

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра электроэнергетики и автоматики

Утверждено
научно-методическим советом
университета

СИЛОВАЯ ЭЛЕКТРОНИКА

Методические указания к выполнению лабораторных работ для
студентов направления, бакалавриата 13.03.02 – электроэнергетика и
электротехника, профиль "Электропривод и автоматика"

Часть 2

Белгород 2017

УДК 621.314(07)

ББК 31.264.5я7

С 36

Составители: канд. техн. наук, доцент, Ю.В. Скурятин,
ассистент Д.И. Прокопишин,

Методические указания к выполнению лабораторных работ по дисциплине «Силовая электроника» для студентов направления бакалавриата 13.03.02 – электроэнергетика и электротехника, профиль "Электропривод и автоматика" / Сост. Ю.В. Скурятин, Д.И. Прокопишин. – Белгород, БГТУ им. В.Г.Шухова, 2017. – Ч.2. – 107 с.

В методических указаниях изложены порядок и методика проведения лабораторных работ по дисциплине «Силовая электроника» раздел «Системы управления преобразователей». Лабораторные работы выполняются на базе демонстрационно-отладочного комплекта с 32-разрядным микроконтроллером семейства **1986VE9X разработки и производства компании "Миландр"**. В качестве среды разработки предлагается использование пакета Keil μ Vision.

Издание предназначено для студентов направления подготовки 13.03.02 – электроэнергетика и электротехника, профиль "Электропривод и автоматика" всех форм обучения.

Методические указания публикуются в авторской редакции.

УДК 621.314(07)

ББК 31.264.5я7

- © Ю.В. Скурятин, 2017
- © Белгородский государственный технологический университет (БГТУ) им. В.Г.Шухова, 2017
- © АО «ПКК Миландр», 2017

СОДЕРЖАНИЕ

ОСНОВНЫЕ ПОЛОЖЕНИЯ	4
ПРАВИЛА ТЕХНИКИ БЕЗОПАСНОСТИ	5
ОПИСАНИЕ ЛАБОРАТОРНОГО СТЕНДА	6
Лабораторная работа 1. ИНТЕГРИРОВАННАЯ СРЕДА	
РАЗРАБОТКИ Keil μ Vision	10
Лабораторная работа 2. ОРГАНИЗАЦИЯ ВВОДА/ВЫВОДА	
ИНФОРМАЦИИ С ПОРТОВ МИКРОКОНТРОЛЛЕРА	
K1986VE92QI	25
Лабораторная работа 3. ТАКТИРОВАНИЕ	
МИКРОКОНТРОЛЛЕРА K1986VE92QI	35
Лабораторная работа 4. ТАЙМЕР	44
Лабораторная работа 5. ГЕНЕРАЦИЯ ШИМ СИГНАЛА	50
Лабораторная работа 6. АНАЛОГО-ЦИФРОВОЙ	
ПРЕОБРАЗОВАТЕЛЬ	61
Лабораторная работа 7. ИСКЛЮЧЕНИЯ И ПРЕРЫВАНИЯ	79
Лабораторная работа 8. ЦИФРО-АНАЛОГОВЫЙ	
ПРЕОБРАЗОВАТЕЛЬ	93
Лабораторная работа 9. МИКРОПРОЦЕССОРНАЯ СИСТЕМА	
ИМПУЛЬСНО-ФАЗОВОГО УПРАВЛЕНИЯ ТИРИСТОРНЫМ	
ПРЕОБРАЗОВАТЕЛЕМ	99
РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА	

ОСНОВНЫЕ ПОЛОЖЕНИЯ

В данных методических указаниях приведен материал для выполнения лабораторных работ по курсу "Силовая электроника" раздел «Системы управления преобразователей».

Целью лабораторных работ является углубление и закрепление знаний теоретического курса, полученных на лекциях, приобретение практических навыков проектирования и экспериментального исследования микропроцессорных устройств.

Выполнение лабораторной работы включает:

- домашнюю подготовку;
- проведение эксперимента на виртуальных (физических) моделях;
- обработка результатов;
- защита лабораторной работы.

Домашняя подготовка осуществляется в соответствии с заданием, приведенном в указаниях к каждой лабораторной работе. Бланк отчета с результатами домашней подготовки составляется студентом и является основным документом при допуске к выполнению работы.

Лабораторные работы выполняются бригадами в составе двух (или трех) человек. Отчет о работе выполняется каждым студентом самостоятельно.

При защите лабораторных отчетов студент должен знать:

- тему и цель работы;
- порядок выполнения работы;
- принцип действия и сущность алгоритмов управления использованных при проектировании устройств;
- теоретический материал по теме работы расчетные формулы, используемые для обработки результатов измерений.

Защита лабораторных работ проводится на специально выделенных для этого занятиях индивидуально каждым студентом. Для подготовки к защите составлены контрольные вопросы, приведенные в указаниях к каждой работе.

ПРАВИЛА ТЕХНИКИ БЕЗОПАСНОСТИ

При выполнении лабораторных работ необходимо помнить об опасности поражения электрическим током и придерживаться всех мер безопасности. Каждый студент, который приступает к занятиям в лаборатории, должен пройти первичный инструктаж по технике безопасности и расписаться в соответствующих регистрационных журналах по технике безопасности.

Приступая к выполнению лабораторной работы, необходимо помнить и строго придерживаться следующих правил техники безопасности.

- прежде чем приступать к соединению элементов электрической цепи необходимо убедиться, что на стенд не подано электрическое питание;

- категорически запрещается прикасаться к незащищенным частям соединительных проводников, зажимов и других токоведущих частей электрической цепи, находящихся под напряжением;

- подключать питание к стенду можно только с разрешения преподавателя после предварительной проверки им схемы;

- всякие изменения в схеме должны производиться только при выключенном питании, причем после каждого изменения соединений схема подлежит обязательной проверке преподавателем;

- запрещается оставлять без надзора электрическую цепь под напряжением;

- прежде чем разбирать электрическую цепь, необходимо убедиться в том, что источник питания стенда отключен;

- в случае прекращения подачи электрической энергии от сети или возникновения других аварийных ситуаций, исследуемая установка должна быть немедленно отключена от сети, а студенты должны сообщить об этом преподавателю.

Студент несет персональную ответственность за нарушение порядка и правил безопасности в лаборатории, а в случае порчи приборов или оборудования – материальную ответственность.

ОПИСАНИЕ ЛАБОРАТОРНОГО СТЕНДА

Основой лабораторного стенда является демонстрационно-отладочная плата с 32-разрядным микроконтроллером семейства **1986BE9X** разработки и производства компании "Миландр", внешний вид которой представлен на рисунке 1. Описание элементов, входящих в состав платы приведено в таблице 1.

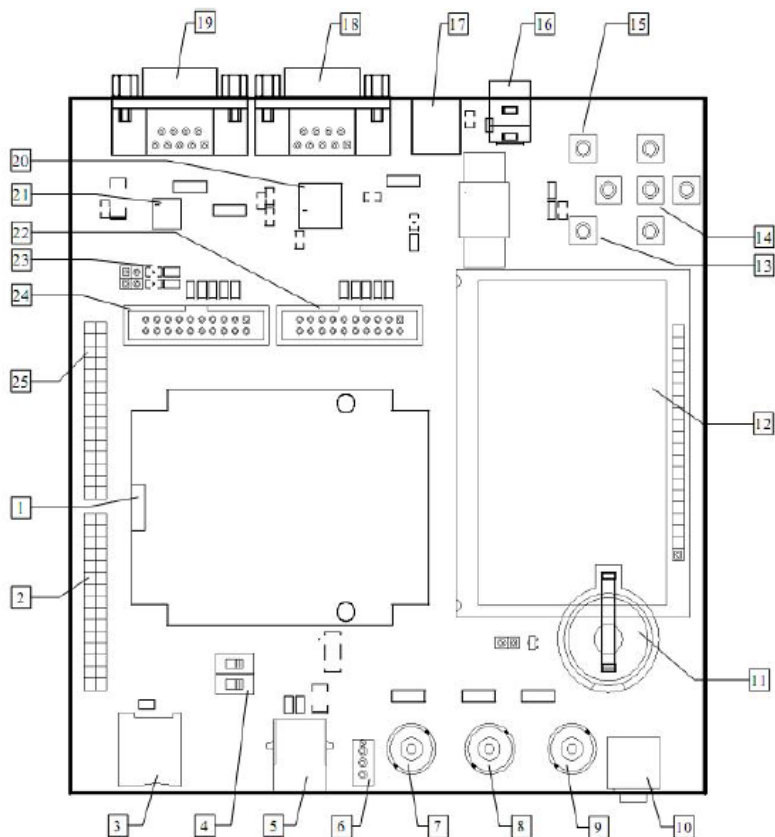


Рисунок 1. Внешний вид отладочной платы

Таблица 1

Описание элементов отладочной платы

№	Наименование
1	Контактирующее устройство для микроконтроллера 1986BE92У.

	Микроконтроллер должен быть установлен в спутник-держатель.
2	Разъем X27 для портов A,E,F микроконтроллера.
3	Разъем карты памяти microSD.
4	Переключатели выбора режима загрузки.
5	Разъем USB-B.
6	Подстроечный резистор на 7-м канале АЦП.
7	Разъем BNC внешнего сигнала на 7-м канале АЦП.
8	Разъем BNC внешнего сигнала на 1-м входе компаратора.
9	Разъем BNC на выходе ЦАП.
10	Разъем Audio 3.5мм на выходе усилителя звуковой частоты, ко входу которого подключен ЦАП.
11	Батарейка 3.0В.
12	Жидкокристаллический индикатор 128х64 точек.
13	Кнопка WAKEUP.
14	Кнопки UP, DOWN, LEFT, RIGHT, SELECT.
15	Кнопка RESET.
16	Разъем питания 5В.
17	Фильтр питания.
18	Разъем RS-232.
19	Разъем CAN.
20	Приемо-передатчик интерфейса RS-232 на микросхеме 5559ИН4.
21	Приемо-передатчик интерфейса CAN на микросхеме 5559ИН14.
22	Разъем программирования и отладки JTAG-B.
23	Набор из двух красных светодиодов, подключенных к выводам 0 и 1 порта C.
24	Разъем программирования и отладки JTAG-A.
25	Разъем X26 для портов B,C,D микроконтроллера.

Питание платы осуществляется от блока питания, который представлен на рисунке 2. Адаптер подключается к разъёму 16 (см. рисунок 1). Выходное напряжение адаптера составляет 5В постоянного напряжения.

Программирование Flash-памяти и отладка программ осуществляется через программатор/отладчик MT-Link, который представлен на рисунке 3



Рисунок 2. Блок питания



Рисунок 3. Программатор/отладчик MT-Link

Первое включение.

1. Установить среду разработки Keil μ Vision 5.
2. Установить программное обеспечение для программатора/отладчика Setup_JLinkARM_V468a.
3. Запустить и установить файл Milandr.MDR1986BExx.1.3.0.
4. В папку \Keil_v5\ARM\Firmware скопировать файл MDR32F9x.FLM.
5. Открыть и установить файл MDR_spl_v1.4.4.
6. Подключить выход блока питания к разъёму питания платы.
7. Установить переключатели4 (см. рисунок 1) в положение FLASH/JTAGA и подключить программатор к разъёму JTAGA.

ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ Keil μ Vision

Создание проекта в среде разработки Keil μ Vision

1. Создать папку проекта и скопировать в неё файл MDR32F9Qx.lib.h.
2. Запустить среду разработки Keil μ Vision и в меню Project выбрать пункт New μ Vision Project (Рисунок 1.1).

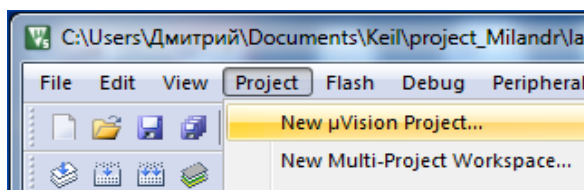


Рисунок 1.1

3. В открывшемся диалоговом окне выбрать папку, где будет находиться проект и указать его имя (Рисунок 1.2).
4. Далее необходимо выбрать марку микроконтроллера MDR1986BE92 (Рисунок 1.3).
5. В появившемся диалоговом окне выбрать программные компоненты: CORE и Startup MDR1986BE9x (Рисунок 1.4).

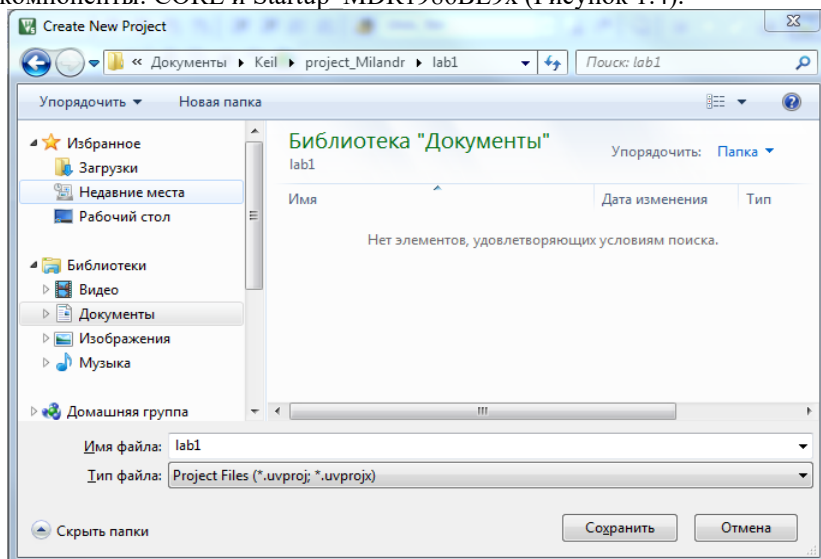


Рисунок 1.2

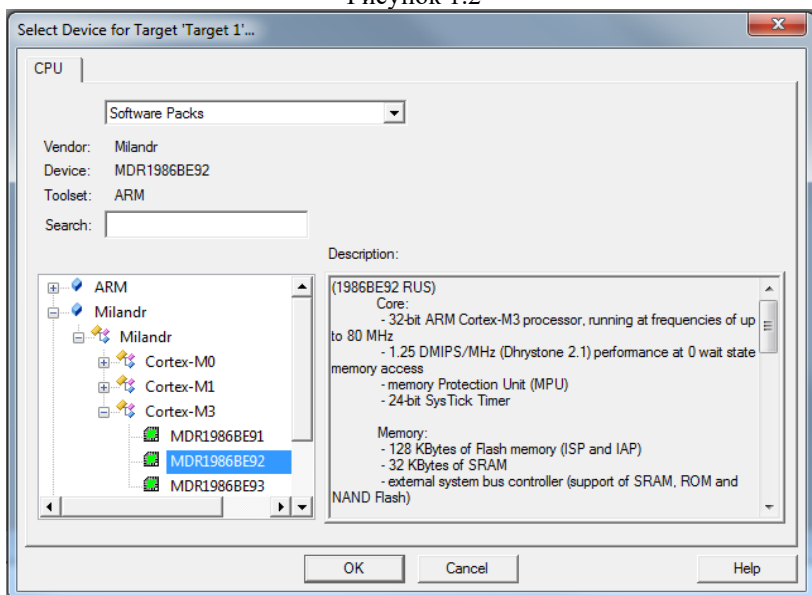


Рисунок 1.3

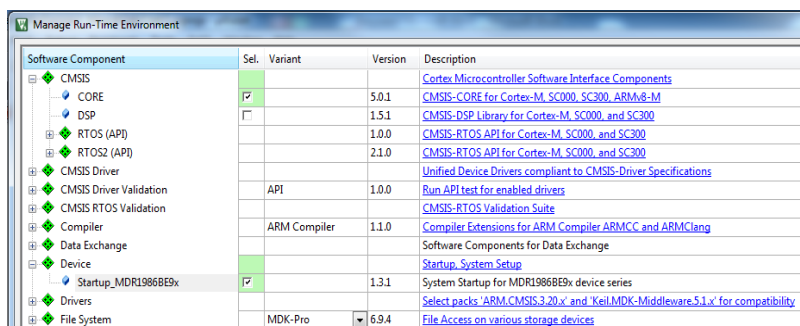


Рисунок 1.4

6. Переименовать группу Source Group 1 в Main (Рисунок 1.5).

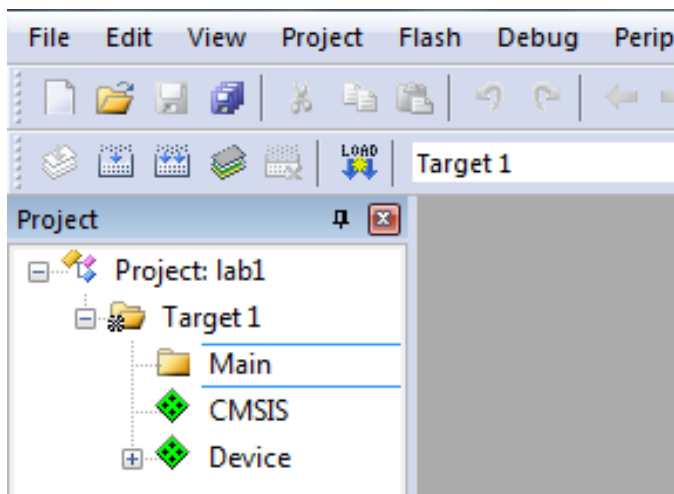
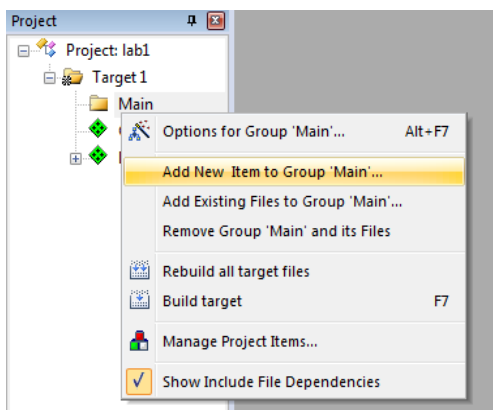


Рисунок 1.5

7. В группу добавить файл с расширением C, который будет содержать код основной программы. Для этого необходимо на значок группы нажать правой кнопкой мыши и выбрать пункт Add New Item to Group 'Main'...(Рисунок 1.6,а). В открывшемся диалоговом окне выбрать тип создаваемого файла (в данном случае C.file) и указать его имя (для файла основной программы main) (Рисунок 1.6,б).



а)



б)

Рисунок 1.6.

8. Далее необходимо указать путь для дополнительных подключаемых файлов. Этот проект будет содержать только один дополнительный файл: MDR32F9Qx_lib.h. Выбрать в меню Project пункт Option for Target `Target 1`... В появившемся диалоговом окне выбрать вкладку C/C++ (Рисунок 1.7). В поле Include Paths нажать кнопку . В появившемся диалоговом окне (Рисунок 1.8.а) нажать кнопку , а затем в новом поле кнопку  и в открывшемся окне указать расположение подключаемого файла (Рисунок 1.8. б).

9. В диалоговом окне Option for Target `Target 1`... выбрать вкладку Debug. Установить в качестве отладчика/программатора J-LINK/J-TRACE Cortex как показано на рисунке 1.9.

Зайти в настройки программатора, нажав кнопку Settings (Рисунок 1.10).

Во вкладке Debug произвести настройку программатора, для этого необходимо в поле Port установить значение SW, а в поле Max Clock 1MHz. (Рисунок 1.11).

Во вкладке Flash Download под полем Programming Algorithm (Рисунок 1.12) нажать кнопку Add и в появившемся окне из списка выбрать MDR32f9x 128kB Flash (Рисунок 1.13) и нажать Add. Удалить в

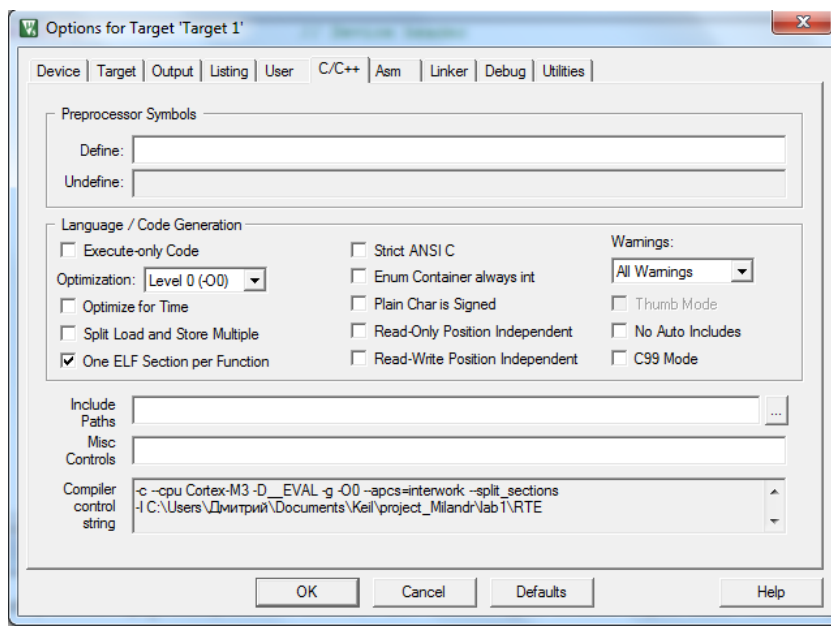
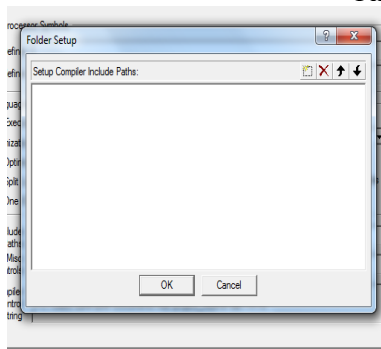
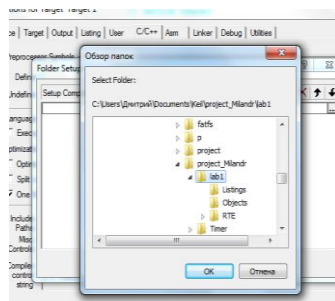


Рисунок 1.7.



а)



б)

Рисунок 1.8.

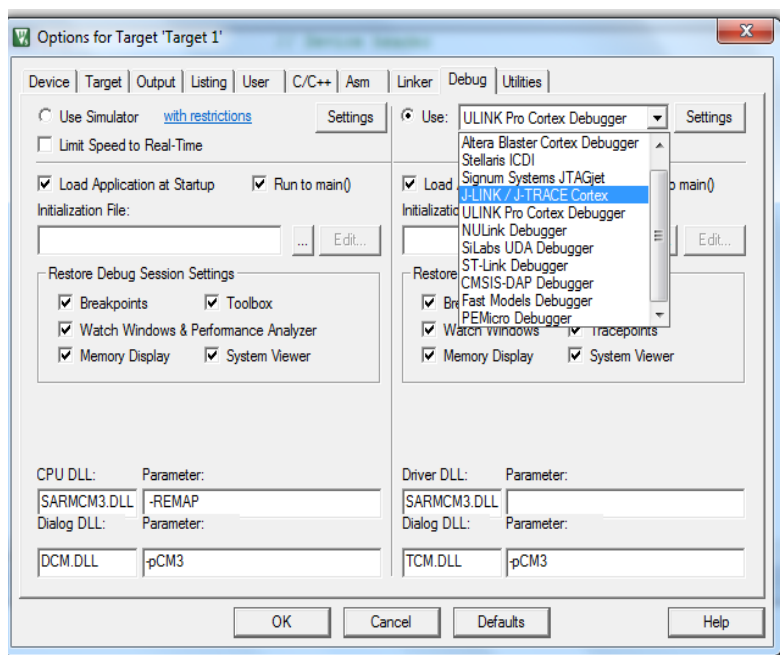


Рисунок 1.9.

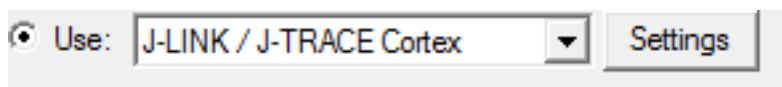


Рисунок 1.10

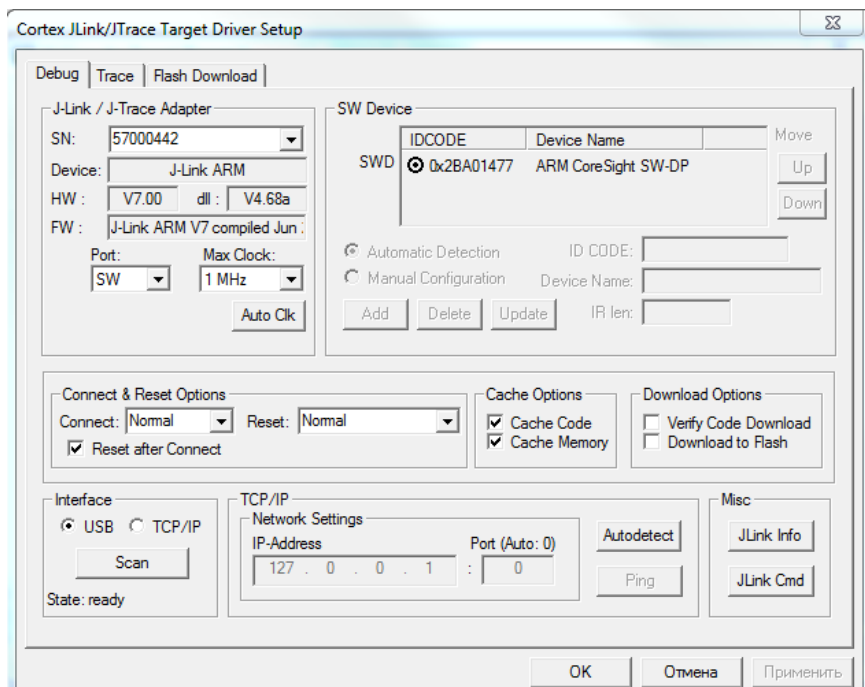


Рисунок 1.11

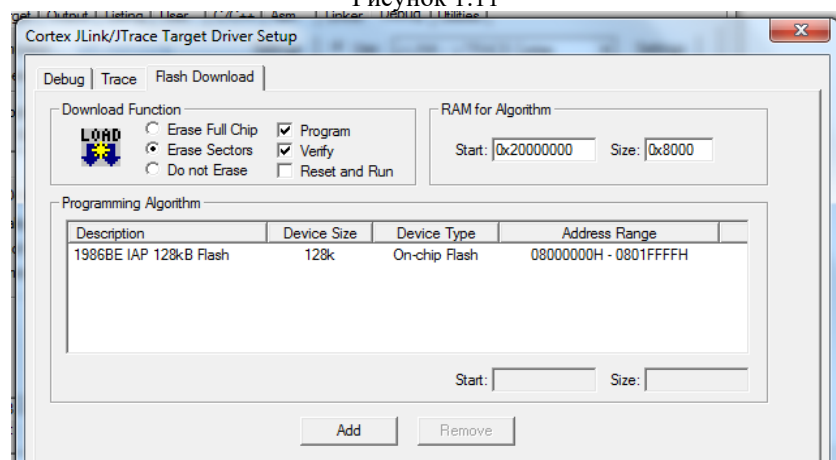


Рисунок 1.12

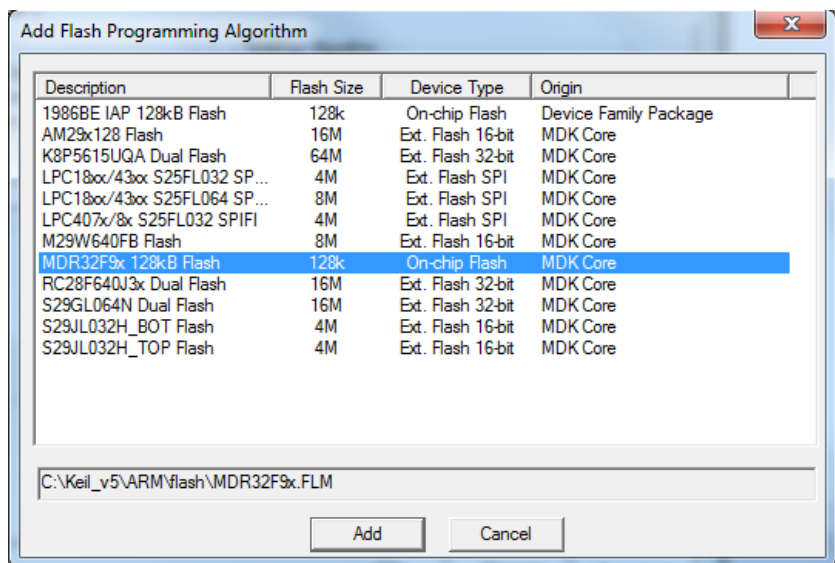


Рисунок 1.13

поле Programming Algorithm строку со значением 1986BE IAP 128kB Flash. Также необходимо в поле функции загрузки (Download Function) установить переключатель на пункт полная очистка чипа (Erase Full Chip) и отметить пункты: программировать (Program), проверить (Verify) и сбросить и запустить (Reset and Run) (Рисунок 1.12).

Создание программы

Программа на языке C для микроконтроллера должна включать в себя функцию `int main()` и строку подключения заголовочного файла. Для подключения заголовочного файла необходимо в окне текстового редактора щёлкнуть правой кнопкой мыши. Из предложенного списка выбрать пункт `Insert '#include <efile>'` и выбрать `MDR32F9x.h` (рисунок 1.14).

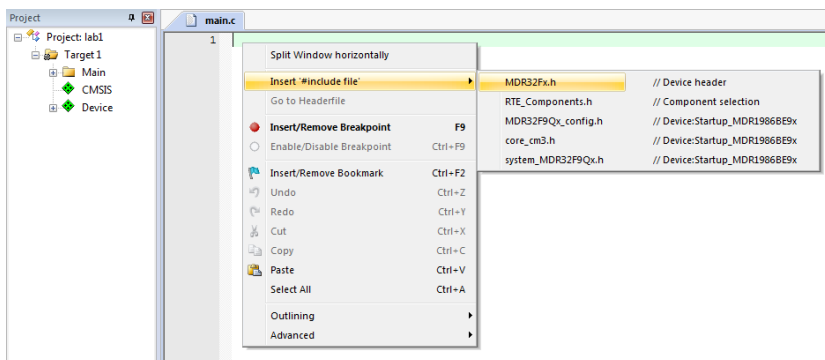


Рисунок1.14

Написать функцию `int main()` как показано на рисунке 1.15. Далее необходимо скомпилировать программу, для этого необходимо нажать кнопку **Build** на панели инструментов. Результаты компиляции отображаются в окне Build Output (рисунок 1.16). В этом окне указываются сообщения об ошибках и предупреждениях, а так же размер сегментов памяти программ: Code, RO-data (Read Only Data), RW-data– (ReadWriteData), ZI-data– (Zero-Initialized Data).

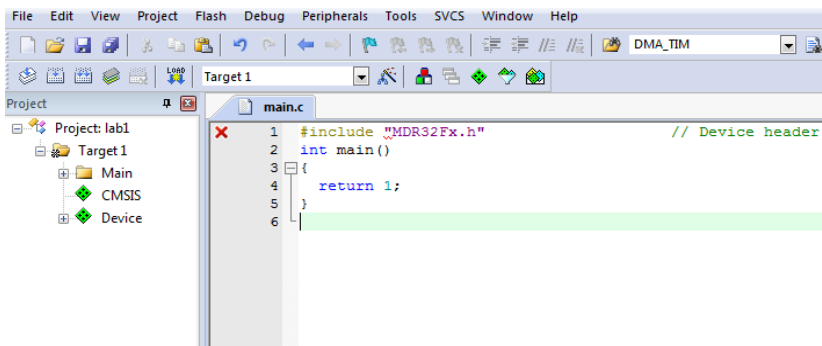


Рисунок 1.15

```
Build Output
compiling main.c...
assembling startup_MDR32F9Qx.s...
compiling system_MDR32F9Qx.c...
linking...
Program Size: Code=548 RO-data=224 RW-data=4 ZI-data=1636
".\Objects\lab1.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:00
```

Рисунок 1.16

Если всё сделано без ошибок, то можно приступить к записи программы во flash-память микроконтроллера.

Запись программы во flash-память контроллера и режим отладки

После того как программа реализована и скомпилирована без ошибок необходимо загрузить её во flash-память контроллера. Делается это нажатием кнопки  (Download). После загрузки программы контроллер перезагрузиться автоматически если установлена опция сбросить и запустить (Reset and Run) (рисунок 1.12) или его необходимо принудительно сбросить, нажав на кнопку RESET на демонстрационной плате, для того чтобы записанная программа начала исполняться.

Для того чтобы загрузить программу и перейти в режим отладки необходимо нажать кнопку “Начать/остановить сеанс отладки” (Start/Stop Debug Session) . После нажатия этой кнопки программа будет загружена во flash-память контроллера, Keil будет запущен в режиме отладки.

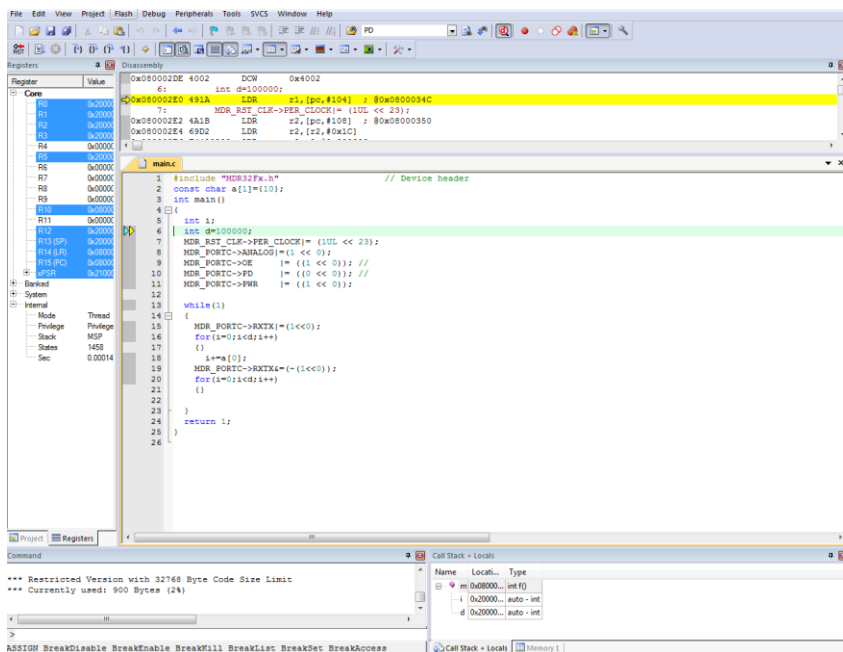


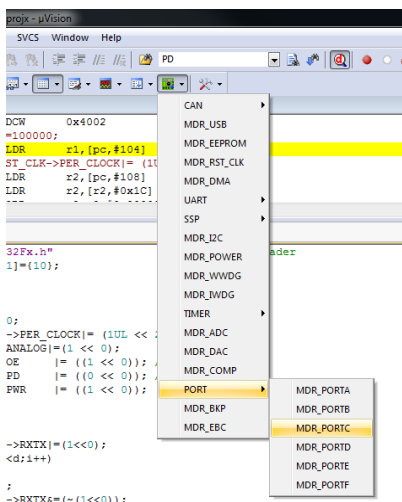
Рисунок 1.17. Режим отладки

В режиме отладки Keil предлагает различные инструменты для анализа работы программы. Ниже будут перечислены некоторые из них. Доступ к ним осуществляется нажатием на соответствующую кнопку на панели инструментов. (Рисунок 1.18)

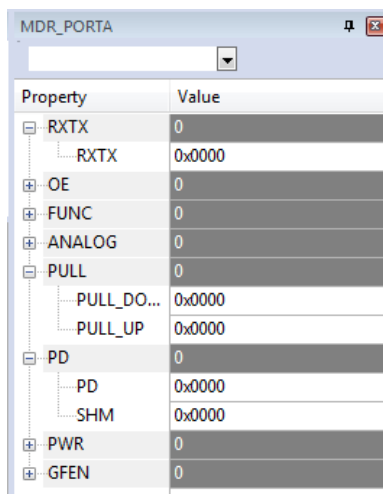


Рисунок 1.18 Панель инструментов в режиме отладки

1. Наблюдения за системой (System Viewer Windows).  Позволяет просматривать состояние всех регистров периферии. На рисунке 1.19 представлен пример просмотра состояния регистров порта A.



a)



б)

Рисунок 1.19. Просмотр регистров порта А с использованием System Viewer Windows

2. Окно дизассемблера (Disassembly Window), вызывается нажатием кнопки . Позволяет посмотреть программу на языке ассемблера (Рисунок 1.20).

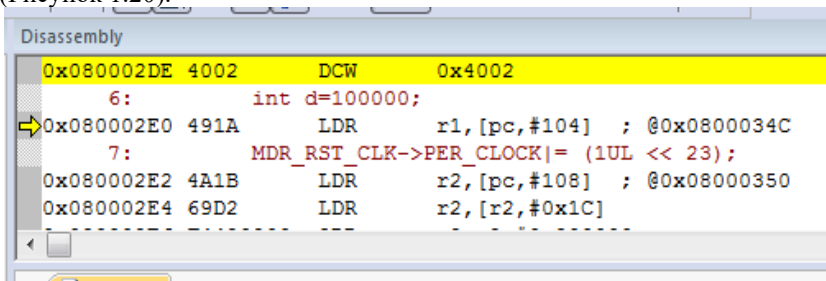


Рисунок 1.20 Окно дизассемблера

3. Кнопка вызова окна просмотра регистров процессора (Registers Windows), вызывается нажатием кнопки . В этом окне можно посмотреть значение всех регистров процессора (рисунок 1.21).

Registers	
Register	Value
Core	
R0	0x20...
R1	0x20...
R2	0x20...
R3	0x20...
R4	0x00...
R5	0x20...
R6	0x00...
R7	0x00...
R8	0x00...
R9	0x00...
R10	0x08...
R11	0x00...
R12	0x20...
R13 (SP)	0x20...
R14 (LR)	0x08...
R15 (PC)	0x08...
xPSR	0x21...
Banked	
System	
Internal	
Mode	Thread
Privilege	Privil...
Stack	MSP
States	1458
Sec	0.00...

Рисунок 1.21 Окно просмотра регистров процессора

4. Окно просмотра глобальных переменных (Watch window), вызывается нажатием кнопки . Позволяет просматривать и записывать значение глобальных переменных и структур (Рисунок 1.22). Для того чтобы добавить переменную в окно необходимо ввести её имя в поле <Enter expression>.

5. Окно просмотра оперативной и flash памяти (MemoryWindows), вызывается нажатием кнопки . В этом окне можно просматривать участок памяти с начальным адресом, указанным в поле Address (Рисунок 1.23).

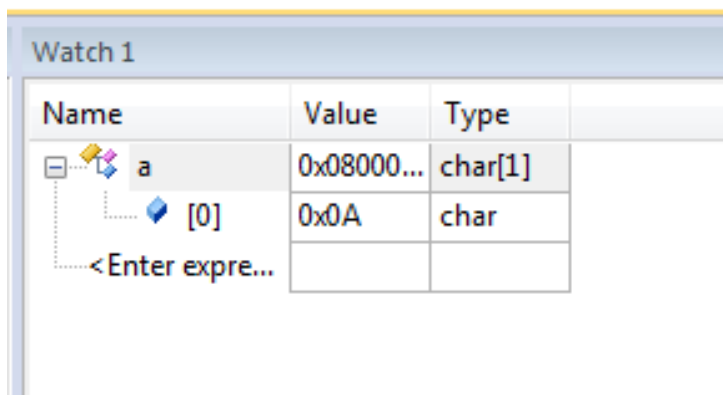


Рисунок 1.22. Окно просмотра глобальных переменных

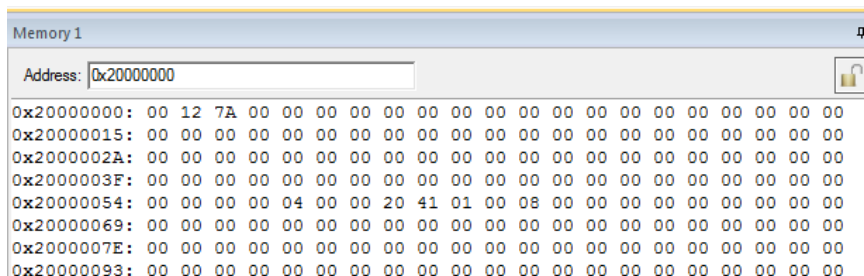


Рисунок 1.23. Окно просмотра оперативной и flash памяти

6. Окно просмотра локальных переменных (Call Stack Window) , вызывается нажатием кнопки . В этом окне можно просматривать и

задавать значения переменных объявленных внутри функций (рисунок 1.25).

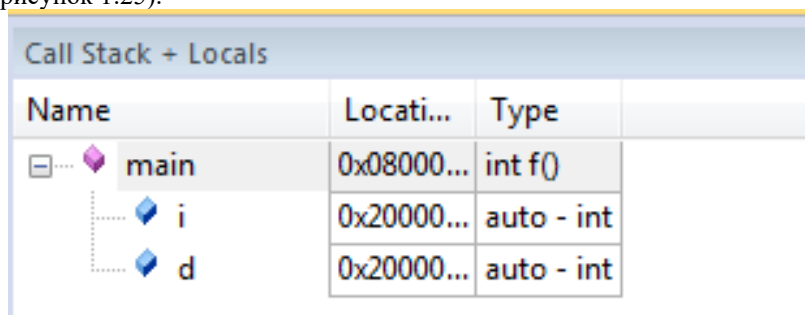


Рисунок 1.24. Окно просмотра локальных переменных

7. Кнопка запуска программы .
8. Кнопка сброса выполнения программы .
9. Кнопка установить точку останова .
10. Кнопка удаления всех точек останова .

Задания для самостоятельной работы.

- 1 Включить компьютер и запустить среду разработки Keil μ Vision. Создать новый проект с уникальным именем.
- 2 В файле main.C записать код, представленный в листинге 1.1.

Листинг 1.1

```
#include "MDR32Fx.h"           // Device header
int main()
{
    int i;
    int d=100000;
    MDR_RST_CLK->PER_CLOCK|= (1UL << 23);
    MDR_PORTC->ANALOG|= (1 << 0);
    MDR_PORTC->OE   |= ((1 << 0)); //
    MDR_PORTC->PD   |= ((0 << 0)); //
    MDR_PORTC->PWR  |= ((1 << 0));
    while(1)
```

```

    {
        MDR_PORTC->RXTX|=(1<<0);
        for(i=0;i<d;i++) {}
        MDR_PORTC->RXTX&=~(1<<0));
        for(i=0;i<d;i++) {}
    }
    return 1;
}

```

- 3 Выполнить сборку проекта и компиляцию программы.
- 4 Загрузить программу во flash-память микроконтроллера.
- 5 В режиме отладки запустить выполнение программы.
- 6 В отчете по лабораторной работе записать результаты компиляции: размер сегментов памяти программ, количество ошибок, количество предупреждений, а также описать результат выполнения программы.

Контрольные вопросы

- 1 Перечислите основные этапы создания проекта в среде Keil μ Vision.
- 2 Как открыть существующий проект в среде разработки Keil μ Vision?
- 3 Какие основные окна среды разработки Keil μ Vision
- 4 Какие заголовочные файлы необходимо добавить в проект?
- 5 Какой минимальный набор программных компонентов необходим?
- 6 Как переименовать папку (группу) в дереве проекта?
- 7 Как подключить существующий файл к существующему проекту?
- 8 Как произвести настройку программатора?
- 9 Назначение функции main в языке C?
- 10 Как вычислить размер программы записываемой во flash-память контроллера?

ОРГАНИЗАЦИЯ ВВОДА/ВЫВОДА ИНФОРМАЦИИ С ПОРТОВ МИКРОКОНТРОЛЛЕРА K1986BE92QI

Краткие теоретические сведения о работе с портами ввода/вывода микроконтроллера K1986BE92QI

Почти все выводы микросхемы K1986BE92QI представляют собой цифровые линии ввода или вывода. Каждую такую линию можно программным путем сконфигурировать как цифровой вход или цифровой выход, а некоторые контакты можно настроить в аналоговом режиме работы (вход АЦП или выход ЦАП). Для удобства использования линии ввода-вывода объединены в порты по 16 линий. Такие порты называют портами ввода-вывода общего назначения. В англоязычной литературе источники линии ввода-вывода принято именовать термином GPIO – General-PurposeInput/Output.

Таблица 2.1.

Порты ввода/вывода микроконтроллера K1986BE92QI

Наименование порта	Количество линий	Наименование линий
PORTA	8	PA0...PA7
PORTB	11	PB0...PB10
PORTC	3	PC0...PC2
PORTD	8	PD0...PD7
PORTE	6	PE0...PE3, PE6...PE7
PORTF	7	PF0...PF6

Внутреннее устройство отдельной линии ввода-вывода представлено на рисунке 2.1. На входе каждой линии ввода-вывода есть два быстродействующих диода Шотки: один подключен к минусу линии питания, второй – к плюсу. Это нужно для того, чтобы защитить линию от чрезмерного напряжения, которое может появиться на входе в результате разных причин: ошибки при разработке схемы, выхода части схемы из строя, статического электричества и т.п. Такой подход позволяет повысить надёжность микросхемы. Каждый цифровой выход у микроконтроллера K1986BE92QI обеспечивает нагрузочную способность до 6 мА.

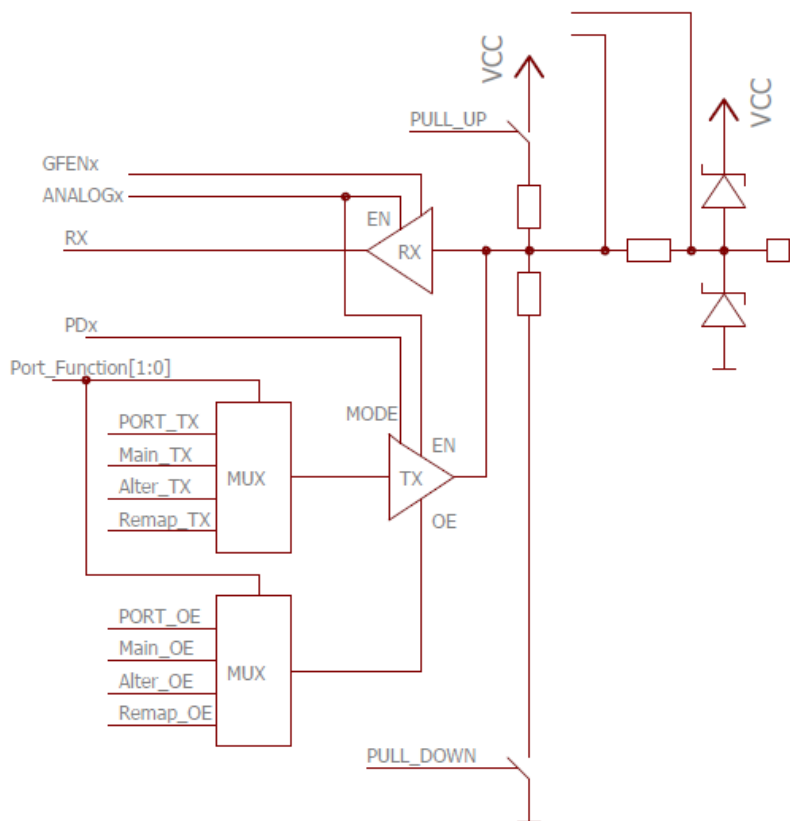


Рисунок 2.1. Схема линии ввода/вывода микроконтроллера

Кроме того, на каждой линии расположены так называемые резисторы подтяжки – PullUp и PullDown, управляемые транзисторными ключами.

Резистор Pull Up позволяет подтянуть потенциал (т.е. подключить) линии к плюсу питания (цепь Vcc). С помощью же резистора Pull Down удастся подтянуть потенциал линии к минусу питания. Резисторы подтяжки часто требуются при разработке схемы устройства. Наличие таких резисторов непосредственно в микроконтроллере значительно снижает размеры печатной платы разрабатываемого устройства, уменьшает количество компонентов, число паек и стоимость изделия в целом. Однако некоторые разработчики с осторожностью используют их, так как сопротивление

этих резисторов плохо нормируется, а сами резисторы сравнительно часто выходят из строя, в связи с этим надежней установить внешний резистор.

Для того чтобы начать использовать линии ввода-вывода, нужно предварительно сконфигурировать их соответствующим образом.

На самом низком уровне работа с портами ввода-вывода (как и со всеми другими периферийными устройствами) осуществляется с помощью специальных регистров микроконтроллера. Эти регистры доступны как ячейки памяти, расположенные по определенным адресам. Зная эти адреса (они описаны в документации на микроконтроллер), можно записывать в регистры определенные значения, задавая требуемую конфигурацию. Через другие регистры можно получать данные от периферийных устройств.

Конфигурирование линий ввода-вывода

1. Вначале необходимо подключить нужный порт к тактированию в регистре PER_CLOCK, установив соответствующий бит (PORTA–бит 21, PORTB–бит 22, PORTC–бит 23, PORTD–бит 24, PORTE–бит 25).

```
MDR_RST_CLK->PER_CLOCK    |= (1 << 21); // подключение к тактированию порта А
```

2. В регистре управления порта (OE) задать режим работы линии, либо как цифровой вход, либо цифровой выход

```
PORTx->OE|= ((0 << pin)); //вход  
PORTx->OE|= ((1 << pin)); //выход
```

3. В регистре режима работы порта (FUNC), задать функцию, которую будет выполнять вывод порта: в режим работы дискретного входа/выхода (режим порт), либо работать под управлением периферийного модуля (UART, выход канала сравнения таймера, CAN и тд.), в этом случае выбирается основная, альтернативная и переопределённая.

```
MDR_PORTA->FUNC|=((0 << pin*2)); //порт  
MDR_PORTA->FUNC|=((1 << pin*2)); // основная функция;  
MDR_PORTA->FUNC|=((2 << pin*2)); // альтернативная функция  
MDR_PORTA->FUNC|=((3 << pin*2)); // переопределенная функция
```

4. В регистре управления аналоговым режимом (ANALOG) работы порта указать, в каком режиме будет работать линия: аналоговый или

цифровой. Если контакт будет работать под управлением АЦП или ЦАП, то необходимо выбирать аналоговый режим, а если под управлением других модулей, то цифровой.

```
PORTx->ANALOG|=(1 <<pin);// цифровой  
PORTx->ANALOG|=(0 <<pin);// аналоговый
```

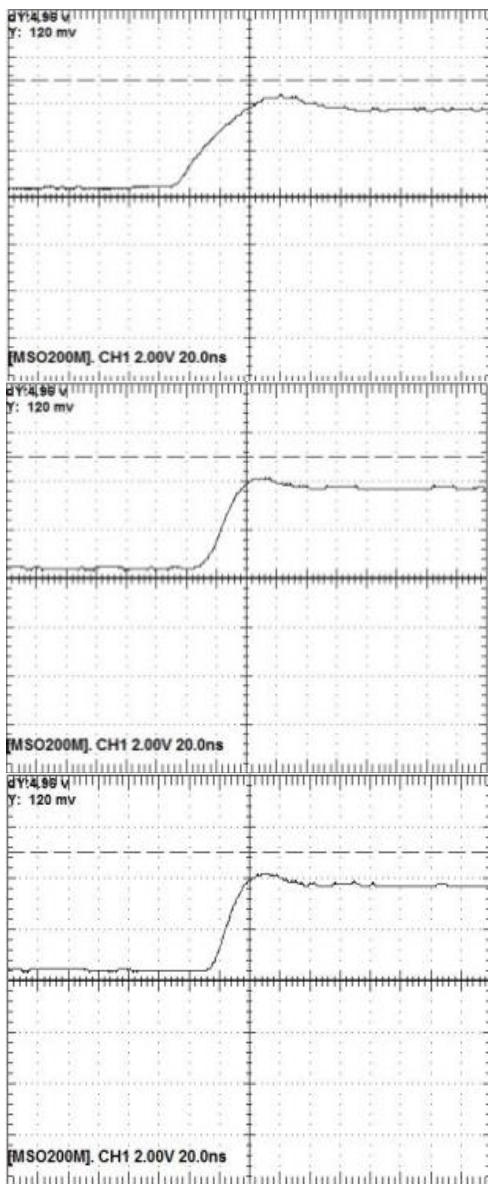
5. В регистре управления подтяжки контакта (PULL) осуществляется подключение подтягивающего резистора. Старшие 16 бит регистра осуществляют подключение подтягивающего резистора к питанию, а младшие к земле.

```
PORTx->PULL|=((0 <<pin) | (1<< (pin+16))); //подтяжка к питанию  
PORTx->PULL|=((1<<pin) | (0<< (pin +16))); //подтяжка к земле
```

6. В регистре PD задать режим работы драйвера. Если контакт будет использоваться в качестве цифрового входа, то возможно указать с каким гистерезисом будет работать триггер Шмитта: 0 – гистерезис 200 мВ, 1 – гистерезис 400мВ. Если контакт будет использоваться в качестве цифрового выхода, то можно указать его тип: 0 – управляемый драйвер, 1 – открытый сток.

```
//для цифрового входа  
PORTA->PD|=((0 << (pin+ 16)));// гистерезис 200мВ  
PORTA->PD|=((1 << (pin + 16)));// гистерезис 400мВ  
//для цифрового выхода  
PORTA->PD|=(0 <<pin);// управляемый драйвер  
PORTA->PD|=(1 <<pin);//открытый сток
```

7. В регистре задания мощности передатчика (PWR) можно задать крутизну нарастания фронтов выходных импульсов, для линии микроконтроллера, которая работает в режиме цифрового выхода. Возможны три режима: медленный ($PWRx[1:0]=01_2$), быстрый ($PWRx[1:0]=10_2$) и максимально быстрый ($PWRx[1:0]=11_2$). На рисунке 2.2 представлены осциллограммы фронтов выходного цифрового сигнала при различных значениях битового поля $PWR[1:0]$.



PWR[1:0]=1

PWR[1:0]=2

PWR[1:0]=3

Рисунок 2.2 Осциллограмма фронтов при различных режимах

```
PORTA->PWR|=((1 <<pin*2));
```

Если вывод микроконтроллера будет работать в режиме цифрового входа, и будет принимать сигнал, в котором возможны помехи, то в регистре PORTx->GFEN можно включить входной фильтр, который позволяет устранять короткие импульсы с длительность до 10 нс.

```
PORTA->GFEN|= ((0 <<pin)); //фильтр выключен
```

```
PORTA->GFEN|= ((1 <<pin)); //фильтр включен
```

Для того что бы прочитать или записать информацию с необходимой цифровой линии, используется регистр RXTX. Данный регистр является шестнадцатиразрядным. Значение каждого из 16 бит соответствует значению цифрового сигнала на линии порта.

В листинге 1.1 представлена программа настройки контакта 1 порта А в режиме работы цифрового выхода.

Листинг 1.1

```
#define PIN 1
```

```
PORTA->FUNC|=((0 <<PIN *2));
```

```
PORTA->ANALOG|= (1 <<PIN);
```

```
PORTA->OE   |= ((1 <<PIN)); //
```

```
PORTA->PULL |= ((0 <<PIN) | (0 << (PIN + 16))); //
```

```
PORTA ->PD   |= ((0 <<PIN)); //
```

```
PORTA->PWR   |= ((1 <<PIN*2));
```

В листинге 1.2 представлена программа настройки контакта 1 порта А в режиме работы цифрового входа.

Листинг 1.2

```
#define PIN 1
```

```
PORTA ->FUNC|=((0 <<PIN *2));
```

```
PORTA ->OE   |= ((0 <<PIN));
```

```
PORTA ->ANALOG|= (1 <<PIN);
```

```
PORTA ->PULL |= ((0 <<PIN) | (0 << (PIN +16)));
```

```
PORTA ->PD   |= ((0 << (PIN<< 16)));
```

```
PORTA ->PWR   |= ((1 <<PIN *2));
```

```
PORTA->GFEN   |= ((1 <<PIN));
```

В листинге 1.3 представлена программа настройки контакта 2 порта D для работы в аналоговом режиме

Листинг 1.3

```
#define PIN 2
MDR_RST_CLK->PER_CLOCK|= (1UL<< 24); // тактирования порта D
MDR_PORTD->OE|= ((0 << PIN)); // направление передачи данных =
Вход
MDR_PORTD->ANALOG |= (0 << PIN); // режим работы Аналоговый
```

В листинге 1.4 представлена программа настройки контакта 1 порта A для работы в аналоговом режиме

Листинг 1.4

```
#define PIN 1
MDR_PORTA->FUNC   |= (2<<PIN*2); // альтернативная функция
MDR_PORTA->ANALOG   |= (1 << PIN); // цифровой режим
MDR_PORTA->PWR      |= (3<<PIN*2); //
максимальная крутизна фронтов
```

В листинге 1.5 представлена программа записи и сброса логической единицы на линию с номером pin порта C.

Листинг 1.5

```
MDR_PORTC->RXTX|= (1<<pin); // установить 1 на линии с номером pin
MDR_PORTC->RXTX&= ~(1<<pin); // установить 0 на линии с номером
pin
```

В листинге 1.6 представлена программа проверки наличия логической единицы на линии с номером pin порта C.

Листинг 1.6

```
if((MDR_PORTC->RXTX & (1<< pin))==0)
{
    //
}
```

Задания для самостоятельной работы

- 1 Собрать схему, представленную на рисунке 2.2.

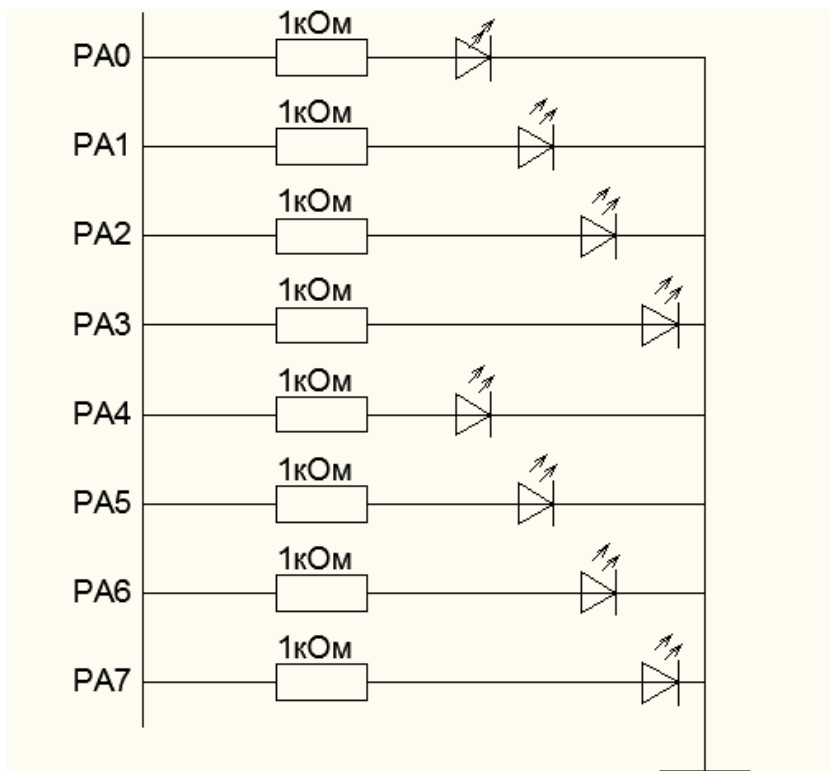


Рисунок 2.2

- 2 Написать программу реализующую вычисление функции $y=f(a,b,c)$ (см. таблицу 2.2). При нажатии заданной кнопки (см. таблицу 2.2) результат вычисления функции должен быть выведен в двоичной форме на светодиоды. Подключение кнопок показано в таблице 2.3.

Таблица 2.2.

Задание №1

Вариант	$y=f(a,b,c)$	Кнопка
1	$y=5*a+b-c$	SEL
2	$y=a*c-b*b$	UP
3	$y=a*b-c*a$	DOWN
4	$y=a+c-b*b$	LEFT
5	$y=2a*c- b$	RIGHT
6	$y=3c+a- b$	SEL
7	$y=3(c+a- b)$	UP
8	$y=2((c+a)- b)$	DOWN
9	$y=2((c+a)- b)$	LEFT
10	$y=(3b-c)+a*a$	RIGHT
11	$y=3a-2b+4c$	SEL
12	$y=a*b-2c$	UP
13	$y=a*a+b*b-c*c$	DOWN
14	$y=2(a-b)+c$	LEFT
15	$y=2*a+(b-3)*c$	RIGHT
16	$y=a+2-(c-b)$	SEL
17	$y=2a-(c+b)$	UP
18	$y=2b+3(a-c)$	DOWN
19	$y=4*(a-3)+b-c$	LEFT
20	$y=(4a-b)+c-10$	RIGHT

Таблица 2.3.

Подключение кнопок на отладочной плате

Кнопка	Контакт
SEL	PC2
UP	PB5
DOWN	PE1
LEFT	PE3
RIGHT	PB6

3 Разработать программу реализующую таблицу истинности в соответствии с вариантом.

Таблица 2.3.

Задание №2

№	Вход					Выхода порта А							
	SEL	UP	DOW N	LEF T	RIGH T	0	1	2	3	4	5	6	7
1	1	0	1	0	0	1	1	0	0	1	1	0	1
2	0	1	1	0	1	1	0	0	0	1	1	0	1
3	0	1	0	1	0	1	1	0	0	1	0	1	0
4	1	1	0	1	0	0	1	0	1	0	0	1	0
5	0	0	1	1	0	1	1	0	1	0	1	1	1
6	1	1	0	0	1	1	0	0	0	0	1	1	1
7	1	1	1	0	1	1	0	1	1	0	0	1	0
8	1	0	0	1	0	1	1	0	0	1	1	0	1
9	0	1	1	0	0	0	0	1	1	1	0	0	0
10	1	0	0	1	1	1	0	1	0	1	0	1	0
11	1	1	1	0	0	0	0	0	1	1	1	0	0
12	0	0	0	1	1	1	0	0	1	0	0	0	1
13	1	0	0	1	1	1	1	1	0	0	0	1	1
14	1	1	0	1	1	1	0	0	0	0	0	0	1
15	0	1	1	1	0	1	1	1	0	0	0	1	1
16	0	0	1	1	1	0	1	1	0	0	1	0	0
17	1	1	0	0	0	1	1	0	0	0	1	1	1
18	1	0	0	1	1	1	0	1	1	1	0	0	0
19	1	1	0	0	1	0	1	1	0	1	0	1	1
20	1	0	0	0	1	1	0	0	1	1	1	0	0

Контрольные вопросы

1. В каких режимах могут работать линии ввода/вывода микроконтроллера?
2. Назначение диодов Шотки во входной/выходной цепи линии ввода/вывода?
3. Каково назначение внутренних подтягивающих резисторов?
4. В чем преимущество и недостаток использования внутренних подтягивающих резисторов?
5. Перечислите последовательность настройки контакта микроконтроллера в режиме работы цифрового входа
6. Перечислите последовательность настройки контакта микроконтроллера в режиме работы цифрового выхода.

7. Перечислите последовательность настройки контакта микроконтроллера в аналоговом режиме работы.

ТАКТИРОВАНИЕ МИКРОКОНТРОЛЛЕРА K1986BE92QI

Краткие теоретические сведения о тактировании микроконтроллера K1986BE92QI

На рисунке 3.1 представлена схема тактирования процессора микроконтроллера K1986BE92QI.

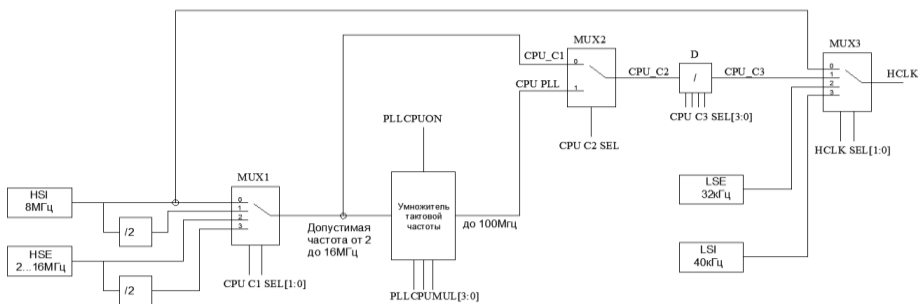


Рисунок 3.1. Схема тактирования процессора микроконтроллера

Из схемы видно, что для тактирования микроконтроллера используются различные источники тактового сигнала: HSI (High Speed Internal – высокочастотный внутренний источник), HSE (High Speed External – высокочастотный внешний источник), LSI (LowSpeedInternal– низкочастотный внутренний источник), LSE(Low Speed External – низкочастотный внешний источник). А для получения необходимой частоты используются мультиплексоры MUX 1-3, умножитель (PLL) и делитель (D) тактовой частоты.

Встроенный RC генератор HSI

Генератор HSI вырабатывает тактовую частоту 8 МГц. Генератор автоматически запускается при появлении питания UCC. 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 40 кГц. Генератор автоматически запускается при появлении питания микроконтроллера. Первоначально тактовая частота генератора LSI используется для формирования дополнительной задержки (~ 4 мс) на включение, для того, чтобы гарантировано установилось напряжение питания. 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.



Рисунок 3.2 Пример внешнего вида кварцевых резонаторов

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.



Рисунок 3.3 Пример внешнего вида кварцевых резонаторов 32 кГц

Генератор запускается при появлении питания BDU_{CC} (напряжение питания от батарейки) и при установленном бите LSEON в регистре BKP_REG_0F. При выходе в нормальный режим работы устанавливается бит 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. В этом режиме Выход OSC_OUT32 находится в третьем состоянии. Так как генератор LSE питается от напряжения питания BDU_{CC} и его регистр управления BKP_REG_0F расположен в батарейном домене, то генератор может продолжать работать при пропадании основного питания U_{CC} . Генератор LSE может использоваться 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.

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.

Последовательность настройки тактовой частоты процессора.

При настройке тактовой частоты процессора вначале необходимо установить задержку для FLASH памяти программ. Скорость доступа во Flash-память ограничена и составляет порядка 40 нс, в результате выдача новых значений из Flash-памяти может происходить с частотой не более 25 МГц. Для того чтобы процессорное ядро могло получать новые инструкции на больших частотах, в микроконтроллере реализуется Flash-память с физической организацией 8K на 128 разрядов (4 сектора 8K на 32 разряда). Данные из различных секторов чередуются в линейном адресном пространстве. Таким образом, за 40 нс из Flash-памяти извлекается 16 байт, в которых может быть закодировано от 4 до 8 последовательных инструкций процессора. И пока ядро выполняет эти инструкции, из памяти извлекается следующая порция данных. Таким образом, тактовая частота может превышать частоту извлечения данных из памяти в несколько раз при линейном выполнении программы. При возникновении переходов в выполнении программы, когда из памяти программ не выбраны нужные инструкции, возникает пауза в несколько тактов процессора для того, чтобы данные успели считаться из Flash. Число тактов паузы зависит от тактовой частоты процессора и задаётся в битовом поле Delay[2:0] регистраMDR_EEPROM->CMD. Зависимость числа тактов паузы от тактовой частоты процессора приведена в таблице 3.1.

Таблица 3.1.

Зависимость тактов паузы при чтении Flash-памяти от тактовой частоты процессора

Delay[2:0]	Тактов паузы	Тактовая частота
0	0	до 25МГц
1	1	до 50 МГц
2	2	до 75 МГц
3	3	до 80 МГц

Если планируется тактирование от внешнего источника HSE, то необходимо установить бит HSE_ON регистра MDR_RST_CLK->HS_CONTROL. И произвести ожидание пока не установится бит HSE_RDY регистра MDR_RST_CLK->CLOCK_STATUS. В случае если необходимо задействовать умножитель то вначале необходимо задать в битовом поле

PLL_CPU_MUL[3:0] регистра MDR_RST_CLK->PLL_CONTROL необходимый коэффициент умножения (PLL_CPU_MUL[3:0] = необходимый коэффициент - 1), а затем включить умножитель, установив бит PLL_CPU_ON регистра MDR_RST_CLK->LL_CONTROL. После чего необходимо подождать пока, установиться бит PLL_CPU_RDY регистра MDR_RST_CLK->CLOCK_STATUS. Далее в регистре MDR_RST_CLK->CPU_CLOCK необходимо произвести переключение мультиплексоров MUX1, MUX2 и MUX3, представленных на рисунке 3.1, а так же значение делителя D (см таблицу 3.2) чтобы сформировать необходимую тактовую частоту процессора HCLK.

Таблица 3.2.

Зависимость тактов паузы при чтении Flash-памяти от тактовой частоты процессора

Значение делителя D	Значение в битовом поле CPU C3 SEL[3:0]
1	0
2	8
4	9
8	10
16	11
32	12
64	13
128	14
256	15

Далее необходимо настроить режим работы встроенного регулятора напряжения в регистре MDR_BKP->REG_0E, записав битовые поля SelectRI[2:0] (Биты с 3 по 5) и LOW[2:0] (Биты с 0 по 2) в соответствии с таблицей 3.3

Таблица 3.3.

Зависимость режима работы внутреннего регулятора напряжения от тактовой частоты процессора

Тактовая частота процессора	SelectRI[2:0]	LOW[2:0]
< 10 МГц	0	0
< 200 кГц	1	1
< 500 кГц	2	2
< 1 МГц	3	3
при выключении всех генераторов	4	4
< 40 МГц	5	5

< 80 МГц	6	6
= 80 МГц	7	7

В листинге 3.1 представлена настройка тактовой частоты процессора на 80МГц при тактировании от внешнего источника HSE на 8 МГц.

Листинг 3.1

```
MDR_EEPROM->CMD |= (3 << 3); // Установить такты паузы для FLASH
MDR_RST_CLK->HS_CONTROL |=
RST_CLK_HS_CONTROL_HSE_ON; // включить HSE осциллятор
while((MDR_RST_CLK->CLOCK_STATUS &
(RST_CLK_CLOCK_STATUS_HSE_RDY)) ==
0x00); // ждём пока осциллятор войдёт в рабочий режим

MDR_RST_CLK->PLL_CONTROL |= ((10-1) << 8);
// установить коэффициент умножения 10
MDR_RST_CLK->PLL_CONTROL |=
RST_CLK_PLL_CONTROL_PLL_CPU_ON; // включить множитель PLL
while((MDR_RST_CLK->CLOCK_STATUS &
RST_CLK_CLOCK_STATUS_PLL_CPU_RDY) !=
RST_CLK_CLOCK_STATUS_PLL_CPU_RDY); //
ожидание готовности множителя
MDR_RST_CLK->CPU_CLOCK = (
    (2 << 0) // переключить MUX1 на вход HSE, CPU_C1=HSE
    | (1 << 2) // переключить MUX2 на вход CPU_PLL,
    CPU_C2=CPU_PLL
    | (0 << 4) // делитель равен 1 CPU_C3
    | (1 << 8)); // переключить MUX3 на вход CPU_C3,
HCLK=CPU_C3
MDR_BKP->REG_0E |= (7 << 0); // режим встроенного регулятора
напряжения DUcc
MDR_BKP->REG_0E |= (7 << 3); // выбор доп. стабилизирующей нагрузки
```

Задания для самостоятельной работы

Реализовать функцию настройки тактовой частоты процессора в соответствии с таблицей задания. Функция должна быть объявлена как void CPU_init(void); Для проверки работоспособности функции необходимо в тело функции main вставить код, представленный в листинге 3.2.

```

int i;
MDR_RST_CLK->PER_CLOCK|= (1UL << 23);
//инициализация для светодиода
PORTC->ANALOG|= (1 << 2);
PORTC->OE |= ((0 << 2));
//инициализация для кнопки
PORTC->ANALOG|= (1 << 0);
PORTC->OE   |= ((1 << 0));
PORTC->PD   |= ((0 << 0));
PORTC->PWR  |= ((1 << 0));
    while(1)
    {
        PORTC->RXTX|= (1 << 0); // включить светодиод
        for(i=0; i<10000000; i++) {} // задержка
        PORTC->RXTX&= ~(1 << 0); // выключить светодиод
        for(i=0; i<10000000; i++) {} // задержка
        // проверка нажатия на кнопку SELECT
        if((PORTC->RXTX & (1 << 2)) != 0)
        {
            CPU_init();
        }
    }

```

Данный код осуществляет алгоритм включения и выключения светодиода и проверку нажатия кнопки SELECT. Если кнопка нажата, то будет вызвана функция инициализации новой тактовой частоты CPU_init();

Таблица 3.4.

Вариант	Источник тактирования	Тактовая частота процессора, МГц
1	HSI	50
2	HSE	72
3	HSI/2	40
4	HSE/2	10
5	HSI	10
6	HSE	25
7	HSI/2	30
8	HSE/2	16

9	HSI	12
10	HSE	24
11	HSI/2	25
12	HSE/2	40
13	HSI	6
14	HSE	1
15	HSI/2	4
16	HSE/2	6
17	HSI	10
18	HSE	70
19	HSI/2	60
20	HSE/2	64

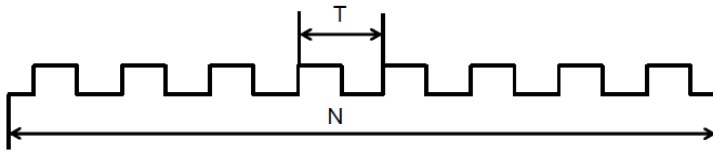
Контрольные вопросы

- 1 Какие источники тактовой частоты могут использоваться для тактирования микроконтроллера?
- 2 Какая максимальная тактовая частота процессора микроконтроллера K1986BE92QI?
- 3 Для чего используется умножитель и делитель тактовой частоты?
- 4 С какой частотой возможна выдача данных из flash-памяти?
- 5 На какой тактовой частоте первоначально запускается процессор микроконтроллера K1986BE92QI? Какой для этого используется источник тактовой частоты?
- 6 Какого назначения источник тактовой частоты LSE?
- 7 Какая максимальная частота допустима для источника LSE?
- 8 Какая максимальная частота допустима для источника HSE?
- 9 Как обозначается тактовая частота процессора в микроконтроллере K1986BE92QI?

ТАЙМЕР

Краткие теоретические сведения о таймерах

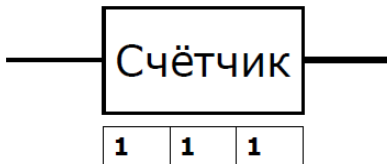
Таймер – это периферийный модуль микроконтроллера, который осуществляет счёт тактовых импульсов поступающих на его вход. Зная частоту и количество отсчитанных импульсов можно рассчитать отрезок времени с момента начала счёта.



$$t = N \cdot T$$

Рисунок 4.1. Тактовые импульсы для таймера

В качестве источника счётных импульсов могут использоваться внутренние (тактовая шина периферии и тактовые импульсы от других таймеров) и внешние (такты, подающиеся на контакты микроконтроллера) источники.



Вход	Выход		
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

n – Разрядность регистра счётчика
 T – Период следования импульсов
 f – Частота следования импульсов

$$t = T \cdot 2^n = 2^n / f$$

Рисунок 4.2.

Для запуска таймера в микроконтроллере K1986BE92QI, необходимо подключить нужный таймер к тактированию, установив соответствующие биты в регистре MDR_RST_CLK->PER_CLOCK (Timer1 – бит14, Timer2 – бит15 и Timer3 – бит16). И в регистре MDR_RST_CLK->TIM_CLOCK. Включить тактовую частоту HCLK(TIMx CLK EN) и выбрать делитель тактовой частоты HCLK (TIMx BRG[7:0]) (таблица 4.1.). Значение делителя определяется по следующей формуле:

$$\text{Коэффициент деления} = 2^{\text{BRG}[7:0]} \quad (4.1)$$

Таблица 4.1.

Биты регистра MDR_RST_CLK->TIM_CLOCK

№ Бита	26	25	24	23...16	15...8	7...0
	TIM3 CLK EN	TIM2 CLK EN	TIM1 CLK EN	TIM3 BRG [7:0]	TIM2 BRG [7:0]	TIM1 BRG [7:0]

Далее необходимо установить значение, до которого будет считать таймер (основание счётчика), в регистре MDR_TIMERx->ARR, а также значение делителя входных тактовых импульсов, записав необходимое значение в регистр MDR_TIMERx->PSG. Кроме того необходимо выбрать один из трёх режимов работы (рисунок 3.3) в регистре MDR_TIMERx->CNTRL, установив биты CNT_MODE и DIR. В регистре MDR_TIMERx->CNT будет находится значение насчитанное таймером. Значения регистров CNT, PSG и ARR можно изменять даже во время работы счетчика, при этом их значения вступают в силу по CNT = ARR или CNT = 0, в зависимости от направления счета. Значение регистра основания счета (ARR) может вступить в силу мгновенно после записи его в регистр при условии установленного поля ARRB_EN = 1 (регистр CNTRL).

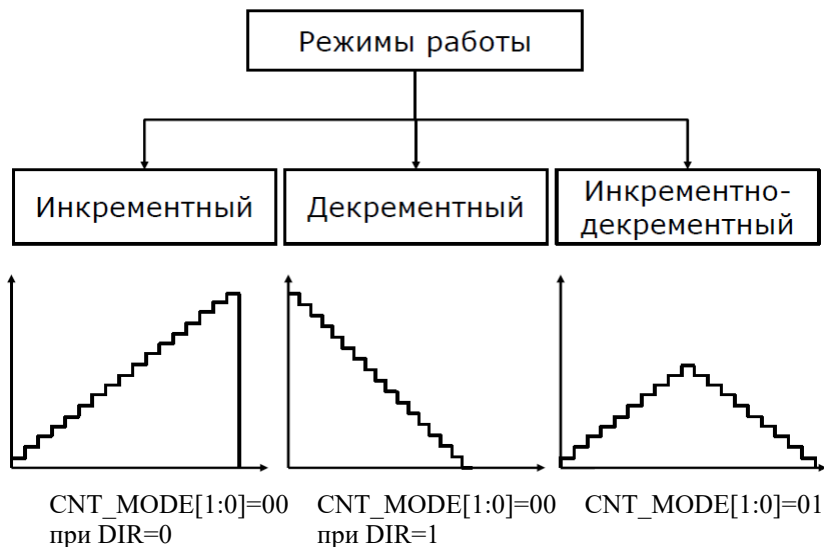


Рисунок 4.3. Режимы работы таймера.

В листинге 4.1 представлена настройка таймера в режиме счёта вверх/вниз

Листинг 4.1

```

MDR_RST_CLK->PER_CLOCK |= (1 << 14); // разрешить тактирование
таймера 1
MDR_RST_CLK->TIM_CLOCK|=(1<<24); // подключить такты HCLK к
таймеру 1
MDR_TIMER1->ARR=0xFFFF; //установить значение основания счёта
MDR_TIMER1->PSG=0xFFFF; // установить значение делителя
MDR_TIMER1->CNTRL=0; //очистка регистра CNTRL
MDR_TIMER1->CNTRL|=(1<<6); // режим счёта вверх вниз
MDR_TIMER1->CNT=0; //очистка счётного регистра
MDR_TIMER1->CNTRL|=1; // запуск таймера.
  
```

Задания для самостоятельной работы

Реализовать функцию мигания светодиодом на контакте PC1. Длительность включенного и выключенного состояния светодиода измеряется в секундах и равна номеру варианта. Примерный код программы представлен в листинге 4.2. В качестве источника счётных

импульсов использовать тактовый сигнал процессора HCLK. Значение делителя TIMx BRG[7:0] определяется по формуле: № варианта * 10, а значение регистра MDR_TIMER1->PSG по формуле: 65 000 + № варианта * 15. Основание счёта для всех вариантов равно 0xFFFF. Использовать таймер под номером 1, в режиме счёта вверх (инкрементный режим).

Листинг 4.2

```
void InitTimer (void);
void InitLED (void);
void DelayT(intd);
void LED_(intcmd);/* присcmd==0 – выключить светодиод, присcmd!=0
включить светодиод*/
unsigned int delay=100; /*у каждого варианта это число будет разным, его
необходимо подобрать*/
int main()
{
    InitTimer (); // функция инициализации таймера
    InitLED (); // инициализация PC1
    while(1)
    {
        DelayT(delay); // задержка
        LED_(1); // включить светодиод
        DelayT(delay); // задержка
        LED_(0); // выключить светодиод
    }
    return 1;
}
```

Примечание: функции InitTimer(), InitLED(), и LED_(intcmd) необходимо реализовать самостоятельно.

Функция DelayT(intd) должна иметь вид, представленный в листинге 4.3

Листинг 4.3

```
DelayT(intd)
{
    MDR_TIMER1->CNT=0; // очистка счётного регистра
    MDR_TIMER1->CNTRL|=1; // запустить таймер
    while(d!= MDR_TIMER1->CNT){} /* ожидаем пока значение
счётного регистра не станет равно аргументу d*/
    MDR_TIMER1->CNTRL&=(~1); // остановить таймер
}
```

Рекомендация: для точной настройки времени включенного и выключенного состояния светодиода используйте режим отладки. Запишите переменную delay в окно WatchWindow (см лабораторную №1) и изменяйте ее до тех пор, пока не добьётесь нужного результата.

На основании рассчитанного значения delay рассчитать **примерный** период счётных импульсов таймер по формуле:

$$T = \text{время включенного состояния светодиода} / \text{delay}$$

4. Контрольные вопросы

- 1 Что такое таймер?
- 2 Что может выступать источником счётных импульсов для таймера в микроконтроллере K1986BE92QI?
- 3 Что такое основание счёта таймера? Где оно настраивается?
- 4 В каких режимах счёта может работать таймер в микроконтроллере K1986BE92QI?

ГЕНЕРАЦИЯ ШИМ СИГНАЛА

Краткие теоретические сведения

Помимо режима счёт, таймеры в микроконтроллере K1986BE92QI могут осуществлять генерацию ШИМ сигнала. Основным параметром ШИМ сигнала является скважность. Скважность это отношение периода сигнала к длительности импульса.

Настройка таймера для генерации ШИМ сигнала.

Для генерации ШИМ сигнала необходимо сконфигурировать выводы микроконтроллера и задать режим генерации ШИМ сигнала в регистрах MDR_TIMERx->CHy_CNTRL, MDR_TIMERx->CHy_CNTRL1, MDR_TIMERx->CHy_CNTRL2 и MDR_TIMERx->CHy_DTG.(где x – номер таймера, y – номер канала) Для задания скважности выходного сигнала используются регистры MDR_TIMERx->CCRy.

Структура блока генерации ШИМ представлена на рисунке 4.5. Для включения режима сравнения для определенного канала необходимо в регистре управления каналом CHy_CNTRL записать “0” в поле CAPnPWM. При работе в режиме ШИМ выходной сигнал может формироваться на основании сравнения значения в регистре CCR и основного счетчика CNT или регистров CCR, CCR1 (если установлен бит CCR1_EN в регистре MDR_TIMERx->CHy_CNTRL2) и значения основного счетчика CNT. Следует отметить, что если используется регистр CCR1, то необходимо что бы его значение было больше чем в регистре CCR. Полученный сигнал может без изменения выдаваться на выводы CHxO и nCHxO. Либо с применением схемы «deadtime» генератора в формируются сигналы с мертвой зоной(см рисунок 4.4.). У каждого канала есть два выхода – прямой и инверсный (если на прямом канале 1 то на инверсном 0 и наоборот), вместе они позволяют формировать комплементарный ШИМ сигнал. Для каждого выхода формируется как сигнал для выдачи, так и сигнал разрешения выдачи, т.е. если выход канала должен всегда выдавать тот или иной уровень, то на выводе разрешения выдачи CHxOE (для прямого) и на CHxNOE (для инверсного) должны формироваться “1”. Если канал работает на вход (например, режим захвата), то там всегда должен быть "0" для прямого канала. Сигналы CHxOE и CHxNOE формируются по тем же принципам, что и просто выходные уровни, но у них есть собственные сигналы разрешения вывода SELOE и nSELOE, в которых можно

выбрать постоянный уровень, либо формируемый на основании сигнала REF.

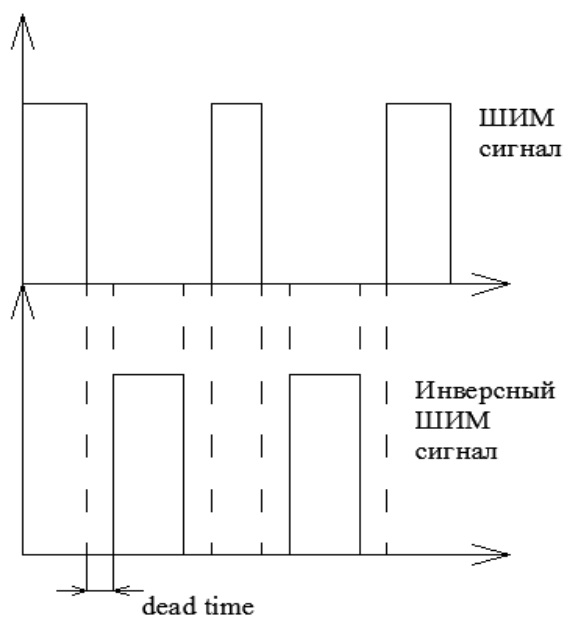


Рисунок 4.4.

В таблице 4.2 представлены биты регистра MDR_TIMERx->CHy_CNTRL, которые необходимы для настройки канала в режиме работы ШИМ. Бит CAPnPWM отвечает за режим работы канала. Битовое поле OCCM[2:0] определяет каким образом будет формироваться сигнал REF (см. рисунки 4.6-4.8).

Таблица 4.2.

Биты регистра MDR_TIMERx->CHy_CNTRL

№ бита	Название	Назначение
15	CAPnPWM	Режим работы канала Захват или ШИМ: 1 – канал работает в режиме Захват; 0 – канал работает в режиме ШИМ
11...9	OCCM[2:0]	Формат выработки сигнала REF в режиме ШИМ: Если бит CCR1_EN в регистре MDR_TIMERx->CCRy сброшен: 000 – всегда 0; 001 – 1, если CNT = CCR; 010 – 0, если CNT = CCR; 011 – переключение REF, еси CNT = CCR; 100 – всегда 0; 101 – всегда 1; 110 – 1, если CCR>CNT, иначе 0; 111 – 0, если CCR>CNT, иначе 1; Если CCR1_EN = 1: 000 – всегда 0; 001 – 1, если CNT = CCR или CNT = CCR1; 010 – 0, если CNT = CCR или CNT = CCR1; 011 – переключение REF, если CNT =CCR или CNT =CCR1; 100 – всегда 0; 101 – всегда 1; 110 – 1, если CCR>CNTили CCR1<CNT, иначе 0; 111 – 0, если CCR>CNTили CCR1<CNT, иначе 1; При условии, что CCR < CCR1

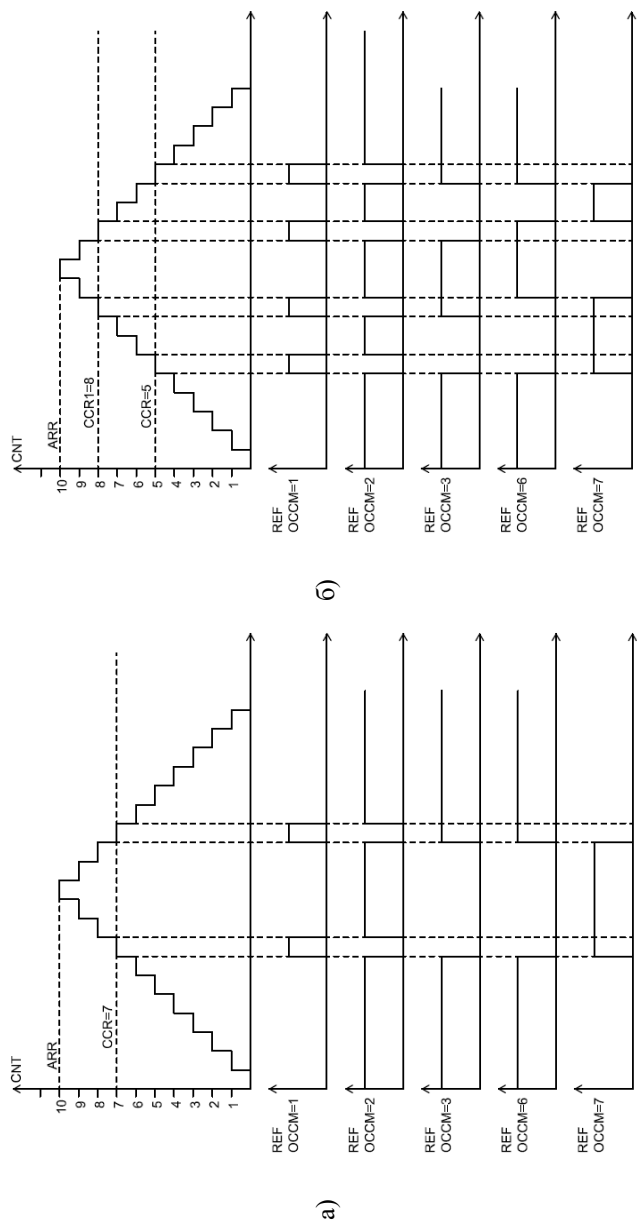


Рисунок 4.6. Сигнал REF при счёте таймера вверх/вниз и различных значениях битового поля OCCM[2:0]
а- регистр CCR1 не используется; б- регистр CCR1 используется

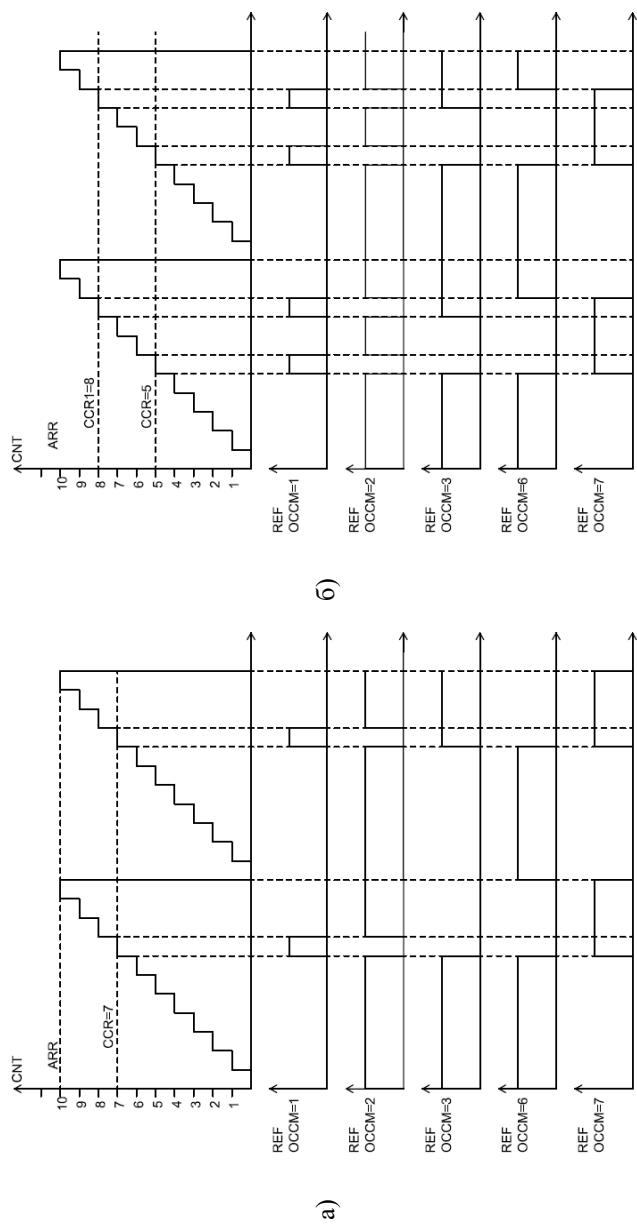


Рисунок 4.7. Сигнал REF при счёте таймера вверх и различных значениях битового поля OCCM[2:0]
а- регистр CCR1 не используется; б- регистр CCR1 используется

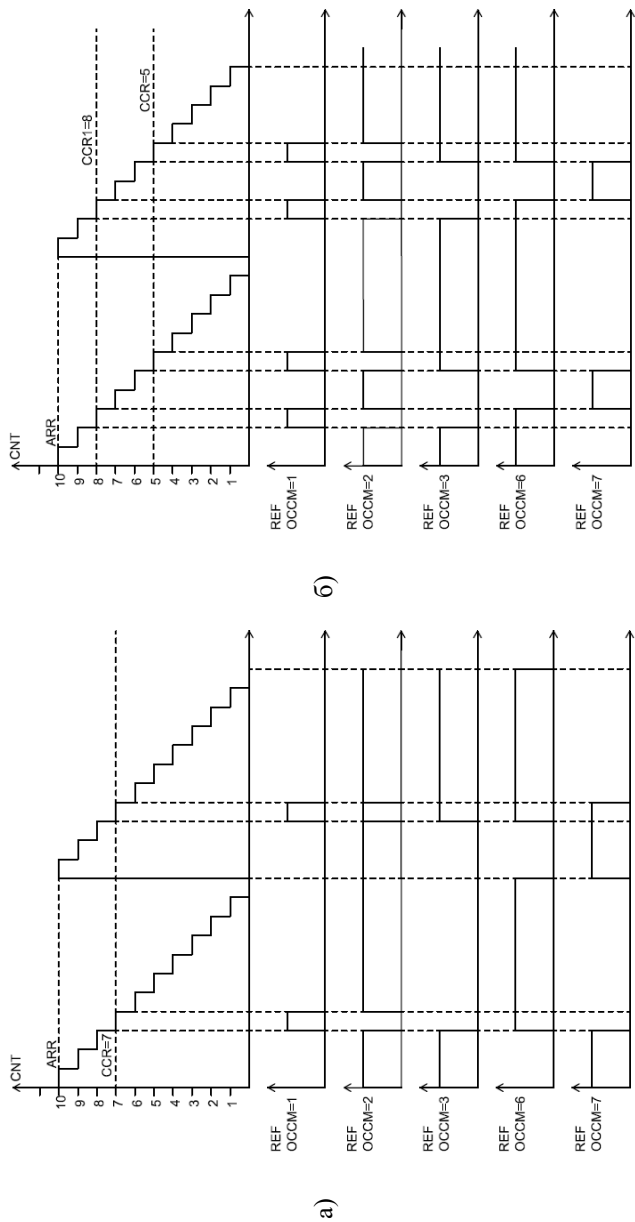


Рисунок 4.8. Сигнал REF при счёте таймера вниз и различных значениях битового поля OCCM[2:0]
а- регистр CCR1 не используется; б- регистр CCR1 используется

В таблице 4.3 представлены биты регистра MDR_TIMERx->CHy_CNTRL1, которые необходимы для настройки канала в режиме работы ШИМ.

Таблице 4.3.

Описание битов регистра MDR_TIMERx->CHy_CNTRL1

№ бита	Название	Назначение
12	NINV	Режим выходной инверсии инверсного канала nCHy: 0 – выход не инвертируется; 1 – выход инвертируется
11..10	NSELO[1:0]	Режим работы выхода инверсного канала nCHy: 00 – всегда на выход выдается 0, канал на выход не работает; 01 – всегда на выход выдается 1, канал всегда работает на выход; 10 – на выход выдается сигнал REF; 11 - на выход выдается сигнал с DTG
9...8	NSELOE[1:0]	Режим работы разрешающего сигнала инверсного канала nCHy на выход 00 – всегда на nCHyOE выдается 0, канал на выход не работает; 01 – всегда на nCHyOE выдается 1, канал всегда работает на выход; 10 – на nCHyOE выдается сигнал REF, при REF = 0 третье состояние, при REF = 1 выход; 11 - на nCHyOE выдается сигнал с DTG, при nCHyOE = 0 третье состояние, при nCHyOE = 1 выход
4	INV	Режим выходной инверсии прямого канала CHy: 0 – выход не инвертируется; 1 – выход инвертируется
3...2	SELO[1:0]	Режим работы выхода прямого канала CHy: 00 – всегда на выход выдается 0, канал на выход не работает; 01 – всегда на выход выдается 1, канал всегда работает на выход; 10 – на выход выдается сигнал REF; 11 - на выход выдается сигнал с DTG
1...0	SELOE[1:0]	Режим работы разрешающего сигнала прямого

		канала СHy на выход: 00 – всегда на СHyOE выдается 0, канал на выход не работает; 01 – всегда на СHyOE выдается 1, канал всегда работает на выход; 10 – на СHyOE выдается сигнал REF, при REF = 0 третье состояние, при REF = 1 выход; 11 - на СHyOE выдается сигнал с DTG, при СHyOE = 0 третье состояние, при СHyOE = 1 выход
--	--	---

В таблице 4.4 представлены биты регистра MDR_TIMERx->CHy_CNTRL2, которые необходимы для настройки канала в режиме работы ШИМ.

Таблица 4.4

Описание битов регистра MDR_TIMERx->CHy_CNTRL2

№ бита	Название	Назначение
3	CRRRLD	Разрешение обновления регистров CCR и CCR1: 0 – обновление возможно в любой момент времени; 1 – обновление будет осуществлено только при CNT = 0
2	CCR1_EN	Разрешение работы регистра CCR1: 0 – CCR1 не используется; 1 – CCR1 используется

В таблице 4.5 представлены биты регистра MDR_TIMERx->CHy_DTG, которые необходимы для настройки канала в режиме работы ШИМ.

Таблица 4.5

Описание битов регистра MDR_TIMERx->CHy_DTG

№ бита	Название	Назначение
15...8	DTG[7:0]	Основной делитель частоты. Задержка $DTG_{del} = DTG * (DTG_x + 1)$
4	EDTS	Частота работы DTG: 0 – TIM_CLK; 1 – FDTs
3...0	DTGx[3:0]	Предварительный делитель частоты DTGx

В листинге 4.2. представлена программ настройки первого канала таймер 1 в режиме ШИМ с прямым и инверсным каналом без паузы dead-time.

Листинг 4.1

```
MDR_RST_CLK->TIM_CLOCK |=
RST_CLK_TIM_CLOCK_TIM1_CLK_EN;
MDR_RST_CLK->PER_CLOCK|=(1<<14);
MDR_TIMER1->ARR=1000;
MDR_TIMER1->PSG=0x009;
MDR_TIMER1->CNTRL=0;
MDR_TIMER1->CNTRL|=( (1<<6));// счётвверх/вниз
MDR_TIMER1->CH1_CNTRL|= (6<<9);/*REFформируетсяпоусловию: 1,
приCCR<CNT,иначе 0*/
MDR_TIMER1->CH1_CNTRL|= ( (2<<10)//
| (1<<8)//
| (2<<2)//
| (1<<0));//
MDR_TIMER1->CNTRL=1;// запуск счётчика
```

Задания для самостоятельной работы

1 Реализовать функцию настройки таймера для генерации ШИМ сигнала voidInitPWM(void) и функцию initPortPWM() настройки линии 1 порта A. Использовать таймер №1 и первый канал сравнения. Значение регистра MDR_TIMER1->PSG рассчитывается по формуле: $N \cdot 10$, а регистра MDR_TIMER1->ARR по формуле $N \cdot 1000$ (где N–номер варианта). Режим счёта таймер для каждого варианта определяется из последовательности: 1 вариант – счёт вверх/вниз, 2 вариант – счёт вверх, 3 вариант – счёт вниз, 4 вариант – счёт вверх/вниз и тд. Для нечётных вариантов сигнал REF формируется по условию: 1, если $CCR > CNT$, иначе 0, а для чётных по условию: 0, если $CCR > CNT$, иначе 1;Регистр CCR1 не используется для всех вариантов.

2 Реализовать функцию intmain() в соответствии с листингом 4.2

```
int main()
{
    initPortPWM();
    initTimerPWM();
    while(1) {}
    return 1;
}
```

3 Скомпилировать программу записать её в контроллер и запустить режим отладки

4 Используя инструмент System Viewer Windows, в регистр MDR_TIMER1->CCR установить половинное значение регистра MDR_TIMER1->ARR

5 Используя осциллограф, определить длительность импульса выходного сигнала и рассчитать его скважность.

6 Поочерёдно установить в регистр MDR_TIMER1->CCR значения рассчитываемые по формуле $2*((MDR_TIMER1->ARR)/N*10)$, $4*((MDR_TIMER1->ARR)/(N*10))$ и $8*((MDR_TIMER1->ARR)/(N*10))$ и повторить действия описанные в пункте 5.

. Контрольные вопросы

1. Что такое скважность импульса?
2. Что такое dead-time?
3. Что такое комплементарный ШИМ сигнал?

АНАЛОГО-ЦИФРОВОЙ ПРЕОБРАЗОВАТЕЛЬ

Краткие теоретические сведения об АЦП

Аналого-цифровой преобразователь (АЦП) – это устройство, которое позволяет преобразовать аналоговый сигнал в цифровой код. Основными характеристиками АЦП являются разрядность, частота преобразования, максимальное входное напряжение, тип измеряемого напряжения: однополярное или биполярное.

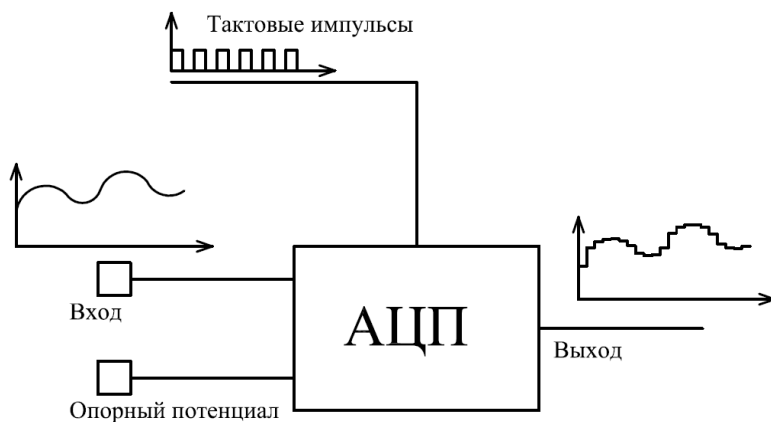


Рисунок 6.1.

Значение напряжения, соответствующее единичному числовому коду, определяется по формуле:

$$\frac{U_{\text{ref}}}{2^{\text{разрядность}}}$$

где U_{ref} - опорное напряжение.

В микроконтроллере K1986BE92QI имеется два АЦП, разрядность которых составляет 12 бит, а частота преобразования может достигать 512820 преобразований в секунду (1,95 мкс на одно преобразование). Скорость преобразования зависит от частоты тактовых импульсов поступающих на АЦП. Для одного преобразования необходимо не мене

28 тактовых импульсов. **Максимальная частота тактирования АЦП не может превышать 14МГц.**

Максимальное входное напряжение которое может измерить АЦП микроконтроллера K1986BE92QI составляет 3.3В однополярного напряжения.

ВНИМАНИЕ! При подаче на АЦП микроконтроллера K1986BE92QI напряжения боле 3.3 В он может быть повреждён.

Многие АЦП в том числе в микроконтроллере K1986BE92QI имеют несколько входов, что позволяет **в разное время** производить преобразование нескольких аналоговых сигналов одним АЦП. Принципиально АЦП с несколькими входами представлен на рисунке 6.2, каждый аналоговый сигнал поступает на аналоговый мультиплексор, который соединяет вход АЦП с указанным входом.

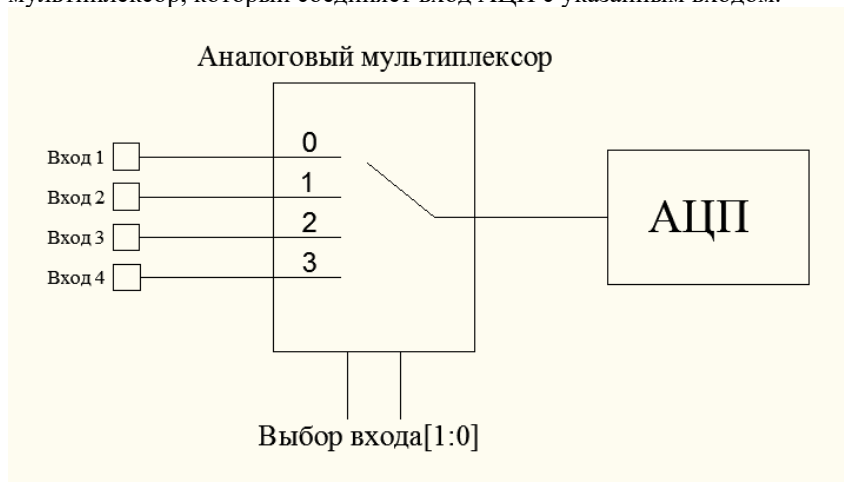


Рисунок 6.2. АЦП с несколькими входами.

Каждый АЦП в микроконтроллере K1986BE92QI имеет 16 входов, соединённых с контактами микросхемы. Выбор нужного входа осуществляется через специальный регистр. Следует отметить, что возможно настроить два АЦП микроконтроллера, таким образом, что бы они начинали отцифровывать один и тот же вход с заданной задержкой относительно начала работы одного из АЦП (т.е если АЦП1 начал преобразовывать вход 1, то АЦП2 начнёт отцифровывать через заданную задержку), это позволяет увеличить скорость преобразования в 2 раза.

Настройка АЦП в микроконтроллере K1986BE92QI

Настройка контактов микроконтроллера. Для входа АЦП используется 16 контактов порта D. Необходимы контакт нужно настроить в аналоговом режиме работы, для этого необходимо порт подключить к тактированию в регистре MDR_RST_CLK->PER_CLOCK (установить бит 24) а в регистрах MDR_PORTD->OE и MDR_PORTD->ANALOG в битах соответствующих необходимым входам должны быть записаны 0.

Настройка тактирования АЦП. Перед настройкой АЦП, его, как и любой периферийный модуль в микроконтроллере K1986BE92QI необходимо подключить к тактированию, возведя соответствующий бит в регистре MDR_RST_CLK->PER_CLOCK. Для АЦП – это бит 17. Далее необходимо для нужного АЦП выбрать источник тактирования: тактовый сигнал CPU_CLK, который представляет собой сигнал процессора HCLK(см. рисунок 5.3) делённый на значение битового поля Cfg_REG_DIVCLK[3:0] регистраMDR_ADC->ADCx_CFG (где x–номер АЦП), или сигнал ADC_CLK, формирование которого представлено на рисунке 6.3. Биты регистра MDR_RST_CLK->ADC_MCO_CLOCK описаны в таблице 6.1. В качестве источников для сигнал ADC_C1 могут использоваться тактовые сигналы CPU_C1, CPU_C2 (см. рисунок 6.3), а также сигналы тактовой частот USB модуля USB_C1 и USB_C2 (в данной работе они рассматриваться не будут). Для ADC_C2 источниками тактирования могут быть выбраны внешний генератор LSE, внутренний генератор LSI(40 кГц), внутренний генератор HIS (8МГц) или ADC_C1. После настройки сигналаADC_Cx необходимо установить бит ADC CLK EN для разрешения выдачи этих тактовых сигналов. Схема тактирования АЦП1 и АЦП2 представлена на рисунке 6.3.

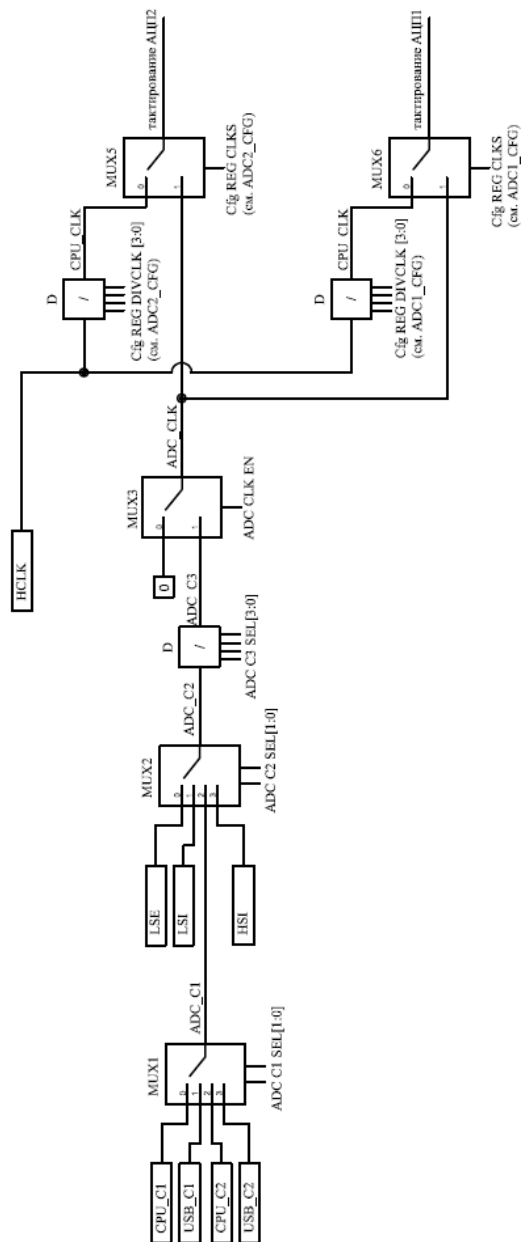


Рисунок 6.3. Схема тактирования АЦП1 и АЦП2

Таблица 6.1.
MDR_RST_CLK-

Описание битов регистра

>ADC MCO CLOCK

№ бита	Название	Назначение
13	ADC CLK EN	Бит разрешения выдачи тактовой частоты ADC CLK: 0 – запрещен; 1 – разрешен.
11...8	ADC C3 SEL[3:0]	Биты выбора делителя для ADC_C3: 0xxx – ADC_C3 = ADC_C2 1000 - ADC_C3 = ADC_C2 / 2 1001 - ADC_C3 = ADC_C2 / 4 1010 - ADC_C3 = ADC_C2 / 8 ... 1111 - ADC_C3 = ADC_C2 / 256
5...4	ADCC2SEL[1:0]	Биты выбора источника для ADC_C2: 00 – LSE 01 – LSI 10 – ADC_C1 11 – HSI_C1
1...0	ADCC1SEL[1:0]	Биты выбора источника для ADC_C1: 00 – CPU_C1 01 – USB_C1 10 – CPU_C2 11 – USB_C2

Внимание! При настройке тактирования АЦП важно помнить, что максимальная частота тактовых импульсов не должна превышать 14МГц.

Настройка источника опорного напряжения. В качестве источника опорного напряжения могут использоваться:

- внутренний источник опорного напряжения (AUсс и AGND),;
- внешний источник опорного напряжения (от контактов микроконтроллера ADC0_REF+ и ADC1_REF-).

Параметры источника опорного напряжения для АЦП в микроконтроллере K1986BE92Q1 приведены в таблице 6.2.

Таблица 6.2

Параметры источника опорного напряжения

	Обозначение	Предельно допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
Напряжение источника питания АЦП, ЦАП, В	U_{CCA}	2.4	3.6	-	4
Напряжение нижней границы внешнего опорного напряжения АЦП, В, при: ADC1_Cfg_M_REF = 1 или ADC2_Cfg_M_REF = 1	U_{ADC1_REF} -	0	U_{CCA} -2,4	-	4
Напряжение верхней границы внешнего опорного напряжения АЦП, В, при: ADC1_Cfg_M_REF = 1 или ADC2_Cfg_M_REF = 1	U_{ADC0_REF} +	2.4	U_{CCA}	-	4

Выбор источника опорного напряжения осуществляется битом CfgM_REF в регистре MDR_ADC->ADCx_CFG (где x – номер АЦП).

Настройка режимов работы АЦП. Настройка режима работы АЦП1 и АЦП2 производится в регистрах MDR_ADC->ADC1_CFG и MDR_ADC->ADC2_CFG соответственно. Описание битов этих регистров приведено в таблицах 6.3 и 6.4. В таблице 6.5 представлено описание битов регистра MDR_ADC->ADCx_STATUS (где x – номер АЦП).

Каждый АЦП в микроконтроллере поддерживает следующие режимы работы:

1. Режим одиночного преобразования выбранного канала. Этот режим задаётся, когда сброшен бит Cfg_REG_SAMPLE. Преобразование в этом режиме осуществляется только после записи бита Cfg_REG_GO, после завершения преобразования этот бит сбрасывается автоматически.
2. Режим последовательного преобразования, т.е. каждое преобразование запускается после завершения предыдущего.

- Этот режим выбирается, когда установлен бит Cfg_REG_SAMPLE. Между преобразования можно установить паузу, которая задаётся битовым полем Delay_Go[2:0]
3. Режим последовательного преобразования нескольких каналов. Этот режим настраивается установкой бита Cfg_REG_CHCH, а в регистре MDR_ADC->ADCx_CHSEL (где x – номер АЦП) необходимо указать какие каналы необходимо преобразовать, для этого необходимо установить бит, соответствующий номеру канала.
 4. Режим синхронной работы двух АЦП. Для этого режима необходимо возвести бит Cfg_Sync_Convert. При этом два АЦП производят преобразования канал выбранного в битовом поле Cfg_REG_CHS[4:0] регистра MDR_ADC->ADC1_CFG, или последовательности каналов, если выбран режим последовательного преобразования нескольких каналов. Задержка между началом преобразования АЦП1 и АЦП2 задаётся битовым полем Delay_ADC[3:0].
 5. Режим контроля уровня. Если значение MDR_ADC->ADCx_RESULT (где x – номер АЦП) выходит за пределы значений установленных в регистрах MDR_ADC->ADCx_H_LEVEL и MDR_ADC->ADCx_L_LEVEL (где x – номер АЦП), то установиться бит Flg_REG_AWOFEN
 6. Комбинированный режим. Комбинация из вышеперечисленных режимов.

Таблица 6.3.

Описание битов регистра MDR_ADC->ADC1_CFG

№ бита	Название	Назначение
31...28	Delay ADC [3:0]	Задержка между началом преобразования ADC1 и ADC2 при последовательном переборе, либо работе на один канал: 0000 – 1 такт CPU_CLK; 0001 – 2 такта CPU_CLK; ... 1111 – 16 тактов CPU_CLK
27...25	Delay Go [2:0]	Дополнительная задержка перед началом преобразования после выбора канала: 000 – 1 такт CPU_CLK; 001 – 2 такта CPU_CLK; ... 111 – 8 тактов CPU_CLK

24...21	TR [3:0]	Подстройка опорного напряжения. Смотрите диаграмму на Рисунок 92. Зависимость источника опорного напряжения от подстройки
20	SEL VREF	Выбор для оцифровки источника опорного напряжения на 1,23 В: 0 – не выбран; 1 – выбран. Должен использоваться совместно с выбором канала Cfg_REG_CHS = 30
19	SEL TS	Выбор для оцифровки датчика температуры: 0 – не выбран; 1 – выбран. Должен использоваться совместно с выбором канала Cfg_REG_CHS = 31
18	TS BUF EN	Включения выходного усилителя для датчика температуры и источника опорного напряжения: 0 – выключен; 1 – включен. Используется при TS_EN = 1
17	TS EN	Включения датчика температуры и источника опорного напряжения: 0 – выключен; 1 – включен. При включении датчика температуры и источника опорного напряжения выходной сигнал стабилизируется в течение времени 1 мс
16	Cfg Sync Conver	Запускает работу двух АЦП одновременно, при этом биты конфигурации второго АЦП, такие как Cfg_REG_DIVCLK, Cfg_REG_ADON, Cfg_M_REF и Cfg_REG_CHS берутся из регистра конфигурации первого: 0 – независимые АЦП; 1 – синхронные АЦП

15...12	Cfg REG DIVCLK [3:0]	Выбор коэффициента деления частоты процессора: 0000 – CPU_CLK = HCLK; 0001 – CPU_CLK = HCLK/2; 0010 – CPU_CLK = HCLK/4; 0011 – CPU_CLK = HCLK/8; ... 1011 – CPU_CLK = HCLK/2048 Остальные CPU_CLK = HCLK
11	Cfg M_REF	Выбор источника опорных напряжений: 0 – внутренне опорное напряжение (от AUсс и AGND); 1 – внешнее опорное напряжение (от ADC0_REF+ и ADC1_REF-)
10	Cfg REG RNGC	Разрешение автоматического контроля уровней: 0 – не разрешена; 1 – разрешена выработка флага при выходе за диапазон в регистрах границы
9	Cfg REG CHCH	Выбор переключения каналов: 0 – используется только выбранный канал; 1 – переключение включено (перебираются каналы, выбранные в регистре выбора канала)
8...4	Cfg REG CHS [4:0]	Выбор аналогового канала, по которому поступает сигнал для преобразования: 00000 – 0 канал; 00001 – 1 канал; ... 11111 – 31 канал
3	Cfg REG SAMPLE	Выбор способа запуска АЦП: 0 – одиночное; 1 – последовательное. Автоматический запуск после завершения предыдущего преобразования
2	Cfg REG CLKS	Выбор источника синхросигнала CLK работы ADC: 0 – CPU_CLK; 1 – ADC_CLK
1	Cfg REG GO	Начало преобразования. Запись “1” начинает процесс преобразования, сбрасывается автоматически
0	Cfg REG	Включение АЦП: 0 – выключено;

	ADON	1 – включено
--	------	--------------

Таблица 6.4.

Описание битов регистра MDR_ADC->ADC2_CFG

№ бита	Название	Назначение
27...25	Delay Go [2:0]	Задержка перед началом следующего преобразования после завершения предыдущего при последовательном переборе каналов: 000 – 0 тактов CPU_CLK 001 – 1 такт CPU_CLK ... 111 – 7 тактов CPU_CLK
18	ADC2 OP	Выбор источника опорного напряжения 1,23 В: 0 – внутренний (неточный); 1 – от датчика температуры (точный). Необходимо чтобы бит TS_EN был равен 1
17	ADC1 OP	Выбор источника опорного напряжения 1,23 В: 0 – внутренний (неточный); 1 – от датчика температуры (точный). Необходимо чтобы бит TS_EN был равен 1
15...12	Cfg REG DIVCLK [3:0]	Выбор коэффициента деления частоты процессора: 0000 – CPU_CLK = HCLK; 0001 – CPU_CLK = HCLK/2; 0010 – CPU_CLK = HCLK/4; 0011 – CPU_CLK = HCLK/8; ... 1011 – CPU_CLK = HCLK/2048 Остальные CPU_CLK = HCLK;
11	Cfg M_REF	Выбор источника опорных напряжений: 0 – внутренне опорное напряжение (от AUSS и AGND); 1 – внешнее опорное напряжение (от ADC0_REF+ и ADC1_REF-)
10	Cfg REG RNGC	Разрешение автоматического контролирования уровней: 1 – разрешено, выработка прерывания при выходе за диапазон в регистрах границы обработки; 0 – не разрешено

9	Cfg REG CHCH	Выбор переключения каналов: 0 – используется только выбранный канал; 1 – переключение включено (перебираются каналы, выбранные в регистре выбора канала)
8...4	Cfg REG CHS [4:0]	Выбор аналогового канала, по которому поступает сигнал для преобразования: 00000 – 0 канал; 00001 – 1 канал; ... 11111 – 31 канал
3	Cfg REG SAMPLE	Выбор способа запуска АЦП: 0 – одиночное; 1 – последовательное. Автоматический запуск после завершения предыдущего преобразования.
2	Cfg REG CLKS	Выбор источника синхросигнала CLK работы ADC: 0 – CPU_CLK; 1 – ADC_CLK
1	Cfg REG GO	Начало преобразования. Запись “1” начинает процесс преобразования, сбрасывается автоматически
0	Cfg REG ADON	Включение АЦП: 0 – выключено; 1 – включено

Таблица 6.5.

Описание битов регистра MDR_ADC->ADCx STATUS

№ бита	Название	Назначение
4	EOIF_IE	Флаг разрешения генерирования прерывания по событию Flg_REG_EOIF: 0 – прерывания не генерируется; 1 – прерывание генерируется
3	AWOIF_IE	Флаг разрешения генерирования прерывания по событию Flg_REG_AWOIFEN: 0 – прерывания не генерируется; 1 – прерывание генерируется
2	FlgREGEO CIF	Флаг выставляется, когда закончено преобразования и данные еще не считаны.

		Очищается считыванием результата из регистра ADCx_RESULT: 1 – есть готовый результат преобразования; 0 – нет результата
1	FlgREGAWOIFEN	Флаг выставляется, когда результат преобразования выше верхней или ниже нижней границы автоматического контролирования уровней. Сбрасывается только при записи нуля в данный бит регистр флагов: 0 – результат в допустимой зоне; 1 – вне допустимой зоны
0	FlgREGOVERWRITE	Данные в регистре результата были перезаписаны, данный флаг сбрасывается только при записи нуля в данный бит регистр флагов: 0 – не было события перезаписи несчитанного результата; 1 – был результат преобразования, который не был считан

Результат преобразования АЦП. Сигналом о завершении преобразования является бит Flg_REG_EOCIF в регистре MDR_ADC->ADCx_STATUS (где x – номер АЦП). Этот бит сбрасывается автоматически после того как считывается результат преобразования из регистра MDR_ADC->ADCx_RESULT (где x – номер АЦП), описание которого представлено в таблице 6.6.

Таблица 6.6.

Описание битов регистра MDR_ADC->ADCx_RESULT

№ бита	Название	Назначение
20...16	CHANNEL [11:0]	Канал результата преобразования
11...0	RESULT [11:0]	Значение результата преобразования

В листинге 6.1 представлена программа настройки АЦП1 в режиме последовательного преобразования каналов 0,1 и 2

Листинг 6.1.

```
MDR_RST_CLK->PER_CLOCK|= ((1<< 24)| //разрешение тактирования
порта D
(1 << 17)); //разрешение тактирования АЦП
// настройка линии на аналоговый режим работы
MDR_PORTD->OE      |= ((0 << 2)); //направление передачи данных -
Вход
```

```

MDR_PORTD->ANALOG |= ((0 << 2)); //режим работы контроллера
Аналоговый
//настройка АЦП
MDR_RST_CLK->ADC_MCO_CLOCK|=(1<<13);
MDR_ADC->ADC1_CFG|=((0<<25)//задержка между преобразованиями
каналов
(0<<21)// подстройка опорного напряжения
(0<<12)// делитель тактов частоты процессора
(0<<11)// выбран источника опорного напряжения (внутр)
(1<<9)// автоматическое переключение каналов
(1<<3)// последовательное преобразование
(0<<2)//источник синхронизации АЦП (такты CPU)
);
MDR_ADC->ADC1_CHSEL=((1<<0)| // указываются каналы для
поочерёдного
(1<<1)| //преобразования
(1<<2));
MDR_ADC->ADC1_CFG|=3; // включение АЦП и запуск преобразования

```

Задания для самостоятельной работы

Реализовать функцию инициализации АЦП voidInitADC (void) в соответствии с таблицей задания 6.7. Использовать внутренний источник опорного напряжения. Для тактирования АЦП использовать тактовый сигнал процессора HCLK.

Таблица 6.7

Таблица задания

№	Номера каналов для преобразования	Комбинация режимов(смстр. 57-58)	Примечание
1	4	2 и 5	Контроль уровня: 4000- верхняя граница 1000- нижняя граница
2	5	2 и 4	

3	2,3,4,5	2 и 3	Значение битового поля Cfg REG DIVCLK =11 ₁₀
4	2,4,6	3 и 4	
5	6	2 и 5	Контроль уровня: 3500- верхняя граница 500- нижняя граница
6	4	2 и 4	
7	2,4,6	2 и 3	Значение битового поля Cfg REG DIVCLK =11 ₁₀
8	2,3,4,5	3 и 4	
9	2	2 и 5	Контроль уровня: 2700- верхняя граница 800- нижняя граница
10	3	2 и 4	
11	4,5,6	2 и 3	Значение битового поля Cfg REG DIVCLK =11 ₁₀
12	4,5,6	3 и 4	
13	5	2 и 5	Контроль уровня: 3100- верхняя граница 800- нижняя граница
14	2	2 и 4	
15	3,5,6	2 и 3	Значение битового поля Cfg REG DIVCLK =11 ₁₀
16	2,3,6	3 и 4	
17	5	2 и 5	Контроль уровня: 3100- верхняя граница 800- нижняя граница
18	2	2 и 4	

19	3,4,6	2 и 3	Значение битового поля Cfg REG DIVCLK =11 ₁₀
20	2,3,5	3 и 4	

Таблица 6.8

Подключения каналов АЦП к контактам микросхемы K1986BE92QI

Контакт микросхемы	канал АЦП
PD2	канал 2
PD3	канал 3
PD4	канал 4
PD5	канал 5
PD6	канал 6

Реализовать функцию main:

- для вариантов с комбинацией 2 и 5

```
voidLED_1(intcmd); // ф-цияуправления светодиодом 1 (PC0)
```

```
voidLED_2(intcmd); // ф-цияуправления светодиодом 2 (PC1)
```

```
int adc_value1=0;
```

```
int main
```

```
{
```

```
    initADC();
```

```
    initLED();// ф-ция инициализации контактов PC0 и PC1
```

```
// установить границы значений
```

```
    MDR_ADC->ADC1_H_LEVEL=4000;
```

```
    MDR_ADC->ADC1_L_LEVEL=200;
```

```
    while(1)
```

```
    {
```

```
        // проверить установлен ли флаг выхода за верхнюю или нижнюю границу
```

```
        if(MDR_ADC->ADC1_STATUS&
```

```
        ADC_STATUS_FLG_REG_AWOIFEN)
```

```
        {
```

```
            // прочитайте результат преобразования
```

```
            adc_value1=((MDR_ADC->ADC1_RESULT)&0xFFFF);
```

```
            if(adc_value1<MDR_ADC->ADC1_L_LEVEL)
```

```
            {
```

```
                LED_1(1);
```

```
            LED_2(0);
```

```
            }
```

```

        if(adc_value1>MDR_ADC->ADC1_H_LEVEL)
        {
            LED_2(1);
            LED_1(0);
        }
        MDR_ADC->ADC1_STATUS&=~(ADC_STATUS_FLG_REG_AWOIFEN); //
        сбросить флаги выхода за границу
    }
    else
    {
        LED_1(0);
        LED_2(0);
    }
}
return 1;
}

```

- для вариантов с комбинацией 2 и 4

```

int adc_value1=0;
int adc_value2=0;
int main
{
    initADC();
    while(1)
    {
        if(MDR_ADC->ADC1_STATUS&
        ADC_STATUS_FLG_REG_EOCIF)
        {
            adc_value1=(MDR_ADC->ADC1_RESULT)&0xFFF;
        }
        if(MDR_ADC->ADC2_STATUS&
        ADC_STATUS_FLG_REG_EOCIF)
        {
            adc_value2=(MDR_ADC->ADC2_RESULT)&0xFFF;
        }
    }
    return 1;
}

```

- для вариантов с комбинацией 2 и 3

```

int adc_value1=0;
int adc_values[3];
int ch=0;
int main
{
    initADC();
    while(1)
    {
        if(MDR_ADC->ADC1_STATUS&
        ADC_STATUS_FLG_REG_EOCIF)
        {
            adc_value1=(MDR_ADC->ADC1_RESULT)&0xFFF;
            ch=(MDR_ADC->ADC1_RESULT)>>16;// номерканала
            switch (ch)
            {
                case 0:
                    adc_values[0]=adc_value1;
                    break;
                case 1:
                    adc_values[1]=adc_value1;
                    break;
                case 2:
                    adc_values[2]=adc_value1;
                    break;
            }
        }
    }
    return 1;
}

```

- для вариантов с комбинацией 3 и 4

```

int adc_values1[3];
int adc_values2[3];
int ch=0;
int main
{
    initADC();
    while(1)
    {
        if(MDR_ADC->ADC1_STATUS& ADC_STATUS_FLG_REG_EOCIF)
        {

```

```

        adc_value1=(MDR_ADC-
>ADC1_RESULT)&0xFFF;
        ch=(MDR_ADC->ADC1_RESULT)>>16; ;//
номерканала
        switch (ch)
        {
                case 0:
                adc_values1[0]=adc_value1;
                break;
                case 1:
                adc_values1[1]=adc_value1;
                break;
                case 2:
                adc_values1[2]=adc_value1;
                break;
        }
    }
    if(MDR_ADC->ADC2_STATUS&
ADC_STATUS_FLG_REG_EOCIF)
    {
        adc_value2=(MDR_ADC-
>ADC2_RESULT)&0xFFF;
        ch=(MDR_ADC->ADC1_RESULT)>>16; ;//
номерканала
        switch (ch)
        {
                case 0:
                adc_values2[0]=adc_value2;
                break;
                case 1:
                adc_values2[1]=adc_value2;
                break;
                case 2:
                adc_values2[2]=adc_value2;
                break;
        }
    }
    return 1;
}

```

Контрольные вопросы

1. Что такое АЦП?
2. Какие основные характеристики есть у АЦП?
3. Что такое опорное напряжение?
4. Какие виды источников опорного напряжения могут использоваться для АЦП в микроконтроллере K1986BE92QI?
5. Как рассчитать значение напряжения соответствующее самому младшему разряду данных АЦП?
6. Какая максимальная частота тактирования допустима для АЦП в микроконтроллере K1986BE92QI?
7. Какая разрядность у АЦП в микроконтроллере K1986BE92QI?
8. Какие режимы работы есть у АЦП в микроконтроллере K1986BE92QI?

ИСКЛЮЧЕНИЯ И ПРЕРЫВАНИЯ

Краткие теоретические сведения

При работе процессора микроконтроллера часто возникают ситуации, когда требуется прервать выполнение текущей программы и перейти к подпрограмме, обеспечивающей необходимую реакцию системы на создавшиеся обстоятельства. Такие ситуации называются прерываниями (или исключениями) в зависимости от причин, вызывающих их возникновение.

Исключения являются нештатными ситуациями (ошибки), возникающими при выполнении программы процессором.

Прерывания или IRQ(InerruptReQuest–запрос прерывания) – это исключения, вызываемые периферийными устройствами (таймеры, АЦП, компараторы и тд.) или программными запросами. Все прерывания асинхронны (т.е. могут происходить в любой момент времени) по отношению к выполняемым инструкциям.

Основные термины при работе с прерываниями:

Приоритет прерывания (исключения). При одновременном поступлении запросов на прерывание (исключение) выполняться будет то прерывание (исключение) чей приоритет выше. Самый высокий приоритет прерывание (исключения) в микроконтроллере K1986BE92QI имеет значение -3 а самый низкий 31.

Маска прерывания– это числовой код, который при записи в соответствующий регистр разрешает необходимое прерывание.

Маскируемое прерывание – прерывание, которое требует для своей работы разрешения, в виде записи *маски прерывания* в регистр, который отвечает за разрешение прерываний.

Активное прерывание- это прерывание, которое находится в процессе обработки.

Вектор прерывания – это адрес инструкции, с которой процессор начнет выполнять программу обработки прерывания,

Исключения в микроконтроллере K1986BE92QI

Исключение **RESET** вызывается при включении питания и горячем сбросе.

Исключение **Hard Fault** возникает при ошибке при обработке исключений или потому, что исключение не может быть обработано каким-либо другим механизмом. Hard fault имеет фиксированный

приоритет -1, означающий, что оно имеет больший приоритет, чем любое из исключений с конфигурируемым приоритетом.

Исключение **Memory Management fault** возникает при срабатывании по защите памяти. Блок MPU или фиксированные защитные настройки определяют это исключение, как для данных, так и для инструкций. Исключение используется для прерывания доступа за инструкцией в область EXECUTE NEVER (XN), если блок MPU не используется.

Исключение **BusFault** возникает при ошибке памяти при выполнении выборки инструкций или обращения за данными. Это может быть при возникновении ошибки на шинах доступа к памяти, например, обращение в несуществующую память.

Исключение **USAGE FAULT** возникает при сбоях при выполнении инструкции.

Например:

- выполнение неизвестной инструкции;
- обращение к некорректно выровненным данным;
- некорректное состояние при выполнении инструкции;
- ошибка при возвращении из обработчика.

Данное исключение может быть сконфигурировано и используется для обработки следующих ситуаций:

- не выровненный адрес при обращении за полусловами halfword и словами word;
- деление на ноль.

Исключение **SysTick** генерируется системным таймером, когда он обнуляется. Программное обеспечение также может генерировать исключение SysTick. В приложениях с использованием операционной системы процессор может использовать это исключение для подсчета системных циклов.

Полный перечень исключений и более подробная информация приведена в [2].

В таблице 7.1 приведён список всех прерывание и исключений в микроконтроллере K1986BE92QI с указанием их номеров (масок) и приоритетов.

Таблица 7.1.

Исключения и прерывания микроконтроллера K1986BE92QI

Номер IRQn (маска)	Название	Приоритет	
Исключения			
-	RESET	-3 Наивысший	
-14	NonMaskableInt_IRQn	-2	
-13	HardFault_IRQn	-1	
-12	MemoryManagement_IRQn	Настраи- ваемый	
-11	BusFault_IRQn		
-10	UsageFault_IRQn		
-5	SVCall_IRQn		
-2	PendSV_IRQn		
-1	SysTick_IRQn		
Прерывания			
0	CAN1_IRQn	Настраи- ваемый	
1	CAN2_IRQn		
2	USB_IRQn		
5	DMA_IRQn		
6	UART1_IRQn		
7	UART2_IRQn		
8	SSP1_IRQn		
10	I2C_IRQn		
11	POWER_IRQn		
12	WWDG_IRQn		
14	Timer1_IRQn		
15	Timer2_IRQn		
16	Timer2_IRQn		
17	DC_IRQn		
19	COMPARATOR_IRQn		
20	SSP2_IRQn		
27	BACKUP_IRQn		
28	EXT_INT1_IRQn		
29	EXT_INT2_IRQn		
30	EXT_INT3_IRQn		
31	EXT_INT4_IRQn		

Таблица 7.2.

Некоторые типы прерывания микроконтроллера K1986BE92QI

Тип прерывания	Обработчик
Прерывание от таймера 1	Timer1_IRQHandler
Прерывание от таймера 2	Timer2_IRQHandler
Прерывание от таймера 3	Timer3_IRQHandler
Прерывание от АЦП	ADC_IRQHandler
Прерывание от компаратора	COMPARATOR_IRQHandler
Внешнее прерывание 1	EXT_INT1_IRQHandler
Внешнее прерывание 2	EXT_INT2_IRQHandler
Внешнее прерывание 3	EXT_INT3_IRQHandler
Внешнее прерывание 4	EXT_INT4_IRQHandler

В таблице 7.3 представлен список основных функций для работы с прерываниями в микроконтроллере K1986BE92QI.

Таблица 7.3.

Функции для настройки исключений и прерываний

Функция	Описание
void disable_irq(void)	Запретить прерывания
void enable_irq(void)	Разрешить прерывания
void NVIC_SetPriorityGrouping (uint32_t priority_grouping)	Установить группировку приоритетов
void NVIC_EnableIRQ (IRQn_t IRQn)	Разрешить прерывание под номером IRQn
void NVIC_DisableIRQ (IRQn_t IRQn)	Запретить IRQn
NVIC_ClearPendingIRQ(Timer1_IRQHandler);	Сбросить состояние ожидания обслуживания для IRQn

Прерывания периферийных модулей

Прерывания таймера

Таймер в микроконтроллере способен формировать следующие прерываний:

1. Прерывание по переднему фронту сигнала события захвата (CAP1) для регистра CCR1 для необходимого канала.
2. Прерывание по переднему фронту сигнала события захвата (CAP) для регистра CCR для необходимого канала.
3. Прерывание по переднему фронту сигнала сравнения REF для указанного канала.
4. Прерывание по сигналу на контакте BRK.
5. Прерывание по переднему фронту сигнала на контакте ETR.

6. Прерывание по заднему фронту сигнала на контакте ETR.
7. Прерывание по событию равенства регистров CNT и ARR.
8. Прерывание по событию равенства регистра CNT нулю.

Все перечисленные прерывания вызывают один обработчик – voidTimerX_IRQHandler() (где X – номер таймера). Включение необходимых прерываний осуществляется в регистре MDR_TIMER->IE, описание битов которого представлено в таблице 7.4.

Для того чтобы определить какое прерывание произошло, используется регистр MDR_TIMER->STATUS описание битов которого представлено в таблице 7.5.

Таблица 7.4.

Описание битов регистра MDR_TIMER->IE

№ бита	Название	Назначение
16...13	CCR CAP1 EVENT IE [3:0]	Флаг разрешения прерывания по событию переднего фронта на выходе CAP1 каналов таймера: 0 – нет прерывания; 1 – прерывание разрешено. Бит 0 – первый канал. Бит 3 – четвертый канал
12...9	CCR REF EVENT IE[3:0]	Флаг разрешения прерывания по событию переднего фронта на выходе REF каналов таймера: 0 – нет прерывания; 1 – прерывание разрешено. Бит 0 – первый канал. Бит 3 – четвертый канал
8...5	CCR CAP EVENT	Флаг разрешения прерывания по событию переднего фронта на выходе CAP каналов таймера: 0 – нет прерывания; 1 – прерывание разрешено. Бит 0 – первый канал. Бит 3 – четвертый канал
4	BRK EVENT IE	Флаг разрешения по состоянию входа BRK, синхронизированному по PCLK: 0 – нет прерывания; 1 – прерывание разрешено
3	ETR FE EVENT	Флаг разрешения прерывания по заднему фронту на входе ETR: 0 – нет прерывания;

	IE	1 – прерывание разрешено
2	ETR RE EVENT IE	Флаг разрешения прерывания по переднему фронту на входе ETR: 0 – нет прерывания; 1 – прерывание разрешено
1	CNT ARR EVENT IE	Флаг разрешения прерывания по событию совпадения CNT и ARR: 0 – нет прерывания; 1 – прерывание разрешено
0	CNT ZERO EVENT IE	Флаг разрешения прерывания по событию совпадения CNT и нуля: 0 – нет прерывания; 1 – прерывание разрешено

Таблица 7.5.

Описание битов регистра MDR_TIMER->STATUS

№ бита	Название	Назначение
16...13	CCR CAP1 EVENT[3:0]	Событие переднего фронта на входе CAP1 каналов таймера: 0 – нет события; 1 – есть событие. Сбрасывается записью 0, если запись одновременно с новым событием. Приоритет у нового события. Бит 0 – первый канал. Бит 3 – четвертый канал
12...9	CCR REF EVENT[3:0]	Событие переднего фронта на выходе REF каналов таймера 0 – нет события; 1 – есть событие. Сбрасывается записью 0, если запись одновременно с новым событием. Приоритет у нового события. Бит 0 – первый канал. Бит 3 – четвертый канал
8...5	CCR CAP EVENT[3:0]	Событие переднего фронта на входе CAP каналов таймера: 0 – нет события; 1 – есть событие. Сбрасывается записью 0, если запись

		одновременно с новым событием. Приоритет у нового события. Бит 0 – первый канал. Бит 3 – четвертый канал
4	BRK EVENT	Состояние входа BRK, синхронизированное по PCLK: 0 – BRK = 0; 1 – BRK = 1. Сбрасывается записью 0, при условии наличия 0 на входе BRK
3	ETR FE EVENT	Событие заднего фронта на входе ETR: 0 – нет события; 1 – есть событие. Сбрасывается записью 0, если запись одновременно с новым событием. Приоритет у нового события
2	ETR RE EVENT	Событие переднего фронта на входе ETR: 0 – нет события; 1 – есть событие. Сбрасывается записью 0, если запись одновременно с новым событием. Приоритет у нового события
1	CNT ARR EVENT	Событие совпадения CNT с ARR: 0 – нет события; 1 – есть событие. Сбрасывается записью 0, если запись одновременно с новым событием совпадения. Приоритет у нового события. Если с момента совпадения до момента программного сброса CNT и ARR не изменили состояния, то флаг повторно не взводится
0	CNT ZERO EVENT	Событие совпадения CNT с нулем: 0 – нет события; 1 – есть событие. Сбрасывается записью 0, если запись одновременно с новым событием совпадения. Приоритет у нового события. Если с момента совпадения до момента программного сброса CNT не изменил состояния, то флаг повторно не взводится

Прерывания АЦП

Модуль АЦП в микроконтроллере способен формировать следующие прерывания:

1. Прерывание по окончанию преобразования
2. Прерывание по событию выхода сигнала за пределы верхнего или нижнего значения указанных в регистрах MDR_ADC->ADCx_L_LEVEL и MDR_ADC->ADCx_H_LEVEL

Все прерывания АЦП вызывают один обработчик – voidADC_IRQHandler(). Включение необходимого прерывания, а также индикация произошедшего прерывания осуществляется в регистре MDR_TIMER->STATUS описание битов которого представлено в таблице 7.6.

Таблица 7.6.

Описание битов регистра MDR_ADC->ADCx STATUS

№ бита	Название	Назначение
4	ECOIF_IE	Флаг разрешения генерирования прерывания по событию Flg_REG_ECOIF: 0 – прерывания не генерируется; 1 – прерывание генерируется
3	AWOIF_I E	Флаг разрешения генерирования прерывания по событию Flg_REG_AWOIFEN: 0 – прерывания не генерируется; 1 – прерывание генерируется
2	FlgREG OCIF	Флаг выставляется, когда закончено преобразования и данные еще не считаны. Очищается считыванием результата из регистра ADCx_RESULT: 1 – есть готовый результат преобразования; 0 – нет результата
1	FlgREGA WOIFEN	Флаг выставляется, когда результат преобразования выше верхней или ниже нижней границы автоматического контролирования уровней. Сбрасывается только при записи нуля в данный бит регистр флагов: 0 – результат в допустимой зоне; 1 – вне допустимой зоны
0	FlgREG OVERWR ITE	Данные в регистре результата были перезаписаны, данный флаг сбрасывается только при записи нуля в данный бит регистр флагов:

		0 – не было события перезаписи несчитанного результата; 1 – был результат преобразования, который не был считан
--	--	--

Последовательность настройки прерывания в микроконтроллере K1986BE92Q1

1. Глобально разрешить прерывания, вызвав функцию `__enable_irq()`
2. Настроить прерывания в периферийном модуле.
3. Разрешить прерывание вызвав функцию `NVIC_EnableIRQ (IRQn_tIRQn)`. При необходимости установить приоритет.

В листинге 7.1 приведён пример настройки прерывания от таймера при равенстве значений счётного регистра и регистра основания счёта.

Листинг 7.1

```
void InitTimerIRQ(void);
int main()
{
    __enable_irq();// глобальное разрешение прерываний
    InitTimerIRQ();
    while(1)
    {
    }
    return 1;
}
void InitTimerIRQ(void)
{
    /*
    Настройка таймер
    */
    MDR_TIMER1->IE|=(1<<1);// разрешить прерывания при
    CNT==ARR
    NVIC_EnableIRQ(Timer1_IRQn);// разрешить прерывание от
    таймера 1
    MDR_TIMER1->STATUS&=~2;// очистить флаг CNT==ARR
    MDR_TIMER1->CNTRL|=1; //запустаймера 1
}
// обработчик прерывания
void Timer1_IRQHandler()
{

```

```

if((MDR_TIMER1->STATUS&(1 << 1))!=0)/*проверка, того что
прерывание произошло при CNT==ARR*/
{
    /*Пользовательский код*/
    MDR_TIMER1->STATUS &= ~(1 << 1);// очиститьфлаг
    CNT==ARR
}

NVIC_ClearPendingIRQ(Timer1_IRQn);/* Сбросить состояние
ожидания обслуживания для прерывания от таймера1*/
}

```

Задание для самостоятельной работы

Настроить прерывание в соответствии с таблицей задания

№	Тип	Событие для прерывания	Действия в обработчике	Примечание
1	Таймер 1	CNT==0	переключение светодиода на PC0	счёт вверх/вниз
2	АЦП 2	окончание преобразования	прочитать результат преобразования в глобальныймассив	каналы 2,4,6
3	Таймер 2	Событие переднего фронта на выходе REF канала 1, при OCCM[2:0]=6	переключение светодиода на PC1	счёт вверх
4	АЦП 1	Выход сигнала за пределы верхней или нижней границы	Включить светодиод на PC0 при выходе за нижний предел. Включить светодиод на PC1при выходе за верхний предел.	H=4000 L=100 канал 2
5	Таймер 2	CNT==ARR	переключение светодиода на PC1	счёт вниз
6	АЦП 1	окончание преобразования	прочитать результат преобразования в глобальныймассив	каналы 3,4,5
7	Таймер	Событие	переключение	счёт вверх

	1	переднего фронта на выходе REF канала 1, при $OCCM[2:0]=7$	светодиода на PC0	
8	АЦП 2	Выход сигнала за пределы верхней или нижней границы	Включить светодиод на PC0 при выходе за нижний предел. Включить светодиод на PC1 при выходе за верхний предел.	H=3750 L=250 канал 5
9	Таймер 2	Событие переднего фронта на выходе REF канала 1, при $OCCM[2:0]=6$	переключение светодиода на PC1	счёт вверх/вниз
10	АЦП 1	окончание преобразования	прочитать результат преобразования в глобальный массив	каналы 2,5,6
11	Таймер 2	$CNT==ARR$	переключение светодиода на PC0	счёт вверх/вниз
12	АЦП 1	Выход сигнала за пределы верхней или нижней границы	Включить светодиод на PC0 при выходе за нижний предел. Включить светодиод на PC1 при выходе за верхний предел.	H=2500 L=500 канал 3
13	Таймер 1	Событие переднего фронта на выходе REF канала 1, при $OCCM[2:0]=6$	переключение светодиода на PC0	счёт вниз
14	Таймер 2	$CNT==0$	переключение светодиода на PC1	счёт вниз
15	АЦП 2	окончание преобразования	прочитать результат преобразования в глобальный массив	каналы 2,3,6
16	АЦП 2	Выход сигнала за пределы	Включить светодиод на PC0 при выходе	H=1350 L=500

		верхней или нижней границы	за нижний предел. Включить светодиод на PC1 при выходе за верхний предел.	канал 5
17	Таймер 3	Событие переднего фронта на выходе REF канала 1, при OCCM[2:0]=6	переключение светодиода на PC0	счёт вверх/вниз
18	АЦП 1	окончание преобразования	прочитать результат преобразования в глобальный массив	
19	АЦП 2	Выход сигнала за пределы верхней или нижней границы	Включить светодиод на PC0 при выходе за нижний предел. Включить светодиод на PC1 при выходе за верхний предел.	H=3900 L=400 канал 4
20	Таймер 3	Событие переднего фронта на выходе REF канала 1, при OCCM[2:0]=7	переключение светодиода на PC1	счёт вниз

Для заданий с АЦП использовать режим последовательного преобразования. Тактирования АЦП осуществлять от сигнала HCLK с делителем Cfg REG DIVCLK =0x0B.

Для заданий с таймером значения регистров основания счета ARR и регистра делителя PSG равны 0xFFFF.

Для заданий, в которых в прерывании требуется прочитать значение АЦП при окончании преобразования, результат преобразования необходимо поместить в массив intresult [3]. Например, если необходимо прочесть значения с каналов 2, 3, 4, то значение с канала 2 должно быть записано в result[0], значение с канала 3 в result[1] и значение с канала 4 в result[2].

Контрольные вопросы

1. Что такое исключение?
2. Что такое прерывание?
3. Что такое приоритет прерывания?
4. Что такое маска прерывания?
5. Что такое маскируемое прерывание?
6. Что такое вектор прерывания?
7. Перечислите последовательность настройки прерываний в микроконтроллере K1986BE92QI

ЦИФРО-АНАЛОГОВЫЙ ПРЕОБРАЗОВАТЕЛЬ

Краткие теоретические сведения

Цифро-аналоговый преобразователь (ЦАП)- это устройство, которое осуществляет преобразование цифрового кода в соответствующий уровень напряжения.

В микроконтроллере реализовано 2 АЦП, разрядность которых составляет 12 бит. В качестве источника опорного напряжения может использоваться внутренний и внешний источник опорного напряжения. При использовании внутреннего источника опорного напряжения ЦАП формирует выходной сигнал в диапазоне от 0 до напряжения питания AU_{CC} (2.4-3.6В). В режиме работы с внешней опорой ЦАП формирует выходное напряжение в диапазоне от 0 до значения напряжения на контакте $DACx_REF$. Дополнительные параметры ЦАП микроконтроллера приведены в таблице 8.1.

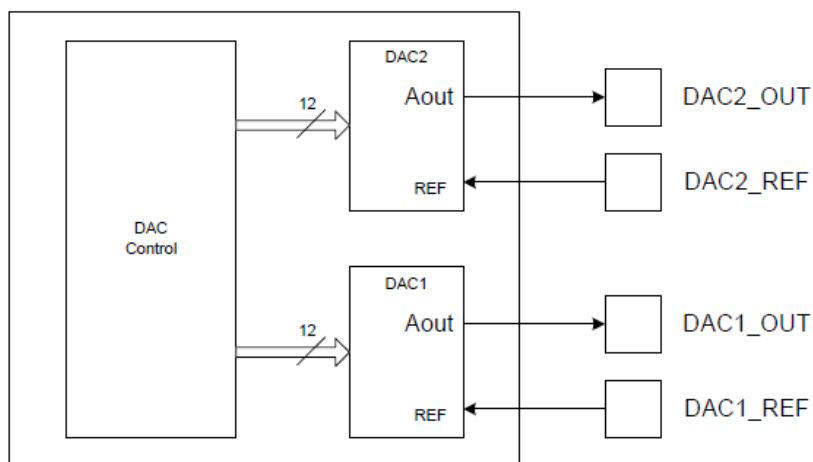


Рисунок 8.1. Структурная схема ЦАП

Таблица 8.1.

Параметры ЦАП в микроконтроллере

Максимальное внешнее опорное напряжение подключенное между DACx_REF и GND	2.4В
Резистивная нагрузка	не менее 10кОм
Емкостная Нагрузка	не более 100пФ

Для подключения ЦАП к тактированию необходимо возвести бит 18 в регистре MDR_RST_CLK->PER_CLOCK.

Выходной аналоговый сигнал подаётся на контакты порта E:PE0–DAC1_OUTиPE9 –DAC2_OUT.Все контакты, который использует ЦАП должны быть настроены в аналоговом режиме. Для подключения внешнего источника опорного напряжения использую контакты порта E:PE1–DAC1_REF иPE10 –DAC2_REF.

Регистры ЦАП представлены в таблицах 6.2 – 6.4.

Таблица 8.2.

Описание битов регистра MDR_DAC->CFG

№ бита	Название	Назначение
4	Cfg_SYNC_A	Синхронизация DAC1 и DAC2: 0 – асинхронные; 1 – синхронные
3	Cfg_ON_DAC1	Включение DAC2: 1 – включен; 0 – выключен
2	Cfg_ON_DAC0	Включение DAC1: 1 – включен; 0 – выключен
1	Cfg_M_REF1	Выбор источника опорного напряжения DAC2: 0 – в качестве опорного напряжения используется напряжение питания с вывода AUCC; 1 – в качестве опорного напряжения используется напряжение на входе опорного напряжения DAC2_REF
0	Cfg_M_REF0	Выбор источника опорного напряжения DAC1: 0 – в качестве опорного напряжения используется напряжение питания с

		вывода AUCC; 1 – в качестве опорного напряжения используется напряжение на входе опорного напряжения DAC1 REF
--	--	--

Таблица 8.3.

Описание битов регистра MDR_DAC->DAC1_DATA

№ бита	Название	Назначение
27...16	DAC1 DATA[11:0]	Данные DAC1 при Cfg_SYNC_A=1. При чтении всегда равны нулю. Читать из DAC2_DATA
11...0	DAC0 DATA[11:0]	Данные DAC1

Таблица 8.4.

Описание битов регистра MDR_DAC->DAC2_DATA

№ бита	Название	Назначение
27...16	DAC0 DATA[11:0]	Данные DAC0 при Cfg_SYNC_A=1. При чтении всегда равны нулю. Читать из DAC1_DATA
11...0	DAC1 DATA[11:0]	Данные DAC2

Примечание – Если бит конфигурации Cfg_SYNC_A установлен, то данные для DAC1 и DAC2 задаются записью в один из регистров DACx_DATA.

В листинге 8.1 представлена функция инициализации ЦАП2

Листинг 8.1

```
voidInitDAC(void)
{
    MDR_RST_CLK->PER_CLOCK|=(1<<18);//
включитьтактирование ЦАП
    MDR_RST_CLK->PER_CLOCK|=(1<<25);//
включитьтактирование порта E
    MDR_DAC->CFG|=(1<<3);// включитьЦАП 2
}
```

Задание для самостоятельной работы

1 Создать проект. Реализовать функции инициализации ЦАП2 как показано в листинге 8.1и инициализации прерывания от таймер 1 при $CNT=ARR$. Таймер должен работать в режиме счёта вверх, регистрPSG=8, а регистр ARR=1000;

2 Добавить в проект файл поддержки математических функций, для этого необходимо нажать кнопку ManagerRun-TimeEnvironment  и в появившемся окне установить галочку в поле DSP, как показано на рисунке 8.2. Далее в опция проекта во вкладке C/C++ в поле Defineнеобходимо записать ARM_MATH_CM3, как показано на рисунке 8.3 и нажать ОК. Затем файле с функцией mainнеобходимо произвести подключение заголовочного файлаarm_math.h, прописав #include "arm_math.h".

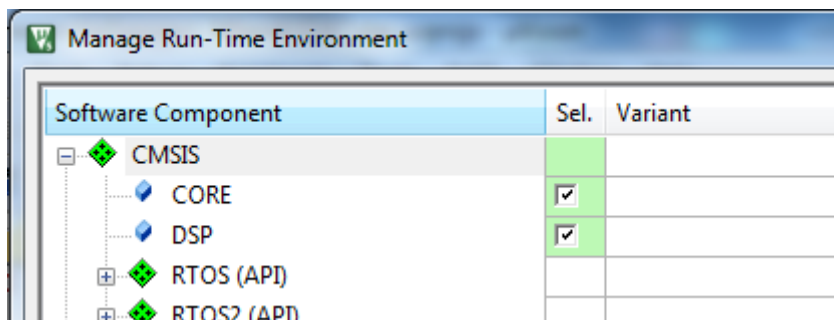


Рисунок 8.2.

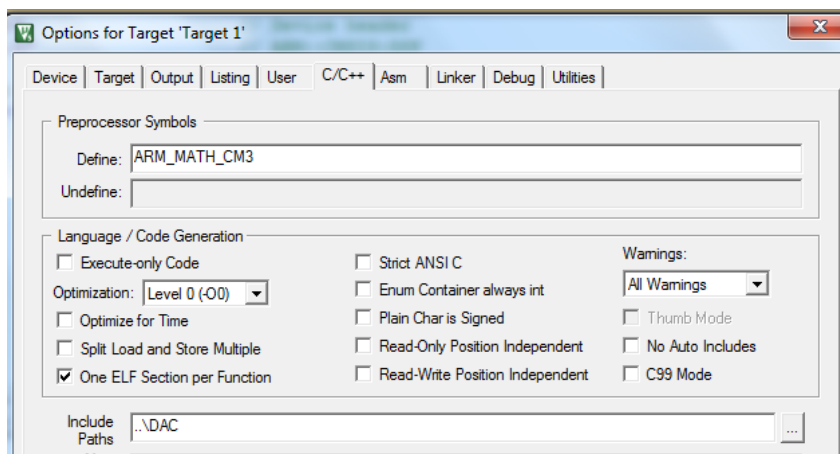


Рисунок 8.3.

В файле main.c реализовать код представленный в листинге 8.2.

Листинг 8.2

```
int data[360];
int i=0;
int main()
{
    __enable_irq();
    for(i=0;i<360;i++)
    {
        data[i]=F(i*3.14/180.0);
    }
    i=0;
    InitDAC();
    InitTimerIRQ();
    while(1) {
        return 1;
    }
}
```

```

}
void Timer1_IRQHandler()
{
    if((MDR_TIMER1->STATUS &(1 << 1))!=0)
    {
        MDR_DAC->DAC2_DATA&=~(0xFF);
        MDR_DAC->DAC2_DATA|=data[i];
        i++;
        if(i==360){i=0; }
    }
    MDR_TIMER1->STATUS &= ~(1 << 1);
    NVIC_ClearPendingIRQ(Timer1_IRQn);
}

```

4 Реализовать функцию floatF (floatx), расчет значений сигнала по формуле:

$$F(x) = a_0 + a_1 \sin(x) + a_2 \sin(2x) + a_3 \sin(3x) + b_1 \cos(x) + b_2 \cos(2x) + b_3 \cos(3x)$$

$$a_0 = 100N + 25 \qquad b_1 = (2N + 25) + N$$

$$a_1 = 5N + N \qquad b_2 = (-1)^N (N - 15)$$

$$a_2 = 2(10N + 1) + 25 \qquad b_3 = 25$$

$$a_3 = 5N + 80$$

где N – номер варианта.

5 Подключить осциллограф к разъёму 9 (см рисунок 1). Сравните форму сигнала с осциллографа с формой сигнала рассчитанной аналитически с использованием пакета Microsoft Excel, математического макета Mathcad или MatLab.

Контрольные вопросы

1. Что такое ЦАП?
2. Какая разрядность ЦАП в микроконтроллере K1986BE92QI?
3. Какие виды источников опорного напряжения могут использоваться для ЦАП в микроконтроллере K1986BE92QI?

ЛАБОРАТОРНАЯ РАБОТА 6
**МИКРОПРОЦЕССОРНАЯ СИСТЕМА ИМПУЛЬСНО-
ФАЗОВОГО УПРАВЛЕНИЯ ТИРИСТОРНЫМ
ПРЕОБРАЗОВАТЕЛЕМ**

Цель работы: изучение работы микропроцессорной СИФУ тиристорного преобразователя.

1. Теоретические сведения

Синхронизация в трехканальной СИФУ, функциональная схема которой приведена на рисунке 9.1 осуществляется по трем фазам. Каждый канал содержит фильтр синхронизации Φ , компаратор K , микропроцессорный блок МП, реализующий задержку времени импульсов управления относительно моментов естественной коммутации и распределение импульсов управления, усилители импульсов.

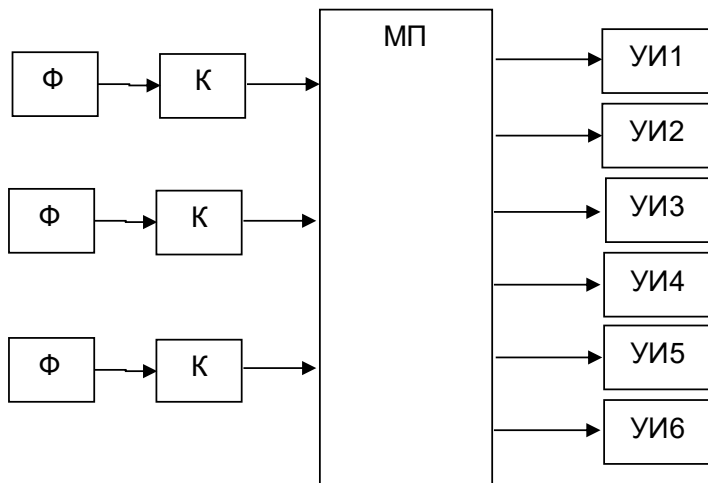


Рисунок 9.1 – Функциональная схема трехканальной СИФУ

Синхронизация осуществляется по переходу напряжения синхронизации через ноль. Для отсчета угла управления используется программируемый таймер, который генерирует прерывания с

интервалом, кратным циклам процессора. Блок-схема алгоритма синхронизации приведена на рисунке 9.2. По прерыванию (переднему или заднему фронту импульса синхронизации) определяется указатель номера тиристора и в таймер записывается значение угла управления α и запускается соответствующий таймер.

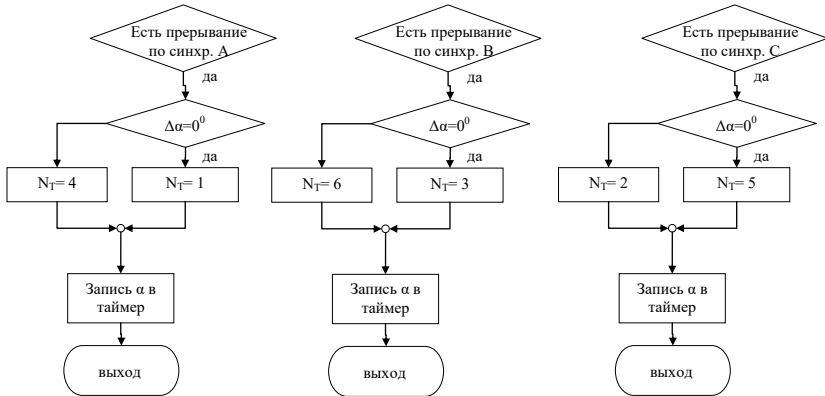


Рисунок 9.2 Блок-схема алгоритма синхронизации

На рисунке 9.3 приведена блок схема алгоритма СИФУ. На каждом интервале периода дискретизации СИФУ определяется приращение угла управления. Если угол управления не меняется, то значения таймеров также не меняются, продолжается счет до обнуления, после чего формируются импульсы управления соответствующими указателю тиристорами. Если приращение угла управления не равно нулю, то для каждого канала вычисляется сумма текущего значения таймера и приращения угла управления. Если указанная сумма больше нуля, то ее значением обновляется значение таймера. Если указанная сумма меньше нуля, то таймер останавливается и генерируется принудительное прерывание от таймера и формируются импульсы по соответствующему каналу тиристора.

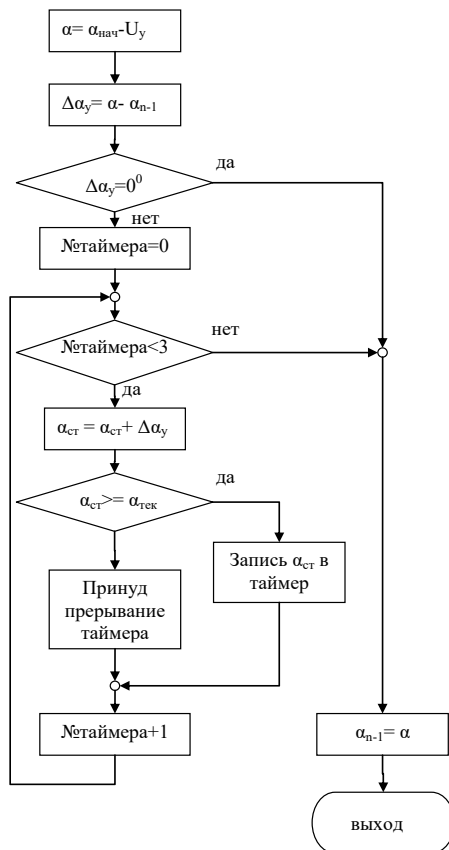


Рисунок 9.3 – Блок-схема алгоритма СИФУ.

2. Самостоятельная подготовка

2.1 Проработать раздел 1 к лабораторной работе, рекомендуемую литературу.

2.2 Составить программу, реализующую алгоритм трехканальной СИФУ.

2.3 Подготовить бланк отчета к лабораторной работе. Отчет должен содержать фамилию и группу студента, название, цель, порядок выполнения работы, а также документацию на микроконтроллер, задание, схему и программу.

3. Порядок выполнения работы

3.1 Создать новый проект в среде Keil μ Vision и составить программу, реализующую алгоритм трехканальной СИФУ.

3.2 Собрать схему СИФУ, приведенную на рисунке 9.4 и подключить управляющие выходы контроллера к драйверам тириستоров.

3.3 Запрограммировать микроконтроллер.

3.4 Запустить моделирование, проверить правильность формируемых импульсов.

3.5 Изменяя напряжение управления добиться изменения фазы импульсов управления вентилями.

3.6 Зарисовать осциллограммы работы системы при напряжении управления равном $U_y=0, 10, 25, 50, 75\%$ от максимального значения.

3.7 Определить диапазон и дискретность изменения угла управления.

3.8 Продемонстрировать результаты работы преподавателю.

4. Контрольные вопросы

1. Особенности цифровых СИФУ.
2. Классификация СИФУ.
3. Принцип построения одноканальных СИФУ.
4. Достоинства и недостатки одноканальных СИФУ.
5. Принцип построения трехканальных СИФУ.
6. Достоинства и недостатки трехканальных СИФУ.
7. Как определяется дискретность угла управления.
8. Влияние разрядности процессора на свойства СИФУ.

ЛАБОРАТОРНАЯ РАБОТА 7
**МИКРОПРОЦЕССОРНАЯ СИСТЕМА УПРАВЛЕНИЯ
МОСТОВЫМ ТРЁХФАЗНЫМ ОДНОУРОВНЕВЫМ
ИНВЕРТОРОМ С ИСПОЛЬЗОВАНИЕМ МЕТОДА СИНУСНОЙ
ШИМ.**

Цель работы: изучение работы микропроцессорной системы управления трёхфазным одноуровневым инвертором на базе транзисторов.

1. Теоретические сведения

Структурная схема микропроцессорной системы управления представлена на рисунке 10.1, где МК- микроконтроллер, Д – драйвер силовых транзистор инвертора, VT1-6 - силовые транзисторы инвертора, VD1-6 - защитные обратные диоды. Для устранения возможных помех создаваемых в силовой части инвертора, может использоваться блок гальванической развязки (ГР), выполненный на высокоскоростных оптронах. Транзисторы включенные последовательно (например транзисторы VT1 и VT2) образуют группу, называемую **плечом** (или **стойкой**). В мостовых трехфазных одноуровневых инвертора используется шесть транзисторов, которые формируют три плеча. Когда открыт верхний транзистор плеча, нижний должен быть закрыт, тоже самое справедливо и для нижнего: когда нижний транзистор плеча открыт, верхний должен быть закрыт. Ситуация когда оба транзистора плеча открыты, является аварийной и может привести к протеканию больших токов (сквозной ток через плечо), которые могут привести к выходу из строя транзисторов плеча.

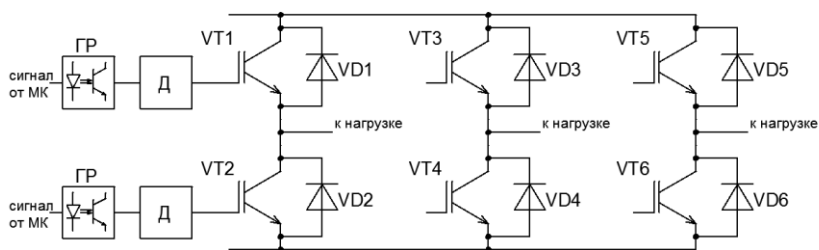


Рис.10.1 Структурная схема микропроцессорной системы управления

При управлении транзисторами, образующими плечо, следует учитывать их динамические характеристики, а именно время включения

t_{on} и t_{off} выключения транзисторов. Так как эти параметры имеют ненулевую величину, то между подачей сигнала на открытия верхнего (нижнего) транзистора и сигнала закрытия нижнего (верхнего) необходимо вставлять временную паузу, называемую **«мёртвым» временем** (dead-time) – т.е. интервал времени в течении которого на обоих транзисторах плеча отсутствует сигнал на открытие. **Отсутствие «мёртвого» времени в сигнала управления может также привести к протекаю сквозного тока.** Современные микроконтроллеры, предназначенные для управления трёхфазными инверторами, позволяют производить вставку «мёртвого» времени в ШИМ сигнал (см. Лабораторная работа 5. ГЕНЕРАЦИЯ ШИМ СИГНАЛА).

Синусная широтно-импульсная модуляция это метод расчёта скважности включенного состояния верхних транзисторов в плечах инвертора (VT1, VT3, VT5) согласно выражения:

$$\begin{cases} y_A = 0.5 * \sin(\omega \cdot t) + 0.5 \\ y_B = 0.5 * \sin(\omega \cdot t + \frac{2\pi}{3}) + 0.5 \\ y_C = 0.5 * \sin(\omega \cdot t - \frac{2\pi}{3}) + 0.5 \end{cases}$$

При использовании микроконтроллеров с таймерами, имеющих три и более блоков сравнения, полученные значения скважностей (y_A , y_B , y_C) необходимо умножить величину до которой способен досчитать таймер (для микроконтроллеров K1986BE92QI эта величина храниться в таймером регистре ARR), и записать полученные значения в соответствующие регистры сравнения.

Для того чтобы значения скважностей менялись, необходимо обеспечить дискретное изменение аргумента синуса, т.е. значения ωt . Для этого необходимо создать событие, которое будет происходить с определённой фиксированной периодичностью (например прерывание от таймера, но это таймер не должен отвечать за генерацию ШИМ для инвертора). И в обработке этого события производить увеличение ωt от 0 до 2π с необходимой дискретностью

3. Порядок выполнения работы

3.1 Создать новый проект в среде Keil μ Vision и составить программу, реализующую управление трёхфазным мостовым инвертором с использованием синусной .

3.2 Собрать схему, приведенную на рисунке 10.1 и подключить управляющие выводы контроллера к драйверам транзисторов.

3.3 Запрограммировать микроконтроллер.

3.4 Запустить моделирование, проверить правильность формируемых импульсов.

3.5 Зарисовать осциллограммы работы системы при амплитуде выходного напряжения равном $U=0, 10, 25, 50, 75\%$ от максимального значения и частоте 25 Гц.

3.6 Продемонстрировать результаты работы преподавателю.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Благодаров А.В. Программирование микроконтроллеров на основе отечественных микросхем семейства 1986BE9х разработки и производства компании «Миландр»// А.В. Благодаров, Л.Л. Владимиров/ Москва 2016 . – 242с.
2. Спецификация на микросхему 32-разрядного однокристалльного микро-ЭВМ с памятью Flash-типа 1986BE9ху, K1986BE9ху, K1986BE9хуK K1986BE92QI, K1986BE92QC, 1986BE91H4, K1986BE91H4, 1986BE94H4, K1986BE94H4 [Электронный ресурс] URL: [http:// ic.milandr.ru /upload/iblock/618/61867f2ba685fa99043947ceba748f02.pdf](http://ic.milandr.ru/upload/iblock/618/61867f2ba685fa99043947ceba748f02.pdf)
3. Джозеф Ю. Ядро Cortex-M3 компании ARM. Полное руководство/ Джозеф Ю; пер. с англ. А.В. Евстеева. – Москва: Додэка-XXI, 2012. – 552с.