из архива Standard Peripherals Library.rar\lib\MDR32F9_2013\lib\IDE\ папку Keil и переместить её содержимое (папки Flash, scatter, SFD) в папку ..\Keil\ARM.

Создание проекта в Keil uVision для разработки программного обеспечения микроконтроллеров на языке С на примере контроллеров MDR1986BE9x фирмы Milandr

При разработке программ для микроконтроллеров в среде Keil uVision 5.XX на языке С создать проект и подключить библиотеки можно двумя способами:

1) С использованием специальной утилиты Manage Run-Time Environment;

2) Распаковав необходимые библиотеки из архива Standard Peripherals Library.rar в папку с проектом.

Рассмотрим оба метода более подробно.

Подключение библиотек с помощью утилиты Manage Run-Time Environment

Последовательность действий:

1. Создать директорию проекта (полный путь к директории не должен содержать русских символов и пробелов)

2. Запустить Keil. В главном меню выбрать Project/New uVision Project. В появившемся диалоговом окне (рисунок 1.1) выбрать директорию и указать название проекта. Сохранить проект.

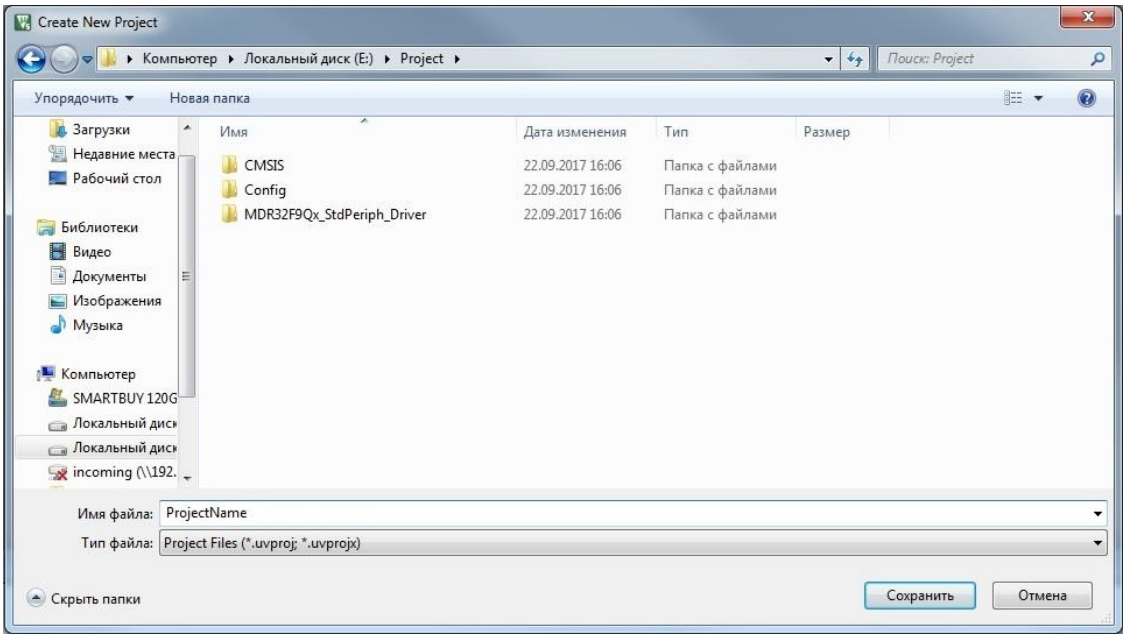


Рисунок 1.1 - Создание проекта

3. Затем необходимо выбрать устройство, под которое разрабатывается программа. В списке необходимо выбрать Milandr/CortexM3/MDR1986BE92 (Рисунок 1.2) (Или MDR1986BE93, при использовании соответствующей отладочной платы)

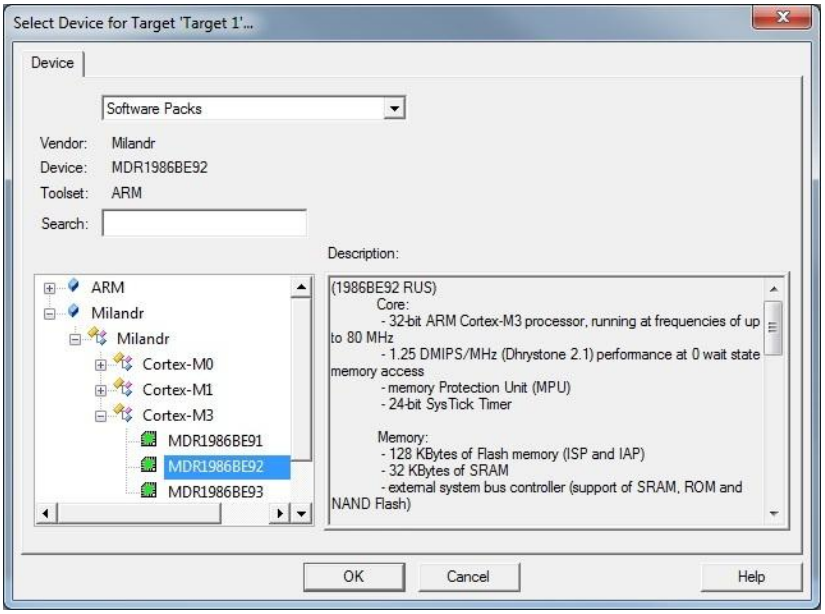


Рисунок 1.2 - Выбор устройства для разработки

4. Следующий этап - выбор библиотек. В появившемся окне Manage Run-Time Environment (рисунок 1.3) требуется выбрать необходимые библиотеки.

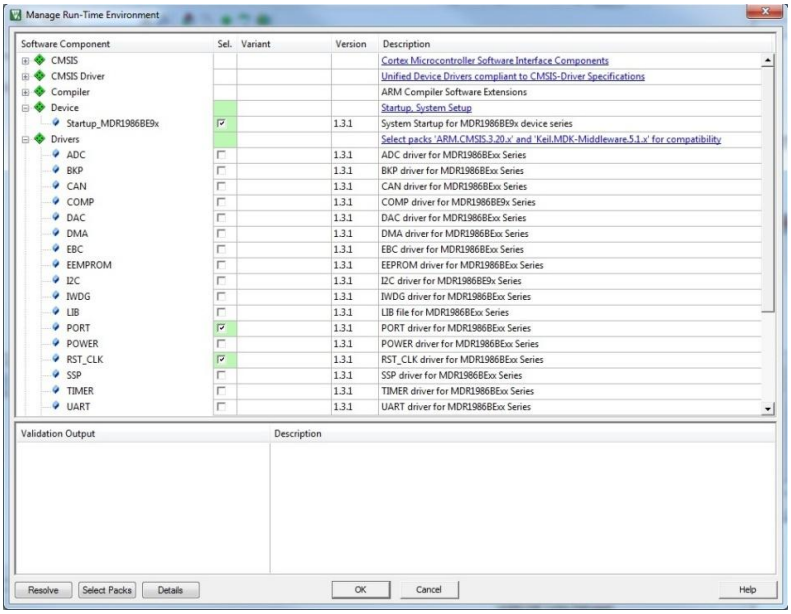


Рисунок 1.3 - Подключение библиотек

5. После необходимо провести настройку параметров проекта соответствующее окно вызывается из главного меню Project/Options for target или сочетанием клавиш (alt+f7).

Окно Options for target состоит из 10 вкладок Device, Target, Output, Listing, User, C/C++, Asm, Linker, Debug, Utilities.

Вкладка Device - позволяет выбрать устройство для проекта.

Вкладка Target - позволяет задать области памяти Read/Only Memory Areas и Read/Write Memory Areas, указать необходимо ли использовать операционную систему RTX Kernel, выбрать компилятор, указать частоту эмуляции. На данной вкладке все параметры можно оставить "по умолчанию", кроме строки System Viewer File. Во избежание проблем с работой отладчика рекомендуется выбрать файл MDR32F9Qx.SFR (или MDR1986BE92.sfr или MDR1986BE93.sfr в зависимости от версии библиотек и используемого контроллера, MDR32F9Qx.SFR – универсальный вариант) из папки ..\Keil\ARM\SFD вручную.

Вкладка Output - позволяет задать расположение получаемых при компиляции файлов и их состав. (рекомендуется оставить значения "по умолчанию")

Вкладка Listing - настройка отчётов

Вкладка User - позволяет прописать дополнительные утилиты, которые будут вызываться при сборке приложения.

Вкладка C/C++ - настройки Си компилятора, оптимизация, дополнительные опции, папки с заголовочными файлами библиотек проекта - Include Paths. Задаётся при ручном подключении библиотек.

Вкладка ASM - настройка компилятора assembler, аналогично C/C++.

Вкладка Linker - – подключение дополнительных утилит.

Вкладка Debug - настройка режимов отладки. Позволяет выбрать на какой платформе отлаживать программу: на симуляторе (simulator) или на реальном микроконтроллере (use: J-Link/J-Trace Cortex или выбрать программатор который используется)

Вкладка Utilities - настройки параметров устройства для загрузки программы (по умолчанию берётся из Debug)

Подключение библиотек вручную

1. Распаковать содержимое архива Standard Peripherals Library.rar в директорию с проектом. Если название распакованной директории имеет другое название, переименуйте таким образом, чтобы оно не содержало символов кириллицы.

2. Подключить необходимые файлы «имя_библиотеки.c» и файл startup.h. к проекту. Для этого необходимо в главном меню выбрать Project/Manage/Project Items см. рисунок 1.4, выбрать любую группу проекта и нажать кнопку «Add Files…» затем добавить к проекту следующие файлы:

а) Для всех проектов

./ Standard Peripherals Library\Libraries\CMSIS\MDR32Fx\DeviceSupport\MDR1986VE9x\startup\arm\system_MDR32F9Qx.c

./ Standard Peripherals Library\Libraries\CMSIS\MDR32Fx\DeviceSupport\MDR1986VE9x\startup\arm\startup_MDR32F9Qx.s

б) При использовании конкретных библиотек периферийных устройств необходимо добавить файлы из директории

./Standard Peripherals Library\Libraries\SPL\MDR32Fx\src

Например:

./Standard Peripherals Library\Libraries\SPL\MDR32Fx\src\MDR32F9Qx_port.c

./Standard Peripherals Library\Libraries\SPL\MDR32Fx\src\MDR32F9Qx_rst_clk.c

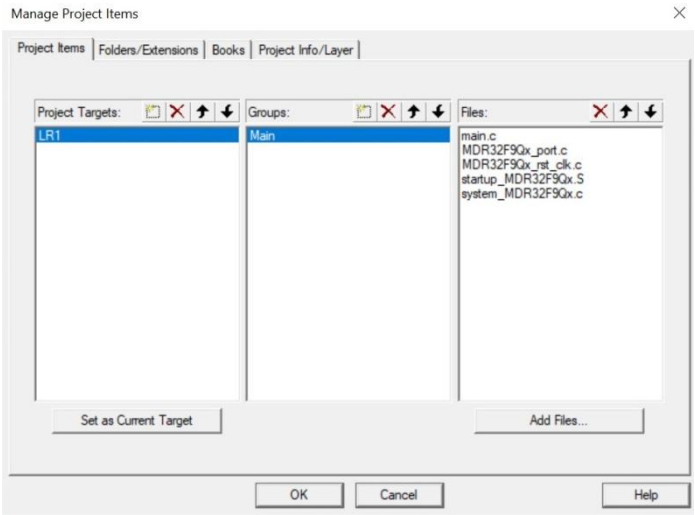


Рисунок 1.4 – добавление файлов библиотек к проекту

3. В панели Project (в левой части окна программы) выбрать и открыть файл MDR32F9Qx_config.h. Снять комментарий с строки `#define USE_MDR1986VE92` или с `#define USE_MDR1986VE93` при работе со стендом с кристаллом 1986BE93. Фрагмент кода представлен ниже.

```
/* Uncomment the line corresponding to the target microcontroller */
/* When using Keil uVision, the definition is set automatically
   (when RTE_Components.h included) according to the selected
   microcontroller in "Options for Target->Device" window.
   For MDR1986VE1T and MDR1986VE3T revision is selected in
   "Device->Startup" variant in "Manage Run-Time Environment" window */
// #define USE_MDR1986VE91
 #define USE_MDR1986VE92   //расскоментировать эту строку
// #define USE_MDR1986VE93//или эту при использовании МК1986ВЕ93
// #define USE_MDR1986VE94
// #define USE_MDR1986VE1T_REV3_4
// #define USE_MDR1986VE1T_REV6
// #define USE_MDR1986VE3T_REV2
// #define USE_MDR1986VE3T_REV3
// #define USE_MDR1901VC1T
```

4. Выполнить настройку согласно пункту 5 первого способа. При этом дополнительно необходимо прописать пути к заголовочным файлам .h. Для этого в окне «Options for target» на вкладке C/C++ в поле «include path» нажать на кнопку «обзор» и в появившемся окне (см. рисунок. 1.5) прописать пути к директориям:

./ Standard Peripherals Library\Libraries\SPL\MDR32Fx\

./ Standard Peripherals Library\Libraries\SPL\MDR32Fx\inc

./Standard Peripherals Library\Libraries\CMSIS\MDR32Fx\CoreSupport\CM3

./Standard Peripherals Library\Libraries\CMSIS\MDR32Fx\DeviceSupport\MDR1986VE3\inc

./Standard Peripherals Library\Libraries\CMSIS\MDR32Fx\DeviceSupport\\MDR1986VE3\startup\arm

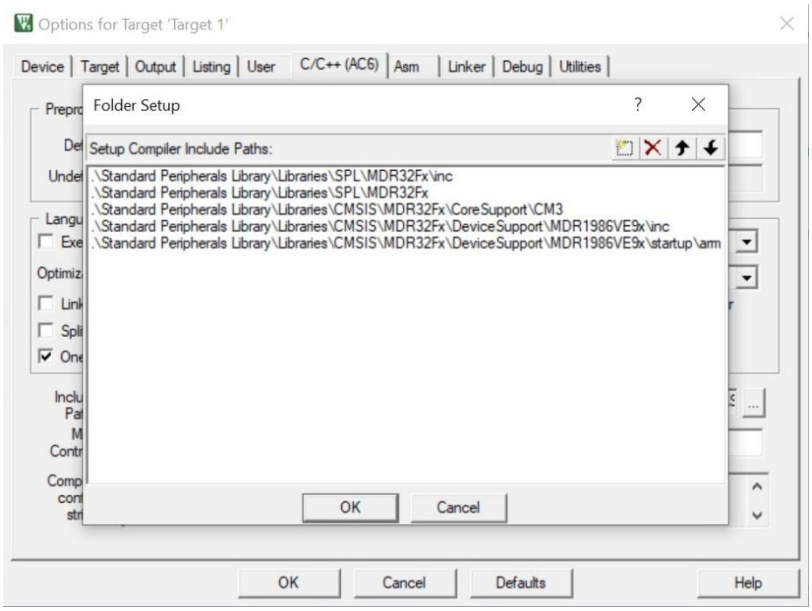


Рисунок 1.5 – Пути к заголовочным файлам

Очевидно, что такой способ подключения более сложный, однако, при необходимости подключения сторонних библиотек, не входящих в пакет CMSIS этот метод является единственным возможным. Более наглядно его применение будет продемонстрировано в работах связанных с LCD дисплеем.

Описание учебного стенда (1986BE92QI)

На рисунке 1.6 представлена структурная схема учебного стенда (отладочной платы).

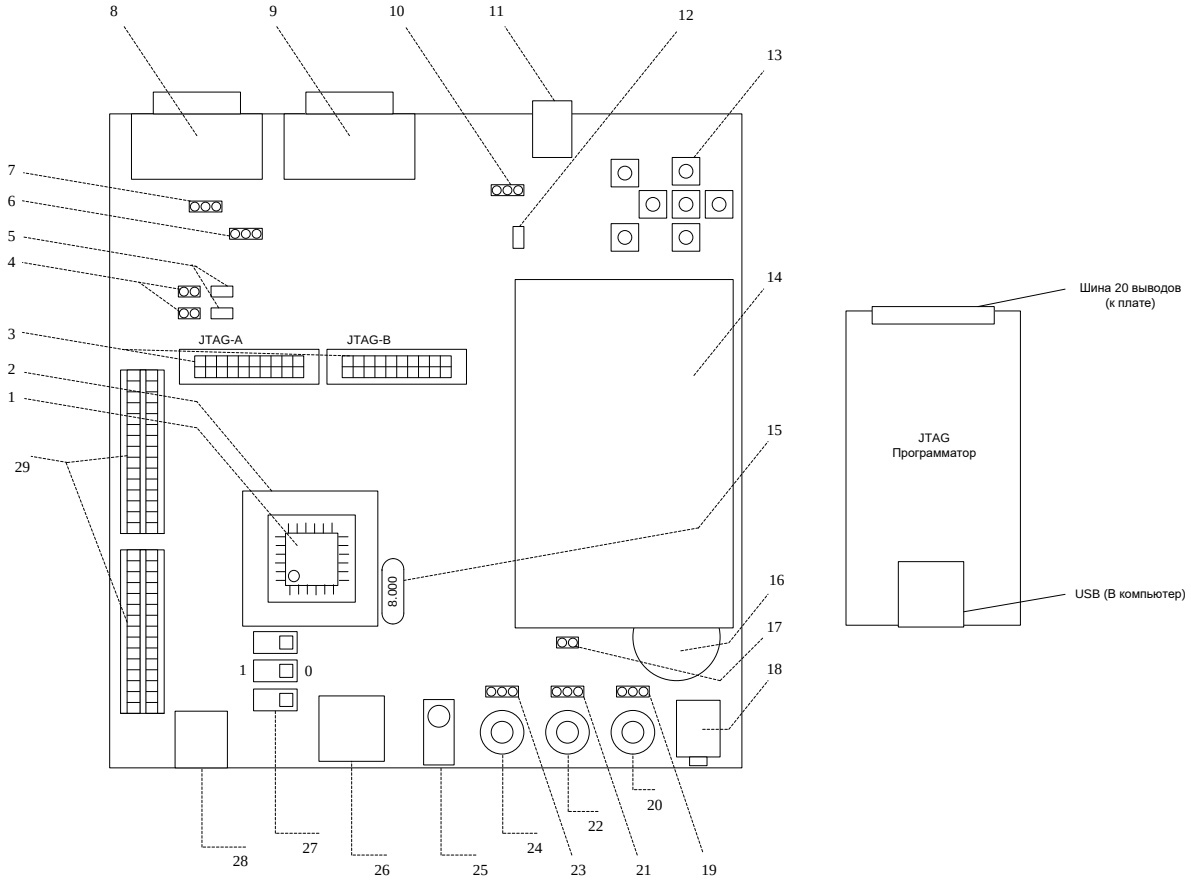


Рисунок 1.6 – Структурная схема отладочного стенда

1. Микроконтроллер Миландр 1986BE92QI
2. Сокет микроконтроллера
3. Разъемы интерфейсов JTAG/SW (A,B)
4. Перемычки, для подключения светодиодов
5. Светодиоды (PC0, PC1)
6. Перемычка скорость передачи CAN (скорость реакции) 125kb/s- 500kb/s
7. Перемычка нагрузки CAN 60Ом-120Ом
8. Разъем D-SUB (интерфейс CAN)
9. Разъем D-SUB (интерфейс RS232)
10. Перемычка. Переключение питания USB/Блок питания
11. Разъем блока питания 5В.
12. Светодиод
13. Кнопочный модуль (Режимы питания, Reset, 5 кнопок общего назначения)
14. Дисплей
15. Кварцевый генератор 8МГц
16. Разъем для источника питания (CR2032 для часов реального времени)
17. Перемычка, подключение источника питания (CR2023)
18. Выход Jack 3.5 AMP (ЦАП)
19. Перемычка для выводов ЦАП (EXT_CON|AMP)
20. Вывод ЦАП
21. Перемычка для выводов аналогового компаратора (EXT_CON|DAC)
22. Вывод аналогового компаратора
23. Перемычка для выводов АЦП (TRIM|EXT_CON)
24. Вывод АЦП
25. Потенциометр (TRIM)
26. Разъём USB
27. Переключатели режима загрузки см. таблицу 1.1
28. Разъем MicroSD
29. Разъемы X26, X27 (см. документ 2 Выводы платы 1986EvBrd_64_Rev3.pdf)

Таблица 1.1 – Режимы загрузки контроллера

SW3	SW2	SW1	Режим загрузки
0	0	0	FLASH/JTAGB
0	0	1	FLASH/JTAGA
0	1	0	EXT_ROM/JTAGB
0	1	1	EXT_ROM
1	1	0	UART2

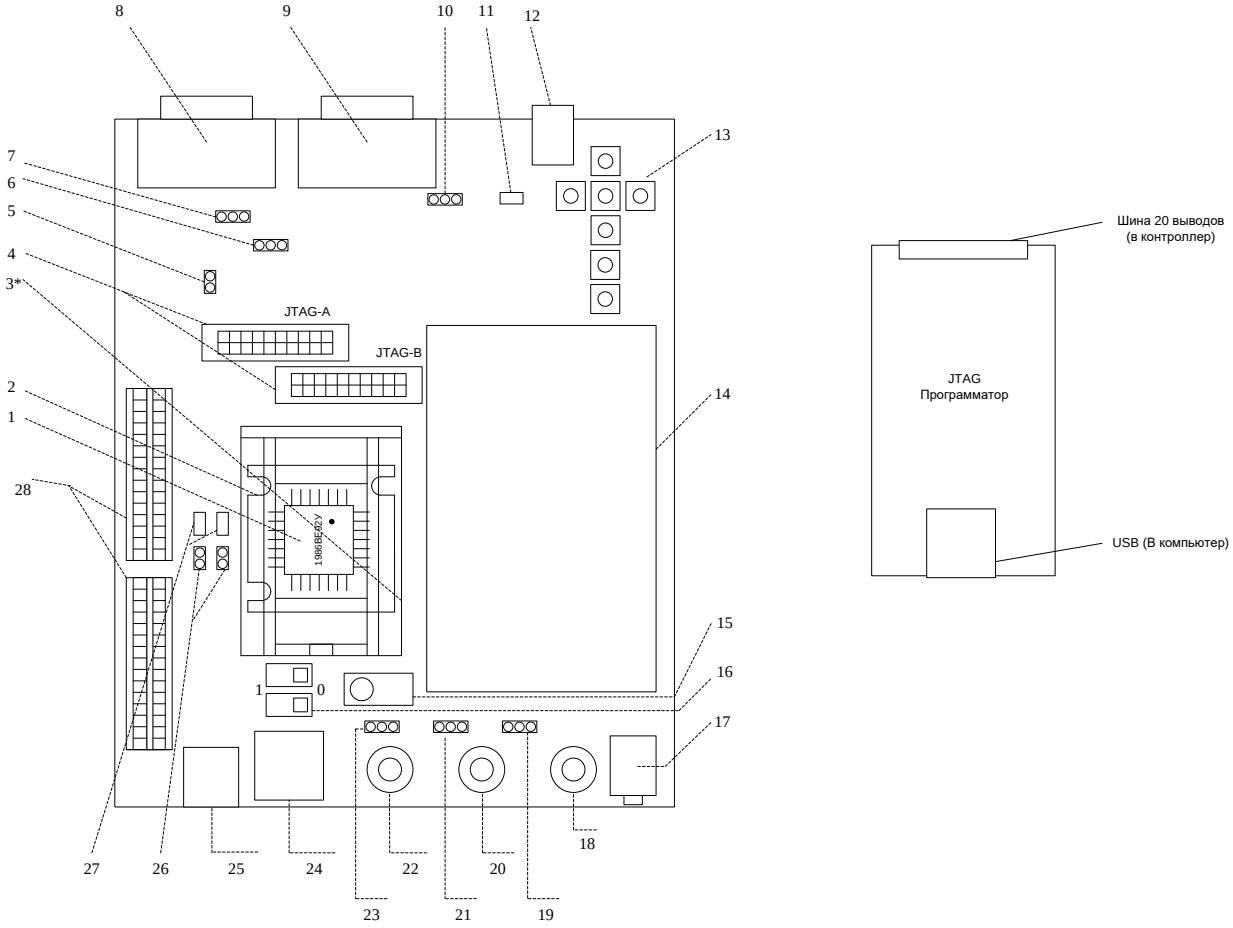


Рисунок 1.7 – Структурная схема отладочного стенда

1. Микроконтроллер Миландр 1986ВЕ93У
2. Сокет микроконтроллера
3*. Кварцевый генератор 8МГц (*расположен на тыльной стороне платы)
4. Разъемы интерфейсов JTAG/SW (A,B)
5. Перемычка X35
6. Перемычка скорость передачи CAN (скорость реакции) 125kb/s- 500kb/s
7. Перемычка нагрузки CAN 60Ом-120Ом
8. Разъем D-SUB (интерфейс CAN)
9. Разъем D-SUB (интерфейс RS232)
10. Перемычка. Переключение питания USB/Блок питания
11. Светодиод. Индикатор питания
12. Разъем блока питания 5В
13. Кнопочный модуль (Режимы питания, Reset, 5 кнопок общего назначения)
14. Дисплей
15. Потенциометр (TRIM)
16. Переключатели режима загрузки см. таблицу 1.2
17. Выход Jack 3.5 AMP (ЦАП)
18. Вывод ЦАП
19. Перемычка для выводов ЦАП (EXT_CON|AMP)
20. Вывод аналогового компаратора
21. Перемычка для выводов аналогового компаратора (EXT_CON|DAC)
22. Вывод АЦП
23. Перемычка для выводов АЦП (TRIM|EXT_CON)
24. Разъём USB
25. Разъем MicroSD
26. Перемычки, для подключения светодиодов
27. Светодиоды (PC0, PC1)
28. Разъемы X26, X27 (см. документ 2 Выводы платы 1986EvBrd_64_Rev3.pdf)

Таблица 1.2 – Режимы загрузки контроллера

SW2	SW1	Режим загрузки
0	0	FLASH/JTAGB
0	1	FLASH/JTAGA
1	0	EXT_ROM/JTAGB
1	1	EXT_ROM

Выводы общего назначения

Выводы общего назначения (General Pin In-Out – GPIO) – группа выводов цифрового устройства не имеющая специализированного назначения. Используется в микропроцессорных системах при нехватке выводов. Может быть использована как для приёма, так и для передачи цифрового или аналогового сигнала.

Семейство контроллеров Milandr 1986BE9X имеет 6 шестнадцатиразрядных портов ввода/вывода. Каждый вывод мультиплексируется между несколькими устройствами внутри контроллера. Вывод может работать в аналоговом режиме (если подключен к АЦП или ЦАП) и 4ёх цифровых: параллельный порт, основная функция, альтернативная или переопределённая. Например, 14ый вывод порта D (PD14) может выступать в качестве 14 канала аналогово-цифрового преобразователя в аналоговом режиме, четвертого канала счётчика 1 в основном режиме, четвёртого канала счётчика 2 в альтернативном режиме, принимающей линии контроллера CAN1 в переопределенном режиме или 14 бита порта D в режиме порт. Конкретное функциональное значение вывода определяется набором параметров настройки порта.

Подробнее о назначении выводов см. техническое описание микроконтроллеров 1986BE9X (Техническое описание ядро 1986BE9x стр. 176-179).

Настройка выводов порта задаётся набором регистров. Для каждого порта выделен свой набор. Он состоит из следующих 32 разрядных регистров (см. таблицу 1.3).

Таблица 1.3 – Регистры GPIO

Имя	Используемые разряды	Назначение
RXTX	[15:0]	Данные порта
OE	[15:0]	Направление порта (ввод или вывод)
FUNC	[31:0]	Режим работы порта
ANALOG	[15:0]	Аналоговый режим работы порта
PULL	[31:0]	Подтяжка порта
PD	[31:0]	Режим работы выходного драйвера
PWR	[31:0]	Режим мощности передатчика
GFEN	[15:0]	Режим работы входного фильтра

Регистр RXTX – хранит данные, принимаемые или передаваемые портом.

Регистр OE – направление порта ввод или вывод данных. При записи 1 в соответствующий бит порта его вывод будет работать на передачу, при 0 на приём информации.

Регистр FUNC – определяет режим работы, каждому выводу порта соответствует пара регистров. Для определения используются следующие значения:

00 – порт;

01 – основная функция;

10 – альтернативная функция;

11 – переопределенная функция.

Регистр ANALOG – определяет в цифровом или аналоговом режиме работает порт: 0 – аналоговый, 1 цифровой.

Регистр PULL – разделён на две части. Биты [31:16] – отвечают за работу режима PULL-UP, Биты [15:0] – отвечают за работу режима PULL-DOWN. 0 в соответствующем бите подтяжка выключена, 1 подтяжка (порядка 50КОм) включена.

Регистр PD – разделён на две части. Биты [31:16] – Режим работы входа: 0 – триггер Шмитта выключен гистерезис 200 мВ; 1 – триггер Шмитта включен гистерезис 400 мВ. Биты [15:0] – Режим работы выхода: 0 – управляемый драйвер; 1 – открытый сток.

Регистр PWR – определяет скорость формирования фронта:

00 – зарезервировано (передатчик отключен)

01 – медленный фронт (порядка 100 нс)

10 – быстрый фронт (порядка 20 нс)

11 – максимально быстрый фронт (порядка 10 нс)

Регистр GFEN – фильтр, подавление высокочастотных помех (длина импульса до 10нс): 0 – выключен, 1 включен.

Рассмотрим пример настройки вывода PORTC1 в режиме «порт» с медленным формированием фронта на приём информации с подтяжкой вверх см. Таблицу 1.4.

Таблица 1.4 – Обращение к регистрам на языке С

Код обращения к регистру на С	Назначение
`MDR_PORTC->ANALOG = 0x00000002;`	PC1 – цифровой (остальные** аналоговые)
`MDR_PORTC->FUNC = 0x00000040;*`	PC1 – режим порт
`MDR_PORTC->GFEN = 0x00000002;`	PC1 – включить фильтр (остальные фильтр отключен)
`MDR_PORTC->OE = 0x00000000;`	PC1 – на ввод (как и остальные)
`MDR_PORTC->PD = 0x00000000`	По умолчанию см. описание PD
`MDR_PORTC->PULL = 0x00040000`	PC1 – PULL-UP (остальные подтяжки выключены)
`MDR_PORTC->PWR = 0x00000040`	PC1 – медленный фронт (остальные выключены)
*Примечания: *0x00000040 равно 0x40, незначащие нули можно опускать. **Остальные – это выводы PC[15:2], PC0*	

Для упрощения операций с портами производителем контроллера предложена библиотека MDR32F9Qx_port.

Описание библиотеки MDR32F9Qx_port

Библиотека MDR32F9Qx_port - предназначена для работы с портами микроконтроллера. Она состоит из следующих параметров (констант и определений):

1. Описание имён портов (для MDR1986VE9x)

MDR_PORTA	Порт A
MDR_PORTB	Порт B
MDR_PORTC	Порт C
MDR_PORTD	Порт D
MDR_PORTE	Порт E
MDR_PORTF	Порт F

2. Направление порта ввод/вывод PORT_OE_TypeDef

PORT_OE_IN	Режим ввода данных
PORT_OE_OUT	Режим вывода данных

3. Аналоговый/цифровой режим PORT_MODE_TypeDef

PORT_MODE_ANALOG	Аналоговый режим
PORT_MODE_DIGITAL	Цифровой режим

4. Использование подтяжки "вверх" PORT_PULL_UP_TypeDef

PORT_PULL_UP_OFF	Подтяжка не используется
PORT_PULL_UP_ON	Подтяжка используется

5. Использование подтяжки "вниз" PORT_PULL_DOWN_TypeDef

PORT_PULL_DOWN_OFF	Подтяжка не используется
PORT_PULL_DOWN_ON	Подтяжка используется

6. Режим работы выходного драйвера (триггер Шмитта) PORT_PD_SHM_TypeDef

PORT_PD_SHM_OFF	Триггер Шмитта выключен
PORT_PD_SHM_ON	Триггер Шмитта включен

7. Режим работы выходного драйвера (Драйвер/открытый сток) PORT_PD_TypeDef

PORT_PD_DRIVER	Управляемый драйвер
PORT_PD_OPEN	Открытый сток

8. Режим работы выходного фильтра PORT_GFEN_TypeDef

PORT_GFEN_OFF	Фильтр выключен
PORT_GFEN_ON	Фильтр включен

9. Функция порта PORT_FUNC_TypeDef;

PORT_FUNC_PORT	Функция "Порт"
PORT_FUNC_MAIN	Основная функция
PORT_FUNC_ALTER	Альтернативная функция
PORT_FUNC_OVERRID	Переопределённая функция

10. Режим мощности передатчика PORT_SPEED_TypeDef

PORT_OUTPUT_OFF	Передатчик отключен
PORT_SPEED_SLOW	Медленный фронт
PORT_SPEED_FAST	Быстрый фронт
PORT_SPEED_MAXFAST	Максимально быстрый фронт

11. Параметр действия BitAction

Bit_RESET	Сброс в "0"
Bit_SET	Установка "1"

12. Имена выводов порта

PORT_Pin_0	Вывод 0	PORT_Pin_4	Вывод 4	PORT_Pin_8	Вывод 8	PORT_Pin_12	Вывод 12
PORT_Pin_1	Вывод 1	PORT_Pin_5	Вывод 5	PORT_Pin_9	Вывод 9	PORT_Pin_13	Вывод 13
PORT_Pin_2	Вывод 2	PORT_Pin_6	Вывод 6	PORT_Pin_10	Вывод 10	PORT_Pin_14	Вывод 14
PORT_Pin_3	Вывод 3	PORT_Pin_7	Вывод 7	PORT_Pin_11	Вывод 11	PORT_Pin_15	Вывод 15
PORT_Pin_All				Использовать все выводы			

Структуры:
1. Структура инициализации порта

```
{
  uint16_t PORT_Pin;    //выводы порта
  PORT_OE_TypeDef PORT_OE;  //направление
  PORT_PULL_UP_TypeDef PORT_PULL_UP;   //подтяжка вверх
  PORT_PULL_DOWN_TypeDef PORT_PULL_DOWN; //подтяжка вниз
  PORT_PD_SHM_TypeDef PORT_PD_SHM;       //триггер Шмитта
  PORT_PD_TypeDef PORT_PD;               //драйвер
  PORT_GFEN_TypeDef PORT_GFEN;           //фильтр
  PORT_FUNC_TypeDef PORT_FUNC;           //функция работы порта
  PORT_SPEED_TypeDef PORT_SPEED;      //режим мощности передатчика
  PORT_MODE_TypeDef PORT_MODE;           //режим порта (цифровой/аналоговый)
}PORT_InitTypeDef;// структура инициализации
```

Процедуры:
1. Деинициализация

```
void PORT_DeInit(MDR_PORT_TypeDef* PORTx);
```

2. Инициализация

```
void PORT_Init(MDR_PORT_TypeDef* PORTx, const PORT_InitTypeDef*
PORT_InitStruct);
```

3. Инициализация структуры

```
void PORT_StructInit(PORT_InitTypeDef* PORT_InitStruct);
```

4. Установить бит/биты порта в "1"

```
void PORT_SetBits(MDR_PORT_TypeDef* PORTx, uint32_t PORT_Pin);
```

5. Сбросить бит/биты порта в "0"

```
void PORT_ResetBits(MDR_PORT_TypeDef* PORTx, uint32_t PORT_Pin);
```

6. Записать значение в разряд порта

```
void PORT_WriteBit(MDR_PORT_TypeDef* PORTx, uint32_t PORT_Pin, BitAction
BitVal);
```

7. Записать значение в порт

```
void PORT_Write(MDR_PORT_TypeDef* PORTx, uint32_t PortVal);
```

Функции:
1. Читать значение бита из порта
```
uint8_t PORT_ReadInputDataBit(MDR_PORT_TypeDef* PORTx, uint32_t PORT_Pin);
```
2. Читать значение из порта
```
uint32_t PORT_ReadInputData(MDR_PORT_TypeDef* PORTx);
```

Практическая часть

Задание 1.1. Настройка проекта. Мигание диодом

Выполните следующую последовательность действий:

1.1 Создать каталог D:/users/"№ группы"/LR1/ (В пути не должно быть пробелов и русских символов!)

1.2 Открыть SDK - Keil uVision 5.XX

1.3 Создать новый проект в директории указанной в п.1

1.4 В окне Select Device for Target выбрать Milandr/CortexM3/MDR1986BE92 (или Milandr/CortexM3/MDR1986BE93 при использовании соответствующей отладочной платы)

1.5 В окне Manage Run Time Environment выбрать следующие бибилотеки:

- Device/Startup_mdr1986BE9x (в более поздних библиотеках startup) - подключение файла инициализации прерываний и точки входа в программу;

- Drivers/rts_clk - подключение библиотеки генератора тактовых сигналов;

- Drivers/port - подключение библиотеки параллельных портов.

1.6 В поле "Project" щелкнуть правой кнопкой мыши по значку Target1 и выбрать "Manage Project Items". В поле Project Targets - переименовать Target1 в LR1. В поле "Groups" - Source Group 1 переименовать в Main group. Нажать на кнопку OK.

1.7 В поле Project выбрать группу Main group и щелкнув правой кнопкой мыши выбрать "Add New Item To Group Main group".

1.9 Создать файл .C с именем main.c

1.8 В главном меню выбрать Edit/Configuration. В появившемся окне в параметр Encoding установить «Encode in UTF8 without signature» (правильное отображение комментариев на русском языке).

1.10 Внесите следующие строки в файл main.c

```
#include "MDR32F9Qx_port.h"// подключение заголовочного файла библиотеки параллельных портов
#include "MDR32F9Qx_rst_clk.h" // подключение заголовочного файла библиотеки тактового генератора

static PORT_InitTypeDef PortInit; //объявление переменной типа структура данных для инициализации параметров порта
void Delay (int del)//объявление процедуры "задержки"(пустой цикл с задаваемым числом повторений)
{
      for (int i=0; i<del; i++);//описание цикла в соответствии с синтаксисом языка C
}
void LedPinCfg (void) //описание процедуры инициализации порта
{
//инициализация порта осуществляется через структуру типа PORT_InitTypeDef
      PortInit.PORT_Pin   = (PORT_Pin_0);//выбор вывода порта, в данном случае 0 пин порта C
      PortInit.PORT_OE    = PORT_OE_OUT;//порт для вывода информации
      PortInit.PORT_FUNC  = PORT_FUNC_PORT;//порт в режиме стандартной функции
      PortInit.PORT_MODE  = PORT_MODE_DIGITAL;//цифровой режим порта
      PortInit.PORT_SPEED  =  PORT_SPEED_SLOW;//скорость перехода сигнала от одного уровня к другому - медленно
      PORT_Init(MDR_PORTC, &PortInit);//инициализация порта
}

int main (void)//
{
      RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTC,   ENABLE);//включить   тактирование порта C
      LedPinCfg();//выполнение функции инициализации порта
while (1) {//бесконечный цикл
      PORT_SetBits(MDR_PORTC, PORT_Pin_0);//установить единицу в 0 бит порта C
      Delay (100000); //Задержка в 100 000 операций процессора
      PORT_ResetBits(MDR_PORTC, PORT_Pin_0);//сбросить младший бит порта C
      Delay (100000); //Задержка в 100 000 операций процессора
      }
}
```

**Для микроконтроллера 1986BE93У – изменить константу MDR_PORTC на MDR_PORTF в процедуре инициализации, при вызове функций тактирования, установки бита и сброса.*

1.11 Подключите программатор к плате, выбрав любой из портов JTAG (желательно, чтобы порт не использовался с другим назначением в проекте, подробнее в работе 2). Сконфигурируйте перемычки BOOT SELECT (см. №27 на рисунке 1.6, таблицу 1.3, комментарий на плате). Подключите питание к контроллеру. В зависимости от источника питания (от USB или отдельного разъема внешнего питания) сконфигурируйте перемычку POWER_SEL (см. №10 на рисунке 1.6).

1.12 Откройте настройки проекта Projects/Opions for target LR1. Проверьте следующие настройки.

- На вкладке Device - выбрано устройство MDR1986BE92QI (или MDR1986BE93У)

- На вкладке Target в поле System Viewer File - выбрано MDR1986BE92.svd (или MDR1986BE93.svd), значение Xtal – 8МГц.

- На вкладке C/C++ установлена галочка в поле C99mode

- На вкладке Debug в поле use выбрано "J-link/j-trace cortex"

- Нажмите на кнопку Settings рядом с полем use, в случае появления окна J-link v4.98 Device Selection (см. Рисунок 1.7), нажмите Yes в появившемся окне со списком устройств выберите Unspecified Cortex-M3. После перехода к окну Cortex J-LINK/J-TRACE Target Driver Setup убедитесь, что в поле SN - указан серийный номер программатора. Это свидетельствует о том что программатор подключен в компьютеру и работает корректно. Переключите выпадающий список Port из состояния JTAG в состояние SW. В этом случае в правой части окна появится IDCODE контроллера ARM (см. рисунок 1.8). Это говорит о том, что отладочная плата подключена правильно. (Если это не произошло - проверьте подключение программатора к плате. Проверьте в правильный ли разъем JTAG подключен программатор, убедитесь что правильно настроены переключатели BOOT Select) Перейдите на вкладку Flash Download и выберите следующие функции: Erase Full Chip (полная очистка памяти контроллера в случае программирования), Program (программировать), Verify (проверить), Reset and Run (перезапустить контроллер после прошивки и начать выполнение программы) (см. рисунок 1.9).

Рисунок 1.7 - Окно J-link v4.98 Device Selection

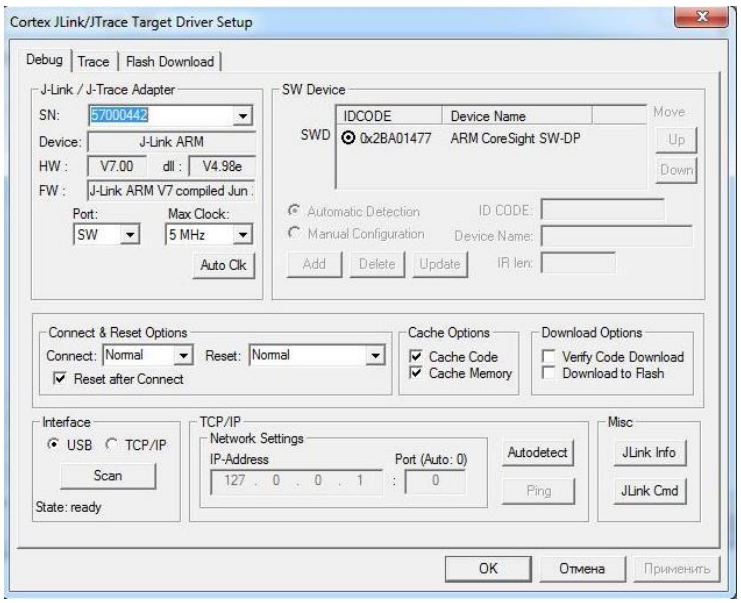


Рисунок 1.8 - Вкладка Debug окна Cortex Jlink/Jtrace target Driver Setup

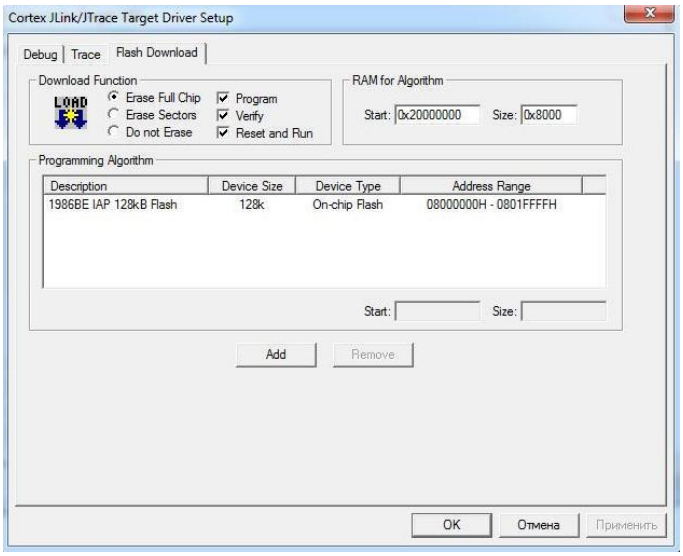


Рисунок 1.9 - Вкладка Flash Download окна Cortex Jlink/Jtrace target Driver Setup

1.13 Выполните загрузку программы в контроллер, нажав на кнопку LOAD.

1.14 Проанализируйте работу устройства.

Обратите внимание, в программе есть строка, отвечающая за тактирование порта. По умолчанию все периферийные устройства микроконтроллера выключены, от них отключен тактовый генератор для экономии энергии. Управление тактированием описано в библиотеке MDR32F9Qx_rst_clk. Особенности тактирования ядра и периферии будут описаны в работе 3.

Задание 1.2: Инициализация нескольких бит одного порта

Измените программу таким образом, чтобы два диода подключенные на плате к выводам PC0 и PC1 (для 1986BE93 PF0, PF1) загорались и гасли поочерёдно. (Примечание: для инициализации можно использовать уже имеющуюся процедуру LedPinCfg, достаточно внести следующую правку "PortInit.PORT_Pin = (PORT_Pin_0 | **PORT_Pin_1**);", тогда оба вывода порта будут проинициализированы одинаково)

Задание 1.3: Вывод информации через параллельный порт

3.1 Напишите процедуру инициализации всех выводов порта A (для этого можно использовать параметр PORT_Pin_All)

3.2 Измените процедуру main таким образом, чтобы в ней выполнялся вывод значения переменной на 8 выводов порта A, а сама переменная увеличивала своё значение с каждым шагом цикла While(1) (вывод несколько битного значения осуществляется процедурой

void PORT_Write(MDR_PORT_TypeDef* PORTx, uint32_t PortVal);)

3.3 Подключите логический анализатор к плате согласно схеме на рисунке 1.10. (Для отладочной платы МК1986ВЕ93У разъем X25, выполнен аналогично представленному рисунку, кроме выводов PE1, PF6. Подробнее см. документ 4 - Выводы платы 1986BE93).

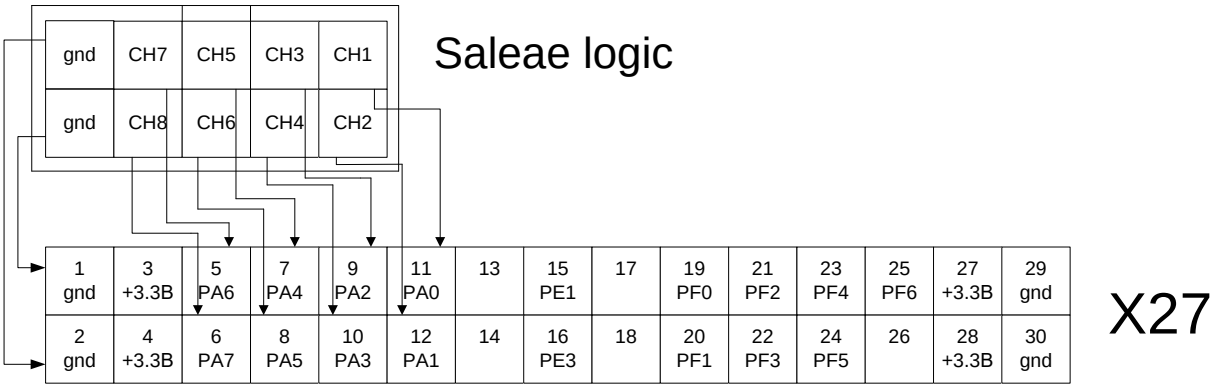


Рисунок 1.10 - схема подключения логического анализатора к порту A

3.4 Запустите программу в контроллер, запустите Logic и проанализируйте полученные значения.

Вопросы для самоконтроля

1. Объясните понятие «GPIO».
2. Чем понятие «GPIO» отличается от понятия «параллельный порт»?
3. Зачем используется утилита Manage Run-Time Environment?
4. Какие преимущества даёт использование библиотеки MDR32F9Qx_port?
5. Поясните, зачем выводу порта может требоваться подтяжка?
6. Сколько режимов работы может быть у вывода GPIO МК 1986BE9x.
7. Зачем нужен регистр GFEN?
8. Как вы считаете, зачем микроконтроллеру режим медленного (100нс) и быстрого (20нс) фронта, если он способен формировать фронт за время 10нс?
9. Какие значения должны принять регистры GPIO чтобы настроить вывод PD7 на приём информации в режиме «порт» с подтяжкой вверх, Триггером Шмитта с гистерезисом 400 мВ и подавлением высокочастотных помех?
10. Напишите фрагмент программы, который обеспечивает инициализацию порта С, в режиме параллельного порта, на передачу 8ми битных данных.

Работа №2

"Программирование микроконтроллеров. Настройка среды Keil uVision. Выводы общего назначения. Параллельные порты" (Часть 2)

Цель работы: закрепить общие представления о выводах общего назначения, как интерфейсе периферийных устройств микропроцессорных систем, а также получить навыки настройки и создания проектов в среде Keil uVision для создания программ на языке С.

Задачи:

- познакомиться с процессом установки и настройки программного обеспечения для программирования микроконтроллеров Миландр с ядром Cortex M3;

- научиться выполнять настройку проекта для программирования микроконтроллера 1986BE9x;

- закрепить теоретические знания о понятиях «выводы общего назначения» и «параллельный порт»;

- изучить особенности выводов общего назначения микроконтроллеров семейства 1986BE9x;

- получить навыки работы с библиотеками Common Microcontroller Software Interface Standard (CMSIS);

- получить навыки программирования выводов общего назначения микроконтроллеров 1986BE9x в режиме «порт».

Используемое оборудование:

1. Отладочная плата с микроконтроллером Миландр MDR1986BE92QI/MDR1986BE93У.
2. Комплект Программатора
   2.1 программатор JLINK (USB-JTAG);
   2.2 Кабель USB 2.0 A – B;
   2.3 Шина (20 проводников).
3. Источник питания 5В (или дополнительный USB кабель)
4. Комплект Логического анализатора
   4.1 Логический анализатор
   4.2. Кабель USB 2.0A – mini-B
   4.3 Шина 10 проводников.

Используемая документация:

1. Техническое описание на ядро Миландр MDR1986BE9x (файл «1_Тех_описание_ядро_1986BE9X.pdf или по ссылке с сайта разработчика https://ic.milandr.ru/upload/iblock/a33/ioaf9ygfmq1lbxfhd5aad0mukg3dc93s/1986%D0%92%D0%959X.pdf)

2. Выводы отладочной платы микроконтроллера 1986BE92QI (файл «2_Выводы_платы_1986BE92QI.pdf или по ссылке* с сайта разработчика https://ic.milandr.ru/upload/iblock/8f6/8f67b8b736b3ec94edbbeb4777a9c4db.zip)

3. Выводы отладочной платы микроконтроллера 1986BE93 (файл «2_Выводы_платы_1986BE93У.pdf или по ссылке* с сайта разработчика https://ic.milandr.ru/upload/iblock/782/782c4c3b486d6f8d92995e9a44a94401.zip)

При загрузке схемотехнической документации с сайта Milandr (информация о выводах платы), загружается zip архив, который содержит 1 или 2 файла со схемой размещения выводов микроконтроллера на плате и несколько схемотехнических файлов для разводки и печати платы.

Теоретическая часть

Выводы общего назначения

См. теоретические материалы лабораторной работы 1.

Подготовка проекта. Дополнительные настройки

В некоторых случаях неправильная инициализация периферийных устройств или системы тактирования может привести к блокировке (залочиванию) контроллера.

Одной из функций выводов общего назначения является загрузка программы в память микроконтроллера и её отладка. По умолчанию выводы интерфейсов JTAG/SW инициализированы в этом режиме. При решении задач, возможно использование портов, выводы которых задействованы для загрузки программы. При этом можно выполнить инициализацию таким образом, что JTAG будет больше не доступен, и перепрограммировать микроконтроллер будет невозможно. В библиотеках, обеспечивающих работу с выводами общего назначения, есть специальные «заглушки», запрещающие операции с портами, у выводов которые имёют функции загрузки/отладки программы. В микроконтроллерах 1986BE9x для JTAG используются следующие порты:

JTAGA – PORTB (PB0, PB1, PB2, PB3, PB4);

JTAGB – PORTD (PD0, PD1, PD2, PD3, PD4).

В заголовочном файле MDR32F9Qx_config.h (фрагмент файла представлен в таблице 2.1) объявлены символические константы USE_JTAG_A и USE_JTAG_B.

Если в проекте не используются выводы порта B, то они могут быть использованы для загрузки программы в контроллер. В этом случае необходимо снять комментарий у строки `#define USE_JTAG_A` и закомментировать строку `#define USE_JTAG_B`. Далее при загрузке программы в контроллер целесообразно использовать интерфейс JTAGA.

Если не используются выводы порта D, то целесообразно использовать JTAGB для загрузки программы, а в файле MDR32F9Qx_config.h снять

комментарий у строки `#define USE_JTAG_B` и закомментировать строку `#define USE_JTAG_A`.

Таблица 2.1 – фрагмент файла MDR32F9Qx_config.h

```
/* Uncomment the line(s) below to define used JTAG port(s). Leave all commented
       * if there is no JTAG ports */
      #if (defined(USE_MDR1986VE9x) || defined (USE_MDR1901VC1T))
       //#define USE_JTAG_A
       #define USE_JTAG_B
      #endif
```

Константы USE_JTAG_A и USE_JTAG_B используются в библиотеке для работы с портами – MDR32F9Qx_port. В файле MDR32F9Qx_port.h они определяют маски, используемые в функциях файла MDR32F9Qx_port.c (см. таблицу 2.2) и запрещают операции над битами, отведёнными под соответствующий JTAG. Однако значения всё равно могут быть изменены при прямом обращении к регистрам порта (например, `PORTB->RXTX=0x0F`). Поэтому рекомендуется выполнять операции с портами с использованием библиотеки MDR32F9Qx_port и правильной конфигурацией MDR32F9Qx_config.h.

Таблица 2.2 – фрагмент файла MDR32F9Qx_port.h

```
#if defined (USE_JTAG_A)
#define PORT_JTAG       MDR_PORTB    /*!< Port containing JTAG interface */
#define PORT_JTAG_Msk   0x001FU      /*!< JTAG pins */

#elif defined (USE_JTAG_B)
#define PORT_JTAG       MDR_PORTD    /*!< Port containing JTAG interface */
#define PORT_JTAG_Msk   0x001FU      /*!< JTAG pins */

#endif

#if defined (PORT_JTAG)
#define JTAG_PINS(PORT) (((PORT) == PORT_JTAG) ? PORT_JTAG_Msk : 0)

#else
#define JTAG_PINS(PORT) 0

#endif

      #define  IS_NOT_JTAG_PIN(PORT,  PIN)    (((PIN)  &  JTAG_PINS(PORT))  ==
0x00)
```

Дополнительную защиту от блокировки контроллера на этапе отладки программы можно сделать, добавив время простоя контроллера после включения подачи питания / перезагруки.

После перезагрузки, оперативная память контроллера, в том числе регистры периферийных устройств очищаются. Требуется инициализация параметров системы. До того как это произошло – параметры стоят по умолчанию и контроллер поддаётся перепрограммированию. В файле system_MDR32F9Qx.c описана процедура инициализации параметров `void SystemInit (void)`. Она получает управление перед функцией main. Если в начало процедуры поставить пустой цикл, это обеспечит задержку инициализации и передачи управления основной программе, в которой также может выполняться настройка периферийный устройств. При обеспечении задержки, достаточной для того, чтобы успеть нажать кнопку Load в программе Keil после перезагрузки контроллера и загрузить новую прошивку.

Таблица 2.3 – функция SystemInit, библиотека system_MDR32F9Qx.c

```
void SystemInit (void)
{
  //добавить строчку
  for (int i=0;i<=10000;i++); //задержка перед инициализацией контроллера
  /* Reset the RST clock configuration to the default reset state */
  /* Reset all clock but RST_CLK & BKP_CLC bits */
  MDR_RST_CLK->PER_CLOCK   = (uint32_t)0x8000010;
  /* Reset CPU_CLOCK bits */
  MDR_RST_CLK->CPU_CLOCK   &= (uint32_t)0x00000000;
  /* Reset PLL_CONTROL bits */
  MDR_RST_CLK->PLL_CONTROL &= (uint32_t)0x00000000;
  /* Reset HSEON and HSEBYP bits */
  MDR_RST_CLK->HS_CONTROL  &= (uint32_t)0x00000000;
  /* Reset USB_CLOCK bits */
  MDR_RST_CLK->USB_CLOCK   &= (uint32_t)0x00000000;
  /* Reset ADC_MCO_CLOCK bits */
  MDR_RST_CLK->ADC_MCO_CLOCK   &= (uint32_t)0x00000000;
  SystemCoreClockUpdate();
}
```

Пошаговая отладка на микроконтроллере

Использование интерфейсов JTAG/SW позволяет производить пошаговую отладку кода с анализом изменения состояний регистров

процессора, периферийных устройств и других видов памяти. Чтобы выполнить пошаговую отладку необходимо:

1. Убедиться, что файл – System Viewer File (Главное меню/Projects/Options For Target вкладка Target поле System Viewer File) соответствует – MDR32F9Qx.SFR (или MDR1986BE92.sfr или MDR1986BE93.sfr). (см. лр.1 «Создание проекта в Keil uVision …»);

2. Убедиться, что плата готова к работе, программатор подключен и микроконтроллер отображается в системе – см. задание 1.1 (ЛР1).

3. В главном меню выбрать Debug – Start/Stop Debug session

4. Для пошаговой отладки могу использоваться следующие горячие клавиши:

- Ctrl + F5 – Начало/приостановка операций отладки (аналогично Debug – Start/Stop Debug session)

- F5 – Запуск кода на выполнение (удобно использовать при отладке с использованием точек останова).

- F11 – Пошаговая отладка с входом в подпрограмму

- F10 – Пошаговая отладка без входа в подпрограмму (подпрограмма считается одним шагом)

- Ctrl + F11 – Выход из анализа работы подпрограммы, продолжение выполнения кода, откуда была вызвана подпрограмма.

- Ctrl + F10 – Выполнение до курсора

- F9 – Установка/Удаление точки останова

Все эти функции могут быть вызваны из Главного меню/Debug.

При наличии в программе циклов с задержкой удобно использовать не пошаговый режим отладки, а с использованием точек останова. Для этого удобно поставить Breakpoint на операцию, следующую за теми, которые надо пропустить и нажать кнопку F5. Можно воспользоваться режимом выполнения до курсора. Если не требуется анализировать выполнение операций функции, а необходимо лишь посмотреть результат её выполнения, то удобно использовать режим пошаговой отладка без входа в подпрограмму

F10. Если при отладке выполнен вход в подпрограмму анализ операций, которых не требуется можно воспользоваться операцией выхода из анализа работы подпрограммы, продолжение выполнения кода, откуда была вызвана подпрограмма – Ctrl + F11.

В режиме отладки Keil позволяет использовать следующие окна:

Registers Window – окно регистров и состояния процессора, Memory windows – окна анализа памяти, Serial Windows – окна работы с последовательным портом, Analysis Windows – различные виды анализаторов работы, Files окно программы, Disassembly Window – окно программы на assembler, Watch Windows окно состояния переменных и т.п. Доступ к ним можно получить через Главное меню/View.

Практическая часть

Задание 2.1: Настройка проекта. Подбор параметров задержки, пошаговая отладка.

Создайте проект (см. последовательность действий лр.1). Добавьте библиотеки системы тактирования MDR32F9Qx_rst_clk и портов MDR32F9Qx_port.

Опишите процедуру инициализации вывода PC0 (PF0 для 1986BE92QI) для работы светодиода. В основной подпрограмме выполните инициализацию и тактирование порта и добавьте команду включения светодиода.

Откройте файл system_MDR32F9Qx.c см. окно Project Window в левой части окна программы. Найдите описание функции SystemInit (). Добавьте пустой цикл, с произвольным числом итераций.

Выполните сборку программы. Запустите выполнение в режиме отладки (Debug – Start/Stop Debug session).

Для отладки потребуется окно регистров процессора (Registers Windows);

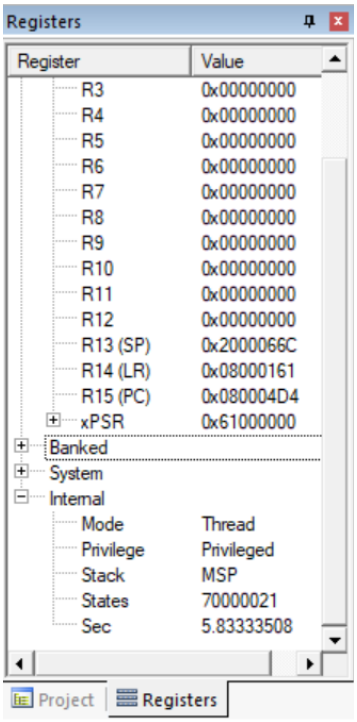


Рисунок 2.1 – Панель Registers

Используя значение параметра sec подберите параметры цикла таким образом, чтобы задержка составляла время согласно варианту задания. Повторите подбор, используя счётчик циклов (states) и значение частоты тактового генератора 8МГц. Сравните результаты.

Таблица 2.4– Варианты задания 1

Вариант	1	2	3	4	5	6	7	8	9	10
Время, с	5	7	10	12	15	17	20	3	22	25

Выполните сборку проекта. Загрузите программу в контроллер. Проанализируйте работу устройства. По включению светодиода оцените время задержки.

Задание 2.2: Обработка сигнала от кнопок. Выполнение действия по нажатию

1.1 Добавьте в программу, разработанную в задании 2.1, процедуру инициализации кнопок (кнопки подключены в соответствии с рисунком 2.2 или 2.3)

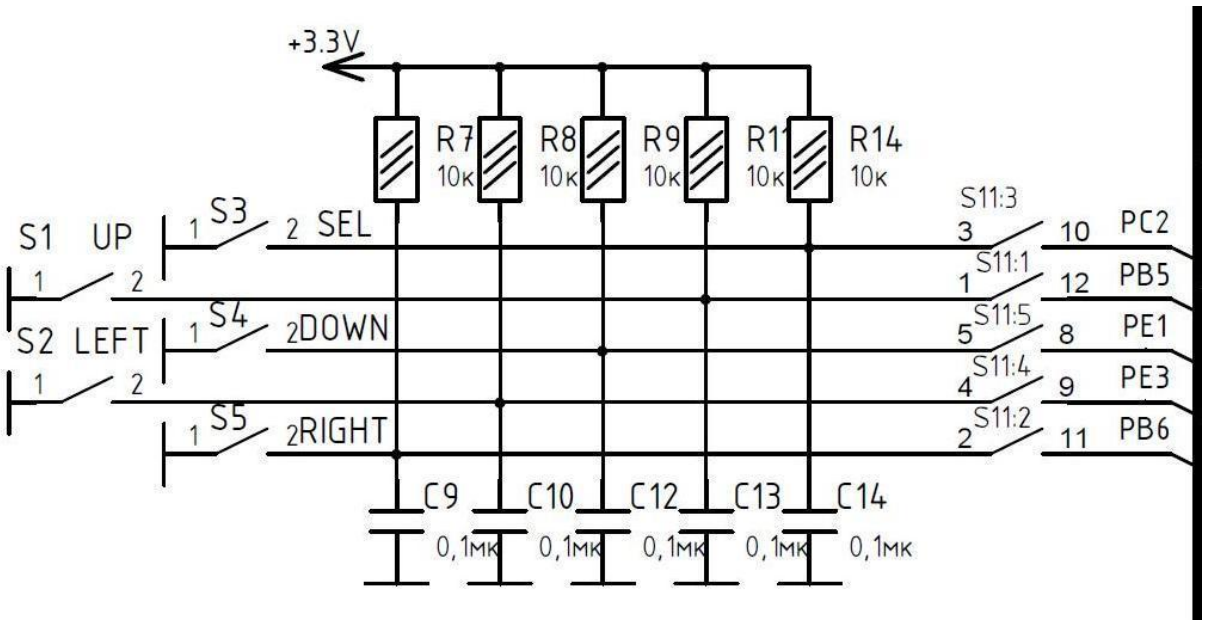


Рисунок 2.2 - Схема подключения кнопок на плате

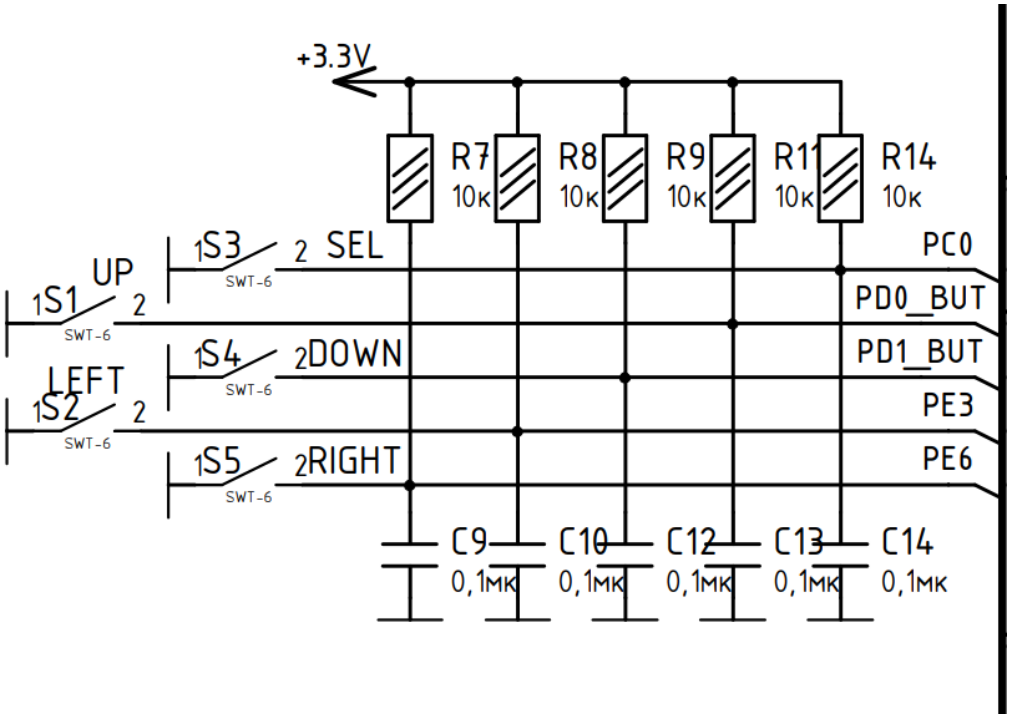


Рисунок 2.3 - Схема подключения кнопок на плате

Таблица 2.5 – Выводы с подключением кнопок на отладочной плате

Кнопка	Select	Up	Down	Left	Right
Порт (1986BE92QI)	PC2	PB5	PE1	PE3	PB6
Порт (1986BE93У)	PC0	PD0	PD1	PE3	PE6

Процедуру main исправьте таким образом, чтобы диод 1 загорался по нажатию на кнопку UP, а диод 2 загорался по нажатию на кнопку DOWN.

Учтите возможный дребезг контакта – многократные неконтролируемые замыкания и размыкания контактов за счет конструкции деталей контактной системы, на некоторое время контакты отскакивают друг от друга при соударениях, размыкая и замыкая электрическую цепь. Выполните сборку проекта. Загрузите программу в контроллер. Проанализируйте работу устройства.

Задание 2.3: Обработка сигнала от кнопок. Изменение состояния по нажатию

Измените программу из задания 2.2 таким образом, чтобы по нажатию на кнопку (в соответствии с вариантом задания см. таблицу 2.4), светодиод менял своё состояние на противоположное. Загрузите программу в контроллер. Выполните пошаговую отладку программы (Главное меню/Debug – Start/Stop Debug session). Для анализа работы портов необходимо в запущенном режиме отладки открыть Главное меню/Peripherials/System Viewer/ и выбрать группу регистров интересующих периферийных устройств. Проанализируйте изменения в регистрах процессора, модуле тактирования и модуле PortA. Зафиксируйте изменения в отчёте. Проанализируйте работу системы в рабочем режиме (без отладки) с использованием логического анализатора.

Таблица 2.6 – Варианты заданий

Вариант	1	2	3	4	5	6	7	8	9	10
Кнопка	UP	DOWN	RIGHT	LEFT	SELECT	UP	DOWN	RIGHT	LEFT	SELECT
Светодиод	1	1	1	1	1	2	2	2	2	2

Таблица 2.7 – Подключение светодиодов к GPIO на отладочных платах

Светодиод	1	2
Порт подключения (1986BE92QI)	PC0	PC1
Порт подключения (1986BE93У)	PF0	PF1

Задание 2.4: Программная генерация импульсов определенной длины

Модуляция – процесс изменения параметров одного сигнала (несущего) под воздействием другого (модулирующего, информационного).

Широко применяются: частотная, амплитудная, фазовая и широтная виды модуляций. В микропроцессорной технике широко применяется широтно-импульсная модуляция.

Широтно-импульсная модуляция (ШИМ, PWM, Pulse width modulation) – процесс изменения длительности прямоугольного импульса, при сохранении периода (частоты) и фазы несущего сигнала. На рисунке 2.3 изображен пример сигнала ШИМ. На рисунке можно отметить, что сигнал принимает два состояния 0 и 1. Периоды появления импульса одинаковы (T_1=T_2=T_3=T_4). Длина импульса от такта к такту может меняться. Отношение длительности импульса к длительности периода называется скважностью. Используя ШИМ можно передавать данные, управлять поворотом приводов, управлять мощностью (чем больше значение скважности, тем больше передаваемая мощность).

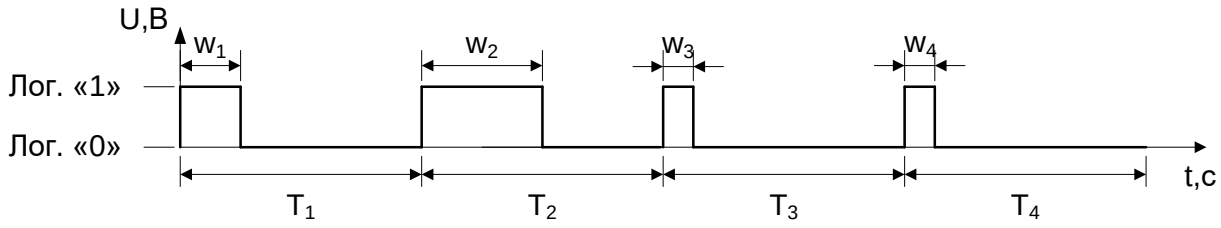


Рисунок 2.3 – ШИМ

Создайте проект (см. последовательность действий лр.1). Добавьте библиотеки системы тактирования MDR32F9Qx_rst_clk и портов MDR32F9Qx_port. Добавьте в описание функции SystemInit () в файле system_MDR32F9Qx.c пустой цикл, с числом итераций для обеспечения задержки в 5 секунд.

Добавьте файл main.c к проекту со следующим содержимым:

```
#include "MDR32F9Qx_port.h"// подключение заголовочного файла библиотеки параллельных портов
#include "MDR32F9Qx_rst_clk.h" // подключение заголовочного файла библиотеки тактового генератора
static PORT_InitTypeDef PortInit; //объявление переменной типа структура данных для инициализации параметров порта
void Delay (int del)//объявление процедуры "задержки"(пустой цикл с задаваемым числом повторений)
{
	for (int i=0; i<del; i++);//описание цикла в соответствии с синтаксисом языка C
}
```

```
void LedPinCfg (void) //описание процедуры инициализации порта
{
//инициализация порта осуществляется через структуру типа PORT_InitTypeDef
      PortInit.PORT_Pin   = (PORT_Pin_0);//выбор вывода порта, в данном случае
0 пин порта C
      PortInit.PORT_OE    = PORT_OE_OUT;//порт для вывода информации
      PortInit.PORT_FUNC  = PORT_FUNC_PORT;//порт в режиме стандартной функции
      PortInit.PORT_MODE  = PORT_MODE_DIGITAL;//цифровой режим порта
      PortInit.PORT_SPEED  =  PORT_SPEED_SLOW;//скорость  перехода  сигнала  от
одного уровня к другому - медленно
      PORT_Init(MDR_PORTA, &PortInit);//инициализация порта
}
int main (void)//
{
      RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTA,    ENABLE);//включить    тактирование
порта A
      LedPinCfg();//выполнение функции инициализации порта
      int T=1000; //период
      int w=100;  //длина импульса
while (1) {//бесконечный цикл
      MDR_PORTA->RXTX |= 0x01;
      Delay(w);
      MDR_PORTA->RXTX &=~0x01;
      Delay(T-w);
      }
}
```

Выполните сборку программы. Загрузите программу в контроллер. Выполните пошаговую отладку программы. Проанализируйте изменения в регистрах процессора, модуле тактирования и модуле PortA. Зафиксируйте изменения в отчёте в виде трассы. Проанализируйте работу системы в рабочем режиме (без отладки) с использованием логического анализатора.

Задание 2.4: Программная генерация ШИМ

Измените программу из задания 2.3 таким образом, чтобы ширину импульса можно было регулировать нажатием на кнопки в соответствии с условиями см. таблицу 2.4. Выполните сборку программы. Загрузите программу в контроллер. Проанализируйте работу устройства с помощью логического анализатора, изменяя длину импульса.

Таблица 2.4 Условия для задания 2.4

Вариант	1	2	3	4	5	6	7	8	9	10
Увеличение длины импульса										
Кнопка	right	up	up	select	up	left	select	right	right	left
Изменение	+5	+10	+2	+12	+25	+50	+75	+100	+125	+150
Уменьшение длины импульса										
Кнопка	left	select	right	right	left	right	up	up	select	up
Изменение	-5	-10	-2	-12	-25	-50	-75	-100	-125	-150

Задание 2.5: Питание светодиода через ШИМ

Измените программу из задания 2.4 таким образом, чтобы вместо PA0 сигнал ШИМ управлял светодиодом (1 или 2 из таблицы 2.5 на ваш выбор). Выполните сборку проекта. Загрузите программу в контроллер. Проанализируйте работу устройства, изменяя ширину импульса. Как влияет изменение ширины импульса на работу светодиода?

Вопросы для самоконтроля

1. Зачем нужна процедура SystemInit()? В какой момент она вызывается?

2. Зачем нужны константы USE_JTAG_A и USE_JTAG_B? Как их правильно использовать?

3. В чем достоинства, и в чем недостатки прямого обращения к регистрам порта (без использования библиотек)?

4. Опишите процедуру инициализации кнопки select (в зависимости от используемой отладочной платы) без использования библиотеки MDR32F9Qx_port.

5. Опишите процедуру обработки сигналов от кнопки select (в зависимости от используемой отладочной платы) без использования библиотеки MDR32F9Qx_port.

6. Какой режим формирования фронта SLOW, FAST или MAXFAST рационально использовать при обработке сигналов от кнопок.

7. Что называют дребезгом контактов?

8. Как бороться с дребезгом контактов?

9. Что такое ШИМ?

10. Как вы считаете, какие недостатки есть у программно-генерируемой ШИМ?

Работа 3. Модуль управления (контроллер) тактовых частот RST_CLK. Настройка тактирования процессора. Включение периферийных устройств

Цель работы: познакомиться с общими принципами тактирования узлов в микропроцессорных системах.

Задачи:

– познакомиться с понятиями тактового генератора и тактового сигнала;

– познакомиться с узлами систем тактирования цифровых устройств;

– изучить особенности модуля управления тактовыми сигналами в микроконтроллерах Cortex M3 на примере семейства Миландр 1986ВЕ9х;

– получить навыки настройки тактирования процессора МК;

– изучить основы включения тактирования периферийных устройств;

– закрепить навыки применения библиотек периферийных устройств МК на примере библиотеки модуля RST_CLK;

– закрепить навыки программирования микроконтроллеров на языке С.

Используемое оборудование:

1. Отладочная плата с микроконтроллером Миландр MDR1986BE92QI/MDR1986BE93У.
2. Комплект Программатора
   2.1 Программатор JLINK (USB-JTAG);
   2.2 Кабель USB 2.0 A – B;
   2.3 Шина (20 проводников).
3. Источник питания 5В (или дополнительный USB кабель)
4. Комплект Логического анализатора
   4.1 Логический анализатор
   4.2. Кабель USB 2.0A – mini-B
   4.3 Шина 10 проводников.

Используемая документация:

1. Техническое описание на ядро Миландр MDR1986BE9x (файл «1_Тех_описание_ядро_1986BE9X.pdf или по ссылке с сайта разработчика https://ic.milandr.ru/upload/iblock/a33/ioaf9ygfmq1lbxfhd5aad0mukg3dc93s/1986%D0%92%D0%959X.pdf)

2. Выводы отладочной платы микроконтроллера 1986BE92QI (файл «2_Выводы_платы_1986BE92QI.pdf или по ссылке* с сайта разработчика https://ic.milandr.ru/upload/iblock/8f6/8f67b8b736b3ec94edbbeb4777a9c4db.zip)

3. Выводы отладочной платы микроконтроллера 1986BE93 (файл «2_Выводы_платы_1986BE93У.pdf или по ссылке* с сайта разработчика https://ic.milandr.ru/upload/iblock/782/782c4c3b486d6f8d92995e9a44a94401.zip)

При загрузке схемотехнической документации с сайта Milandr (информация о выводах платы), загружается zip архив, который содержит 1 или 2 файла со схемой размещения выводов микроконтроллера на плате и несколько схемотехнических файлов для разводки и печати платы.

Теоретическая часть

Генератор тактовых импульсов (тактовый генератор, генератор тактовой частоты, clock generator) – устройство, предназначенное для синхронизации различных процессов в цифровых устройствах. Вырабатывает импульсы определённой частоты, которая используется как эталонная. Наиболее распространенной формой тактового сигнала является меандр – прямоугольный сигнал с рабочим циклом 50%. Рабочий цикл – отношение длительности импульса к его периоду. В вычислительной технике один такт соответствует выполнению одной атомарной (не делимой) операции. Выполнение одной команды может выполняться за один или несколько тактов. Частота тактовых импульсов определяет скорость выполнения операций.

В зависимости от особенностей тактируемого устройства используют несколько видов генераторов: LC, RC, керамический и кварцевый.

LC генератор представляет собой один или несколько последовательно включенных инверторов (или иных комбинационных схем) последовательно подключенных через LC-цепь (катушка индуктивности и конденсатор). Схема проста в реализации, имеет низкую стоимость, но невысокую стабильность.

RC генератор аналогичен по конструкции LC, но вместо индуктивности используется сопротивление (RC – резистор и конденсатор). Конструкция также имеет простую реализацию, низкую стоимость, не высокую стабильность, но меньшие по сравнению с RL размеры.

Керамический генератор использует механические колебания керамического элемента, заключенного в электроды, под действием напряжения для формирования электрических колебаний определённой частоты. От формы керамического элемента, а также от способа подключения электродов зависит частота электрических колебаний. Имеет большую по сравнению с LC и RC стоимость, но и большую стабильность.

Кварцевый генератор имеет устройство аналогичное керамическому, отличие заключается в материале, генерирующем механические колебания. Частоты также задаются формой элемента (кварца), а также расположением электродов. Кварцевый генератор имеет наибольшую стабильность, но при этом и наибольшую стоимость.

На рисунке 1.6 (отладочная плата с контроллером 1986BE92QI) кварцевый генератор отмечен под номером 15 (расположен рядом с сокетом микроконтроллера), на рисунке 1.7 (отладочная плата с контроллером 1986BE93У) отмечен под цифрой 3 (расположен на тыльной стороне платы).

В микроконтроллерах 1986BE9x управление тактовыми частотами осуществляется периферийным модулем RST_CLK. При включении питания контроллер тактируется встроенным высокочастотным генератором (HSI) на частоте 8МГц. Все периферийные устройства контроллера по умолчанию выключены, кроме модуля RST_CLK. Включение тактирования осуществляется через регистр PER_CLOCK. Схема модуля RST_CLK представлена на рисунке 3.1.

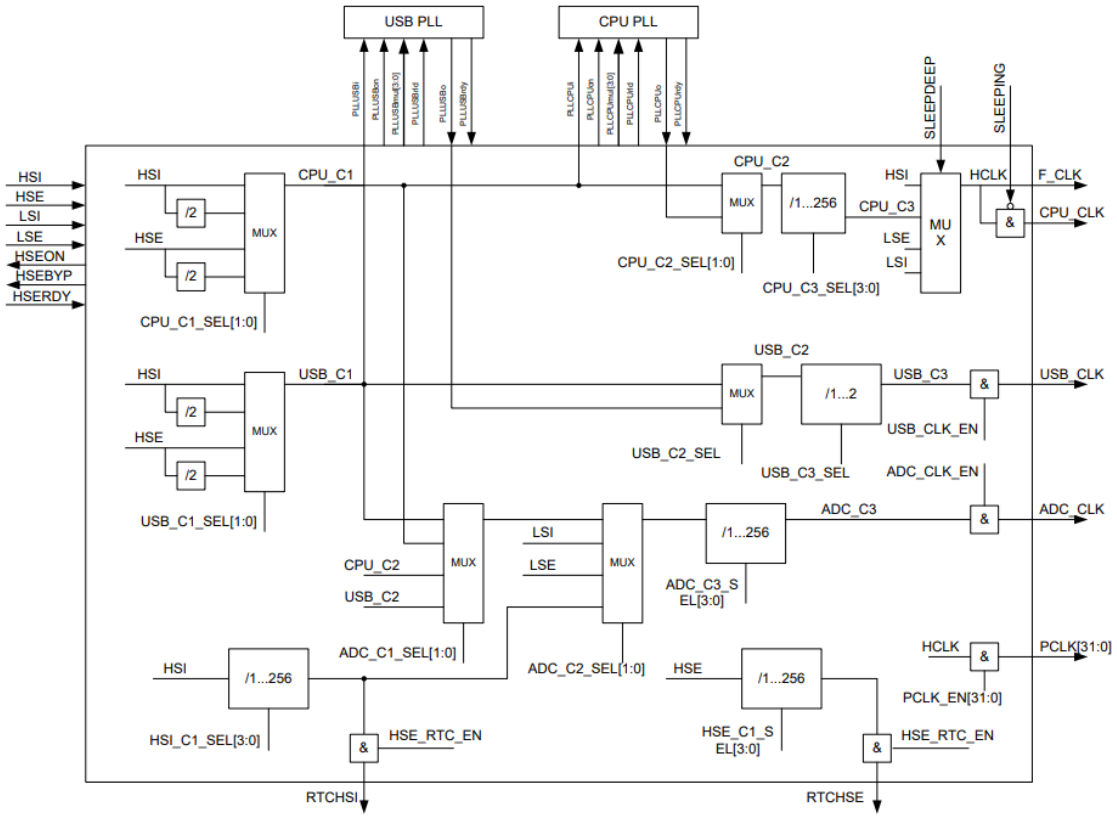


Рисунок 3.1 – Схема модуля RST_CLK

Микроконтроллер поддерживает 4 источника тактового сигнала HSI (High Speed Internal – высокочастотный внутренний генератор), HSE (High Speed External – высокочастотный внешний генератор), LSI (Low Speed Internal –низкочастотный внутренний генератор), LSE (Low Speed External – низкочастотный внешний генератор). Внутренние источники – часть микроконтроллера. Внешние – резонаторы, подключаются к выводам МК, размещаются на плате.

HSI генератор (RC) вырабатывает тактовую частоту f_{O_HSI} с типовым значением 8 МГц (диапазон 6-10МГц). Генератор автоматически запускается при появлении питания U_{CC} 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. Также генератор может быть подстроен при помощи сигнала HSITRIM в регистре BKP_REG_0F.

LSI генератор вырабатывает тактовую частоту f_{O_LSI} с типовым значением 40 кГц (диапазон 10-60 КГц). Генератор автоматически запускается при появлении питания U_{CC} 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 (время запуска после сброса). При дальнейшей работе генератор LSI может быть отключен при помощи сигнала LSION в регистре BKP_REG_0F. Может использоваться для тактирования АЦП, часов реального времени, сторожевого таймера IWDG,

HSE генератор (RC) предназначен для выработки тактовой частоты 2..16 МГц с помощью внешнего резонатора. Генератор запускается при появлении питания U_{CC} 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.

LSE генератор предназначен для выработки тактовой частоты 32 кГц с помощью внешнего резонатора. Генератор запускается при появлении питания BDU_{CC} и сигнала разрешения LSEON в регистре BKP_REG_0F. При выходе в нормальный режим работы вырабатывает сигнал LSERDY в регистре BKP_REG_0F. Также он 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 в третьем состоянии. Так как генератор LSE питается от напряжения питания BDU_{CC} и его регистр управления BKP_REG_0F расположен в батарейном домене, генератор может продолжать работать при пропадании основного питания U_{CC}. Генератор LSE используется для работы часов реального времени.

Для управления частотой тактовых сигналов используются схемы умножителей и делителей. В микроконтроллерах 1986BE9x два PLL (phase-locked loop – Фазовая автоподстройка частоты ФАПЧ) блока умножителя. Один используется для управления частотой процессора, второй для шины USB. Делителем частоты оснащены многие блоки микроконтроллера.

Управление тактированием процессора

На рисунке 3.2 представлена схема формирования тактового сигнала CPU_CLK.

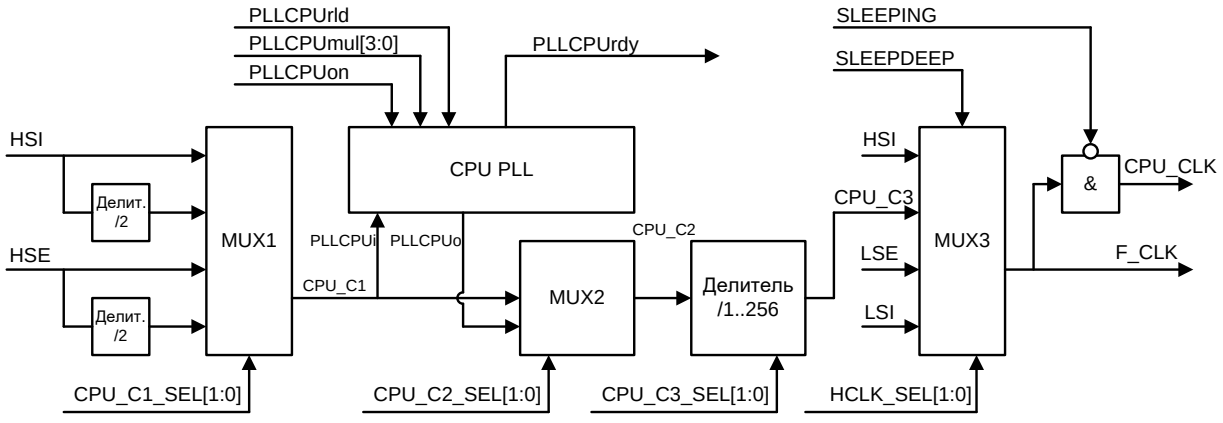


Рисунок 3.2 – схема формирования тактового сигнала

По умолчанию сигнал HCLK_SEL[1:0] = ‘b00. Мультиплексор MUX3 передаёт тактовый сигнал от HSI на CPU_CLK. Тактовая частота процессора 8МГц. Для изменения частоты можно воспользоваться или низкочастотными генераторами тактовых сигналов LSI и LSE, или переключить мультиплексор MUX3 на передачу тактового импульса с входа CPU_C3. В этом случае тактирование осуществляется от генератора HSI или HSE с возможностью снижения частоты на 2 (MUX1). При этом возможно использование PLL (умножителя) с коэффициентами от 1 до 16. Выходная частота PLL до 100МГц. Выбор сигнала от PLL осуществляется с помощью MUX2. Затем тактовый сигнал проходит через делитель с коэффициентами 1, 2, 4, 8, …256.

!При настройке тактирования процессора нужно учитывать, что его максимальная рабочая частота 80МГц.

Изменение источника тактового сигнала необходимо осуществлять по следующему алгоритму:

1. Включить необходимый источник тактового сигнала;

2. Проверить состояние выбранного источника и, если он готов, перейти к следующим действиям, иначе повторить п.2;

3. Если требуется использовать делитель и/или умножитель частоты (сигнал CPU_C3), то переход к п.4 иначе переход к п.7

4. Включить модуль PLL;

5. Указать необходимые значения умножителя и делителя;

6. Проверить состояние PLL и, если он готов, переключиться на сигнал CPUo и перейти к следующим действиям, иначе повторить п.6;

7. Переключить MUX3 в требуемый режим.

Регистры модуля RST_CLK

Управление модулем RST_CLK осуществляется через свой набор регистров: 12 регистров с базовым адресом 0x40020000. Назначение регистров смотри таблицу 3.1.

Таблица 3.1 – Назначение регистров контроллера тактовой частоты

Базовый адрес	Название	Описание (назначение)
0x40020000	MDR_RST_CLK	Контроллер тактовой частоты
Смещение		
0x00	CLOCK_STATUS	Регистр состояния блока управления тактовой частотой
0x04	PLL_CONTROL	Регистр управления блоками умножения частоты
0x08	HS_CONTROL	Регистр управления высокочастотным генератором и осциллятором**
0x0C	CPU_CLOCK	Регистр управления тактовой частотой процессорного ядра
0x10	USB_CLOCK	Регистр управления тактовой частотой контроллера USB
0x14	ADC_MCO_CLOCK	Регистр управления тактовой частотой контроллера АЦП
0x18	RTC_CLOCK	Регистр управления формированием высокочастотных тактовых сигналов блока RTC
0x1C	PER_CLOCK	Регистр управления тактовой частотой периферийных блоков
0x20	CAN_CLOCK	Регистр управления тактовой частотой CAN
0x24	TIM_CLOCK	Регистр управления тактовой частотой TIMER
0x28	UART_CLOCK	Регистр управления тактовой частотой UART
0x2C	SSP_CLOCK	Регистр управления тактовой частотой SSP (SPI)

**Обращение к регистрам контроллера тактовых сигналов на С можно выполнить следующим образом: MDR_RST_CLK-> «имя регистра». Например: MDR_RST_CLK->CPU_CLOCK|=0x100 – установка 1 в 8ой бит регистра без изменения других.*

***Осциллятор (лат. oscillo — качаюсь) — система, совершающая колебания (показатели системы периодически повторяются во времени).*

Назначение бит регистров тактирования процессора и включения периферийных устройств представлено ниже.

(Регистры USB_CLOCK, ADC_MCO_CLOCK, RTC_CLOCK, CAN_CLOCK, TIM_CLOCK, UART_CLOCK, SSP_CLOCK в данной работе не рассматриваются).

Регистр CLOCK_STATUS

Номер бита	31..3	2	1	0
Режим доступа*	-	RO	RO	RO
Состояние после включения (сброса)	-	0	0	0
Назначение	-	**HSE RDY**	**PLL CPU RDY**	**PLL USB RDY**

*RO – read only (только чтение), WO – write only (только запись), R/W – Read/Write (чтение и запись)

HSE RDY – Флаг выхода в рабочий режим осциллятора HSE: 0 – осциллятор не запущен или не стабилен; 1 – осциллятор запущен и стабилен.

PLL CPU RDY – Флаг выхода в рабочий режим CPU PLL: 0 – PLL не запущена или не стабильна; 1 – PLL запущена и стабильна

PLL USB RDY – Флаг выхода в рабочий режим USB PLL: 0 – PLL не запущена или не стабильна; 1 – PLL запущена и стабильна

Регистр PLL_CONTROL

Номер бита	31..12	11..8	7..4	3	2	1	0
Режим доступа	-	R/W	R/W	R/W	R/W	R/W	R/W
Состояние после включения (сброса)	-	0	0	0	0	0	0
Назначение	-	**PLL CPU MUL[3:0]**	**PLL USB MUL[3:0]**	**PLL USB RLD**	**PLL CPU ON**	**PLL USB RLD**	**PLL USB ON**

PLL CPU MUL[3:0] – Коэффициент умножения CPU PLL: PLLCPUo = PLLCPUi x (PLLCPUMUL+1)
PLL USB MUL[3:0] – Коэффициент умножения USB PLL: PLLUSBo = PLLUSBi x (PLLUSBMUL+1)
PLL CPU PLD – бит перезапуска PLL CPU. При смене коэффициента умножения в рабочем режиме необходимо задать равным 1
PLL CPU ON – Бит включения PLL CPU: 0 – PLL выключена; 1 – PLL включена
PLL USB PLD – Бит перезапуска PLL USB. При смене коэффициента умножения в рабочем режиме необходимо задать равным 1
PLL USB ON – Бит включения PLL USB: 0 – PLL выключена; 1 – PLL включена

Регистр HS_CONTROL

Номер бита	31..2	1	0
Режим доступа	-	R/W	R/W
Состояние после включения (сброса)	-	0	0
Назначение	-	**HSE BYP**	**HSE ON**

SE ON – Бит управления HSE осциллятором: 0 – выключен; 1 – включен
HSE BYP – Бит управления HSE осциллятором: 0 – режим осциллятора; 1 – режим внешнего генератора

Регистр CPU_CLOCK

Номер бита	31..10	9..8	7..4	3	2	1..0
Режим доступа	-	R/W	R/W	-	R/W	R/W
Состояние после включения (сброса)	-	0	0	-	0	0
Назначение	-	**HCLK SEL[1:0]**	**CPU C3 SEL[3:0]**	-	**CPU C2 SEL**	**CPU C1 SEL[1:0]**

CPU C1 SEL [1:0] – Биты выбора источника для CPU_C1:

00 – HIS
01 – HSI/2
10 – HSE
11 – HSE/2

CPU C2 SEL – Биты выбора источника для CPU_C2: 0 – CPU_C1 1 – PLLCPUo
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
1011 - CPU_C3 = CPU_C2 / 16
1100 - CPU_C3 = CPU_C2 / 32
1101 - CPU_C3 = CPU_C2 / 64
1110 - CPU_C3 = CPU_C2 / 128
1111 - CPU_C3 = CPU_C2 / 256

HCLK SEL [1:0] – Биты выбора источника для HCLK:

00 – HSI
01 – CPU_C3
10 – LSE
11 – LSI

Регистр PER_CLOCK

Номер бита	31..0
Режим доступа	R/W
Состояние после включения (сброса)	
Назначение	**PCLK EN [31:0]**

Биты разрешения тактирования периферийных блоков: 0 – запрещено; 1 – разрешено.

PCLK[0] – CAN1 PCLK[1] – CAN2 PCLK[2] – USB PCLK[3] – EEPROM_CNTRL PCLK[4] – RST_CLK 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

Часть параметров модуля RST CLK задаётся через регистр REG_0F модуля BKP – блока батарейного домена (Базовый адрес модуля BKP – 0x400D_8000, смещение регистра 0x3C).

Регистр MDR_BKP->REG_0F

Номер бита	31	30	29:24	23	22	21	20:16
Режим доступа	R/W	R/W	RO	R/W	R/W	RO	R/W
Состояние после включения (сброса)	0	0	0	1	1	1	0
Назначение	**RTC RESET**	**STANDBY**	**HIS TRIM [7:0]**	**HSI RDY**	**HSI ON**	**LSI RDY**	**LSI TRIM[4:0]**

Номер бита	15	14	13	12:5	4	3:2	1	0
Режим доступа	R/W	-	RO	R/W	R/W	R/W	R/W	R/W
Состояние после включения (сброса)	1	-	0	0	0	0	0	0
Назначение	**LSE ON**	-	**LSE RDY**	**CALL[7:0]**	**RTC EN**	**RTC SEL [1:0]**	**LSE BYP**	**LSE ON**

RTC RESET Сброс часов реального времени: 0 – не сбрасываются; 1 – сбрасываются
STANDBY Режим отключения регулятора DUCC: 0 – регулятор включен и выдает напряжение; 1 – регулятор выключен Триггер сбрасывается флагом ALRF или по низкому уровню на выводе WAKEUP
HIS TRIM [7:0] Коэффициент подстройки частоты генератора HSI (см. рисунок 3.3а).
HIS RDY Флаг выхода генератора HSI в рабочий режим: 0 – генератор не запущен или не вышел в режим; 1 – генератор работает в рабочем режиме
HSI ON Бит управления генератором HSI: 0 – генератор выключен; 1 – генератор включен
LSI RDY Флаг выхода генератора LSI в рабочий режим: 0 – генератор не запущен или не вышел в режим; 1 – генератор находится в рабочем режиме
LSI TRIM [4:0] Коэффициент подстройки частоты генератора LSI. (см. рисунок 3.3б)
LSE ON Бит управления генератором LSI: 0 – LSI выключен; 1 – LSI включен
LSE RDY Флаг выхода генератора LSE в рабочий режим: 0 – генератор не запущен или не вышел в режим; 1 – генератор работает в рабочем режиме
CAL [7:0] Коэффициент подстройки тактовой частоты часов реального времени, из каждых 220 тактов будет замаскировано CAL тактов: 00000000 – 0 тактов; 00000001 – 1 такт; ….; 11111111 – 256 тактов. Таким образом, при частоте генератора 32768.00000 Гц, при CAL = 0 тактов – частота = 32768.00000 Гц; при CAL = 1 такт – частота = 32767,96875 Гц; …; при CAL = 256 тактов частота = 32760.00000 Гц.

RTC EN Бит разрешения работы часов реального времени:

0 – работа запрещена; 1 – работа разрешена

RTC SEL [1:0] Биты выбора источника тактовой синхронизации часов реального времени:

00 – LSI	10 – HSIRTC (формируется в блоке CLKRST)
01 – LSE	11 – HSERTC (формируется в блоке CLKRST)

LSE BYP Бит управления генератором LSE:

0 – режим осциллятора; 1 – режим работы на проход (внешний генератор)

LSE ON Флаг выхода генератора LSI в рабочий режим: 0 – генератор не запущен или не вышел в режим; 1 – генератор работает в рабочем режиме

Описание функций и параметров библиотеки MDR32F9Qx_rst_clk

1. Переключение на тактовый генератор HSI и полный сброс параметров контроллера (значений делителей, умножителей частоты, параметров мультиплексоров)

```
void RST_CLK_WarmDeInit(void);
```

2. Полный сброс параметров контроллера тактовых частот (включает в себя вызов функции `RST_CLK_WarmDeInit`, со сбросом параметров модуля батарейного домена BKP (BKP –предназначен для обеспечения функций часов реального времени и сохранения некоторого набора пользовательских данных при отключении основного источника питания, в том числе параметров работы контроллера тактовых частот).

```
void RST_CLK_DeInit(void);
```

Источники тактовых сигналов

3. Конфигурация генератора HSE

```
void RST_CLK_HSEconfig(uint32_t RST_CLK_HSE);
```

4. Запрос состояния HSE

```
ErrorStatus RST_CLK_HSEstatus(void);
```

5. Конфигурация генератора LSE

```
void RST_CLK_LSEconfig(uint32_t RST_CLK_LSE);
```

6. Запрос состояния LSE

```
ErrorStatus RST_CLK_LSEstatus(void);
```

7. Запрос состояния HSI

```
void RST_CLK_HSIcmd(FunctionalState NewState);
```

Подстройка внутреннего высокочастотного источника тактовой частоты (HSI)

```
void RST_CLK_HSIadjust(uint32_t HSItrimValue);
```

8. Запрос состояния HSI

```
ErrorStatus RST_CLK_HSIstatus(void);
```

9. Запрос состояния LSI

```
void RST_CLK_LSIcmd(FunctionalState NewState);
```

Подстройка внутреннего низкочастотного источника тактовой частоты (LSI)

```
void RST_CLK_LSIadjust(uint32_t LSItrimValue);
```

10. Запрос состояния LSI

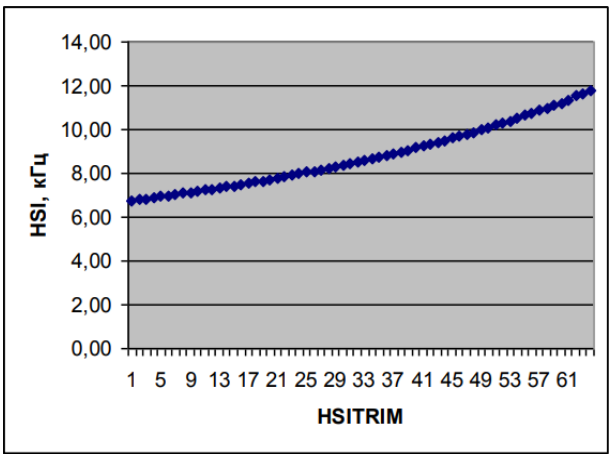
```
ErrorStatus RST_CLK_LSIstatus(void);
```

Параметр `RST_CLK_HSE`	Назначение
RST_CLK_HSE_OFF	HSE – выключен
RST_CLK_HSE_ON	HSE – включен
RST_CLK_HSE_Bypass	HSE – проброс через RST_CLK

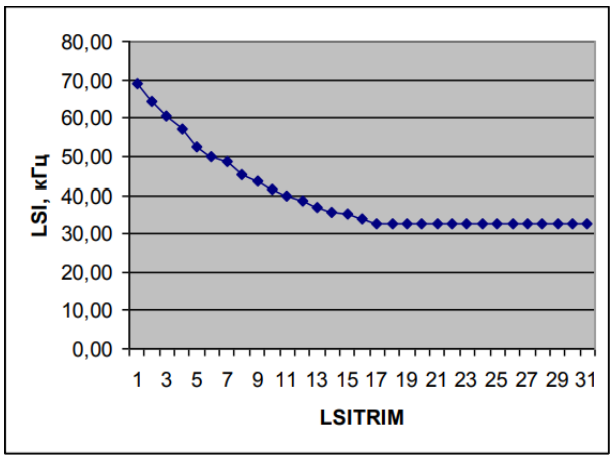
Параметр `RST_CLK_LSE`	Назначение
RST_CLK_LSE_OFF	LSE – выключен
RST_CLK_LSE_ON	LSE – включен
RST_CLK_LSE_Bypass	HSE – проброс через RST_CLK

Параметры типа FunctionalState во всех функциях библиотек МК могут принимать значения ENABLE – включить, DISABLE – выключить. (библиотека MDR32Fx.h). Тип ErrorStatus может принимать значения SUCCESS – успешно (готов) и ERROR – ошибка (библиотека MDR32Fx.h). Переменные типа FlagStatus могут принимать значения SET – установлен (1), RESET – сброшен (0) (библиотека MDR32Fx.h).

Параметры HSItrimValue и LSItrimValue определяются в соответствии с рисунком 3.3:



а) подстройка HSI



б) подстройка LSI

Рисунок – 3.3 Параметры подстройки внутренних тактовых генераторов

Настройка PLL

11. Выбор источника тактового сигнала и коэффициента умножения

```
void RST_CLK_CPU_PLLconfig(uint32_t RST_CLK_CPU_PLLsource, uint32_t RST_CLK_CPU_PLLmul);
```

12. Подключение CPU PLL к формированию тактового сигнала процессора (изменение режима MUX2)

```
void RST_CLK_CPU_PLLuse(FunctionalState UsePLL);
```

13. Включение/выключение модуля CPU PLL (сигнал PLLCPUon)

```
void RST_CLK_CPU_PLLcmd(FunctionalState NewState);
```

14. Запрос состояния CPU PLL

```
ErrorStatus RST_CLK_CPU_PLLstatus(void);
```

Параметр `RST_CLK_CPU_PLLmul`	Значение умножителя
RST_CLK_USB_PLLmul1	1
RST_CLK_USB_PLLmul2	2
RST_CLK_USB_PLLmul3	3
RST_CLK_USB_PLLmul4	4
RST_CLK_USB_PLLmul5	5
RST_CLK_USB_PLLmul6	6
RST_CLK_USB_PLLmul7	7
RST_CLK_USB_PLLmul8	8
RST_CLK_USB_PLLmul9	9
RST_CLK_USB_PLLmul10	10
RST_CLK_USB_PLLmul11	11
RST_CLK_USB_PLLmul12	12
RST_CLK_USB_PLLmul13	13
RST_CLK_USB_PLLmul14	14
RST_CLK_USB_PLLmul15	15
RST_CLK_USB_PLLmul16	16

Параметр `RST_CLK_CPU_PLLsource`	Назначение
RST_CLK_CPU_PLLsrcHSIdiv1	HSI
RST_CLK_CPU_PLLsrcHSIdiv2	HSI/2
RST_CLK_CPU_PLLsrcHSEdiv1	HSE
RST_CLK_CPU_PLLsrcHSEdiv2	HSE/2

Тактирование процессора

15. Настройка предделителя для тактового сигнала CPU

```
void RST_CLK_CPUclkPrescaler(uint32_t CPUclkDivValue);
```

16. Выбор источника тактового сигнала

```
void RST_CLK_CPUclkSelection(uint32_t CPU_CLK);
```

Настройка предделителей

17. Настройка предделителя для сигнала от тактового генератора HSI

```
void RST_CLK_HSIclkPrescaler(uint32_t HSIclkDivValue);
```

18. Включение/отключение генератора тактовых сигналов HSI

```
void RST_CLK_RTC_HSIclkEnable(FunctionalState NewState);
```

19. Настройка предделителя для сигнала от тактового генератора HSE

```
void RST_CLK_HSEclkPrescaler(uint32_t HSEclkDivValue);
```

20. Включение/отключение генератора тактовых сигналов HSE

```
void RST_CLK_RTC_HSEclkEnable(FunctionalState NewState);
```

Параметр CPUclkDivValue	Параметр HSIclkDivValue	Параметр HSEclkDivValue	Деление частоты на
RST_CLK_CPUclkDIV1	RST_CLK_HSIclkDIV1	RST_CLK_HSEclkDIV1	1
RST_CLK_CPUclkDIV2	RST_CLK_HSIclkDIV2	RST_CLK_HSEclkDIV2	2
RST_CLK_CPUclkDIV4	RST_CLK_HSIclkDIV4	RST_CLK_HSEclkDIV4	4
RST_CLK_CPUclkDIV8	RST_CLK_HSIclkDIV8	RST_CLK_HSEclkDIV8	8
RST_CLK_CPUclkDIV16	RST_CLK_HSIclkDIV16	RST_CLK_HSEclkDIV16	16
RST_CLK_CPUclkDIV32	RST_CLK_HSIclkDIV32	RST_CLK_HSEclkDIV32	32
RST_CLK_CPUclkDIV64	RST_CLK_HSIclkDIV64	RST_CLK_HSEclkDIV64	64
RST_CLK_CPUclkDIV128	RST_CLK_HSIclkDIV128	RST_CLK_HSEclkDIV128	128
RST_CLK_CPUclkDIV256	RST_CLK_HSIclkDIV256	RST_CLK_HSEclkDIV256	256

Параметр CPU_CLK	Назначение
RST_CLK_CPUclkHSI	Тактирование CPU от HSI
RST_CLK_CPUclkCPU_C3	Тактирование CPU от HSE или LSE через делитель и PLL
RST_CLK_CPUclkLSE	Тактирование CPU от LSE
RST_CLK_CPUclkLSI	Тактирование CPU от LSI

Тактирование периферийных устройств

21. Включение/отключение тактирования периферийных устройств

```
void RST_CLK_PCLKcmd(uint32_t RST_CLK_PCLK, FunctionalState NewState);
```

Константа (параметр RST_CLK_PCLK)	Тактируемое устройство
RST_CLK_PCLK_CAN1	Последовательный интерфейс CAN1
RST_CLK_PCLK_CAN2	Последовательный интерфейс CAN2
RST_CLK_PCLK_USB	Последовательный интерфейс USB
RST_CLK_PCLK_EEPROM	Энергонезависимая память EEPROM
RST_CLK_PCLK_RST_CLK	Контроллер тактовых частот RST_CLK (включен по умолчанию)
RST_CLK_PCLK_DMA	Контроллер прямого доступа к памяти DMA
RST_CLK_PCLK_UART1	Последовательный интерфейс UART1
RST_CLK_PCLK_UART2	Последовательный интерфейс UART2
RST_CLK_PCLK_SSP1	Последовательный интерфейс SSP1(SPI1)
RST_CLK_PCLK_09	Зарезервировано PCLK_BIT(0x40048000)
RST_CLK_PCLK_I2C	Последовательный интерфейс I2C
RST_CLK_PCLK_POWER	Детектор напряжения питания POWER
RST_CLK_PCLK_WWDG	Оконный сторожевой таймер Window watchdog WWDG
RST_CLK_PCLK_IWDG	Независимый сторожевой таймер Independent watchdog IWDG
RST_CLK_PCLK_TIMER1	Таймер счётчик TIMER1
RST_CLK_PCLK_TIMER2	Таймер счётчик TIMER2
RST_CLK_PCLK_TIMER3	Таймер счётчик TIMER3
RST_CLK_PCLK_ADC	Аналогово-цифровой преобразователь ADC
RST_CLK_PCLK_DAC	Цифро-аналоговый преобразователь (ЦАП) DAC
RST_CLK_PCLK_COMP	Аналоговый компаратор COMP
RST_CLK_PCLK_SSP2	Последовательный интерфейс SSP2(SPI2)
RST_CLK_PCLK_PORTA	GPIO PORTA
RST_CLK_PCLK_PORTB	GPIO PORTB
RST_CLK_PCLK_PORTC	GPIO PORTC
RST_CLK_PCLK_PORTD	GPIO PORTD
RST_CLK_PCLK_PORTE	GPIO PORTE
RST_CLK_PCLK_26	Зарезервировано PCLK_BIT(0x400D0000)
RST_CLK_PCLK_BKP	BKP
RST_CLK_PCLK_28	Зарезервировано PCLK_BIT(0x400E0000)
RST_CLK_PCLK_PORTF	PORTF
RST_CLK_PCLK_EBC	Внешняя системная шина EBC
RST_CLK_PCLK_31	Зарезервировано PCLK_BIT(0x400F8000)

22. Запрос состояния устройства

```
FlagStatus RST_CLK_GetFlagStatus(uint32_t RST_CLK_FLAG);
```

Параметр RST_CLK_FLAG	Назначение
RST_CLK_FLAG_HSIRDY	Готовность генератора HSI
RST_CLK_FLAG_LSIRDY	Готовность генератора HSE
RST_CLK_FLAG_HSERDY	Готовность генератора LSI
RST_CLK_FLAG_LSERDY	Готовность генератора LSE
RST_CLK_FLAG_PLLCPURDY	Готовность PLL CPU
RST_CLK_FLAG_PLLUSBRDY	Готовность PLL USB
RST_CLK_FLAG_PLLDSPRDY	Готовность PLL SPR (нет в 1986BE9x)

23. Получение значения частот контроллера

```
void RST_CLK_GetClocksFreq(RST_CLK_FreqTypeDef* RST_CLK_Clocks);
```

RST_CLK_FreqTypeDef – структура, состоящая из следующих параметров:
CPU_CLK_Frequency – частота тактирования процессора МК
USB_CLK_Frequency – частота тактирования контроллера USB
ADC_CLK_Frequency – частота тактирования АЦП
RTCHSI_Frequency – частота часов реального времени при тактировании от HSI
RTCHSE_Frequency – частота часов реального времени при тактировании от HSE

Процедуры библиотеки, не используемые в работе

Процедуры настройки тактирования АЦП

```
void RST_CLK_ADCclkSelection(uint32_t ADC_CLK);
void RST_CLK_ADCclkPrescaler(uint32_t ADCclkDivValue);
void RST_CLK_ADCclkEnable(FunctionalState NewState);
```

Процедуры настройки тактирования контроллера USB

```
void RST_CLK_USB_PLLconfig(uint32_t RST_CLK_USB_PLLsource, uint32_t
RST_CLK_USB_PLLmul);
void RST_CLK_USB_PLLuse(FunctionalState UsePLL);
void RST_CLK_USB_PLLcmd(FunctionalState NewState);
ErrorStatus RST_CLK_USB_PLLstatus(void);
void RST_CLK_USBclkPrescaler(FunctionalState NewState);
void RST_CLK_USBclkEnable(FunctionalState NewState);
```

Также в библиотеке описаны функции тактирования устройств, которых нет в семействе МК 1986BE9x:

1. Частота тактирования аппаратного аудиокодека AUCCLK (МК 1986BE3 и 1901ВЦ1Т)
2. Настройка второго внешнего генератора HSE2(МК 1986BE3 и 1901ВЦ1Т)
3. Частота цифрового сигнального процессора (DSP) (1901ВЦ1Т)

Замечание: В файле MDR32F9Qx_rst_clk.c описаны константы базовых значений частот тактовых генераторов HSE, HSI, LSE, LSI в герцах. Их значения участвуют в вычислении параметров тактовых сигналов периферийных устройств при использовании библиотек CMSIS. При настройке проекта необходимо определить, какие частоты выдают внешние тактовые генераторы (HSE, LSE), и указать их вместо значений, выделенных зелёным.

#ifndef HSI_Value /* Typical Value of the HSI in Hz */ #define HSI_Value ((uint32_t)8000000) #endif /* HSI_Value */	#ifndef HSE_Value /* Typical Value of the HSE in Hz */ #define HSE_Value ((uint32_t)8000000) #endif /* HSE_Value */
#ifndef LSI_Value /* Typical Value of the LSI in Hz */ #define LSI_Value ((uint32_t)40000) #endif /* LSI_Value */	#ifndef LSE_Value /* Typical Value of the LSE in Hz */ #define LSE_Value ((uint32_t)32768) #endif /* LSE_Value */

Практическая часть

Задание 3.1. Переключение на внешний генератор HSE.

Создайте проект согласно описанному в первой работе алгоритму. Добавьте следующий программный код в файл main.c.

```
#include "MDR32F9Qx_port.h"         //подключение библиотеки работы с портами
#include "MDR32F9Qx_rst_clk.h"      //подключение    библиотеки    контроллера
тактовых сигналов
static PORT_InitTypeDef PortInit;   //стуктура инициализации порта
void Delay(int del)                 //процедура задржки del - кол-во циклов
{
      for (int i=0; i<del; i++);    //1 цикл – n тактов процессора
}
void LedConfig (void)               //Процедура инициализации
{
      PortInit.PORT_Pin = PORT_Pin_0;           //вывод 0
      PortInit.PORT_FUNC = PORT_FUNC_PORT;      //режим порт
      PortInit.PORT_MODE = PORT_MODE_DIGITAL;   //цифровой
      PortInit.PORT_OE = PORT_OE_OUT;           //сигнал на вывод
      PortInit.PORT_SPEED = PORT_SPEED_SLOW;    //медленный фронт
      PORT_Init(MDR_PORTC, &PortInit);          //инициализация параметров
}
void CPUCLKConfig(void) //настройка тактирования ЦПУ
{
      RST_CLK_HSEconfig(RST_CLK_HSE_ON);              //включение      внешнего
высокочастотного генератора
      while (RST_CLK_HSEstatus()!=SUCCESS);           //запрос       состояния,
только если готов, то переход к следующим действиям
      RST_CLK_CPUclkSelection(RST_CLK_CPUclkCPU_C3);//переключение на CPU_C3
}
int main (void)               //основная процедура
{
RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTC, ENABLE);//разрешение тактирования PORTC
      LedConfig();                          //инициализация вывода светодиода
      PORT_SetBits(MDR_PORTC, 0);           //включение светодиода
      Delay(10000);                         //задержка
      CPUCLKConfig();                       //переключение тактового генератора
      PORT_ResetBits(MDR_PORTC,0);          //выключение светодиода
      while(1)                              //бесконечный цикл
      {
            Delay(10000);                   //задержка
            PORT_SetBits(MDR_PORTC, 0);     //включение светодиода
            Delay(10000);                   //задержка
            PORT_ResetBits(MDR_PORTC, 0); //выключение светодиода
      }
}
```

Выберите используемый порт JTAG файла MDR32F9Qx_port.h, Добавьте процедуру задержки в функцию SystemInit() файла system_MDR32F9Qx.c (см. работу 2).

Уточните, какую частоту формирует генератор HSE (расположение см. п. 15 рисунок 1.6 для 1986BE92QI, или п. 3* рисунок 1.7 для 1986BE93У), его значение указано на корпусе генератора. Проверьте, соответствует ли значение константы HSE_Value (MDR32F9Qx_rst_clk.c) частоте генератора.

Изобразите схему алгоритма процедуры main(), включая действия процедуры CPUCLKConfig. Выполните сборку программы. Запустите выполнение программы в режиме Debug – Главное меню/Debug/Start Stop Debug Session (для отладки можно не подключать контроллер, но тогда необходимо переключить режим отладки Главное меню/Project/Options for Target вкладка Debug флажок USE с JLink/J-Trace Cortex на симулятор). Используя средства отладки (точки останова, выполнение кода до точки и пошаговое выполнение), определите, за сколько тактов выполняется 1 итерация цикла Delay. Определите, сколько времени выполняется одна итерация при заданной частоте. Загрузите программу в контроллер. Проанализируйте работу системы.

Добавьте код для уменьшения частоты источника в 2 раза. Выполните сборку программы, загрузите в контроллер, проанализируйте работу системы.

Задание 3.2. Анализ изменения частоты работы с использованием анализатора.

Измените программу из задания 3.1 таким образом, чтобы на выводы порта A выводилось значение от счётчика (аналогично заданию 1.3). Измените частоту и источник тактового сигнала в соответствии с вариантом задания см.таблицу 3.2.

Таблица 3.2 – Варианты для задания 3.2

Вариант	1	2	3	4	5	6	7	8	9	10
Источник	HSE	HSI	HSE	HSI	HSE	HSI	HSE	HSI	HSE	HSI
Значение делителя	/2	/2	/8	/8	/32	/32	/128	/128	/256	/256

Выполните сборку программы, загрузите в контроллер, проанализируйте работу системы.

Задание 3.3. Подключение модуля CPU PLL

Измените процедуру CPUCLKConfig() согласно следующему примеру:

```
//включение внешнего высокочастотного генератора
RST_CLK_HSEconfig(RST_CLK_HSE_ON);
//запрос состояния, только если готов, то переход к следующим действиям
while (RST_CLK_HSEstatus()!=SUCCESS);
//запрос состояния,
RST_CLK_CPU_PLLconfig(RST_CLK_CPU_PLLsrcHSEdiv1, RST_CLK_CPU_PLLmul8);
//запрос состояния,
RST_CLK_CPU_PLLcmd();
//запрос состояния,
while (RST_CLK_HSEstatus()!=SUCCESS);
//запрос состояния,
RST_CLK_CPU_PLLuse(ENABLE);
);//переключение на тактовый сигнал CPU_C3
RST_CLK_CPUclkSelection(RST_CLK_CPUclkCPU_C3
```

Изобразите схему алгоритма процедуры main(), включая действия процедуры CPUCLKConfig. Выполните сборку программы. Загрузите программу в контроллер. Проанализируйте изменение работы системы.

Задание 3.4. Управление частотой.

Измените программу из задания 3.3 таким образом, чтобы можно было изменять частоту процессора нажатием на кнопки (см. таблицу 3.3). Выполните сборку программы. Загрузите программу в микроконтроллер. Проанализируйте работу системы.

Таблица 3.3 – Варианты для задания 3.4

Кнопка	RIGHT	LEFT	UP	DOWN
Действие	PLL+	PLL-	DIV-	DIV+

Задание 3.5. Настройка модуля RST_CLK прямым обращением к регистрам.

Перепишите подпрограмму инициализации из задания 3.3, не используя библиотечные функции. Все операции с контроллером тактовых частот выполнить прямым обращением к регистрам. Выполните сборку программы. Перед загрузкой программы в микроконтроллер покажите код преподавателю. После загрузки программы проанализируйте работу системы и сравните её с заданием 3.3.

Содержание отчета

Общие требования. В отчёте для каждого задания должны быть представлены: формулировка задания (основная задача); код программы с подробными комментариями (даже если программа представлена в качестве примера); подробное описание реакции системы при выполнении на стенде; фрагмент трассы выполнения при работе в симуляторе; при использовании логического анализатора должны быть представлены снимки экрана с полученными временными диаграммами; представьте выводы по заданию.

Требования к представлению отдельных заданий. Для первого задания должна быть схема алгоритма работы программы и описание методики расчёта количества тактов и времени (при заданной частоте) на выполнение одной итерации цикла процедуры Delay, а также комментарий по уточнению константы HSE_Value. Для третьего задания должна быть представлена схема алгоритма.

Требования к оформлению. Общая структура отчета должна содержать следующие пункты: цель работы; выполнение заданий; общие выводы по работе, сформулированные по результатам выполнения заданий (а не только из цели работы). Текст должен быть оформлен в одном стиле (текст отчёта и код программы могут иметь разные стили), рисунки и таблицы должны быть подписаны и иметь ссылки в тексте, страницы пронумерованы.

Вопросы для самоконтроля

1. Что называют тактовым генератором?

2. Какие функции выполняет тактовый генератор?

3. Какие виды генераторов выделяют? В чём их отличия?

4. Что формирует тактовый сигнал при включении (перезагрузке) контроллера?

5. Какие модули микроконтроллера тактируются при включении (перезагрузке) контроллера?

6. Как правильно переходить от одного тактового генератора к другому?

7. Каким образом можно изменить частоту процессора МК?

8. От каких источников (генераторов) можно изменить частоту тактирования процессора МК и на какие значение?

9. Поясните назначение и порядок включения модуля PLL. Сколько таких модулей у МК 1986ВЕ9х.

10. Какое функциональное назначение имеет регистр PER_CLOCK.