



Государственное бюджетное образовательное учреждение высшего образования
Московской области

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Строганова С.М., Теодорович Н.Н.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ по микроконтроллерам семейства

1986ВЕ9Х компании Миландр

для студентов специальности 27.03.04 Управление в технических системах

Королев

2016

| | |
|---|-----------|
| Введение | 3 |
| I. Архитектура 32-разрядных микроконтроллеров семейства 1986VE9x компании Миландр..... | 6 |
| II. Отладочный комплект M/CX 1986VE91T(94T)..... | 14 |
| III. Меры безопасности при работе с лабораторным отладочным макетом | 22 |
| ЛАБОРАТОРНАЯ РАБОТА № 1. Знакомство с отладочной платой для микроконтроллера семейства 1986VE9X компании Миландр и средой программирования Keil μVision | 23 |
| ЛАБОРАТОРНАЯ РАБОТА № 2. Порты ввода-вывода. Управление светодиодом | 33 |
| ЛАБОРАТОРНАЯ РАБОТА № 3. «Бегущие огни»..... | 37 |
| ЛАБОРАТОРНАЯ РАБОТА № 4. Логические выражения | 41 |
| ЛАБОРАТОРНАЯ РАБОТА № 5. Таймеры общего назначения | 45 |
| ЛАБОРАТОРНАЯ РАБОТА № 6. Контроллер прерываний | 51 |
| ЛАБОРАТОРНАЯ РАБОТА № 7. Модель осветительных приборов автомобиля | 54 |
| ЛАБОРАТОРНАЯ РАБОТА № 8. «Светофор» | 56 |
| ЛАБОРАТОРНАЯ РАБОТА № 9. Создание действующей модели движения пассажирского лифта | 57 |
| ЛАБОРАТОРНАЯ РАБОТА № 10. Оцифровка и фильтрация аналогового сигнала (АЦП) | 59 |
| ЛАБОРАТОРНАЯ РАБОТА № 11. ШИМ..... | 68 |
| ЛАБОРАТОРНАЯ РАБОТА № 12. Аппаратная реализация широтно- импульсной модуляции..... | 73 |
| ЛАБОРАТОРНАЯ РАБОТА № 13. Генерация аналогового сигнала | 79 |
| ЛАБОРАТОРНАЯ РАБОТА № 14. Вывод графической информации на ЖК-дисплей..... | 83 |
| Литература..... | 84 |

Введение

Современная микропроцессорная техника является важнейшим средством при решении самых разнообразных задач в области сбора и обработки данных, систем автоматического управления и др.

Микроконтроллеры представляют собой ряд широко известных представителей микропроцессорной техники. Они объединяют на одном кристалле высокопроизводительный процессор, память, периферийные устройств и позволяют широкий спектр систем управления как объектами, так и процессами с минимальными затратами реализовать. [2,5]

В рамках обучения принципам работы и архитектуры микроконтроллеров студентам будет предложено познакомиться микроконтроллерами серии 1986VE9х, основанных на ядре архитектуре, производства одного из лидера разработки отечественной микроэлектронной элементной базы ЗАО «ПКК МИЛАНДР». [5,6]

Студентам направления «Управление в технических системах» будет предложено выполнять разработку программного обеспечения интегрированной среде программирования Keil uVision. Данная среда предоставляет пользователю набор средств для написания и отладки кода программ для микроконтроллеров на основе ядра ARM7, ARM9, Cortex M3 и других. В бесплатный дистрибутив входят следующие средства:

- интегрированная среда разработки;
- C/C++ компилятор;
- макроассемблер и линковщик;
- дебаггер uVision;
- дополнительные утилиты.

Единственным ограничением демоверсии является длина кода в 32 бита, но для получения первичных навыков работы этого вполне достаточно. Тем более, что опыт преподавания говорит в пользу решения нескольких подобных типовых задач, которые позволят отработать и закрепить навыки работы с микроконтроллерами, а также реализовать основной принцип

обучения от простого к сложному. К простым задачам этого этапа изучения дисциплины можно отнести, например, изменение какого-либо параметра физического сигнала-носителя, вычислительные преобразования и др.[8]

Лабораторный практикум разрабатывается на основе демонстрационно-отладочного комплекса АО «ПКК Миландр», в состав которого входят: отладочная плата, микроконтроллер 1986BE91T (или 1986BE94T), кабель RS-232/RS-232, кабель USB/USB, блок питания для отладочной платы, диск с документацией, схемотехническими файлами и исходными кодами программ.

Целью лабораторного практикума является:

Изучение технологии применения микроконтроллеров в системах управления техническими объектами и технологическими процессами, проектирования систем управления на базе микроконтроллеров, разработки их аппаратно-программного обеспечения и методов контроля.

Основными задачами лабораторного практикума являются:

- приобретение базовых знаний для понимания принципов создания и функционирования микроконтроллеров
- изучение принципов построения, функциональных возможностей и архитектурных решений микроконтроллеров.
- освоение методики составления и отладки программ для микроконтроллеров
- формирование навыков самостоятельного проектирования фрагментов программного обеспечения микроконтроллера

После завершения освоения данной дисциплины студент должен:

знать: принципы построения и архитектуру микроконтроллеров, структуру аппаратных и программных средств микроконтроллеров, основные задачи, решаемые с помощью микроконтроллеров;

уметь: проводить отладку, диагностику и проектирование микроконтроллеров, использовать стандартные терминологию, определения и обозначения;

владеть: навыками самостоятельного проектирования фрагментов программного обеспечения микроконтроллера

В **первом** части лабораторного практикума кратко представлено описание архитектуры, конструктивных особенностей и основных функциональных возможностях 32-разрядных микроконтроллеров серии 1986BE9x, построенных на базе процессорного RISC ядра ARM Cortex-M3. Коротко рассмотрена схема питания микроконтроллеров данной серии, показана схема организация внешней памяти.

Во **второй** разделе представлен отладочный комплект М/СХ 1986BE91Т(94Т) демонстрационно-отладочного комплекса АО «ПКК Миландр», на основе которого выполняются задания лабораторного практикума и кратко рассмотрены его возможности.

В **третьем** разделе говорится о мерах безопасности при работе с лабораторным отладочным макетом.

В **четвертом** разделе приводятся задания к лабораторным работам, примеры, требования к результатам, содержанию и оформлению отчетов.

I. Архитектура 32-разрядных микроконтроллеров семейства 1986BE9x компании Миландр

Основной характеристикой любой микропроцессорной системы является ее производительность, под которой в общем случае понимают количество выполняемых в единицу времени элементарных операций и время доступа к памяти и внешним устройствам. Критериями максимальной производительности микропроцессорной системы следует считать в первую очередь минимальное время доступа к памяти и максимально возможную тактовую частоту процессора. Простейшая микропроцессорная система представлена на рисунке 1.[2]

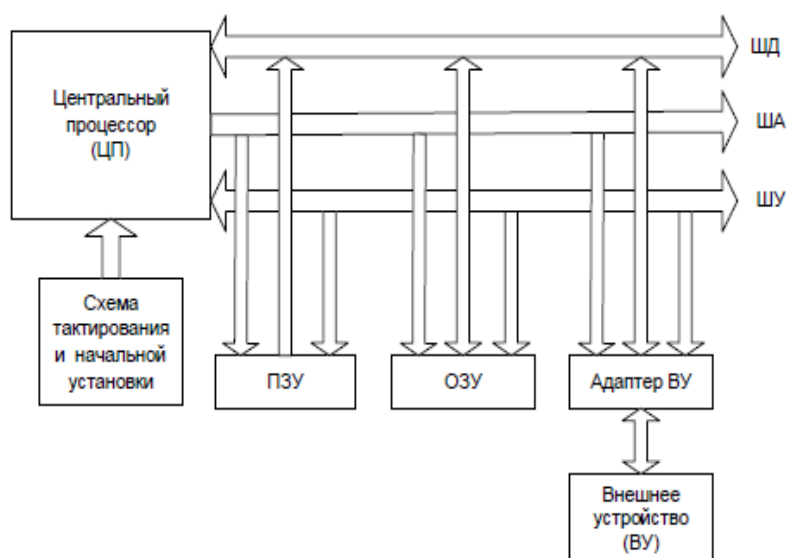


Рисунок 1. Схема элементарной микропроцессорной системы

На сегодняшний день в мире существуют тысячи типов микроконтроллеров, выпускаемые такими известными компаниями как Atmel, Analog Devices, Oki Semiconductor, Dallas Semiconductor, Philips Semiconductors, Intel, Infineon Technologies Hitachi Semiconductor, STMicroelectronics, Microchip Technology Inc., Texas Instruments, Mitsubishi Electronics, Motorola Semiconductor, National Semiconductor, Temic, Toshiba и многие другие.

Естественно, что для умения свободно ориентироваться среди такого обширного ряда продукции требуются грамотные специалисты, но здесь

появляются сложности, связанные в первую очередь с бурным развитием отрасли и выпуском новых поколений МК.

Поэтому в задачу вуза входит поиск оптимальных решений по подготовке специалистов, способных работать по избранной профессии долгие годы и постоянный тесный контакт с разработчиками материальной базы.

Одним из важнейших моментов в обеспечении фундаментальности образования в данной сфере является необходимость выделения среди быстро меняющейся предметной области тех основополагающих ключевых моментов, которые позволят создать базовый слой знаний, который будет устойчив к трансформациям реализуемых микроконтроллеров. [4]

К таким базовым знаниям можно отнести:

- общие структурные и функциональные схемы;
- принципы обмена данными во внутренних магистралях;
- основные принципы функционирования процессорного ядра;
- построение памяти программ;
- посторонние памяти данных;
- понятие о классах периферийных устройств, таких как порты, таймеры, средства повышения надежности функционирования;
- средства обмена данными и т.д.

И этими знаниями студенты должны овладевать уже на первом этапе изучения дисциплины, в то время как на втором этапе проводится знакомство уже с конкретными представителями микропроцессорной техники. [7,8]

Для организации учебной и преподавательской работы с информацией на современном уровне на кафедре внедряется ряд информационных сервисов. Выбор этих сервисов основывается на возможности учиться и работать с реальными средствами и инструментами, которые производятся отечественной промышленностью и с которыми сегодняшним студентам предстоит встретиться на практике. Микроконтроллеры представляют собой основу, на которой можно создавать современные экономичные и

высокопроизводительные системы многоцелевого назначения.

В одной микросхеме микроконтроллер включает в себя микропроцессор, память программ (обычно на основе ПЗУ), память данных (обычно на основе ОЗУ), устройство ввода/вывода, генератор тактовых сигналов, аппаратную поддержку интерфейсов I2C, SPI и многое другое.

Микроконтроллеры серии 1986BE9x построены на базе процессорного RISC ядра ARM Cortex-M3, которое обладает высокой производительностью. Они содержат 32 Кбайт ОЗУ и встроенную 128 Кбайт Flash-память программ. Тактовая частота микроконтроллеров порядка 80 МГц. Периферия микроконтроллера состоит из контроллера USB интерфейса, со встроенным аналоговым приемопередатчиком, имеющим скорости передачи 12 Мбит/с (Full Speed) и 1,5 Мбит/с (Low Speed), стандартные интерфейсы UART, SPI и I2C, контроллер внешней системной шины. Это обеспечивает работу с внешними микросхемами статического ОЗУ и ПЗУ, NAND Flash-памятью, а также другими внешними устройствами.

За счет матрицы системных шин в архитектуре системы памяти возможно свести к минимуму вероятные конфликты при работе системы, а также добиться повышения общей производительности. Ускорить обмен информацией между ОЗУ и периферией без участия процессорного ядра позволяет контроллер DMA.

Встроенный регулятор предназначен для формирования питания внутренней цифровой части. Он формирует напряжение 1,8 В и не требует дополнительных внешних элементов. Следовательно, для работы микроконтроллера достаточно одного внешнего напряжения питания в диапазоне от 2,2 до 3,6 В.

В микроконтроллерах также реализован батарейный домен, который работает от внешней батареи, и который предназначен для обеспечения функций часов реального времени. Он позволяет сохранить некоторый объем данных при отключении основного питания. По имеющиеся встроенным детекторам напряжения питания легко отследить как уровень внешнего

основного питания, так и уровень напряжения питания на батарее.

С помощью аппаратных схем сброса при уменьшении напряжения питания дают возможность исключить сбои в работе микросхемы при снижении напряжения питания ниже допустимых параметров.

В зависимости от корпуса, в котором выпускается микросхема, изменяются функциональные возможности микроконтроллеров, но при этом объем памяти программ и ОЗУ остается одинаковым.

Таблица 1 – Основные характеристики микроконтроллеров серии 1986BE9x

| | 1986BE91T 1986BE94T | K1986BE91H4 | 1986BE92Y K1986BE92QI K1986BE92QC | 1986BE93Y |
|----------------------|--|--------------|---|------------|
| Корпус | 132 вывода | бескорпусная | 64 вывода | 48 выводов |
| Ядро | ARM Cortex-M3 | | | |
| ПЗУ | 128 Кбайт Flash | | | |
| ОЗУ | 32 Кбайт | | | |
| Питание | 2,2...3,6 В | | | |
| Частота | 80 МГц | | | |
| USER IO | 96 | 96 | 43 | 30 |
| USB | Device и Host FS (до 12 Мбит/с) встроенный PHY | | | |
| UART | 2 | 2 | 2 | 2 |
| CAN | 2 | 2 | 2 | 2 |
| SPI | 2 | 2 | 2 | 1 |
| I2C | 1 | 1 | 1 | - |
| 2 x 12-разрядных АЦП | 16 каналов | 16 каналов | 8 каналов | 4 канала |
| ЦАП 12 разрядов | 2 | 2 | 1 | 1 |
| Компаратор | 3 входа | 3 входа | 2 входа | 2 входа |
| Внешняя шина | 32 разряда | 32 разряда | 8 разрядов | - |

Для успешной работы с микропроцессорами серии 1986BE9x необходимо хорошо представлять себе схему питания. В частности, следует знать, что они имеют несколько типов выводов питания (Рисунок 2).

UCC выводы:

Основное питание микросхемы, включает питание пользовательских выводов, встроенного регулятора напряжения, USB PHY и генераторов. Входное напряжение должно быть в пределах от 2,2 до 3,6 В. Если используется интерфейс USB, то входное напряжение должно быть в пределах от 3,0 до 3,6 В. Если используется АЦП или ЦАП, то входное напряжение должно быть в пределах от 2,4 до 3,6 В.

DUCC выводы:

Питание внутренней цифровой части, памяти ОЗУ и Flash-памяти. Это питание формируется внутренним регулятором напряжения из UCC. В нормальном режиме работы этот вывод должен остаться неподсоединенным. В некоторых корпусах данные выводы отсутствуют. Напряжение на выводе DUCC должно быть в пределах от 1,62 до 1,98 В.

BUCC вывод:

Питание батарейного домена используется при отсутствии основного питания UCC для питания батарейного домена и LSE генератора. Переключение с основного питания на батарейное происходит автоматически при снижении уровня UCC ниже 2,0 В. Переключение с батарейного питания на основное происходит автоматически спустя примерно 4 мс после превышения уровнем UCC порога в 2,0 В. Входное напряжение должно быть в пределах от 1,8 до 3,6 В. Если в системе не требуется батарейного питания, то вывод BUCC должен быть объединен с UCC.

BDUCC вывод:

Результирующие напряжения после выбора между BUCC и UCC при питании батарейного домена. В нормальном режиме этот вывод должен остаться не подсоединенным. В некоторых корпусах данные выводы отсутствуют.

AUCC выводы:

Питание аналоговых блоков АЦП, ЦАП и Компаратора выведено на отдельные выводы для уменьшения помех, создаваемых работой других блоков. На данные выводы должно подаваться напряжение из того же источника, что и UCC, но при этом на печатной плате должны быть применены меры по снижению помех. Для корректной работы АЦП входное напряжение должно быть в пределах от 2,4 до 3,6 В. Если входное напряжение будет в пределах от 2,2 до 2,4 В, то корректная работа АЦП не гарантируется.

AUCC1 выводы:

Питание аналоговых блоков и схем PLL выведено на отдельные выводы для уменьшения помех, создаваемых работой других блоков. На данные выводы должно подаваться напряжение из того же источника, что и UCC, но при этом на печатной плате должны быть применены меры по снижению помех.

GND выводы:

Основная «земля» питания.

AGND выводы:

Земля аналогового питания AUCC. Данные выводы должны соединяться с GND, но при этом на печатной плате должны быть применены меры по снижению помех.

AGND1 выводы:

Земля аналогового питания AUCC1. Данные в соединяться с GND, но при этом на печатной плате должны быть применены меры по снижению помех.[1]

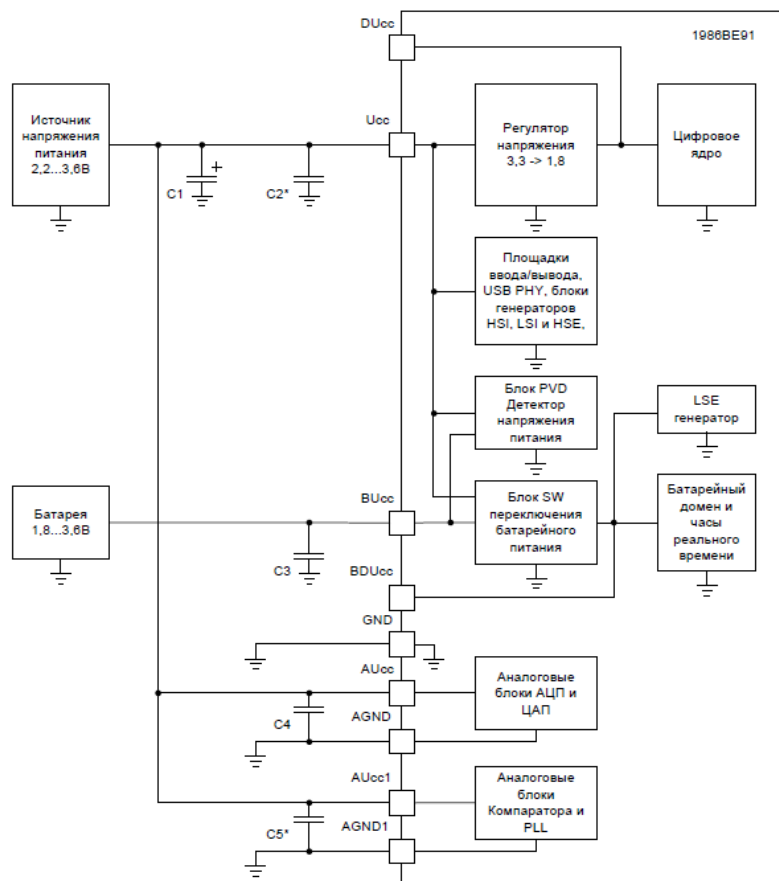


Рисунок 2. Структурная блок-схема подачи питания

Следует также обратить внимание на организацию структурной памяти.

Ядро процессора имеет три системных шины:

- Шину выборки инструкций (I Code);
- Шину выборки данных, которые располагаются в коде программы (D Code);
- Шину выборки данных, которые расположены в области ОЗУ (S Bus).

В микроконтроллере существует также контроллер прямого доступа в память (DMA), осуществляющий выборку через шину DMA Bus.

Все адресное пространство микроконтроллера едино и имеет максимальный объем 4 Гбайт и это адресное пространство отображаются различные модули памяти и периферии. [1]

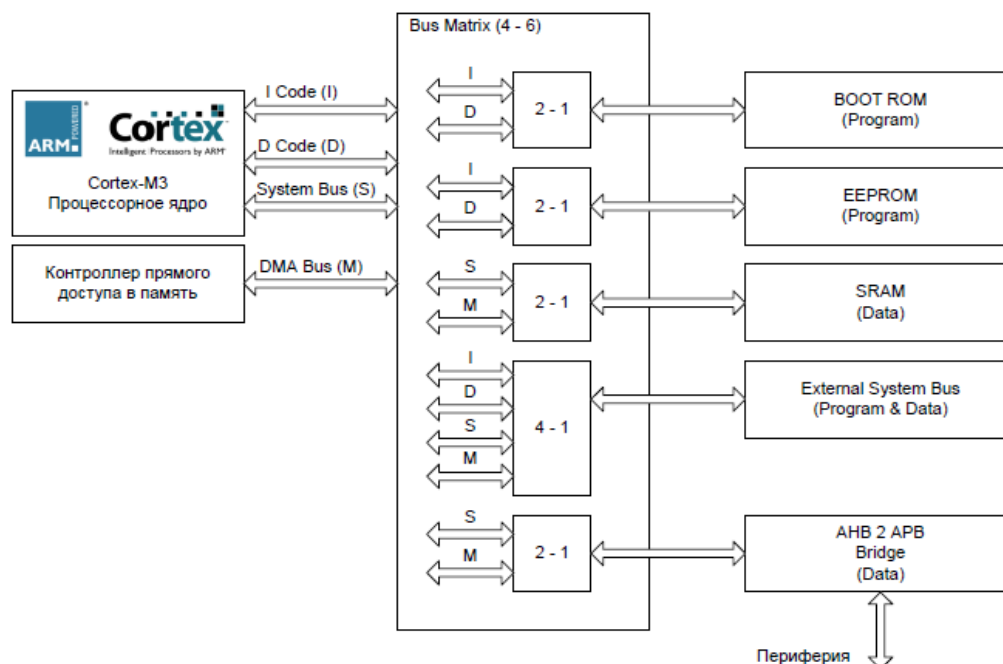


Рисунок 4. Структурная схема организации памяти

Таким образом, микропроцессор способен выполнять множество операций, что определяется программой, управляющей информацией, которая представляет собой набор команд (цифровых кодов). При этом его

быстродействие, гибкость, удобство использования определяется структурой и объемом системы команд процессора. Процессор может иметь десятки и даже сотни команд, при этом система команд может быть рассчитана на узкий круг решаемых задач или на максимально широкий круг задач в зависимости от того, с каким типом процессора мы имеем дело (специализированным или универсальным, например). Коды команд в каждом случае имеют различное количество разрядов, и каждая команда имеет свое время выполнения. Следовательно, время выполнения всей программы зависит как от количества команд в той или иной программе, так и от конкретных используемых команд. [3,5]

Для выполнения команд в структуре процессора существуют внутренние регистры, арифметико-логическое устройство (АЛУ, ALU — Arithmetic Logic Unit), мультиплексоры, буферы, регистры и другие узлы. Работа всех узлов синхронизируется с помощью общего внешнего тактового сигнала процессора.

Однако на определенном этапе можно считать процессор черным ящиком, который получая на входе управляющие коды производит некую операцию и выдает сигналы на выходе. Для этого достаточно знать систему команд, режимы работы процессора и протоколы обмена информацией, как и о внутренней структуре процессора необходимо только то, что нужно для выбора определенной команды или режима работы, а также иметь общее, представление о физических процессах, протекающих внутри.

II. Отладочный комплект М/СХ 1986ВЕ91Т(94Т)

Комплект отладочный для м/сх 1986ВЕ91Т(94Т) предназначен для [13]:

- демонстрации функционирования и оценки производительности микроконтроллера 1986ВЕ91Т(94Т) и его основных периферийных модулей;
- демонстрации функционирования интерфейсных микросхем CAN и RS-232 интерфейсов;
- отладки собственных проектов с применением установленных на плате интерфейсных микросхем, разъемов и жидкокристаллического дисплея;
- программирования памяти программ микроконтроллеров 1986ВЕ91Т(94Т).

Внешний вид устройства отладочного для м/сх 1986ВЕ91Т(94Т) приведен на рис 5.

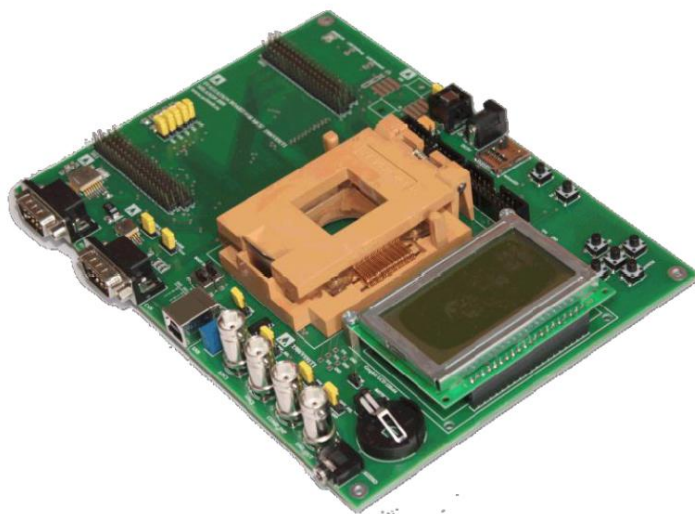


Рисунок 5. Внешний вид устройства отладочного для м/сх 1986ВЕ91Т(94Т)



Рисунок 6. Программатор-отладчик MT-LINK

Органы управления и коммутации, установленные на устройстве отладочном для м/сх 1986BE91T(94T), показаны на рис. 7, их описание содержится в таблице 2.

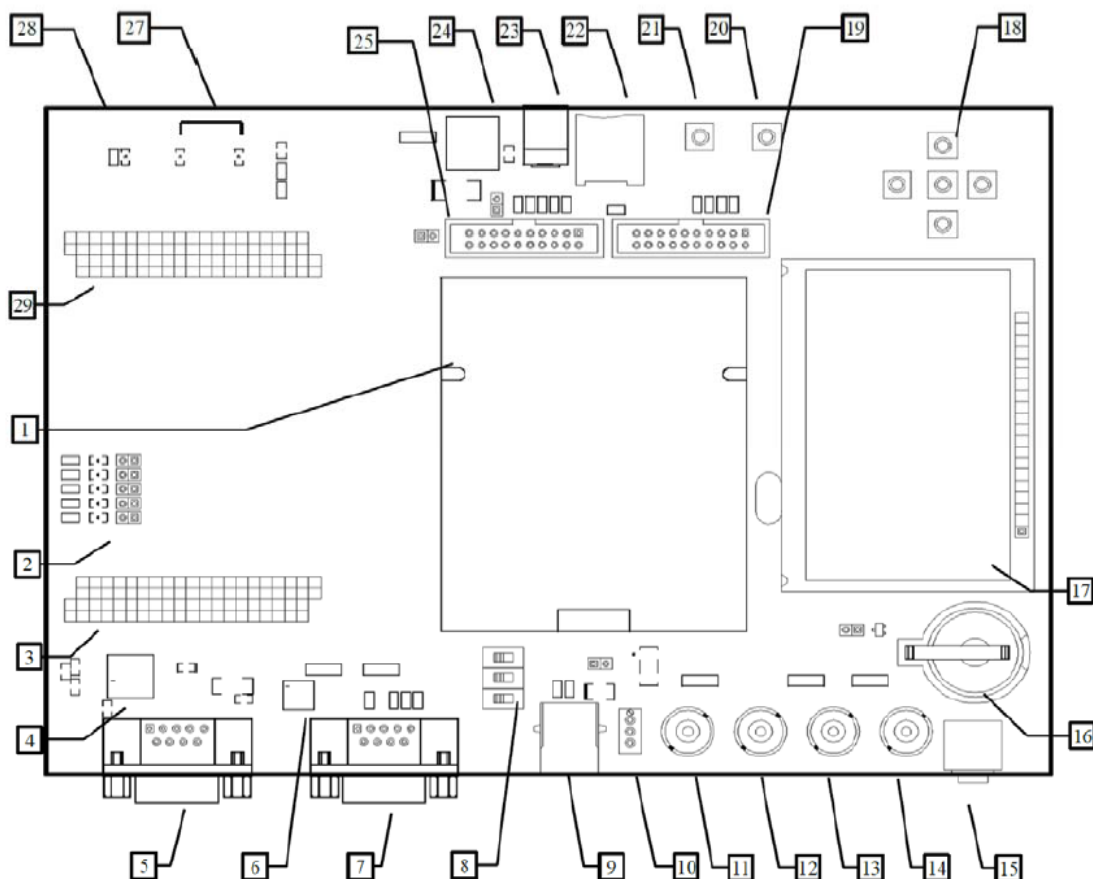


Рисунок 7. расположение органов управления и коммутации на устройстве отладочном для м/сх 1986BE91T(94T)

Таблица 2.

| Номер позиции на рисунке 6 | Описание |
|----------------------------|---|
| 1. | Контактирующее устройство для микроконтроллера 1986BE91T(94T). Микроконтроллер должен быть установлен в спутник-держатель СН132/0,625 ВШУК.301156.005ТУ |
| 2. | Набор светодиодов на порту D |
| 3. | Разъемы внешней системной шины XP8 и XP9 |
| 4. | Приемо-передатчик RS-232 5559ИН4У |
| 5. | Разъем RS-232 |
| 6. | Приемо-передатчик CAN 5559ИН14У |
| 7. | Разъем CAN |
| 8. | Переключатели выбора режима загрузки |
| 9. | Разъем USB-B. |
| 10. | <u>Подстроечный резистор</u> на 7-м канале АЦП. |
| 11. | Разъем BNC внешнего сигнала на 7-м канале АЦП. |
| 12. | Разъем BNC внешнего сигнала на 8-м канале АЦП. |
| 13. | Разъем BNC внешнего сигнала на 3-м входе компаратора. |
| 14. | Разъем BNC выхода ЦАП1. |
| 15. | Разъем <u>Audio</u> 3,5 мм выхода ЦАП1 через звуковой усилитель |
| 16. | Батарея 3,0 В. |
| 17. | ЖК индикатор 128x64. |
| 18. | Кнопки UP, DOWN, LEFT, RIGHT, SELECT. |
| 19. | Разъем отладки JTAG-B. |
| 20. | Кнопка WAKEUP. |
| 21. | Кнопка RESET. |
| 22. | Разъем карты памяти <u>microSD</u> . |
| 23. | Разъем питания 5В. |
| 24. | Фильтр питания. |
| 25. | Разъем отладки JTAG-A. |
| 26. | Светодиоды перегрузки по питанию 1,8В и 3,3В |
| 27. | Светодиод питания 5В |
| 28. | Разъемы внешней системной шины XP10 и XP11 |

Подключение портов микроконтроллера к разъемам XP8, XP9, XP10 и XP11 показано в таблице 3.

Таблица 3.

| Вывод | № контакта контактного устройства для м/сх 1986ВЕ91Т(94Т) | Подключение к разъему | | | | |
|---------------|---|-----------------------|-----|------|------|---------------|
| | | XP8 | XP9 | XP10 | XP11 | Дополнительно |
| Порт А | | | | | | |
| PA0 | 130 | | | 38 | | |
| PA1 | 129 | | | 35 | | |
| PA2 | 128 | | | 36 | | |
| PA3 | 127 | | | 31 | | |
| PA4 | 126 | | | 32 | | |
| PA5 | 125 | | | 29 | | |
| PA6 | 124 | | | 30 | | |
| PA7 | 123 | | | 27 | | |

| Вывод | № контакта контактного устройства для м/сх 1986ВЕ91Т(94Т) | Подключение к разъему | | | | |
|---------------|---|-----------------------|-----|------|------|---------------|
| | | XP8 | XP9 | XP10 | XP11 | Дополнительно |
| PA8 | 122 | | | 28 | | |
| PA9 | 121 | | | 25 | | |
| PA10 | 119 | | | 26 | | |
| PA11 | 118 | | | 23 | | |
| PA12 | 117 | | | 24 | | |
| PA13 | 115 | | | 21 | | |
| PA14 | 114 | | | 22 | | |
| PA15 | 113 | | | 19 | | |
| Порт В | | | | | | |
| PB0 | 92 | | | 20 | | XP5(13) |
| PB1 | 93 | | | 17 | | XP5(7) |
| PB2 | 94 | | | 18 | | XP5(9) |
| PB3 | 95 | | | 15 | | XP5(5) |
| PB4 | 96 | | | 16 | | XP5(3) |
| PB5 | 102 | | | 13 | | |
| PB6 | 103 | | | 14 | | |
| PB7 | 104 | | | 11 | | |
| PB8 | 105 | | | 12 | | |
| PB9 | 106 | | | 9 | | |
| PB10 | 107 | | | 10 | | |
| PB11 | 108 | | | 7 | | |
| PB12 | 109 | | | 8 | | |
| PB13 | 110 | | | 6 | | |
| PB14 | 111 | | | 4 | | |
| PB15 | 112 | | | 3 | | |

| Порт С | | | | | |
|---------------|----|--|--|----|--|
| PC0 | 91 | | | 10 | |
| PC1 | 90 | | | 11 | |
| PC2 | 89 | | | 12 | |
| PC3 | 88 | | | 5 | |
| PC4 | 87 | | | 6 | |
| PC5 | 86 | | | 7 | |
| PC6 | 85 | | | 8 | |
| PC7 | 84 | | | 9 | |
| PC8 | 83 | | | 25 | |

| Вывод | № контакта контактного устройства для м/сх 1986ВЕ91Т(94Т) | Подключение к разъему | | | | |
|---------------|---|-----------------------|-----|------|------|---------------|
| | | XP8 | XP9 | XP10 | XP11 | Дополнительно |
| PC9 | 82 | | | | 27 | |
| PC10 | 81 | | | | 30 | |
| PC11 | 80 | | | | 29 | |
| PC12 | 79 | | | | 23 | |
| PC13 | 78 | | | | 26 | |
| PC14 | 77 | | | | 28 | |
| PC15 | 76 | | | | 24 | |
| Порт D | | | | | | |
| PD0 | 65 | | | | 17 | XP4(7) |
| PD1 | 66 | | | | 22 | XP4(9) |
| PD2 | 67 | | | | 13 | XP4(3), X2(7) |
| PD3 | 68 | | | | 15 | XP4(5), X2(2) |
| PD4 | 64 | | | | 18 | XP4(13) |
| PD5 | 69 | | | | 16 | X2(5) |
| PD6 | 70 | | | | 14 | X2(3) |
| PD7 | 63 | | | | | XP6 |
| PD8 | 62 | | | | | XS6 |
| PD9 | 71 | | 9 | | | |
| PD10 | 61 | 29 | | | | XP18 |
| PD11 | 60 | 30 | | | | XP19 |
| PD12 | 59 | 31 | | | | XP16 |
| PD13 | 58 | 32 | | | | XP21 |
| PD14 | 57 | 34 | | | | XP22 |
| PD15 | 56 | | 10 | | | |

| Порт E | | | | | | |
|---------------|----|----|----|--|--|------------|
| PE0 | 53 | 21 | | | | |
| PE1 | 52 | 22 | | | | |
| PE2 | 45 | 23 | | | | |
| PE3 | 44 | 24 | | | | |
| PE4 | 42 | | 24 | | | |
| PE5 | 41 | | 23 | | | |
| PE6 | 33 | | | | | OSC_IN32 |
| PE7 | 32 | | | | | OSC_OUT32 |
| PE8 | 43 | | | | | XP13 |
| PE9 | 51 | | | | | XP12, XP13 |

| Вывод | № контакта контактного устройства для м/сх 1986ВЕ91Т(94Т) | Подключение к разъему | | | | |
|-------|---|-----------------------|-----|------|------|---------------|
| | | XP8 | XP9 | XP10 | XP11 | Дополнительно |
| PE10 | 50 | | | | | |
| PE11 | 23 | | 21 | | | |
| PE12 | 20 | 25 | | | | |
| PE13 | 19 | 26 | | | | |
| PE14 | 40 | 27 | | | | |
| PE15 | 18 | 28 | | | | |
| | | | | | | |

| Порт F | | | | | | |
|----------------------|----|----|----|--|----|------------------|
| PF0 | 2 | | 13 | | | |
| PF1 | 3 | | 14 | | | |
| PF2 | 4 | 5 | | | | |
| PF3 | 5 | 6 | | | | |
| PF4 | 6 | 7 | | | | |
| PF5 | 7 | 8 | | | | |
| PF6 | 8 | 9 | | | | |
| PF7 | 9 | 10 | | | | |
| PF8 | 10 | 11 | | | | |
| PF9 | 11 | 12 | | | | |
| PF10 | 12 | 13 | | | | |
| PF11 | 13 | 14 | | | | |
| PF12 | 14 | 15 | | | | |
| PF13 | 15 | 16 | | | | |
| PF14 | 16 | 19 | | | | |
| PF15 | 17 | 20 | | | | |
| Системное управление | | | | | | |
| RESET | 37 | | | | 21 | XP5(15), XP4(15) |
| WAKEUP | 35 | | | | | |
| STANDBY | 31 | | | | | |
| OSC_IN | 38 | | | | | |
| OSC_OUT | 39 | | | | | |
| USB интерфейс | | | | | | |
| DP | 21 | | | | | |
| DN | 22 | | | | | |

| Выход | № контакта контактного устройства для м/сх 1986BE91T(94T) | Подключение к разъему | | | | |
|---------|---|-----------------------|-----|------|------|---------------|
| | | XP8 | XP9 | XP10 | XP11 | Дополнительно |
| Питание | | | | | | |
| Ucc | 1,28,29,72,73,98,99 | | | | | |
| AUcc | 55 | | | | | |
| AUcc1 | 48,49 | | | | | |
| BUcc | 30 | | | | | |
| GND | 26,27,74,100,132 | | | | | |
| AGND | 54 | | | | | |
| AGND1 | 46,47 | | | | | |

Назначение установленных на плате конфигурационных перемычек (джамперов):

POWER_SEL(XP1)–выбор источника питания платы:

- USB–разъем USB XS5;
- EXT_DC–внешний источник питания +5В.

SLEWRATE(XP20) –выбор скорости передачи данных интерфейса CAN между 125кбит/с и 500кбит/с.

CANLOAD(XP17) –выбор нагрузки линии CAN между 60 Ом и 120 Ом.

ADC_INP_SEL(XP6) –выбор источника сигнала для 7-го канала АЦП между переменным резистором R32 (TRIM) и BNC разъемом XS4 (ADC1).

COMP_INP_SEL(XP13) –выбор источника сигнала на 3-м входе компаратора между BNC разъемом XS12 («COMP_INP») и выходом ЦАП1.

DAC_OUT_SEL(X20) –выбор назначения сигнала с выхода ЦАП1 между BNC разъемом XS10 («DAC_OUT») и звуковым усилителем D7.

Назначение установленных на плате переключателей и клавиш:

Переключатели:

- SA2, SA3, SA4 – переключатели выбора режима работы BOOT SELECT.

Режимы работы отладочного устройства, в зависимости от положения переключателей SA2, SA3 и SA4, показаны в таблице 4.

Таблица 4.

| SA4 | SA3 | SA2 | Режим работы |
|-----|-----|-----|--|
| 0 | 0 | 0 | Режим микроконтроллера, код выполняется из Flash памяти, начиная с адреса 0x0800_0000, отладка через разъем JTAG_B. |
| 0 | 0 | 1 | Режим микроконтроллера, код выполняется из Flash памяти, начиная с адреса 0x0800_0000, отладка через разъем JTAG_A. |
| 0 | 1 | 0 | Режим микропроцессора, код выполняется из внешней памяти, начиная с адреса 0x1000_0000, отладка через разъем JTAG_B. |
| 0 | 1 | 1 | Режим микропроцессора, код выполняется из внешней памяти, начиная с адреса 0x1000_0000. JTAG заблокирован. |
| 1 | 1 | 0 | Микроконтроллер через интерфейс UART2 на выводах PF[1:0] получает код программы в ОЗУ для исполнения. |

Кнопки:

- UP, DOWN, LEFT, RIGHT, SELECT – программируемые пользователем клавиши;

- RESET – сигнал аппаратного сброса МК;

- WAKEUP – сигнал внешнего выхода из режима Standby.

Для демонстрации функционирования, устройство отладочное подключается к:

- к COM порту персонального компьютера;

- к CAN или COM (RS-232) интерфейсу дополнительного внешнего устройства, например, аналогичному устройству отладочному;

- к источнику питания +5В.

Для программирования памяти программ микроконтроллеров 1986BE91T(94T) применяется внешний внутрисхемный программатор ULINK2 (Keil), J-LINK (SEGGER) или JEM-ARM-V2 (Phyton).

Питание устройства осуществляется от адаптера постоянного тока напряжением +5 вольт или от шины USB.

III. Меры безопасности при работе с лабораторным отладочным макетом

Отладочный макет предназначен для использования в качестве средства разработки аппаратуры и программного обеспечения в условиях учебной/промышленной лаборатории. Для облегчения использования компоненты платы и соединительные проводники открыты для пользователя и окружающей среды.

При работе с отладочным макетом может произойти разряд статического электричества, что может повлечь повреждение компонентов платы. Поэтому, устройство должно иметь постоянную защиту от электростатического разряда. В дополнение к изложенным выше, необходимо соблюдать следующие рекомендации:

- Пока питание платы включено, не изменяйте подключение электронных компонентов на разъёмах зоны макетирования.
- Пока питание платы включено, не касайтесь открытых проводников и электронных компонентов.
- Будьте осторожны при манипуляциях с переключателями, кнопками, ручками управления при включённом питании платы.
- Перед переноской устройства или начала работы с ним уравняйте потенциалы Вашего тела и платы касанием снятия заряда статического электричества.

ЛАБОРАТОРНАЯ РАБОТА № 1. Знакомство с отладочной платой для микроконтроллера семейства 1986VE9X компании Миландр и средой программирования Keil μ Vision

Цель работы:

Познакомиться с устройством микроконтроллера семейства 1986VE9X компании Миландр, его спецификацией, с отладочной платой и средой программирования Keil μ Vision. Создание простейшего проекта и конфигурирование среды разработки.

Порядок выполнения работы:

Подготовка отладочной платы к работе

1. Перед выполнением заданий лабораторной работы познакомьтесь с спецификацией микроконтроллера семейства 1986VE9X компании Миландр, его архитектурой и техническим описанием отладочной платы.
2. Подключить программатор к порту JTAG-B платы.
3. Установить переключатели SW1, SW2 и SW3 в положение 0.
4. Подключить блок питания к плате.
5. Подключить программатор J-Link к USB порту компьютера и дождаться окончания установки драйверов.
6. Открыть проект MDRProject в среде программирования Keil μ Vision.

Среда программирования Keil μ Vision

При выполнении цикла лабораторных работ на микроконтроллере будет использоваться интегрированная среда разработки (ИСП) (Integrated Development Environment - IDE) компании Keil (An ARM Company). IDE Keil μ Vision поддерживает разработку программ для различных ARM-микроконтроллеров (версия μ Vision 3.0 и выше), в том числе для микроконтроллеров серии 1986 фирмы «Миландр» (версия μ Vision 4.2 и выше). Укомплектована C/C++ компилятором, ассемблером, отладчиком, средствами для трассировки и внутрисхемного программирования, а также

поддерживает USB JTAG адаптеры J-Link и ULINK2. Имеется бесплатная полнофункциональная версия с ограничением кода программы в 32Кб.

Установка и первый запуск интегрированной среды разработки Keil uVision

Среду программирования Keil uVision можно найти на сайте разработчика по адресу <https://www.keil.com/download/product/>. Здесь необходимо скачать и установить MDK-ARM (требуется регистрация).

Для поддержки микроконтроллеров серии 1986BE9x необходимо скачать и установить последнюю версию библиотеки стандартной периферии от ПКК Миландр <http://milandr.ru/index.php?page=programmnoe-obespech>.

Также требуется установить драйверы JTAG-отладчика Segger Jlink.

После установки всех компонентов среды программирования Keil uVision можно приступать к созданию первого проекта. Для этого необходимо создать новый проект Project – New uVision Project... (рисунок 8) и выбрать директорию для нового проекта (в названии пути к директории не должно быть русских символов).

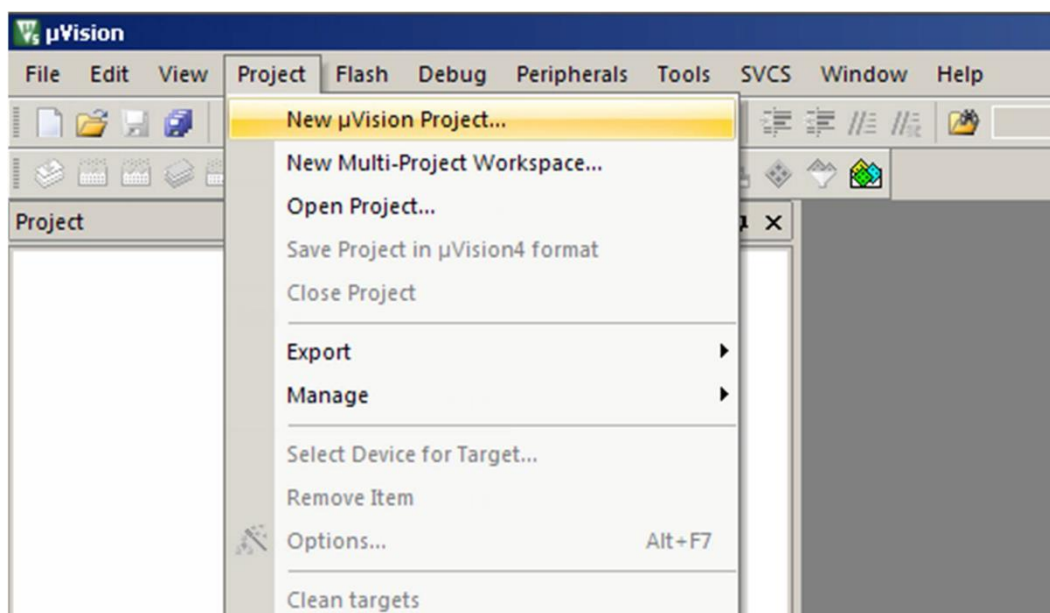


Рисунок 8 –Создание нового проекта.

Затем будет предложено выбрать устройство, для которого будет написана программа. Выбираем микроконтроллер MDR1986BE9X (Milandr – Milandr – Cortex-M3- MDR1986BE9X).

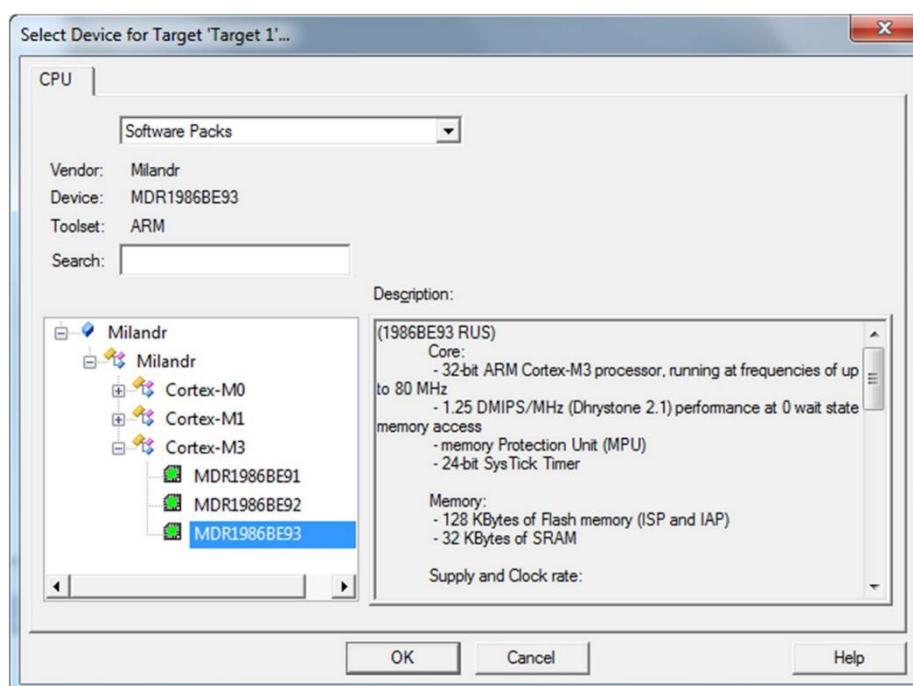


Рисунок 9 –Создание нового проекта.

После выбора целевого микроконтроллера необходимо определить периферию и окружение компиляции. Для первого проекта выберем (рисунок 10) Device\Startup_MDR1986BE9x (поддержка ядра микроконтроллера серии 1986BE9x), Drivers\PORT (порты ввода/вывода), Drivers\RST_CLK (сигналы тактовой частоты).

| Software Component | Sel. | Variant | Version | Description |
|---------------------|-------------------------------------|---------|---------|---|
| CMSIS | <input checked="" type="checkbox"/> | | | Cortex Microcontroller Software Interface Components |
| CMSIS Driver | <input checked="" type="checkbox"/> | | | Unified Device Drivers compliant to CMSIS-Driver Specifications |
| Device | <input checked="" type="checkbox"/> | | | Startup_System_Setup |
| Startup_MDR1986BE9x | <input checked="" type="checkbox"/> | | 1.3.1 | System Startup for MDR1986BE9x device series |
| Drivers | <input checked="" type="checkbox"/> | | | Select packs 'ARM.CMSIS.3.20.x' and 'Keil.MDK-Middleware.5.1.x' for compatibility |
| ADC | <input type="checkbox"/> | | 1.3.1 | ADC driver for MDR1986BE9x Series |
| BKP | <input type="checkbox"/> | | 1.3.1 | BKP driver for MDR1986BE9x Series |
| CAN | <input type="checkbox"/> | | 1.3.1 | CAN driver for MDR1986BE9x Series |
| COMP | <input type="checkbox"/> | | 1.3.1 | COMP driver for MDR1986BE9x Series |
| DAC | <input type="checkbox"/> | | 1.3.1 | DAC driver for MDR1986BE9x Series |
| DMA | <input type="checkbox"/> | | 1.3.1 | DMA driver for MDR1986BE9x Series |
| EBC | <input type="checkbox"/> | | 1.3.1 | EBC driver for MDR1986BE9x Series |
| EEPROM | <input type="checkbox"/> | | 1.3.1 | EEPROM driver for MDR1986BE9x Series |
| I2C | <input type="checkbox"/> | | 1.3.1 | I2C driver for MDR1986BE9x Series |
| IWDG | <input type="checkbox"/> | | 1.3.1 | IWDG driver for MDR1986BE9x Series |
| LIB | <input type="checkbox"/> | | 1.3.1 | LIB file for MDR1986BE9x Series |
| PORT | <input checked="" type="checkbox"/> | | 1.3.1 | PORT driver for MDR1986BE9x Series |
| POWER | <input type="checkbox"/> | | 1.3.1 | POWER driver for MDR1986BE9x Series |
| RST_CLK | <input checked="" type="checkbox"/> | | 1.3.1 | RST_CLK driver for MDR1986BE9x Series |
| SSP | <input type="checkbox"/> | | 1.3.1 | SSP driver for MDR1986BE9x Series |
| TIMER | <input type="checkbox"/> | | 1.3.1 | TIMER driver for MDR1986BE9x Series |
| UART | <input type="checkbox"/> | | 1.3.1 | UART driver for MDR1986BE9x Series |
| USB Library | <input type="checkbox"/> | | 1.3.1 | USB Library for MDR1986BE9x Series |
| USB | <input type="checkbox"/> | | 1.3.1 | USB driver for MDR1986BE9x Series |
| File System | <input type="checkbox"/> | MDK-Pro | 6.2.0 | File Access on various storage devices |
| Graphics | <input type="checkbox"/> | MDK-Pro | 5.26.1 | User Interface on graphical LCD displays |

Рисунок 10 – Окно выбора окружения компиляции

Затем в дереве проекта щелкнуть правой кнопкой мыши на «Source group 1» и выбрать «add new item to Group 'Source group 1'» (рис. 11). В появившемся окне выбрать файл расширения «*.h», задать ему имя «MDR32F9Qx_board.h» и сохранить в папку «config» в той же директории, что и проект.

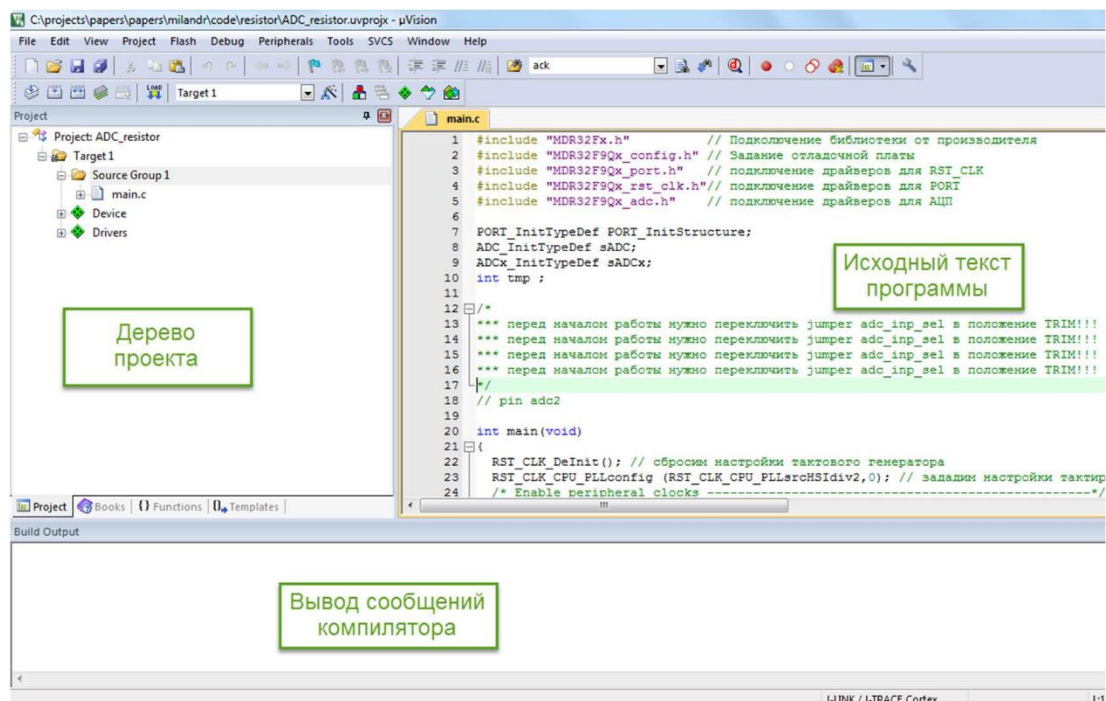


Рисунок 11 –Главное окно программы Keil uVision5

Далее необходимо указать компилятору путь до файла «MDR32F9Qx_board.h». Для этого в дереве проекта щелкнуть правой

кнопкой мыши на «Target 1», выбрать «Options for Target 1» и перейти на вкладку «C/C++». В поле «Include path» добавить строку «./config» (рисунок 12).

Далее необходимо подключить демонстрационно-отладочную плату 1986EvBrd к компьютеру. Для этого нужно подключить JTAG-отладчик при помощи шлейфа к разъему «JTAG-B» отладочной платы.

Выбрать режим загрузки «Flash/JTAG_V», установкой значений «0» и «0» на переключателях «SW1» и «SW2» соответственно.

Подключить отладчик к компьютеру при помощи кабеля USB, удостовериться, что операционная система правильно обнаружила и установила драйверы устройства.

Завершающий шаг - настройка JTAG-отладчика. Для его настройки необходимо в дереве проекта щелкнуть правой кнопкой мыши на «Target 1 - Options for Target 1» и перейти на вкладку «Debug» (рисунок 13).

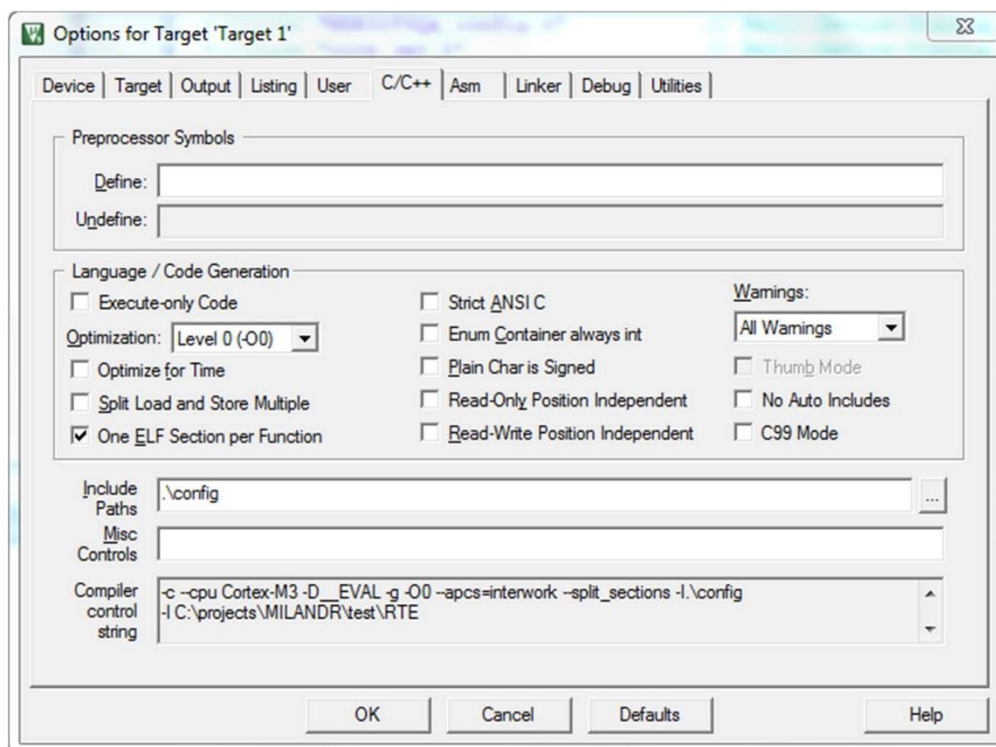


Рисунок 12 – Окно настроек Target 1, вкладка C/C++

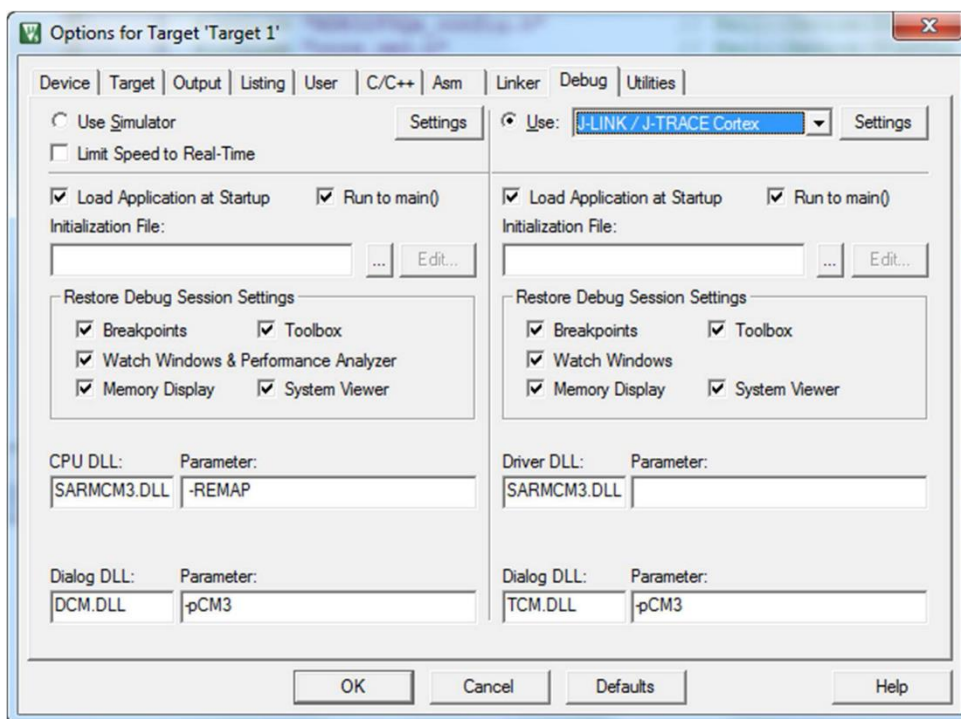


Рисунок 13 – Окно настроек Target 1, вкладка Debug

В выпадающем списке выбрать «J-LINK/J-TRACE Cortex». Затем нажать кнопку «Settings». В выпадающем меню «Port» необходимо выбрать режим «SW» (рисунок 14), и задать скорость передачи данных («Max Clock») не более 2 MHz. Удостовериться, что в поле «SW Device» присутствует устройство ARMCoresightSW-DP.

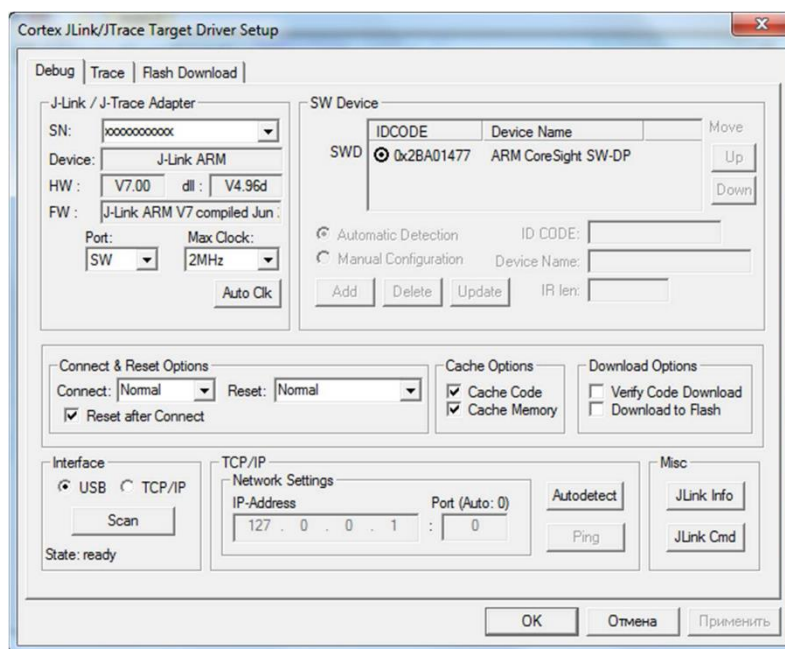


Рисунок 14 – Окно настроек отладчика, вкладка Debug

Далее необходимо задать порядок загрузки программы в память микроконтроллера. Для этого перейти на вкладку «Flash Download» (рисунок 15), нажать кнопку «add» и из списка выбрать «1986BE IAP 128kB Flash». Затем отметить позиции: «Erase Full Chip», «Program», «Verify», «Reset and Run» на вкладке «Flash Download».

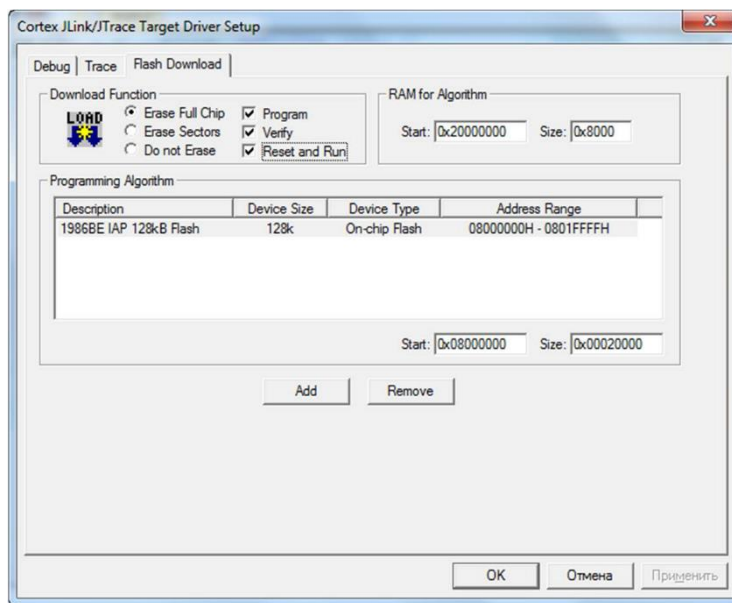


Рисунок 15 –Окно настроек отладчика, вкладка Flash Download

Теперь проект настроен и готов к работе: можно вводить текст программы, отлаживать ее и компилировать. Для этого необходимо добавить в проект новый файл main.c (в дереве проекта щелкнуть правой кнопкой мыши на «Source group 1 - add new item to Group `Source group 1"и выбрать «CFile (.c)»). В появившемся окне можно писать код программы. Например, любой код из представленных в методических указаниях.

Далее нужно скомпилировать код программы. Для этого необходимо выбрать «Project-Build target», либо нажать на соответствующую кнопку на панели инструментов или «горячей» клавишей F7 (рисунок 16).

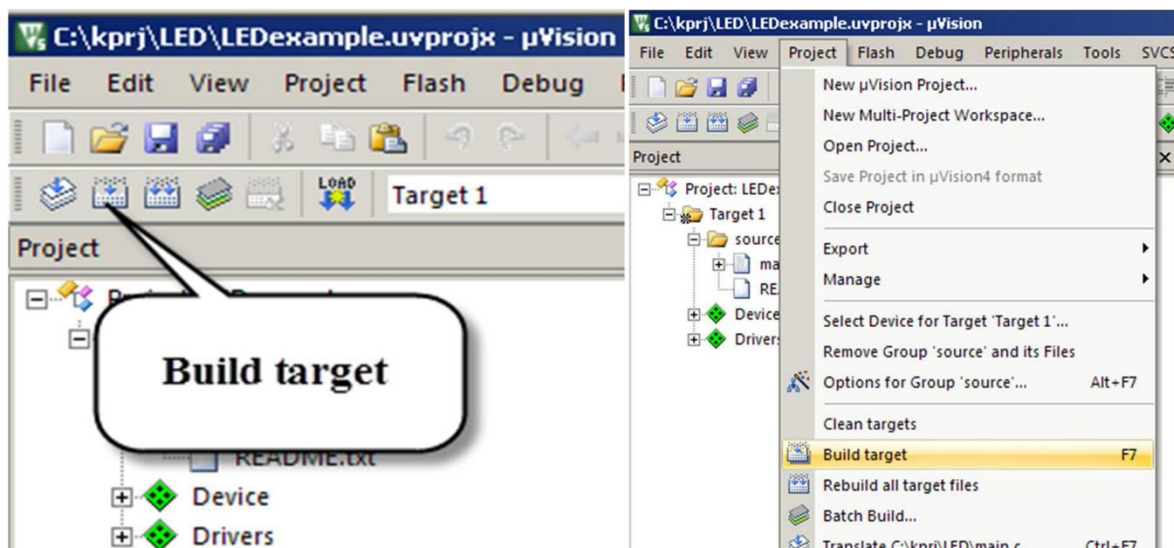


Рисунок 16 –Компиляция кода программы Скомпилированная без ошибок программа может быть загружена в память микроконтроллера. Для этого нужно выбрать «Flash-Download» или нажать на соответствующую кнопку на панели инструментов (рисунок 17).

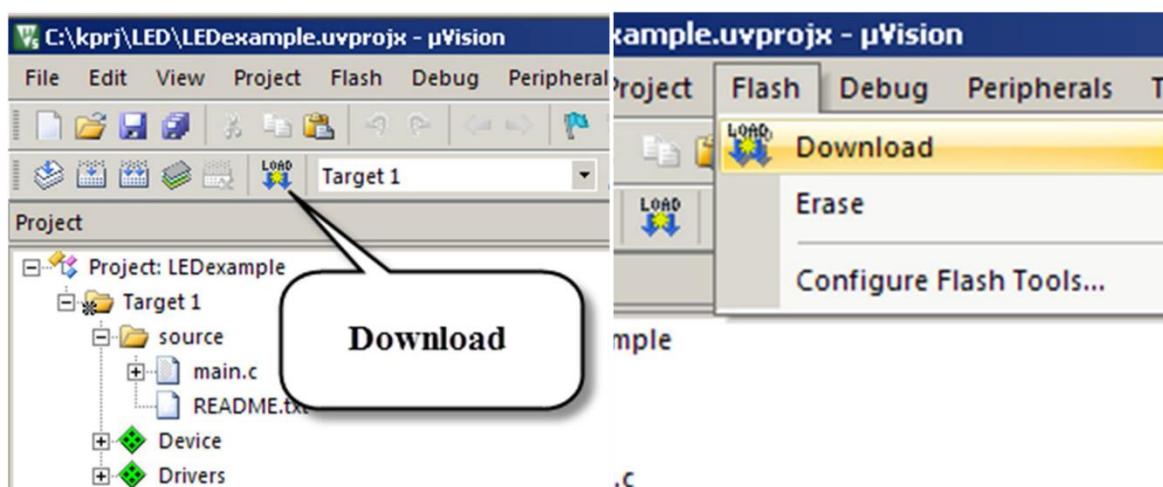


Рисунок 17 –Загрузка скомпилированного кода программы в память

Для отладки программы в среде программирования Keil uVision могут быть использованы средства JTAG-отладчика. Для этого необходимо нажать на кнопку «start/stop debug» (рисунок 18). Режим отладки предоставляет возможность ставить точки остановки выполнения программы на микроконтроллере.

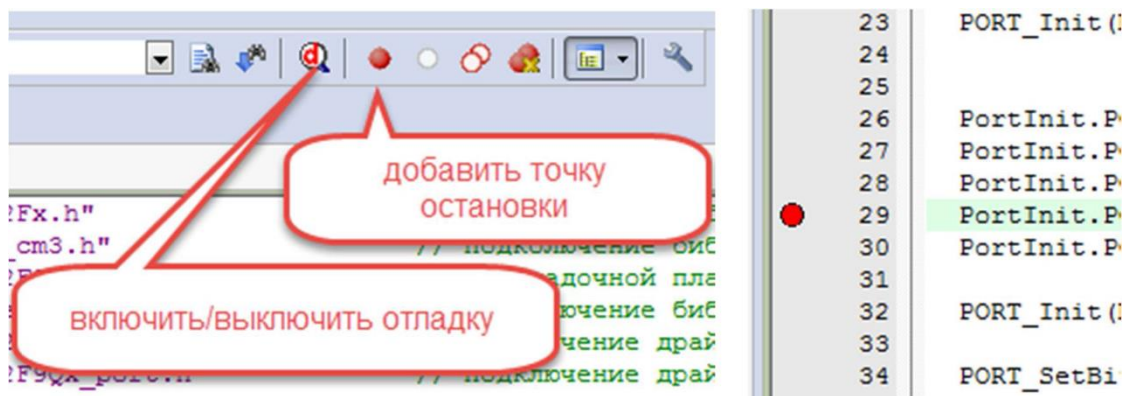


Рисунок 18 –Использование режима отладки

После нажатия кнопки «start/stop debug» окно программы Keil uVision изменяется и принимает вид, показанный на рисунке19. При этом программа на микроконтроллере не выполняется до тех пор, пока пользователь не нажмет соответствующую кнопку.

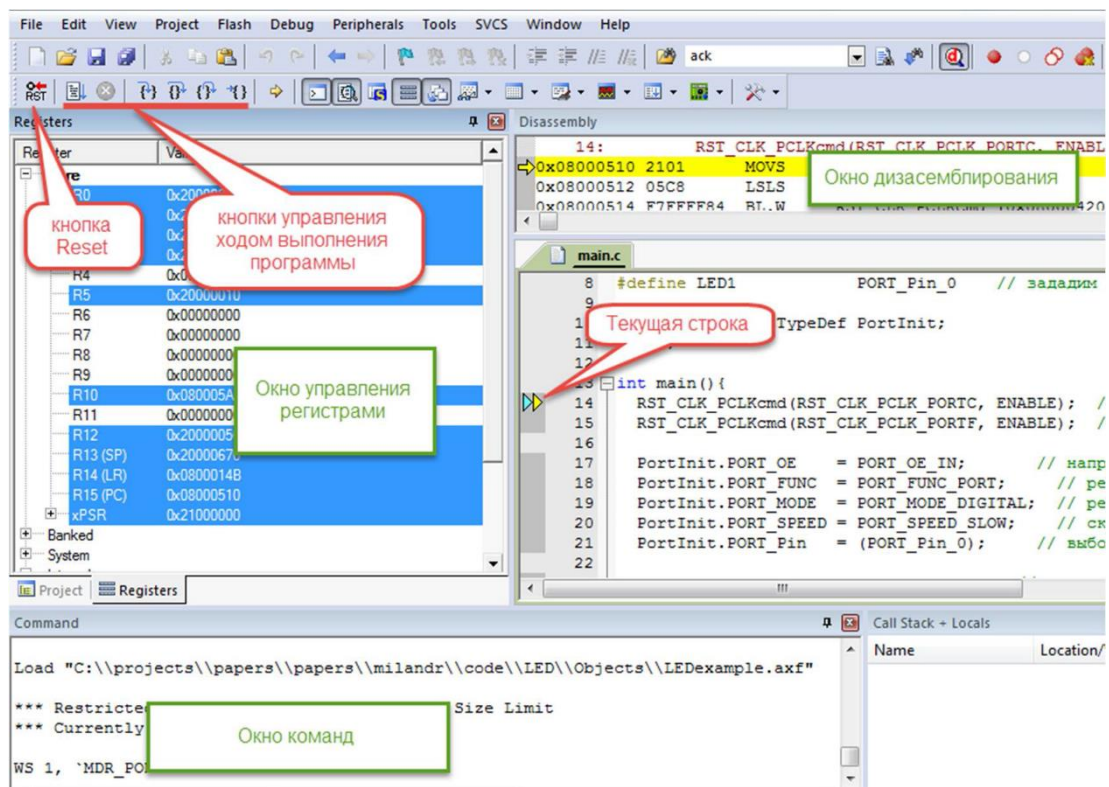


Рисунок 19 –Главное окно в режиме отладки

Для управления ходом выполнения программы предусмотрен соответствующий блок кнопок:

- Кнопка Run запускает выполнение программы на микроконтроллере;
- Кнопка Stop останавливает выполнение программы;
- Кнопка Step выполняет переход к следующей строке по ходу выполнения программы;
- Кнопка Step over выполняет переход к следующей строке по ходу выполнения программы, не заходя в текущую функцию;
- Кнопка Step out выполняет переход к точке выхода из текущей функции.

В любой момент времени текущая строка подсвечивается при помощи символа стрелки в окне отображения кода программы. Над окном кода программы отображается окно дизассемблирования, в котором показывается ход выполнения программы в кодах ассемблера. Слева от окна кода программы отображается окно управления регистрами микроконтроллера, позволяющее просматривать и редактировать все регистры микроконтроллера. Под окном управления регистрами располагается командное окно, позволяющее выполнять произвольные команды в любой момент времени.

Для выхода из режима отладки необходимо нажать на кнопку «start/stop debug».

Требования к отчету

Успешная сдача лабораторной работы предполагает наличие в итоговом отчете следующего содержания.

1. Название, цель и краткий конспект лабораторной работы.
2. Описание хода выполнения задания.
3. Описание с пояснением функций периферийных элементов отладочной платы.
4. Описание назначений и функций разъемов и переключателей платы.
5. Вывод по результатам проделанной работы.

ЛАБОРАТОРНАЯ РАБОТА № 2. Порты ввода-вывода.

Управление светодиодом

Цель работы:

Изучение основ программирования для микроконтроллеров (МК) ARM на примере программы мигания светодиодами. Изучение работы портов ввода-вывода микроконтроллера 1986VE91T.

Порты ввода/вывод

Микроконтроллер имеет 6 портов ввода/вывода. Порты 16-разрядные, и их выходы мультиплексируются между различными функциональными блоками, управление для каждого вывода отдельное. Для того, чтобы выходы порта перешли под управление того или иного периферийного блока, необходимо задать для нужных выводов выполняемую функцию и настройки.

Для работы с портами ввода/вывода используются библиотека MDR32F9Qx_port.h, которая описывает следующие регистры:

- MDR_PORTA
- MDR_PORTB
- MDR_PORTC
- MDR_PORTD
- MDR_PORTE
- MDR_PORTF

Для инициализации используется структура типа PORT_InitTypeDef с полями:

- PORT_OE – направление передачи данных
- PORT_FUNC – режим работы вывода порта
- PORT_MODE – режим работы контроллера
- PORT_SPEED – скорость фронта вывода
- PORT_Pin – выбор выводов для инициализации

Функциональное назначение портов приведено в таблице 120 и 121 стр.184 спецификации микроконтроллеров серии 1986BE9x.

Задание:

Разработать устройство управления одним светодиодным индикатором при помощи одной кнопки. При каждом нажатии кнопки светодиод должен поочередно включаться и отключаться. При первом нажатии кнопки светодиод должен включиться, при следующем отключиться и т.д.

Модифицируйте задачу следующим образом. Кнопка должна включать и выключать мигание светодиода. Пока кнопка отпущена, светодиод не должен светиться. Всё время, пока кнопка нажата, светодиод должен мигать с частотой 5 Гц.

Мигание светодиодом с помощью кнопки

К выводу 0 порта C подключена кнопка («Select»), к порту F – светодиоды (VD2 и VD3). При старте загораются все светодиоды на порту F. Пока кнопка не нажата, светодиод на выводе 0 порта F (VD2) выключен, при нажатии на кнопку светодиод загорается.

Листинг 1

```
#include "MDR32Fx.h"
#include "core_cm3.h"
#include "MDR32F9Qx_config.h"
#include "system_MDR32F9Qx.h"
#include "MDR32F9Qx_rst_clk.h"
#include "MDR32F9Qx_port.h"

#define LED1    PORT_Pin_0 //определить нулевой вывод как LED1
static PORT_InitTypeDef PortInit; //объявление структуры PortInit
int main(){
    RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTC, ENABLE); //включить
    тактирование порта C
```

```
RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTF,          ENABLE);  
);//включить тактирование порта F
```

```
//Инициализация порта C на вход  
PortInit.PORT_OE = PORT_OE_IN; // направление передачи данных =  
вход  
PortInit.PORT_FUNC = PORT_FUNC_PORT; // режим работы вывода  
порта = Порт  
PortInit.PORT_MODE = PORT_MODE_DIGITAL; // режим работы  
вывода =цифровой  
PortInit.PORT_SPEED = PORT_SPEED_SLOW; // скорость фронта  
вывода= медленный  
PortInit.PORT_Pin = (PORT_Pin_0); // выбор вывода 0 для  
инициализации  
PORT_Init(MDR_PORTC, &PortInit); //инициализация порта C  
заданными параметрами
```

```
//Инициализация порта F на выход  
PortInit.PORT_OE = PORT_OE_OUT; // направление передачи данных =  
Выход  
PortInit.PORT_FUNC = PORT_FUNC_PORT; // режим работы вывода  
порта = Порт  
PortInit.PORT_MODE = PORT_MODE_DIGITAL; // режим работы  
вывода = Цифровой  
PortInit.PORT_SPEED = PORT_SPEED_SLOW; // скорость фронта  
вывода = медленный  
PortInit.PORT_Pin = (PORT_Pin_All);// выбор всех выводов для  
инициализации  
PORT_Init(MDR_PORTF, &PortInit); //инициализация порта F  
заданными параметрами
```

```
PORT_SetBits(MDR_PORTF, PORT_Pin_All); // включить все  
светодиоды при старте
```

```
while(1){ //бесконечный цикл  
if (PORT_ReadInputDataBit(MDR_PORTC,PORT_Pin_0) == 0)  
//если кнопка не нажата...  
{  
PORT_SetBits(MDR_PORTF, LED1); // включить светодиод на выводе 0  
порта F  
}  
else //иначе  
{  
PORT_ResetBits(MDR_PORTF, LED1); // выключить светодиод на  
выводе 0 порта F  
}  
}  
}
```

Содержание отчета

1. Титульный лист, содержащий название работы, номер варианта индивидуального задания, ФИО, группу студента.
2. Цель работы
3. Краткие теоретические сведения.
4. Электрическая принципиальная схема устройства.
5. Код программы
6. Выводы.

ЛАБОРАТОРНАЯ РАБОТА № 3. «Бегущие огни»

Цель работы:

Закрепление навыков работы с командами микроконтроллера, и организации циклов.

Задание

Разработать автомат «Бегущие огни». Светодиоды должны переключаться согласно описанию, приведенному в таблице по вариантам, с частотой 1 – 2 Гц.

Светодиоды, мигающие раз в секунду

К выводам 0 и 1 порта F подключены светодиоды(VD2 и VD3), которые должны переключаться раз в секунду. Это можно сделать с помощью прерываний при переполнении системного таймера SysTick.

Листинг 2

```
#include "MDR32Fx.h"
#include "core_cm3.h"
#include "MDR32F9Qx_config.h"
#include "system_MDR32F9Qx.h"
#include "MDR32F9Qx_rst_clk.h"
#include "MDR32F9Qx_port.h"

static PORT_InitTypeDef PortInit;//объявление структуры PortInit volatile
uint32_t delay_dec = 0;// объявление переменной delay_dec

//Обработчик прерывания системного таймера void SysTick_Handler
(void)
{
    if (delay_dec !=0) delay_dec--;//вычитать из delay_dec, пока не станет
    равен 0
}
```

```

//функция временной задержки
void delay_ms (uint32_t delay_ms)
{
    delay_dec = delay_ms; //присвоить delay_dec значение delay_ms
    while (delay_dec) {}; // выполнять функцию пока delay_dec не станет
    равным 0
}

int main(){

    RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTF,          ENABLE);
}; //включить тактирование порта F

//Инициализация порта F на выход
PortInit.PORT_OE = PORT_OE_OUT; // направление передачи данных =
Выход
PortInit.PORT_FUNC = PORT_FUNC_PORT; // режим работы вывода
порта = Порт
PortInit.PORT_MODE = PORT_MODE_DIGITAL; // режим работы
вывода = цифровой
PortInit.PORT_SPEED = PORT_SPEED_SLOW; // скорость фронта
вывода = медленная
PortInit.PORT_Pin = (PORT_Pin_All); // выбор вывода для
инициализации

PORT_Init(MDR_PORTF, &PortInit); //инициализация порта
F заданными параметрами

//Инициализация системного таймера

```

```
    SysTick->LOAD |= (8000000/1000)-1; //значение задержки прерывания  
при тактовой частоте 8 МГц = 1мс
```

```
    SysTick->CTRL |= SysTick_CTRL_CLKSOURCE_Pos; //источник  
тактирования HCLK
```

```
    SysTick->CTRL |= SysTick_CTRL_COUNTFLAG_Pos; // при  
досчитывании до нуля таймер генерирует прерывание
```

```
    SysTick->CTRL |= ~SysTick_CTRL_ENABLE_Pos; //включить работу  
таймера
```

```
    while(1){  
        delay_ms(1000); //задержка 1 с  
        PORT_SetBits(MDR_PORTF, PORT_Pin_All); // включить светодиоды  
        delay_ms(1000); //задержка 1 с  
        PORT_ResetBits(MDR_PORTF, PORT_Pin_All); // ВЫКЛЮЧИТЬ  
светодиоды  
    }  
}
```

Содержание отчета

1. Титульный лист, содержащий название работы, номер варианта индивидуального задания, ФИО, группу студента.
2. Цель работы
3. Краткие теоретические сведения.
4. Электрическая принципиальная схема устройства.
5. Код программы
6. Выводы.

Варианты индивидуальных заданий

| № варианта | Модификация | № варианта | Модификация |
|------------|---|------------|--|
| 1 | На линейке светодиодов бегущий огонь с верхнего светодиода вниз. | 16 | На линейке светодиодов бегущий огонь с нижнего светодиода вверх. |
| 2 | Бегущий огонь начиная сверху по два светодиода вниз. | 17 | Бегущий огонь начиная снизу по два светодиода вверх. |
| 3 | Начиная с 3-го светодиода бегущий огонь вниз. | 18 | Начиная с 6-го светодиода бегущий огонь вверх. |
| 4 | Начиная с середины, два огня движутся в разные стороны. | 19 | Начиная с краев, два огня движутся на встречу друг другу. |
| 5 | Горят все светодиоды, а потухший начинает двигаться с верхнего вниз. | 20 | Горят все светодиоды, а потухший начинает двигаться нижнего вверх. |
| 6 | Движение начинается с крайнего светодиода вверх. | 21 | Движение начинается с 3-го светодиода вверх. |
| 7 | Бегущий огонь с верхнего светодиода через один вниз. | 22 | Бегущий огонь с нижнего светодиода через один вверх. |
| 8 | Начиная с 4-го нижнего вверх. | 23 | Начиная с 4-го верхнего вниз. |
| 9 | Начиная со 2-го верхнего, по два зажженных светодиода вверх. | 24 | Начиная со 2-го нижнего, по два зажженных светодиода вниз. |
| 10 | Бегущий огонь с 6-го через один вверх. | 25 | Бегущий огонь со 2-го через один вниз. |
| 11 | Начиная с 5-го, светодиоды загораются по одному вниз. | 26 | Начиная с 5-го, светодиоды загораются по одному вверх. |
| 12 | С первого верхнего, по одному, загораются все светодиоды. Дойдя до последнего, гаснут в обратном направлении. | 27 | С первого нижнего, по одному, загораются все светодиоды. Дойдя до последнего, гаснут в обратном направлении. |
| 13 | Начиная с верхнего, по одному загораются все светодиоды. Следом гаснут по два в том же направлении. | 28 | Начиная с нижнего, по одному загораются все светодиоды. Следом гаснут по два в том же направлении. |
| 14 | Начиная с верхнего, через два, загораются светодиоды. Потом, со второго через два. Далее с третьего, через два. | 29 | Начиная с нижнего, через два, загораются светодиоды. Потом, со второго через два. Далее с третьего, через два. |
| 15 | Бегущий огонь с нижнего светодиода вверх и обратно. | 30 | Бегущий огонь с верхнего светодиода вниз и обратно. |

ЛАБОРАТОРНАЯ РАБОТА № 4. Логические выражения

Цель работы:

Изучение логических операций и операций ветвления микроконтроллера серии 1986BE9х, операций ввода/вывода данных с помощью кнопок и светодиодов.

Задание

Реализовать устройство определения попадания точки в заштрихованную область. Область ограничена линиями контура.

Ввод координат точек по оси X осуществить с помощью кнопок KEY1-KEY4, по оси Y – кнопок KEY5-KEY8. Проверка попадания точки в область должна производиться по нажатию кнопки SELECT. Вывод результата реализовать на светодиоды LED1-LED8. При попадании точки в

заштрихованную область светодиоды LED1-LED8 должны моргнуть один раз, если точка не попадает в область – моргнуть два раза.

Комбинация выставленных состояний кнопок соответствует 3 разрядному целому числу со знаком в бинарном виде. Старший разряд отвечает за знак (если кнопка включена – то число с минусом). Пример сформированных чисел показан на рисунке 21.

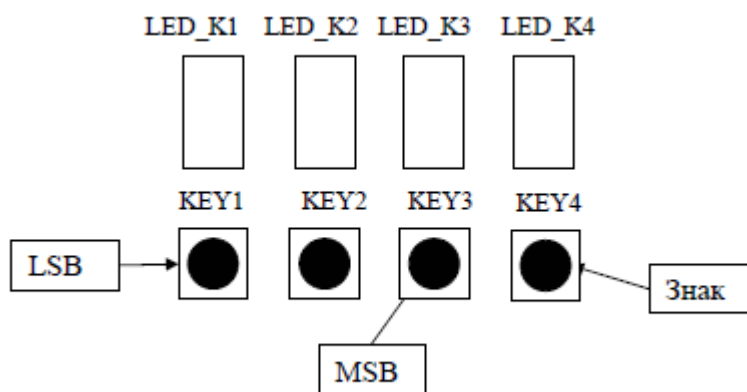


Рисунок 20 – Структура формирования числа

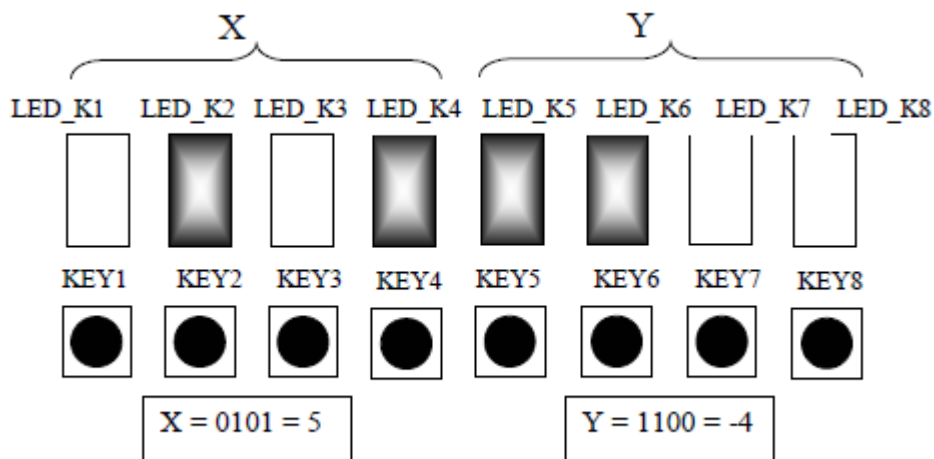
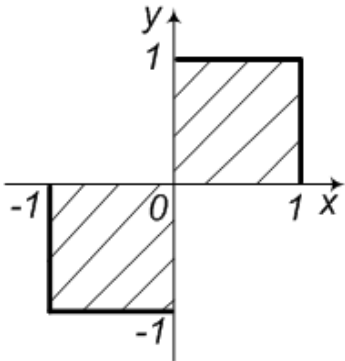
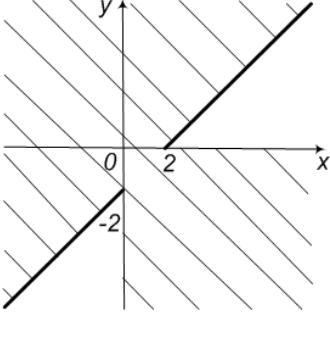
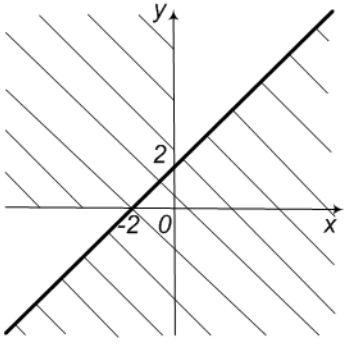
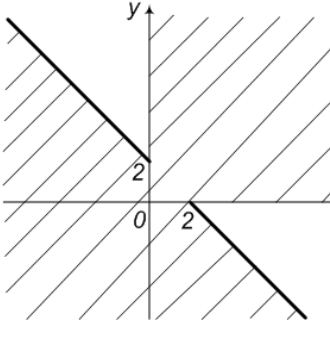


Рисунок 21 – Пример формирования координат по Хи Y

Варианты индивидуального задания

| № варианта | Область | № варианта | Область |
|------------|---------|------------|---------|
| 1 | | 11 | |
| 2 | | 12 | |
| 3 | | 13 | |

| | | | |
|---|--|----|--|
| 4 | | 14 | |
| 5 | | 15 | |
| 6 | | 16 | |
| 7 | | 17 | |
| 8 | | 18 | |

| | | | |
|----|---|----|---|
| 9 |  | 19 |  |
| 10 |  | 20 |  |

Содержание отчета

1. Титульный лист, содержащий название работы, номер варианта индивидуального задания, ФИО, группу студента.
2. Цель работы
3. Краткие теоретические сведения.
4. Электрическая принципиальная схема устройства.
5. Код программы
6. Выводы.

ЛАБОРАТОРНАЯ РАБОТА № 5. Таймеры общего назначения

Цель работы:

Изучение основных особенностей работы с таймером

Таймеры общего назначения

Все блоки таймеров выполнены на основе 16-битного перезагружаемого счетчика, который синхронизируется с выхода 16-битного предделителя. Перегружаемое значение хранится в отдельном регистре. Счет может быть прямой, обратный или двунаправленный (сначала прямой до определенного значения, а затем обратный).

Каждый из трех таймеров микроконтроллера содержит 16-битный счетчик, 16-битный предделитель частоты и 4-канальный блок захвата/сравнения. Их можно синхронизировать системной синхронизацией, внешними сигналами или другими таймерами.

Помимо составляющего основу таймера счетчика в каждый блок таймера также входит четырехканальный блок захвата/сравнения. Данный блок выполняет как стандартные функции захвата и сравнения, так и ряд специальных функций. Таймеры имеют 4 канала схем захвата и ШИМ с функциями формирования «мертвой зоны» и аппаратной блокировки. Каждый из таймеров может генерировать прерывания и запросы DMA.

Для работы с таймерами используется структура

TIMER_CntInitTypeDef с полями:

TIMER_Prescaler – значение величины предделителя

TIMER_Period – период таймера

TIMER_CounterMode – режим счета

TIMER_CounterDirection – направление счета

TIMER_EventSource – источник событий для таймера

TIMER_FilterSampling – указывает фильтр событий

TIMER_ARR_UpdateMode – режим сброса счетчика

TIMER_ETR_FilterConf – задает параметры выхода ETR

TIMER_ETR_Prescaler – задает параметры предделителя фильтра
выхода ETR

TIMER_ETR_Polarity – задает полярность выхода ETR

TIMER_BRK_Polarity – задает полярность выхода BRK Структура
TIMER_ChnInitTypeDef полями:

TIMER_CH_Mode – задает режим работы таймера

TIMER_CH_REF_Format – формат выработки сигнала REF в режиме
ШИМ

TIMER_CH_Number – номер канала таймера Структура
TIMER_ChnOutInitTypeDef полями:

TIMER_CH_DirOut_Polarity – полярность выхода CHx таймера

TIMER_CH_DirOut_Source – задает сигнал на выходе CHx таймера

TIMER_CH_DirOut_Mode – задает сигнал на выходе CHx таймера

TIMER_CH_NegOut_Polarity – полярность инверсного выхода CHx
таймера

TIMER_CH_NegOut_Source – задает сигнал на инверсном выходе CHx
таймера

TIMER_CH_NegOut_Mode – задает сигнал на инверсном выходе CHx
таймера

TIMER_CH_Number – номер канала

Описание работы таймеров общего назначения представлено на стр.274
спецификации микроконтроллеров серии 1986BE9х.

Задание

Реализуйте передачу символа на ПК каждые 5 секунд через интерфейс
RS-232 с использованием таймеров.

Вывод 5 порта В назначен в качестве линии передачи TXD
приемопередатчика UART1, а вывод 6 того же порта – в качестве линии
приема RXD. Для преобразования уровней последовательных сигналов,
используемых микроконтроллером к уровням используемых интерфейсом

RS-232 на демонстрационно-отладочной плате 1986EvBrd используется микросхема 5559ИН4.

При старте микроконтроллер начинает отсылать по последовательному интерфейсу на вход приемопередатчика UART1 цифровое значение переменной *i* каждые 5 секунды. Значение переменной *i* после каждой отправки увеличивается на 1.

Листинг 3

```
#include "MDR32F9Qx_config.h"
#include "MDR32Fx.h"
#include "MDR32F9Qx_uart.h"
#include "MDR32F9Qx_port.h"
#include "MDR32F9Qx_rst_clk.h"

static PORT_InitTypeDef PortInit;//объявление структуры PortInit
static UART_InitTypeDef UART_InitStructure;//объявление структуры
UART_InitStructure

volatile uint32_t delay_dec = 0;// объявление переменной delay_dec

//Обработчик прерывания системного таймера(см. 4.2) void
SysTick_Handler (void)
{
    if (delay_dec !=0) delay_dec--;
}

//функция временной задержки(см. 4.2) void delay_ms
(uint32_t delay_ms)
{
    delay_dec = delay_ms; while (delay_dec) {};
```

```

}

int main (void)
{
    uint8_t i = 0; //объявление переменной счетчика, хранящей передаваемое
по UART значение

    //Инициализация системного таймера для функции задержки
    SysTick->LOAD |= (8000000/1000)-1;
    SysTick->CTRL |= SysTick_CTRL_CLKSOURCE_Pos;

    SysTick->CTRL |= SysTick_CTRL_COUNTFLAG_Pos; SysTick->CTRL
|= ~SysTick_CTRL_ENABLE_Pos;

    RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTB,ENABLE);
); //включить тактирование порта B

    //Инициализация порта B для функции UART PortInit.PORT_PULL_UP
= PORT_PULL_UP_OFF; //подтяжка в питание выключена
    PortInit.PORT_PULL_DOWN = PORT_PULL_DOWN_OFF; //подтяжка в
ноль выключена
    PortInit.PORT_PD_SHM = PORT_PD_SHM_OFF; //режим триггера
Шмитта выключен
    PortInit.PORT_PD = PORT_PD_DRIVER; //режим управляемого
драйвера
    PortInit.PORT_GFEN = PORT_GFEN_OFF; //входной фильтр выключен
    PortInit.PORT_FUNC = PORT_FUNC_ALTER; //альтернативная функция
порта
    PortInit.PORT_SPEED = PORT_SPEED_MAXFAST; //скорость фронта
вывода = быстрая

```



```
PortInit.PORT_MODE = PORT_MODE_DIGITAL; //режим работы
вывода =цифровой
```

```
//Инициализация вывода 5 как UART_TX PortInit.PORT_Pin =
PORT_Pin_5; PORT_Init(MDR_PORTB, &PortInit);
```

```
//Инициализация вывода 6 как UART_RX PortInit.PORT_OE =
PORT_OE_IN; PortInit.PORT_Pin = PORT_Pin_6; PORT_Init(MDR_PORTB,
&PortInit);
```

```
RST_CLK_CPU_PLLconfig
(RST_CLK_CPU_PLLsrcHSIdiv2,0);//конфигурация источника тактирования
HSI делением частоты на 2 и без умножения
```

```
RST_CLK_PCLKcmd(RST_CLK_PCLK_UART1, ENABLE);
);//включить тактирование UART1
```

```
UART_BRGInit(MDR_UART1, UART_HCLKdiv1);//делитель тактовой
частоты UART = 1
```

```
//КонфигурацияUART
```

```
UART_InitStructure.UART_BaudRate = 115000; //скорость передачи
данных 15000 бод/сек
```

```
UART_InitStructure.UART_WordLength = UART_WordLength8b;//длина
слова в посылке = 8бит
```

```
UART_InitStructure.UART_StopBits = UART_StopBits1;//один стоповый
бит
```

```
UART_InitStructure.UART_Parity = UART_Parity_No;//без проверки
четности
```

```
UART_InitStructure.UART_FIFOmode = UART_FIFO_ON;//включить
работу буфера FIFO приемника и передачи
```

```

UART_InitStructure.UART_HardwareFlowControl           =
UART_HardwareFlowControl_RXE |
    UART_HardwareFlowControl_TXE;//Разрешить прием, разрешить
передачу данных

//ИнициализацияUART1 с заданными параметрами UART_Init
(MDR_UART1,&UART_InitStructure);
    UART_Cmd(MDR_UART1,ENABLE); //включить сконфигурированный
UART while (1) //бесконечный цикл
    {
        UART_SendData (MDR_UART1, i++); //Послать значение счетчика по
UART1, и прибавить к нему 1
        delay_ms(5000);//подождать 5 секунд
    }
}

```

Содержание отчета

1. Титульный лист, содержащий название работы, номер варианта индивидуального задания, ФИО, группу студента.
2. Цель работы
3. Краткие теоретические сведения.
4. Электрическая принципиальная схема устройства.
5. Код программы
6. Выводы.

ЛАБОРАТОРНАЯ РАБОТА № 6. Контроллер прерываний

Цель работы:

Изучение основных особенностей работы с таймерами с использованием прерывания при программировании микроконтроллеров ARM.

Контроллер прерываний

Векторный контроллер прерываний с возможностью вложения (NVIC Nested Vectored Interrupt Controller) обеспечивает:

- программное задание уровня приоритета в диапазоне от 0 до 15 независимо каждому прерыванию. Более высокое значение соответствует меньшему приоритету, таким образом, уровень 0 отвечает наивысшему приоритету прерывания;
- срабатывание сигнала прерывания по импульсу и по уровню;
- динамическое изменение приоритета прерываний;
- разделение исключений по группам с одинаковым приоритетом и по подгруппам внутри одной группы;
- передача управления из одного обработчика исключения на другой без восстановления контекста.

Процессор автоматически сохраняет в стеке свое состояние (контекст) по входу в обработчик прерывания и восстанавливает его по завершению обработчика без необходимости непосредственного программирования этих операций. Это обеспечивает обработку исключительных ситуаций с малой задержкой.

Поскольку прерывание может возникнуть при выполнении любой произвольной команды фона, её адрес запоминается в так называемом программном стеке. После чего выполнение передается на часть программы, специально написанную разработчиком для реакции на событие, вызвавшее данное прерывание. Эта небольшая часть программы называется обработчиком прерывания. Когда обработчик будет выполнен до конца, программа, воспользовавшись адресом, сохранённым в программном стеке,

вернётся в то место, откуда была вызвана для обработки данного прерывания.

Для использования вектора прерывания необходимо:

- Разрешить использование прерываний в программе;
- Разрешить вызов интересующего нас прерывания специальным битом в соответствующем регистре;
- Создать условия для возникновения прерывания, например, если это переполнение таймера, то запустить его;
- Разместить в программе обработчик прерывания, оформив его в соответствии с требованиями компилятора.

Для разрешения использования прерываний в программе используется функция `void NVIC_EnableIRQ(IRQn_t IRQn)`, в которую передается имя прерывания.

Для разрешения вызовов прерываний у каждого модуля описан свой регистр, например для таймера `TIMER_ITConfig (MDR_TIMER1, TIMER_STATUS_CNT_ARR, ENABLE)`;

Описание прерываний происходит в файле, формат которого предоставляет производитель микроконтроллера `MDR32F9Qx_it.c` и соответствующем ему заголовочном файле `MDR32F9Qx_it.h`.

Задание

Доработать программу лабораторной работы № 3 «Бегущие огни», изменив процедуру формирования задержки. Создать программу с использованием прерываний по таймеру.

Содержание отчета

1. Титульный лист, содержащий название работы, номер варианта индивидуального задания, ФИО, группу студента.
2. Цель работы
3. Краткие теоретические сведения.
4. Электрическая принципиальная схема устройства.
5. Код программы

6. Выводы.

.

ЛАБОРАТОРНАЯ РАБОТА № 7. Модель осветительных приборов автомобиля

Цель работы:

Построение и реализация модели внешних световых приборов автомобиля, закрепление навыков организации циклов и реализации команд ветвления.

Задание

Используя программы предыдущих лабораторных работ, разработать автомат, моделирующий осветительные приборы автомобиля (рис. 22).

Зажигание светодиодов осуществляется по следующей схеме:

Первый (верхний) тумблер – сигнал правого поворота (соответствующий светодиод «мигает»);

Второй тумблер – включение фар (соответствующие светодиоды горят);

Третий тумблер – включение габаритных огней (соответствующие светодиоды горят);

Четвертый тумблер – сигнал левого поворота (соответствующий светодиод «мигает»);

Пятый тумблер – аварийная сигнализация (указатели правого и левого поворота «мигают» одновременно).

Светодиоды распределяются следующим образом:



Рисунок 22. Модель осветительных приборов автомобиля

Запуск аварийной сигнализации не допускается одновременным включением правого и левого сигнала поворотов. При одновременном включении первого и четвертого тумблеров, должен активизироваться сигнал ошибки, проявляющийся в виде «мигания» всех восьми светодиодов порта.

Содержание отчета

1. Титульный лист, содержащий название работы, номер варианта индивидуального задания, ФИО, группу студента.
2. Цель работы
3. Краткие теоретические сведения.
4. Электрическая принципиальная схема устройства.
5. Код программы
6. Выводы.

ЛАБОРАТОРНАЯ РАБОТА № 8. «Светофор»

Цель работы:

Построение и реализация модели управления светофора.

Задание

Используя программы предыдущих лабораторных работ, разработать автомат, моделирующий осветительные приборы автомобиля (рис. 23).

Для реализации модели задайте три фиксированные временные задержки для трех светодиодов, например, А – 10 секунд, В – 4 секунды, С – 1 секунда, и алгоритм работы светофора будет следующим:

«Красный» - А – «Красный + Желтый» - В – «Зеленый» - А – «Все погашено» - С – «Зеленый» - С – «Все погашено» - С – «Зеленый» - С – «Все погашено» - С – «Зеленый» - С – «Все погашено» - С – «Зеленый» - С – «Все погашено» - С – «Зеленый» - С – «Все погашено» - С – «Зеленый» - С – «Все погашено» - С – «Зеленый» - С – «Желтый» - В – «Красный». И так далее – в цикле.



Рисунок 23. Модель светофора

Содержание отчета

1. Титульный лист, содержащий название работы, номер варианта индивидуального задания, ФИО, группу студента.
2. Цель работы
3. Краткие теоретические сведения.
4. Электрическая принципиальная схема устройства.
5. Код программы
6. Выводы.

ЛАБОРАТОРНАЯ РАБОТА № 9. Создание действующей модели движения пассажирского лифта

Цель работы:

Создание модели движения пассажирского лифта, закрепление навыков работы с командами микроконтроллера, организации циклов, работы с командами ветвления и установления таймеров.

Задание

Используя программы предыдущих лабораторных работ, разработать автомат, моделирующий движение пассажирского лифта (рис. 24).

Модель лифта:

Лифт обслуживает пять этажей. На этаж вызывается кратковременным включением соответствующего тумблера порта «А» (после включения тумблер должен быть немедленно возвращен в положение «отключено»). Светодиоды порта «В» имитируют состояние и положение кабины лифта. Тумблеры порта «А» имитируют кнопки вызова кабины на этажах.

Распределить светодиоды порта «В» следующим образом:

| Номер светодиода | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|-------------------------|--------------------|--------------------|--------------------|-----------------------|-------------------|-----------------------------|-----------------------------|------------------------------|
| Назначение | Первый этаж | Второй этаж | Третий этаж | Четвертый этаж | Пятый этаж | Двери кабины открыты | Двери кабины закрыты | Лифт занят (свободен) |

Свечение светодиодов №№ 1,2,3,4,5 отображают нахождение кабины на соответствующем этаже; свечение светодиода № 6 имитирует открытое состояние дверей кабины; свечение светодиода № 7 имитирует закрытое состояние дверей кабины; свечение светодиода № 8 означает, что лифт свободен и может быть вызван на любой этаж. Если светодиод № 8 погашен - значит лифт занят. Во время занятого состояния лифта кнопки вызова (порт «А») должны быть заблокированы, и не позволять произвести вызов кабины.

Пример алгоритма работы:

Пусть кабина лифта находится на четвертом этаже и готова к вызову (горят светодиоды № 4, 7, 8). Произведем вызов кабины на первый этаж, путем кратковременного переключения нижнего тумблера порта «А» в положение «включено» с последующим возвратом в положение «отключено». При этом гаснет светодиод № 8, и через одну секунду лифт начинает двигаться вниз (режим бегущего огня от третьего светодиода к первому). На первом этаже лифт останавливается и открывает двери (горят светодиоды №№ 1, 6; светодиод № 7 гаснет). Через пять секунд лифт закрывает двери (светодиод № 6 гаснет, а № 7 загорается). Через две секунды лифт готов к вызову (горят светодиоды №№ 1, 7, 8).

Содержание отчета

1. Титульный лист, содержащий название работы, номер варианта индивидуального задания, ФИО, группу студента.
2. Цель работы
3. Краткие теоретические сведения.
4. Электрическая принципиальная схема устройства.
5. Код программы
6. Выводы.

ЛАБОРАТОРНАЯ РАБОТА № 10. Оцифровка и фильтрация аналогового сигнала (АЦП)

Цель работы:

Изучить принципы аналогово-цифрового преобразования сигналов, познакомиться с базовыми методами обработки цифрового входного сигнала.

Контроллер АЦП.

В микроконтроллере реализовано два 12-разрядных АЦП. С помощью АЦП можно оцифровать сигнал от 16 внешних аналоговых выводов порта D и от двух внутренних каналов, на которые выводятся датчик температуры и источник опорного напряжения. Скорость выборки составляет до 512 тысяч преобразований в секунду для каждого АЦП.

В качестве опорного напряжения преобразования могут выступать:

- питание АЦП с выводов AVCC и AGND
- внешние сигналы с выводов ADC0_REF+ и ADC_REF-

Контроллер АЦП позволяет:

- оцифровать один из 16 внешних каналов;
- оцифровать значение встроенного датчика температуры;
- оцифровать значение встроенного источника опорного напряжения;
- осуществить автоматический опрос заданных каналов;
- выработать прерывание при выходе оцифрованного значения за заданные пределы;
- запускать два АЦП синхронно для увеличения скорости выборки.

Для осуществления преобразования требуется не менее 28 тактов синхронизации CLK. В качестве синхросигнала может выступать частота процессора CPU_CLK, либо частота ADC_CLK, формируемая в блоке «Сигналы тактовой частоты».

Для работы с портами ввода/вывода используются библиотека MDR32F9Qx_adc.h, которая описывает регистры АЦП с помощью задания структур ADC_InitTypeDef для настройки преобразователя и ADCx_InitTypeDef для настройки канала преобразователя.

Структура ADC_InitTypeDef имеет следующие поля:

- ADC_SynchronousMode – выбор режима работы двух преобразователей;
- ADC_StartDelay – определяет задержку начала преобразований от старта системы [0:15];
- ADC_TempSensor – включение/выключение температурного датчика;
- ADC_TempSensorAmplifier – включение/выключение усилителя температурного датчика;
- ADC_TempSensorConversion – включение/выключение преобразования показаний от температурного датчика;
- ADC_IntVRefConversion – включение/выключение преобразования показаний опорного напряжения;
- ADC_IntVRefTrimming – определяет интервал считывания значений опорного напряжения;

Структура ADCx_InitTypeDef имеет следующие поля:

- ADC_ClockSource – указывает источник тактирующего сигнала;
- ADC_SamplingMode – задает режим считывания показаний;
- ADC_ChannelSwitching – включение/выключение возможности переключения каналов АЦП;
- ADC_ChannelNumber – номер канала;
- ADC_Channels – маска номеров каналов;
- ADC_LevelControl – включение/выключение слежения за уровнем АЦП;
- ADC_LowLevel – значение нижнего уровня АЦП;

- ADC_HighLevel – значение верхнего уровня АЦП;
- ADC_VRefSource – определяет источник питания АЦП;
- ADC_IntVRefSource – определяет тип напряжения источника питания АЦП;
- ADC_Prescaler – задает параметры предусилителя;
- ADC_DelayGo – задержка начала преобразований в последовательном режиме;

Подробное описание регистров блока контроллера АЦП приведено в таблице 292 стр.318 спецификации микроконтроллеров серии 1986BE9х.

Задание

Реализовать устройство оцифровки сигнала, преобразования цифровым фильтром в соответствии с вариантом индивидуального задания и передачу на персональный компьютер (листинг 4).

Устройство должно оцифровывать входной аналоговый сигнал, подаваемого с платы расширения на ножку PD3 и выводить значения по интерфейсу RS-232. С помощью осциллографа производить чтение данных и выводить на экран в виде графика.

При включении кнопки KEY1 должна производиться фильтрация сигнала и передача по интерфейсу RS-232 уже отфильтрованного сигнала.

Для выполнения лабораторной работы необходимо переключить режим работы платы расширения с помощью программного обеспечения «Lab_changer».

Варианты индивидуальных заданий

| № | Задание |
|----|---|
| 1. | Медианный фильтр с ядром 3 |
| 2. | Пороговый фильтр по уровню $0,7U_{max}$ |
| 3. | Медианный фильтр с ядром 7 |
| 4. | Пороговый фильтр по уровню $0,4U_{max}$ |

| | |
|-----|--|
| 5 | Медианный фильтр с ядром 5 |
| 6. | Усредняющий фильтр с ядром 5 |
| 7. | Усредняющий фильтр с ядром 7 |
| 8. | Пороговый фильтр по уровню $0,9U_{\max}$ |
| 9. | Усредняющий фильтр с ядром 3 |
| 10. | Пороговый фильтр по уровню $0,5U_{\max}$ |

Здесь U_{\max} - максимальная амплитуда входного аналогового сигнала

[В]

Вывод значения оцифрованного сигнала, управляемого переменным резистором, на ПК с помощью интерфейса RS-232

К выводу 2 порта D с помощью переключки X4-X5 «ADC_INP_SEL» может быть подключен разъем BNC X2 «ADC» или многооборотный переменный резистор R1 на 10 кОм.

С помощью встроенного высокоскоростного 12-разрядного аналогово-цифрового преобразователя будем оцифровывать сигнал, изменяемый переменным резистором R1 и передавать оцифрованные значения по интерфейсу RS-232 на ПК. Средствами среды математического моделирования MATLAB будем принимать значения оцифрованного сигнала и строить график его изменения.

Листинг 4.

```
#include "MDR32Fx.h
#include "MDR32F9Qx_config.h"
#include "MDR32F9Qx_port.h
#include "MDR32F9Qx_rst_clk.h"
#include "MDR32F9Qx_adc.h"
#include "MDR32F9Qx_uart.h"
```

```

#include <string.h>

PORT_InitTypeDef PortInit;//объявление структуры
PortInit ADC_InitTypeDef sADC;//объявление структуры ADC
ADCx_InitTypeDef sADCx;//объявление структуры ADCx
UART_InitTypeDef    UART_InitStructure;//объявление    структуры
InitStructure

uint16_t tmp; //переменная хранения текущего значения АЦП
char buf[2]; //массив-буфер для передачи двух байтов

//Функция инициализации UART
void init_UART(void)
{
RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTB,ENABLE);

PortInit.PORT_PULL_UP = PORT_PULL_UP_OFF;
PortInit.PORT_PULL_DOWN = PORT_PULL_DOWN_OFF;
PortInit.PORT_PD_SHM = PORT_PD_SHM_OFF;
PortInit.PORT_PD = PORT_PD_DRIVER;
PortInit.PORT_GFEN = PORT_GFEN_OFF;
PortInit.PORT_FUNC = PORT_FUNC_ALTER;
PortInit.PORT_SPEED = PORT_SPEED_MAXFAST;
PortInit.PORT_MODE = PORT_MODE_DIGITAL;

PortInit.PORT_Pin = PORT_Pin_5;
PORT_Init(MDR_PORTB, &PortInit);
PortInit.PORT_Pin = PORT_Pin_6;
PORT_Init(MDR_PORTB, &PortInit);

```

```

RST_CLK_CPU_PLLconfig (RST_CLK_CPU_PLLsrcHSIdiv2,0);
RST_CLK_PCLKcmd(RST_CLK_PCLK_UART1, ENABLE);
UART_BRGInit(MDR_UART1, UART_HCLKdiv1);
UART_InitStructure.UART_BaudRate = 57600;
UART_InitStructure.UART_WordLength = UART_WordLength8b;
UART_InitStructure.UART_StopBits = UART_StopBits1;
UART_InitStructure.UART_Parity = UART_Parity_No;
UART_InitStructure.UART_FIFOMode = UART_FIFO_ON;
UART_InitStructure.UART_HardwareFlowControl =
UART_HardwareFlowControl_RXE | UART_HardwareFlowControl_TXE;
UART_Init (MDR_UART1,&UART_InitStructure);
UART_Cmd(MDR_UART1,ENABLE);
}

```

```

//функция временной задержки (см. 4.2) volatileuint32_tdelay_dec = 0;

```

```

void SysTick_Handler (void)

```

```

{
if (delay_dec !=0) delay_dec--;
}

```

```

void delay_ms (uint32_t delay_ms)

```

```

{
delay_dec = delay_ms; while (delay_dec) {};
}

```

```

int main(void)

```

```

{

```



```

//Инициализация системного таймера для функции задержки
SysTick->LOAD |= (8000000/1000)-1;
SysTick->CTRL |= SysTick_CTRL_CLKSOURCE_Pos;
SysTick->CTRL |= SysTick_CTRL_COUNTFLAG_Pos;
SysTick->CTRL |= ~SysTick_CTRL_ENABLE_Pos;

RST_CLK_PCLKcmd(RST_CLK_PCLK_ADC,
ENABLE);//включение тактирования АЦП
RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTD,ENABLE);//включение
тактирования порта D

PortInit.PORT_Pin = PORT_Pin_2; // номер вывода
PortInit.PORT_OE = PORT_OE_IN; // направление работы вывода-вход
PortInit.PORT_MODE = PORT_MODE_ANALOG; // режим работы-
аналоговый

PORT_Init(MDR_PORTD, &PortInit); //инициализировать порт D с
заданными параметрами

/* настройка АЦП */
sADC.ADC_SynchronousMode = ADC_SyncMode_Independent; // режим
работы АЦП-независимый
sADC.ADC_StartDelay = 0; // задержка включения АЦП от начала
работы = 0
sADC.ADC_IntVRefTrimming = 1; //подстройка опорного напряжения
ADC_Init (&sADC); //инициализировать АЦП с заданными
параметрами

/* настройка канала ADC1 АЦП */
ADCx_StructInit (&sADCx);

```

```

sADCx.ADC_ClockSource = ADC_CLOCK_SOURCE_CPU; // источник
тактирования
sADCx.ADC_SamplingMode =
ADC_SAMPLING_MODE_CICLIC_CONV; // режим считывания
sADCx.ADC_ChannelSwitching = ADC_CH_SWITCHING_Disable; //
выключить переключение каналов
sADCx.ADC_ChannelNumber = ADC_CH_ADC2; // номер канала
sADCx.ADC_Channels = 0; // маска каналов нужна только, если
предусмотрено переключение каналов
sADCx.ADC_LevelControl = ADC_LEVEL_CONTROL_Disable; //
отключить контроль за уровнем сигнала
sADCx.ADC_LowLevel = 0; // задать нижний уровень сигнала
sADCx.ADC_HighLevel = 0; // задать верхний уровень сигнала
sADCx.ADC_VRefSource = ADC_VREF_SOURCE_INTERNAL; //
источник опорного напряжения
sADCx.ADC_IntVRefSource = ADC_INT_VREF_SOURCE_INEXACT;
// тип опорного напряжения
sADCx.ADC_Prescaler = ADC_CLK_div_512; // делитель частоты
sADCx.ADC_DelayGo = 7; // задержка начала измерений

ADC1_Init (&sADCx);
ADC1_Cmd (ENABLE); // включить канал ADC1
init_UART(); // инициализация UART

while(1){
tmp = MDR_ADC->ADC1_RESULT & 0x0FFF; // считать текущее
значение АЦП
memcpy(buf, &tmp, sizeof(uint16_t)); // записать значения в буфер buf

```

```
while(UART_GetFlagStatus (MDR_UART1, UART_FLAG_BUSY)){  
//дождаться, когда UART будет готов к передачи  
    UART_SendData(MDR_UART1, buf[1]);// передать 1-ый байт значения  
АЦП  
  
while(UART_GetFlagStatus (MDR_UART1, UART_FLAG_BUSY)){  
//дождаться, когда UART будет готов к передачи UART_  
SendData(MDR_UART1, buf[0]); }// передать 1-ый байт значения АЦП  
  
delay_ms(500); // подождать 500 мс  
}  
}
```

Содержание отчета

1. Титульный лист, содержащий название работы, номер варианта индивидуального задания, ФИО, группу студента.
2. Цель работы
3. Краткие теоретические сведения.
4. Электрическая принципиальная схема устройства.
5. Код программы
6. Выводы.

ЛАБОРАТОРНАЯ РАБОТА № 11. ШИМ

Цель работы:

Получить навыки использования широтно-импульсной модуляции для управления яркостью свечения светодиода.

Управление яркостью свечения светодиода с помощью программной реализации широтно-импульсной модуляции

Широтно-Импульсная Модуляция (ШИМ) – управление средней мощностью нагрузки с помощью серии высокочастотных импульсов. Регулируется усреднённая мощность изменением длительности импульсов и пауз между ними. Чем длиннее импульсы и короче паузы между ними, тем средняя мощность в нагрузке выше. ШИМ – простой случай цифро-аналогового преобразования – плавного управления напряжением или мощностью цифровыми схемами, имеющими только два состояния.

ШИМ представляет собой импульсный сигнал постоянной частоты и переменной скважности (отношение длительности импульса к периоду его следования). С помощью задания скважности можно менять среднее напряжение на выходе ШИМ. Таким способом, меняя выходную мощность, можно управлять яркостью светодиода.

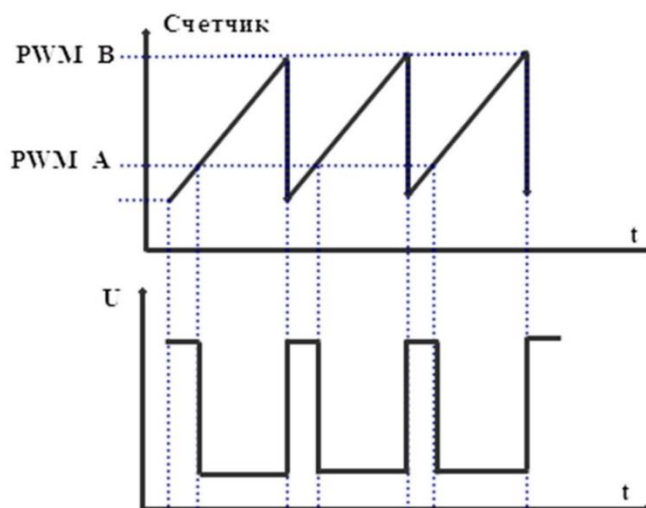


Рисунок 24 – Диаграмма генерации широтно-импульсной модуляции

Задание

К выводу 0 порта F подключен светодиод VD2. К выводу 0 порта C подключена кнопка SELECT. С помощью метода широтно-импульсной модуляции реализуйте управление яркостью светодиода. Кнопка SELECT задает уровень скважности. ШИМ генерируется на основе 8 разрядного счетчика заданного переменной PWM_Counter.

Листинг 5

```
#include "MDR32F9Qx_config.h"
#include "MDR32Fx.h"
#include "MDR32F9Qx_rst_clk.h"
#include "MDR32F9Qx_port.h"

#define LED1      PORT_Pin_0 //ОпределитьPORT_Pin_0какLED1 static
PORT_InitTypeDef PortInit;

//Функция инициализации порта F на выход
void init_leds()
{
    RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTF, ENABLE);
    PortInit.PORT_Pin = (LED1);
    PortInit.PORT_OE = PORT_OE_OUT;
    PortInit.PORT_FUNC = PORT_FUNC_PORT;
    PortInit.PORT_MODE = PORT_MODE_DIGITAL;
    PortInit.PORT_SPEED = PORT_SPEED_FAST;

    PORT_Init(MDR_PORTF, &PortInit);
}

//Функция инициализация порта C на вход (см. 4.1) void init_button()
{
```

```
RST_CLK_PCLKcmd (RST_CLK_PCLK_PORTC, ENABLE);
```

```
PortInit.PORT_Pin = (Button_select);
```

```
PortInit.PORT_OE = PORT_OE_IN;
```

```
PortInit.PORT_FUNC = PORT_FUNC_PORT;
```

```
PortInit.PORT_MODE = PORT_MODE_DIGITAL;
```

```
PortInit.PORT_SPEED = PORT_SPEED_FAST;
```

```
PORT_Init(MDR_PORTC, &PortInit);
```

```
}
```

```
uint8_t PWM_Counter = 0; //объявление счетчика PWM_Counter
```

```
uint8_t PWM_A = 0; //объявление переменной PWM_A (задает  
длительность импульса)
```

```
uint8_t PWM_B = 40; //объявление переменной PWM_B (задает период  
импульса)
```

```
static uint8_t btn_old_state = 0; //переменная, хранящая предыдущее  
состояние кнопки
```

```
static uint8_t btn_state; //переменная, хранящая текущее состояние  
кнопки
```

```
int main(){
```

```
init_leds(); //инициализация светодиода
```

```
init_button(); ///инициализация кнопки
```

```
while(1){ //бесконечный цикл
```

```
//Увеличение переменной PWM_A только в момент нажатия кнопки
```

```
btn_state = PORT_ReadInputDataBit(MDR_PORTC, Button_select);
```

```
//запомнить текущее состояние кнопки (нажата или нет)
```

```
if(btn_old_state == 0 && btn_state == 1)//если кнопка была не нажата, а  
теперь нажата
```

```

{
PWM_A++; //увеличитьPWM_A
if(PWM_A >=PWM_B) // если досчитали до значенияPWM_B
{
PWM_A = 0; //сбросить в ноль
}
}

btn_old_state = btn_state; // запомнить предыдущее состояние кнопки

//Генерация ШИМ
if(PWM_Counter>= PWM_B) //если счетчик досчитал до значения
PWM_B
{
PWM_Counter = 0; //сбросить счетчик
PORT_SetBits(MDR_PORTF, LED1); //включить

} else if (PWM_Counter == PWM_A){ //если счетчик равен PWM_A

PWM_Counter++; //увеличить счетчик
PORT_ResetBits(MDR_PORTF, LED1);

//выключить светодиод
} else { //иначе
PWM_Counter++; //увеличить счетчик
}
}
}

```

Содержание отчета

1. Титульный лист, содержащий название работы, номер варианта индивидуального задания, ФИО, группу студента.
2. Цель работы
3. Краткие теоретические сведения.
4. Электрическая принципиальная схема устройства.
5. Код программы
6. Выводы.

ЛАБОРАТОРНАЯ РАБОТА № 12. Аппаратная реализация широтно-импульсной модуляции

Цель работы:

Закрепление навыков использования аппаратного ШИМ микроконтроллера на основе прерываний от счетчика таймера.

Задание

Реализуйте аппаратный ШИМ на основе примера, приведенного ниже.

Аппаратная реализация широтно-импульсной модуляции

Программная реализация ШИМ приведенная в предыдущей лабораторной работе, используется в крайних случаях, когда на микроконтроллере отсутствует или уже занят вывод специального назначения для таймера. В общем случае рекомендуется реализовывать аппаратный ШИМ микроконтроллера на основе прерываний от счетчика таймера. Таймер микроконтроллера работает независимо от центрального процессора и в таком случае не требует дополнительных вычислительных ресурсов.

На выводах 1, 2, 3, 4, 5 порта А располагаются каналы 1, 2 и 3 таймера1, которые могут быть сконфигурированы в режиме ШИМ. Зафиксировать сигнал ШИМ с разной скважностью можно на ножках 7, 8, 9, 10,12 разъема X25 с помощью цифрового осциллографа.

Листинг 6

```
#include "MDR32F9Qx_config.h"  
#include "MDR32Fx.h"  
#include "MDR32F9Qx_timer.h"  
#include "MDR32F9Qx_rst_clk.h"  
#include "MDR32F9Qx_port.h"
```

```
/*Объявление структур данных*/
```

```

TIMER_CntInitTypeDef sTIM_CntInit;
TIMER_ChnInitTypeDef sTIM_ChnInit;
TIMER_ChnOutInitTypeDef sTIM_ChnOutInit;
PORT_InitTypeDef PORT_InitStructure;

uint16_t CCR1_Val = 500;
uint16_t CCR2_Val = 1000;
uint16_t CCR3_Val = 2000;

int main(void)
{
/* Включим тактирование для устройств периферии*/
RST_CLK_PCLKcmd((RST_CLK_PCLK_RST_CLK), ENABLE);
RST_CLK_PCLKcmd((RST_CLK_PCLK_TIMER1), ENABLE);
RST_CLK_PCLKcmd((RST_CLK_PCLK_PORTA), ENABLE);

/* Конфигурация выводов 1, 2, 3, 4, 5 порта A */
PORT_InitStructure.PORT_Pin = (PORT_Pin_1 | PORT_Pin_2 |
PORT_Pin_3 | PORT_Pin_4 | PORT_Pin_5);
PORT_InitStructure.PORT_OE      = PORT_OE_OUT;
PORT_InitStructure.PORT_FUNC    = PORT_FUNC_ALTER;
PORT_InitStructure.PORT_MODE    = PORT_MODE_DIGITAL;
PORT_InitStructure.PORT_SPEED  = PORT_SPEED_FAST;
PORT_Init(MDR_PORTA, &PORT_InitStructure);

/*Конфигурация таймера
Генерируем 5 ШИМ сигналов с разной скважностью:
TIM1CLK = 8 MHz, Prescaler = 0, TIM1 counter clock = 8 MHz
TIM1 частота = TIM1CLK/(TIM1_Period + 1) = 1.95 KHz

```

TIM1 канал 1 & канал 1N скважность = $TIM1->CCR1 / (TIM1_Period + 1) = 12.5\%$

TIM1 канал 2 & канал 2N скважность = $TIM1->CCR2 / (TIM1_Period + 1) = 25\%$

TIM1 канал 3 скважность = $TIM1->CCR3 / (TIM1_Period + 1) = 50\%$

-- */

/* Настройка счетчика TIMER1

sTIM_CntInit.TIMER_Prescaler = 0x0; // предусилитель

sTIM_CntInit.TIMER_Period = 4000; // период таймера

sTIM_CntInit.TIMER_CounterMode = TIMER_CntMode_ClkFixedDir; //

режим счета

sTIM_CntInit.TIMER_CounterDirection = TIMER_CntDir_Up; //

направление счета

sTIM_CntInit.TIMER_EventSource = TIMER_EvSrc_None; / источник

событий для таймера

sTIM_CntInit.TIMER_FilterSampling =

TIMER_FDTS_TIMER_CLK_div_1;

// указывает фильтр событий

sTIM_CntInit.TIMER_ARR_UpdateMode =

TIMER_ARR_Update_Immediately; // режим сброса счетчика

sTIM_CntInit.TIMER_ETR_FilterConf =

TIMER_Filter_1FF_at_TIMER_CLK; // задает параметры выхода ETR

sTIM_CntInit.TIMER_ETR_Prescaler = TIMER_ETR_Prescaler_None;

// задает предусилитель выхода ETR

sTIM_CntInit.TIMER_ETR_Polarity = TIMER_ETRPolarity_NonInverted;

// задает полярность выхода ETR

sTIM_CntInit.TIMER_BRK_Polarity =

TIMER_BRKPolarity_NonInverted;

```

// задает полярность выхода BRK
TIMER_CntInit (MDR_TIMER1,&sTIM_CntInit);
// инициализация каналов 1,1N,2,2N,3
sTIM_ChnInit.TIMER_CH_Mode = TIMER_CH_MODE_PWM; // режим
работы канала
sTIM_ChnInit.TIMER_CH_REF_Format = TIMER_CH_REF_Format6; //
формат выработки сигнала REF в режиме ШИМ

sTIM_ChnInit.TIMER_CH_Number = TIMER_CHANNEL1; // номер
канала
TIMER_ChnInit(MDR_TIMER1, &sTIM_ChnInit);
sTIM_ChnInit.TIMER_CH_Number = TIMER_CHANNEL2;
TIMER_ChnInit(MDR_TIMER1, &sTIM_ChnInit);

sTIM_ChnInit.TIMER_CH_Number = TIMER_CHANNEL3;
TIMER_ChnInit(MDR_TIMER1, &sTIM_ChnInit);

TIMER_SetChnCompare(MDR_TIMER1, TIMER_CHANNEL1,
CCR1_Val); // задать значение CCR0 для канала 1
TIMER_SetChnCompare(MDR_TIMER1, TIMER_CHANNEL2,
CCR2_Val); // задать значение CCR0 для канала 2
TIMER_SetChnCompare(MDR_TIMER1, TIMER_CHANNEL3,
CCR3_Val); // задать значение CCR0 для канала 3
//Настройки выходных сигналов каналов 1,1N,2,2N,3 Output
sTIM_ChnOutInit.TIMER_CH_DirOut_Polarity =
TIMER_CHOPolarity_NonInverted; // полярность прямого выхода
sTIM_ChnOutInit.TIMER_CH_DirOut_Source =

```

```
TIMER_CH_OutSrc_REF;// источник опорного напряжения прямого  
выхода
```

```
sTIM_ChnOutInit.TIMER_CH_DirOut_Mode =  
TIMER_CH_OutMode_Output;// режим работы прямого выхода = выход  
sTIM_ChnOutInit.TIMER_CH_NegOut_Polarity =  
TIMER_CHOPolarity_NonInverted;// полярность инверсного выхода  
sTIM_ChnOutInit.TIMER_CH_NegOut_Source =  
TIMER_CH_OutSrc_REF;// источник опорного напряжения инверсного
```

```
выхода
```

```
sTIM_ChnOutInit.TIMER_CH_NegOut_Mode =  
TIMER_CH_OutMode_Output;  
// режим работы инверсного выхода = выход
```

```
sTIM_ChnOutInit.TIMER_CH_Number = TIMER_CHANNEL1; // номер  
канала
```

```
TIMER_ChnOutInit(MDR_TIMER1, &sTIM_ChnOutInit);
```

```
sTIM_ChnOutInit.TIMER_CH_Number = TIMER_CHANNEL2;
```

```
TIMER_ChnOutInit(MDR_TIMER1, &sTIM_ChnOutInit);
```

```
sTIM_ChnOutInit.TIMER_CH_Number = TIMER_CHANNEL3;
```

```
TIMER_ChnOutInit(MDR_TIMER1, &sTIM_ChnOutInit);
```

```
TIMER_BRGInit(MDR_TIMER1, TIMER_HCLKdiv1); // Включить  
тактирование TIMER1
```

```
TIMER_Cmd(MDR_TIMER1, ENABLE); //включить TIMER1
```

```
while(1)
```

```
{
```

```
}
```

}

Содержание отчета

1. Титульный лист, содержащий название работы, номер варианта индивидуального задания, ФИО, группу студента.
2. Цель работы
3. Краткие теоретические сведения.
4. Электрическая принципиальная схема устройства.
5. Код программы
6. Выводы.

ЛАБОРАТОРНАЯ РАБОТА № 13. Генерация аналогового сигнала

Цель работы:

Изучить принципы цифро-аналогового преобразования сигналов

Контроллер ЦАП.

В микроконтроллере реализовано два ЦАП. Для включения ЦАП необходимо чтобы бит `Cfg_ON_DACx` был установлен в 1, используемые выводы ЦАП порта E были сконфигурированы как аналоговые и были отключены какие-либо внутренние подтяжки. Оба ЦАП могут работать независимо или совместно. При независимой работе ЦАП (бит `Cfg_SYNC_A=0`) после записи данных в регистр данных `DACx_DATA` на выходе `DACx_OUT` формируется уровень напряжения, соответствующий записанному значению. При синхронной работе (бит `Cfg_SYNC_A=1`) данные обоих ЦАП могут быть обновлены одной записью в один из регистров `DACx_DATA`. ЦАП может работать от внутренней опоры `Cfg_M_REFx=0`, тогда ЦАП формирует выходной сигнал в диапазоне от 0 до напряжения питания `AUCC`. В режиме работы с внешней опорой `Cfg_M_REFx=1` ЦАП формирует выходное напряжение в диапазоне от 0 до значения `DACx_REF`.

Для работы с аналоговыми портами ввода/вывода используется библиотека `MDR32F9Qx_dac.h`. Для работы ЦАП достаточно сконфигурировать соответствующий вывод на работу в аналоговом режиме и использовать функцию `DAC1_Init` или `DAC2_Init` для указания источника опорного напряжения.

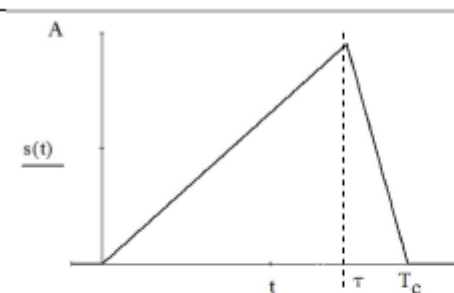
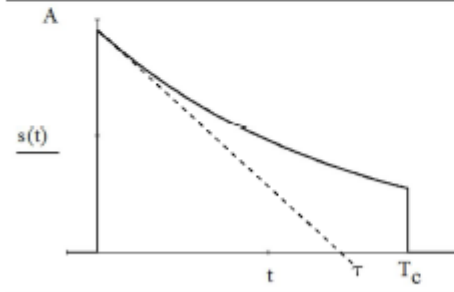
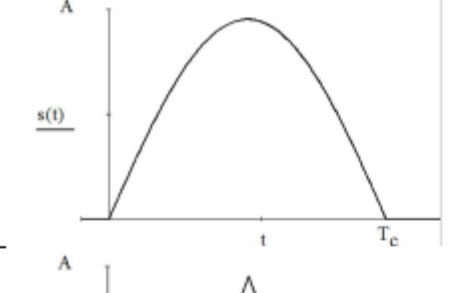
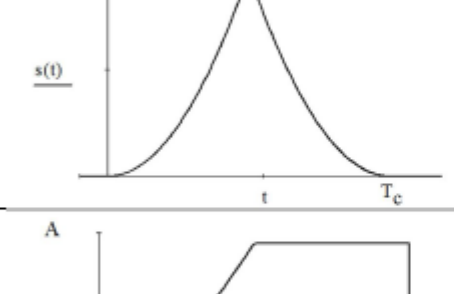
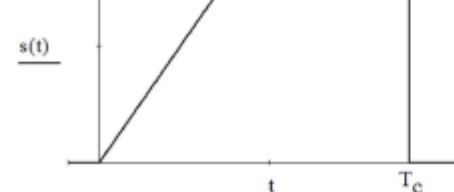
Описание регистров блока контроллера АЦП приведено в таблице 307 стр.326 спецификации микроконтроллеров серии 1986BE9x.

Задание

Реализовать устройство генерации заданного в соответствии с вариантом индивидуального задания аналогового сигнала с использованием встроенного в микроконтроллер цифро-аналогового преобразователя (ЦАП).

Устройство должно генерировать аналоговый сигнал на разьеме X14 «DAC_OUT» демонстрационно-отладочной платы плата 1986EvBrd.

Варианты индивидуальнй заданий

| № | Вид сигнала | Функция |
|----|---|---|
| 1. |  | $s(t) = \begin{cases} \frac{A}{\tau}t & \text{при } t \in [0, \tau] \\ A \frac{t-T_c}{\tau-T_c} & \text{при } t \in [\tau, T_c] \\ 0 & \text{при других } t \end{cases}$ |
| 2. |  | $s(t) = \begin{cases} A \cdot e^{-\frac{t}{\tau}} & \text{при } t \in [0, T_c] \\ 0 & \text{при других } t \end{cases}$ |
| 3. |  | $s(t) = \begin{cases} A \cdot \sin\left(\frac{\pi}{T_c}t\right) & \text{при } t \in [0, T_c] \\ 0 & \text{при других } t \end{cases}$ |
| 4. |  | $s(t) = \begin{cases} \frac{4A}{T_c^2}t^2 & \text{при } t \in \left[0, \frac{T_c}{2}\right] \\ \frac{4A}{T_c^2}(t-T_c)^2 & \text{при } t \in \left[\frac{T_c}{2}, T_c\right] \\ 0 & \text{при других } t \end{cases}$ |
| 5. |  | $s(t) = \begin{cases} \frac{2A}{T_c}t & \text{при } t \in \left[0, \frac{T_c}{2}\right] \\ A & \text{при } t \in \left[\frac{T_c}{2}, T_c\right] \\ 0 & \text{при других } t \end{cases}$ |

| | | |
|-----|--|--|
| 6. | | $s(t) = \begin{cases} \frac{A}{2} & \text{при } t \in \left[0, \frac{T_c}{2}\right] \\ A & \text{при } t \in \left[\frac{T_c}{2}, T_c\right] \\ 0 & \text{при других } t \end{cases}$ |
| 7. | | $s(t) = \begin{cases} \frac{3A}{T_c} & \text{при } t \in \left[0, \frac{T_c}{3}\right] \\ A & \text{при } t \in \left[\frac{T_c}{3}, \frac{2T_c}{3}\right] \\ 3A\left(1 - \frac{t}{T_c}\right) & \text{при } t \in \left[\frac{2T_c}{3}, T_c\right] \\ 0 & \text{при других } t \end{cases}$ |
| 8. | | $s(t) = \begin{cases} A \cdot \cos\left(\frac{\pi}{2T_c} t\right) & \text{при } t \in [0, T_c] \\ 0 & \text{при других } t \end{cases}$ |
| 9. | | $s(t) = \begin{cases} A \left[1 - \cos\left(\frac{\pi}{T_c} t\right)\right] & \text{при } t \in \left[0, \frac{T_c}{2}\right] \\ A \cos\left[\pi\left(\frac{t}{T_c} - \frac{1}{2}\right)\right] & \text{при } t \in \left[\frac{T_c}{2}, T_c\right] \\ 0 & \text{при других } t \end{cases}$ |
| 10. | | $s(t) = \begin{cases} \frac{2A}{T_c} t & \text{при } t \in \left[0, \frac{T_c}{2}\right] \\ 0 & \text{при других } t \end{cases}$ |

Значение параметров амплитуды A и длительности T_c и τ подбирается студентом самостоятельно.

Содержание отчета

1. Титульный лист, содержащий название работы, номер варианта индивидуального задания, ФИО, группу студента.
2. Цель работы

3. Краткие теоретические сведения.
4. Электрическая принципиальная схема устройства.
5. Код программы
6. Выводы.

ЛАБОРАТОРНАЯ РАБОТА № 14. Вывод графической информации на ЖК-дисплей

Цель работы:

Научиться использовать устройство ЖК дисплея совместно с микроконтроллером для вывода символьной и графической информации.

Задание

Реализовать устройство вывода 3-х информационных сообщений на ЖК дисплей – фамилия, имя, отчество.

Выбор информационного сообщение осуществляется с помощью кнопок KEY1, KEY2, KEY3.

Содержание отчета

1. Титульный лист, содержащий название работы, номер варианта индивидуального задания, ФИО, группу студента.
2. Цель работы
3. Краткие теоретические сведения.
4. Электрическая принципиальная схема устройства.
5. Код программы
6. Выводы.

Литература

1. Спецификация на серию 1986VE9x + errata версия 3.9.0
2. Мельников А.А. (ст.), Мельников А.А. (мл.) Микропроцессоры, микроконтроллеры и однокристальные микропрограммируемые устройства // Спутник+, Москва, 2010, 269с.
3. Есаулов С.М. Микроконтроллеры в учебном процессе // Новые компьютерные технологии. 2006. Т. 4. № 1 (4). С. 21-2.2
4. Ермак М. Российские микроконтроллеры с ядром Cortex-M3 и пример реализации проекта // Компоненты и Технологии. 2010. №110 С.74-77.
5. Комиссаров В. Микроконтроллеры компании "Миландр" – эффективное средство программирования ПЛИС // Электроника: Наука, технология, бизнес. 2013. № 2 (124). С. 70-75.
6. Жмакин А. П., Селиванов Д. И. О разработке программных моделей микроконтроллеров // Ученые записки. Электронный научный журнал Курского государственного университета. 2012. №4 (24) С.106-112.
7. Васильев А.Е., Шилов М.М., Мурго А.И. Научно-методические аспекты преподавания дисциплин цикла «Встраиваемые микроконтроллеры» // Информационно-управляющие системы. 2011. №6 С.68-77.
8. Благодаров А.В. Программирование микроконтроллеров семейства 1986VE9X компании Миландр. – Горячая линия – Телеком, 2016. – 230 с. ISBN 978-5-9912-0584-9.
9. Васильев А.С., Лашманов О.Ю., Пантюшин А.В. Основы программирования микроконтроллеров. – СПб: Университет ИТМО, 2016. – 95с.
10. Огородников, И.Н. Микропроцессорная техника: введение в Cortex-M3: учеб. пособие – Екатеринбург: Изд-во Урал. ун-та, 2015. – 116 с.
11. Алалуев Р. В. Основы программирования 32-разрядных микроконтроллеров 1986VE91T компании «Миландр»: руководство к

выполнению лабораторных работ / Р. В. Алалуев, В.М. Глаголев, А.А. Мосур, Л. Л. Владимиров. – М., 2015. – 53 с.

12. Недяк С.П., Шаропин Ю.Б. Лабораторный практикум по микроконтроллерам семейства Cortex-M. Методическое пособие по проведению работ на отладочных платах фирмы Миландр. - Томск: ТУСУР, 2013. - 80 с.

13. Комплект отладочный ДЛЯ M/CX 1986VE91T(94T) паспорт ТСКЯ.468998.014ПС

14. Демонстрационно-отладочная плата 1986VE91_EvBrd.
Техническое описание.