

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет
им. Г.И. Носова»
Многопрофильный колледж



УТВЕРЖДАЮ
Директор
И.С.А. Махновский
08.02.2023г

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ
ДЛЯ ЛАБОРАТОРНЫХ И ПРАКТИЧЕСКИХ ЗАНЯТИЙ
МЕЖДИСЦИПЛИНАРНОГО КУРСА
МДК.02.01 Разработка модулей программного обеспечения для компьютерных систем
для обучающихся специальности
09.02.01 Компьютерные системы и комплексы**

Магнитогорск, 2023

ОДОБРЕНО

Предметно-цикловой комиссией «Информатика
и вычислительная техника»

Председатель Т.Б. Ремез

Протокол № 6 от «25» января 2023 г.

Методической комиссией МпК

Протокол № 4 от «8» февраля 2023 г.

Разработчик:

преподаватель ФГБОУ ВО «МГТУ им. Г.И. Носова» Многопрофильный колледж Татьяна
Борисовна Ремез

Методические указания по выполнению практических и лабораторных работ разработаны на основе рабочей программы профессионального модуля 02 «Проектирование управляющих программ компьютерных систем и комплексов» МДК.02.01 «Разработка модулей программного обеспечения для компьютерных систем».

Содержание практических и лабораторных работ ориентировано на подготовку обучающихся к освоению вида деятельности ВД 2 «Проектирование управляющих программ компьютерных систем и комплексов» программы подготовки специалистов среднего звена по специальности 09.02.01. Компьютерные системы и комплексы и овладению профессиональными компетенциями.

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ	4
2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ	6
Лабораторное занятие №1. Разработка модульной структуры проекта, перечня артефактов и протоколов проекта	6
Лабораторное занятие №2. Настройка работы системы контроля версий (типов импортируемых файлов, путей, фильтров и др. параметров импорта в репозиторий)	16
Лабораторное занятие №3. Разработка и интеграция модулей проекта (командная работа)	25
Лабораторное занятие №4. Интеграция модулей проекта (командная работа)	27
Лабораторное занятие №5. Работа в среде программирования и отладки Keil-C	29
Лабораторное занятие №6. Организация ввода-вывода информации через параллельные порты МК ADuC842	41
Лабораторное занятие №7. Разработка программы управления клавиатурой матричного типа	45
Лабораторное занятие №8. Разработка программы управления символьным ЖКИ	46
Лабораторное занятие №9. Организация ввода-вывода информации через параллельные порты МК Atmega 8535	50
Лабораторное занятие №10. Исследование работы регистра состояний SREG МК Atmega 8535	54
Лабораторное занятие №11. Разработка программы для организации программной задержки (с использованием стека)	56
Лабораторное занятие №12. Организация работы 8-ми разрядного таймера в режиме ШИМ	61
Лабораторное занятие №13. Организация работы 8-ми разрядного таймера в режиме создания временных интервалов	65
Лабораторное занятие №14. Организация работы АЦП МК Atmega 8535	69
Лабораторное занятие №15. Разработка программы управления сегментным индикатором	73
Практическое занятие №1. Изучение схемы типовой МПС	80
Практическое занятие №2. Изучение устройства параллельных портов МК ADuC842	83
Практическое занятие №3. Изучение схемы подключения матричной клавиатуры к МК ADuC842	88
Практическое занятие №4. Изучение схемы подключения ЖКИ к МК ADuC842	90
Практическое занятие №5. Изучение ассемблера и системы команд МК AVR	94
Практическое занятие №6. Изучение работы среды программирования AVRStudio	103
Практическое занятие №7. Изучение устройства параллельных портов МК Atmega8535	107
Практическое занятие №8. Изучение работы регистра состояний SREG МК Atmega 8535	110
Практическое занятие №9. Изучение работы стека МК Atmega 8535	113
Практическое занятие №10. Изучение работы таймеров в различных режимах МК Atmega 8535	115
Практическое занятие №11. Изучение работы АЦП МК Atmega 8535	123
Практическое занятие №12. Изучение работы сегментного и ЖК индикаторов под управлением МК Atmega 8535	127

1 ВВЕДЕНИЕ

Важную часть теоретической и профессиональной практической подготовки студентов составляют практические и лабораторные занятия.

Состав и содержание практических и лабораторных работ направлены на реализацию действующего федерального государственного образовательного стандарта среднего профессионального образования.

Ведущей дидактической целью практических занятий является формирование практических умений - профессиональных (умений выполнять определенные действия, операции, необходимые в последующем в профессиональной деятельности), необходимых в последующей учебной деятельности по профессиональным модулям.

Ведущей дидактической целью лабораторных работ является экспериментальное подтверждение и проверка существенных теоретических положений (законов, зависимостей). В соответствии с рабочей программой ПМ.02. «Проектирование управляющих программ компьютерных систем и комплексов», МДК.02.01 «Разработка модулей программного обеспечения для компьютерных систем» предусмотрено проведение практических и лабораторных занятий.

В результате их выполнения, обучающийся должен:

уметь:

- применять стандартные алгоритмы в соответствующих областях;
- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- применять нормативные документы, определяющие требования к оформлению программного кода;
- организовывать заданную интеграцию модулей в программные средства на базе имеющейся архитектуры и автоматизации бизнес-процессов;
- выполнять тестирование интеграции
- использовать выбранную систему контроля версий;
- использовать приемы работы в системах контроля версий;
- выполнять ручное и автоматизированное тестирование программного модуля;

Содержание практических и лабораторных занятий ориентировано на подготовку обучающихся к освоению профессионального модуля программы подготовки специалистов среднего звена по специальности и овладению **профессиональными компетенциями:**

ПК 2.1. Проектировать, разрабатывать и отлаживать программный код модулей управляющих программ

ПК 2.2. Владеть методами командной разработки программных продуктов

ПК 2.3. Выполнять интеграцию модулей в управляющую программу

ПК 2.5. Выполнять установку и обновление версий управляющих программ (с учетом миграции - при необходимости)

А также формированию общих компетенций:

ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам;

ОК 04. Эффективно взаимодействовать и работать в коллективе и команде.

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста;

ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, применять знания об изменении климата, принципы бережливого производства, эффективно действовать в чрезвычайных ситуациях;

ОК 08. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности;

ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках.

Выполнение обучающимися практических и лабораторных работ по ПМ.02. «Проектирование управляющих программ компьютерных систем и комплексов», МДК.02.01 «Разработка модулей программного обеспечения для компьютерных систем» направлено на:

- обобщение, систематизацию, углубление, закрепление, развитие и детализацию полученных теоретических знаний по конкретным темам учебной дисциплины;

- применять полученные знания на практике, реализацию единства интеллектуальной и практической деятельности;

- развитие интеллектуальных умений у будущих специалистов: аналитических, проектировочных, конструктивных и др.;

- выработку при решении поставленных задач профессионально значимых качеств, таких как самостоятельность, ответственность, точность, творческая инициатива.

Практические и лабораторные занятия проводятся после соответствующей темы, которая обеспечивает наличие знаний, необходимых для ее выполнения.

2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Тема 1.2. Проектирование программного Обеспечения

Лабораторное занятие № 1

Разработка модульной структуры проекта, перечня артефактов и протоколов проекта

Цель: Создание структуры проекта и заполнение базовой информации о проекте. Изучение процесса создания модульной структуры программного обеспечения, осуществляемого с помощью структурных карт Константайна. Получение первичных навыков планирования работ по разработке и внедрению автоматизированных информационных систем, разработка протоколов проекта.

Выполнив задания, Вы будете:

уметь:

- применять стандартные алгоритмы в соответствующих областях;
- применять нормативные документы, определяющие требования к оформлению программного кода;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации- производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;
- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных.
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

Персональный компьютер, пакет Microsoft Office, доступ к Internet ресурсам, программа MS Project, среда программирования Visual Studio 2019.

Задание:

Издать структуру проекта и заполнить базовую информацию о проекте. Разработать модульную структуру программного обеспечения с помощью структурных карт Константайна.

Порядок выполнения работы:

1. Создать новый проект любым способом, заполнить сведения о проекте, изменить базовый календарь проекта, включить в проект дополнительную документацию.
2. В соответствии с требованиями, предъявляемыми техническим заданием, и результатами внешнего проектирования разработать модульную структуру подсистемы обслуживания клиента по его кредитной карте в банкомате.
3. Сформировать календарный план выполнения работ по проекту на основе технического задания на разработку и внедрение автоматизированной информационной системы:

Краткие теоретические сведения:

1. Работа в программе MS Project

Новый проект в программе MS Project может быть создан как с нуля, так и используя один из предлагаемых стандартных шаблонов. Шаблон представляет собой особый тип файла проекта, содержащий набор информации, призванной упростить работу над проектом. В состав шаблона обычно входит список заранее организованных и размещенных определенным образом задач, а также информация о ресурсах, пользовательские представления, календари, отчеты, макросы и т.д. Любая информация, предлагаемая шаблоном, может быть изменена в соответствии с требованиями конкретного проекта. В качестве шаблона также может быть использован созданный ранее проект. При создании проекта из шаблона необходимо выбрать

на панели *Консультанта* ссылку *Общие шаблоны*. Далее на вкладке *Шаблоны проектов* выбирается необходимый шаблон.

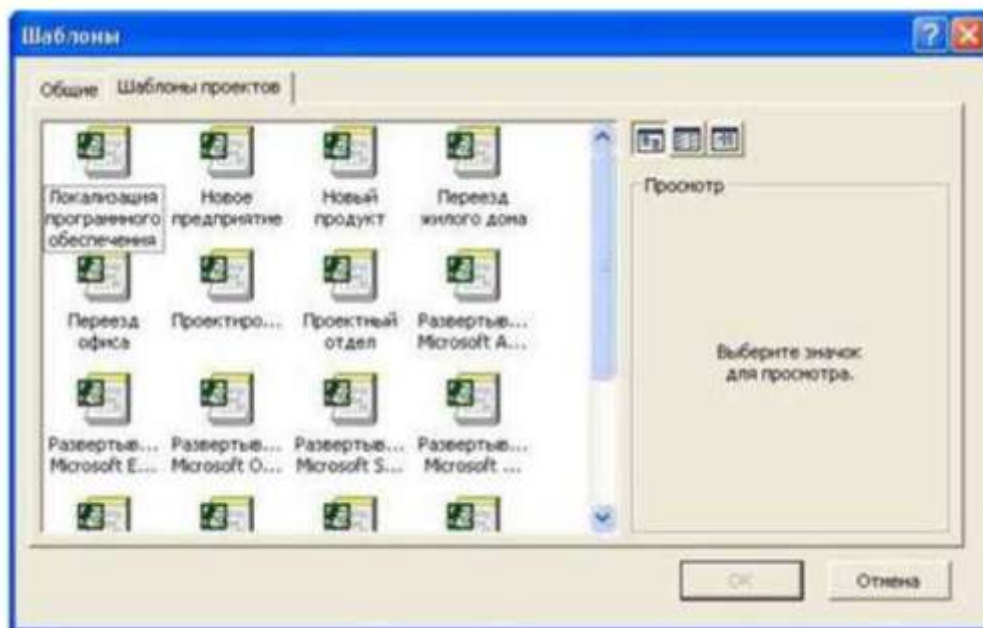


Рис.1. Выбор шаблона проекта

Рабочее пространство программы называется видом или представлением. По умолчанию после создания проекта активен вид *Диаграмма Ганта* (рис.2). Данная диаграмма служит для отображения последовательности задач проекта как в текстовом так и в графическом виде.

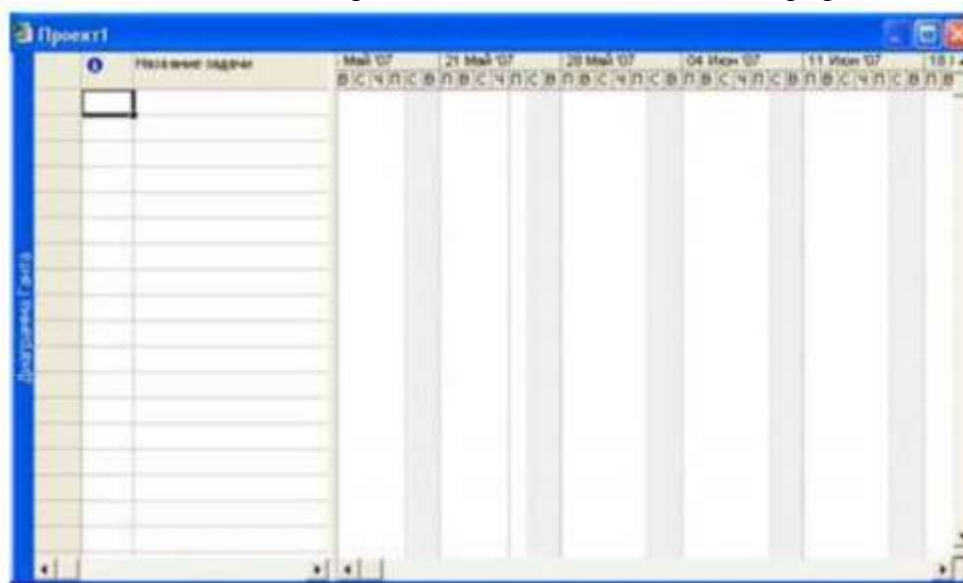


Рис.2. Окно диаграммы Ганта

После создания проекта необходимо настроить его основные параметры. Для этого удобно использовать мастер *Новый проект*. Для этого нажимаем кнопку *Задачи* на панели *Консультанта* и выбираем ссылку *Определение проекта*. Ответив на вопросы о дате начала проекта и совместной работе над проектом и сохранив результат, выбираем ссылку *Определение рабочего времени проекта* для запуска мастера *Рабочее время проекта*. Таким образом мы можем настроить календарь проекта. Следующим решением, которое необходимо принять на стадии создания, является выбор исходной даты проекта. План проекта может быть составлен от даты начала или завершения проекта. Для настройки планирования от начальной даты выберите в меню *Проект* пункт *сведения о проекте*. В появившемся окне (рис.3) выбираем планирование *От даты начала проекта* и ставим *Дату начала*. Да окончания будет рассчитана далее автоматически. В случае планирования от конечной даты выбираем *От даты окончания проекта* и ставим *Дату окончания*. В этом случае автоматически будет рассчитываться дата начала.

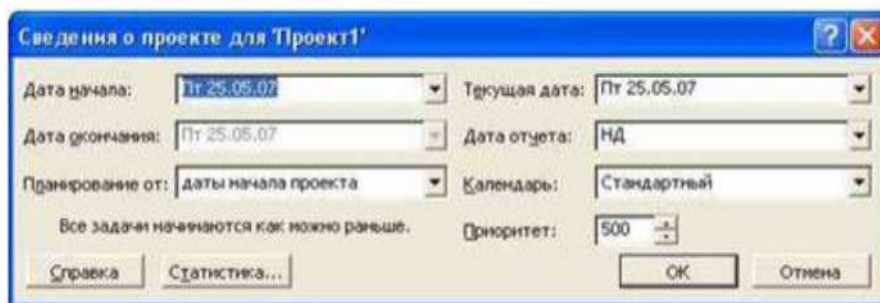


Рис.3. Настройка сведений о проекте

Также в этом окне мы можем выбрать календарь для проекта. В состав пакета MS Project входит три базовых календаря - стандартный, ночная смена и 24 часа. В *стандартном календаре* рабочий день начинается с 8:00 и заканчивается в 17:00

с обеденным перерывом с 12:00 до 13:00. Рабочая неделя начинается с понедельника и заканчивается в пятницу. Это календарь, применяемый по умолчанию. В *календаре ночной смены* рабочий день начинается с 23:00 и заканчивается в 8:00 с часовым перерывом с 03:00 до 04:00. В *календаре «24 часа»* рабочее время продолжается круглые сутки без выходных и обеденных перерывов. Базовые календари можно редактировать для этого в меню *Сервис* необходимо выбрать пункт *Изменение рабочего времени*. В появившемся окне (рис.4.) выбираем базовое расписание, которое мы хотим отредактировать. Для изменения рабочего времени одного дня необходимо выбрать этот день в календаре. Далее, если необходимо сделать этот день выходным, мы выбираем параметр *нерабочее время*, если же мы хотим только изменить временные рамки рабочего дня, то выбираем параметр *нестандартное рабочее время* и в полях ниже вводим время начала и завершения рабочего дня.

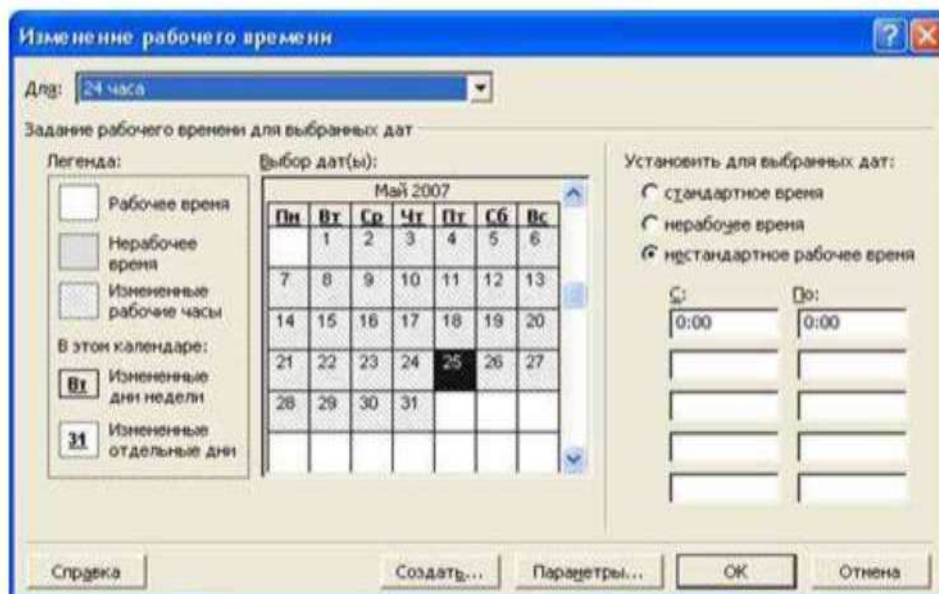


Рис.4. Изменение рабочего времени

Можно также создать новое базовое расписание. Для этого в окне *Изменение рабочего времени* нажимаем кнопку *Создать*. В появившемся окне (рис.5) выбираем создание нового календаря на основе стандартного или создание копии любого другого календаря. Значения рабочего времени для вновь созданного календаря могут также быть отредактированы через окно *Изменение рабочего времени*.

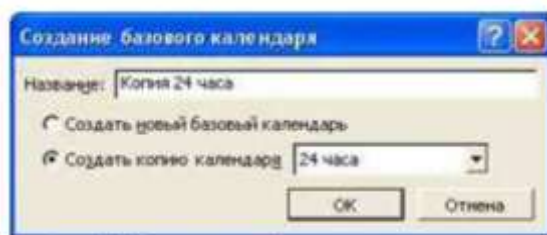


Рис.5. Создание базового календаря



Рис.6. Сведения о задаче

Создаваемый проект может быть использован в качестве хранилища проектной документации, например, обзора проекта, результатов проведенных анализов или спецификации создаваемого продукта. Для присоединения такой документации целесообразно использовать т.н. суммарную задачу проекта, содержащую итоговую информацию о датах и стоимости проекта. Для отображения суммарной задачи на диаграмме Ганта необходимо в меню *Сервис* выбрать пункт *Параметры* и перейти на вкладку *Вид*. На данной вкладке необходимо выбрать параметр *Показать суммарную задачу проекта* под заголовком *Параметры структуры для проекта*. Суммарная задача появится в нулевом ряду диаграммы Ганта. Проектная документация может как включаться в файл проекта, так и быть доступной через гиперссылки. Для включения документов в файл проекта необходимо выбрать суммарную задачу проекта и нажать кнопку *Сведения о задаче*, расположенную на стандартной панели задач. В открывшемся окне (рис.6) выбираем вкладку *Заметки*. На вкладке нажимаем кнопку *Вставить объект*. В открывшемся окне необходимо выбрать опцию *Создать из файла*. После этого указываем путь к файлу документа, который предполагается включить в проект.

После закрытия окна сведений о суммарной задаче в диаграмме Ганта появится индикатор примечаний. Для создания гиперссылки к документу необходимо нажать кнопку *Гиперссылка* на панели задач. В поле *Текст* открывшегося диалогового окна *Добавление гиперссылки* (рис.7) введите название связываемого документа, затем выберите документ в списке. В поле индикаторов диаграммы Ганта появится индикатор гиперссылок.

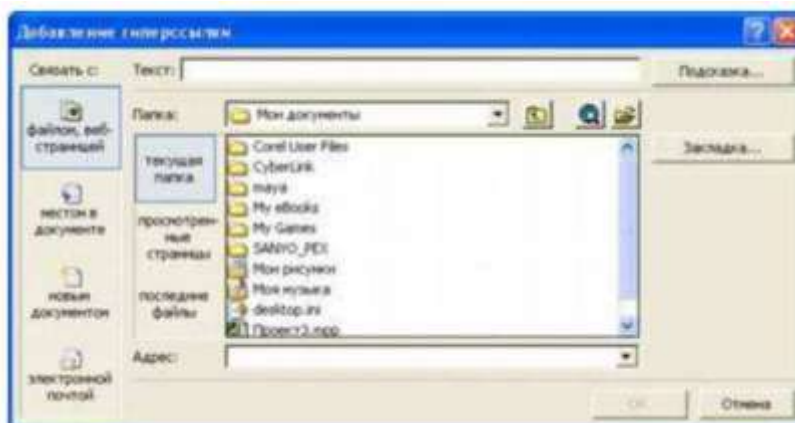


Рис.7. Добавление гиперссылки

2. Разработка модульной структуры проекта

Обмен данными между программными модулями осуществляется через общую область памяти, в которую модуль управления устройством считывания помещает данные о пароле (Parol), атрибуты клиента (Client Attributes) и лимит денег на счету (Limit of money). Модуль аутентификации получает из этой общей области памяти сведения о пароле и возвращает в головной модуль управляющий параметр Autentification flag, содержащий результат аутентификации. Модуль получения и обработки запроса на обслуживание для своей работы получает из общей области памяти атрибуты клиента и лимит денег на счету.

Чтобы добиться декомпозиции на модули максимальной прочности и минимального сцепления, необходимо спроектировать модульную структуру в виде дерева, в том числе и со сросшимися ветвями. В узлах такого дерева размещаются программные модули, а направленные дуги (стрелки) показывают статическую подчинённость модулей, т.е. каждая дуга показывает, что в тексте модуля, из которого она исходит, имеется ссылка на модуль, в который она входит.

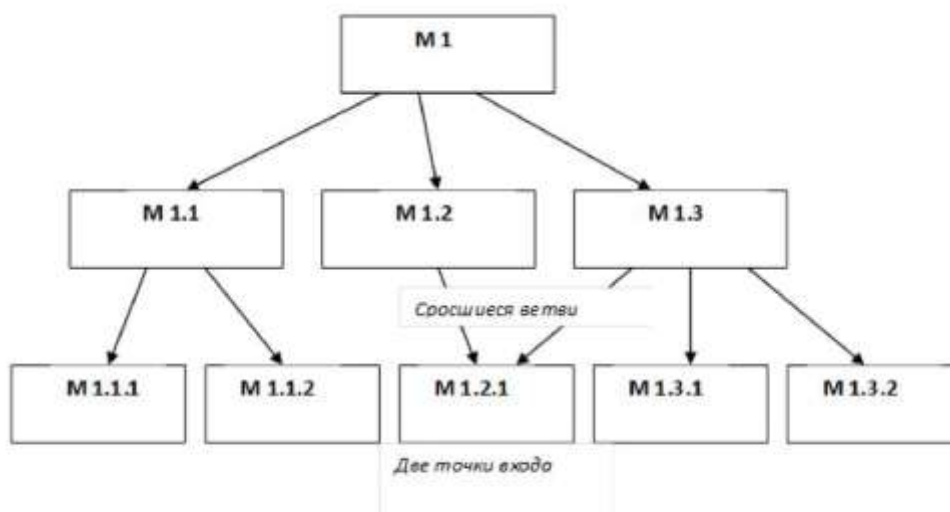


Рис. 1. Пример неархического дерева модулей

При этом модульная структура программы должна, помимо картинки, включать спецификацию программного модуля.

1. спецификация программного модуля должна содержать:
2. синтаксическую спецификацию его входов (имя модуля, типы передаваемых ему параметров, типы возвращаемых результатов, синтаксис обращения к любому его входов)
3. функциональную спецификацию (описание семантики функций, выполняемых модулем по каждому из его входов).

В процессе разработки модульная структура может по-разному использоваться для определения порядка программирования модулей — восходящая и нисходящая разработка.

В восходящей разработке модули программируются, начиная с нижних уровней, и сразу тестируются, исходя из функциональных спецификаций. Такой порядок представляется естественным, т.к. каждый новый модуль выражается через уже запрограммированные и проверенные модули. Однако современная технология не рекомендует этот прием, т.к. при этом трудно обеспечить концептуальную целостность ПС.

Концептуальная целостность предполагает общие принципы реализации, предположения, структуры данных, - а они могут быть ещё не ясны в начальных стадиях разработки.

Перепрограммирование же модулей нижних уровней связано с большими затратами, т.к. требует не только повторной разработки текстов, но и повторного тестирования.

Предпочтительной является нисходящая разработка. В этой технологии программирование начинается с модуля с самого верхнего уровня. При этом для тестирования модули нижних уровней заменяются простыми по конструкции имитаторами, которые либо моделируют работу нижних уровней (например, реализуют таблицы; вход-отклик), либо просто сообщают о своём вызове и завершаются признаком успеха. Такая реализация обеспечивает

большую концептуальную целостность и меньший объем разрабатываемых тестов, каждый модуль здесь тестируется при т.н. «естественном» состоянии информационной среды, т.к. он вызывается реальным (оттестированным) модулем верхнего уровня.

3. Формирование календарного плана выполнения работ по проекту

Для проведения успешного проекта нужно оценить объем предстоящих работ, возможный риск, требуемые ресурсы, предстоящие задачи, определить контрольные точки, стоимость и план работ, которому желательно следовать. Процесс руководство программным проектом включает решение вышеперечисленных задач. Этот процесс начинается перед технической работой, продолжается по мере развития ПО от идеи к реальности и достигает наибольшей интенсивности к концу работ. Основной задачей при планировании является определение структуры распределения работ WBS (Work Breakdown Structure) с помощью средств планирования работ (например, MS Project). План выполнения работ составляется на основе декомпозиции проекта вплоть до постановки элементарных задач, которые могут быть выяснены по результатам предварительного анализа. При этом возможно применение содержательных моделей системного анализа. Например, использование модели декомпозиции типа «жизненный цикл» позволит разбивать отдельные задачи на подзадачи путем определения последовательности действий.

Процесс декомпозиции будет определяться принятой моделью жизненного цикла разработки программного обеспечения.



Рис. 1. Декомпозиция задач, которые необходимо решить в процессе выполнения проекта по разработке программного обеспечения

Для каждой элементарной задачи должны быть определены:

1. ресурсы, необходимые для решения задачи (в том числе трудовые);
2. объем работ, выраженный в принятой системе метрик;
3. стоимость работ (может быть вычислена на основе объема работ и стоимости привлекаемых ресурсов);

привлекаемых ресурсов);

4. длительность работ (может быть вычислена на основе объема работ, количества привлекаемых трудовых ресурсов и принятых нормативов производительности).

Между отдельными элементарными задачами могут быть определенные зависимости, заключающиеся в том, что одни задачи могут выполняться параллельно, другие - в строгой последовательности (для выполнения одних задач могут требоваться результаты выполнения других).

После определения зависимостей можно приступать к распределению элементарных задач по времени. При этом особое внимание следует остановить на задачах, выполняемых параллельно. Параллельность действий повышает требования к планированию.

Необходимо четко отследить наличие ресурсов, необходимых для выполнения каждой задачи. Если план предусматривает использование ресурса 1 для выполнения задач А и Б, то эти задачи не могут выполняться параллельно, даже если между ними нет концептуальной зависимости. Кроме того, руководитель проекта должен знать задачи, лежащие на критическом

пути. Для того чтобы весь проект был выполнен в срок, необходимо выполнять в срок все критические задачи.

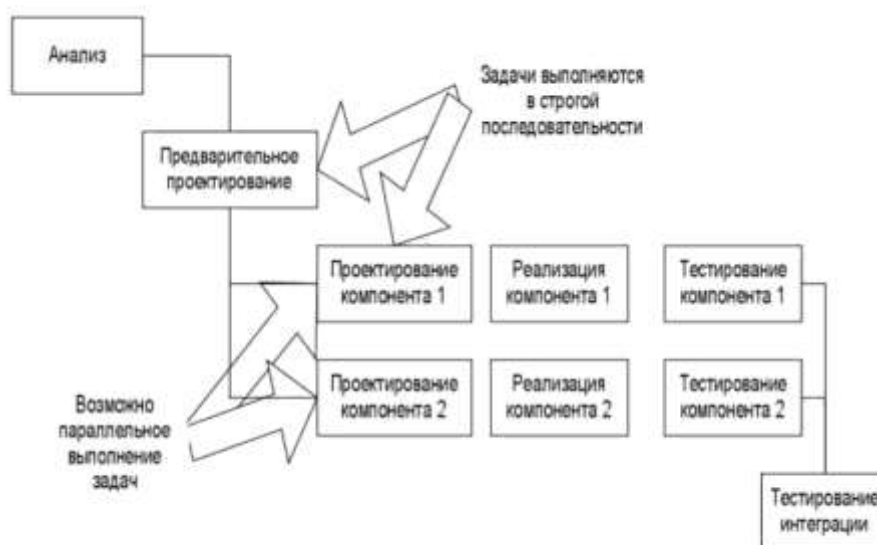


Рис. 2. Параллельное и последовательное выполнение задач

Календарный план помимо распределения задач и ресурсов по времени должен предусматривать процедуры контроля промежуточных результатов. Такие процедуры обычно называют вехами. Очень важно, чтобы вехи были расставлены через регулярные интервалы вдоль всего процесса разработки программного обеспечения. Кроме того, желательно чтобы вехи совпадали со сроком выполнения критических задач. Это даст руководителю возможность не только регулярно получать информацию о текущем положении дел, но и объективно оценивать риски срыва сроков выполнения проекта, принимать оперативные решения по снижению этих рисков. В первую очередь определите основные фазы выполнения проекта.

В основу может быть положена принятая модель жизненного цикла процесса разработки программного обеспечения. Например, при использовании каскадной модели основными фазами будут являться анализ, проектирование, реализация, тестирование, внедрение. Далее определите состав работ внутри каждой фазы, в соответствии с сутью разработки.

Таким образом будет определен состав работ по проекту. Каждая фаза должна заканчиваться вехой - специальной единицей работы, подразумевающей контроль выполнения работ по проекту и достижение некоторого промежуточного или окончательного результата. Далее определите длительность каждой работы, входящей в план работ.

Для определения длительности могут быть использованы различные регрессионные модели (например, СОСОМО II), или же может применяться прямой метод оценки. Следующим этапом является определение связей между задачами. В большинстве средств планирования (например, в MS Project), существует четыре вида связей: Связь типа окончание - начало означает, что задача Б не может начаться раньше окончания задачи А (например, если в процессе выполнения задачи Б используются результаты, получаемые при решении задачи А).

Связь типа начало - начало означает, что задача Б не может начаться до тех пор, пока не началось выполнение задачи А. Например, тестирование программного компонента не может начинаться до того, как была начата его разработка, но, в то же время, для написания тестов не обязательно дожидаться окончания разработки этого компонента. Связь типа окончание - окончание означает, что работа Б не может окончиться до тех пор, пока не завершится выполнение работы А. Например, проектирование базы данных не может быть закончено до того, как будет завершено семантическое моделирование предметной области.

Связь окончание - начало означает, что задача Б не может закончиться до того, как началась задача А. Обычно такая связь используется в том случае, когда А является задачей с фиксированной датой начала, которую нельзя изменить. После того, как определен состав работ, нужно определить, кто эти задачи будет выполнять и какое оборудование должно использоваться.

Сформируйте список ресурсов, для каждого ресурса определите название и стоимость его использования. Далее назначьте ресурсы на выполнение конкретных задач. При первом

назначении ресурса будут автоматически рассчитаны трудозатраты. В тех случаях, когда необходимо ускорить выполнение тех или иных задач, на них могут быть назначены дополнительные ресурсы. После распределения ресурсов необходимо выполнить выравнивание их нагрузки. В тех случаях, когда на параллельно выполняемые задачи назначается один и тот же ресурс, нагрузка на него может превысить максимально допустимую. Для выравнивания нагрузки установите дополнительные связи между задачами таким образом, чтобы задачи, использующие один и тот же ресурс, выполнялись последовательно.

Ход работы:

Задание 1. Создать новый проект любым способом, заполнить сведения о проекте, изменить базовый календарь проекта, включить в проект дополнительную документацию.

1. С помощью меню кнопки Пуск вызвать приложение Microsoft Project на рабочий стол.
2. Вызвать на поле рабочего окна приложения ранее подготовленный проект, используя библиотеку шаблонов: в области задач Создание проекта перейти на поле группы Создание с помощью шаблона и выбрать гиперссылку Общие шаблоны. На вкладке Шаблоны открыть лист Шаблоны проектов, выбрать шаблон Новый продукт и щелкнуть по кнопке ОК.
3. Сохранить шаблон проекта под новым именем: открыть меню Файл, активизировать команду Сохранить как и перейти на поле диалогового окна Сохранение документа; в диалоговом окне перейти на рабочую папку, записать в поле имени файла новое имя проекта, например FIO_Project, и щелкнуть по кнопке Сохранить.
4. Закрыть пустой проект, т.е. перейти на поле проекта Проект 1, активизировать команду Закрыть в меню Файл.
5. На поле текущего проекта убрать Область задач, для чего следует щелкнуть по кнопке Закрыть, размещенной в правом верхнем углу области.
6. Разместить на рабочем поле различные представления о состоянии проекта: вызвать Панель представлений, для чего в меню Вид щелкнуть по соответствующей команде. Панель представлений позволяет вызвать различные формы представления информации о проекте с помощью соответствующих кнопок; настроить комбинированное представление, используя команду Разделить в меню Окно и соответствующие кнопки панели инструментов, например Диаграмма Ганта и График ресурсов, Использование задач и Использование ресурсов, Диаграмма Ганта и Использование задач. Найти необходимую информацию об использовании ресурсов на Графике ресурсов, данные об их использовании (временной загрузке).
7. Настройка таблицы диаграммы Ганта: снять разделение на рабочей области с помощью команды Разделить из меню Окно и вызвать представление Диаграмма Ганта. Для вызова соответствующего столбца используется меню команды Таблица. Далее следует ознакомиться с основными опциями этого меню; перейти на поле меню Вид и раскрыть меню опций с помощью команды Таблица.

В исходном положении выбрана опция Ввод, которая устанавливает рядом с диаграммой первые два столбца таблицы: Наименование задачи (постоянный столбец) и столбец Длительность задачи; выбрать опцию Гиперссылка — рядом со столбцом задач появится столбец Гиперссылка. В ячейках этого столбца можно записать вспомогательные сведения о задачах путем составления записок, вложения файлов или формирования гиперссылок на сопутствующую информацию, находящуюся в файле проекта или в других местах. Это позволяет подготовить библиотеки документов и связать их с проектами и задачами.

После этого руководители проекта и другие заинтересованные стороны смогут просматривать сопровождающие документы в своих веб-обозревателях; выбрать опцию Затраты. В этом случае появляются три столбца затрат: Фиксированные затраты, Начисления фактических затрат и Общие затраты. перейти на опцию Использование. На рабочем поле таблицы появятся два столбца: Трудозатраты и Длительность; просмотреть опцию Календарный план. Она вызывает четыре столбца: Начало, Окончание, Позднее начало, Позднее окончание; активировать опцию Отклонение и убедиться, что последние два столбца на поле будут заменены на столбцы Базовое начало и Базовое окончание; вызвать опцию Отслеживание, что позволяет вызвать другую группу из четырех столбцов: Фактическое

начало, Фактическое окончание, % завершения и Физический % завершения; щелкнуть по опции Суммарные данные, что позволит установить такую последовательность столбцов: Длительность, Начало, Окончание и % завершения; использование опции Трудозатраты, чтобы одновременно увидеть следующие данные: Трудозатраты, Базовые, Отклонения, Фактические.

8. Настроить таблицу, добавляя необходимые и удаляя лишние столбцы: добавить новые столбцы в таблицу следует в меню Вставка, выбрать команду Столбец и в поле диалога Определение столбца с помощью раскрывающегося списка Имя поля выбрать новое поле, например, Трудозатраты; удалить установленный столбец с помощью контекстного меню, которое следует вызвать щелчком правой клавиши мыши по полю удаляемого столбца. В контекстном меню следует активизировать команду Скрыть столбец.
9. Выполнить фильтрацию данных диаграммы Ганта: выбрать кнопку Другие представления на панели представлений. На поле диалогового окна в списке Представления выбрать строку Подробная Диаграмма Ганта и щелкнуть по кнопке Применить; раскрыть список Фильтр, размещенный на панели форматирования, и щелкнуть по строке Вехи.
10. Выполнить сортировку задач проекта по длительности: в меню Проект выбрать команду Сортировка, в меню опций которой выбрать Сортировать по; в диалоговом окне Сортировка раскрыть список Сортировать по и выбрать в нем строку Длительность. Установить флажок По возрастанию и щелкнуть по кнопке Сортировать; отодвинув поле диаграммы так, чтобы видеть столбец Длительность, убедиться в правильности выполненной операции.
11. Настроить изображение диаграммы Ганта: для настройки формы и цвета отрезков в меню Формат выбрать команду Стили отрезков. В верхней части вкладки выбрать тип задачи и стиль отрезка, который следует изменить. На нижней части вкладки перейти на лист Отрезки, где выполнить операции по изменению стиля отрезка, его формы, узора и цвета; показать текст, который следует разместить рядом с отрезком (информация, отображаемая соответствующим элементом диаграммы). Для этого на вкладке Стили отрезков раскрыть лист Текст и показать, где (слева, справа, снизу, сверху или внутри отрезка) следует разместить текст; настроить шкалу времени диаграммы. Перевести курсор на поле шкалы времени и вызвать контекстное меню, где выбрать опцию Шкала времени. На соответствующей вкладке выбрать уровень шкалы времени (Верхний, Средний, Нижний) и в раскрывающемся списке Отображать выбрать строку Три уровня. Шкала времени может состоять из трех уровней (например, год, квартал, месяц), но обязателен только Средний уровень; перейти на лист Верхний уровень и установить Год в строке Единицы, показать текстовое обозначение года в раскрывающемся списке Надписи, выбрать необходимый стиль форматирования надписи и др.; перейти на лист Средний уровень и выбрать в качестве единицы измерения Квартал; перейти на лист Нижний уровень и установить в качестве единицы измерения времени Месяцы; проверить полученный результат настройки диаграммы Ганта.
12. Сохранить проект в рабочей папке и закрыть приложение.

Задание 2. В соответствии с требованиями, предъявляемыми техническим заданием, и результатами внешнего проектирования разработать модульную структуру подсистемы обслуживания клиента по его кредитной карте в банкомате.

В составе программного обеспечения можно выделить следующие программные модули: Головной модуль (Main module), Модуль управления устройством считывания кредитной карты (Credit card control module), Модуль аутентификации (Authentication module) и Модуль получения и обработки запроса на обслуживание (Reception and processing module). Кроме этого в состав ПО необходимо включить модуль данных кредитной карты (Credit card data).

Основной функцией Головного модуля является организация общего управления поведением подсистемы и выполняет вызов всех остальных программных модулей.

Модуль управления устройством считывания кредитной карты выполняет функции связанные с обработкой кредитной карты: ввод, считывание хранящейся на ней информации, удаление.

Модуль аутентификации выдает сообщение клиенту на ввод ключевых данных, выполняет получение пароля и проверку его правильности.

Модуль получения и обработки запроса на обслуживание выполняет следующие функции: Получение запроса на обслуживание и проверка возможности его исполнения, Обработка запроса на обслуживание, включающая такие действия как:

- обработка внутренней банковской документации по клиенту;
- распечатка баланса клиента;
- выдача наличных денег и информирование компьютера банка об изъятых из банка деньгах;
- распечатка операции клиента.

На рис. 2 приведена структурная карта, демонстрирующая отношения между указанными модулями системы.

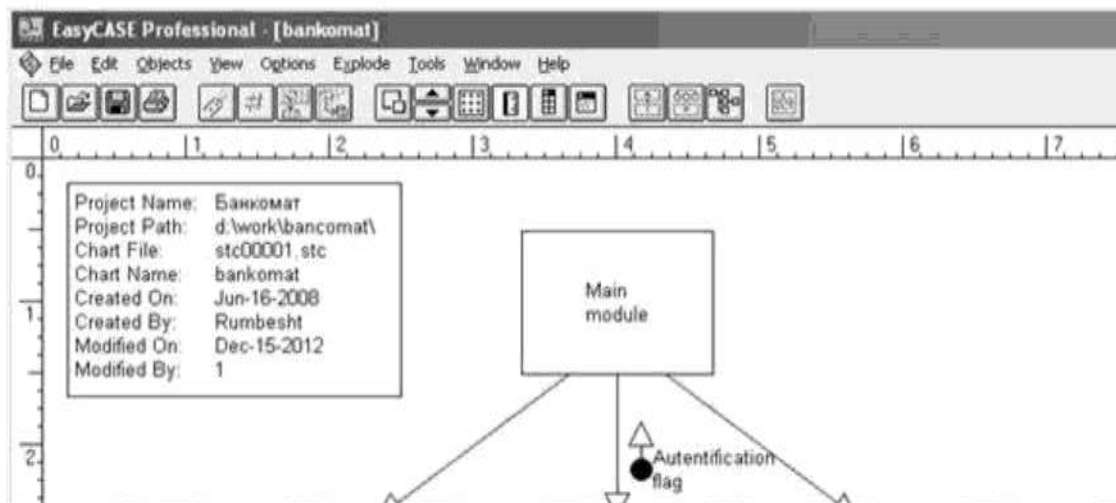


Рис. 2. Модульная структура программного обеспечения

Согласно этой диаграмме головной модуль обращается к модулям управления устройством считывания кредитной карты, аутентификации и получения и обработки запроса на обслуживание. Вызов указанных модулей осуществляется согласно внутренней логики головного модуля, реализующей следующий сценарий: При инициации действий со стороны клиента головной модуль, вызывает модуль управления устройством считывания кредитной карты для ее ввода и считывания с нее информации. После завершения считывания управление возвращается головному модулю, который затем обращается к модулю аутентификации. Модуль аутентификации проверяет подлинность клиента и вместе с результатом этой проверки возвращает управление головному модулю. В зависимости от результатов аутентификации головной модуль либо вызывает модуль управления устройством считывания для удаления кредитной карты, либо обращается к модулю получения и обработки запроса на обслуживание для предоставления требуемого сервиса. Если осуществляется вызов получения и обработки запроса на обслуживание, то после завершения его работы головной модуль обращается к модулю управления устройством считывания для удаления кредитной карты.

Задание 3. Сформировать календарный план выполнения работ по проекту на основе технического задания на разработку и внедрение автоматизированной информационной системы.

1. Ознакомьтесь с техническим заданием.
2. Выберите модель жизненного цикла процесса разработки и внедрения ПО, которая, по вашему мнению, в наибольшей степени соответствует рассматриваемой ситуации.
3. Выделите основные этапы работ.
4. Выделите основные задачи внутри отдельных этапов работ.
5. Определите зависимости между задачами.
6. Определите порядок выполнения отдельных задач.
7. Назначьте исполнителей на решаемые задачи.
8. Сбалансируйте нагрузку исполнителей.

Форма представления результата:

1. Текст должен быть написан шрифтом Times New Roman, 12. Интервал между строками и абзацами - 1,5. Отступ слева 1,5. Ориентация текста - по ширине страницы. Скриншоты необходимо подписать. Название практической работы, цель работы, ход работы, вывод, ответы на контрольные вопросы должны быть выделены жирным шрифтом.
2. Заключение (выводы).
3. Список используемой литературы.

Контрольные вопросы:

1. Зачем необходимы шаблоны проектов?
2. В чем разница между планированием проекта от даты начала или даты его окончания?
3. Какие существуют базовые календари в программе MS Project?
4. Как внести изменения в базовый календарь?
5. Как включить в проект проектную документацию?
6. Цель разработки модульной структуры.
7. Понятие программного модуля, передачи управления, организации связи по управлению и по данным.
8. Виды связности модулей.
9. Виды целостности модулей.
10. Типовые модульные структуры.
11. Проектирование модульной структуры с помощью структурных карт.
12. Построение структурных карт с помощью программного продукта EasyCASE Professional Version 4.21.016.
13. Опишите понятие техническое задание.
14. Что такое жизненный цикл ПО?
15. В чем разница между параллельным и последовательным выполнением задач?

Критерии оценки:

Работа выполнена полностью и не содержит ошибок, студент учел и рассмотрел особенности предметной области и грамотно представил отчет - оценка «отлично».

Работа выполнена полностью, но содержит не более двух ошибок, студент учел и рассмотрел особенности предметной области и грамотно представил отчет - оценка «хорошо».

Работа выполнена с ошибками, студент не полностью учел и рассмотрел особенности предметной области и представил краткий отчет - оценка «удовлетворительно».

Работа выполнена с грубыми ошибками, студент не учел особенности предметной области, отчет составлен неграмотно - оценка «неудовлетворительно».

Лабораторное занятие № 2

Настройка работы системы контроля версий (типов импортируемых файлов, путей, фильтров и др. параметров импорта в репозиторий)

Цель: Изучить на практике понятия и компоненты систем контроля версий (СКВ), приемы работы с ними.

Выполнив работу, Вы будете:

уметь:

- применять стандартные алгоритмы в соответствующих областях;
- применять нормативные документы, определяющие требования к оформлению программного кода;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации- производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;
- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных.

- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы.

Материальное обеспечение: персональный компьютер, среда программирования Visual Studio 2019.

Задание:

Создать новый проект, осуществить ветвление и слияние в локальном репозитории.

Порядок выполнения работы:

1. Установите TortoiseSVN на компьютере.
2. Создайте новый проект.
3. Создайте локальный репозиторий для своего проекта.
4. Удалите созданный проект на своем компьютере и обновите проект из репозитория.
5. Внесите изменения в файлах с исходными кодами и сохраните изменения в репозитории. Обновите файлы с исходными кодами из репозитория.
6. Внесите изменения в файлах с исходными кодами таким образом, чтобы у двух участников проекта изменения были в одном и том же файле. Попробуйте сохранить изменения в репозитории. Устраните обнаруженные конфликты версий. Повторно сохраните изменения в репозитории.
7. Создайте отдельную ветку проекта. Внесите изменения в файлы с исходными кодами. Сохраните изменения в репозитории.
8. Объедините созданную на предыдущем шаге ветку с основной веткой проекта.
9. Выведите на экран лог изменений файла, в котором было наибольшее количество изменений.
10. Отобразите на экране сравнение файла до и после внесения одного из изменений.
11. Создайте репозиторий в сети Интернет. Повторите шаги 4 - 6.

Краткие теоретические сведения:

Управление версиями - это искусство работы с изменениями информации. Долгое время оно было жизненно важным инструментом программистов, которым необходимо произвести небольшие изменения в программе, или же сделать —откат! изменений, возвращаясь к предыдущей версии. Однако полезность систем управления версиями выходит далеко за пределы мира разработчиков программного обеспечения. Управление версиями требуется повсюду, где можно встретить людей, использующих компьютер для работы с постоянно изменяющейся информацией

Software Configuration Management или **Конфигурационное управление** подразумевает под собой комплекс методов, направленных на то, чтобы систематизировать изменения, вносимые разработчиками в программный продукт в процессе его разработки и сопровождения, сохранить целостность системы после изменений, предотвратить нежелательные и непредсказуемые эффекты, а также сделать процесс внесения изменений более формальным. Изначально управление конфигурацией применялось не в программировании, но в связи с высокой динамичностью сферы разработки ПО, в ней она особенно полезна. К процедурам можно отнести создание резервных копий, контроль исходного кода, требований проекта, документации и т. д. Степень формальности выполнения данных процедур зависит от размеров проекта, и при правильной ее оценке данная концепция может быть очень полезна. Конфигурационное управление требует выполнения множества трудоемких рутинных операций. На практике, в большинстве случаев, для конфигурационного управления применяются специальные системы контроля версий исходного кода программ. В качестве примера такой системы рассмотрим самую распространенную на сегодняшний день -

Subversion.

Subversion - это бесплатная система управления версиями с открытым исходным кодом. Subversion позволяет управлять файлами и каталогами, а так же сделанными в них изменениями во времени. Это позволяет восстановить более ранние версии данных, дает возможность изучить историю всех изменений. Благодаря этому многие считают систему управления версиями своего рода «машиной времени».

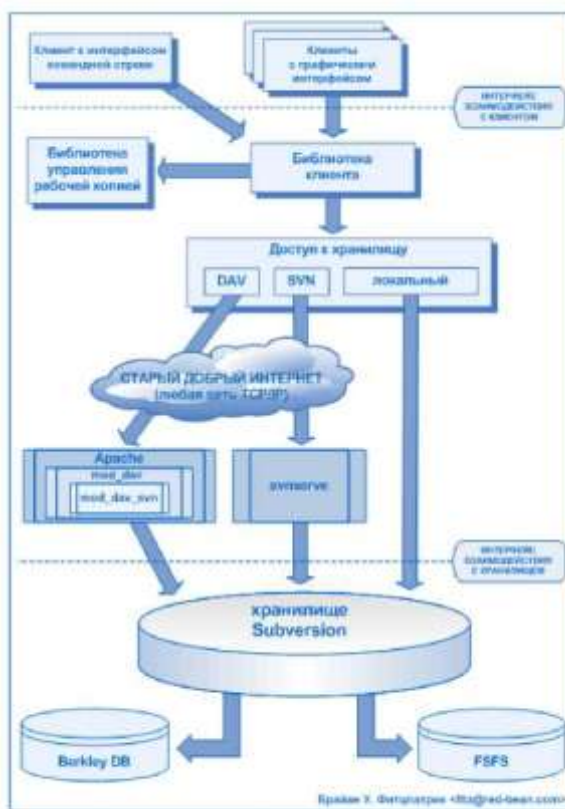


Рисунок 1 – Схема работы

Схема общей архитектуры Subversion. В нижней части схемы изображено хранилище Subversion, в котором хранится информация с версиями. В верхней части схемы показана программа-клиент Subversion, которая управляет локальными отражениями различных фрагментов этих данных (также называемыми «рабочими копиями»). Между этими сторонами проложены различные маршруты, проходящие через разные слои доступа к хранилищу. Некоторые из этих маршрутов используют компьютерные сети и сетевые сервера, чтобы достичь хранилища, в то время как другие маршруты в сети не нуждаются и ведут к хранилищу напрямую.

Subversion может работать через сеть, что позволяет использовать ее на разных компьютерах. В какой-то степени, возможность большого количества людей независимо от их местоположения совместно работать над единым комплектом данных поощряет сотрудничество. Когда нет того ответственного звена цепи, того контролирующего элемента, который утверждает все изменения, работа становится более эффективной. При этом не нужно опасаться, что отказ от контролирующего элемента повлияет на качество, ведь благодаря сохранению истории изменений, даже если при изменении данных будут допущены ошибки, всегда можно сделать откат изменений к прежнему состоянию.

Некоторые системы управления версиями выступают также в качестве систем управления конфигурацией программного обеспечения. Такие системы специально созданы для управления деревьями исходного кода и имеют множество особенностей, непосредственно относящихся к разработке программ: они понимают языки программирования и предоставляют инструменты для сборки программ. Subversion не является такой системой, она представляет собой систему общего назначения, которую можно использовать для управления *любым* набором файлов. Для Вас это будут исходники Ваших программ, а для кого-то другого это будет список продуктов или сведённое цифровое видео.

TortoiseSVN

TortoiseSVN - это бесплатный Windows-клиент с открытым исходным кодом для системы управления версиями Apache™ Subversion®. То есть TortoiseSVN управляет файлами и директориями во времени. Файлы хранятся в центральном *хранилище*. Хранилище больше похоже на обычный файловый сервер, кроме того он запоминает каждое изменение, когда-либо сделанное в ваших файлах и директориях. Это позволяет вам восстановить старые версии ваших файлов и проверить историю изменений — как, когда и кто изменял ваши данные. Вот почему многие думают о Subversion, и вообще о системах управления версиями, как своего рода «машине времени».

Некоторые системы контроля версий являются также и системами управления конфигурацией программ (software configuration management - SCM). Такие системы специально созданы для управления деревьями исходного кода, и имеют множество возможностей, специфичных для разработки программ, таких как непосредственное понимание языков программирования, или предоставление инструментов для сборки программ. Однако Subversion не является такой системой, она является системой общего назначения, которая может быть использована для управления *любым* набором файлов, включая и исходные коды программ.

Возможности TortoiseSVN

Что делает TortoiseSVN таким хорошим клиентом Subversion? Вот краткий список возможностей:

Интеграция с оболочкой

TortoiseSVN интегрируется непосредственно в оболочку Windows (т.е. в Проводник). Это значит, что вы можете работать с уже знакомыми инструментами, и вам не надо переключаться на другое приложение каждый раз, когда вам необходимы функции для управления версиями!

И вам даже не обязательно использовать именно Проводник. Контекстные меню TortoiseSVN работают во многих других файловых менеджерах, и в диалогах для открытия файлов, используемых в большинстве стандартных Windows-приложений. Однако вы должны учитывать, что TortoiseSVN изначально разработан как расширение для Проводника Windows, и, возможно, в других приложениях интеграция будет не полной, например, могут не отображаться пометки на значках.

Пометки на значках

Статус каждого версированного файла и папки отображается при помощи маленькой пометки поверх основного значка. Таким образом, вы сразу можете видеть состояние вашей рабочей копии.

Графический интерфейс пользователя

При просмотре списка изменений файла или папки вы можете кликнуть на ревизию, чтобы увидеть комментарии для этой фиксации. Также доступен список измененных файлов - всего лишь сделайте двойной клик на файле, чтобы увидеть, какие конкретно изменений были внесены.

Диалог фиксации - это список, в котором перечислены все файлы и папки, которые будут включены в фиксацию. У каждого элемента списка имеется флажок, чтобы вы могли выбрать именно то, что вы хотите включить в фиксацию. Неверсированные файлы также могут быть представлены в этом списке, чтобы вы не забыли добавить в фиксацию новый файл или папку.

Простой доступ к командам Subversion

Все команды Subversion доступны из контекстного меню Проводника. TortoiseSVN добавляет туда собственное подменю.

Поскольку TortoiseSVN является клиентом Subversion, мы хотели бы показать и некоторые

из возможностей самой Subversion:

Версирование папок

CVS отслеживает только историю отдельных файлов, тогда как Subversion реализует «виртуальную» версионную файловую систему, которая отслеживает изменения в целых деревьях папок во времени. Файлы *и* папки являются версированными. В результате, есть

команды **переместить** и **копировать**, реально выполняемые на стороне клиента и работающие непосредственно с файлами и папками.

Атомарные фиксации

Фиксация сохраняется в хранилище либо полностью, либо не сохраняется вообще. Это позволяет разработчикам фиксировать изменения, собранные в логически связанные части.

Версированные метаданные

Каждый файл и папка имеет прикрепленный невидимый набор «свойств». Вы можете создавать и сохранять произвольные пары ключ/значение для собственных нужд. Свойства тоже версируются во времени, как и содержимое файла.

Возможность выбора сетевого уровня

В Subversion есть абстрагируемое понятие доступа к хранилищу, которое упрощает реализацию новых сетевых механизмов. «Усовершенствованный» сетевой сервер Subversion является модулем для веб-сервера Apache, который использует для взаимодействия диалект HTTP под названием WebDAV/DeltaV. Это дает Subversion большие преимущества в стабильности и совместимости, и предоставляет различные ключевые возможности без дополнительных затрат: проверка личности (аутентификация), проверка прав доступа (авторизация), сжатие потока данных при передаче, просмотр хранилища. Также доступна меньшая, автономная версия сервера Subversion, взаимодействующая по собственному протоколу, который легко может быть туннелирован через ssh.

Единый способ обработки данных

Subversion получает различия между файлами при помощи бинарного разностного алгоритма, который работает одинаково как с текстовыми (читаемыми человеком), так и с бинарными (не читаемыми человеком) файлами. Оба типа файлов содержатся в хранилище в сжатом виде, а различия передаются по сети в обоих направлениях.

Эффективные ветки и метки

Стоимость создания веток и меток не обязательно должна быть пропорциональна размеру проекта. Subversion создаёт ветки и метки, просто копируя проект с использованием механизма, похожего на жёсткие ссылки в файловых системах. Благодаря этому, операции по созданию веток и меток происходят за одинаковое, очень малое время и занимают очень мало места в хранилище.

Ход работы:

Установка

TortoiseSVN поставляется в виде простого в использовании установочного файла (на странице загрузки <http://tortoisesvn.net/downloads.html> необходимо выбрать дистрибутив, соответствующий разрядности системы). Сделайте на нем двойной клик и следуйте инструкциям - остальное он сделает за вас. Также на странице загрузки необходимо скачать языковой пакет и установить после установки основной программы. Не забудьте перезагрузить компьютер после установки.

Основная концепция

Перед тем, как мы погрузимся в работу с настоящими файлами, важно получить представление о том, как работает Subversion и какие термины используются.

Хранилище

Subversion использует центральную базу данных, которая содержит все ваши версированные файлы с их полной историей. Эта база данных называется *хранилищем*. Хранилище обычно находится на файловом сервере, на котором установлен Subversion, по запросу поставляющий данные клиентам Subversion (например, TortoiseSVN). Если вы делаете резервное копирование, то копируйте ваше хранилище, так как это оригинал всех ваших данных.

Рабочая копия

Это именно то место, где вы работаете. Каждый разработчик имеет собственную рабочую копию, иногда называемую песочницей, на своем локальном компьютере. Вы можете получить из хранилища последнюю версию файлов, поработать над ней локально, никак не взаимодействуя с кем-либо еще, а когда вы будете уверены в изменениях вы можете зафиксировать эти файлы обратно в хранилище.

Рабочая копия не содержит историю проекта, но содержит копию всех файлов, которые были в хранилище до того, как вы начали делать изменения. Это обозначает, что можно легко узнать, какие конкретно изменения вы сделали.

Вам также нужно знать, где найти TortoiseSVN, потому что в меню "Пуск" его нет. TortoiseSVN - расширение проводника Windows, поэтому для начала нужно запустить проводник.

Сделайте правый клик на папке в проводнике, и вы увидите новые пункты в контекстном меню, такие как эти:

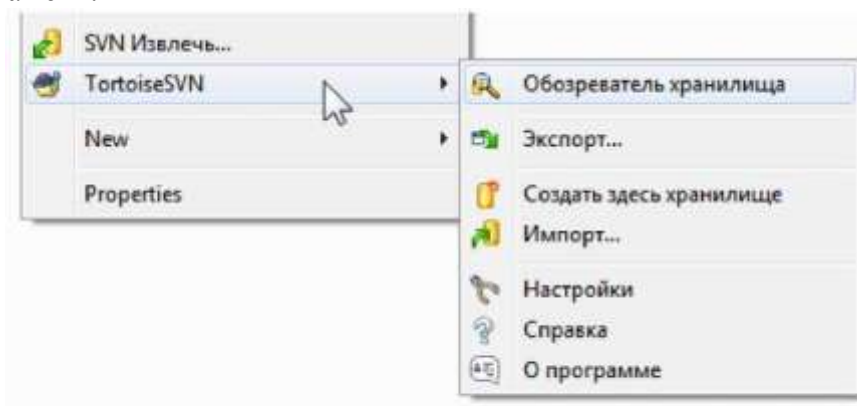


Рис.1. Меню TortoiseSVN для неверсированных папок

Создание хранилища

Для настоящего проекта вам понадобится хранилище, созданное в безопасном месте, и сервер Subversion, чтобы управлять им. Для обучения мы будем использовать функцию локального хранилища Subversion, которая разрешает прямой доступ к хранилищу, созданного на вашем жестком диске, и не требует наличия сервера.

Сначала создайте новую пустую директорию на вашем ПК, где угодно, но в этом руководстве мы собираемся назвать ее C:\svn repos. Теперь сделайте правый клик на новой папке и в контекстном меню выберите **TortoiseSVN ^ Создать здесь хранилище...**. Хранилище, созданное внутри папки, готово к использованию. Также мы создадим внутреннюю структуру папок нажав кнопку.

Импорт проекта

Сейчас у нас есть хранилище, но оно совершенно пустое в данный момент. Давайте предположим, что у меня есть набор файлов в C:\Projects\Widget1, который я хотел бы добавить. Перейдите к папке Widget1 в Проводнике и сделайте правый клик на ней. Теперь выберите пункт **TortoiseSVN Импорт...**, который вызовет диалог:

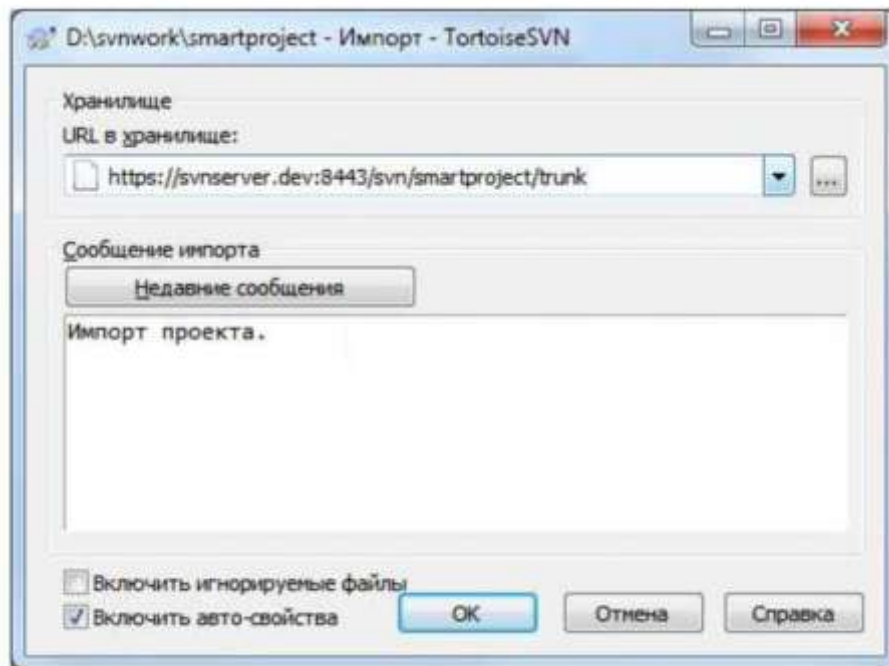


Рис. 2. Диалог импорта

К хранилищу Subversion обращаются по URL-адресу, который позволяет нам указать хранилище где угодно в Интернете. В данном случае нам нужно указать на наше локальное хранилище, которое имеет URL-адрес file:///c:/svn_repos/trunk, и к которому мы добавляем имя нашего проекта Widget1. Обратите внимание, что после file: есть 3 слэша и везде используются прямые слэши.

Другая важная функция данного диалога - это окно Сообщение импорта, в которое вы можете добавить сообщение о том, что вы делаете. Когда вам понадобится просмотреть историю проекта, эти сообщения будут ценным подспорьем для просмотра какие изменения и когда были сделаны. В нашем случае мы напишем что-нибудь простое как «Импорт проекта Виджет1». Нажмите, чтобы добавить папку в ваше хранилище.

Извлечение рабочей копии

Сейчас у н©к есть проект в нашем хранилище, и нам надо создать рабочую копию для повседневной работы. Заметьте, что импортирование папки не превращает автоматически эту папку в рабочую копию. Для создания свежей рабочей копии в Subversion используется термин Извлечь. Мы собираемся извлечь папку Widget1 из нашего хранилища в папку для разработки называемую C:\Projects\Widget1-Dev. Создайте эту папку, затем сделайте правый клик на ней и выберите пункт **TortoiseSVN Извлечь...** Введите URL-адрес для извлечения, в данном случае file:///c:/svn_repos/trunk/Widget1, и кликните на . Наша папка для разработки заполнится файлами из хранилища.

Вы заметите что внешний вид этой папки отличается от обычной папки. У каждого файла появился зелёный флажок в левом углу. Это значки статуса TortoiseSVN, которые присутствуют только в рабочей копии. Зеленый статус означает, что файл не отличается от версии файла, находящегося в хранилище.

Внесение изменений

Можно приступать к работе. В папке Виджет1 -Дев мы начинаем изменять файлы - предположим, мы вносим изменения в файлы Виджет! и ПрочтиМеня/txt. Обратите внимание, что значки на этих файлах теперь стали красными и показывают, что изменения были сделаны локально.

Но какие были изменения? Нажмите правой кнопкой на одном из изменённых файлов и выберите команду **TortoiseSVN Различия**. Запустится инструмент TortoiseSVN для сравнения файлов и покажет какие точно строки в файлах были изменены.

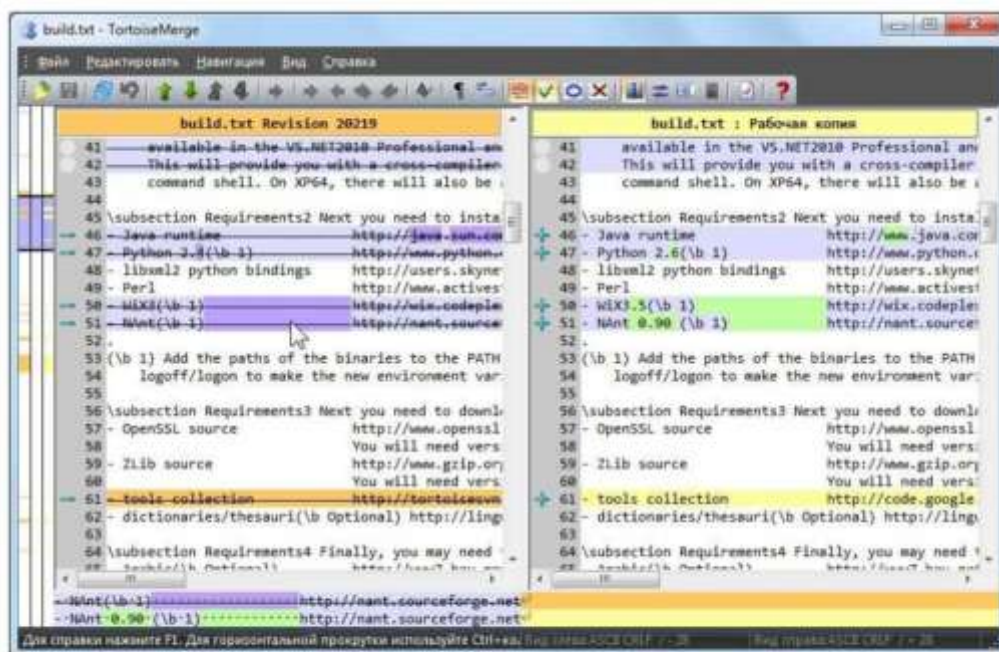


Рис. 3. Просмотрщик изменений в файлах

Ок, настраивают изменения, поэтому давайте обновим хранилище. Это действие называется Фиксировать изменения. Нажмите правой кнопкой на папке Виджет1-Дев и выберите команду **TortoiseSVN Фиксировать**. Появится диалог фиксации со списком изменённых файлов и напротив каждого будет галочка. Вы можете выбрать лишь несколько

файлов из списка для фиксации, но в нашем случае мы будем фиксировать изменения в обоих файлах. Введите сообщение с описанием сделанных изменений и нажмите ОК. Появится диалог с прогрессом процесса фиксации файлов в хранилище и мы закончили фиксацию.

Добавление новых файлов

Во время работы над проектом, вам понадобится добавлять новые файлы - предположим вы добавили новые функции в файле Экстра.с и добавили справку в существующем файле Создать файл. Нажмите правой кнопкой на папке и выберите команду **TortoiseSVN Добавить**. Диалог добавления показывает все неверсированные файлы, и вы можете выбрать те файлы, которые вы хотите добавить. Другой способ добавления файлов - это нажать правой кнопкой на самом файле и выбрать команду **TortoiseSVN Добавить**.

Теперь, если вы откроете папку для фиксации, новый файл будет отображаться как *Добавлен* и существующий файл как *Изменён*. Обратите внимание, что вы можете дважды нажать на изменённый файл, чтобы просмотреть какие именно изменения были сделаны.

Просмотр истории проекта

Одной из самых полезных функций TortoiseSVN является диалоговое окно журнала. Оно показывает список всех фиксаций изменений в файле или папке, а также все те детальные сообщения, которые вы вводили при фиксации изменений ;-)

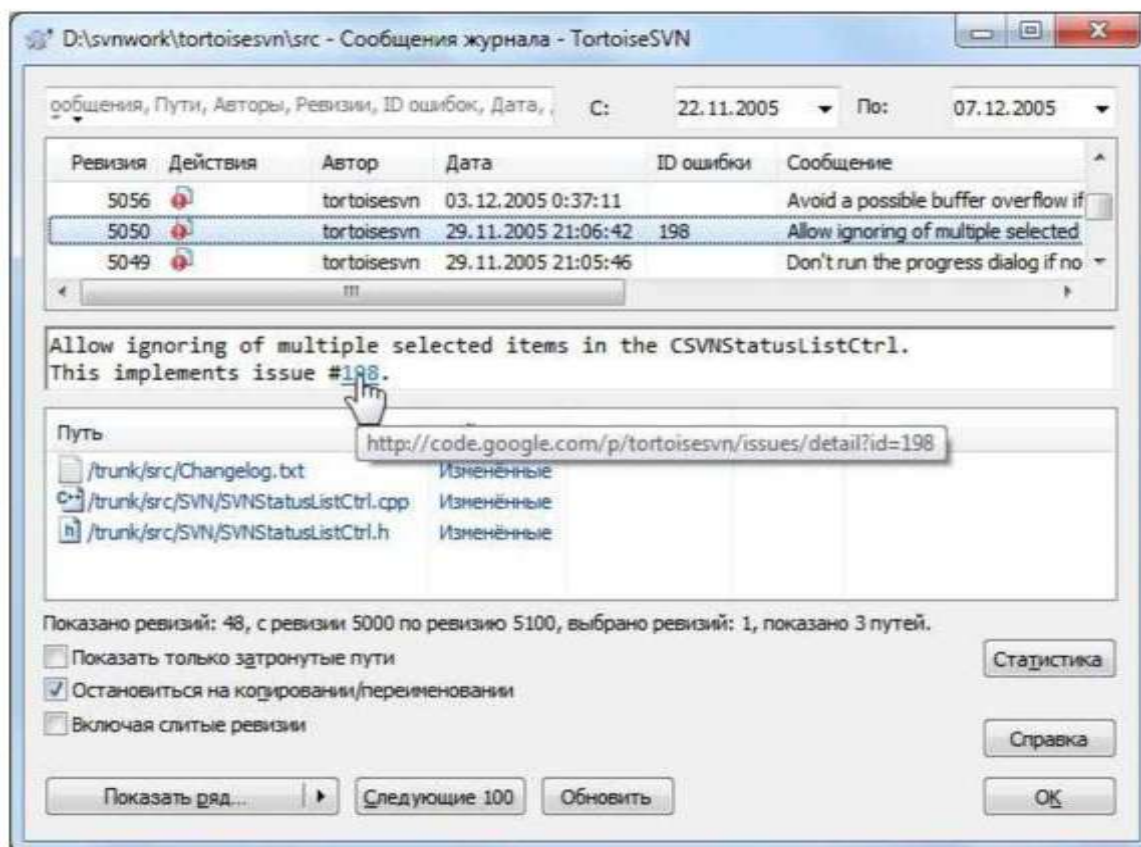


Рис. 4. Диалоговое окно журнала

Верхняя панель показывает список всех фиксированных ревизий вместе с началом сообщения фиксации. Если вы выберете одну из этих ревизий, то средняя панель отобразит полное сообщение журнала для той ревизии и нижняя панель покажет список измененных файлов и папок.

У каждой из этих панелей есть контекстное меню, которое предоставляет много других способов использования информации. В нижней панели вы можете **дважды нажать** на файл, чтобы просмотреть какие именно изменения были внесены в той ревизии. Прочтите [«Диалоговое окно журнала ревизий»](#), чтобы узнать больше.

Отмена изменений

Одной общей функцией всех систем управления ревизиями является функция, которая позволяет вам отменить изменения, которые вы внесли ранее. Как вы и догадались, в TortoiseSVN это легко сделать.

Если вы хотите избавиться от изменений, которые вы еще не успели фиксировать и

восстановить нужный файл в том виде, в котором он был перед началом изменений, то выберите команду **TortoiseSVN ^ Убрать изменения**. Это действие отменит ваши изменения (в Корзину) и вернет фиксированную версию файла, с которой вы начинали. Если же вы хотите убрать лишь некоторых изменения, то вы можете использовать инструмент TortoiseMerge для просмотра изменений и выборочного удаления измененных строк.

Если вы хотите отменить действия определенной ревизии, то начните с диалогового окна журнала и найдите проблемную ревизию. Выберите команду **Контекстное меню Отменить изменения из этой ревизии** и те изменения будут отменены.

Работа с сетью.

С Subversion можно работать как посредством сети интернет, так и локально. Воспользуемся сервисом [Assembla \(https://www.assembla.com/\)](https://www.assembla.com/). Зарегистрировавшись там, вы получите 1 Gb места под репозиторий. Создав его и настроив, вы получите ссылку вида https://subversion.assembla.com/svn/название_репозитория, которую можно использовать в любом SVN клиенте. К примеру, чтобы в Visual SVN добавить свой проект в репозиторий, вам нужно нажать Add Solution to Subversion, после чего указать локальное хранилище вашего проекта, нажать Далее и ввести вашу ссылку. Все, теперь сверху появится панель с основными SVN- функциями (Show Log, Update, Commit, Switch Branch, Branch и Merge) и можно приступать к полноценной работе.



В общем и целом, сайт позволяет так же настроить репозиторий и получить на неделю или две пробный премиум (платные доп. функции, источник жизни сайта). Основные функции же полностью бесплатны. Чтобы товарищи по команде могли работать с общим для команды репозиторием, на assembla.com, необходима их регистрация на сайте. После регистрации, пользователи могут быть добавлены владельце репозитория в список команды. На самом сайте можно посмотреть всю информацию о проекте и изменениях, и даже поставить свой баннер с ссылкой.

Так, например, выглядит страница Stream, где отображаются последние изменения.



Форма представления результата:

1. Текст должен быть написан шрифтом Times New Roman, 12. Интервал между строками и абзацами - 1,5. Отступ слева 1,5. Ориентация текста - по ширине страницы. Скриншоты необходимо подписать. Название практической работы, цель работы, ход работы, вывод, ответы на контрольные вопросы должны быть выделены жирным шрифтом.

2. Заключение (выводы).
3. Список используемой литературы.

Контрольные вопросы:

1. Что такое системы контроля версий (СКВ) и для решения каких задач они

предназначаются?

2. Объясните следующие понятия СКВ и их отношения: хранилище, commit, история, рабочая копия.

3. Что представляют собой и чем отличаются централизованные и децентрализованные СКВ? Приведите примеры СКВ каждого вида.

4. Опишите действия с СКВ при единоличной работе с хранилищем.

5. Опишите порядок работы с общим хранилищем в централизованной СКВ.

6. Что такое и зачем может быть нужна разность (diff)?

7. Что такое и зачем может быть нужно слияние (merge)?

8. Что такое конфликты (conflict) и каков процесс их разрешения (resolve)?

9. Поясните процесс синхронизации с общим хранилищем («обновления») в децентрализованной СКВ.

10. Что такое и зачем могут быть нужны ветви (branches)?

11. Объясните смысл действия rebase в СКВ Git.

12. Как и зачем можно игнорировать некоторые файлы при commit?

Критерии оценки:

Работа выполнена полностью и не содержит ошибок, студент учел и рассмотрел особенности предметной области и грамотно представил отчет - оценка «отлично».

Работа выполнена полностью, но содержит не более двух ошибок, студент учел и рассмотрел особенности предметной области и грамотно представил отчет - оценка «хорошо».

Работа выполнена с ошибками, студент не полностью учел и рассмотрел особенности предметной области и представил краткий отчет - оценка «удовлетворительно».

Работа выполнена с грубыми ошибками, студент не учел особенности предметной области, отчет составлен неграмотно - оценка «неудовлетворительно».

Лабораторное занятие № 3

Разработка и интеграция модулей проекта (командная работа)

Цель: закрепление практических навыки работы с системой Visual Studio 2019, MS SQL Server, проведение интеграции программных модулей.

Выполнив работу, Вы будете:

уметь:

- применять стандартные алгоритмы в соответствующих областях;
- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры;
- выполнять процедуры сборки программных модулей и компонент в программный продукт;
- производить настройки параметров программного продукта и осуществлять запуск процедур сборки;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации- производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;
- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- организовывать работу коллектива и команды;
- эффективно работать в команде;
- взаимодействовать с коллегами, руководством, клиентами в ходе профессиональной деятельности;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- соблюдать нормы экологической безопасности;

- пользоваться средствами профилактики перенапряжения, характерными для данной специальности;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение: персональный компьютер, среда программирования Visual Studio 2019, MS SQL Server.

Задание:

1. Разработать базу данных, включающую в себя таблицу Пользователи, используя среду MS SQL Server.
2. Создать приложение.
3. Создать модель данных на основе разработанной ранее базы данных.
4. Сохранить данные пользователя в созданной базе данных.
5. Создать отчет о проделанной работе.

Ход работы:

Для создания базы данных в СУБД, запустите MS SQL Server Management Studio. Установите соединение с сервером.

Для реализации базы данных, хранящей сведения о пользователях необходимо создать две сущности (таблицы): Пользователь (хранит основные сведения, характеризующие пользователя) и

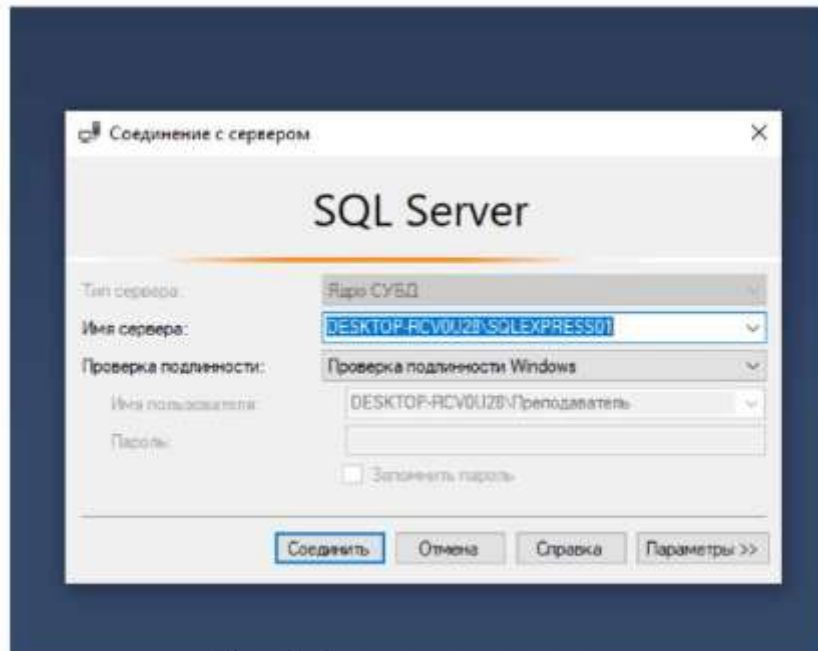


Рис. 1. Соединение с сервером

Секретный вопрос (хранит список секретных вопросов, необходимых для авторизации) с указанными атрибутами.

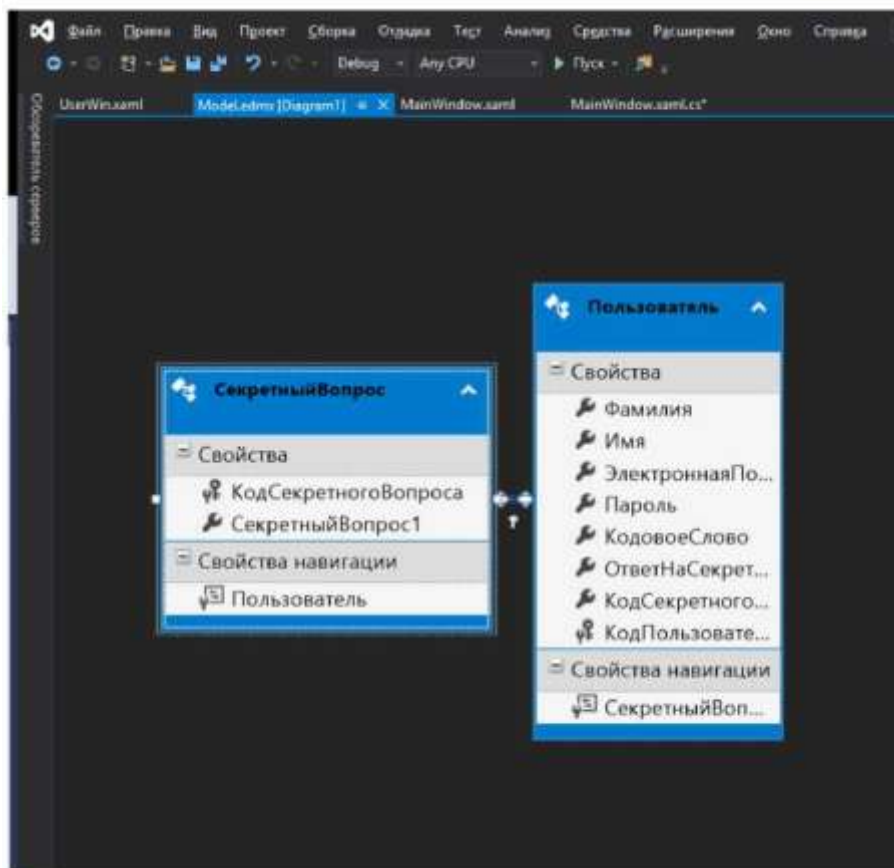
На ключевых полях КодПользователя и КодСекретногоВопроса настройте спецификацию идентификатора с начальным значением - 1 и шагом приращения - 1 для автоматического заполнения указанных полей.

СекретныйВопрос				
Имя столбца	Сжатый тип	Допускает значения NULL	Идентификатор	
КодСекретн...	int	Нет	<input checked="" type="checkbox"/>	
Секретный...	nvarchar(1...	Нет	<input type="checkbox"/>	
			<input type="checkbox"/>	

Пользователь				
Имя столбца	Сжатый тип	Допускает значения NULL	Идентификатор	
Фамилия	nvarchar(50)	Нет	<input type="checkbox"/>	
Имя	nvarchar(50)	Нет	<input type="checkbox"/>	
Электронна...	nvarchar(50)	Нет	<input type="checkbox"/>	
Пароль	nvarchar(50)	Нет	<input type="checkbox"/>	
КодовоеСл...	nvarchar(50)	Нет	<input type="checkbox"/>	
ОтветНаСе...	nvarchar(50)	Нет	<input type="checkbox"/>	
КодСекретн...	int	Нет	<input type="checkbox"/>	
КодПользо...	int	Нет	<input checked="" type="checkbox"/>	
			<input type="checkbox"/>	

Рис. 2. Диаграмма базы данных

Для передачи данных необходимо создать модель данных



Форма представления результата:

1. Текст должен быть написан шрифтом Times New Roman, 12. Интервал между строками и абзацами - 1,5. Отступ слева 1,5. Ориентация текста - по ширине страницы. Скриншоты необходимо подписать. Название практической работы, цель работы, ход работы, вывод, ответы на контрольные вопросы должны быть выделены жирным шрифтом.

2. Заключение (выводы).

3. Список используемой литературы.

Контрольные вопросы:

1. Что такое интеграция?
2. Как реализуется создание базы данных?
3. Что такое модель данных?
4. Как передаются данные в MSSQL Server

Критерии оценки:

Работа выполнена полностью и не содержит ошибок, студент учел и рассмотрел особенности предметной области и грамотно представил отчет - оценка «отлично».

Работа выполнена полностью, но содержит не более двух ошибок, студент учел и рассмотрел особенности предметной области и грамотно представил отчет - оценка «хорошо».

Работа выполнена с ошибками, студент не полностью учел и рассмотрел особенности предметной области и представил краткий отчет - оценка «удовлетворительно».

Работа выполнена с грубыми ошибками, студент не учел особенности предметной области, отчет составлен неграмотно - оценка «неудовлетворительно».

Лабораторное занятие № 4 Интеграция модулей проекта (командная работа)

Цель: закрепление практических навыков работы с системой Visual Studio 2019, MS SQL Server, проведение интеграции программных модулей.

Выполнив работу, Вы будете:

уметь:

- применять стандартные алгоритмы в соответствующих областях;
- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры;
- выполнять процедуры сборки программных модулей и компонент в программный продукт;
- производить настройки параметров программного продукта и осуществлять запуск процедур сборки;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации- производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;
- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- организовывать работу коллектива и команды;
- эффективно работать в команде;
- взаимодействовать с коллегами, руководством, клиентами в ходе профессиональной деятельности;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- соблюдать нормы экологической безопасности;
- пользоваться средствами профилактики перенапряжения, характерными для данной специальности;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение: персональный компьютер, среда программирования Visual Studio 2019, MS SQL Server.

Задание:

В проекте, разработанном на лабораторном занятии № 3, создать окно авторизации пользователя.

Код страницы авторизации показан в приложении 1. Ход работы:

Окно авторизации настройте следующим образом (Рис. 2).

Рис 3. Окно авторизации

После создания модели создаем новый экземпляр класса модели данных и используя технологию Entity Framework передаем данные в базу.

```

private void Window_Loaded(object sender, RoutedEventArgs e)
{
    quest.ItemsSource = context.СекретныйВопрос.Select(i => i.СекретныйВопрос1).ToList();
}

private void SignIn_Click(object sender, RoutedEventArgs e)
{
    context.Пользователь.Add(new Пользователь
    {
        Фамилия = fName.Text,
        Имя = name.Text,
        ЭлектроннаяПочта = eMail.Text,
        Пароль = passw.Password,
        КодовоеСлово = word.Text,
        ОтветНаСекретныйВопрос = atvet.Text,
        КодСекретногоВопроса = context.СекретныйВопрос.Where(i => i.СекретныйВопрос1 == quest.Text).First().КодСекретногоВопроса;
    });
    context.SaveChanges();
    MessageBox.Show($"Пользователь {fName.Text} добавлен");
}

```

Форма представления результата:

Текст должен быть написан шрифтом Times New Roman, 12. Интервал между строками и абзацами - 1,5. Отступ слева 1,5. Ориентация текста - по ширине страницы. Скриншоты необходимо подписать. Название практической работы, цель работы, ход работы, вывод, ответы на контрольные вопросы должны быть выделены жирным шрифтом.

Критерии оценки:

Работа выполнена полностью и не содержит ошибок, студент учел и рассмотрел особенности предметной области и грамотно представил отчет - оценка «отлично».

Работа выполнена полностью, но содержит не более двух ошибок, студент учел и рассмотрел особенности предметной области и грамотно представил отчет - оценка «хорошо».

Работа выполнена с ошибками, студент не полностью учел и рассмотрел особенности предметной области и представил краткий отчет - оценка «удовлетворительно».

Работа выполнена с грубыми ошибками, студент не учел особенности предметной области, отчет составлен неграмотно - оценка «неудовлетворительно».

Лабораторное занятие № 5

Работа в среде программирования и отладки Keil-C

Цель работы: изучить интегрированную среду программирования Keil-C.

Выполнив работу, Вы будете:

уметь:

- формирование умений применять стандартные алгоритмы в соответствующих областях;
- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять современные компиляторы, отладчики и оптимизаторы программного кода;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы.

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением.
2. Мультимедиа проектор.
3. Лабораторные стенды «LESO1»

Задание: Изучить следующие вопросы:

1. Построение файловой системы персонального компьютера.
2. Изучить правила пользования текстовым редактором.
3. Создание и сохранение текстовых файлов.
4. Изучить порядок создания программного проекта в интегрированной среде программирования Keil-C.
5. Изучить настройку свойств программного проекта в интегрированной среде программирования Keil-C.
6. Изучить использование окна управления программным проектом в интегрированной среде программирования Keil-C.
7. Изучить методы трансляции отдельных файлов в интегрированной среде программирования Keil-C.
8. Изучить методы трансляции программного проекта в интегрированной среде программирования Keil-C.
9. Изучить работу отладчика программ в интегрированной среде программирования Keil-C.

Краткие теоретические сведения:

1 Написание программы

В настоящее время программа пишется на одном из языков программирования в виде текстового файла. Это означает, что для написания программы можно воспользоваться любым текстовым редактором. Для того чтобы программа-транслятор могла преобразовать исходный текст программы в машинные коды микропроцессора, этот текст программы должен быть записан с использованием символов ASCII или ANSI таблиц. К сожалению, некоторые текстовые редакторы для увеличения возможностей редактирования используют для записи текстов формат rtf, или свои собственные форматы (например, редактор WORD). Такие текстовые файлы не понимаются программами-трансляторами и, следовательно, не могут быть использованы для записи исходного текста программы.

2 Использование интегрированной среды программирования

Для облегчения процесса разработки программы часто используются интегрированные среды программирования. В состав интегрированной среды программирования включается определенный набор программных средств: редактор исходного текста, трансляторы с

выбранного языка программирования, редакторы связей, загрузчики и так далее. Редактор текстов обычно является первой программой, которую приходится использовать в процессе разработки программ: благодаря нему пользователь получает возможность набирать исходные тексты программ, написанных на ассемблере или на каком-нибудь языке высокого уровня и сохранять их на жёстком диске.

3 Работа с текстовым редактором интегрированной среды программирования Keil-C

Работа с текстовыми файлами начинается с создания нового файла. Создать текстовый файл можно несколькими способами. Первый способ – воспользоваться главным меню, как показано на рисунке 26.

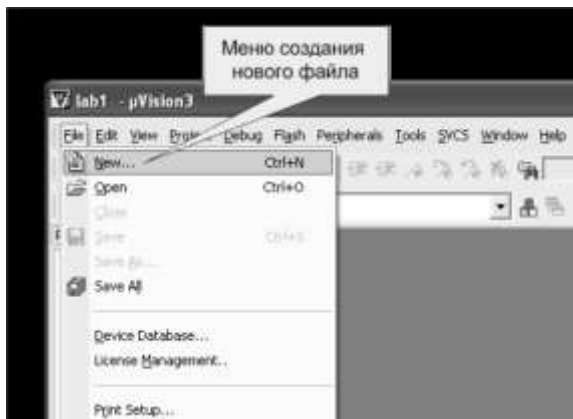


Рисунок 26 – Создание нового файла через главное меню

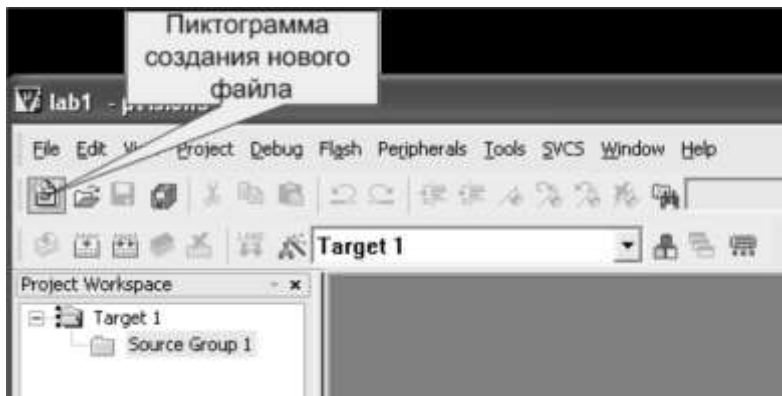


Рисунок 27 – Создание нового файла при помощи пиктограммы

Второй способ – использовать быстрые клавиши Ctrl+N. И третий способ – это нажать на пиктограмму создания нового файла, как показано на рисунке 27. После выполнения этих действий открывается окно текстового редактора, в котором можно вводить исходный текст программы.

Внешний вид программы с открытым окном текстового редактора показан на рисунке 28. Ввод программы производится с клавиатуры. Стирание одиночных ошибочно введённых символов возможно при помощи кнопок —Delete| и —Backspace|

После создания окна нового файла можно вводить исходный текст программы, используя клавиатуру. Набрав исходный текст программы в окне текстового редактора, файл необходимо сохранить на диске компьютера. Для этого можно воспользоваться меню файл, как это показано на рисунке 28:

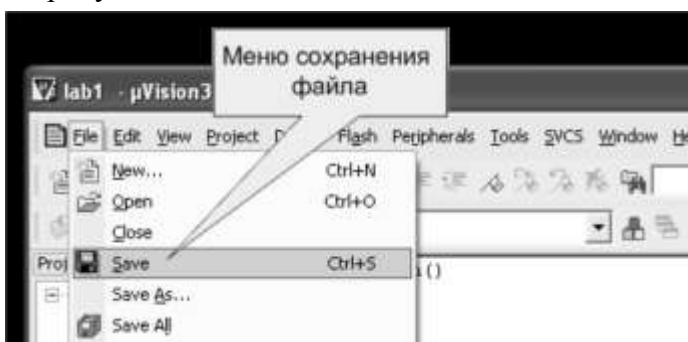


Рисунок 29 – Сохранение файла при помощи пиктограммы

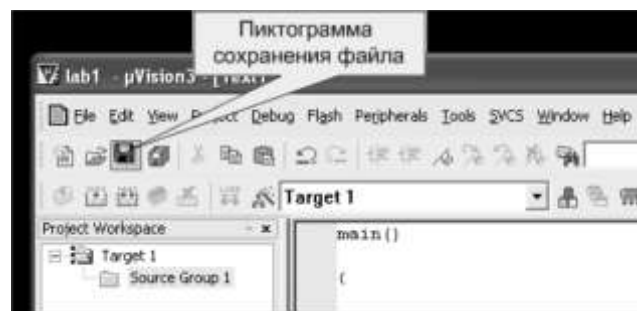


Рисунок 28 – Сохранение файла через главное меню

Второй способ – использовать быстрые клавиши Ctrl+S. И третий способ – это нажать на пиктограмму сохранения файла, как показано на рисунке 29. При написании программ часто требуется копировать участки программ из одного файла в другой. Для этого в текстовом редакторе открываются оба файла. Затем необходимый участок текста выделяется при помощи мыши или клавиатуры. Для выделения строк нажимается левая кнопка мыши в начале выделяемого фрагмента и, не отпуская её, курсор мыши перемещается в конец этого фрагмента.

Для выделения столбцов производятся те же действия, но, кроме того, нажимается кнопка —Alt на клавиатуре.

После выделения необходимого фрагмента текста, этот фрагмент копируется в буфер обмена. Скопировать можно, щёлкнув мышью по пиктограмме копирования, как это показано на рисунке 30, кроме того удобно копировать в буфер обмена комбинацией клавиш —Ctrl+Cl.

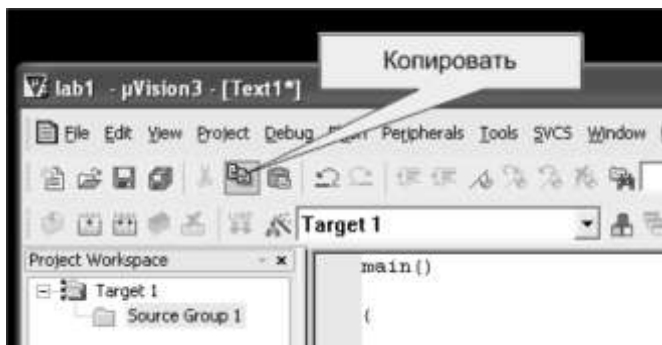


Рисунок 30 – Копирование выделенного фрагмента в буфер обмена при помощи пиктограммы

Теперь можно переключиться в окно редактирования файла, куда нужно поместить скопированный текстовый фрагмент при помощи главного меню, как это показано на рисунке 31, и вставить этот фрагмент перед текстовым курсором, вставку также можно осуществить нажатием —Ctrl+V. Вставка фрагмента из буфера обмена может быть произведена либо из меню —Edit, либо при помощи пиктограммы —Paste как это показано на рисунке 32. Фрагмент исходного текста будет вставлен в то место набираемого текста, где в настоящее время находится курсор.

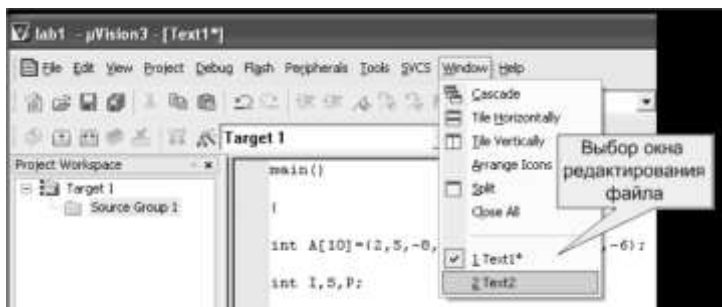


Рисунок 31 – Выбор окна редактирования файла

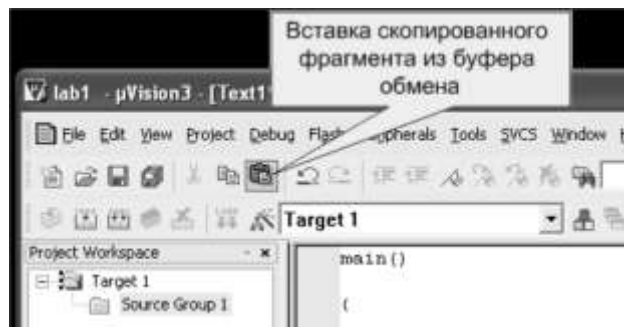


Рисунок 32 – Вставка скопированного фрагмента из буфера обмена

Напомним, что копирование участка исходного текста программы эквивалентно набору этого текста программы с клавиатуры и поэтому файл, в который производилось копирование текста, необходимо сохранить на диске любым из методов, описанных ранее.

4 Создание программных проектов

4.1 Разработка программных средств

На первом шаге разработки программных средств формулируются технические требования к системе, и составляется блок-схема процесса решения нужных задач, которая обеспечит реализацию заданных требований. Блок-схема должна быть надлежащим образом структурирована, чтобы гарантировалась высокая эффективность логики программ.

Для того чтобы было легко ориентироваться в файлах на компьютере, каждую задачу помещают в отдельную директорию (папку). Написание программы тоже следует начинать в отдельной папке, тем более что как будет показано далее, даже простейшая программа состоит из нескольких файлов.

В большинстве случаев программа состоит из нескольких программных модулей. Использование в составе одной программы нескольких программных модулей позволяет увеличить скорость трансляции программ, поручать написание программных модулей различным программистам, увеличивать понятность программ.

Принцип разбиения единой программы на программные модули заключается в том, что для реализации работы с отдельными узлами аппаратуры пишутся отдельные подпрограммы, которые относительно слабо связаны с остальными частями программы. Эти подпрограммы можно выделить в отдельную программу, которую можно хранить в отдельном файле, и транслировать отдельно от остальной программы. Часто одни и те же модули могут входить в состав нескольких программ, выполняющих различные задачи, но использующие при этом одни и те же устройства.

В качестве примера можно назвать работу с клавиатурой, индикацию различных видов информации, работу с последовательными портами, с АЦП, с ЦАП. Каждое из этих устройств может обслуживаться отдельными программными модулями. Этот список можно продолжать и далее, но для составления представления о программных модулях этого достаточно.

Программные проекты можно создавать, и поддерживать вручную, но в последнее время обычно используются различные системы поддержки разработок. Это создаёт ряд дополнительных преимуществ.

Часто программа пишется, и отлаживается на одной аппаратуре (например, на оценочных платах, предлагаемых фирмами изготовителями микросхем), а используется на другой. При этом программа при отладке незначительно отличается от программы, которая будет использоваться в реальной аппаратуре. Для этого в составе программного проекта создаются назначения проекта. В качестве примера назначений программного проекта можно назвать отладку и реализацию, а также версии программы.

4.2 Использование системы поддержки разработок

Для облегчения процесса разработки программы часто используются системы поддержки разработок. В состав систем поддержки разработок включается определенный набор программных средств: редакторы текстов, ассемблер, редакторы связей, компиляторы, загрузчики и тому подобное. Каждая из этих программ создаёт свой файл (или свои файлы) на диске компьютера. Для того чтобы не запутаться в этих файлах при разработке программы, все эти файлы размещаются в своей, отдельной директории. Имя этой директории обычно назначают по имени программно-аппаратного проекта. Например: измеритель характеристик транзисторов или цифровой осциллограф.

4.3 Создание программного проекта в интегрированной среде программирования Keil-C

Работа с программными проектами начинается с создания нового файла проекта. Для создания файла проекта в интегрированной среде разработки программ можно воспользоваться главным меню, как показано на рисунке 33.

После создания новой директории и нового файла программного проекта, интегрированная среда программирования предлагает выбрать конкретную микросхему из семейства MCS-51, как это показано на рисунке 34.

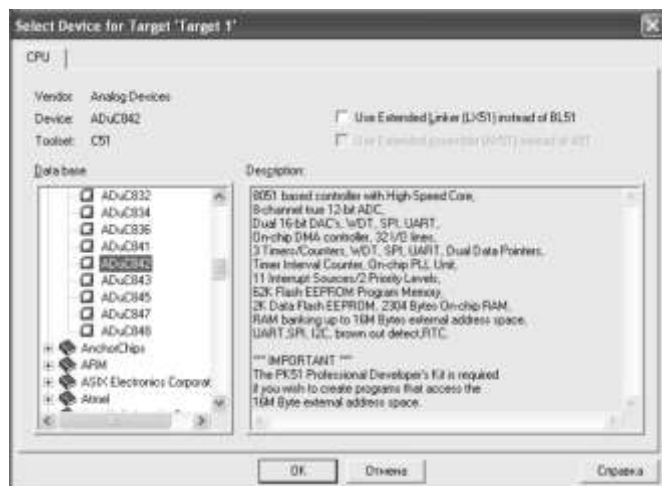
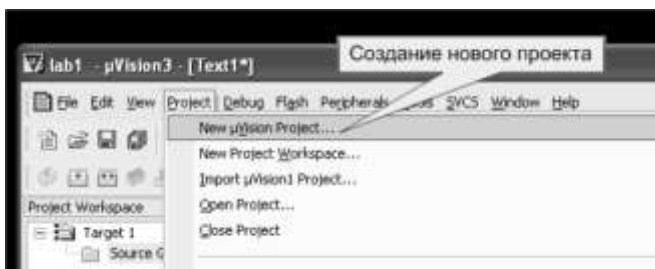


Рисунок 33 – Создание нового программного проекта

При этом в интегрированной среде программирования окно менеджера проекта приобретает вид, показанный на рисунке 35. Название назначения программного проекта можно изменить, щёлкнув манипулятором —мышь по названию назначения программного проекта в окне менеджера проекта (Например: отладка, реализация или сопровождение). Точно так же можно изменить название устройства в составе программного проекта (Например: носимая радиостанция, автомобильная радиостанция, стационарная радиостанция или базовая радиостанция).

Рисунок 34 – Диалоговое окно выбора микросхемы для программного проекта

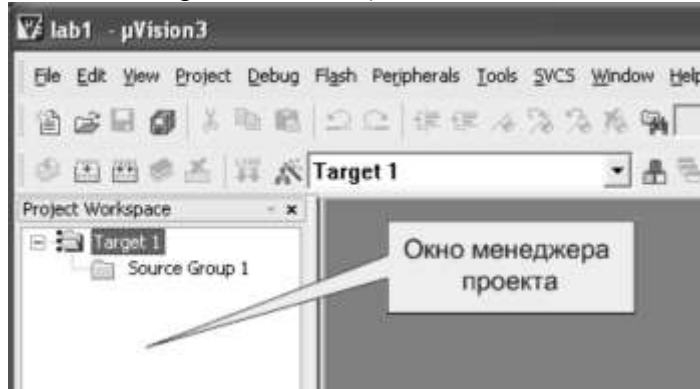


Рисунок 35 – Внешний вид окна менеджера проекта после создания программного проекта

4.4 Настройка свойств программного проекта в интегрированной среде программирования Keil-C

После создания программного проекта в интегрированной среде программирования keil-c конечным файлом трансляции является абсолютный файл. Для загрузки в микросхему обычно используется HEX файл. Для создания этого файла необходимо включить соответствующую опцию в свойствах программного проекта.

Изменить свойства программного проекта можно несколькими способами. Первый способ – воспользоваться главным меню, как показано на рисунке 36. Второй способ – это нажать на кнопку изменения свойств программного проекта, как показано на рисунке 37.

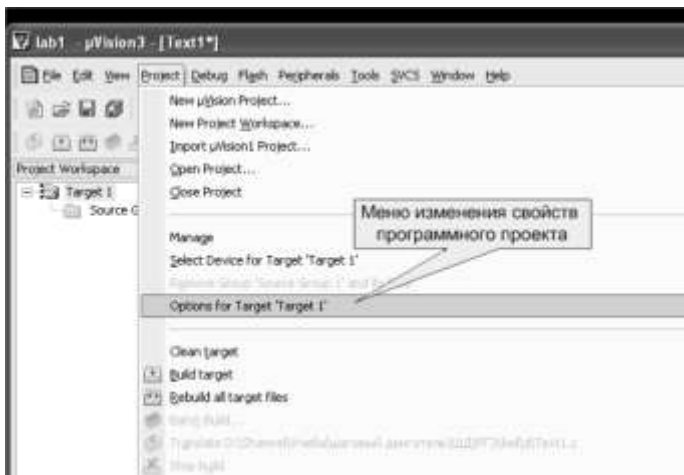


Рисунок 36 - Изменение свойств программного проекта

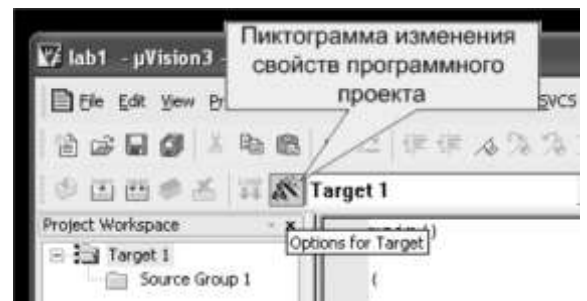


Рисунок 37 – Изменение свойств программного проекта при помощи пиктограммы

При этом

на экране компьютера появляется диалоговое окно изменения свойств программного проекта как показано на рисунке 38. В этом окне необходимо ввести параметры внешней памяти программ и памяти данных.

Затем необходимо установить выходные параметры программного проекта. Для этого открываем закладку выход (Output), как это показано на рисунке 39. В этой закладке убеждаемся, что установлена галочка создания выходного загрузочного файла в .hex формате. Для того чтобы не загромождать директорию проекта файлами объектных кодов можно создать отдельную директорию. Например, с названием OBJ. Новая директория может быть создана после нажатия на кнопку —Select Folder for Objectl.

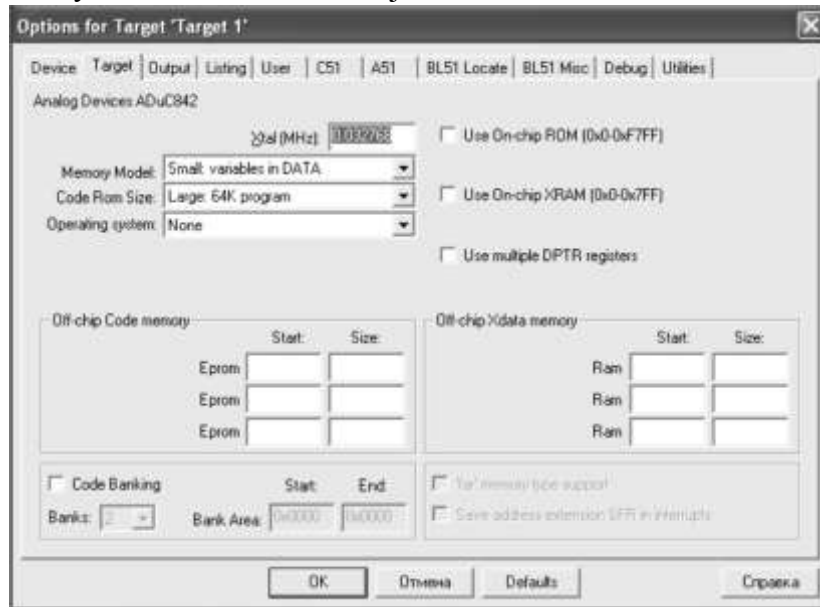


Рисунок 38 – Диалоговое окно настройки свойств программного проекта

Точно так же можно создать директорию (папку) для файлов листингов. Для этого необходимо выбрать закладку —Listing. В файлах листингов помещается информация об ошибках, ассемблерный код и соответствующий ему машинный код программного модуля. Использование листингов позволяет оптимизировать программу, а при работе без интегрированной среды программирования и находить синтаксические ошибки программы. Отметим, что создание файлов листингов замедляет процесс трансляции. Обычно имя для папки листингов выбирают LST.

Необходимость создания отдельных папок для листингов и объектных кодов возникает при большом количестве программных модулей, а значит при большом количестве файлов в одной папке. Обычно директории для листингов и для объектных кодов располагают внутри папки программного проекта, где содержатся исходные тексты программы.

Для настройки параметров компиляции выбирается закладка —C51. В этой закладке настраивается уровень оптимизации транслируемого программного модуля и цель оптимизации (по скорости работы программы или по размеру выходного файла). Кроме того, в этой закладке заносится адрес векторов прерывания. После настройки свойств программного проекта в диалоговом окне, это окно закрывается нажатием кнопки —OK.

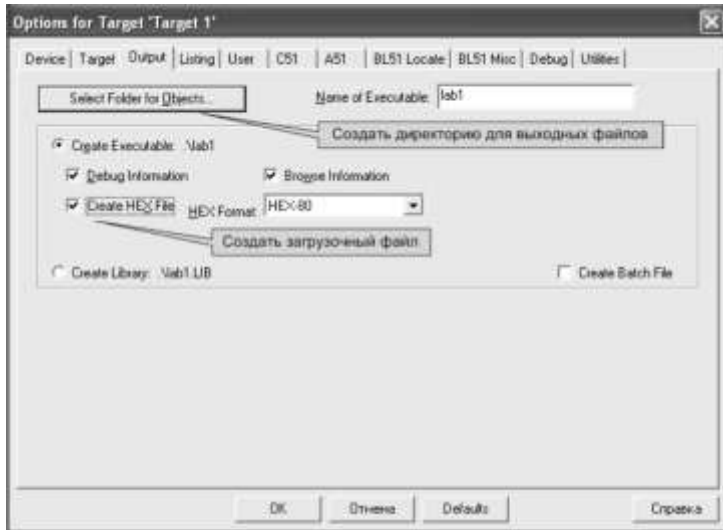


Рисунок 39 – Диалоговое окно настройки выходных параметров программного проекта

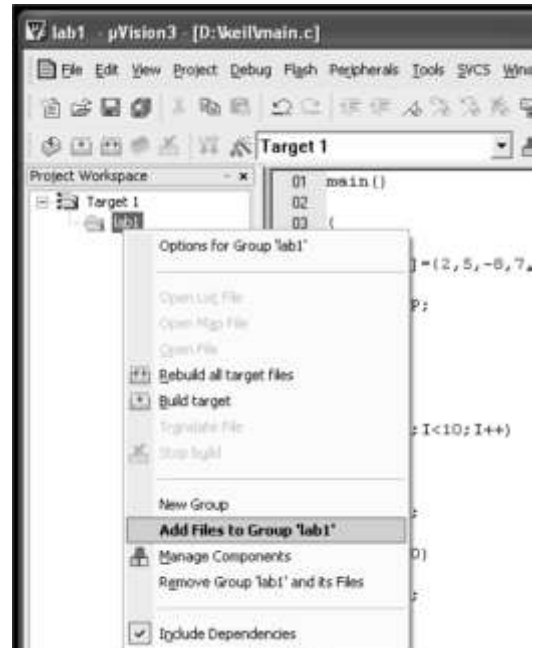


Рисунок 40 – Всплывающее меню менеджера проектов с выбранной опцией добавления файлов к программному проекту

Теперь можно подключать к программному проекту файлы с исходным текстом программных модулей. Для этого можно щёлкнуть правой кнопкой мыши по значку группы файлов в окне менеджера проектов, как это показано на рисунке 40, и выбрать опцию добавления файлов к программному проекту. Добавляемые файлы должны быть предварительно созданы. Если файлом является программа на языке C, то этот файл должен иметь расширение .C. Например, LAB1.c. Если проект состоит из нескольких программных модулей, опцию добавления файлов следует повторить соответствующее число раз.

4.5 Работа с программным проектом в интегрированной среде программирования Keil-C

После завершения создания программного проекта можно открыть исходные тексты программных модулей, щёлкнув левой кнопкой мыши на названии соответствующего файла в окне менеджера проектов.

Точно так же можно переключаться между программными модулями проекта, используя окно менеджера проектов.

В случае необходимости можно выключать окно менеджера программных проектов, но обычно это окно удобно при написании программы.

Если окно менеджера проектов отключено, то переключаться между модулями можно, используя меню —window.

5 Указания по трансляции программ и программных проектов

5.1 Трансляция программных модулей

Как уже указывалось, программный проект может состоять из нескольких файлов. В программном проекте, состоящем из нескольких файлов, появляется возможность нескольких видов трансляции. Прежде всего, можно оттранслировать только один файл, не транслируя остальные файлы программного проекта. Это позволяет обнаружить, и исправить все синтаксические ошибки в отдельном программном модуле.

При трансляции каждого программного модуля на жёстком диске формируются перемещаемый файл в объектном формате, который затем будет использоваться для создания загрузочного файла программного проекта. Этот файл называется объектный модуль. Одновременно с объектным модулем на диске формируется файл листинга программного модуля, в который помещается исходный текст программного модуля и сообщения о синтаксических ошибках.

При трансляции программного модуля, написанного на языке программирования ассемблер, в листинг помещаются машинные коды команд процессора, и их адрес относительно начала программного модуля. При трансляции исходного текста программы, написанной на языке программирования высокого уровня, таком как С программа транслятор может быть настроена так, что она будет создавать файл с текстом исходного модуля, написанного на ассемблере или помещать этот текст в листинг программного модуля. Таким образом, появляется возможность писать, и отлаживать программу на языке высокого уровня, а затем переводить её на язык программирования ассемблер и оптимизировать её вручную.

5.2 Связывание объектных модулей и получение загрузочного файла

После того, как оттранслированы без ошибок все программные модули, и тем самым получены файлы объектных модулей, производится трансляция всего программного проекта (связывание объектных модулей). При этом на диске формируются абсолютный и загрузочный файлы программного проекта. Для контроля ошибок связывания формируется файл листинга редактора связей с расширением m51.

Если обнаруживаются ошибки связывания, то абсолютный и загрузочный файлы программного проекта не формируются. В этом случае необходимо изменить исходный текст программного модуля из-за которого возникла ошибка связывания, оттранслировать его и снова попытаться произвести трансляцию программного проекта (связывание объектных модулей).

После получения загрузочного модуля можно начинать отладку программы.

5.3 Трансляция программных проектов

Интегрированная среда программирования позволяет максимально облегчить трансляцию программных проектов. Так как параметры программного проекта уже настроены, то для трансляции исходного текста программного модуля достаточно загрузить исходный текст этого программного модуля в окно текстового редактора. Это можно сделать одним из способов рассмотренных ранее.

После загрузки исходного текста программного модуля достаточно нажать на кнопку трансляции программного модуля, как это показано на рисунке 41.

Ещё один способ трансляции программного модуля, это воспользоваться главным меню, как это показано на рисунке 42.

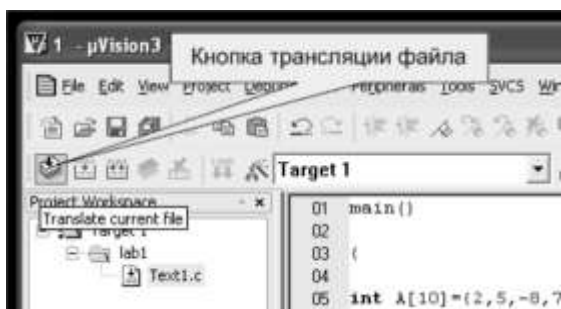


Рисунок 41 – Трансляция программного модуля при помощи кнопки трансляции файла

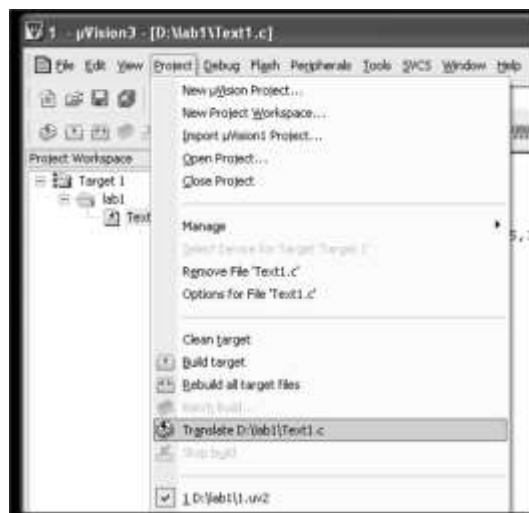


Рисунок 42 – Трансляция программного модуля при помощи главного меню

Надо отметить, что в составе интегрированной среды программирования для поиска синтаксических ошибок удобнее пользоваться не файлом листинга, а окном `_build` (расположено внизу рабочей области), где выводятся все сообщения об ошибках. При этом если дважды щёлкнуть мышью по сообщению об ошибке в окне `_build`, то в окне текстового редактора будет выделена строка программы, где была обнаружена данная ошибка.

Трансляция программного модуля и получение загрузочного файла в интегрированной среде программирования производится нажатием кнопки `_Build target`, как это показано на рисунке 43.

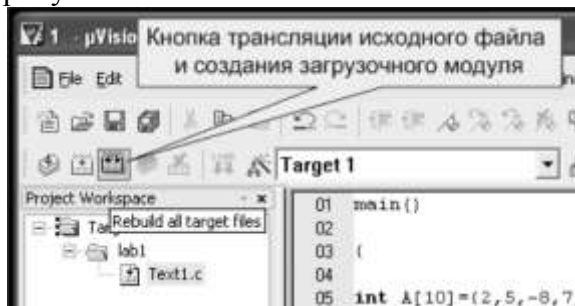


Рисунок 43 – Трансляция программного модуля проекта при помощи кнопки `_Build target`

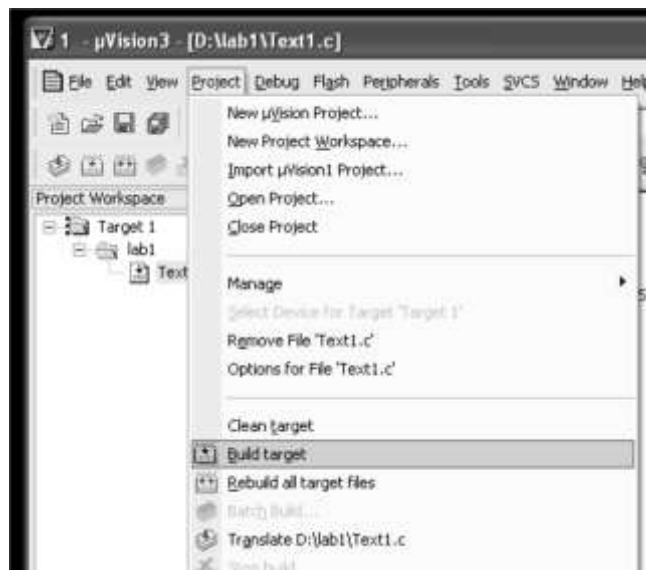


Рисунок 44 – Трансляция программного модуля при помощи главного меню

Ещё один способ трансляции программного проекта в интегрированной среде программирования, это воспользоваться главным меню, как это показано на рисунке 44, но удобнее всего нажать на клавиатуре клавишу `—F7`.

Если же необходимо оттранслировать все программные модули вне зависимости имеются объектные модули или нет, и получить загрузочный файл, то нажимается кнопка `_Rebuild all target files` или выбирается соответствующее меню.

6 Указания по отладке программ во встроенном отладчике программ

6.1 Способы отладки программ

Отладка программ заключается в проверки правильности работы программы и аппаратуры. Программа, не содержащая синтаксических ошибок, тем не менее, может содержать логические ошибки, не позволяющие программе выполнять заложенные в ней функции. Логические ошибки могут быть связаны с алгоритмом программы или с неправильным пониманием работы аппаратуры, подключённой к портам микроконтроллера.

Встроенный отладчик позволяет отладить те участки кода программы, которые не зависят от работы аппаратуры, не входящей в состав микросхемы микроконтроллера. Для отладки программ обычно применяют три способа:

1. Пошаговая отладка программ с заходом в подпрограммы;
2. Пошаговая отладка программ с выполнением подпрограммы как одного оператора;
3. Выполнение программы до точки останова.

Пошаговая отладка программ заключается в том, что выполняется один оператор программы и, затем контролируются те переменные, на которые должен был воздействовать данный оператор.

Если в программе имеются уже отлаженные подпрограммы, то подпрограмму можно рассматривать, как один оператор программы и воспользоваться вторым способом отладки программ.

6.2 Использование встроенного отладчика программ

Вызов встроенного отладчика удобнее всего осуществить, нажав на кнопку отладчика на панели инструментов `_file` как показано на рисунке 45 или воспользоваться быстрой кнопкой `—Ctrl+F5`

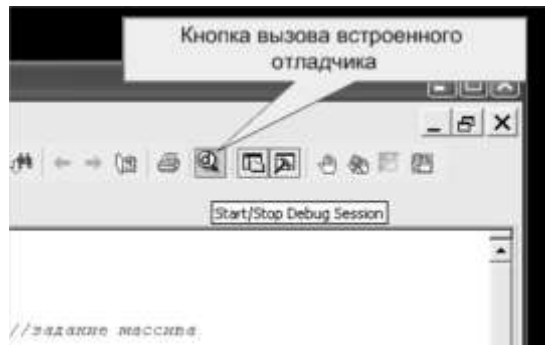


Рисунок 45 – Вызов встроенного отладчика с использованием кнопки на панели —file

После этого внешний вид интегрированной среды программирования принимает вид, показанный на рисунке 47. В верхней части программы появляется дополнительная панель инструментов отладчика программ (рисунок 46). В нижней части программы появляется окно просмотра памяти контроллера и окно контроля переменных —Watch.

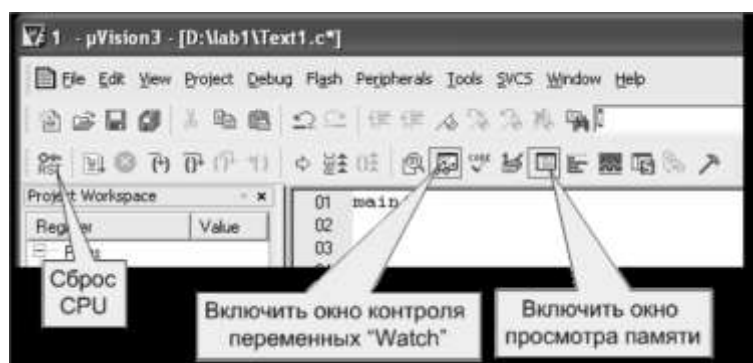


Рисунок 46 – Дополнительная панель инструментов отладчика программ.

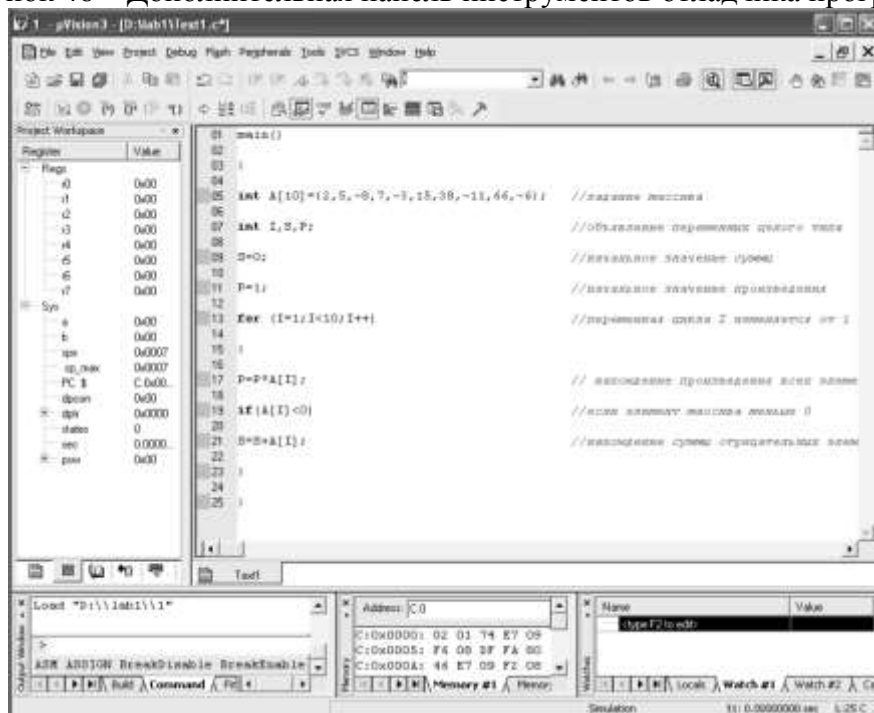


Рисунок 47 – Внешний вид интегрированной среды программирования в режиме отладки программ

Окно просмотра памяти контроллера можно настроить на просмотр памяти программ или памяти данных, введя в диалоговое окно —адрес| ключ, двоеточие и адрес начальной ячейки памяти.

Например:

- d:0 – просмотреть память данных начиная с нулевой ячейки;
- c:0 – просмотреть память программ начиная с нулевой ячейки;
- x:0 – просмотреть внешнюю память данных начиная с нулевой ячейки.

При использовании встроенного отладчика программ для контроля переменных можно воспользоваться окном —Watch. В большинстве случаев это намного выгоднее, чем использовать просмотр памяти данных. Переменные в этом окне отображаются в том формате, в котором они были объявлены в программе. Для добавления переменной в окно —Watch достаточно щёлкнуть правой кнопкой мыши по переменной в тексте программы в окне отладчика программ, как это показано на рисунке 48.

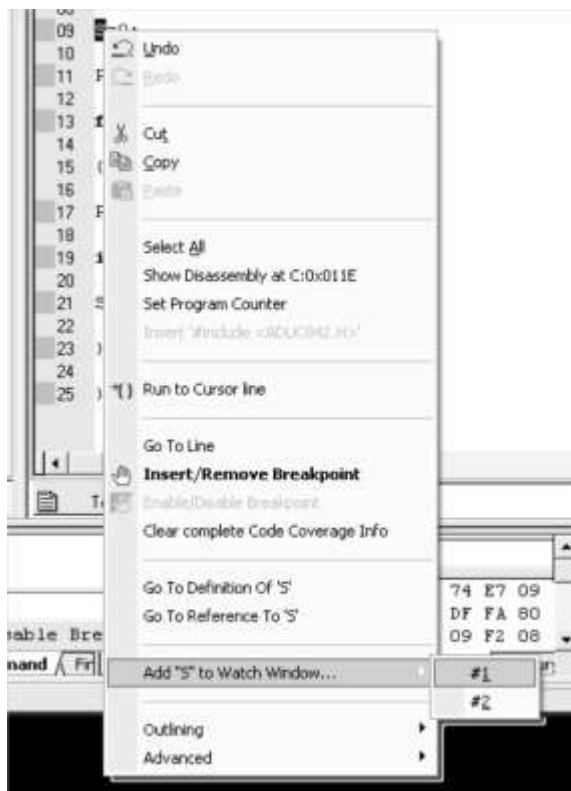


Рисунок 48– Добавления переменной в окно просмотра —Watch

Окно просмотра переменных содержит две закладки —Watch #1 и —Watch #2. Это позволяет группировать переменные, по какому либо признаку, например по отлаживаемым подпрограммам. При добавлении переменной Вы выбираете номер окна просмотра окна переменных.

Кроме просмотра глобальных переменных, которые существуют на протяжении всей программы, окно просмотра переменных содержит закладку —locals!. Эта закладка позволяет отслеживать локальные переменные, которые существуют только внутри подпрограммы. Вводить имена локальных переменных в эту закладку не нужно. Они появляются в этой закладке автоматически, как только Вы попадаете в подпрограмму, в которой используются локальные переменные.

При отладке программ на языке программирования ассемблер очень важно контролировать содержимое внутренних регистров микроконтроллера. Это позволяет сделать закладка Regs в окне менеджера проектов, показанная на рисунке 22 (левая рабочая область). В этом окне можно проконтролировать содержимое регистров текущего банка, указателя стека и программного счётчика, содержимое аккумуляторов A и B, а также


состояние рабочих флагов микроконтроллера в регистре PSW.


Один оператор программы может быть выполнен нажатием кнопки F11. Если вызов подпрограммы рассматривается как один оператор, то пошаговая отладка программы осуществляется нажатием кнопки F10.

Использование точек останова позволяет пропускать уже отлаженную часть программы. Для того, чтобы установить точку останова, можно воспользоваться кнопкой (Insert/remove Breakpoint) на панели файлов или воспользоваться главным или всплывающим меню. Перед тем как нажать на кнопку установки точки останова, необходимо установить курсор на строку исходного текста программы, где необходимо остановить выполнение программы.

Точка останова устанавливается в местах, где необходимо проверить содержимое переменных или просто проконтролировать, передаётся ли управление данному оператору.

После того, как установлены все необходимые точки останова осуществляется выполнение программы в свободнобегущем режиме. Для этого можно воспользоваться кнопкой (Run) или нажать на кнопку F5 на клавиатуре.

Может возникнуть ситуация, что программа не передаёт управление ни одному из операторов, на которых установлены точки останова. В этом случае для прекращения выполнения программы следует воспользоваться кнопкой  (Halt) или нажать на кнопку —Esc на клавиатуре.

Точка останова может быть использована многократно. Иногда же возникает необходимость однократно пропустить часть операторов. В этом случае можно воспользоваться кнопкой выполнения программы до курсора  (Run to Cursor line). При нажатии на эту кнопку программа будет выполняться до тех пор, пока управление не будет передано оператору, на котором находится курсор. Как только это произойдет, выполнение программы будет остановлено, и можно будет проконтролировать переменные и продолжить выполнение программы в пошаговом или свободnobегущем режиме.

Порядок выполнения работы:




1. Войдите в интегрированную среду программирования
2. Создайте новый файл исходного текста программы. Имя файла может быть, например, L1.c (расширение *.c обязательно).
Текст программы:


```
main()
{
//задание массива int A[10]={2,5,-8,7,-3,15,38,-11,66,-6};
//объявление переменных целого типа
int I,S,P;
//начальное значение суммы
S=0;
//начальное значение произведения
P=1;
//переменная цикла I изменяется от 1 до 10 с шагом 1
for (I=1;I<10;I++)
{
// нахождение произведения всех элементов массива
P=P*A[I];
//если элемент массива меньше 0
if(A[I]<0)
S=S+A[I];
//нахождение суммы отрицательных элементов массива
}}
```

Эта программа находит сумму отрицательных элементов массива A[10]. После выполнения программы результат (сумма) будет находиться в ячейке памяти S. 3. Создайте проект с именем LAB1.

4. Добавьте в проект файл с программой.
5. Оттранслируйте программный проект.
6. Убедитесь, что при трансляции программного модуля не обнаружены синтаксические ошибки.
7. Убедитесь, что в директории проекта созданы загрузочный файл с расширением .lst и загрузочный hex-файл с расширением .hex.
8. Выполните пошаговую отладку программы с использованием кнопки F11. На каждом шаге выполнения программы запишите значения используемых переменных программы: A[i] и S.

Ход работы:

1. Включите ЭВМ, и вызовите интегрированную среду программирования, щелкнув мышью по значку ...
2. Создайте новый файл исходного текста программы.
3. Введите заданный исходный текст программы, используя клавиатуру.
4. Откройте диалоговое окно сохранения файла, щелкнув мышью по значку .
5. Создайте новую папку с именем LAB1. Для этого щелкните мышью по значку .
6. Введите имя файла, например L1.c (расширение файла должно быть обязательно *.c), и нажмите на кнопку клавиатуры —Enter.

7. Создайте новый проект. Для этого выберите подменю —New project из меню —project. Выберите папку размещения нового проекта – LAB1. Укажите имя проекта – L1.
8. Теперь можно подключать к программному проекту файл с исходным текстом программы. Для этого можно щёлкнуть правой кнопкой мыши по значку группы файлов в окне менеджера проектов, как это показано на рисунке 40, и выбрать опцию добавления файлов к программному проекту.
9. Оттранслируйте проект, нажав кнопку —F7 (Rebuild all target files). Если есть ошибки, исправьте их в текстовом редакторе среды Keil.
10. При отсутствии ошибок убедитесь, что в директории проекта появился загрузочный файл с расширением .hex.
11. Для вызова отладчика щелкните мышью значок его запуска .
12. Выполните пошаговую отладку программы с использованием кнопки —F11. На каждом шаге выполнения программы запишите значения используемых переменных программы A[I] и S. Если при вызове отладчика в нижней части экрана отсутствует окно просмотра переменных, его можно включить, выбрав в меню —View команду —Watch & call stack window, либо воспользоваться иконками, показанными на рисунке 46. Для добавления переменной в окно —Watch достаточно щёлкнуть правой кнопкой мыши по нужной переменной в окне программы и в появившемся окне выбрать ADD to Watch window.

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. результаты работы;
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие № 6

Организация ввода-вывода информации через параллельные порты МК ADuC842

Цель работы: разработка программы управления светодиодами с использованием параллельного порта МК

Выполнив работу, Вы будете:

уметь:

- формирование умений применять стандартные алгоритмы в соответствующих областях;
- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять современные компиляторы, отладчики и оптимизаторы программного кода;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением.
2. Мультимедиа проектор.

3. Лабораторные стенды «LESO1»

Подготовка к работе

Изучить теоретический материал по теме (см. Практическую работу №2)

Работа с регистрами специальных функций

Все программно доступные регистры управления, конфигурации и данных микроконтроллера включены в область регистров специальных функций (Special Function Register — SFR). Эта область формально занимает старшие 128 байт внутренней памяти данных, но обращение должно осуществляться по определенным адресам ячеек или отдельным битам. Регистры SFR реализуют интерфейс между микропроцессорным ядром и внутренней периферией микроконтроллера. Все регистры специальных функций имеют как символические имена (мнемоники), так и адреса в качестве ячеек внутренней памяти. Информацию об SFR можно найти в документации на микроконтроллер, обычно указывается не только название и адрес регистра, но и значение, записанное в них после сброса контроллера.

Часть регистров, адреса которых заканчиваются на 0h или 8h, содержит прямо адресуемые биты, адреса этих битов находятся в диапазоне 80h – FFh. Кроме адресов эти биты имеют еще и имена, которые могут быть определены в языках программирования. Таким образом, к прямо адресуемым битам регистрам специальных функций можно обращаться по именам. Если в регистре биты не прямо адресуемые, то их имена служат только для удобства описания функционирования соответствующего блока микроконтроллера. Для обращения к определенному биту такого регистра должна производиться выборка байта целиком, а затем накладываться нужная маска.

Для работы с портами микроконтроллера на языке C51 введены специальные типы переменных – регистры специальных функций sfr. Синтаксис определения регистра выглядит следующим образом:

```
sfr name = address;
```

где name – имя регистра SFR (любой идентификатор);

address – адрес регистра SFR.

Пример объявления:

```
sfr P0 = 0x80; /* Порт-0, адрес 80h */  
sfr P1 = 0x90; /* Порт-1, адрес 90h */  
sfr P2 = 0xA0; /* Порт-2, адрес A0h */  
sfr P3 = 0xB0; /* Порт-3, адрес B0h */
```

После такого объявления в программе можно использовать регистр как обычную однобайтовую переменную.

Объявление отдельных битов в побитно адресуемых регистрах происходит с помощью ключевого слова sbit одним из следующих способов:

```
sbit name = sfr-name^bit-position;  
sbit name = sfr-address^bit-position;  
sbit name = sbit-address;
```

где:

name — имя бита в SFR;

sfr-name – имя заранее объявленного SFR;

bit-position – позиция бита в соответствующем регистре;

sfr-address – адрес регистра SFR;

sbit-address – непосредственный адрес бита.

Таким образом, второй бит порта P0 может быть записан одним из способов:

```
sbit p0_2 = 0x82; // непосредственный адрес  
sbit p0_2 = 0x80^2; // используя адрес SFR  
sbit p0_2 = P0^2; // заранее объявленный P0
```

Для определения адресов регистров и отдельных разрядов в них следует обратиться к рисунку 5.

Объявление регистров специальных функций следует делать перед основной подпрограммой. SFR не может быть объявлен внутри функции.

Работа загрузчика nwFlash

Для загрузки исполняемого кода во внутреннюю память микропроцессора и взаимодействия лабораторного стенда с ПК разработана программа nwFlash. Программа nwFlash позволяет:

- производить поиск подключенных к компьютеру по USB интерфейсу лабораторных стендов;
- активировать соединение с одним из найденных стендов;
- выполнять сброс микроконтроллера (Reset);
- загружать во flash – память микроконтроллера пользовательскую программу;
- принимать и отправлять данные в текстовом и шестнадцатеричном виде по интерфейсу UART (режим терминала).

Интерфейс nwFlash состоит из трех элементов: главного меню, окна терминала и окна состояния.

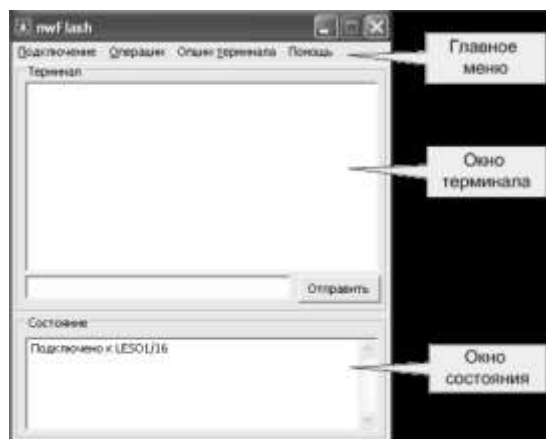


Рисунок 49 – Интерфейс загрузчика nwFlash

Главное меню позволяет производить операции со стендом, а также настраивать параметры терминала. Окно терминала служит для отображения данных, посылаемых микроконтроллером по интерфейсу UART, а также для отправки пользовательских данных от компьютера микроконтроллеру. В окне состояния отображаются результаты всех проведенных операций для контроля.

Для работы с программой nwFlash следует запустить программу. При нажатии на пункт главного меню "Подключение" программа выполнит поиск подключенных стендов и отобразит их названия в раскрывшемся меню. Если вы забыли подключить стенд, то появится надпись "нет подключенных стендов", в этом случае подключите стенд и снова раскройте меню "Подключение".

После выбора стенда из меню "Подключение". В окне состояния должна появиться надпись "Подключено к "имя_стенда". После этого становится доступным пункт меню "Операции", где можно:

- выполнить сброс микроконтроллера. На стенде начнёт выполняться программа, записанная в микроконтроллер в последний раз;
- стереть flash-память микропроцессора;
- загрузить исполняемый *.hex файл в память микроконтроллера.

В появившемся окне необходимо указать путь к *.hex файлу.



Рисунок 50 – Интерфейс загрузчика nWFlash. Операция.

После выполнения работы со стендом, выберите пункт "Отключиться" в меню "Подключение", затем закройте программу.

Порядок выполнения работы

1. Вывод информации через параллельный порт

1. По принципиальной схеме установите, к каким портам микроконтроллера подключены светодиоды.
2. По таблице регистров специальных функций (SFR) определите адреса регистров требуемых портов.
3. Войдите в интегрированную среду программирования Keil-C.
4. Создайте файл проекта.
5. Введите текст программы в соответствии с заданием:
6. Каждому студенту требуется зажечь светодиоды, соответствующие номеру своего варианта в бинарном виде.
7. Оттранслируйте программу, и исправьте синтаксические ошибки.
8. Загрузите полученный *.hex файл в лабораторный стенд LESO1.
9. Убедитесь, что на лабораторном стенде LESO1 загораются требуемые светодиоды.
10. Покажите результат преподавателю.

2. Ввод информации через параллельный порт (дополнительно)

1. По принципиальной схеме установите, к какому порту микроконтроллера подключена кнопка S2.
2. Определите адрес порта, к которому подключена кнопка.
3. Измените исходный текст программы таким образом, чтобы требуемые светодиоды загорались только по нажатию кнопки.
4. Оттранслируйте программу, и исправьте синтаксические ошибки.
5. Загрузите полученный *.hex файл в лабораторный стенд LESO1.
6. Убедитесь, что при нажатии на кнопку загорается светодиоды, соответствующие вашему варианту.
7. Покажите результат преподавателю.

1. Примеры программ приведены в файлах 1.с и 2.с (см. Приложение)

Внесите изменения (по вариантам) в программу 1.с для включения заданных светодиодов, используя параллельный порт МК. Включенные светодиоды помечены X, отключенные – 0.

№ варианта	№ светодиода			
	1	2	3	4
1	X	0	0	0
2	0	X	0	0
3	0	0	X	0
4	0	0	0	X
5	X	0	0	X
6	X	0	X	0

Внесите изменения в программу 2.с для включения заданных светодиодов по нажатию на кнопку, используя параллельный порт МК. Включенные светодиоды помечены X, отключенные – 0.

№ варианта	№ светодиода				№ светодиода			
	1	2	3	4	1	2	3	4
1	X	0	0	0	0	0	0	X
2	0	X	0	0	0	0	X	0
3	0	0	X	0	0	X	0	0
4	0	0	0	X	X	0	0	0
5	X	0	0	X	0	X	X	0
6	X	0	X	0	0	X	0	X

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Графическую схему алгоритма программы.
3. Исходный текст программы.
4. Содержимое файла листинга программного проекта.
5. Выводы по выполненной лабораторной работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №7.

Разработка программы управления клавиатурой матричного типа

Цель работы: разработать и отладить программу управления матричной клавиатурой с помощью МК

Выполнив работу, Вы будете:

уметь:

- формирование умений применять стандартные алгоритмы в соответствующих областях;
- применять выбранные языки программирования для написания программного кода;
 - использовать выбранную среду программирования;
 - применять нормативные документы, определяющие требования к оформлению программного кода;
 - использовать возможности имеющейся технической и/или программной архитектуры;
 - применять современные компиляторы, отладчики и оптимизаторы программного кода;
 - распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
 - анализировать задачу и/или проблему и выделять её составные части;
 - грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
 - понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением.
2. Мультимедиа проектор.
3. Лабораторные стенды «LESO1»

Подготовка к работе:

изучить необходимый теоретический материал по теме (см. Практическая работа №3)

Рекомендации к составлению программы

Программа для микроконтроллера жестко зависит от принципиальной схемы разрабатываемого устройства. Невозможно написать программу для микроконтроллерного устройства не имея перед глазами его схемы. Поэтому, перед началом работы по принципиальной схеме учебного стенда LESO1 следует изучить способ подключения клавиатуры и светодиодов к микроконтроллеру: определить, к каким портам подключены светодиоды, столбцы и строки клавиатуры. Затем по таблице SFR нужно узнать адреса регистров задействованных портов ввода-вывода.

Программа, управляющая микроконтроллером, запускается при включении питания устройства и не завершает свою работу, пока не будет выключено питание. Поэтому в

программе обязательно должен быть организован бесконечный цикл. В теле цикла должен производиться опрос клавиатуры, анализ полученных данных и вывод результата на светодиод. Опрос клавиатуры заключается в последовательном сканировании каждого столбца, для этого на соответствующую линию порта вывода подается логический ноль (эквивалент общего провода), на остальных столбцах должен быть высокий уровень, после чего с порта ввода, к которому подключены строки, считывается код. Если считаны все единицы, то ни одна из клавиш не нажата, в противном случае код содержит информацию о нажатых клавишах. Стоит заметить, что считанный код содержит не только номер замкнутого контакта, но и информацию о нажатии нескольких кнопок одновременно, поэтому лучше хранить в памяти контроллера непосредственно считанный код, а не готовый номер кнопки. Для хранения считанного кода следует ввести специальную переменную.

При написании программы нужно помнить об особенности параллельного порта P1 в микроконтроллере ADuC842. Этот порт по умолчанию настроен на ввод аналоговых сигналов (функция АЦП). Для того чтобы перевести порт в режим цифрового входа, в соответствующий бит порта необходимо записать логический ноль. Сделать это нужно один раз при инициализации микроконтроллера. Порт не имеет внутреннего усиливающего транзистора, и потому при вводе дискретной информации через него не требуется записывать в разряды логическую единицу.

Порядок выполнения работы:

1. По принципиальной схеме установите, к каким портам микроконтроллера подключены светодиоды, а также столбцы и строки клавиатуры.
2. По таблице регистров специальных функций (SFR) определите адреса регистров требуемых портов.
3. Войдите в интегрированную среду программирования Keil-C.
4. Создайте и настройте должным образом проект.
5. Введите текст программы в соответствии с заданием: При нажатии на кнопку, согласно варианту, загорается комбинация светодиодов, соответствующая в бинарном виде номеру кнопки; при отпускании кнопки, светодиоды должны погаснуть.
6. Оттранслируйте программу, и исправьте синтаксические ошибки.
7. Загрузите полученный *.hex файл в лабораторный стенд LESO1.
8. Убедитесь, что программа функционирует должным образом.

Пример программы приведен в файле 3.с (см. Приложение)

Внесите изменения в программу 3.с для включения в работу только определенных клавиш клавиатуры, остальные - отключены.

№ варианта	Включенные клавиши		
	1	2	3
1	1	2	3
2	4	5	6
3	7	8	9
4	1	4	7
5	2	5	8
6	3	6	9

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Графическую схему алгоритма программы.
3. Исходный текст программы.
4. Содержимое файла листинга программного проекта.
5. Выводы по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №8.**Разработка программы управления символьным ЖКИ**

Цель работы: разработка и отладка программы вывода информации на ЖКИ с помощью МК

Выполнив работу, Вы будете:

уметь:

формирование умений применять стандартные алгоритмы в соответствующих областях;

- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять современные компиляторы, отладчики и оптимизаторы программного кода;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением.
2. Мультимедиа проектор.
3. Лабораторные стенды «LESO1»

Задание: изучить необходимый теоретический материал по теме (см. Практическая работа №4)

Перед началом работы требуется произвести инициализацию ЖКИ согласно алгоритму, показанному на рисунке 54.

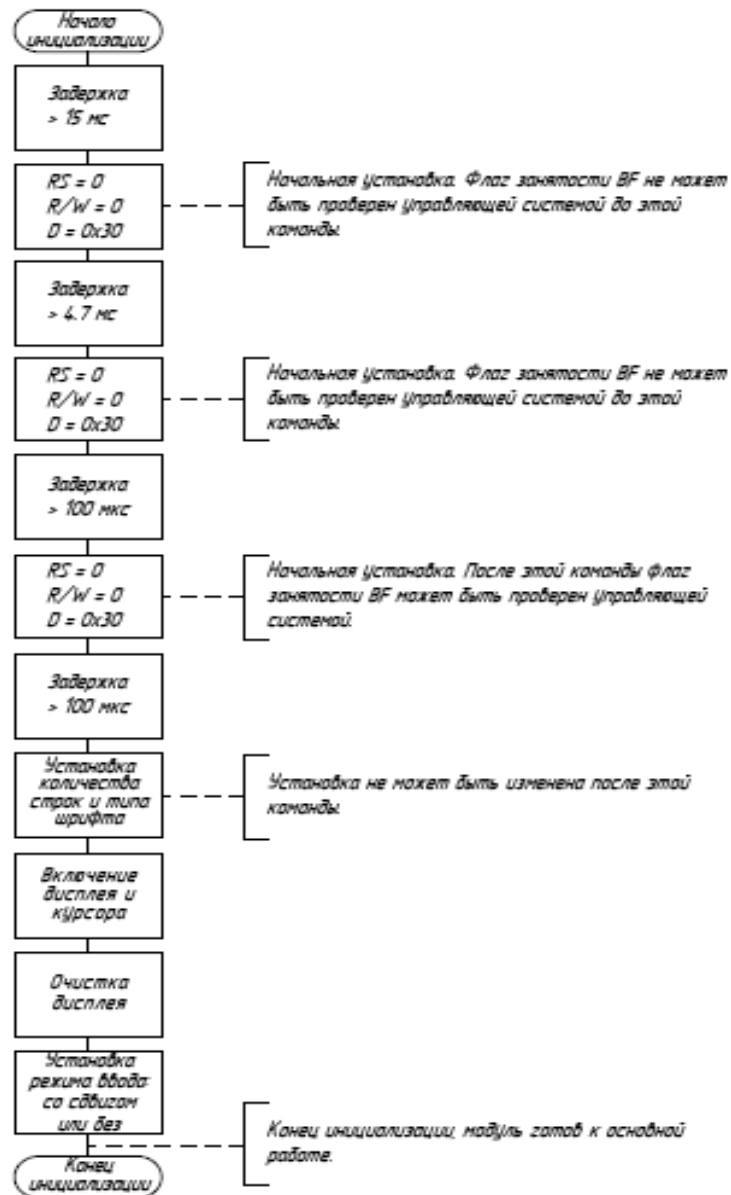


Рисунок 54 – Алгоритм инициализации ЖКИ

		Старшая часть байта (04 - 07)																			
		0h	1h	2h	3h	4h	5h	6h	7h	8h	9h	Ah	Bh	Ch	Dh	Eh	Fh				
Младшая часть байта (D0 - D3)	0h				0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
	1h			!	1	A	Q	a	n						Г	Я	Ш	Ц	Щ	Ъ	Ы
	2h			"	2	B	R	б	г						ё	ё	ь	ш	щ	ъ	ы
	3h			#	3	C	S	с	с						№	В	У	У	У	У	У
	4h			\$	4	D	T	t	t						Э	Г	Ь	Ъ	Ъ	Ъ	Ъ
	5h			%	5	E	U	u	u						Н	ё	а	х	ц	ц	ц
	6h			&	6	F	V	v	v						Ж	ж	ж	ш	ш	ш	ш
	7h			^	7	G	W	w	w						Л	э	я	л	л	л	л
	8h			^	8	H	X	x	x						П	И	а	и	и	и	и
	9h			>	9	I	Y	y	y						У	а	а	т	т	т	т
	Ah			*	*	J	Z	z	z						Ф	к	а	а	а	а	а
	Bh			+	+	K	[k	[Ч	а	а	а	а	а	а
	Ch			,	<	L]	l]						Ш	н	н	н	н	н	н
	Dh			-	=	M	^	m	^						Ъ	н	а	а	а	а	а
	Eh			.	>	N	^	n	^						Ы	п	а	а	а	а	а
	Fh			/	?/	O	_	o	_						Э	т	е	а	а	а	а

Рисунок 55 – Таблица символов знакогенератора

2 Рекомендации по программному управлению ЖКИ

Программу для работы с ЖКИ следует организовать в виде функций, выполняющих определенные действия, причем более сложные функции могут включать в себя простейшие. Простейшими могут быть такие подпрограммы, как функция, отправляющая команду контроллеру дисплея; функция, устанавливающая счетчик адреса; или функция, записывающая данные в DDRAM. В любом случае, общий алгоритм передачи информации контроллеру не изменится. Руководствуясь диаграммой передачи информации (рисунок 24), определим последовательность действий при передаче информации в ЖКИ следующим образом: устанавливаем требуемое значение **RS**, на линию **R/W** подаем логический ноль, затем на линию **E** выводим логическую единицу, после чего подаем на шину **D** значение передаваемого байта. Контроллер ЖКИ считает этот байт и состояние управляющих линий (**RS**, **R/W**) только после подачи на линию **E** логического нуля. При этом, если временные задержки, указанные на диаграмме, меньше длительности машинного цикла, то ими можно пренебречь.

Код программы, реализующей запись в память ЖКИ байта данных, показан ниже:

```
RS=1;           // выбираем команды или данные
RW=0;          // выбираем направление передачи
E=1;
```

```
Data=symbol;           // выводим байт данных на шину D
E=0;                   // переводим сигнал на линии E из 1 в 0
delay ();              /* ждем, пока контроллер выполняет внутренние операции*/
```

В приведенном участке программы подразумевается, что переменные **RS**, **RW** и **E** объявлены как **sbit**, а переменная **Data** – как **sfr**. Аналогично будет происходить передача любой команды контроллеру ЖКИ.

При реализации чтения информации из контроллера необходимо пользоваться диаграммой, приведенной на рисунке 25. Следует помнить, что для того, чтобы ввести информацию с параллельного порта, в него предварительно должны быть записаны логические единицы.

Для того, чтобы не загромождать основную программу алгоритм инициализации (рисунок 54) можно реализовать в виде отдельной подпрограммы. Временные задержки, указанные в алгоритме, следует задавать с помощью таймеров, как это делалось в лабораторной работе «Изучение таймеров микроконтроллера».

Порядок выполнения работы

1 Вывод символа на ЖКИ

1. Разработайте алгоритм программы, выводящей на экран ЖКИ ваше имя в заданной строке. Режим работы ЖКИ и номер строки определяется согласно варианту задания (таблица 10).
2. По принципиальной схеме учебного стенда LESO1 определите, к каким выводам микроконтроллера ADuC842 подключен ЖКИ. По таблице SFR определите адреса используемых портов ввода-вывода.
3. Разработайте и введите текст программы в соответствии с созданным алгоритмом.
4. Оттранспируйте программу, и исправьте синтаксические ошибки.
5. Загрузите полученный *.hex файл в лабораторный стенд LESO1.
6. Убедитесь, что на экране дисплея в заданной позиции появился требуемый символ.

2 Управление ЖКИ через последовательный порт персонального компьютера (дополнительно)

1. Измените программу таким образом, чтобы на экране ЖКИ выводилась информация, переданная с персонального компьютера через UART. Передача команды осуществляется через терминал pwFlash. Выбор источника синхронизации и скорости передачи данных осуществляется по усмотрению студента.
2. Загрузите полученный *.hex файл в лабораторный стенд LESO1.
3. Через терминал pwFlash передайте коды символов, убедитесь, что соответствующие символы выводятся на экране индикатора.

Пример программы приведен в файле 6.c (см. Приложение)

Внесите изменения в программу 6.c для отображения на ЖКИ Вашего имени.

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Принципиальную схему подключения ЖКИ к управляющему микроконтроллеру.
3. Структурную схему ЖКИ.
4. Диаграммы передачи данных по параллельному интерфейсу.
5. Расчет параметров таймера.
6. Графическую схему алгоритма работы программы.
7. Исходный текст программы.
8. Содержимое файла листинга программного проекта.
9. Выводы по выполненной лабораторной работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №9.

Организация ввода-вывода информации через параллельные порты МК Atmega 8535

Цель работы: изучить процесс обмена информацией через порты ввода/вывода МК.

Выполнив работу, Вы будете:

уметь:

- формирование умений применять стандартные алгоритмы в соответствующих областях;
- применять выбранные языки программирования для написания программного кода;
 - использовать выбранную среду программирования;
 - применять нормативные документы, определяющие требования к оформлению программного кода;
 - использовать возможности имеющейся технической и/или программной архитектуры;
 - применять современные компиляторы, отладчики и оптимизаторы программного кода;
 - распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
 - анализировать задачу и/или проблему и выделять её составные части;
 - грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
 - понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

Стенд «Микроконтроллер», ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

Пример 1 (текст программы см. ПРИЛОЖЕНИЕ).

Написать программу, осуществляющую сложение двух младших и двух старших бит порта C с выводом результата на порт D.

```
-----  
;Программа для сложения двух двухбитных чисел A (биты PC7:PC6) и B  
(биты PC1:PC0) с последующим выводом результата на PORTD  
;Входы: PINC7:PINC6 и PINC1:PINC0  
;Выходы: PORTD  
.include "m8535def.inc" ;Подключение библиотеки ATmega8535  
.cseg ;Начало сегмента кода  
.org 0  
reset: ;Инициализация портов ввода/вывода  
 ldi r16,0xFF ;Установка PORTD на вывод информации: r16←0xFF  
 out DDRD,r16 ;PORTD←r16  
 clr r16 ;Установка PORTC на ввод информации: r16←0  
 out DDRC,r16 ; PORTD←r16  
main:  
 in r16,PINC ;ввод данных из порта C в регистр: r16←PINC  
 mov r17,r16 ;Копирование результата r17←r16  
 andi r16,0x03 ;наложение маски: обнуление всех бит, кроме 0 и 1  
 andi r17,0xC0 ;наложение маски: Обнуление всех бит, кроме 6 и 7  
 lsr r17 ;Логический сдвиг r17 вправо на 6 бит  
 lsr r17  
 lsr r17  
 lsr r17  
 lsr r17  
 add r16,r17 ;Сложение r16 и r17: r16←r16+r17  
 out PORTD,r16 ;Вывод результата в порт D: PORTD←r16  
 rjmp main ;возврат на main  
-----
```

Рассмотрим программу более подробно.

1. Подключение стандартной библиотеки контроллера.

Строка `.include "m8535def.inc"` подключает библиотеку контроллера Atmega8535, которая содержит всю необходимую информацию о контроллере, а именно список регистров с их адресами, объемы памяти, список периферийных устройств и другие особенности.

2. Выбор сегмента кода и адреса начала написания программы.

Строки `.cseg` и `.org 0` устанавливаются начало сегмента кода на нулевой адрес.

3. Инициализация портов ввода/вывода.

Рассмотрим следующие строки программы:

```
ldi r16, 0xFF
out DDRD, r16
clr r16
out DDRC, r16
```

Эти команды инициализируют порт С на ввод данных, а порт D - на вывод. Поскольку порт D необходимо инициализировать на вывод, в регистр DDRD требуется записать 0xFF (1111 1111 в двоичном коде). Для этого сначала в регистр общего назначения r16 записывается число 0xFF (инструкция «ldi»), а затем содержимое этого регистра переписывается в регистр DDRD (инструкция «out»). Аналогичная операция производится с портом С.

4. Ввод данных и запись в регистры общего назначения. В главной программе согласно заданию необходимо, опросить состояние регистра PINC и выделить два числа: в первом хранятся два младших бита, во втором - два старших бита. Для этого вначале выполняется ввод, всего регистра PINC в регистры r16 и r17, (инструкции «in» и «mov»). Далее для корректного считывания чисел необходимо наложить, так называемые, маски на оба числа: в первом числе обнулить все биты кроме двух младших, во втором - все биты кроме двух старших. Это осуществляется с помощью команды «побитовое умножение» (инструкция «andi»): содержимое регистра r16 перемножается с константой 0x03 (0000 0011), а регистра r17 — с константой 0xC0 (1100 0000). Эти операции выполняются строками:

```
in r16, PINC
mov r17, r16
andi r16, 0x03
andir17, 0xC0
```

5. Приведение переменных к одному весовому коэффициенту. Полученные значения переменных в регистрах r16 и r17 имеют разные весовые коэффициенты. Необходимо преобразовать число XX00 0000, содержащееся в r17, в число вида 0000 00XX. Для этого в программе используется 6 раз одна и та же инструкция логического сдвига вправо «lsr» содержимого регистра r17 на 1 бит.

6. После выполнения операции сдвига r17 можно произвести сложение r16 и r17 и вывод результата сложения на PORTD:

```
add r16, r17
out PORTD, r16
```

7. В конце программы ее необходимо «зациклить», т.е. вернуться на метку «main» для ее повторного выполнения:

```
rjmp main
```

Пример 2 (текст программы см. ПРИЛОЖЕНИЕ).

Реализовать на микроконтроллере расчет логического уравнения:

$$D = A \cdot (\bar{A} \oplus B + C)$$

где A, B, C логические сигналы, поступающие на входы например, PA0 (0 бит порта A), PA1 (1 бит порта A), PA2 (2 бит порта A), а D результат решения логического уравнения, который выводится на 0 бит порта D.

```

;-----
; Программа для решения логического уравнения D=A*(NA⊕B+C)
; Входы:  A=PINAO
;         B=PINAI
;         C=PINAI2
; Выход:  D=PORTDO

.include "m8535def.inc" ; Подключение библиотеки Atmega8535
.def A=r20              ; Объявление переменных и присвоение их имен
.def B=r21              ; регистрам общего назначения r20...r23
.def C=r22
.def D=r23

.cseg                  ; начало сегмента кода
.org 0

reset:                ; инициализация портов ввода/вывода
    ldi r16,0x01
    out DDRD,r16      ; PORTDO - на вывод информации
    clr r16
    out DDRA,r16      ; PINAO ..PINAI2 - на ввод информации

main:
    in r16,PINA        ; Ввод значения PINA в POH r16
    mov A,r16          ; Копирование r16 в регистры A, B, C
    mov B,r16
    mov C,r16
    andi A,0x01        ; Выделение 0 бита числа A
    andi B,0x02        ; Выделение 1 бита числа B
    andi C,0x04        ; Выделение 2 бита числа C
    lsr B              ; Сдвиг B вправо на 1 бит
    lsr C
    lsr C              ; Сдвиг C вправо на 2 бита
    mov r16,A          ; Копирование A в POH r16
    com r16            ; Инверсия содержимого r16
    andi r16,0x01      ; Формирование числа NA
    eor r16,B          ; Расчет r16=NA⊕B
    or r16,C           ; Расчет r16=r16+C
    and r16,A          ; Расчет r16=r16*A
    out PORTD,r16     ; Вывод результата на PORTDO
    rjmp main         ; Возврат на main
;-----

```

Рассмотрим программу более подробно.

1. Инициализация контроллера и переменных. Вначале выполняются инициализация микроконтроллера и присвоение имен A, B, C, Регистрам общего назначения r20 ... r23. Для этого используется директива «defA=r20»

```

.includе "m8535def.inc"
.def A=r20
.def B=r21
.def C=r22
.def D=r23
.cseg .org 0

```

2. Инициализация портов ввода/вывода. В данной программе производится аналогично предыдущему примеру с следующими отличиями: порт D инициализирован на вывод только 1 младшего бита, порт A - полностью на ввод данных.

3. Ввод данных. В основной программе сначала в регистр r16 вводится значение порта A, затем это значение переписывается в регистры A, B и C, и далее «наложение» маски на эти регистры: в этих регистрах выполняется выделение только отдельных битов: в A - 0 бит, B - 1 бит, C - 2 бит:

```

in r16,PINA
mov A,r16
mov B,r16
mov C,r16
andi A,0x01
andi B,0x02
andi C,0x04

```

4. Выравнивание весовых коэффициентов переменных. Биты в регистрах A, B, C имеют разные весовые коэффициенты, для их выравнивания выполняется смещение значений регистров B и C вправо на 1 и 2 бита соответственно.

5. Формирование инверсного значения переменной. Для формирования инверсного значения регистра А это значение копируется в регистр r16, далее оно инвертируется (инструкция «com») и опять выделяется только младший бит:

```
mov r16,A
com r16
and r16,0x01
```

1. Вычисление логического выражения и вывод данных. Логические операции выполняются согласно уравнению $D=A \cdot (A \cdot B + C)$, далее результат выводится на индикацию в порт Ди программа закидывается:

```
and r16,B
orr r16,C
and r16,A
out PORTD,r16
rjmp main
```

Задание на выполнение

1. На базе примера №1 составить программу для вычисления:

- суммы двух 3-разрядных чисел;
- суммы двух 8-разрядных чисел;
- разности двух 2-разрядных чисел;
- разности двух 4-разрядных чисел.

2. Разработать логическую систему автоматизации - составить программу для своего варианта по реализации заданного логического уравнения, ввести программу в МК и проверить ее работоспособность на контроллере.

1. $PC3 = \overline{(PA0 + PA1)} \cdot PA3$	11. $PD5 = PA1 + PA2 \oplus PD0$
2. $PD4 = PB1 \oplus PB2 + PB3 \cdot PB4$	12. $PD3 = PB7 + PB6 \cdot PB5$
3. $PD0 = PA1 + (PA2 \oplus PA3 \cdot PA4)$	13. $PC3 = \overline{(PA0 + PB0)} + PD7$
4. $PD7 = PA7 \oplus PA6 \cdot PA5$	14. $PC0 = PD7 \oplus PD1 + PD2$
5. $PC0 = \overline{(PA3 \cdot PA4)} + PA5$	15. $PA0 = PC0 \oplus PC1 \cdot PA7$
6. $PC7 = PD2 \oplus PD3 + PD4$	16. $PC1 = PC2 + PC3 \cdot PC4$
7. $PD1 = (PC0 \cdot PC1) \oplus (PB0 \cdot PB1)$	17. $PA0 = PC0 \cdot PC1 + PD0 \oplus PD1$
8. $PC1 = PB2 + PB3 \cdot PB4$	18. $PD7 = PA0 \cdot PB1 \oplus PC2$
9. $PA1 = PC0 \oplus PC1 + PC6 \oplus PC7$	19. $PC1 = \overline{PA0} \cdot PA1 + PA3$
10. $PD7 = PD0 \cdot PD1 + PD2$	20. $PA7 = PB0 \oplus PB2 \cdot PB1 + PB3$

3. Составить программу и проверить ее работоспособность в микроконтроллере, в которой число набранное в порт А изменяется (используя битовые операции) и выводится в порт С:

- увеличивается в 2 раза;
- уменьшается в 4 раза;
- выводится в обратном коде;
- выводится в дополнительном коде.

4. Составить программу и проверить ее работоспособность в микроконтроллере, в которой вводится два четырехразрядных числа А (бит PA0...PA3 и В (биты PB0...PB3), результат выводится в порт С и выполняется следующая операция;

- арифметическая сумма чисел;
- поразрядная логическая сумма;
- арифметическое произведение чисел;
- поразрядное логическое произведение;
- арифметическая разность

Контрольные вопросы

1. Сколько портов имеет микроконтроллер ATmega8535?
2. Какие регистры определяют режим работы порта? Поясните их назначение
3. Определите регистры работы порта С если известно, что 2 бита порта работают на ввод данных, остальные на вывод.
4. Для каких целей используется директивы '.def', '.cseg', '.org'?
5. Какие инструкции по выполнению логических операций вы знаете?
6. Как наложить маску на считываемое значение регистра состояния?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Выполненное задание
3. Ответы на контрольные вопросы
4. Вывод по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено

Лабораторное занятие №10.

Исследование работы регистра состояний SREG МК Atmega 8535

Цель работы: научиться применять биты регистра состояния при написании программ.

Выполнив работу, Вы будете:

уметь:

- формирование умений применять стандартные алгоритмы в соответствующих областях;
- применять выбранные языки программирования для написания программного кода;
 - использовать выбранную среду программирования;
 - применять нормативные документы, определяющие требования к оформлению программного кода;
 - использовать возможности имеющейся технической и/или программной архитектуры;
 - применять современные компиляторы, отладчики и оптимизаторы программного кода;
 - распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
 - анализировать задачу и/или проблему и выделять её составные части;
 - грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
 - понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

Стенд «Микроконтроллер», ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

Рассмотрим практическое использование флагов регистра состояния на примере программы вычитания двух чисел: первое число задается младшим полубайтом порта А и имеет формат:

<u>Бит</u>	<u>7</u>	<u>6</u>	<u>5</u>	<u>4</u>	<u>3</u>	<u>2</u>	<u>1</u>	<u>0</u>
<u>A=</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>PA3</u>	<u>PA2</u>	<u>PA1</u>	<u>PA0</u>

второе число задается старшим полубайтом порта А и имеет формат:

<u>Бит</u>	<u>7</u>	<u>6</u>	<u>5</u>	<u>4</u>	<u>3</u>	<u>2</u>	<u>1</u>	<u>0</u>
<u>B=</u>	<u>PA7</u>	<u>PA6</u>	<u>PA5</u>	<u>PA4</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>

Результат должен выводиться в следующем виде:

- младшие 4 разряда порта С - модуль разности (А-В);
- старший бит порта С - знак результата.

```

//-----
#include "m8535def.inc" ;подключение библиотеки ATmega8535
.cseg ;начало сегмента кода
.org 0
ldi r16,0xFF ;инициализация портов ввода/вывода
out PORTA,r16 ;порт А - на ввод информации
out DDRC,r16 ;порт С - на вывод информации
main:
in r16,PINA ;ввод данных PINA в POH r16
mov r17,r16 ;копирование r16 в r17
andi r16,0x0F ;выделение числа А
andi r17,0xF0 ;выделение числа В
clr r18 ;очистка POH r18
m1:
lsr r17 ;сдвиг r17 на 4 разряда вправо
inc r18
cpi r18,0x04
brne m1
sub r16,r17 ;выполнение вычитания r16-r17 (А-В)
brmi m2 ;если в результате установлен флаг N, то
;осуществляется переход на метку m2
rjmp m3 ;иначе - переход на метку m3
m2:
com r16 ;выделение модуля результата
;инверсия результата
inc r16 ;увеличение результата на +1
ori r16,0x80 ;прибавление к результату сигнала «знак»
m3:
out PORTC,r16 ;вывод результата на PORTC
rjmp main ;и возврат на main
//-----

```

Рассмотрим программу более подробно.

1. Инициализация портов ввода/вывода: порт А - на ввод, порт С - на вывод информации:

```

ldi r16,0xFF
out PORTA,r16
out DDRC,r16

```

2. В начале основного цикла производится опрос регистра PINAприведение чисел к одному виду:

- результат заносится в регистр r16 (**inr16, PINA**) и r17 (**movr17, r16**);
- затем в регистре r17 осуществляется сдвиг числа на 4 разряда вправо для того, чтобы преобразовать его из числа вида «XXXX0000» в число «0000 XXXX»:

```

main:
in r16,PINA
mov r17,r16
andi r16,0x0F
andi r17,0xF0
clr r18

```

```

m1:
lsr r17
inc r18
cpi r18,0x04
brne m1

```

3. Операция вычитания, после выполнения которой флаги в регистре SREGвыставляются определенным образом:

```

sub r16, r17

```

4. Для обработки результатов вычитания необходимо проверить флаг отрицательного значения N:

- если он отсутствует, то результат - положительное число и его сразу можно выводить на индикацию;

- если же флаг N установлен, то результат - отрицательное число, и перед выводом на индикацию-необходимо выделить его модуль:

```
brmim2  
rjmpm3
```

Команда brmim2 осуществляет проверку флага N. Если он установлен, осуществляется переход на метку m2, при переходе на которую осуществляется выделение модуля числа. Если флаг N не установлен, выполняется следующая за командой директива rjmpm3. По этой команде выполняется переход на метку m3, по которой осуществляется вывод результата на индикацию.

5. При переходе на метку m2 результат сначала инвертируется (comr16), а затем к нему прибавляется +1 (incr16). Таким образом, осуществляется выделение из дополнительного кода модуля результата вычитания:

m2:

```
comr16  
incr16  
orig16,0x80
```

m3:

```
out PORTC, r16  
rjmpmain
```

Командой orig16, 0x80 осуществляется установка сигнала «Знак» в старшем разряде регистра r16, который затем выдается на индикацию на PORTCoutPORTC,r16. После этого основной цикл программы замыкается.

Задание (одно на выбор)

1. Используя логические операции и биты регистра состояния реализовать на микроконтроллере схему:

- полусумматора;
- полного одноразрядного сумматора.

2. Используя биты регистра состояния вычислить модуль разности двух 5-ти разрядных чисел.

3. Используя биты регистра состояния реализовать компаратор: сравниваются два 4-х разрядных числа (порты A и B): если равны – включается 0 бит порта D, если первое число больше – 1 бит, если второе – 2 бит.

4. Используя бит хранения информации регистра состояния реализовать

- RS-триггер;
- D-триггер;
- T-триггер

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Выполненное задание с комментариями (работоспособность программы демонстрируется на стенде)
3. Вывод по работе (указать какие флаги были использованы и почему)

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено

Лабораторное занятие №11.

Разработка программы для организации программной задержки (с использованием стека)

Цель работы: научиться применять стек при организации программной задержки.

Выполнив работу, Вы будете:

уметь:

- формирование умений применять стандартные алгоритмы в соответствующих областях;
- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять современные компиляторы, отладчики и оптимизаторы программного кода;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

Стенд «Микроконтроллер», ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

Программная задержка

Одним из наиболее наглядных применений указателя стека является реализация программной задержки времени.

В микроконтроллере ATmega8535 присутствуют три таймера, с помощью которых можно достаточно просто реализовать любую временную задержку. Однако простейшую программную задержку необходимо уметь реализовывать без использования дополнительных аппаратных средств микроконтроллера.

Суть программной задержки состоит в том, чтобы заставить процесс выполнять циклически одно и то же действие, например, инкремент какого-то числа от нуля до максимума, с дальнейшим переходом при окончании выполнения этого действия к дальнейшему выполнению программы.

Ввиду того, что процессор производит операции с большой скоростью, для сколько-нибудь ощутимой задержки приходится производить операции большими числами.

Пусть нам задан некоторый элемент программы. Рассчитаем примерное время исполнения этого элемента (в комментариях прописано количество тактов процессора для выполнения данной инструкции):

```
//-----  
    clr r16      ; 1 такт процессора  
ml:   ; не выполняется, нет задержки  
    inc r16     ; 1 такт процессора  
    cpi r16,0x20 ; 1 такт процессора  
    brne met_1  ; 2 такта процессора, если условие выполняется  
                    ; (переход к метке ml) и 1 такт процессора,  
                    ; если условие не выполняется (выход из цикла)  
//-----
```

В программе значение регистра РОН r16 увеличивается на 1, пока значение в r16 не достигнет значения 0x20 (десятичное число 32). При этом постоянно идет возврат на метку ml, и только после того, как значение, записанное в r16 достигает значения 0x20, разрешается выполнение последующих команд программы. Таким образом, в программе реализован цикл по переменной **r16**, время исполнения которого она «ничего не делает», т.е. как бы «зависает» некоторое время. Это процесс и называется *программной задержкой*.

Зная время выполнения команд (прил. 1) можно рассчитать время исполнения любой части программы. Рассчитаем время исполнения рассмотренной части программы.

При попадании в цикл суммарное количество тактов процессора будет N, то есть при подсчете чисел от 1 до 32 суммарное количество тактов будет равняться $N=4 \cdot 32=128$. После этого необходимо вычесть один цикл, который остался неучтенным при выполнении ложного

условия **brnemet_1b** конце подсчета r16 и прибавить один такт на выполнение команды **clrr16**. Итого, суммарное количество тактов равно: $N=4 \cdot 32 - 1 + 1 = 128$.

Если умножить полученное число на время выполнения одного такта процессора, которое обратно частоте колебаний кварцевого резонатора f_{osc} (в нашем случае - 8 МГц), то можно найти время задержки T_z :

$$T_z = N * \frac{1}{f_{osc}} = 128 * \frac{1}{8 * 10^6} = 16 \text{ мкс}$$

Для реализации более существенных временных задержек приходится иметь дело с большими числами или вложенными циклами. Пример программной задержки с применением вложенных циклов приведен далее.

Пример. Написать программу бегущего огня на PORTC. При этом время задержки между переключениями разрядов порта задается с помощью программной задержки.

```

;-----
;include "m8535def.inc" ;стандартная библиотека Atmega 8535
.cseg ;начало сегмента кода
.org 0
ldi r16,$5f ;размещение вершины стека по адресу
ldi r17,$02 ;старшей ячейки ОЗУ
out spl,r16
out sph,r17
ldi r16,0xFF ;инициализация портов ввода/вывода
out DDRC,r16 ;PORTC - на вывод
ldi r16,0x01 ;занесение в PCH r16 числа 1
main: ;начало главной программы
out PORTC,r16 ;вывод на PORTC значения r16
in r17,SREG ;сохранение регистра SREG
rcall delay1 ;вызов подпрограммы задержки
out SREG,r17 ;восстановление SREG после возврата
rol r16 ;циклический поворот r16 вправо
rjmp main ;возврат на main
delay1: ;подпрограмма #1
clr r18 ;очистка регистра r18
met1:
rcall delay2 ;переход на 2 подпрограмму - delay2
inc r18 ;инкремент r18
cpi r18,0x10 ;сравнение значения r18 с числом 16
brne met1 ;если значение в r18 != 15, то переход на met1
ret ;иначе - возврат в основную программу
delay2: ;подпрограмма #2
clr r19 ;очистка регистра r19
met2:
rcall delay3 ;переход на 3 подпрограмму - delay3
inc r19 ;инкремент r19
cpi r19,0xFA ;сравнение значения r19 с числом 250
brne met2 ;если значение в r19 != 255, то переход на met2
ret ;иначе - возврат в подпрограмму #1

```

Рассмотрим программу более подробно.

1. В начале программы вершина стека помещается по адресу последней ячейки ОЗУ:

```

ldi r16,$5F
ldi r17,$02
out spl,r16
out sph,r17

```

2. Инициализируется порт ввода/вывода C, а регистру r16 присваивают значение $r16=0x01$ (начальная позиция бегущего огня):

```

ldi r16, 0xFF
out DDRC,r16
ldi r16,0x01

```

3. В главной программе значение регистра r16 выводится на индикацию PORTC, после чего вызывается подпрограмма задержки (**rcall delay**), по прошествии которой значение r16 сдвигается на один разряд вправо (**rol r16**). Далее программа замыкается:

main:

```

out PORTC,r16
in r17,SREG
rcall delay

```

```
out SREG,r17
```

```
rol r16
```

```
rjmp main
```

Здесь необходимо отметить, что перед вызовом подпрограммы задержки значение регистра SREG необходимо сохранить, так как иначе в результате выполнения подпрограммы этот регистр может изменить свое значение. После возврата подпрограммы значение SREG восстанавливается.

4. Подпрограмма задержки реализована по принципу, рассмотренному выше. В подпрограмме реализованы вложенные циклы:

```
delay1:
```

```
clr r18
```

```
met1:
```

```
rcall delay2
```

```
inc r18
```

```
cpi r18,0x10
```

```
brne met1
```

```
ret
```

```
delay2:
```

```
clr r19
```

```
met2:
```

```
rcall delay3
```

```
inc r19
```

```
cpi r19,0xFA
```

```
brne met2
```

```
ret
```

```
delay3:
```

```
clr r20
```

```
met3:
```

```
inc r20
```

```
cpi r20, 0xFA
```

```
brne met3
```

```
ret
```

Так, при входе в первый цикл, ограниченный меткой **met1** условием **brnemet1**, перед инкрементом регистра r18 происходит вызов подпрограммы **delay2**, в которой данная операция повторяется (выполняется второй цикл, вложенный в первый) и осуществляется переход на подпрограмму **delay3**, в которой выполняется третий цикл, вложенный во второй. Таким образом, сначала происходит выполнение цикла в подпрограмме **delay3** (инкремент регистра r20 по метке **met3**), потом - цикла в подпрограмме **delay2** (инкремент r19 по метке **met2**), после чего снова осуществляется переход на **delay1** (инкремент r18 по метке **met1**) и т.д..

Дисассемблирование программы

При оформлении отчета по проделанной работе необходимо произвести дисассемблирование написанной программы. Дисассемблирование представляет собой инструмент AVRStudio, благодаря которому можно увидеть работу программы, включая указатель стека и счетчик команд.

Дисассемблирование программы выполняется выбором в меню *View* AVR Studio пункта *Disassembler*. В этом случае на экране появляется программа, в которой указывается адрес каждой команды и необходимая служебная информация.

Одновременно на экран необходимо вывести окно контроля содержимого памяти *Togglememorywindow* (для вызова набрать комбинацию Alt+0), в котором необходимо выбрать память данных (*Data*). В конце области этой памяти будет располагаться указатель стека.

В отчете по работе необходимо привести таблицу дисассемблера со значениями, записанными в стеке во время исполнения программы (ее первого прохода). В таблице необходимо указать:

- все инструкции и метки программы;
- адрес памяти всех инструкций;
- текущее значение указателя стека во время исполнения программы;
- значения всех ячеек ОЗУ, в которые записывается стек, во время исполнения программы (первого прохода). Пример такой таблицы приведен в табл. 1.

Таблица 1: Дисассемблер программы со стеков

Адрес памяти программ	Команда / метка	Указатель стека SPHiSPL	Стек (ячейки ОЗУ)		
			025A:025B	025C:025D	025E:025F
+00000000	ldi r16,\$5F	00 00	FF:FF	FF:FF	FF:FF
+00000001	ldi r17,\$02	00 00	FF:FF	FF:FF	FF:FF
+00000002	out spl,r16	00 5F	FF:FF	FF:FF	FF:FF
+00000003	out sph,r17	02 5F	FF:FF	FF:FF	FF:FF
+00000004	ldi r16,0xFF	02 5F	FF:FF	FF:FF	FF:FF
+00000005	out DDRC,r16	02 5F	FF:FF	FF:FF	FF:FF
+00000006	ldi r16,0x01	02 5F	FF:FF	FF:FF	FF:FF
	main:				
+00000007	out PORTC,r16	02 5F	FF:FF	FF:FF	FF:FF
+00000008	in r17,SREG	02 5F	FF:FF	FF:FF	FF:FF
+00000009	rcall delay1	02 5D	FF:FF	FF:FF	00:0A
+0000000A	out SREG, r17	02 5F	00:15	00:0F	00:0A
+0000000B	rol r16	02 5F	00:15	00:0F	00:0A
+0000000C	rjmp main	02 5F	00:15	00:0F	00:0A
	delay1:				
+0000000D	clr r18	02 5D	FF:FF	FF:FF	00:0A
	met1:				
+0000000E	rcall delay2	02 5B	FF:FF	00:0F	00:0A
+0000000F	inc r18	02 5D	00:15	00:0F	00:0A
+00000010	cpi r18,0x10	02 5D	00:15	00:0F	00:0A
+00000011	brne met1	02 5D	00:15	00:0F	00:0A
+00000012	Ret	02 5F	00:15	00:0F	00:0A
	delay2:				
+00000013	clr r19	02 5B	FF:FF	00:0F	00:0A
	met2:				
+00000014	rcall delay3	02 59	00:15	00:0F	00:0A
+00000015	inc r19	02 5B	00:15	00:0F	00:0A
+00000016	cpi r19,0xFA	02 5B	00:15	00:0F	00:0A
+00000017	brne met2	02 5B	00:15	00:0F	00:0A
+00000018	Ret	02 5D	00:15	00:0F	00:0A
	delay3:				
+00000019	clr r20	02 59	00:15	00:0F	00:0A
+0000001A	inc r20	02 59	00:15	00:0F	00:0A
+0000001B	cpi r20,0xFA	02 59	00:15	00:0F	00:0A
+0000001C	brne met3	02 59	00:15	00:0F	00:0A
+0000001D	ret	02 5B	00:15	00:0F	00:0A

Задание на выполнение

1. Разработать программу «бегущий огонь» с заданной по вариантам программной задержкой, которая изменяется в зависимости от состояния входов.

Варианты программы «бегущий огонь» представлены в таблице.

№ вар.	1 такт	2 такт	3 такт	4 такт
1	PD0, PD1 - 2 с	PD1, PD2 - 1 с	PD2, PD3 - 0,5 с	-
2	PA0...PA3 - 0,1 с	PA4...PA7 - 0,2 с	PC0...PC3 - 0,3 с	PC4...PC7 - 0,4 с
3	PC0 - 0,5 с	PC2 - 0,5 с	PC4 - 1,5 с	PC6 - 1,5 с
4	PB0 - 1 с	Нет - 0,5 с	PB1 - 1 с	Нет - 0,5 с
5	PC7 - 5 с	PC6, PC7 - 5 с	PC5...PC7 - 5 с	PC4...PC7 - 5 с
6	PA0 - 2 с	PA1 - 4 с	-	-
7	PC0...PC3 - 10 с	PC1...PC4 - 10 с	PC2...PC5 - 10 с	-
8	PD7, PC7 - 0,5 с	Нет - 2 с	PD0, PC0 - 0,5 с	Нет - 2 с
9	PC0 - 0,2 с	PC1 - 0,2 с	PC2 - 0,2 с	Нет - 1 с
10	PA0...PA3 - 0,1 с	Нет - 0,2 с	PA4...PA7 - 0,1 с	Нет - 0,2 с
11	Порт D - 5 с	Нет - 1 с	Порт C - 5 с	-
12	PA3 - 2 с	Нет - 4 с	-	-
13	PC7 - 1 с	Нет - 5 с	PC6 - 1 с	Нет - 5 с
14	PA0...PA3 - 2 с	Нет - 1 с	PD0...PD3 - 2 с	-
15	PC0...PC3 - 0,1 с	PC1...PC3 - 0,1 с	PC2...PC3 - 0,1 с	PC3 - 0,5 с
16	Порт D - 2 с	Порт A - 2 с	Нет - 4 с	-
17	PC0...PC2 - 2 с	PC1...PC3 - 2 с	PC2...PC4 - 2 с	Нет - 5 с
18	Порт A - 0,1 с	Нет - 1 с	Порт C - 0,1 с	Нет - 1 с
19	PC0 - 0,1 с	PC1 - 0,1 с	PC2 - 0,1 с	Нет - 0,5 с
20	PB0 - 5 с	PB1 - 2,5 с	PB2 - 1,25 с	Нет - 5 с
21	PD7 - 1 с	PD6 - 1 с	PD0...5 - 5 с	-
22	PA6 - 1 с	PA7 - 4 с	-	-
23	PC0, PC3 - 2 с	PC1, PC4 - 2 с	Нет - 5 с	-
24	PD4...PD7 - 2,5 с	Нет - 5 с	PD0...PD3 - 2,5 с	Нет - 5 с
25	Нет - 1,5 с	Порт C - 0,5 с	Нет - 1,5 с	Порт A - 1,5 с
26	PB0 - 5 с	Нет - 10 с	-	-
27	PC6,7 - 2 с	PC4, PC5 - 2 с	PC2...PC3 - 2 с	PC0...PC1 - 4 с
28	PA0...PA3 - 2 с	PA4...PA7 - 2 с	PA0...PA7 - 2 с	Нет - 6 с
29	PD0, PD2, PD4 - 5 с	PD1, PD3, PD5 - 5 с	Нет - 10 с	-
30	PC0 - 1,5 с	PC1 - 3 с	PC2 - 4,5 с	-

Примечание: символ «-» обозначает что такт отсутствует, например, в 7 варианте программа выполняется только за 3 такта.

Контрольные вопросы

1. Поясните назначение стека.
2. В каких случаях в программе необходимо выполнять инициализацию стека?
3. Что произойдет с программой, если определить следующие значения указателя стека:
 - а) SPH=0, SPL=0
 - б) SPH=0, SPL=0x5F
 - в) SPH=0x02, SPL=0
4. Каким образом посчитать время исполнения инструкции? Части программы?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Выполненное задание с комментариями (работоспособность программы демонстрируется на стенде):
 - листинг программы;
 - дизассемблированную программу;
 - алгоритм;
 - представить расчет времени задержки по количеству команд;
 - представить таблицу значений стека во время исполнения программы.
3. Вывод по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №12. Организация работы 8-ми разрядного таймера в режиме ШИМ

Цель работы: разработка программы управления светодиодами с использованием параллельного порта МК

Выполнив работу, Вы будете:

уметь:

- формирование умений применять стандартные алгоритмы в соответствующих областях;
- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять современные компиляторы, отладчики и оптимизаторы программного кода;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

Стенд «Микроконтроллер», ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

Рассмотрим применение таймера на примере программы, в которой таймер T0 работает в режиме быстрого ШИМ. В зависимости от состояния битов порта A регулируется яркость светодиода, подключенного к выходу ШИМ таймера T0. Логическим сигналом с 0 бита порта DШИМ изменяется с инвертирующего на неинвертирующий

```

; ШИМ на таймере T0
; входы:
;   PORTA0...PORTA7   - задание уставки таймера
;   PORTD0            - инвертирующий (1)/неинвертирующий (0) ШИМ
; выходы:
;   PORTB3           - ШИМ на таймере T0

.include "m8535def.inc" ; подключение стандартной библиотеки
.cseg                  ; начало сегмента кода
.org $0                ; по адресу 0
rjmp reset             ; и переход на reset

reset:
  ldi r16,0x01         ; инициализация портов ввода вывода.
  out PORTD,r16       ; PORTD0 - на ввод информации
  clr r16
  out DDRD,r16
  ldi r16,0xFF
  out PORTA,r16       ; PORTA - на ввод информации
  clr r16
  out DDRA,r16
  out PORTB,r16
  ldi r16,0x08
  out DDRB,r16       ; PORTB3 - на вывод информации
  clr r16
  out TCCR0,r16       ; сброс регистра управления таймера T0
  out OCR0,r16       ; сброс регистра сравнения таймера T0
  out TCNT0,r16      ; сброс регистра счета таймера T0
  ldi r16,0x69
  out TCCR0,r16      ; установка режима быстрого неинвертирующего
                    ; ШИМ таймера T0
main:
  in r16,PINA        ; считывание значений порта ввода/вывода PORTA
  out OCR0,r16       ; и его отправка в регистр сравнения таймера T0
  in r16,PIND        ; считывание значений порта ввода/вывода PORTD
  andi r16,0x01     ; и выделение бита PORTD0
  cpi r16,0x01      ; Если PORTD0=1
  breq met1         ; то переход на метку met1
  ldi r17,0x69      ; иначе запись в r17 значения для неинверт. ШИМ
  rjmp met2         ; и переход на метку met2
met1:
  ldi r17,0x79      ; По метке met1
                    ; запись в r17 значения для инвертирующего ШИМ

```

```

met2:                ;По метке met2
    out TCCR0,r17    ;установка заданного режима работы таймера T0
    rjmp main       ;и возврат на main

```

Рассмотрим программу более подробно.

1. Инициализации портов ввода/вывода в соответствии с поставленным заданием:

```

ldi r16,0x01
out PORTD,r16
clr r16
out DDRD,r16
ldi r16,0xFF
out PORTA, r16
clr r16
out DDRA,r16
out PORTB,r16
ldi r16,0x08
out DDRB,r16

```

2. Инициализация таймера T0 на заданный режим работы:

```

clr r16
out TCCR0, r16
out OCR0,r16
out TCNT0,r16
ldi r16, 0x69
out TCCR0,r16

```

Сначала очищаются все регистры таймера (**clr r16; out TCCR0, r16; out OCR0, r16; out TCNT0, r16**). После этого в регистр управления TCCR0, в соответствии с табл. 1 записываются необходимые комбинации управляющих бит. Установкой WGM01=1, WGM00=1 выбирается режим быстрого ШИМ; установкой COM01=1, COM00=0 выбирается режим неинвертирующего ШИМ; биты CS02:CS01:CS00=001 определяют работу таймера без делителя частоты процессора clk/1.

3. В начале цикла происходит считывание данных с порта A и их запись в регистр сравнения таймера T0:

main:

```

in r16,PINA
out OCR0,r16

```

4. После этого производится опрос порта D и выделение младшего значащего бита:

```

inr16,PIND
and r16,0x01

```

5. Сравнение PORTD с логическими уровнями 0 и 1:

```

cpi r16,0x01
br eq met1
ldi r17,0x69
rjmp met2

```

met1:

```

ldi r17,0x79

```

Если в результате сравнения (cpi r16,0x01) PORTD=1, то происходит переход на метку met1, по которой в регистр r17 записывается значение, которое необходимо записать в регистр управления TCCR0 таймера T0 для реализации инвертирующего ШИМ (ldi r17,0x79). Иначе в r17 производится запись значения TCCR0 для неинвертирующего ШИМ (ldi r17, 0x69).

6. При переходе на метку met2 происходит запись полученного в r17 значения в регистр управления TCCR0 таймера T0 и закливание программы:

met2:

```

out TCCR0,r17

```

rtmp main

Подробное рассмотрение программы показывает, что при использовании в таймере режима ШИМ пользователь может не использовать прерывания по совпадению и переполнению таймера, так как в режиме ШИМ все действия таймер делает автоматически. Это существенно облегчает организацию программы и уменьшает ее размер.

Задание на выполнение

Составить программу для своего варианта, которая реализует вывод сигнала с ШИМ с изменением скважности для заданных таймеров по коду входных сигналов (биты задания).

Варианты задания (Условные обозначения: Б - быстрый ШИМ, Ф - фазовый ШИМ, П - прямой ШИМ, И - инверсный)

№ вар	Таймер (ы), канал (ы)	Режим	Диапазон частот ШИМ	Входы (биты)	Биты задания, количество дискретных значений, диапазон изменения скважности
1	T2	Б/ПиИ	[1000, 10000]	РА7-переключает между ПиИ	РС5...РС7, 8 положений, 0...0,5
2	T0 и T2	Ф/И	[1000, 10000]	РАО- включает таймер Т0 или Т2	РА4...РА7, 16 положений, 0,1... 0,9
3	T0	Б/И	[100,1000]	РС5 - меняет частоту ШИМ	Порт А, 256 положений, 0... 1
4	T0 и T2	Ф/ПиИ	[200,1000]	Одновременно работают 2 канала	РС0...РС3, 16 положений, 0...0,75
5	T2	БиФ/ ПиИ	<200	РАО-БиФ РА7 -ПиИ	РС0...РС4, 32 положения, 0,2...0,8
6	T0 и T2	Б/П	>10000	РАО - вкл Т0 РА1 - вкл Т2	РВ0...РВ2, 8 положений, 0...1
7	T0	Ф/И	>10000	РСО-вкп/откл	РАО...РА3, 16 положений, 0...1

8	T0	ФиБ/И	<200	РА4 - переключает между ФиБ	Порт С, 256 положений, 0...1
9	T2	Б/П	[1000, 10000]	РС1 - изменяет частоту ШИМ	РА2...РА3, 4 положения, 0...0,5
10	T0 и T2	Ф/ПиИ	[1000, 10000]	РС7 - выбирает Т0 или Т2	РС0...РС5, 64 положения, 0...1
11	T0 и T2	Б/И	[200, 1000]	РА0-0, РА1-Т0, РА2-Т2, РА3-Т0 и Т2	РС4...РС7, 16 положения, 0... 1
12	T0	Ф/П	[30, 35000]	РС0...РС2 - 6 частот ШИМ	Порт А, 256. положений, 0...0,5
13	T2	Б/ПиИ	<200	РАО - выбор ПиИ	РА2...РА5, 16 положений, 0...1
14	T0 и T2	Ф/П	>10000	РА0=0-Т0 и Т2 выкл РА0=1-Т0 и Т2 вкл	РС3...РС5, 8 положений, 0... 0,75
15	T0	Б/И	>10000	РС7- выкл/вкл	РВ0...РВ3, 16 положений, 0...0,8
16	T0 и T2	Ф/ПиИ	<200	РАО- меняет режим с П на И Т0 и Т2 одновременно	РА3...РА7, 32 • положений, 0;..1
17	T2	Б/И	[1000, 10000]	РВ3 - вкл/выкл выход таймера	РВ0...РВ2, '8 положений. 0...1
18	T0 и T2	ФиБ/ П и И	[1000, 10000]	РА7- ФиБ РА6- ПиИ	РА0...РА3, 16 положений, 0...0,5
19	T0 и T2	Б/П	[200, 1000]	РА4-Т0 РА5-Т2	РС1...РС3, ? положений, 0,1... .6,9
20	T0	ФиБ/И	[200, 1000]	РАО - вкл/выкл Т0 РА1 - Б и Ф	РА4...РА7, 16 положений, 0,25... 1
21	T2	Б/ПиИ	<200	РС0-ПиИ	РВ0...РВ4, 32 положений, 0... 1
22	T0 и T2	Ф/ПиИ	>10000	РА3- вкл Т0 или Т2	РА3...РА7, 32 ' положений, 0...0,8

23	ТО	Б/И	>10000	РАО- частота ШИМ	РС, 256 положений, 0...1
24	Т0иТ2	Ф/ПиИ	<200	РАО - выбор Т0 или Т2, РА1 - выбор П или И	РА5...РА7,8 положений, 0,2...0,9
25	Т0 и Т2	ФиБ/ ПиИ	[1000, 10000]	РС6 - выбор Ф или В, РС7 - выбор П или И	РВ0...РВ2, 8 положений, 0...0.75

Форма представления результата:

Отчет должен содержать:

1. Цель работы.

2. Выполненное задание с комментариями (работоспособность программы демонстрируется на стенде):

- исходное задание;
- функциональную схему;
- представить расчет скважности, настройку таймера
- запись данных (скважности) в ОЗУ или ПЗУ;
- листинг программы;
- дизассемблированную программу;
- алгоритм;
- представить таблицу значений стека во время исполнения программы.

3. Вывод по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №13.

Организация работы 8-ми разрядного таймера в режиме создания временных интервалов

Цель работы: разработка программы управления светодиодами с использованием параллельного порта МК

Выполнив работу, Вы будете:

уметь:

- формирование умений применять стандартные алгоритмы в соответствующих областях;
- применять выбранные языки программирования для написания программного кода;
 - использовать выбранную среду программирования;
 - применять нормативные документы, определяющие требования к оформлению программного кода;
 - использовать возможности имеющейся технической и/или программной архитектуры;
 - применять современные компиляторы, отладчики и оптимизаторы программного кода;
 - распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
 - анализировать задачу и/или проблему и выделять её составные части;
 - грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
 - понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

Стенд «Микроконтроллер», ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

Рассмотрим применение таймера на примере программы, осуществляющей инкремент какого-либо регистра до значения, заданного на входе порта D, при этом содержимое регистра выводится с интервалом 500 мс в порт C.

```

;-----
;Программа вывода в PORTC с периодом 500 мс увеличивающегося значения
;двоичного кода до значения, заданного на входах PIND.
;Входы:
;P0_PD7 - задание максимального значения числа
;PC0_PC7 - индикация результата
.include "m8535def.inc" ;подключение библиотеки контроллера
.cseg ;начало сегмента кода
.org $0 ;по адресу 0
    rjmp reset ;переход на метку reset
.org $13 ;адрес обработки прерывания по совпадению T0
    rjmp T0_compare ;и переход по этому адресу в случае
                    ;возникновения прерывания
                    ; инициализация стека
reset:
    ldi r16, low(RAMEND) ;запись в указатель стека адреса конца
    ldi r17, high(RAMEND) ;памяти данных
    out spl, r16
    out sph, r17
                    ; инициализация портов
    ldi r16, 0xFF ;инициализация портов ввода/вывода:
    out PORTD, r16 ;порт D - на ввод
    out DDRC, r16 ;порт C - на вывод
    clr r16
    out DDRD, r16
    out PORTC, r16
                    ; инициализация и запуск таймера T0
    ldi r16, 0x00 ;обнуление регистров таймера T0:
    out TCCR0, r16 ;регистра управления TCCR0
    out TCNT0, r16 ;регистра счета TCNT0
    ldi r16, 0x4E ;запись в регистр сравнения OCR0 числа 0x4E
    out OCR0, r16 ;соответствующего уставке 10 мс
    ldi r16, 0x05 ;запуск таймера T0 с делителем clk/1024
    out TCCR0, r16 ;в нормальном режиме работы
                    ; разрешение работы прерывания
    ldi r16, 0x02 ;для работы прерывания по совпадению T0
    out TIMSK, r16 ;накладывается маска OCIE0 в регистре TIMSK

```

```

    clr r16 ;необходимые в программе регистры
    clr r17 ;предварительно очищаются
    clr r18
    sei ;глобальное разрешение прерываний
main:
    rjmp main ;основная часть программы
                    ; пустой цикл
                    ; обработчик прерывания
T0_compare:
    cli ;при совершении прерывания по совпадению T0
                    ; глобальный запрет прерываний
    in r24, SREG ;сохранение регистра SREG
    clr r16 ;обнуление счетного регистра TCNT0
    out TCNT0, r16
    inc r17 ;инкремент POH r17
    cpi r17, 0x32 ;сравнение r17 с уставкой 0x32 (50)
    brne quit ;если r17 не равен уставке, то переход на
    quit
    clr r17 ;иначе очистка r17
    in r16, PIND ;ввод данных, содержащихся на PIND
    cp r16, r18 ;и сравнение POH r16 с регистром счета r18
    brpl met_1 ;если r16>r18, то переход на met_1
    clr r18 ;иначе очистка регистра счета r18
met_1:
    out PORTC, r18 ;далее - вывод на индикацию r18
    inc r18 ;и инкремент регистра счета
quit:
    out SREG, r24 ;восстановление регистра SREG
    sei ;глобальное разрешение прерываний
    reti ;выход из подпрограммы прерываний
;-----

```

Рассмотрим особенности программы.

1. Строками программы

```
.org$13
```

```
rjmp T0_compare
```

выполняется инициализация адреса памяти, по которому начинается подпрограмма обработки прерывания по совпадению таймера T0. Адреса подпрограмм обработки прерываний для каждого микроконтроллера заданы жестко и изменению не подлежат. Таблица адресов приведена в приложении.

2. В программе используется стек. Это связано с использованием в программе прерывания и необходимости сохранения в стеке адреса возврата из прерывания в основную программу. Поэтому в начале программы инициализируются регистры указателя стека SPH:SPL.

3. Для использования таймера T0 в начале необходимо выполнить его инициализацию. Для этого выполняются:

- обнуление счетного регистра TCNT0 и регистра управления TCCR0;
- устанавливается значение уставки в регистр сравнения OCR0.

Выполним расчет уставки. В случае данной программы получить уставку 500 мс на таймере не получается, так как даже при делителе $clk/1024$ максимальная уставка таймера равняется:

$$T_{уст} = \left(\frac{8000000}{1024}\right)^{-1} \cdot 256 = 32,768 \text{ мс}$$

Как реализовать на таймере T0 уставку большую значения 32,768 мс? Один из вариантов считать количество прерываний таймера. Например, для получения уставки 500 мс можно запрограммировать таймер на уставку 10 мс и производить подсчет срабатываний таймера — каждое 50-е срабатывание таймера будет соответствовать времени $T_{уст} = 50 \cdot 10 \text{ мс} = 500 \text{ мс}$. Рассчитаем значение уставки таймера в 10 мс при делителе $clk/1024$:

$$OCR0 = 10 \cdot 10^{-3} \cdot \frac{8000000}{1024} = 78,125.$$

Таким образом, округленное значение, записываемое в регистр сравнения OCR0, равно 78 (0x4E).

- запуск таймера. Выполняется установкой коэффициента делителя равного 1024, после этого таймер запускается в работу в нормальном режиме.

4. Разрешение работы прерывания по совпадению таймера T0. Для этого:

- в регистре масок прерываний таймеров TIMSK устанавливается в единичное значение 1 бит для разрешения прерывания по совпадению таймера;
- разрешается работа всех прерываний (инструкция **sei**) путем записи логической «1» в старший бит регистра SREG.

Основная программа

5. Основной цикл `main` пустой, т.к. не содержит никаких операторов.

6. Выполнение обработчика прерывания. При наступлении прерывания по совпадению T0 осуществляется переход из любого места основной программы (в данном случае только из строки `rjmp main`) по адресу обработки прерывания, указанному в строке `rjmp t0_compare`. После этого:

- запрещаются все прерывания (cli); которые разрешаются только по окончании обработки прерывания;
- сохраняется значение регистра SREG (inr24, sREG), который при выходе из подпрограммы восстанавливается (outsREG, R24);
- обнуление счетного регистра таймера T0;
- счет количества срабатываний прерывания таймера. Регистр r17 постоянно инкрементируется и если его значение не совпадает с уставкой 0x32 (десятичное значение равно 50) (инструкция spir17,0x32) происходит выход из подпрограммы обработки прерывания (brnequit);
- при совпадении значения регистра r17 с уставкой 0x32, т.е. каждые 500 мс счетчика прерываний обнуляется (clr r17), считываются данные с порта D (inr16, PiND), выполняется сравнение с регистром счета r18 (ср r 16, r 18).
- если уставка, заданная на PiND, меньше, чем текущее значение r18, то r18 обнуляется. После этого значение r18 выводится на PORTC (outPORTC,r18) и значение этого регистра инкрементируется;
- в конце подпрограммы вновь разрешаются прерывания (инструкция sei) и завершается обработка прерывания (инструкция reti) и выполняется.

Задание на выполнение

1. С использованием таймера T0 составить программу «бегущий огонь» по вариантам с включением заданных выходов портов, периодами их включения и изменениями в тактах работы.

Варианты программы «бегущий огонь»

Примечание: символ «-» обозначает что такт отсутствует, например, в 1 варианте программа выполняется только за 3 такта.

№ вар.	1 такт	2 такт	3 такт	4 такт
1	PD0, PD1 - 2 с	PD1, PD2 - 1 с	PD2, PD3 - 0,5 с	-
2	PA0...PA3 - 0,1 с	PA4...PA7 - 0,2 с	PC0...PC3 - 0,3 с	PC4...PC7 - 0,4 с
3	PC0 - 0,5 с	PC2 - 0,5 с	PC4 - 1,5 с	PC6 - 1,5 с
4	PB0 - 1 с	Нет - 0,5 с	PB1 - 1 с	Нет - 0,5 с
5	PC7 - 5 с	PC6, PC7 - 5 с	PC5...PC7 - 5 с	PC4...PC7 - 5 с
6	PA0 - 2 с	PA1 - 4 с	-	-
7	PC0...PC3 - 10 с	PC1...PC4 - 10 с	PC2...PC5 - 10 с	-
8	PD7, PC7 - 0,5 с	Нет - 2 с	PD0, PC0 - 0,5 с	Нет - 2 с
9	PC0 - 0,2 с	PC1 - 0,2 с	PC2 - 0,2 с	Нет - 1 с
10	PA0...PA3 - 0,1 с	Нет - 0,2 с	PA4...PA7 - 0,1 с	Нет - 0,2 с
11	Порт D - 5 с	Нет - 1 с	Порт C - 5 с	-
12	PA3 - 2 с	Нет - 4 с	-	-
13	PC7 - 1 с	Нет - 5 с	PC6 - 1 с	Нет - 5 с
14	PA0...PA3 - 2 с	Нет - 1 с	PD0...PD3 - 2 с	-
15	PC0...PC3 - 0,1 с	PC1...PC3 - 0,1 с	PC2...PC3 - 0,1 с	PC3 - 0,5 с
16	Порт D - 2 с	Порт A - 2 с	Нет - 4 с	-
17	PC0...PC2 - 2 с	PC1...PC3 - 2 с	PC2...PC4 - 2 с	Нет - 5 с
18	Порт A - 0,1 с	Нет - 1 с	Порт C - 0,1 с	Нет - 1 с
19	PC0 - 0,1 с	PC1 - 0,1 с	PC2 - 0,1 с	Нет - 0,5 с
20	PB0 - 5 с	PB1 - 2,5 с	PB2 - 1,25 с	Нет - 5 с
21	PD7 - 1 с	PD6 - 1 с	PD0...5 - 5 с	-
22	PA6 - 1 с	PA7 - 4 с	-	-
23	PC0, PC3 - 2 с	PC1, PC4 - 2 с	Нет - 5 с	-
24	PD4...PD7 - 2,5 с	Нет - 5 с	PD0...PD3 - 2,5 с	Нет - 5 с
25	Нет - 1,5 с	Порт C - 0,5 с	Нет - 1,5 с	Порт A - 1,5 с
26	PB0 - 5 с	Нет - 10 с	-	-
27	PC6,7 - 2 с	PC4, PC5 - 2 с	PC2...PC3 - 2 с	PC0...PC1 - 4 с
28	PA0...PA3 - 2 с	PA4...PA7 - 2 с	PA0...PA7 - 2 с	Нет - 6 с
29	PD0,PD2,PD4 - 5 с	PD1, PD3,PD5 - 5 с	Нет - 10 с	-
30	PC0 - 1,5 с	PC1 - 3 с	PC2 - 4,5 с	-

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Выполненное задание с комментариями (работоспособность программы демонстрируется на стенде):
 - исходное задание;

- функциональную схему;
- листинг программы; дизассемблированную программу;
- блок-схему алгоритма;
- представить настройку таймера и расчет времени задержки реализованной на таймере;
- представить таблицу значений стека во время исполнения программы.

3. Вывод по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №14.

Организация работы АЦП МК Atmega 8535

Цель работы: разработка программы управления светодиодами с использованием параллельного порта МК

Выполнив работу, Вы будете:

уметь:

- формирование умений применять стандартные алгоритмы в соответствующих областях;
- применять выбранные языки программирования для написания программного кода;
 - использовать выбранную среду программирования;
 - применять нормативные документы, определяющие требования к оформлению программного кода;
 - использовать возможности имеющейся технической и/или программной архитектуры;
 - применять современные компиляторы, отладчики и оптимизаторы программного кода;
 - распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
 - анализировать задачу и/или проблему и выделять её составные части;
 - грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
 - понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

Стенд «Микроконтроллер», ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

Рассмотрим применение АЦП на примере программ.

Пример 1. Подать аналоговый сигнал на вход ADC2 АЦП и вывести 8-разрядный результат на порт ввода/вывода С. Не использовать прерывание готовности результата преобразования АЦП.

```

;-----
Входы:
    PA2 - аналоговый вход АЦП
Выходы:
    PC0_PC7 - индикация кода результата преобразования.

.include "m8535def.inc" ;Подключение стандартной библиотеки
.cseg ;Начало сегмента кода
.org $0 ;по адресу 0

    ldi r16,low(RAMEND) ;Запись адреса вершины стека
    ldi r17,high(RAMEND) ;В конце памяти данных
    out spl,r16
    out sph,r17
    ldi r16,0xFB ;Инициализация неиспользуемых ножек порта А
    out PORTA,r16 ;на ввод информации
    clr r16 ;Используемый вход АЦП не инициализируется
    out DDRA,r16
    out PORTC,r16 ;Инициализация порта С на вывод информации
    ser r16
    out DDRC,r16
    ldi r16,0x62 ;Инициализация мультиплексора АЦП
    out ADMUX,r16
    ldi r16,0xC7
    out ADCSRA,r16 ;Инициализация АЦП и его запуск

main: ;Начало основного цикла
    in r16,ADCSRA ;Опрос регистра ADCSRA
    SBRC r16,4 ;Если бит 4 чист, то следующая команда
    пропускается
    rcall ADC_ready ;Иначе вызов ADC_ready
    rjmp main ;Возврат на main

ADC_ready: ;По метке ADC_ready
    in r16,SREG ;сохранение значения регистра SREG
    ldi r17,0x97 ;Обнуление флага ADIF записью в него логической
    «1»
    out ADCSRA,r17
    in r17,ADCH ;Считывание результата преобразования из ADCH
    out PORTC,r17 ;и вывод результата на PORTC
    ldi r17,0xC7 ;Перезапуск АЦП
    out ADCSRA,r17
    out SREG,r16 ;Восстановление регистра SREG
    ret ;возврат по адресу, хранящемуся в стеке
;-----

```

Рассмотрим программу более подробно.

1. Сначала происходит подключение стандартной библиотеки контроллера Atmega8535, указывается адрес начала сегмента кода и выполняется инициализация стека, вершина которого располагается в конце памяти данных:

```
include "m8535def.inc"
```

```
cseg
```

```
org $0
```

```
ldi r16,low(RAMEND)
```

```
ldi r17,high(RAMEND)
```

```
out spl,r16
```

```
out sph,r17
```

2. Производится инициализация портов ввода/вывода. В программе используются порты ввода/вывода А и С. При этом необходимо запомнить, что, поскольку АЦП использует альтернативные функции порта А, то выводы порта, используемые АЦП, инициализировать не нужно. Остальные выводы рекомендуется «притянуть» к уровню логического «0» или «1» во избежание наведения на них помех. Порт С инициализируется на вывод информации:

```
ldi r16,0xFB
```

```
out PORTA,r16
```

```
clr r16
```

```
out DDRA,r16
```

```
out PORTC,r16
```

```
ser r16
```

```
out DDRC,r16
```

3. Далее выполняется инициализация АЦП установкой необходимых управляющих бит в регистрах ADMUX и ADCSRA:

```
ldi r16, 0x62
```

```
out ADMUX,r16
```

```
ldi r16,0xC7
```

```
outADCSRA,r16
```

В качестве источника сигнала опорного напряжения выбирается напряжение электропитания процессора, поданное на вывод AVCC. Также выбирается «левое» выравнивание результата преобразования ADLAR. Выбирается второй канал АЦП (ldi r16, 0x62; outADMUX,r16). В регистре ADCSRA устанавливаются биты ADEN (включение АЦП), ADSC (запуск преобразования АЦП), а также выбирается делитель clk/128 для повышения точности преобразования.

4. В основном цикле ведется опрос флага окончания преобразования АЦП, расположенного в регистре ADCSRA:

```
main:
```

```
in r16,ADCSRA
```

```
SBRC r16,4
```

```
rcallADC_ready
```

```
rjmpmain
```

Сначала считывается значение регистра ADCSRA в регистр r16 (inr16,ADCSRA). После этого опрашивается 4-й бит этого регистра, соответствующий флагу ADIF (SBRC r16,4). Если бит чист, то пропускается следующая за директивой SBRC инструкция. Иначе происходит переход на метку ADC_ready (rcallADC_ready).

5. По метке ADC_ready происходит обработка результата преобразования АЦП:

```
ADC_ready:
```

```
in r16,SREG
```

```
ldi r17,0x97
```

```
out ADCSRA,r17
```

```
in r17,ADCH
```

```
outPORTC,r17
```

```
ldi r17,0xC7
```

```
out ADCSRA,r17
```

```
out SREG,r16
```

```
ret
```

При переходе по метке ADC_ready сначала в регистр r16 необходимо сохранить значение регистра SREG (inr16,SREG). После этого необходимо вручную сбросить флаг готовности преобразования АЦП записью в него логической «1» (ldi r17,0x97; outADCSRA,r17). Необходимо запомнить, что все флаги в микроконтроллерах AVR сбрасываются записью в них логической «1». Далее в регистре r17 считывается результат преобразования АЦП, хранящийся в ADCH (inr17,ADCH). Два младших бита, хранящиеся в ADCL, отбрасываются. Содержимое r17 передается в порт ввода/вывода C (outPORTC, r17), осуществляется перезапуск АЦП, восстановление регистра SREG (outSREG, r16) и выход из подпрограммы на адрес, записанный в стеке (ret).

Пример 2. Задача повторяет пример 1, но используется прерывание по готовности результата преобразования АЦП. Не использовать «левое» выравнивание результата.

```
-----  
; Входы:  
; PA2 - аналоговый вход АЦП  
; Выходы:  
; PC0...PC7 - индикация кода результата преобразования.  
.include "m8535def.inc" ; Подключение стандартной библиотеки
```

```

cseg
org$0
jmp reset
org$0E *
jmp ADC_ready

reset:
cli
ldi r16, low(RAMEND)
ldi r17, high(RAMEND)
out spl, r16
out sph, r17
ldi r16, 0xFB
out PORTA, r16
clr r16
out DDRA, r16
out PORTC, r16
ser r16
out DDRC, r16
ldi r16, 0x42
out ADMUX, r16
ldi r16, 0xCF
out ADCSRA, r16
sei
ain:
rjmp main

ADC_ready:
cli
in r16, SREG
in r17, ADCL
in r18, ADCH
lsl r17
lsl r17
clr r19
met1:
lsl r18
inc r19
cpi r19, 0x06
brne met1
or r17, r18
out PORTC, r17
ldi r17, 0xCF
out ADCSRA, r17
out SREG, r16
sei
reti

```

Рассмотрим программу более подробно, показав отличия от программы, рассмотренной в примере 1.

1. В начале программы, помимо подключения библиотеки контроллера Atmega8535 указываются адреса прерываний: reset- нулевое прерывание по адресу 0, прерывание готовности результата преобразования АЦП по адресу \$0E:
 .include "m8535def.inc"

```

.cseg
.org$0
rjmp reset
.org$0E
rjmp ADC_ready

```

При попадании на эти векторы происходит переход на соответствующие метки reset и ADC_ready.

2. Порты ввода/вывода и указатель стека инициализируются так же, как и в примере 1. Однако инициализация АЦП производится по-другому:

```

reset:
cli
ldi r16, low(RAMEND)
ldi r17, high(RAMEND)
out spl, r16
out sph, r17
ldi r16, 0xFB
out PORTA, r16

```

```

clr r16
out DDRA,r16
out PORTC,r16
ser r16
out DDRC,r16
ldi r16,0x42
out ADMUX,r16
ldi r16,0xCF
out ADCSRA,r16
sei

```

В регистре управления мультиплексором ADMUX убирается бит «левого» выравнивания результата. Это не связано с использованием прерывания по готовности АЦП, а сделано для демонстрации работы с двумя регистрами хранения результата АЦП: ADCH и ADCL. В регистре управления АЦП ADCSRA ставится маска ADIFSC для прерывания готовности АЦП для его активации. После всех установок необходимо осуществить глобальное разрешение прерываний (sei).

3. В отличие от примера 1, основной цикл программы выполнен пустым:

```
main:
```

```
rjmp main
```

Это сделано по причине того, что при использовании нет необходимости занимать процессор постоянной проверкой флага прерывания готовности АЦП. При появлении флага прерывания основной цикл main будет автоматически прерван, адрес возврата сохранен в стеке, а программа перейдет по метке ADC_ready, указанной в векторе \$0E.

4. При переходе по метке ADC_ready осуществляется обработка прерывания по готовности АЦП:

```
ADC_ready:
```

```
cli
in r16,SREG
in r17,ADCL
in r18,ADCH
lsl r17
lsl r17
clr r19

```

```
stl:
```

```
lsl r18
inc r19
cpi r19,0x06
brne met1
or r17,r18
out PORTC,r17
ldi r17,0xCF
out ADCSRA,r17
out SREG,r16
sei
reti

```

При входе в подпрограмму сначала производится глобальный запрет всех прерываний (cli). При использовании одного прерывания этого можно не делать, однако правильно при входе в любую подпрограмму обработки прерываний — запрещать.

После этого производится сохранение регистра SREG в ПОН r16. Обнуление флага прерывания не производится, так как при использовании прерывания это делается автоматически.

Результат преобразования считывается из регистров ADCH(старший) и ADCL(младший). Поскольку результат преобразования АЦП - 10-разрядное слово, в регистре ADCH(r18) будут использованы два младших бита: 0000 00XXа регистре ADCL(r17) - все биты: XXXX XXXX.

Для использования восьми старших битов результата необходимо сместить спотерей0 и 1 битов регистр r17, приведя его к виду: 00XXXXXX(lsr17; lsr r17), сместить два младших бита регистра r18 на 6 позиций влево, приведя его к виду: XX00 0000.

met1

```
lsrr18
inc r19
spi r19, 0x06
brnemet1
```

После этого регистры r17 и r18 складываются (orr17, r18),а результат выводится в порт С. Далее АЦП перезапускается, регистр SREGвосстанавливается (outSREG, R16),производится глобальное разрешение всех прерываний (sei), а этом - выход из подпрограммы обработки прерывания (reti).

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Выполненное задание (на выбор пример 1 или 2) - работоспособность программы демонстрируется на стенде:

- листинг программы;
- комментарии.

3. Вывод по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №15.

Разработка программы управления сегментным индикатором

Цель работы: научиться составлять программы для вывода информации на семисегментные индикаторы под управлением МК.

Выполнив работу, Вы будете:

уметь:

- формирование умений применять стандартные алгоритмы в соответствующих областях;
- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять современные компиляторы, отладчики и оптимизаторы программного кода;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

Стенд «Микроконтроллер», ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

Рассмотрим управление индикаторами на примере программ.

Пример 1. Написать программу, осуществляющую вывод на индикатор- числа от 0 до Fв шестнадцатеричном коде в соответствии с комбинацией сигналов на входе порта ввода/вывода А

```
//-----  
;Программа вывода чисел 0..F на один разряд семисегментного индикатора  
;Входы:  
; PA3_PA0 - задание кода числа  
;Выходы:  
; PD0 - управление подачей напряжения на индикатор  
; PC7_PC0 - управление сегментами индикатора  
  
.include "m8535def.inc" ;Подключение библиотеки Atmega8535  
.cseg ;Начало сегмента кода.  
.org $0 ;По адресу 0  
; во Flash  
reset: ;По метке reset осуществляется:  
 ldi r16,low(RAMEND) ;Инициализация стека. Вершина стека - в  
 ldi r17,high(RAMEND);конце памяти данных  
 out spl,r16  
 out sph,r17  
 clr r16 ;Инициализация портов ввода/вывода  
 out PORTC,r16 ;Порт C - на вывод  
 out PORTD,r16 ;Порт D - на вывод  
 ser r16  
 out DDRC,r16  
 out DDRD,r16  
 ldi r16,0x0F  
 out PORTA,r16 ;младшие 4 бита порта А - на ввод  
 clr r16  
 out DDRA,r16  
 ldi r16,0x01 ;Установка младшего бита порта D с целью  
 out PORTD,r16 ; подачи напряжения на общие аноды индикатора  
  
main: ;По метке main  
 in r16,PINA ;Происходит считывание данных с порта А  
 andi r16,0x0F ;и выделение бит PA3_PA0.  
 mov r30,r16 ;формирование адреса flash исходя из  
 ldi r31,0x02 ;считанных данных  
 lpm r16,Z ;извлечение в r16 данных из flash  
 out PORTC,r16 ;и вывод их на порт C (катоды индикатора).  
 rjmp main ;переход на main и запуск программы  
  
.org $100 ;По адресу 100  
.db 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07 ; таблица  
.db 0x7F, 0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71 ; кодов символов  
  
//-----
```

Рассмотрим программу более подробно.

1. Сначала производится подключение библиотеки используемого контроллера Atmega8535. После этого объявляется адрес начала сегмента кода.

2. По метке `resetsначала` происходит инициализация стека, затем - инициализация периферийных устройств микроконтроллера. В данном случае из периферийных устройств используются только порты ввода/вывода.

set:

```
ldi r16,low(RAMEND)
ldi r17,high(RAMEND)
out spl,r16
out sph,r17
clr r16
out PORTC,r16
out PORTD,r16
ser r16
out DDRC,r16
out DDRD,r16
ldi r16,0x0F
out PORTA,r16
clr r16
out DDRA,r16
ldi r16,0x01
outPORTD,r16
```

В регистры указателя стека записывается адрес вершины стека, который отвечает концу памяти данных (`ldi r16, low(RAMEND); ldi r17, high(RAMEND)`; `outspl,r16; outsph, r17`). Это необходимо для правильной работы программы при использовании подпрограмм и переходов, порт С в программе будет подключен к катодам индикатора, поэтому он инициализируется на вывод, младший бит порта D по заданию управляет подачей напряжения питания на общие аноды индикатора, поэтому порт инициализируется . вывод, биты PA3...PA0 порта ввода/вывода А инициализируются на ввод информации.

3. В основном цикле программы сначала происходит опрос порта ввода/вывода А:

in:

```
in r16,PINA
andir16,0x0F
```

Производится опрос всего порта (`inr16,PiNA`), после чего путем побитового сложения результата на число `0xFF` (`andir16,0x0F`) происходит выделение младших значащих бит порта А.

4. По результату, считанному с порта А, производится формирование адреса Flash-памяти, по которому хранится соответствующий символ. Поскольку во Flash-памяти данные хранятся словами (2 байта), то для доступа к отдельному байту указывается двойной адрес. Поскольку адрес начала массива данных `0x100` (адрес в словах), то адрес первого байта - `0x200` (в байтах).

Для чтения данных из Flash в Z-регистре (`r31:r30`) указывается адрес памяти, а затем командой `lpm` происходит считывание данных:

```
mov r30,r16
ldi r31,0x02
lpm r16,Z
```

5. После считывания данных из Flash они выводятся на порт С, соединенный с индикатором:

```
out PORTC,r16
rjmp main
```

Далее программа закичивается (`rjmp main`) для постоянного опроса порта ввода/вывода А.

1. По адресу Flash-памяти записывается таблица кодов символов:

```
.org$100 ;По адресу 100
```


.db 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07 ; таблица

.db 0x7F, 0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71 ; кодов символов

Таблица кодов символов включает 16 кодовых комбинаций, каждая из которых соответствует выводимому на индикатор символу в шестнадцатеричном коде. Таблица цифр от 0 до F приведена в табл. 1.

Табл. 1. Соответствие цифры и его шестнадцатеричного и двоичного кодов

Цифра	Двоичный	Шестнадцатеричный
0	0b00111111	0x3F
1	0b00000110	0x06
2	0b01011011	0x5B
3	0b01001111	0x4F
4	0b01100110	0x66
5	0b01101101	0x6D
6	0b01111101	0x7D
7	0b00000111	0x07
8	0b01111111	0x7F
9	0b01101111	0x6F
A	0b01110111	0x77
b	0b01111100	0x7C
C	0b00111001	0x39
d	0b01011110	0x5E
E	0b01111001	0x79
F	0b01110001	0x71

```
ldi r16,low(0x00000000); ;Инициализация указателя стека
ldi r17,high(RAMEND)
out spl,r16
out sph,r17
clr r16 ;Инициализируются порты ввода/вывода
out PORTC,r16
out PORTD,r16
ser r16
out DDRC,r16 ;PORTC - на вывод информации
out DDRD,r16 ;PORTD - на вывод информации
ldi r17,0x01 ;Включается младший разряд PORTD
out PORTD,r17 ;Для подачи напряжения на HG1

main: ;По метке main
cpi r17,0x01 ;производится опрос, какой разряд HG работает
breq HG1 ;если первый, то переход на метку HG1
cpi r17,0x02 ;если второй,
breq HG2 ;то переход на метку HG2
cpi r17,0x04 ;если третий,
breq HG3 ;то переход на метку HG3
cpi r17,0x08 ;если четвертый,
breq HG4 ;то переход на метку HG4
HG1: ;По метке HG1
```

Пример 2. Написать программу индикации на четырехразрядном семисегментном индикаторе числа 1234 с использованием динамической индикации. Код числа выдается на порт C, управление разрядами осуществляется с порта D.

```

    ldi r31,0x02      ;в Z-регистр задается адрес числа 4 в Flash
    ldi r30,0x04
    rjmp met1        ;и переход на метку met1
2:                   ;По метке HG2
    ldi r31,0x02      ;в Z-регистр задается адрес числа 3 в Flash
    ldi r30,0x03
    rjmp met1        ;и переход на метку met1
3:                   ;По метке HG3
    ldi r31,0x02      ;в Z-регистр задается адрес числа 2 в Flash
    ldi r30,0x02
    rjmp met1        ;и переход на метку met1
4:                   ;По метке HG4
    ldi r30,0x01      ;в Z-регистр задается адрес числа 1 в Flash
    rjmp met1        ;и переход на метку met1
t1:                  ;По метке met1
    lpm r16,Z         ;из Flash по указанному адресу считываются данные
    out PORTC,r16     ;и выводятся на PORTC
    clr r16           ;Осуществляется программная задержка времени
t2:
    inc r16
    cpi r16,0xFF
    brne met2
    lsl r17            ;Значение r17 сдвигается на один разряд влево
    cpi r17,0x10      ;Если r17=0x10, то
    brne met3
    ldi r17,0x01     ;в r17 записывается 0x01
t3:
    clr r16
    out PORTC,r16     ;обнуляется PORTC
    out PORTD,r17     ;обнуляется PORTD
    rjmp main        ;осуществляется переход на main

rg$100              ;По адресу 100
b 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07 ; массив данных
b 0x7F, 0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71 ; кодов символов

```

Рассмотрим программу более подробно.

1. Сначала производится подключение библиотеки используемого контроллера Atmega8535. После этого объявляется адрес начала сегмента кода:

```
include"m8535def.inc"
```

```
seg
rg$0
```

2. По метке reset осуществляется конфигурирование периферийных устройств контроллера и инициализация указателя стека:

```
set:
ldi r16,low(RAMEND)
ldi r17,high(RAMEND)
out spl,r16
out sph,r17
clr r16
out PORTC,r16
out PORTD,r16
ser r16
out DDRC,r16
out DDRD,r16
ldi r17,0x01
outPORTD,r17
```

После инициализации устройств в регистр r17 записывается число 0x01 (ldi r17,0x01), после чего содержимое регистра выводится на порт управления разрядами индикатора (outPORTD,r17). Далее в r17 будет находиться значение, соответствующее включенному состоянию PORTD.

3. По метке main сначала производится опрос включенного состояния PORTD путем

опроса регистра r17:

main:

```
    cpi r17,0x01
    breq HG1
    cpi r17,0x02
    breq HG2
    cpi r17,0x04
    breq HG3
    cpi r17,0x08
    breqHG4
```

В зависимости от состояния r17 производится переход на метки HG1...HG4. Так если в данный момент работает разряд HG1 (сpi r17,0x01), то осуществляется переход на метку HG1 (breqHG1).

4. Далее, в зависимости от метки, производится запись в Z-регистр значения адреса Hash-памяти, по которому находится код нужного символа:

HG1:

```
    ldi r31,0x02
    ldi r30,0x04
    rjmp met1
```

HG2:

```
    ldi r31,0x02
    ldi r30,0x03
    rjmp met1
```

HG3:

```
    ldi r31,0x02
    ldi r30,0x02
    rjmp met1
```

HG4:

```
    ldi r30,0x01
    rjmp met1
```

met1:

```
    lpm r16,Z
    outPORTC,r16
```

После записи адреса символа осуществляется переход на метку met1, по которой происходит считывание данных из Flash-памяти в РОН r16 (lpm r16,Z) и вывод их на PORTC(outPORTC,r16).

5. В момент передачи данных на PORTC на одном из разрядов индикатора загорается нужный символ, после чего необходимо сделать небольшую задержку времени:

```
    clr r16
```

met2:

```
    inc r16
    cpi r16,0xFF
    brnemet2
```

Для реализации задержки сначала содержимое РОН r16 обнуляется (clr r16), после чего происходит его инкремент (inc r16) с последующим сравнением с уставкой (cpi r16, 0xFF). При значении r16, меньшем уставки, происходит возврат на метку met2 (brnemet2). При достижении уставки программа следует дальше.

6. По окончании выдержки времени необходимо выключить индикатор и переключиться на индикацию другого разряда:

```
    lsl r17
    cpi r17,0x10
    brne met3
    ldi r17,0x01
```

met3:

```
clr r16
out PORTC,r16
out PORTD,r17
rjmpmain
```

С этой целью содержимое r17 последовательно сдвигается на один разряд влево (lslr17), после чего производится выдача его содержимого на PORTD(outPORTD, r17). Однако при достижении r17 значения 0x10 (spir17,0x10) в него необходимо записать значение 0x01 (ldir17,0x01) для того, чтобы разряды PD0...PD3 включались циклично, а не происходило включение разрядов PD4...PD7 порта D. Во избежание ложного отображения информации перед сменой включенного разряда индикатора происходит выключение сегментов индикатора (clr r16; outPORTC,r16). После этого программа закичивается (rjmpmain).

7. Адрес Flash-памяти, по которому записывается таблица кодов символов:

```
.org$100
.db 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07
.db 0x7F, 0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71
```

Задание на выполнение

Составить программу реализации динамической индикации для заданной частоты. Данные предварительно записать в ОЗУ, выводить их на индикацию по адресу записи.

Варианты индивидуальных заданий

№ вар	Частота динамической индикации, Гц	Таймер	Биты PINA0 и PINA1			
			00	01	10	11
1	1000	T0	Дата.День.Рождения	Год.Рождения	-	-
2	200	T1	Имя	Год.Рождения	-	-
3	500	T2	«Add»	«Ldi»	«Clr»	«spi»
4	4000	T0	НомерГруппы	Год.Поступления	-	-
5	400	T1	«Пуск»	«Стоп»	-	-
6	3000	T2	Век	Год(2послед.Цифры)	-	-
7	250	T0	Дата.Рождения	День.Рождения	Год.Рождения	-
8	5000	T1	12	12 в 2 системе сч.	12 в 16 системе сч.	12 в 8 системе сч.
9	900	T2	«Аbc»	«BCd»	-	-
10	300	T0	Кол-во студ. в группе	НомерГруппы	Год.поступления	Год.окончания
11	600	T1	«Ab»	«bc»	«cd»	«DE»
12	450	T2	123	456	789	000
13	1500	T0	Имя1	Имя2	-	-
14	150	T1	День.Рождения	Месяц.Рождения	Год.Рождения	-
15	330	T2	1	12	123	1234
16	10000	T0	«in»	«out»	«call»	-
17	333	T1	Дата.рождения	Месяц.рождения	Год.рождения	«...»
18	700	T2	Дата.День.Рождения	Год.Рождения	-	-
19	3500	T0	Год.рождения	«год»	-	-
20	1200	T1	15	15 в 2-ой системе счисл.	15 в 16-ой системе счисл.	15 в 8-ой системе счисл.
21	4200	T2	jan	feb	arg	Jun
22	560	T0	74	Члб	66	Сад
23	780	T1	руб	euro	dol	-
24	2200	T2	cos	sin	Ln	Lg
25	350	T0	32	64	128	256
26	125	T1	День.рождения	Месяц.рождения	Год.рождения	Год.поступления
27	220	T2	10 в 2-ой системе счисл.	10 в 3-ой системе счисл.	10 в 4-ой системе счисл.	10 в 8-ой системе счисл.
28	490	T0	31.28	31.30	31.30	31.31
29	850	T1	2009	2010	2011	2012
30	3300	T2	abc	def	ghi	-

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Выполненное задание с комментариями (работоспособность программы демонстрируется на стенде):
 - исходное задание;
 - функциональную схему;
 - представить расчет частоты динамической индикации и кодов данных;

- листинг программы;
- дизассемблированную программу;
- алгоритм;
- представить таблицу значений стека во время исполнения программы.

3. Вывод по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Практическое занятие №1. Изучение схемы типовой МПС

Цель работы: изучить принципы построения и функционирования МПС на примере МПС стенда LESO1.

Выполнив работу, Вы будете:

уметь:

- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

Стенд LESO1

Теоретические сведения

Учебный лабораторный стенд предназначен для освоения студентами микропроцессорной архитектуры MCS-51 и методов разработки микропроцессорных систем различного назначения. Стенд может стать основой дипломного проектирования студентов, а также базой исследовательской работы бакалавров и магистров. На базе стенда возможна разработка промышленных автоматизированных систем. Структурная схема стенда приведена на рисунке 1. Расположение основных элементов показано на рисунке 2. Электрическая схема стенда приведена на рисунке

3.

Электрические параметры:

–напряжение питания 5 В от шины USB ПК;

–потребляемая мощность не более 2 Вт.

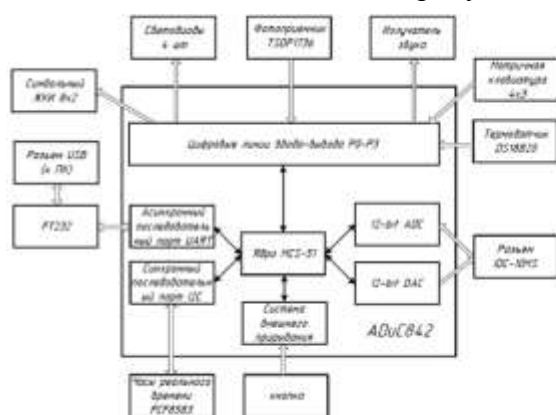


Рисунок 1 – Структурная схема учебного лабораторного стенда LESO1

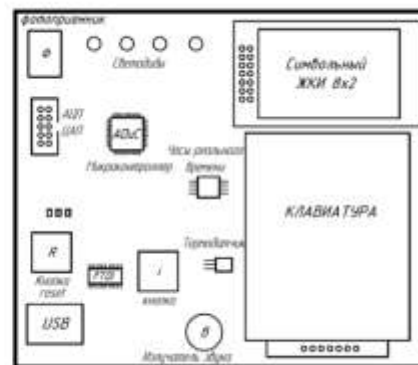


Рисунок 2 – Расположение основных элементов на стенде

Состав изделия:

–микроконтроллер ADuC842;

- жидкокристаллический символьный индикатор 8x2;
- матричная клавиатура 4x3;
- датчик температуры DS18B20;
- часы реального времени PCF8583;
- излучатель звука;
- инфракрасный фотоприемник TSOP1736;
- четыре красных светодиода;
- микросхема преобразования интерфейсов фирмы FTDI.

Учебный стенд модели LESO1 представляет собой настольную конструкцию. Для загрузки исполняемого кода во внутреннюю память микропроцессора и взаимодействия лабораторного стенда с ПК разработана программа nwFlash. Программа nwFlash позволяет:

- производить поиск подключенных к компьютеру по USB интерфейсу лабораторных стендов;
- активировать соединение с одним из найденных стендов;
- выполнять сброс микроконтроллера (Reset);
- загружать во flash - память микроконтроллера пользовательскую программу;
- принимать и отправлять данные в текстовом и шестнадцатеричном виде по интерфейсу UART (режим терминала).

Микроконтроллер ADuC842 содержит ядро 8052 с одноцикловыми командами (с быстродействием до 16MIPS, Макс част. 16.7МГц), быстродействующий 12-разрядный АЦП с частотой выборки до 400К в секунду, 62КБ FLASH/EE внутренней памяти программ, 4КБ FLASH/EE внутренней памяти данных. Сохранность Flash/EE 100 лет, максимальное число циклов программирования 100К 2304 байт внутренней памяти данных — ОЗУ (RAM). МК поставляются в 52-контактных корпусах PQFP. Внешний кварцевый резонатор на 32КГц.

Датчик температуры DS18B20 это цифровой измеритель температуры, с разрешением преобразования 9 - 12 разрядов и функцией тревожного сигнала контроля за температурой. Параметры контроля могут быть заданы пользователем и сохранены в энергонезависимой памяти датчика. DS18B20 обменивается данными с микроконтроллером по однопроводной линии связи, используя протокол интерфейса 1-Wire. Питание датчик может получать непосредственно от линии данных, без использования внешнего источника. В этом режиме питание датчика происходит от энергии, запасенной на паразитной емкости. Диапазон измерения температуры составляет от -55 до +125 °С. Для диапазона от -10 до +85 °С погрешность не превышает 0,5 °С.

У каждой микросхемы DS18B20 есть уникальный серийный код длиной 64 разряда, который позволяет нескольким датчикам подключаться на одну общую линию связи. Т.е. через один порт микроконтроллера можно обмениваться данными с несколькими датчиками, распределенными на значительном расстоянии. Режим крайне удобен для использования в системах экологического контроля, мониторинга температуры в зданиях, узлах оборудования.

Часы реального времени PCF8583 – микросхема выполняет функции часов реального времени, таймера, счетчика событий и статического ОЗУ емкостью 240 байт. Для передачи данных используется шина I2C. Микросхема изготавливается в 8-ми выводном корпусе. Фактически микросхема PCF8583 представляет собой статическое ОЗУ, емкостью 256 байт, у которого первые 16 байт являются регистрами специального назначения. Назначение некоторых регистров зависит от режима работы микросхемы. Режим задается в регистре состояния имеющем адрес 00h. Все числа в регистрах часов по умолчанию хранятся в BCD формате.

Инфракрасный фотоприемник TSOP1736 - устройство содержит в своем корпусе приемник, усилитель сигнала и демодулятор, на выходе TSOP1736 получаем стандартный сигнал величиной в пределах 0.3-6В. TSOP1736 часто применяется в системах дистанционного управления (в основном в бытовой технике), работает на несущей частоте 36 кГц.

Матричная клавиатура 4x3 содержит 12 клавиш для ввода информации.

Жидкокристаллический символьный индикатор 8x2 предназначен для отображения информации и имеет 2 строки по 8 символов.

Синхронный последовательный порт I²C Интерфейс I2C является двухпроводным последовательным интерфейсом, разработанным фирмой Philips. Изначально, в стандартном режиме шина была предназначена для скорости передачи данных до 100 кбит/с. В усовершенствованном быстром режиме поддерживается передача данных со скоростью до 400 кбит/с. На одной шине одновременно могут работать устройства и стандартного, и быстрого режима.

Четыре красных светодиода предназначены для вывода информации в виде четырехразрядного двоичного кода.

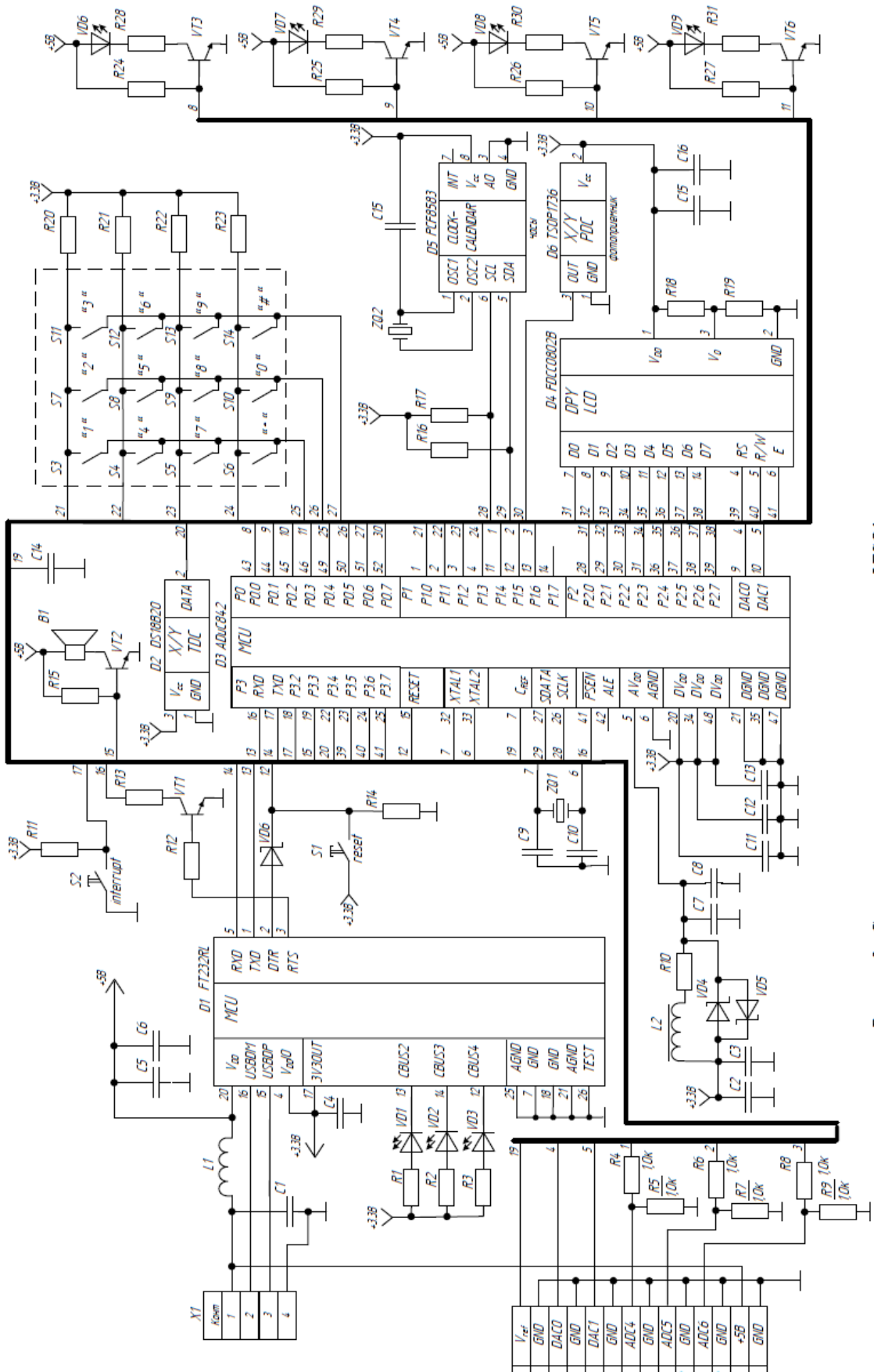


Рисунок 3 - Схема электрическая принципиальная стенда LESO1

Задание

Изучить принципиальную схему МПС (назначение устройств и принцип их соединения, рис.3) и нарисовать обобщенную структурную схему МПС (рис.1) с указанием типовых устройств, входящих в состав МПС. Заполнить таблицу 1. Ответить на контрольные вопросы.

Таблица 1.

Компонент МПС	Функциональное назначение
ADuC 842	
Символьный ЖКИ	
Светодиоды	
Фотоприемник	
Матричная клавиатура	
Термодатчик	
Часы реального времени	
Последовательный порт	

Контрольные вопросы

1. Каково назначение кнопки S1?
2. Каково назначение ИМС D3?
3. К каким портам микроконтроллера (МК) подключены строки и столбцы матричной клавиатуры?
4. К каким портам МК подключены светодиодные индикаторы VD6-VD9?
5. Какое позиционное обозначение на принципиальной схеме соответствует ЖКИ?
6. К каким портам МК подключены управляющие и информационные линии ЖКИ?
7. Какое позиционное обозначение на принципиальной схеме соответствует часам?
8. Какое позиционное обозначение на принципиальной схеме соответствует фотоприемнику?
9. К каким портам МК подключены часы реального времени?
10. К каким портам МК подключен термодатчик?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Перечень обязательных компонентов МПС и их назначение (таблица 1).
3. Структурная схема МПС
4. Ответы на контрольные вопросы и вывод по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Практическое занятие №2.

Изучение устройства параллельных портов МК ADuC842

Цель работы: изучить особенности работы параллельных портов микроконтроллера, а именно работу со светодиодными индикаторами и способами управления ими.

Выполнив работу, Вы будете:

уметь:

- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;

- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

не требуется.

Теоретические сведения

1. Устройство параллельных портов микроконтроллера

Параллельны порты предназначены для обмена многоразрядной двоичной информацией между микроконтроллером и внешними устройствами, при этом в качестве внешнего устройства может использоваться другой микроконтроллер. Каждый из портов содержит восьмиразрядный регистр, имеющий байтовую и битовую адресацию для установки (запись —1) или сброса (запись —0) разрядов этого регистра с помощью программного обеспечения. Выходы этих регистров соединены с внешними ножками микросхемы. С точки зрения внешнего устройства порт представляет собой обычный источник или приемник информации со стандартными цифровыми логическими уровнями (обычно ТТЛ), а с точки зрения микропроцессора — это ячейка памяти, в которую можно записывать данные или в которой сама собой появляется информация.

В качестве внешнего устройства может служить любой объект управления или источник информации (различные кнопки, датчики, микросхемы, дополнительная память, исполнительные механизмы, двигатели, реле и так далее).

В зависимости от направления передачи данных параллельные порты называются портами ввода, вывода или портами ввода вывода.

В качестве простейшего порта вывода может быть использован параллельный регистр, так как он позволяет запоминать данные, выводимые микропроцессором и хранить их до тех пор, пока подается питание. Все это время сигналы с выхода этого регистра поступают на внешнее устройство. Упрощенная функциональная схема порта вывода показана на рисунке 2. Двоичные слова с системной шины данных микропроцессора записываются в регистр по сигналу —WR||. Выходы "Q" регистра могут быть использованы как источники логических уровней для управления внешними устройствами. Этот регистр называется регистром данных порта вывода.

Для отображения этого регистра только в одну ячейку памяти адресного пространства микропроцессорного устройства в составе порта ввода-вывода всегда присутствует дешифратор адреса. Этот регистр называется регистром данных порта вывода. На выходе дешифратора адреса устанавливается единица лишь тогда, когда двоичное слово на его входе имеет заданное строго определенное значение, соответствующее адресу ячейки памяти, во всех остальных случаях на выходе дешифратора адреса удерживается логический ноль.

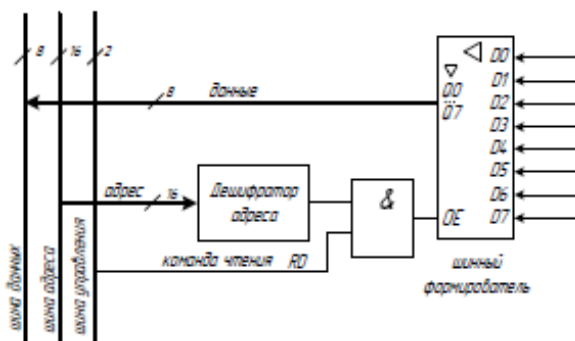
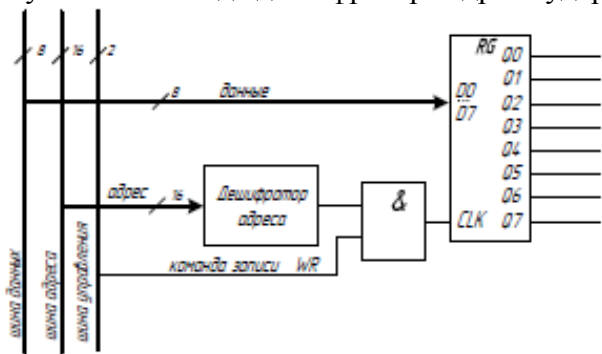


Рис. 2 – Функциональная схема порта вывода Рис. 3 – Функциональная схема порта ввода

В качестве порта ввода может быть использован шинный формирователь. Для построения порта ввода выходы шинного формирователя (Q0-Q7) подключены к внутренней шине данных микропроцессора, а на его вход подключаются сигналы, которые нужно ввести в микропроцессорную систему. Упрощенная функциональная схема порта ввода приведена на рисунке 3.

Шинный формирователь работает следующим образом: данные с входов D0-D7 поступают на выходы Q0-Q7 лишь тогда когда на входе OE (Output Enable — разрешение выхода) установлен высокий логический уровень (1), когда же на OE логический ноль выходы переходят в третье состояние и никак не влияют на шину данных. Значение сигнала с внешнего вывода порта передается на шину данных (считывается) по управляющему сигналу RD.

Для отображения шинного формирователя порта ввода в один адрес пространства адресов микроконтроллера используется дешифратор адреса. Выделяемый дешифратором адрес называют адресом регистра данных порта ввода. Из порта ввода возможно только чтение информации.

Так как из порта ввода возможно только чтение информации (команд RD), а в порт вывода только запись (команда WR), то для портов ввода и вывода можно отвести один и тот же адрес в адресном пространстве микропроцессора. Упрощенная схема одного разряда параллельного порта ввода-вывода приведена на рисунке 4. Подобное схемотехническое решение применяется в микроконтроллерах архитектуры MCS-51, такой порт называется квазидвунаправленным. Для микроконтроллеров других архитектур схема может отличаться.

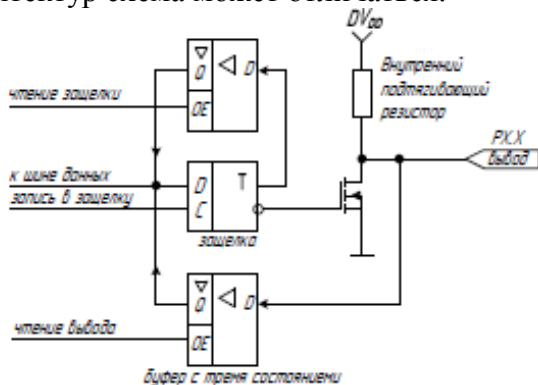


Рисунок 4 – Упрощенная схема одного бита параллельного квазидвунаправленного порта

Один разряд регистра порта представляет собой D-триггер, запись входных данных в который происходит по высокому уровню синхросигнала (вход триггера — Cl). На рисунке такой сигнал назван «запись в защелку». Мощность сигнала с инвертирующего выхода триггера усиливается при помощи МОП (металл — окись — полупроводник) транзистора, а за тем поступает на внешний вывод микросхемы. Внутренний подтягивающий резистор служит для обеспечения выходного тока порта при установленном высоком логическом уровне. Как правило, вместо резистора в портах микроконтроллера используется управляемый генератор стабильного тока (ГСТ), собранный на МОП транзисторах. Следует обратить внимание, что у некоторых портов вывода, внутреннего генератора тока может и не быть, пример тому P0.

2. Подключение внешних устройств к порту параллельному порту микроконтроллера

Внешними устройствами называются любые устройства, которыми управляет, от которых получает или которым передает информацию микроконтроллер. В качестве внешних устройств могут выступать устройства ввода-вывода информации — светодиодный индикатор, дисплей, датчик, кнопка, клавиатура, другой микропроцессор.

Внутренняя схема порта построена таким образом, чтобы максимально упростить подключение внешних устройств к микроконтроллеру. Присутствие в схеме порта выходного транзистора позволяет подключить к выводам порта светодиодные индикаторы непосредственно, без усилителя мощности, как это показано на рисунке 5. При этом необходимо следить за максимальной допустимой мощностью, рассеиваемой на микросхеме и отдельных выводах порта. Светодиод в такой схеме загорается при низком потенциале на выводе порта, для этого в порт должен быть записан логический ноль. Резистор R1 ограничивает ток через светодиод, если этот резистор не поставить, то ток по цепи индикатора может достигнуть недопустимой величины и светодиод или внутренний транзистор выйдут из строя.

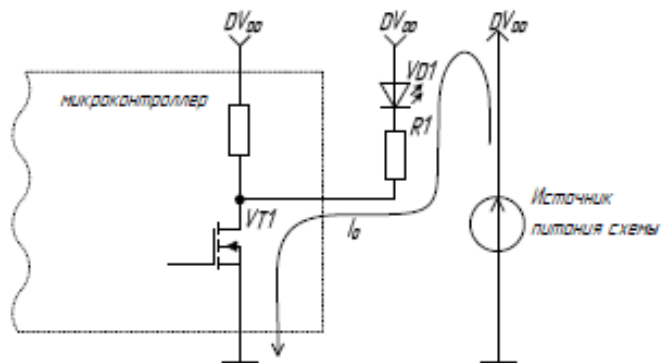


Рисунок 5 – Эквивалентная схема подключения светодиодного индикатора к параллельному порту

Если же требуется обеспечить больший ток нагрузки, то для усиления тока порта используют внешний транзистор, как показано на рисунке 6.

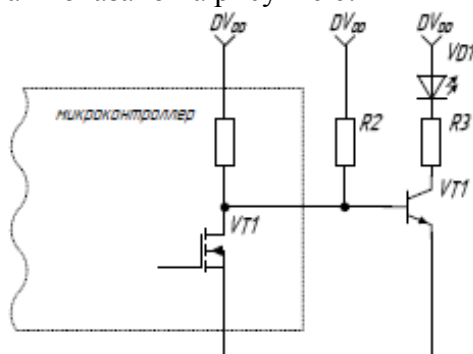


Рисунок 6 – Схема подключения светодиодного индикатора через транзисторный ключ

В приведенной схеме база транзистора подключена непосредственно к выводу порта. Если выходного порта контроллера достаточно для открывания транзистора, то резистор R2 может быть исключен из схемы. Этот резистор нужен для увеличения тока базы транзисторного ключа. Резистор R3 рассчитывается исходя из допустимого тока светодиода. Для зажигания светодиода необходимо в соответствующий разряд параллельного порта записать логическую единицу, а для его гашения — логический ноль.

Двунаправленный порт позволяет получать цифровую информацию от различных источников — кнопок, датчиков, микросхем. Согласование микросхем между собой не представляет трудностей, так как практически все современные цифровые микросхемы по входу и выходу согласованы между собой и имеют строго определенные значения логических уровней. С датчиками и кнопками дело обстоит несколько сложнее. Все датчики выполняются так, что они с точки зрения электрической схемы представляют собой контакты, работающие на замыкание-размыкание. Поэтому схема подключения датчиков и кнопок принципиально не отличаются. Со стороны микроконтроллера замыкание-размыкание должно интерпретироваться как подача на вход логических уровней. Простейшая схема, преобразующая замыкание контактов в логические уровни показана на рисунке 7.

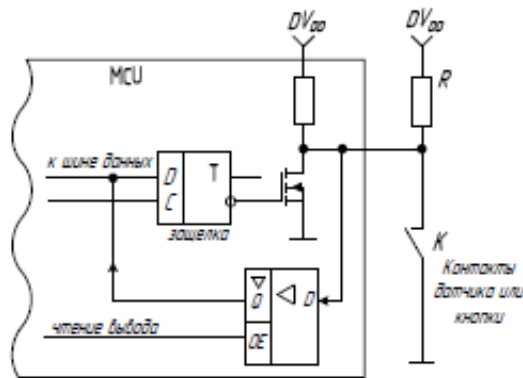


Рисунок 7 – Подключение источника цифровой информации

Когда контакт К разомкнут то на вход микроконтроллера через резистор R подается напряжение питания, интерпретируемое контроллером как логическая единица. Если контакт замкнут, то на вход подается потенциал общего провода, что соответствует логическому нулю.

Для того что бы ввести информацию с параллельного порта в него должна быть записаны логические единицы. Дело в том, что если в разряд записать логическую единицу, то выходной транзистор закрывается и на выводе микросхемы за счет внутреннего подтягивающего резистора устанавливается высокий уровень. Этот уровень может быть изменен на нулевой потенциал замыканием вывода микросхемы на общий провод. И информация может быть правильно интерпретирована микропроцессором. В случае же когда в разряд порта записан логический ноль, внутренний транзистор открыт, на выходе появляется низкий потенциал, изменить который извне невозможно. Поэтом, перед тем как осуществить ввод информации через параллельный порт, соответствующий разряд необходимо настроить на ввод — записать в него логическую единицу.

3. Особенность параллельных портов микроконтроллера ADuC842

В микроконтроллере ADuC842, фирмы Analog Devices, для обмена информации с внешними устройствами используется четыре параллельных порта ввода-вывода. Дополнительно к своему основному назначению некоторые порты могут использоваться для подключения внешней памяти.

Port0 (P0.0 – P0.7) — Двухнаправленный 8-ми разрядный параллельный порт ввода-вывода с открытым стоком. При записи в бит регистра порта логической единицы соответствующая линия порта переходит в режим высокоимпедансного входа. Для работы в режиме порта ввода-вывода необходимо внешнее подтягивание каждого разряда порта к уровню логической единицы.

Port1 (P1.0 – P1.7) — Входной порт, по умолчанию настроен на ввод аналоговых сигналов (функция АЦП). Каждый ввод может быть переведен в режим цифрового входа, для этого в соответствующий бит порта должен быть записан логический ноль. Порт не имеет внутреннего усиливающего транзистора и потому при вводе дискретной информации через него не требуется записывать в разряды логическую единицу. В режиме порта цифрового ввода необходимо внешнее подтягивание каждого разряда порта к уровню логической единицы.

Port2(P2.0 – P2.7), Port3(P3.0 – P3.7) — Двухнаправленные 8-ми разрядные параллельные порты ввода-вывода. При записи в бит регистра порта логической единицы, соответствующая линия порта переходит в режим высокоимпедансного входа со слабым подтягиванием сигнала к уровню логической единицы.

Формат и адреса портов P0-P3 приведены на рисунке 8. На этом же рисунке показаны адреса отдельных битов портов в битовом пространстве памяти.

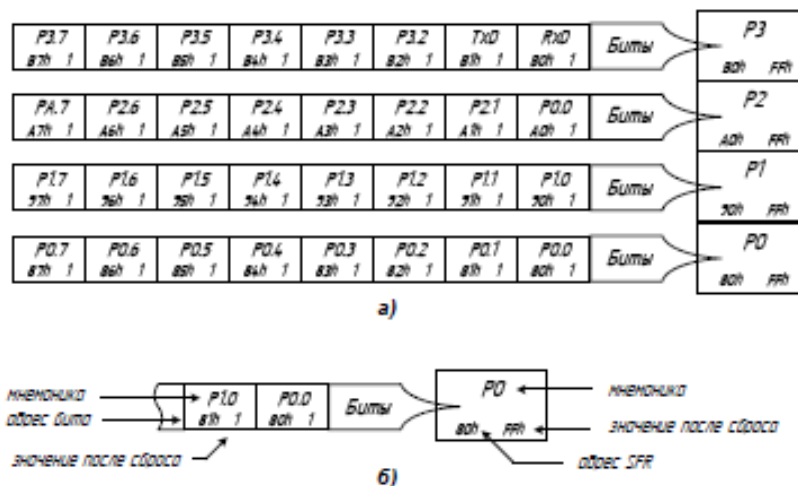


Рисунок 8 – Адреса SFR параллельных портов

Микроконтроллер ADuC842 в лабораторном практикуме исследуется в составе учебного лабораторного стенда LESO1, для того чтобы определить какое периферийное устройство подключено к какому порту следует изучить принципиальную схему учебного стенда.

Задание

Зарисовать в тетрадь функциональные схемы портов ввода-вывода, а также схемы подключения внешних устройств к параллельным портам (индикатор и кнопка), знать назначение всех компонентов.

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Эквивалентная схема подключения светодиода к параллельному порту.
3. Эквивалентная схема подключения источника цифрового сигнала к параллельному порту.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Практическое занятие №3.

Изучение схемы подключения матричной клавиатуры к МК ADuC842

Цель работы: изучить особенности работы параллельных портов микроконтроллера, а также изучить схемы подключения кнопок и матричной клавиатуры к микроконтроллеру.

Выполнив работу, Вы будете:

уметь:

- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;

- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

не требуется.

Теоретические сведения

Применение матричной клавиатуры для ввода информации в микропроцессорную систему

Для реализации взаимодействия пользователя с микропроцессорной системой используют различные устройства ввода-вывода информации. В самом простом случае в роли устройства ввода может выступать кнопка, представляющая собой элементарный механизм, осуществляющий замыкание-размыкание контактов под действием внешней механической силы. Схема подключения кнопки к линии ввода параллельного порта ввода микроконтроллера показана на рисунке 9. Когда контакты кнопки S1 разомкнуты через резистор R1 на вход контроллера поступает высокий логический уровень —1, когда же контакты замкнуты, то вход оказывается соединенным с общим проводом, что соответствует логическому уровню —0. Если параллельный порт микроконтроллера имеет встроенный генератор тока, то в схеме можно обойтись без резистора R1.

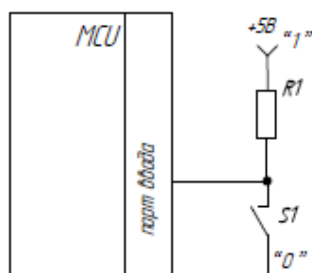


Рисунок 9 – Подключение одиночной кнопки к параллельному порту

Недостаток приведенной схемы заключается в том, что для подключения каждой кнопки требуется отдельная линия параллельного порта. Так как часто требуется вводить информацию с большого количества кнопок, то для уменьшения количества линий ввода-вывода используется клавиатура, представляющая собой двухмерную матрицу кнопок, организованных в ряды и столбцы (рисунок 10).

Подключение клавиатуры отличается от схемы подключения одиночной кнопки тем, что потенциал общего провода на опрашиваемые кнопки подается не непосредственно, а через порт вывода.

В каждый момент времени сигнал низкого уровня (логический ноль) подается только на один столбец кнопок, на остальные должна подаваться логическая единица. Это исключит неоднозначность определения номера нажатой кнопки. Двоичные сигналы, присутствующие при этом на строках клавиатуры, считываются через порт ввода микроконтроллера.

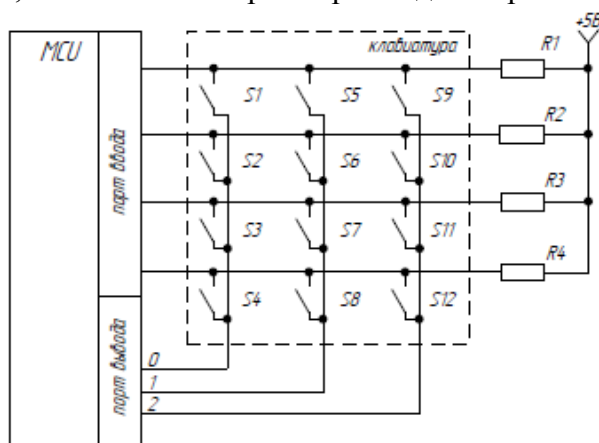


Рисунок 10 – Подключение матричной клавиатуры к параллельному порту

Временная диаграмма напряжений на портах вывода при выполнении программы опроса клавиатуры приведена на рисунке 11.

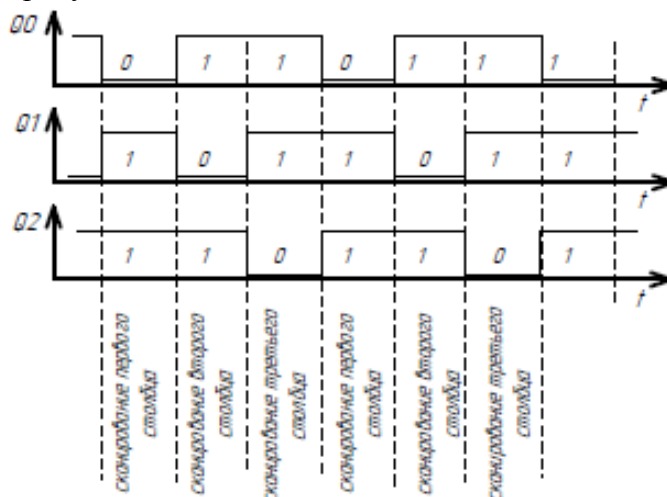


Рисунок 11 – Временные диаграммы работы порта вывода

В каждый момент времени производится чтения информации из порта ввода. Программа микроконтроллера по считанной комбинации должна определить номер нажатой кнопки клавиатуры.

Задание

Зарисовать в тетрадь схемы подключения одиночной кнопки и матричной клавиатуры к параллельным портам (индикатор и кнопка), знать назначение всех компонентов.

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Эквивалентная схема подключения кнопки к параллельному порту.
3. Эквивалентная схема подключения клавиатуры к параллельному порту.

Критерии оценки:

- Оценка «отлично» ставится, если задание выполнено верно и полностью.
- Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.
- Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.
- Оценка «неудовлетворительно» ставится, если задание не выполнено.

Практическое занятие №4.

Изучение схемы подключения ЖКИ к МК ADuC842

Цель работы: изучить особенности организации работы с ЖКИ

Выполнив работу, Вы будете:

уметь:

- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

не требуется.

Теоретические сведения

1. Устройство и принцип работы символьного жидкокристаллического индикатора

В настоящее время в микропроцессорных системах для отображения широко используют жидкокристаллические индикаторы (ЖКИ). Условно все ЖКИ можно разделить на две категории: символьные, или знаковосинтезирующие, и графические. Графические индикаторы представляют собой матрицу из m строк и n столбцов, на пересечении которых находятся пиксели. Пиксель представляет собой неделимый объект прямоугольной или круглой формы, обладающий определённым цветом; пиксель – наименьшая единица растрового изображения. Если на определенный столбец и строку подать электрический сигнал, то пиксель на их пересечении изменит свой цвет. Подавая группу сигналов на столбцы и строки можно формировать по точкам произвольное графическое изображение. Так работает графический ЖКИ. В символьном же ЖКИ матрица пикселей разбита на подматрицы, каждая подматрица предназначена для формирования одного символа: цифры, буквы или знака препинания. Как правило, для формирования одного символа используют матрицу из восьмистрок и пяти столбцов. Символьные индикаторы бывают одно-, двух- и четырехстрочными.

Для упрощения взаимодействия микропроцессорной системы и ЖКИ используют специализированную микросхему – контроллер (драйвер) ЖКИ. Он управляет пикселями жидкокристаллического дисплея и интерфейсной частью индикатора. Обычно такой контроллер входит в состав индикатора. В целом жидкокристаллический индикатор представляет собой печатную плату, на которой смонтирован сам дисплей, контроллер и необходимые дополнительные электронные компоненты. Внешний вид ЖКИ показан на рисунке ниже.



Рисунок 20 – Внешний вид жидкокристаллического индикатора

В учебном стенде LESO1 использован двухстрочный восьмисимвольный ЖКИ. Структурная схема показана на рисунке 21.

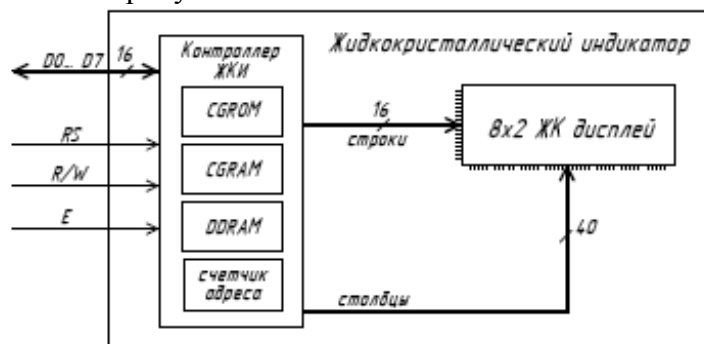


Рисунок 21 – Структурная схема ЖКИ

В состав контроллера ЖКИ входят три вида памяти: **CGROM**, **CGRAM**, **DDRAM**. Когда микроконтроллер передает в контроллер ЖКИ ASCII-коды символов, то они записываются в **DDRAM** (Display data RAM – ОЗУ ASCII-кодов отображаемых символов), такую память называют **видеопамятью** или **видеобуфером**. Videобуфер в символьных индикаторах обычно содержит 80 ячеек памяти – больше, чем число знакомест дисплея. У двухстрочных индикаторов ячейки с адресами от 0x00 и до 0x27 отображаются на верхней строке дисплея, а ячейки с адресами 0x40 ... 0x67 – на нижней строке. Смещая видимое окно дисплея относительно DDRAM, можно отображать на дисплее различные области видеопамяти. Сдвиг окна индикатора относительно видеобуфера для верхней и нижней строк происходит синхронно, как это показано

на рисунке 22. Курсор будет виден на индикаторе только в том случае, если он попал в зону видимости дисплея (и если предварительно была подана команда отображать курсор).

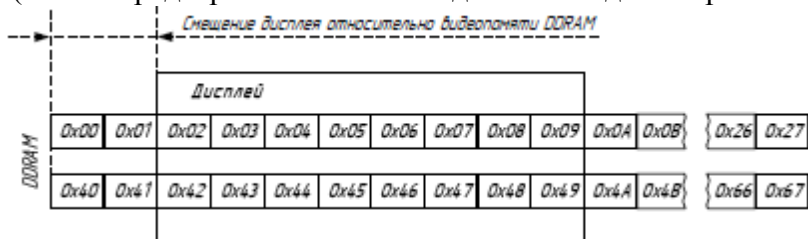


Рисунок 22 – Отображение символов из видеобuffers

Матрицы начертания символов хранятся в памяти знакогенератора. Память знакогенератора включает в себя **CGROM** (Character generator ROM – ПЗУ знакогенератора), в которую на заводе-изготовителе загружены начертания символов таблицы ASCII. Содержимое CGROM изменить нельзя. Для того, чтобы пользователь смог самостоятельно задать начертание нужных ему символов, в знакогенераторе имеется специальное ОЗУ – **CGRAM** (Character generator RAM). Под ячейки CGRAM отведены первые (младшие) 16 адресов таблицы кодов.

Схема подключения ЖКИ к микроконтроллеру ADuC842 показана ниже на рисунке:

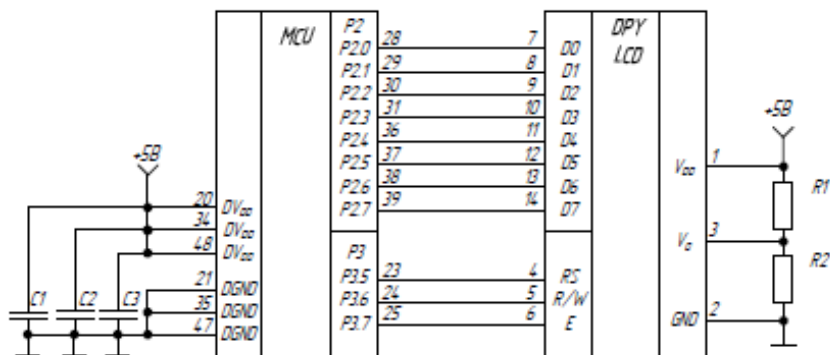


Рисунок 23 – Схема подключения ЖКИ к микроконтроллеру

Интерфейс подключения – параллельный. Для соединения индикатора с микроконтроллером используется 11 линий — восемь для передачи данных (**D0 - D7**) и три линии управления. Линия **RS** служит для сообщения контроллеру индикатора о том, что именно передается по шине: команда или данные (**RS = 1** — данные, **RS = 0** — команда). По линии **E** передается строб- сигнал, сопровождающий запись или чтение данных: по переходу сигнала на линии **E** из 1 в 0 осуществляется запись данных во входной буфер микроконтроллера индикатора. Запись информации в ЖКИ происходит по спаду этого сигнала. Потенциал на управляющем выводе **R/W (Read/Write)** задает направление передачи информации, при **R/W = 0** осуществляется запись в память индикатора, при **R/W = 1** – чтение из нее. Еще три линии предназначены для подачи питающего напряжения (**VDD, GND**) и напряжения смещения, которое управляет контрастностью дисплея.

Диаграммы передачи данных от управляющего микроконтроллера к контроллеру ЖКИ и от контроллера ЖКИ в управляющий микроконтроллер показаны на рисунках 24 и 25 соответственно.

После приема информации контроллеру ЖКИ требуется некоторое время на выполнение команд, в это время управляющий контроллер не должен давать следующую команду или пересылать данные.

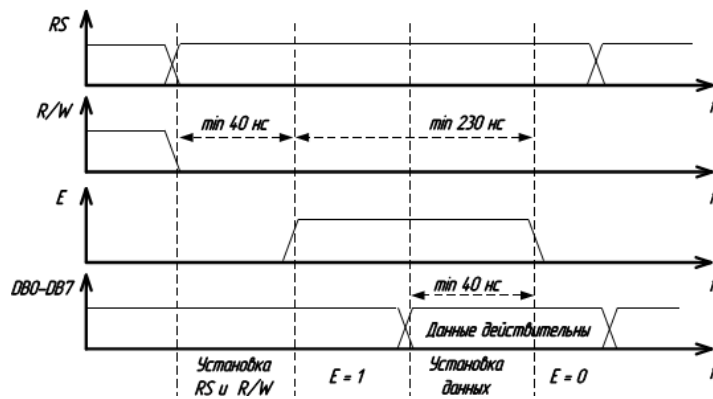


Рисунок 24 – Диаграмма передачи информации контроллеру ЖКИ

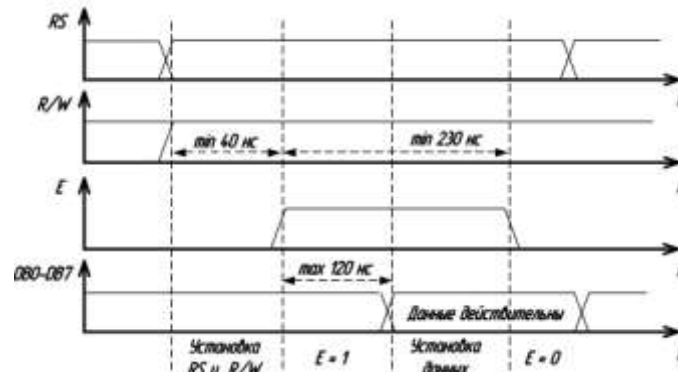


Рисунок 25 – Диаграмма чтения информации из контроллера ЖКИ

В таблице 8 приведены команды контроллера ЖКИ и время, необходимое для выполнения этих команд. Для того чтобы можно было определить, когда ЖКИ закончит свои внутренние операции, контроллер ЖКИ содержит специальный флаг занятости – **BUSY-флаг (BF)**. Если контроллер занят выполнением внутренних операций, то **BF** установлен (**BF = 1**), если же контроллер готов принять следующую команду, то **BF** сброшен (**BF = 0**). Более простой способ организации обмена заключается в том, что управляющий микроконтроллер, зная, сколько времени требуется ЖКИ на обработку той или иной команды, после каждой передачи информации ждет соответствующее время.

Таблица 8 – Команды контроллера ЖКИ

Команда	Код										Описание	Время исполнения команды
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0		
Очистка дисплея	0	0	0	0	0	0	0	0	0	1	Записывает код 0x20 (пробел) во все ячейки DDRAM, устанавливает счетчик адреса DDRAM в 0x00.	1,5 мс
Возврат в начальную позицию	0	0	0	0	0	0	0	0	1	x	Устанавливает счетчик адреса DDRAM в 0x00 и возвращает курсор в начальную позицию. Содержимое DDRAM не изменяется.	1,5 мс
Установка режима ввода	0	0	0	0	0	0	0	1	I/D	SH	Задаёт направление перемещения курсора (I/D) и разрешает сдвиг сразу всех символов (SH).	38 мкс
Включение-выключение дисплея	0	0	0	0	0	0	1	D	C	B	Устанавливает/отключает биты, отвечающие за включения дисплея (D), отображение курсора (C), мерцание курсора (B).	38 мкс
Сдвиг курсора или видимой области дисплея	0	0	0	0	0	1	D/C	R/L	x	x	Бит D/C определяет то, что будет перемещаться – видимая область дисплея или курсор (при D/C = 1 перемещается видимая область, при D/C = 0 – курсор), R/L задает направление перемещения. DDRAM не изменяется.	38 мкс
Начальные установки	0	0	0	0	1	DL	N	F	x	x	Определяет разрядность шины интерфейса (DL = 1 – 8-бит, DL = 0 – 4-бита), количества строк на дисплее (N = 1 – две строки, N = 0 – одна строка) и размера символов (F = 1 – 5×11 точек, F = 0 – 5×8 точек).	38 мкс
Установка адреса CGRAM	0	0	0	1	A5	A4	A3	A2	A1	A0	Установка счетчика адреса CGRAM	38 мкс
Установка адреса DDRAM	0	0	1	A6	A5	A4	A3	A2	A1	A0	Установка счетчика адреса DDRAM	38 мкс
Чтение BF и счетчика адреса	0	1	BF	A6	A5	A4	A3	A2	A1	A0	Если BF = 1 то контроллер ЖКИ выполняет внутреннюю операцию. A6-A0 – текущее значение адреса DDRAM.	0
Запись данных в RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Запись данных в ОЗУ (DDRAM или CGRAM)	38 мкс
Чтение данных из RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Чтение данных из ОЗУ (DDRAM или CGRAM)	38 мкс

Задание

Зарисовать в тетрадь структурную схему ЖКИ и схему подключения ЖКИ к МК, временные диаграммы обмена информацией между ЖКИ и МК.

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Структурная схема ЖКИ.
3. Схема подключения ЖКИ к МК
4. Временные диаграммы работы с ЖКИ

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Практическое занятие №5.

Изучение ассемблера и системы команд МК AVR

Цель работы: изучить основные понятия ассемблера и систему команд МК AVR

Выполнив работу, Вы будете:

уметь:

- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

не требуется

Теоретические сведения

Основные понятия

Ассемблер - это инструмент, с помощью которого создаётся программа для микроконтроллера. Ассемблер транслирует ассемблируемый исходный код программы в объектный код, который может использоваться в симуляторах или эмуляторах AVR. Также ассемблер генерирует код, который может быть не посредственно введен в программную память микроконтроллера.

При работе с ассемблером нет никакой необходимости в непосредственном соединении с микроконтроллером.

Язык ассемблера - это символическое представление машинного языка. Все процессы в машине на самом низком, аппаратном уровне приводятся в действие только командами (инструкциями) машинного языка. Отсюда понятно, что, несмотря на общее название, язык ассемблера для каждого типа компьютера свой.

Программа на ассемблере представляет собой совокупность блоков памяти, называемых сегментами памяти. Программа может состоять из одного или нескольких таких блоков-сегментов. Каждый сегмент содержит совокупность предложений языка, каждое из которых занимает отдельную строку кода программы.

Предложения ассемблера бывают четырех типов:

- 1) команды или инструкции, представляющие собой символические аналоги машинных команд. В процессе трансляции инструкции ассемблера преобразуются в соответствующие команды системы команд микропроцессора;
- 2) макрокоманды - оформляемые определенным образом предложения текста программы, замещаемые во время трансляции другими предложениями;
- 3) директивы, являющиеся указанием транслятору ассемблера на выполнение некоторых действий. У директив нет аналогов в машинном представлении;
- 4) строки комментариев, содержащие любые символы, в том числе и буквы русского алфавита. Комментарии игнорируются транслятором.

Предложения, составляющие программу, могут представлять собой синтаксическую конструкцию, соответствующую команде, макрокоманде, директиве или комментарию. Для того чтобы транслятор ассемблера мог распознать их, они должны формироваться по определенным синтаксическим правилам. Для этого лучше всего использовать формальное описание синтаксиса языка наподобие правил грамматики.

Исходный файл, с которым работает ассемблер, должен содержать мнемоники, директивы и метки.

Мнемоника – это краткое буквенное обозначение команды:

СЛОЖЕНИЕ → ADDITION → ADD

Перед каждой строкой программы можно ставить метку, которая является алфавитно-цифровой строкой, заканчивающейся двоеточием. Метки используются как указание для безусловного перехода и команд условного перехода.

Строка программы может быть в одной из четырех форм:

[Метка:] директива [операнды] [Комментарий]

[Метка:] команда [операнды] [Комментарий]

Комментарий

Пустая строка

Символы квадратной скобки [...] означают, что использование элемента необязательно.

Практически каждое предложение содержит описание объекта, над которым или при помощи которого выполняется некоторое действие. Эти объекты называются *операндами* - это объекты (некоторые значения, регистры или ячейки памяти), на которые действуют инструкции или директивы, либо это объекты, которые определяют или уточняют действие инструкций или директив.

Комментарий имеет следующую форму:

; [Текст]

Таким образом любой текст после символа “ ; ” игнорируется ассемблером и имеет значение только для пользователя.

Операнды можно задавать в различных форматах:

десятичный: 10,255

шестнадцатеричный (два способа): 0x1a или \$1a

двоичный: 0b00001010, 0b11111111

восьмеричный: 010, 077

Символы языка ассемблера

Допустимыми символами при написании текста программ являются:

- 1) все латинские буквы: A-Z, a-z. При этом заглавные и строчные буквы считаются эквивалентными;
- 2) цифры от 0 до 9;
- 3) знаки ?, @, \$, _, &;
- 4) разделители , . [] () < > { } + / * % ! ' " ? = # ^.

Предложения ассемблера формируются из лексем, представляющих собой синтаксически неразделимые последовательности допустимых символов языка, имеющие смысл для транслятора.

Директивы ассемблера

Ассемблер поддерживает множество директив. Директивы не транслируются непосредственно в коды операции. Напротив, они используются, чтобы корректировать местоположение программы в памяти, определять макрокоманды, инициализировать память и так далее. То есть это указания самому ассемблеру, а не команды микроконтроллера.

Все директивы ассемблера приведены в табл. 1.

Таблица 1. Директивы ассемблера

Директива	Описание
BYTE	Зарезервировать байт под переменную
CSEG	Сегмент кодов
DB	Задать постоянным(и) байт(ы) в памяти
DEF	Задать символическое имя регистру
DEVICE	Задать для какого типа микроконтроллера компилировать
DSEG	Сегмент данных
DW	Задать постоянное(ые) слово(а) в памяти
EQU	Установите символ равный выражению
ESEG	Сегмент EEPROM
EXIT	Выход из файла
INCLUDE	Включить исходный код из другого файла
LIST	Включить генерацию .lst - файла
NOLIST	Выключить генерацию .lst - файла
ORG	Начальный адрес программы
SET	Установите символ равный выражению

Синтаксис всех директив следующий:

. <директива>

То есть перед директивой должна стоять точка. Иначе ассемблер воспринимает это как метку.

Поясим наиболее важные директивы ассемблера.

1. *CSEG- Codesegment* - указывает на начало сегмента кодов. Ассемблируемый файл может иметь несколько кодовых сегментов, которые будут объединены в один при ассемблировании.

Синтаксис:

- CSEG

Пример:

```
.DSEG          ; Начало сегмента данных
vartab: .BYTE 4 ; Резервируется 4 байта в СОЗУ
.CSEG         ; Начало сегмента кодов
const:  .DW 2   ; Записать 0x0002 в программной памяти
mov r1, r0    ; Что-то делать
```

2. *DSEG – DataSegment* - указывает на начало сегмента данных. Ассемблируемый файл может содержать несколько сегментов данных, которые потом будут собраны один при ассемблировании. Обычно сегмент данных состоит лишь из директивы BYTE и меток.

Синтаксис:

- DSEG

Пример:

```
.DSEG          ; Начало сегмента данных
var1: .BYTE 1  ; Резервировать 1 байт под переменную
table: .BYTE tab_size ; Резервировать tab_size байтов.
.CSEG
ldi r30, low(var1)
ldi r31, high(var1)
ld r1, Z
```

3. *ESEG* - *EEPROMSegment* - указывает на начало сегмента EEPROM памяти. Ассемблируемый файл может содержать несколько EEPROM сегментов, которые будут собраны в один сегмент при ассемблировании. Обычно сегмент EEPROM состоит из DB и DW директив (и меток). Сегмент EEPROM памяти имеет свой собственный счетчик. Директива ORG может использоваться для размещения переменных в нужной области EEPROM.

Синтаксис:

- ESEG

Пример:

```
.DSEG                ; Начало сегмента данных
var1: .BYTE 1        ; Резервировать 1 байт под переменную
table: .BYTE tab_size ; Зарезервировать tab_size байт.
.ESEG
eevar1: .DW 0xffff   ; Записать 1 слово в EEPROM
```

4. *ORG* - *установить адрес начала программы* - присваивает значения локальным счетчикам. Используется только совместно с директивами .CSEG, .DSEG, .ESEG.

Синтаксис:

- ORG<адрес>

Пример:

```
.DSEG                ; Начало сегмента данных
.ORG 0x37            ; Установить адрес СОЗУ на 37h
variable: .BYTE 1    ; Зарезервировать байт СОЗУ по адресу 37h
.CSEG
.ORG 0x10            ; Установить счетчик команд на адрес 10h
mov r0,r1            ; Чего-нибудь делать
```

5. *DB* – *определить байт(ы) в программной памяти или в EEPROM*- резервирует ресурсы памяти в программной памяти или в EEPROM. Директиве должна предшествовать метка. DB задает список выражений, и должна содержать по крайней мере одно выражение. Размещать директиву следует в сегменте кодов или в EEPROM сегменте. Список выражений представляет собой последовательность выражений, разделенных запятыми. Каждое выражение должно быть величиной между -128 и 255. Если директива указывается в сегменте кодов и список выражений содержит более двух величин, то выражения будут записаны так, что 2 байта будут размещаться в каждом слове Flash-памяти.

Синтаксис:

LABEL: .DB< список выражений>

Пример:

```
.CSEG
const1: .DB 0, 255, 0b01010101, -128, 0xaa
.ESEG
const2: .DB 1, 2, 3
```

6. *DW* – *определить слово в программной памяти или в EEPROM* - Директива DW резервирует ресурсы памяти в программной памяти или в EEPROM. Директиве должна предшествовать метка. DW задает список выражений, . и должна содержать по крайней мере одно выражение. Размещать директиву следует в сегменте кодов или в EEPROM сегменте. Список выражений представляет собой последовательность выражений, разделенных запятыми. Каждое выражение должно быть величиной между -32768 и 65535.

Синтаксис:

LABEL: .DW< список выражений>

Пример:

```
.CSEG
varlist: .DW 0, 0xffff, 0b1001110001010101, -32768, 65535
.ESEG
eevarlist: .DW 0, 0xffff, 10
```

7. *DEF* – присвоить имя регистру- позволяет присвоить символическое имя регистру. Регистр может иметь несколько символических имен.

Синтаксис:

`.DEF<Имя>-<Регистр>`

Пример:

```
.DEF temp=R16
.DEF ior=R0
.CSEG
ldi temp,0xf0      ; Загрузить 0xf0 в регистр temp
in ior,0x3f        ; Прочитать SREG в регистр ior
eor temp, ior      ; Выполнить операцию
```

8. *EQU* – присвоить имя выражению- директива присваивает значение метке. Эта метка может быть использована в других выражениях. Значение этой метки нельзя изменить или переопределить.

Синтаксис:

`.EQU<метка>=<выражение>`

Пример:

```
.EQU io_offset = 0x23
.EQU porta     = io_offset + 2
.CSEG          ; Начало сегмента кодов
clr r2        ; Очистить регистр r2
out porta,r2  ; Записать в порт A
```

9. *INCLUDE* – вставить другой файл - директива говорит Ассемблеру начать читать из другого файл. Ассемблер будет ассемблировать этот файл до конца файла или до директивы EXIT. Включаемый файл может сам включать директивы INCLUDE.

Синтаксис:

`.INCLUDE"<имя файла>"`

Пример:

```
.EQU sreg = 0x3f ; Регистр статуса
.EQU sphigh = 0x3e ; Старший байт указателя стека.
.EQU splow = 0x3d ; Младший байт указателя стека.
; incdemo.asm
.INCLUDE "iodefs.asm"; Включить файл «iodefs.asm»
in r0,sreg ; Прочитать регистр статуса
```

10. *EXIT* – выйти из файла – директива позволяет ассемблеру остановить ассемблирование текущего файла. Обычно ассемблер работает до конца файла. Если он встретит директиву EXIT, то продолжит ассемблировать со строки, следующей за директивой INCLUDE.

Синтаксис:

`.EXIT`

Пример:

`.EXIT ; выйти из этого файла`

11. *DEVICE* – указать для какого микроконтроллера ассемблировать - директива позволяет пользователю сообщить ассемблеру, для какого типа устройства пишется программа. Если ассемблер встретит команду, которая не поддерживается указанным типом микроконтроллера, то будет выдано сообщение. Также сообщение появится в случае, если размер программы превысит объем имеющейся в этом устройстве памяти.

Синтаксис:

```
.DEVICE AT90S1200 | AT90S2313 | AT90S2323 | AT90S2333 | AT90S2343 |
AT90S4414 | AT90S4433 | AT90S4434 | AT90S8515 | AT90S8534 |
AT90S8535 | ATtiny11 | ATtiny12 | ATtiny22 | ATmega603 |
ATmega103 | ATmega8535
```

Пример:

```
.DEVICE ATmega8535 ;использовать ATmega8535
.CSEG
.ORG 0000
jmp labell ;При ассемблировании появиться сообщение, что
;ATmega8535 не поддерживает команду jmp
```

Система команд микроконтроллеров ATMELCемейства AVRочень большая и в тоже время эффективная. Одной из отличительных особенностей микроконтроллеров AVRявляется то, что почти все команды выполняются за 1 тактовый цикл. Исключение составляют команды перехода. Это существенно увеличивает производительность микроконтроллера даже при относительно невысокой тактовой частоте.

Все команды можно классифицировать на 5 типов:

1. Арифметические команды.
2. Логические команды.
3. Команды перехода.
4. Команды передачи данных.
5. Побитовые команды и команды тестирования битов.

Система команд приведена **ПРИЛОЖЕНИИ 1**.

Некоторые особенности программирования

Память данных почти полностью доступна программе пользователя и большинство команд ассемблера предназначено для обмена данными с ней. Команды пересылки данных предоставляют возможность непосредственной и косвенной адресации ячеек СОЗУ, непосредственной адресации регистров ввода/вывода и регистров общего назначения. Так как каждому регистру сопоставлена ячейка памяти, то обращаться к ним можно не только командами адресации регистров, но и командами адресации ячеек СОЗУ.

Пример, команда:

То же самое относится и к регистрам ввода/вывода. Для них предусмотрены специальные команды:

```
MOV R10,R15 ; скопировать регистр R15 в регистр R10 делает
; абсолютно то же самое, что и команда:
LDS R10,$0015 ; загрузить в регистр R10 содержимое ячейки,с
; адресом $0015
```

При использовании этих команд номер порта указывается в диапазоне $0 < P < 63$. При использовании команд адресации ячеек памяти для работы с регистрами ввода/вывода указывается адрес регистра в памяти данных \$0020-\$005F.

Пример применения разных команд:

```
LDI R16,$FF
OUT $12,R16 ; записать в PORTD число 255
STS $0032,R16 ; записать непосредственно в ячейку $0032
```

Адрес регистра ввода/вывода в СОЗУ получается прибавлением к номеру порта числа \$20.

Памятью программ является ПЗУ и изменяется только при программировании кристалла. Константы можно располагать в памяти программ в виде слов.

Пример: .dw\$033f,\$676d,\$7653,\$237e,\$777f

Для работы с данными, расположенными в памяти программ, предусмотрена командаLPM-загрузить байт памяти программ, на который указывает регистр Zв регистрR0.

Адрес байта константы определяется содержимым регистра Z. Старшие 15 битов определяют слово адреса (от 0 до 4к) состояние младшего бита определяет выбор младшего

```
IN Rd,P ; загрузить данные из порта I/O с номером P
; в регистр Rd
OUT P,Rd ; записать данные из регистра Rd в порт I/O
; с номером P.
```

байта (0) или старшего байта (1).

При работе с портами ввода/вывода следует учитывать следующую особенность: если вывод порта сконфигурирован как выход, то его переключение производится через регистр данных (PORTA, PORTB, PORTC, PORTD), если вывод сконфигурирован как вход, то его опрос следует производить через регистр выводов входа порта (PINA, PINB, PINC, PIND).

Особенностью использования арифметических и логических команд является то, что некоторые из них работают только с регистрами R16-R31.

Пример:

CPIRd, k — сравнить регистр Rdc константой K, $16 < d < 31$.

Команды SBi и SBI работают только с младшими 32-мя регистрами ввода/вывода.

При использовании подпрограмм нужно обязательно определять стек! Для этого нужно занести значения адреса вершины стека в регистры SP и SPL.

Задание 1

1. Рассмотрите пример приведенной программы на ассемблере, определите основные использованные директивы. В тетради опишите их назначение, приведите примеры комментариев и операндов. Приведите примеры строк из программы, которые соответствуют формам:

[Метка:] директива [операнды] [Комментарий],

[Метка:] команда [операнды] [Комментарий].

Обозначьте все элементы (метки, директивы, команды (мнемоника), операнды, комментарии).

Программа, написанная на ассемблере, должна иметь определенную структуру. Предлагается следующий шаблон (для Atmega 8535):

```
-----
; название программы,
; краткое описание, необходимые пояснения
;-----
; подключаемые дополнительные файлы
.include "m8535def.inc" ; файл описания ATmega8535
.include "<имя_файла1.расширение>" ; включение дополнительных
.include "<имя_файла2.расширение>" ; файлов
; глобальные константы
.equ <имя1> = <константа1>
.equ <имя2> = <константа2>
; глобальные регистровые переменные
.def <имя1> = <регистр1>
.def <имя2> = <регистр2>
; сегмент данных
.dseg
.org <адрес> ; адрес первого зарезервированного байта
label1: .BYTE 1 ; резервировать 1 байт под переменную label1
label2: .BYTE m ; резервировать m байт под переменную label2
; сегмент EEPROM (ЭСПЗУ)
```

```

.eseg
.org <адрес>          ; адрес первого зарезервированного байта
.db выражение1, выражение2, ... ; записать список байтов в EEPROM
.dw выражение1, выражение2, ... ; записать список слов в EEPROM
; сегмент кодов
.cseg
.org $0000           ; адрес начала программы в программной памяти
; вектора прерываний (если они используются)
rjmp reset          ; прерывание по сбросу
.org $0002
rjmp INT0           ; обработчик прерывания INT0
.org $0004
rjmp INT1           ; обработчик прерывания INT1
.org adrINTx        ; адрес следующего обработчика прерываний
rjmp INTx           ; обработчик прерывания x
.....              ; далее располагаются обработчики остальных
; прерываний
; начало основной программы
main: <команда> xxxx
.....
; подпрограмма 1
subr1: <команда> xxxx
.....
ret
; подпрограмма 2
subr2: <команда> xxxx
.....
ret
; программы обработчиков прерываний
INT0: <команда> xxxx
.....
reti
.....
; конец программы никак не обозначается

```

Задание 2

Рассмотрите пример приведенной программы на ассемблере, определите основные использованные директивы и команды. В тетради опишите их назначение (используйте ПРИЛОЖЕНИЕ 1). Приведите примеры строк из программы, которые соответствуют формам:

[Метка:] директива [операнды]
 [Комментарий],
 [Метка:] команда [операнды] [Комментарий].
 Обозначьте все элементы (метки, директивы, команды (мнемоника), операнды, комментарии).

Десятичное число	Код
0	3f
1	06
2	5b
3	4f
4	66
5	6d
6	7d
7	07
8	7f
9	6f

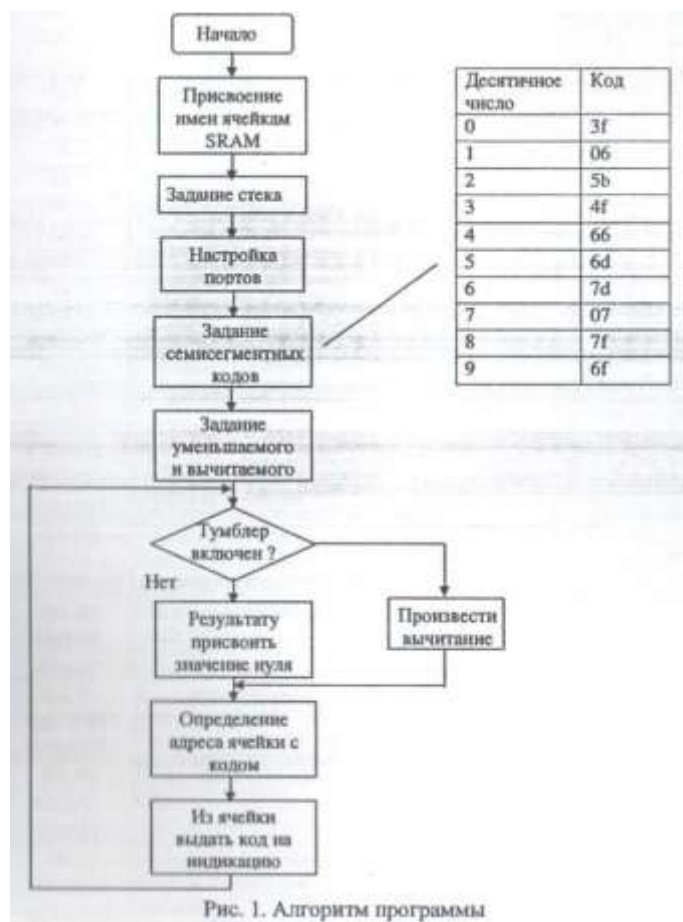


Рис. 1. Алгоритм программы

Листинг программы

```

; Программа №1
.include "m8535def.inc" ;включить файл - описание для ATmega8535
.dseg ;сегмент данных
.equ cod0=$64 ;присвоение имен ячейкам SRAM
.equ cod1=$65
.equ cod2=$66
.equ cod3=$67
.equ cod4=$68
.equ cod5=$69
.equ cod6=$6a
.equ cod7=$6b
.equ cod8=$6c
.equ cod9=$6d
.cseg
.org 0
rjmp reset
.org $30 ;начало программы
reset:
ldi r16,$00 ;определение стека с вершиной по адресу $00ff
out sph,r16
ldi r16,$ff
out spl,r16

ldi z1,$64 ;задание адреса начала зарезервированных ячеек
ldi zh,$00

ldi r16,$ff ;настроить порт C на выход
out ddrc,r16
ldi r16,00 ;настроить порт A на вход
out ddra,r16
ldi r16,$c ;настроить порт B: биты 2 и 3 на выход, остальные на вход
out ddrb,r16
ldi r16,$f0 ;настроить порт D: биты 0...4 на вход, остальные на выход
out ddrd,r16

sbi portB,3 ;выдать 1 на разряд 3 порта B
ldi r17,$3f ;задание семисегментных кодов
sts cod0,r17
ldi r17,$06
sts cod1,r17
ldi r17,$5b
sts cod2,r17
ldi r17,$4f
sts cod3,r17
ldi r17,$66
sts cod4,r17
ldi r17,$6d
sts cod5,r17
ldi r17,$7d
sts cod6,r17
ldi r17,$07
sts cod7,r17
ldi r17,$7f
sts cod8,r17
ldi r17,$6f
sts cod9,r17

ldi r17,5 ;задание уменьшаемого
ldi r18,3 ;задание вычитаемого
m1: sbis pina,4 ;если включен тумблер SA1, то пропустить
rjmp m2 ;следующую команду
mov r20,r17 ; в r20 поместить уменьшаемое
sub r20,r18 ; вычесть вычитаемое
rjmp vv
m2: ldi r20,0
vv: push z1 ;сохранить z1 в стеке
add z1,r20 ;сложить z1 с результатом
ld r0,z ;семисегментный код результата переслать в r20
pop z1 ;извлечь z1 из стека
out portc,r0 ;выдать результат на индикацию
rjmp m1

```

Контрольные вопросы

1. Ассемблер – это...
2. Директива – это...
3. Мнемоника – это...
4. В каких форматах можно задавать операнды в ассемблере?
5. Каково назначение комментариев в программе?
6. Каково назначение директивы CSEG?
7. Каково назначение директивы DB?
8. Каково назначение директивы DW?
9. Каково назначение директивы EXIT?
10. Каково назначение директивы DEVICE?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Основные понятия и определения
3. Выполненное задание
4. Ответы на контрольные вопросы
5. Вывод по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Практическое занятие №6.

Изучение работы среды программирования AVRStudio

Цель работы: изучить основные функции по созданию программ в среде программирования и отладки AVRStudio.

Выполнив работу, Вы будете:

уметь:

- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

ПК с установленной средой программирования и отладки AVRStudio

Стенд «Микроконтроллер», ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

Т.к. микроконтроллер ATmega8535 является программируемым, пользователь должен освоить его программирование в различных программных средах и в различных языках высокого и низкого уровня.

Среди наиболее популярных методов написания программ можно выделить:

-написание программ на машинном коде микроконтроллера. Программы написанные таким способом, являются наиболее быстродействующими, однако для их написания требуется высокая квалификация программиста и глубокое знание архитектуры процессора;

-написание программы на ассемблере. Написание программ на ассемблере существенно проще, чем на машинном коде, однако также требует высокой квалификации программиста. Следует отметить, что язык ассемблера обычно жестко привязан к конкретному типу микропроцессора и может существенно отличаться для разных микропроцессоров одного производителя;

-написание программы на языке высокого уровня (например, Си). Такие программы обычно являются кросс-платформенными, то есть практически не отличаются по синтаксису для микропроцессоров разных типов. Пользователь пишет программу на языке высокого уровня, а компилятор преобразует ее в ассемблер и машинный код конкретного микропроцессора. Однако использование языка высокого уровня при написании программ для микроконтроллеров может существенно ограничить их быстродействие.

Создание проекта в среде AVRStudio

Для написания программ, необходимо создать проект.

С лабораторным комплексом поставляется программа AVRStudiover. 4. Действия по созданию проекта и работе с программой в разных версиях AVRStudio могут несколько отличаться, однако большинство действий соответствуют изложенным далее пунктам.

1. Запустить программу AVRStudio. Ярлык для запуска программы находится на рабочем столе или в меню «Пуск» Windows. В появившемся окне приветствия программы будет предложено создать новый проект или открыть существующий (рис. 1).

2. При выборе нового проекта появляется окно, в котором предлагается выбрать язык программы AtmelAvrAssemblerили AvrGCC.Здесь необходимо выбрать AtmelAvrAssembler, указать имя проекта и имя инициализационного файла с расширением *.asm. Рекомендуется, чтобы имена проекта и инициализационного файла совпадали.

Очень важно: не допускать в имени файла, проекта или пути символов кириллицы. После этого следует нажать кнопку «Next» (Далее).

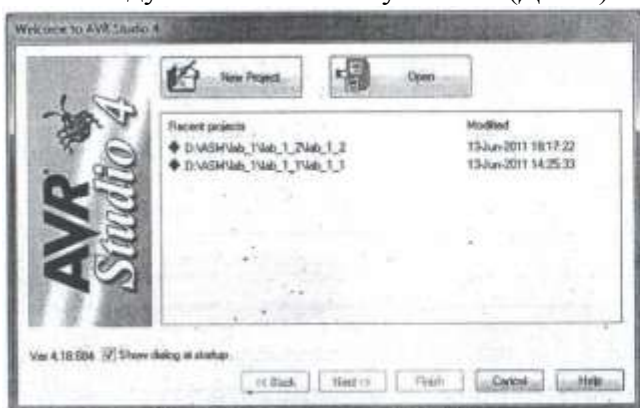


Рис. 1. Окно приветствия AVR Studio



Рис. 2. Выбор языка программы

3. После выбора языка программы и имени проекта появляется окно выбора платформы (Debugplatform) и устройства (Device). Здесь необходимо выбратьAVRSimulatorи процессор ATmega8535, который используется в лабораторном стенде. После этого следует нажать кнопку «Finish» (рис. 3).

4. При нажатии на кнопку «Finish» в программе открывается рабочее окно программирования (рис.4)

Рабочее окно содержит несколько областей:

- окно проекта «Project». Здесь отображается структура проекта, которая содержит его имя и список подключенных файлов и библиотек (рис.4 окно 1);
- центральная рабочая область, в которой осуществляется непосредственно набор программы (окно «2»);
- окно текущих сообщений «built» программы (окно «3»);
- окно регистров микроконтроллера «I/O View» (окно «4»),



Рис. 3. Выбор платформы и типа устройства

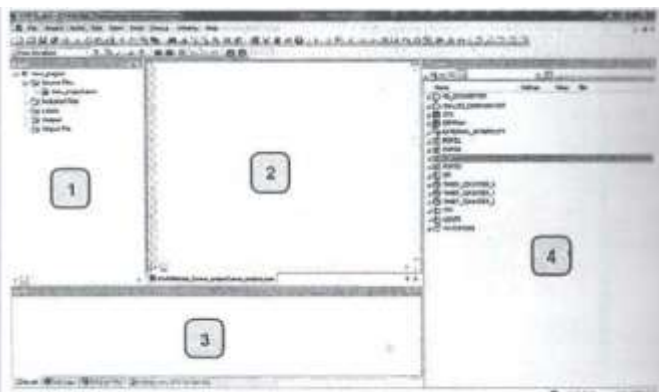


Рис. 4. Рабочее окно программы AVR Studio

В программное обеспечение AVRStudio встроено симулятор микроконтроллеров, с помощью которого можно отладить программу, найти в ней ошибки и исправить неточности в алгоритме, не осуществляя непосредственно программирования микросхемы контроллера. Также симулятор может быть полезен при написании программ в домашних условиях.

Симулятор AVRStudio осуществляет симуляцию микроконтроллера, его портов, таймеров, АЦП, прерываний и т.д. Поскольку симулятор работает с hex-файлом проекта, то для запуска эмулятора необходимо написать программу исключить из нее явные ошибки, с которыми создание hex-файла невозможно.

Для отладки программы в симуляторе необходимо после ее написания нажать сочетание клавиш Ctrl+F7 «Assemble and Run», после чего будет произведено ассемблирование программы и запуск эмулятора.

После успешной компиляции начало программы будет отмечено желтой стрелкой. Для управления процессами отладки программы в строке меню AVRStudio располагается меню «debug», а также панель функциональных кнопок управления симулятором, назначение которых приведено в табл. 1.

Таблица 1. Назначение кнопок управления эмулятором

Название	Сочетание клавиш	Назначение
Start Debugging (начать отладку)	Ctrl+Shift+Alt+F5	Начать процесс отладки программы в эмуляторе.
Stop Debugging (остановить отладку)	Ctrl+Shift+F5	Остановить процесс отладки программы и эмулятора.
Run (запуск эмуляции)	F5	Эмулятор запускается и циклически производит эмуляцию программы без отображения текущих изменений регистров контроллера на экране.
Break (приостановить эмуляцию)	Ctrl+F5	Команда временно приостанавливает эмулятор без потери данных.
Reset (остановить эмуляцию)	Shift+F5	Команда полностью останавливает эмулятор с потерей данных эмуляции.
Step into (Сделать один шаг вперед)	Fit	Команда извлекает только одну инструкцию. После ее завершения все рабочие экраны эмулятора обновляются.
Auto Step (Автоматическое выполнение)	Alt+F5	Команда выполняет эмуляцию программы в пошаговом автоматическом режиме с оперативным обновлением информации на экранах эмулятора после каждого шага.
Toggle breakpoint (Точка остановки)	F9	Команда добавляет или убирает пользовательскую точку остановки, при достижении которой программа будет приостановлена.

При симуляции программы рекомендуется выполнять пошаговое выполнение инструкций, пользуясь командой Stepinto(табл. 1). При этом необходимо контролировать содержимое регистров микроконтроллера, а также портов ввода/вывода.

Контроль состояния регистров и отдельных устройств микроконтроллера осуществляется в окне «4» (рис. 4). Каждое устройство можно развернуть и увидеть содержимое его регистров управления и контроля (рис. 5).

Запись программы в контроллер и проверка ее работоспособности

После успешной отладки программы необходимо «прошить» ее в микроконтроллер (т.е. записать ее в память микроконтроллера) и проверить работу.

Для «прошивки» программы используется та же программа AVRStudio.

Задание.

Написать программу, осуществляющую вывод числа 01010101 или 10101010 на PORTC микроконтроллера, в зависимости от состояния сигнала на входе PORTB0.

После набора программы необходимо произвести ее ассемблирование, то есть сборку, при которой ассемблер производит проверку синтаксиса написанной программы и при отсутствии ошибок преобразует код ассемблера в машинный код микропроцессора. При этом формируется файл с расширением *.hex, который далее записывается в микроконтроллер.

Для ассемблирования написанной программы необходимо нажать кнопку F7 «Assemble». При отсутствии ошибок в окне текущих сообщений «built» AVRStudio отображается сообщение об успешном завершении операции, а при наличии ошибок выводится их список и положение в тексте программы.

Листинг программы

```
-----  
; Пробная программа для микроконтроллера ATMEGA8535  
; Входы: PORTB0  
; Выходы: PORTC0...PORTC7  
  
.include "m8535def.inc" ;подключение библиотеки контроллера  
.cseg ;начало сегмента кода  
.org 0  
reset:  
 ldi r16,0xFF  
 out DDRC,r16 ;назначение PORTC на вывод  
 clr r16  
 out DDRB,r16 ;назначение PORTB на ввод  
main:  
 sbis PINB,0 ;если на PINB0=0, то  
 rjmp PINB0_is_0 ;переход на "PINB0_is_0"  
 ldi r16,0xAA ;иначе вывод на PORTC числа 0xAA (1010 1010)  
 out PORTC,r16  
 rjmp main ;далее - возврат на main  
PINB0_is_0:  
 ldi r16,0x55 ;вывод на PORTC числа 0x55 (0101 0101)  
 out PORTC,r16  
 rjmp main ;далее - возврат на main  
-----
```

Выполнить отладку программы и записать ее в микроконтроллер.

Последовательность действий с использованием AVRStudio следующая:

-включить переключатель «Сеть» модуля «Микроконтроллер» для подачи на него напряжения питания;

-в меню «Tools» AVRStudio выбрать пункт «ProgramAVR», в котором указать способ соединения «AutoConnect». При правильном подключении персональному компьютеру модуль микроконтроллера инициализируется на USBCOM-порт. Необходимо, чтобы номер этого порта был не более COM9 в противном случае необходимо найти этот порт в диспетчере оборудования Windows и переназначить номер порта;

-если номер USBCOM-порта соответствует указанным требованиям, то происходит подключение микроконтроллера к среде AVR-Studio и появляется и окно программирования контроллера (рис. 6);

-в окне программирования необходимо выбрать вкладку main, в которой необходимо выбрать тип используемого контроллера, после чего перейти во вкладку «Program», в которой

выбрать графу «Flash». В этой графе требуется указать путь к *.hex-файлу проекта, после чего произвести запись программы в микроконтроллер нажатием кнопки «Program».

По завершении записи программы необходимо проверить ее корректную работу на микроконтроллере и показать результат преподавателю.

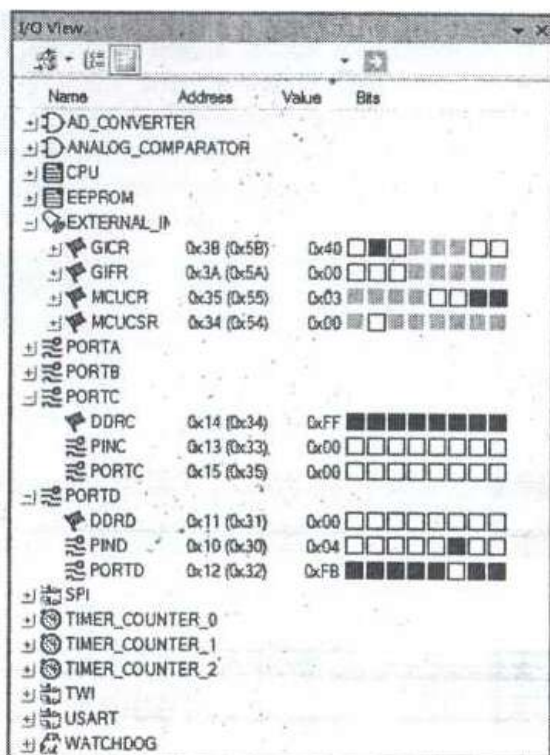


Рис. 5. Контроль состояния регистров микроконтроллера при симуляции

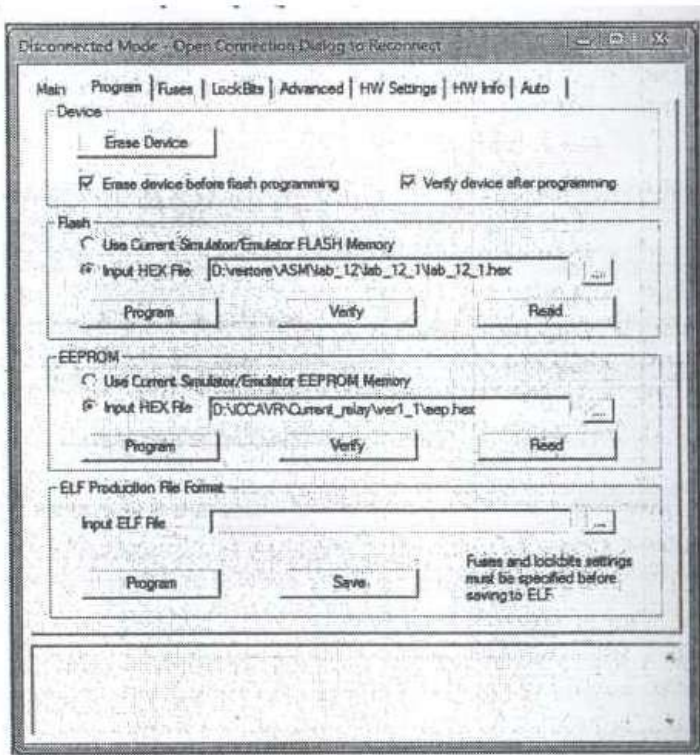


Рис. 6. Окно программирования микроконтроллера

Контрольные вопросы

1. Какие директивы использовались в программе?
2. Каково их назначение?
3. Какие команды использовались в программе?
4. К каким типам команд они относятся?
5. Приведите примеры операндов, использованных в программе.
6. Каково назначение AVR Studio?
7. Какой порт используется для программирования микроконтроллера?
8. Какие порты микроконтроллера использовались в программе и для чего?
9. Файл с каким расширением записывается в микроконтроллер?
10. В каком коде записывались числа 01010101 и 10101010 в программе?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Основные понятия и определения
3. Выполненное задание
4. Ответы на контрольные вопросы
5. Вывод по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Практическое занятие №7.

Изучение устройства параллельных портов МК Atmega8535

Цель работы: изучить принцип работы параллельных портов МК.

Выполнив работу, Вы будете:

уметь:

- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

не требуется

Теоретические сведения

Порты ввода/вывода микроконтроллера предназначены для передачи и приема информации и последующей ее обработки. Микроконтроллеры различных типов содержат различное количество портов ввода/вывода. Микроконтроллер Atmega8535 содержит 4 порта ввода/вывода, каждый из которых содержит 8 разрядов: PORTA, PORTB, PORTC, PORTD.

Порты ввода/вывода непосредственно связаны с выводами микросхемы микроконтроллера, при этом каждый конкретный вывод микроконтроллера жестко «привязан» к конкретному разряду порта ввода/вывода. На рис. 1 представлено расположение выводов микроконтроллера Atmega8535, выполненного в DIP-корпусе.

По умолчанию выводы микросхемы контроллера предназначены для выполнения функций ввода/вывода информации в соответствии с настройками регистров портов ввода/вывода. Однако функции большинства выводов микросхемы могут быть программно изменены. При этом к выводам микросхемы могут быть присоединены выходы таймеров, приемопередатчиков, входы аналогово-цифрового преобразователя, контроллера внешних прерываний.

Альтернативные функции выводов микроконтроллера представлены на рис. 1 (вскобках).

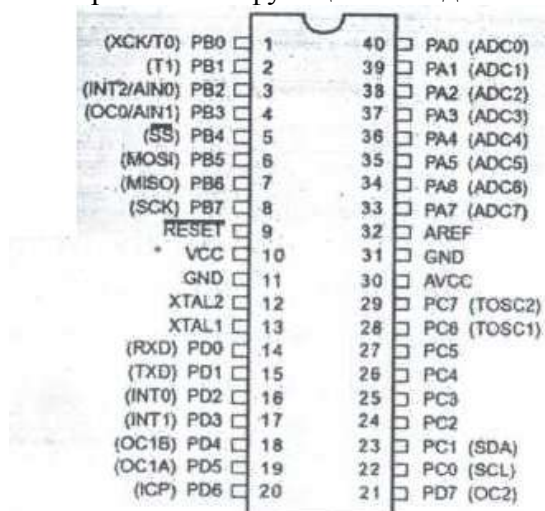


Рис. 1. Расположение выводов микросхемы контроллера ATmega8535 (DIP40)

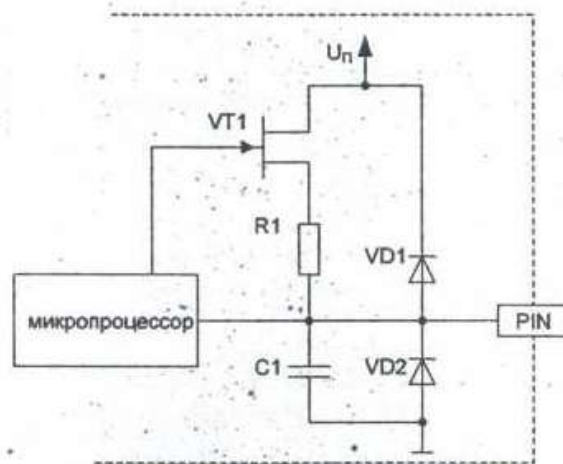


Рис. 2. Структура реализации вывода портов микроконтроллера

Каждый порт состоит из трех регистров, с помощью которых осуществляется установка направления работы порта и выдача/сбор информации (табл. 1).

Таблица 1. Регистры портов ввода/вывода

Наименование и порта	Регистры		
	Регистр направления	Регистр данных	Регистр состояния
Порт А	DDRA	PORTA	PINA

Порт В	DDRB	PORTB	PINB
Порт С	DDRC	PORTC	PINC
Порт D	DDRC	PORTD	PIND

Регистры направления определяют режим работы портов ввода/вывода. Если в каком-либо разряде регистра установлена логическая «1», и соответствующий вывод микросхемы контроллера (рис. 1) работает на вывод информации из микроконтроллера. В противном случае соответствующий вывод микросхемы (рис. 1) работает на ввод информации в микроконтроллер.

Регистры данных предназначены для передачи данных на вывод микросхемы контроллера. Если в каком-либо разряде регистра установлена логическая «1», а соответствующий вывод микросхемы сконфигурирован на выход с помощью регистра направления, то на вывод микросхемы контроллера подается сигнал, соответствующий логической «1». В противном случае на вывод микросхемы контроллера подается сигнал логического «0». Регистры используются для вывода информации из микроконтроллера.

Регистры состояния предназначены для отображения текущего состояния сигналов на выводах микросхемы контроллера. Так, если на выводе микросхемы находится сигнал логической «1», то соответствующий разряд регистра направления находится в состоянии логической «1». Регистры используются для ввода информации в микроконтроллер.

Инициализация порта на ввод и на вывод информации

Для того, чтобы освоить принципы установки порта на ввод или вывод информации, необходимо иметь представление о реализации его структуры. Каждый вывод порта выполнен по схеме, представленной на рис. 2.

При инициализации порта на ввод информации микроконтроллер принимает сигналы, поступающие от внешнего объекта. Эти сигналы подаются на выводы микросхемы (рис. 2, вывод PIN).

Диоды VD1 и VD2 выполняют функцию защиты микропроцессора от сигналов, величина которых находится за пределами диапазона 0...5В. Так, если напряжение на входе микроконтроллера превышает +5В, то открывается диод VD1, а если напряжение оказывается меньше 0В, то открывается диод VD2 (рис. 2). Конденсатор C1 выполняет защиту микроконтроллера от импульсных помех

Поскольку микропроцессор имеет высокое входное сопротивление, его вход является восприимчивым к воздействию помех. По этой причине важно, чтобы сигнал, подаваемый на микропроцессор, имел однозначное значение логического «0» или логической «1». Для этого в микроконтроллере присутствуют так называемые подтягивающие резисторы (PullUp).

Подтягивающий резистор имеет высокое сопротивление, измеряемое десятками кОм, и не оказывает влияния на подключаемые к контроллеру сигналы. Через резистор R1 к порту ввода/вывода подключается напряжение питания с помощью транзистора VT1. Если при этом к выводу микросхемы контроллера не подключена внешняя цепь и вывод «висит в воздухе», то на микропроцессор через подтягивающий резистор R1 подается сигнал логической «1». Это надежно защищает вывод микроконтроллера от воздействия внешних помех.

Для **инициализации порта на ввод информации** и подключения подтягивающих резисторов необходимо:

- задать в регистре направления DDR работу порта на ввод информации установкой нулевых значений в его разрядах;
- включить подтягивающие резисторы порта ввода/вывода установкой сигналов логической «1» в разрядах регистра PORTX.

Считывание данных с порта в данном режиме осуществляется путем опроса регистра состояния PIN.

Пример инициализации порта А на «ввод данных»:

```
ldi r16,0x00 ; в POH r16 записывается число 0000 0000
out DDRA,r16 ; командой out значение r16 посылается в DDRA
ldi r16,0xFF ; в POH записывается число 1111 1111
out PORTA,r16 ; командой out значение r16 посылается в PORTA
in r16, PINA ; командой in вводится значение PINA в POH r16
```

При инициализации порта на вывод информации код, выдаваемый микропроцессором, поступает непосредственно на вывод микросхемы контроллера. Подтягивающие резисторы при этом должны быть отключены.

Для **инициализации порта на вывод информации** необходимо:

- задать в регистре направления DDR работу порта на вывод информации установкой его разрядов в логическую «1»;
- вывести необходимую информацию на порт записью в регистр данных PORT необходимых значений.

Пример инициализации порта А на «вывод данных»:

```
ldi r16,0xFF ; в PORT r16 записывается число 1111 1111
out DDRA,r16 ; командой out значение r16 посылается в DDRA
ldi r16,0x35 ; в r16 записывается любое число, например, 0x35
out PORTA,r16 ; командой out значение r16 посылается в PORTA
```

Задание

Составьте программы:

1. Инициализация порта В (С, D) на ввод данных
2. Инициализация порта В (С, D) на вывод данных

Контрольные вопросы

1. Сколько и каких портов имеет МК Atmega8535?
2. Каково назначение регистра направления?
3. Каково назначение регистра данных?
4. Каково назначение регистра состояния?
5. Каково назначение подтягивающих резисторов?

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Практическое занятие №8.

Изучение работы регистра состояний SREG МК Atmega 8535

Цель работы: изучить работу регистра состояний SREG МК.

Выполнив работу, Вы будете:

уметь:

- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

В микроконтроллерах фирмы Atmel присутствует так называемый регистр состояния SREG (StatusREGister).

Регистр состояния содержит информацию о результатах наиболее часто извлекаемых арифметических операциях. Эта информация может быть использована при написании программы для создания переходов, циклов сравнения чисел и т.д.

Разряды регистра SREG называются флагами. Всего этих флагов восемь:

№бита	7	6	5	4	3	2	1	0
Флаг	I	T	H	S	V	N	Z	C

Назначение разрядов регистра SREG:

Бит 0: C - флаг переноса. Флаг переноса индицирует появление переноса при выполнении арифметических или логических операций. Флаг переноса незаменим при совершении операций с n-байтными числами.

Пример 1. Прибавить к числу 0xFE произвольное 8-разрядное число. Результат - шестнадцатиразрядное число.

```

clr r18 ;очистка регистра r18
ldi r16,0xFE ;запись в r16 значения 0xFE
ldi r17,0x05 ;запись любого числа в r17
mov r20,r16 ;копирование r16 в r20
add r20,r17 ;сложение r20 и r17
adc r21,r18 ;если при этом возникает перенос, то в r21
;прибавляется 0x01. Таким образом, результат
;формируется в r21:r20

```

Бит 1: Z- флаг нуля. Этот флаг индицирует нулевой результат при выполнении арифметических или логических операций. Этот флаг может быть полезен при выполнении такой операции, как сравнение.

Пример 2. Осуществить сравнение двух чисел, задаваемых на портах A и C. При их равенстве выдать сигнал логической «1» на PORTD0

```

main:
    in r16,PINA ;ввод первого числа
    in r17,PINC ;ввод второго числа
    cp r16,r17 ;сравниваются значения регистров r16 и r17
    ;(проведением операции вычитания r16-r17). При их
    ;равенстве устанавливается в '1' флаг Z
    breq m1 ;при Z=1 осуществляется переход на метку m1
    cbi PORTD,0 ;иначе бит PORTD0 очищается
    rjmp main ;и идет возврат на main
m1:
    sbi PORTD,0 ;при переходе на m1 устанавливается бит PORTD0
    rjmp main ;и идет возврат на main

```

Бит 2: N- флаг отрицательного значения. Этот флаг индицирует наличие отрицательного числа как результата выполнения арифметических или логических операций. В микроконтроллерах отрицательное число получается из положительного путем перевода в дополнительный код. Для этого все разряды числа инвертируются, а потом к числу прибавляется единица младшего разряда. Так, если 8-разрядное число +1 записывается в двоичном коде как 0000 0001, то отрицательное число -1 записывается как 1111 1111. При этом старший разряд (бит 7) числа определяет его знак.

Флаг отрицательного значения может быть полезен при совершении операции сравнения двух чисел.

Пример 3. Осуществить сравнение двух чисел A и B, задаваемых на портах A и C. Если число $A \geq B$, то на PORTD0 подается логическая «1». Иначе на PORTD0 подается логический «0».

```

main:
    in r16,PINA ;ввод числа A
    in r17,PINC ;ввод числа B
    cp r16,r17 ;сравнение чисел A и B вычитанием A-B
    bgtmi m1 ;если при этом установлен флаг N, то
    ;осуществляется переход на m1
    sbi PORTD,0 ;иначе устанавливается разряд PORTD0
    rjmp main ;и идет возврат на main
m1:
    cbi PORTD,0 ;по метке m1 очищается разряд PORTD0
    rjmp main ;и идет возврат на main

```

Бит 3: V - флаг переполнения. Этот бит устанавливается при переполнении регистра во время совершения операций над числами. Так, если в регистре было записано число 1111 1111 и

к нему прибавили +2, то происходит переполнение. Результатом такой операции будет число 0000 0001 и установленный флаг V.

Необходимо различать флаг переполнения V и флаг переноса C. Флаг переполнения предназначен, в первую очередь, для работы с дополнительным кодом. Как было сказано выше, в дополнительном коде старший разряд определяет знак числа, а значение числа ограничивается 7 разрядами.

Если при сложении двух положительных чисел в дополнительном коде происходит изменение старшего бита, то это означает, что произошло переполнение числа, хотя переноса не происходит (флаг C не меняется) Например, при сложении чисел:

```

+   0110 0010
+   0110 1111
=   1101 0001

```

флаги устанавливаются следующим образом:

-поскольку произошло изменение старшего разряда, то флаг переполнения V устанавливается: V=1;

- так как не произошло переполнение разрядов регистра, то флаг переноса не возникает: C=0.

Если при сложении двух отрицательных чисел в дополнительном коде происходит изменение старшего бита, то это означает, что также произошло переполнение числа, при этом может возникнуть и флаг переноса. Например, при сложении чисел:

```

+   1001 1110
+   1001 0001
=   1 0010 1111

```

меняются как старший разряд, так и появляется бит переполнения:

-поскольку произошло изменение старшего разряда, устанавливается флаг переполнения V;

-поскольку произошло переполнение разрядов регистра, устанавливаем и флаг переноса C.

Пример 4. Используя инструкцию «add», сложить два положительных числа в дополнительном коде. Если происходит переполнение результата, то результат необходимо ограничить максимальным или минимальным числом. Результат выводится на PORTD.

```

main:
in r16,PINA      ;ввод первого числа A
in r17,PINC      ;ввод второго числа B
add r16,r17      ;сложение A и B
bivc m1          ;если нет переполнения (флаг V=0), то
                 ;осуществляется переход на m1

brcs m2          ;иначе проверка флага переноса. Если флаг C=1,
                 ;то осуществляется переход на m2
ldi r16,0x7F     ;если переноса нет, то результат -
                 ;максимальное число +127 (0111 1111)
rjmp m1          ;далее - переход на m1

m2:
ldi r16,0x80     ;если флаг переноса C=1, то результат -
                 ;минимальное число -128 (1000 0000).

m1:
out PORTD,r16    ;по метке m1 - вывод результата на PORTD
rjmp main        ;и возврат на main

```

Бит 4:S- флаг знака. Этот бит всегда является результатом совершения операции исключающего ИЛИ между флагом отрицательного числа N и флагом переполнения V. Так, если при совершении операции не происходит переполнения (флаг V=0), то знак определяется флагом N. Если же происходит переполнение (V=1), то флаг знака принимает инвертированное значение флага N.

Бит 5:N - флаг половинного переноса. Половинным переносом называется процесс переноса из первой половины байта во вторую. Так, если в байте была комбинация 0000 1111 и к нему прибавили +1, то происходит половинный перенос, а именно 0001 0000, при этом формируется флаг N.

Пример 5. Организовать бегущий огонь в младшем полубайте PORTC.

```

ldi r16,0xFF ;в r16 записывается значение 0xFF
out DDRC,r16 ;порт С инициализируется на вывод информации
ldi r16,0x01 ;в r16 записывается единица младшего разряда
main:
out PORTC,r16 ;значение r16 выводится на PORTC
lsl r16 ;r16 сдвигается влево на один разряд
brhc main ;если флаг Н снят, то переход на main
ldi r16,0x01 ;иначе в r16 записывается 0x01
rjmp main ;и осуществляется переход на main

```

Бит 6: T - хранение бита информации. Инструкции копирования бит используют бит T регистра SREG. При копировании бита из какого-либо регистра он сохраняется в бите T регистра SREG, а затем извлекается из него при копировании в какой-либо другой регистр.

Пример 6: Скопировать из регистра r16 в регистр r19 пятый бит, не изменяя содержимое регистра r16.

```

Без использования флага T:
in r16,PINA ;запись в r16 любого числа
mov r17,r16 ;копирование r16 в любой свободный PCH (r17)
andi r17,0x20 ;выделение 5-го бита в r17
or r19,r17 ;логическое ИЛИ r19 и r17

С использованием флага T:
in r16,PINA ;запись в r16 любого числа
bst r16,5 ;копирование 5-го бита в SREG
bld r19,5 ;копирование содержимого флага T в 5-й бит r19

```

Бит 7: I - общее разрешение прерываний. Назначение этого флага будет пояснено в работах с использованием прерываний таймеров.

Контрольные вопросы

1. Для чего предназначен регистр состояния?
2. Перечислите биты регистра состояния и их назначение.
3. При выполнении каких операций изменяется для флага нуля? Флаг отрицательного значения? Флаг знака?
4. При суммировании двух 4-х разрядных чисел какие биты регистра состояния могут изменить свое значение? Двух восьмиразрядных?
5. Какой бит регистра состояния отвечает за разрешение работы всех прерываний?
6. Как связаны друг с другом флаги: знак, отрицательное значение, переполнение?
7. Для каких целей используется бит T?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Программные примеры работы с флагами
3. Ответы на контрольные вопросы
4. Вывод по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Практическое занятие №9.

Изучение работы стека МК Atmega 8535

Цель работы: изучить организацию и принцип работы стека МК.

Выполнив работу, Вы будете:

уметь:

- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры

- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

При выполнении программы извлечение и выполнение директив (команд) ведется последовательно. То есть сначала исполняется первая в списке команда, потом - вторая, третья и т.д.

Все микроконтроллеры и микропроцессоры имеют так называемый счетчик команд (ProgramCounter). В нем хранится адрес текущей выполняемой команды. При запуске программы счетчик команд равен 0, затем он инкрементируется по мере выполнения команд на 1 или на 2, в зависимости от длины команды (Приложение А). Если по какой-либо причине в программе осуществляется переход в другое место программы (безусловный или условный переход), то содержимое счетчика команд скачком изменяется, указывая на новый адрес вызванной команды.

Если при выполнении программы возникает необходимость осуществить переход на какую-либо подпрограмму, выполнить ее, а затем вернуться на старое место и продолжить выполнение программы, становится необходимым запоминать адрес возврата. Для этого и предназначен указатель стека.

Указатель стека - это специальный регистр, представляющий собой буфер, в котором реализован принцип «LastIn- FirstOut» LIFO(последним пришел - первым уйдешь). Этот принцип означает, что адрес перехода, который был записан в стек самым последним, будет считан самым первым. Таким образом, если в программе последовательно выполняются несколько переходов в подпрограммы, никогда не возникнет путаницы с порядком возврата и подпрограмм обратно.

В микроконтроллере Atmega8535 стек реализован в двух 8-разрядных регистрах SPH, SPL, которые вместе образуют 16-разрядный регистр SPH, SPL. Принцип работы стека поясняется на рис. 1.

При работе программы каждая ее команда имеет адрес, присваиваемый счетчиком команд (рис. 1). Если при выполнении программы возникла необходимость сделать переход на какую-либо подпрограмму, расположенную и какому-либо адресу, например, как показано на рис. 1, по адресу 243, необходимо иметь информацию, на какой адрес необходимо вернуться после выполнения подпрограммы. Поэтому при переходе, например, с адреса 006 на адрес 243 указатель стека записывается адрес возврата, то есть 007.

При вызове следующей подпрограммы, расположенной по адресу 120, указатель стека записывается также адрес возврата. В данном случае - 052.

Адрес возврата записывается в вершину стека.

Вершиной стека называется адрес ОЗУ процессора, в которую ведется запись адреса возврата.

Адрес вершины стека программист обязан указать самостоятельно, при этом общепринято, что вершина стека находится в конце ОЗУ процессора. Это делается для того, чтобы область программы случайно не перекрылась с областью стека, так как если это произойдет, то адрес возврата, записанный в область стека, будет потерян.

Возможен случай, когда при выполнении подпрограммы происходит запрос на следующий переход на другую подпрограмму (рис. 2).

Когда происходит переход из основной программы в подпрограмму 1, в вершину стека записывается адрес возврата 007. Если далее из подпрограммы 1 происходит переход в подпрограмму 2, то в стек записывается адрес 245 возврата в подпрограмму 1.



Рис. 1. Принцип работы указателя стека

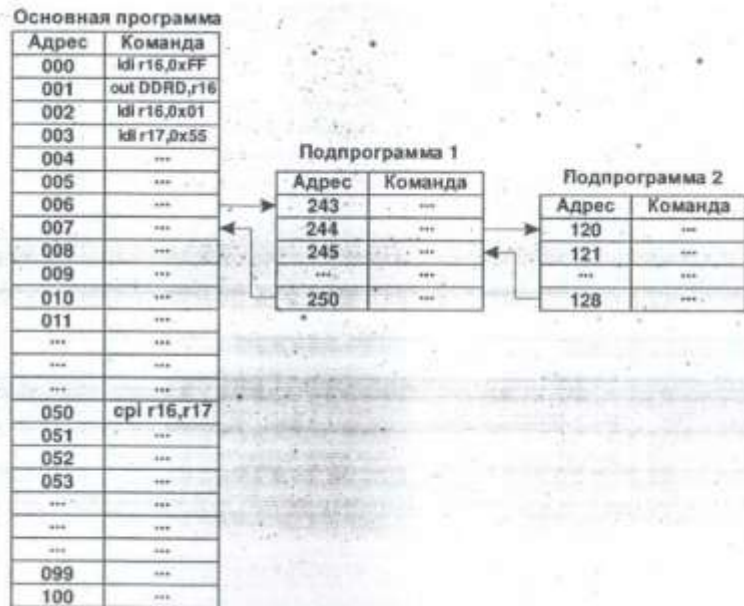


Рис. 2. Принцип заполнения стека

При окончании подпрограммы 2 первым будет прочитан адрес 245, а затем при окончании подпрограммы 1 - адрес 007. Таким образом, благодаря принципу LIFO полностью исключается неправильный переход в неправильное место программы.

В микроконтроллере Atmega8535 указатель стека состоит из двух 8-разрядных регистров SPH и SPL, которые вместе составляют 16-разрядный регистр SPH:SPL. Применение 16-разрядного регистра объясняется объемом памяти программ микроконтроллера - он составляет 8кБ (4096 слов памяти программ) и для обращения к конкретной ячейке памяти необходимо указания 2 байт адреса.

При написании программ вызов подпрограмм обычно осуществляется командой `rcall`, а возврат из подпрограммы осуществляется по директиве `ret`.

При вызове директивы `rcall` в регистр SPH:SPL записывается адрес ячейки памяти, в которую будет сохранен адрес возврата из подпрограммы, и адрес возврата автоматически записывается в стек, а при вызове директивы `ret` происходит чтение из стека по указанному в регистре SPH:SPL адресу.

Контрольные вопросы

1. Что такое стек?
2. Что такое вершина стека?
3. Что такое указатель стека?
4. Что означает термин LIFO?
5. Что представляет собой стек в МК Atmega8535?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Ответы на контрольные вопросы
3. Вывод по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Практическое занятие №10.

Изучение работы таймеров в различных режимах МК Atmega 8535

Цель работы: изучить работу таймеров МК в режиме ШИМ и в режиме создания временных интервалов.

Выполнив работу, Вы будете:

уметь:

- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение

ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

Широтно-импульсная модуляция (ШИМ)

При использовании микроконтроллеров очень часто требуется решать задачи создания точных временных задержек сигналов, оценивать длительность импульсов какого-либо внешнего сигнала, знать его частоту и скважность.

Все эти задачи решаются с помощью использования таймеров/счетчиков.

По сути, таймеры представляют собой счетчики, которые ведут подсчет импульсов либо от внешнего сигнала, либо от внутреннего генератора. Кроме этого таймеры/счетчики имеют возможность работать в режиме широтно - импульсной модуляции (ШИМ).

Широтно-импульсной модуляцией называется формирование среднего значения сигнала с помощью сигналов логического «0» и логической «1» за период модуляции T (рис. 1).

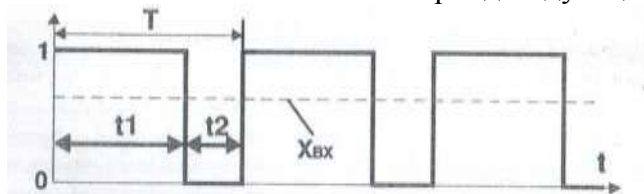


Рис. 1. Принцип широтно-импульсной модуляции сигнала

ШИМ дает возможность с помощью логических сигналов получать на выходе микроконтроллера аналоговый сигнал, среднее значение которого за период можно плавно изменять от 0 до 1, точнее от 0 до напряжения питания. Значение этого сигнала можно рассчитать по следующей формуле:

$$X_{B\Gamma} = 1 \cdot t_1 / T,$$

где t_1 - время включенного состояния (логической единицы);

t_2 - время выключенного состояния (логического нуля);

T - период выходного сигнала.

Микроконтроллер ATmega8535 содержит три таймера общего назначения два восьмиразрядных таймера T0 и T2, один шестнадцатиразрядный таймер T1. Восьмиразрядные таймеры T0 и T2 определяются и работают практически идентично друг другу. Работа таймера T1 имеет существенные отличия. В данной работе изучается работа только таймеров T0 и T2.

Для управления работой таймера T0 используются 3 регистра ввода/вывода:

- счетный регистр TCNT0;
- регистр сравнения OCR0;
- регистр управления TCCR0.

Соответствующие регистры таймера T2 называются TCNT2, OCR2, TCCR2.

Для подключения входов/выходов таймеров к выводам микросхемы контроллера (т.е. соединения таймеров с внешним миром) используются альтернативные функции портов ввода/вывода. После включения альтернативной функции данный бит порта используется только для ввода информации в таймер (например, вход T0/PB0), или вывода информации с выхода таймера (например вывод OC0/PB3). Альтернативные функции портов для работы с таймерами приведены в табл. 1.

Таблица 1. Альтернативные функции портов, используемые таймерам)

Таймер	Обозначение	Описание	Вывод
T0	T0	Внешний вход таймера T0	PB0
	OC0	Внешний выход таймера T0	PB3
T1	T1	Внешний вход таймера T1	PB1
	OC1A	Внешний выход А таймера T1	PD5
	OC1B	Внешний выход В таймера T1	PD4
T2	TOSC1	Внешний вывод 1 для подключения резонатора таймера T2	PC6
	TOSC2	Внешний вывод 2 для подключения резонатора для таймера T2	PC7
	OC2	Внешний выход таймера T2	PD7

Рассм
отрим
работу
таймера
при его
работе от
источника
тактового

сигнала процессора, в качестве которого в лабораторном стенде выступает кварцевый резонатор с тактовой частотой 8 МГц.

Примечание: для использования альтернативных функции портов соответствующие биты портов предварительно необходимо сконфигурировать на ввод или вывод. Например, при использовании альтернативной функции OC0 для вывода информации из таймера T0 вывод микросхемы PB3 должен быть определен как выход: бит DDB3 равен 1

Регистры таймера T0

Счет импульсов источника частоты ведется в 8-разрядном счетном регистре таймера TCNT0 (TimerCounterT0). Перед тем, как попасть в схему счета импульсов, тактовый сигнал поступает на схему деления частоты, которая, в соответствии с параметрами, установленными при инициализации таймера, производит деление частоты тактового сигнала f_{CLK} в следующем соотношении:

- без делителя частоты тактового сигнала f_{CLK} ;
- с делителем частоты тактового сигнала $f_{CLK}/8$;
- с делителем частоты тактового сигнала $f_{CLK}/64$;
- с делителем частоты тактового сигнала $f_{CLK}/256$;
- с делителем частоты тактового сигнала $f_{CLK}/1024$.

Каждый импульс с предделителя частоты считается и инкрементирует значение, содержащееся в счетном регистре TCNT0.

Каждый таймер содержит регистр сравнения. Таймер T0 содержит 8-разрядный регистр OCR0 (OutputCompareRegister). В это регистр записывается уставка, то есть число от 0 до 255, при достижении которого счетным регистром TCNT0 формируется флаг прерывания по совпадению таймера OCF0 (OutputCompareFlag).

В микроконтроллере Atmega8535 таймеры T0 и T2 работают в двух режимах широтно-импульсной модуляции: быстрый ШИМ и фазово-корректный ШИМ.

В режиме *быстрого ШИМ* счетный регистр TCNT0(2) таймера производит формирование пилообразной развертки (рис. 2, а), инкрементируя свое значение по каждому импульсу с предделителя, который устанавливается в регистре управления TCCR0(2) таймера T0(T2). При достижении счетным регистром значения 255 происходит его автоматическое обнуление.

В регистрах сравнения OCR0(2) таймеров T0 и T2 может быть записано любое число от 0 до 255. С этим числом сравнивается значение счетного регистра TCNT0(2) и при их равенстве происходит переключение вывода таймера OC0 или OC2 в соответствии с настройками регистра управления TCCR0(2) таймера. Вслучае, если при совпадении $TCNT0(2)=OCR0(2)$ вывод OC0(2) таймера обнуляется, то ШИМ получается неинвертирующим (рис. 2, б), а если при совпадении вывод устанавливается в состояние логической «1», то ШИМ инвертирующий (рис. 2, в).



Рис. 1. Работа предделителя таймера T0 при коэффициенте делителя 8

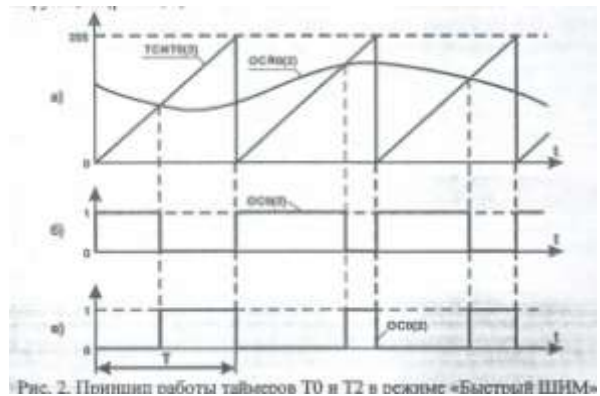


Рис. 2. Принцип работы таймеров T0 и T2 в режиме «Быстрый ШИМ»

В режиме фазового ШИМ счетный регистр TCNT0(2) формирует пилообразный сигнал развертки с нарастающим и спадающим фронтами. Сначала регистр инкрементирует свое значение от 0 до 255, а затем происходит его декремент от 255 до 0 (рис. 3, а).

Если при превышении значения, содержащимся в счетном регистре TCNT0(2), значения, содержащегося в регистре сравнения OCR0(2) происходит обнуление вывода OC0(2) таймера, то ШИМ является неинвертирующим (рис.3, б). В противном случае ШИМ - инвертирующий (рис. 3, в).

Несмотря на то, что в режиме фазового ШИМ частота ШИМ-сигнала меньше, чем в режиме быстрого ШИМ, первый режим обладает большей разрешающей способностью и используется для достижения большей точности модуляции.

Регистры таймера T0 в режиме ШИМ

Режим широтно-импульсной модуляции, как и все остальные режимы работы таймера T0, инициализируется в регистре управления TCCR0 (табл. 2).

Таблица 2. Регистр управления TCCR0 таймера

Бит	7	6	5	4	3	2	1	0
Название	FOCO	WGM00	COM01	COM00	WGM01	CS02	CS01	CS0

За установку режима работы таймера отвечают биты WGM01:WGM00. При установке WGM01=0, WGM00=1 активируется режим фазового ШИМ, при установке WGM01=1, WGM00=1 активируется режим быстрого ШИМ.

При активации режима широтно-импульсной модуляции биты COM01 и COM00 определяют поведение вывода таймера T0, в качестве которого выступает вывод PB3 микроконтроллера. Поведение вывода в соответствии с этими битами представлено в табл. 3 и табл. 4.

Таблица 3. Внешний выход ОСО таймера T0 в режиме быстрого ШИМ

COM01	COM0	Описание
0	0	Нормальная функция порта, вывод ОСО отключен
0	1	Комбинация бит зарезервирована
1	0	Очистка ОСО при совпадении. Неинвертирующий ШИМ
1	1	Установка ОСО при совпадении. Инвертирующий ШИМ

Таблица 4. Внешний выход ОСО таймера T0 в режиме фазового ШИМ

COM01	COM0	Описание
0	0	Нормальная функция порта, вывод ОСО отключен
0	1	Комбинация бит зарезервирована
1	0	Очистка ОСО при совпадении при счете вверх. Неинвертирующий ШИМ
1	1	Установка ОСО при совпадении при счете вверх. Инвертирующий ШИМ

Частота ШИМ-сигнала устанавливается битами CS02...CS00. Эти биты определяют источник задания частоты и коэффициент делителя (табл. 5).

Таблица 5. Установка источника задания частоты таймера T0

CS02	CS01	CS00	Описание
0	0	0	Таймер остановлен
0	0	1	$f_{T0} = f_{CLK} / 1$

<u>0</u>	<u>1</u>	<u>0</u>	$f_{T0}=f_{CLK} / 8$
<u>0</u>	<u>1</u>	<u>1</u>	$f_{T0}=f_{CLK} / 64$
<u>1</u>	<u>0</u>	<u>0</u>	$f_{T0}=f_{CLK} / 256$
<u>1</u>	<u>0</u>	<u>1</u>	$f_{T0}=f_{CLK} / 1024$
<u>1</u>	<u>1</u>	<u>0</u>	Внешний сигнал на входе Т0. Спадающий фронт
<u>1</u>	<u>1</u>	<u>1</u>	Внешний сигнал на входе Т0. Нарастающий фронт

В соответствии с табл. 5, частота ШИМ рассчитывается следующим образом:

- для режима быстрого ШИМ: $f = f_{CLK} / 256$;
- для режима фазового ШИМ: $f = f_{CLK} / 512$.

Регистры таймера Т2 в режиме ШИМ

Режим широтно-импульсной модуляции, как и все остальные режимы работы таймера Т2, инициализируется в регистре управления ТССР2 (табл. 6).

Таблица 6. Регистр управления ТССР2 таймера

Бит	<u>7</u>	<u>6</u>	<u>5</u>	<u>4</u>	<u>3</u>	<u>2</u>	<u>1</u>	<u>0</u>
Название	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS21

Описание и назначение управляющих бит регистра ТССР2 таймера аналогичны описанным битам регистра ТССР0 таймера Т0, поэтому отдельно комментируются. Частота ШИМ-сигнала устанавливается битами CS22...CS; Эти биты определяют источник задания частоты и коэффициент делите (табл. 7).

Таблица 7. Установка источника задания частоты таймера

<u>CS22</u>	<u>CS21</u>	<u>CS20</u>	<u>Описание</u>
<u>0</u>	<u>0</u>	<u>0</u>	Таймер остановлен
<u>0</u>	<u>0</u>	<u>1</u>	$f_{T0}=f_{CLK} / 1$
<u>0</u>	<u>1</u>	<u>0</u>	$f_{T0}=f_{CLK} / 8$
<u>0</u>	<u>1</u>	<u>1</u>	$f_{T0}=f_{CLK} / 32$
<u>1</u>	<u>0</u>	<u>0</u>	$f_{T0}=f_{CLK} / 64$
<u>1</u>	<u>0</u>	<u>1</u>	$f_{T0}=f_{CLK} / 128$
<u>1</u>	<u>1</u>	<u>0</u>	$f_{T0}=f_{CLK} / 256$
<u>1</u>	<u>1</u>	<u>1</u>	$f_{T0}=f_{CLK} / 1024$

Необходимо отметить, что в режиме ШИМ необязательно устанавливать маски и разрешать прерывания по совпадению и переполнению таймера Т0 и Т2. Эти прерывания следует активировать только при необходимости использования в программе.

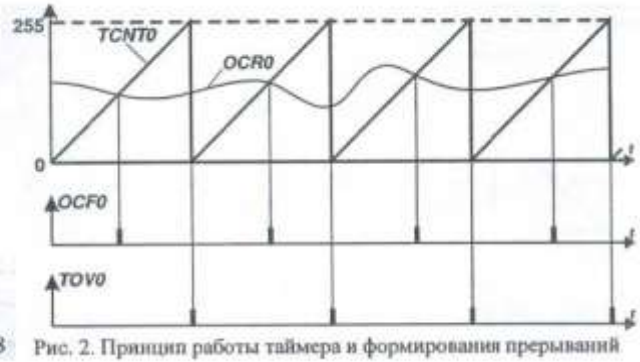
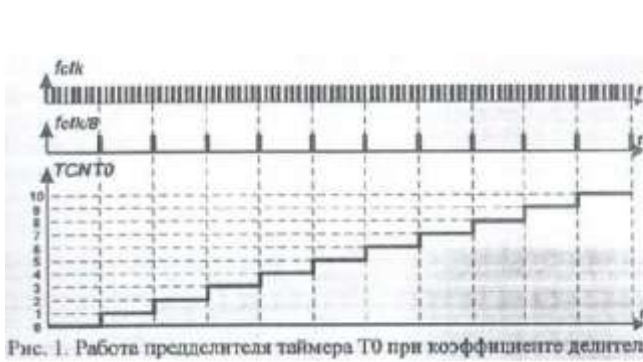
Для запуска режима ШИМ на таймере Т0 или Т2 необходимо:

- остановить таймер обнулением регистра управления ТССР0(2);
- обнулить регистр сравнения OCR0(2) и счетный регистр TCNT0(2);
- установить в регистре управления ТССР0(2) режим ШИМ и выбрать частоту его работы.
- В процессе работы ШИМ в любой момент можно изменять содержимое регистра сравнения OCR0(2) таймеров, при этом среднее значение ШИМ-сигнала на выводе ОС0 и ОС2 микроконтроллера будет пропорционально изменяться.

Работа счетчика Т0 в режиме «Очистка при совпадении»

Каждый таймер содержит регистр сравнения. Таймер Т0 содержит 8- разрядный регистр OCR0 (OutputCompareRegister). В это регистр записывается уставка, то есть число от 0 до 255, при достижении которого счетным регистром TCNT0 формируется флаг прерывания по совпадению таймера OCFO (OutputCompareFlag).

Когда значение, записанное в регистре TCNT0, переполняет максимальное значение, которое можно записать в 8-разрядный регистр сравнения OCR0, формируется флаг прерывания по переполнению таймера TOVO (TimerOverflowFlag) (рис. 2).



Настройки таймера определяет регистр TCCR0 (Timer/CounterControlregister).

Табл. 1. Управляющий регистр TCCR0 таймера T0

Бит		7	6	5	4	3	2	1	0
Значение		FOC0	WGM00	COM01	COM00	WGM01	CS2	CS1	CS0

Бит 7 – FOC0- бит принудительной установки в единичное значение выходного сигнала таймера в режиме ШИМ (в данной работе не рассматривается).

Бит 6 и бит 3 – WGM00:WGM01 - биты управления режимом работы таймера. Эти биты определяют режим работы таймера: нормальный, ШИМ- режимы или сброс при совпадении (обратите внимание 6 бит регистра устанавливает младший разряд режима работы, 3 бит - старший разряд). Виды режимов работы представлены в табл. 2.

Табл. 2. Режимы работы таймера/Счетчика

WGM01	WGM00	Режим
0	0	Нормальный
0	1	Фазовый ШИМ
1	0	Очистка при совпадении
1	1	Быстрый ШИМ

В нормальном режиме работы счетный регистр TCNT0 изменяется от 0 до 255 и далее продолжает счет. В режиме «очистка при совпадении» значение уставки записывается в регистр OCR0. Когда значение счетного регистра TCNT0 доходит до уставки OCR0, регистр TCNT0 автоматически очищается, а таймер - перезапускается. Этот режим очень удобен при необходимости периодического выполнения каких-либо операций (рис. 3).

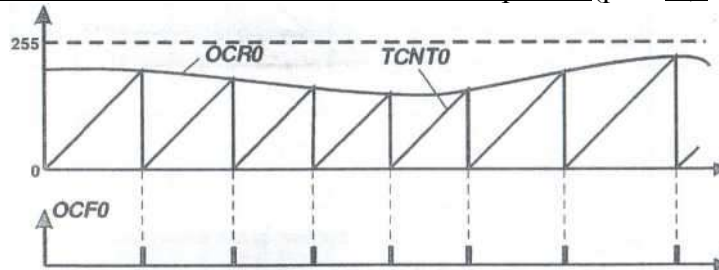


Рис. 3. Режим работы CTC (clear to compare-очистка при совпадении) таймера T0

Биты 5 и 4 – COM01:COM00 - биты управления выводом OC0. Таймер T0 может выводить сигнал на внешние контакты микросхемы, этот вывод обозначается OC0 и определяется как альтернативная функция вывода PB3 порта ввода/вывода В. Управление выводом OC0 выполняется комбинацией бит COM01:COM00 (табл. 3) и зависит также от режима работы таймера (биты WGM01:WGM00).

Табл. 3. Функции вывода OC0 таймера T0 в нормальном режиме работы

COM01	COM00	Описание
0	0	Нормальная функция порта, вывод OC0 отключен
0	1	Изменение OC0 на противоположное при совпадении OCF0
1	0	Очистка OC0 при совпадении OCF0
1	1	Установка OC0 при совпадении OCF0

Биты 3...0 - CS02...CS00 –биты предедлителя таймера и источника задания частоты таймера (таблица 4.)

Табл. 4. Функции бит CS02, CS01, CS00 таймера T0

CS02	CS01	CS00	Описание
<u>0</u>	<u>0</u>	<u>0</u>	<u>Таймер остановлен</u>
<u>0</u>	<u>0</u>	<u>1</u>	$f_{T0}=f_{CLK}/1$
<u>0</u>	<u>1</u>	<u>0</u>	$f_{T0}=f_{CLK}/8$
<u>0</u>	<u>1</u>	<u>1</u>	$f_{T0}=f_{CLK}/64$
<u>1</u>	<u>0</u>	<u>0</u>	$f_{T0}=f_{CLK}/256$
<u>1</u>	<u>0</u>	<u>1</u>	$f_{T0}=f_{CLK}/1024$
<u>1</u>	<u>1</u>	<u>0</u>	Внешний сигнал на входе T0. Спадающий фронт сигнала.
<u>1</u>	<u>1</u>	<u>1</u>	Внешний сигнал на входе T0. Нарастающий фронт сигнала.

Помимо управляющего регистра **TCCR0**, счетного регистра **TCNT0** и регистра сравнения **OCR0**, в микроконтроллере содержатся регистр масок прерываний таймеров **TIMSK** и регистр флагов прерываний таймеров и **TIFR**. Регистры позволяют отслеживать то или иное событие (совпадение, переполнение, захват), происходящее во всех таймерах микроконтроллера.

Регистры **TIMSK** и **TIFR** общие для всех таймеров, поэтому они содержат установки для всех таймеров (T0, T1 и T2) микроконтроллера.

Регистр маски прерываний таймеров **TIMSK** (табл. 5) разрешает то или иное прерывание того или иного таймера. Так, например:

- 0 разряд **TOIE0** регистра установлен в единицу, это означает, что разрешается прерывание «Переполнение таймера T0»;
- 1 разряд **OCIE0** регистра установлен в единицу, это означает, что разрешается прерывание «Совпадение таймера T0» и т.д.

Табл. 5. Регистр масок прерываний таймеров **TIMSK**

Бит	7	6	5	4	3	2	1	0
Название	OCIE2	TOIE2	ICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
Наименование бита разрешения прерывания	Совпадение таймера T2	Переполнение таймера T2	Захват таймера T1	Совпадение A таймера T1	Совпадение B таймера T1	Переполнение таймера T1	Совпадение таймера T0	Переполнение таймера T0

Регистр флагов прерываний таймеров **TIFR** (табл. 6) показывает какое прерывание произошло.

Например, для таймера T0 в регистре **TIFR** используются два бита:

- бит 1 –**OCF0**- флаг прерывания по совпадению таймера;
- бит 0 –**TOV0**- флаг прерывания по переполнению таймера.

Табл. 6. Регистр флагов прерываний таймеров **TIFR**

Бит	7	6	5	4	3	2	1	0
Название	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0
Наименование флага прерывания	Флаг совпадения таймера T2	Флаг переполнения таймера T2	Флаг захвата таймера T1	Флаг совпадения A таймера T1	Флаг совпадения B таймера T1	Флаг переполнения таймера T1	Флаг совпадения таймера T0	Флаг переполнения таймера T0

Значение этих регистров следующее. Когда возникает переполнение или совпадение таймера T0, автоматически формируется флаг прерывания **TOV0** или **OCF0** в регистре **TIFR**. Если в регистре масок прерываний таймеров **TIMSK** на соответствующее прерывание наложена маска, то вызывается процедура обработки прерывания. Если же маска прерывания не наложена, то при совпадении или переполнении таймера ничего не происходит.

8-разрядный таймер/счетчик T2

Принцип и режимы работы таймера T2 практически не отличаются от рассмотренного ранее таймера T0, за исключением следующего:

- а) изменены коэффициенты делителя и источник задания частоты;
- б) в таймере T2 предусмотрен и асинхронный режим работы. Этот режим заключается в том, что если таймер T0 получает тактовый сигнал либо от источника частоты микроконтроллера f_{CLK} , либо от внешнего источника тактирующего сигнала, подающегося на вывод T0 (PB0), то таймер T2 может работать в асинхронном режиме работы, будучи запитанным от отдельного источника частоты (асинхронный режим). В этом режиме тактовый сигнал поступает на выходы TOSC1, TOSC0 микроконтроллера. Асинхронный режим таймера T2 при проведении лабораторных работ не используется, теоретические аспекты его использования могут быть изучены студентами самостоятельно.

Таймер T2, как и таймер T0, содержит 8-разрядный управляющий регистр TCCR2, регистр счета TCNT2 и регистр сравнения TCNT2. Назначение этих регистров подобно назначению соответствующих регистров таймера T0.

Табл. 7. Управляющий регистр TCCR2

Бит	7	6	5	4	3	2	1	0
Название	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20

Бит 7 - FOC2- бит режима принудительной установки выходного сигнала таймера в режиме ШИМ.

Бит 6 и бит 3 -WGM20:WGM21-биты режима работы таймера T2 (табл.8).

Табл. 8. Режимы работы таймера T2

WGM2	WGM2	Режим
0	0	Нормальный
0	1	Фазовый ШИМ
1	0	Очистка при совпадении
1	1	Быстрый ШИМ

Биты 5 и 4 - COM21:COM20 - режим подключения вывода OC2 таймера T2 в качестве альтернативной функции вывода PD7 порта ввода/вывода D. Таймер T2 может управлять этим выводом в соответствии с выбранным режимом работы, определяемым битами WGM21:WGM20, а также в соответствии с комбинацией разрядов COM21:COM20 (табл. 9)

Табл. 9. Функции вывода OC2 таймера T2 в нормальном режиме работы

С	С	Пояснение
0	0	Вывод OC2 отключен
0	1	Изменение OC2 на противоположное при совпадении OCF2
1	0	Очистка OC2 при совпадении OCF2
1	1	Установка OC2 при совпадении OCF2

Биты 3...0 - CS22...CS20- определяют источник задания частоты таймера T2 и делитель таймера (табл. 10).

Табл. 10. Функции бит CS22, CS21, CS20 таймера T2

С	С	С	Пояснение
0	0	0	Таймер остановлен
0	0	1	$f_{T0}=f_{CLK}/1$
0	1	0	$f_{T0}=f_{CLK}/8$
0	1	1	$f_{T0}=f_{CLK}/32$
1	0	0	$f_{T0}=f_{CLK}/64$
1	0	1	$f_{T0}=f_{CLK}/128$
1	1	0	$f_{T0}=f_{CLK}/256$
1	1	1	$f_{T0}=f_{CLK}/1024$

Табл. 11. Регистр асинхронного режима таймера T2

Бит	7	6	5	4	3	2	1	0
Название	-	-	-	-	AS2	TCN2UB	OCR2UB	TCR2UB

Источник тактового сигнала выбирается в регистре ASSR (AsynchronousStatusRegister).

Бит 3 - AS2 определяет, в каком режиме работает таймер. Если бит AS2=0, то таймер работает в синхронном режиме, используя в качестве тактового сигнала источник частоты микропроцессора. В случае, когда AS2=1, таймер получает тактовый сигнал от внешнего кварцевого резонатора, подключенного к выводам TOSC1, TOSC0 микроконтроллера.

Биты 2, 1, 0 - используются в асинхронном режиме и в данной работе не рассматриваются.

В регистре флагов таймеров TIFR (табл. 5) таймер T2 имеет два бита: бит 7 OCF2 (флаг прерывания сравнения таймера T2); бит 6 TOV2 (флаг прерывания по переполнению таймера T2).

В регистре масок прерываний таймеров TIMSK (табл. 6) таймер T2 имеет два бита: бит 7 OCIE2 (маска прерывания сравнения таймера T2); бит 6 TOIE2 (маска прерывания по переполнению таймера T2).

Контрольные вопросы

1. Сколько таймеров содержит микроконтроллер ATmega8535?
2. Какие из них 8-ми разрядные, 16-ти разрядные?
3. Какие регистры определяют работу таймера T0?
4. Поясните назначение регистра управления в целом?
5. Какие биты изменяют режим работы таймера T0? Режим работы вывода?
6. Как установить коэффициент делителя таймера T0 равный 256?
7. Объясните назначение регистров TIMSK и TIFR.
8. Какие функции в программе могут выполнять таймеры?
9. Перечислите регистры ввода/вывода, управляющие работой таймера T0?
10. В каких режимах с ШИМ могут работать таймеры микроконтроллера ATmega8535?
11. Как настроить таймеры T0/T2 для работы в режиме быстрого ШИМ?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Диаграммы работы таймера в различных режимах
3. Ответы на контрольные вопросы
4. Вывод по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Практическое занятие №11.

Изучение работы АЦП МК Atmega 8535

Цель работы: изучить работу 10-ти разрядного АЦП МК.

Выполнив работу, Вы будете:

уметь:

- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;

- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

При использовании микроконтроллера в качестве устройства управления каким-либо процессом часто возникает необходимость оценивать величины налоговых сигналов, в качестве которых могут выступать сигналы с задающих потенциометров, датчиков обратных связей, термопар и др.

Однако, поскольку микроконтроллер является цифровым устройством, непосредственно оценить величину аналогового сигнала путем опроса ножки микроконтроллера, к которой он подключен, не представляется возможным.

Для преобразования аналогового сигнала в цифровой с целью его последующей обработки предназначен аналого-цифровой преобразователь (АЦП), встроенный в микроконтроллер Atmega8535 и другие контроллеры семейства AVR фирмы Atmel.

АЦП микроконтроллера выполнен 10-разрядным, то есть аналоговый сигнал, поступающий на его вход, может быть разложен на $2^{10}=1024$ дискреты. Таким образом, фактически, разрядность АЦП отвечает за его разрешающую способность, или чувствительность преобразователя.

Разрешающая способность ЦАП - это минимальная разница аналогового сигнала при двух измерениях, которую еще различает преобразователь.

Для правильного функционирования АЦП ему необходим некий «эталон», то есть напряжение, которое будет приниматься за базу, относительно которой будет измеряться подаваемый на АЦП аналоговый сигнал. Это эталонное напряжение принято называть **опорным напряжением**. В качестве источника опорного напряжения в микроконтроллерах Atmega8535 может выступать напряжение питания контроллера, внутренний стабилизированный источник 2,56В или внешний сигнал, подключаемый к выводу AREF микросхемы контроллера.

Минимальное напряжение, которое АЦП сможет распознать, можно рассчитать по формуле:

$$U_{min} = \frac{1}{2^n} \cdot U_{ref}$$

При использовании в качестве источника опорного напряжения питания микроконтроллера 5В:

$$U_{min} = \frac{1}{2^{10}} \cdot 5 = 4,88 \text{ В}$$

Ввиду того, что в качестве АЦП в микроконтроллерах AVR используется АЦП последовательного приближения, процесс преобразования аналогового сигнала в пропорциональный ему цифровой код занимает некоторое время. Упрощенная структура АЦП последовательного приближения представлена на рис. 1. АЦП состоит из компаратора, регистра последовательного приближения и цифро-аналогового преобразователя. Работает АЦП следующим образом. В начале преобразования все выходы регистра последовательного приближения устанавливаются в состояние логического «0», за исключением старшего разряда: 10 0000 0000. При этом на выходе внутреннего цифро-аналогового преобразователя формируется аналоговый сигнал, равный половине диапазона АЦП (рис. 2, интервал $t_0..t_1$). Компаратор при этом измеряет разницу между входным сигналом и сигналом с выхода ЦАП. Если напряжение на входе АЦП оказывается больше, чем установленное на выходе ЦАП, то регистр последовательного приближения сохраняет принятое изначально состояние 10 0000 0000. Если же измеряемое напряжение оказывается меньше подаваемого с ЦАП, регистр устанавливается в состояние 01 00000000 (рис. 2, интервал $t_1..t_2$). Если принятое состояние 01 0000 0000 оказывается меньше измеряемого сигнала, 8-й бит кода закрепляется и 7-й бит кода регистра последовательного приближения устанавливается в единичное состояние: 01 1000 0000

(рис. 2, интервал t2-t3). Поскольку и при прибавлении этого бита сигнал на выходе ЦАП оказался меньше входного (рис. 2), 7-й бит закрепляется, а к коду прибавляется 6-й бит: 01 1100 0000 (рис. 2, интервал t3-t4).

В этом случае выходной сигнал ЦАП оказывается больше сигнала на входе, поэтому 6-й бит принимается равным «0», а к коду прибавляется 5-й бит: 01 1010 0000 (рис. 2, интервал t4-t5). Далее процесс повторяется до установки последнего младшего бита кода регистра последовательного приближения.

По окончании процесса преобразования результат передается в два 8-разрядных регистра **ADCH** и **ADCL**.

В микроконтроллерах AVR время преобразования АЦП можно регулировать, для этого в структуру преобразователя встроен предделитель, подобный тому, который встроен в таймеры T0...T2 микроконтроллера. Необходимо отметить, что уменьшением времени преобразования уменьшается точность получаемого результата. Рекомендуемая частота работы АЦП находится в пределах 0...200 кГц. При работе преобразователя в этом диапазоне обеспечивается минимальная погрешность при получении результата.

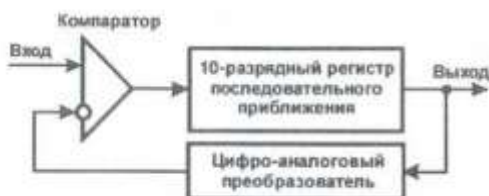


Рис. 1. Структура АЦП последовательного приближения



Рис. 2. Принцип работы АЦП последовательного приближения

Для того, чтобы правильно рассчитать время преобразования АЦП, необходимо помнить, что процесс преобразования занимает 13 тактов АЦП при его непрерывной работе и 25 циклов при его первом запуске. Аналого-цифровой преобразователь микроконтроллера Atmega8535 содержит 8 входов, подключенных к выводам порта ввода/вывода А. Поскольку преобразователь одновременно может работать только с одним входом, они коммутируются специальным коммутатором (мультиплексором), управляемым программно с помощью регистра ADMUX (табл. 1).

Табл. 1 Регистр управления мультиплексором АЦП ADMUX

Бит	7	6	5	4	3	2	1	0
Название	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

Табл. 2. Задание источника опорного напряжения АЦП

REFS1	REFS0	Источник опорного напряжения
0	0	Внешний источник, подключенный к выводу AREF.
0	1	Напряжение питания АЦП, подаваемое на вывод AVCC. К выводу AREF должен быть подключен конденсатор.
1	0	Комбинация не используется.
1	1	Внутренний стабилизированный источник 2,56 В. К выводу AREF должен быть подключен конденсатор.

2).

Биты 6 и 7 - REFS1, REFS0. С помощью этих бит определяется источник опорного напряжения АЦП (табл.

Бит 5 - ADLAR. Этот бит отвечает за «левое» выравнивание результата. Поскольку для хранения 10-разрядного результата преобразования АЦП используются два 8-разрядных регистра ADCH и ADCL, некоторые биты регистра ADCH остаются неиспользуемыми. Если ADLAR=1, то результат преобразования АЦП сохраняется «левым» выравниванием по регистру ADCH. Это

Табл. 3. Представление результата преобразования АЦП в зависимости от управляющего бита ADLAR

ADLAR=0								
ADCH		-	-	-	-	-	ADC9	ADC8
ADCL	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
ADLAR= 1								
ADCH	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2
ADCL	ADC1	ADC0	-	-	-	-		

особенно удобно, если пользователю не нужны два младших бита результата преобразования АЦП (табл. 3).

Биты 4...0 - MUX4...MUX0. С помощью этих бит определяется, какой из входов АЦП подключен к преобразователю. Здесь различают два режима работы входов АЦП - одиночный и дифференциальный (табл. 4).

При решении типовых задач обычно используется одиночное включение ходов (табл. 4), однако в некоторых случаях (например, для исключения и действия на входы микроконтроллера сигнала помехи) возникает необходимость использования дифференциальных входов. В этом случае АЦП (меряет сигнал не между конкретным аналоговым входом ADC0...ADC7 и общей точкой, а между положительным и отрицательным дифференциальными входами табл. 4).

В некоторых случаях, когда необходимо выловить разницу в очень близких сигналах, полезно перед аналого-цифровым преобразованием эти сигналы усилить.

С этой целью перед подачей на АЦП микроконтроллер может усилить дифференциальный сигнал в 1... 10.. 200 раз (табл. 4).

Для калибровки преобразователя можно также ко всем входам подключить напряжения 1,22В (MUX4...0=11110) или общую точку

Настройка аналого-цифрового преобразователя и управление им осуществляется с помощью регистра управления АЦП ADCSRA(табл. 5).

Бит 7 - ADEN. Этот бит разрешает использование АЦП. Записью логического «0» в ADENAЦП будет немедленно выключено.

Бит 6 - ADSC. Запись логической «1» в этот бит разрешает преобразование АЦП. Необходимо записывать в ADCSлогическую «1» для начала каждого преобразования. По окончании преобразования бит будет автоматически сброшен состояние логического «0».

Бит 5 - ADATE. Записью логической «1» в этот бит разрешается автоматический запуск АЦП по событию. Событие выбирается комбинацией битов ADTSв регистре SFIORмикроконтроллера.

Бит 4 - ADIF. Этот бит - флаг готовности результата преобразования АЦП. Устанавливается автоматически по окончании процесса преобразования.

Бит 3 - ADIE. Этот бит - маска прерывания готовности результата преобразования АЦП. Если ADIE=1, то при появлении флага готовности результата ADIFбудет сгенерировано прерывание.

Биты 2...0 - ADPS2...ADPS0. Комбинация этих бит устанавливает делитель тактовой частоты процессора для тактирования АЦП (табл. 6).

Табл. 4. Определение логики работы входов АЦП в соответствии с битами MUX4...MUX0

MUX4...MUX0	Одиночные входы	Положительный дифференциальный вход	Отрицательный дифференциальный вход	Коэффициент усиления
00000	ADC0			
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000		ADC0	ADC0	10
01001		ADC1	ADC0	10
01010		ADC0	ADC0	200
01011		ADC1	ADC0	200
01100		ADC2	ADC2	10
01101		ADC3	ADC2	10
01110		ADC2	ADC2	200
01111		ADC3	ADC2	200
10000		ADC0	ADC1	1
10001		ADC1	ADC1	1
10010		ADC2	ADC1	1
10011		ADC3	ADC1	1
10100		ADC4	ADC1	1
10101		ADC5	ADC1	1
10110		ADC6	ADC1	1
10111		ADC7	ADC1	1
11000		ADC0	ADC2	1
11001		ADC1	ADC2	1
11010		ADC2	ADC2	1
11011		ADC3	ADC2	1
11100		ADC4	ADC2	1
11101		ADC5	ADC2	1
11110	1.22 В			
11111	0В			

Табл. 5 Регистр управления АЦП ADCSRA

бит	7	6	5	4	3	2	1	0
название	ADEN	ADSC	ADIF	ADIF	ADIF	ADPS2	ADPS1	ADPS0

Табл. 6 Пределитель АЦП микроконтроллера Atmega 8535

ADPS2	ADPS1	ADPS0	Пределитель
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

В регистре специальных функций битами ADTS2...ADTS0 задается источник автозапуска АЦП при активации этой функции в регистре ADCSRA.

Табл. 7. Регистр специальных функций контроллера SFIOR

Бит	7	6	5	4	3	2	1	0
Название	ADTS2	ADTS1	ADTS0	-	-	-	-	-

Табл. 8. Источник автозапуска АЦП (биты ADTS2...ADTS0)

ADTS2	ADTS1	ADTS0	Источник автозапуска
0	0	0	Непрерывное преобразование АЦП
0	0	1	Аналоговый компаратор
0	1	0	Внешнее прерывание INT0
0	1	1	Прерывание по совпадению таймера T0
1	0	0	Прерывание по переполнению таймера T0
1	0	1	Прерывание по совпадению канала В таймера T1
1	1	0	Прерывание по переполнению таймера T1
1	1	1	Прерывание по захвату сигнала таймера T1

Контрольные вопросы

1. Что такое АЦП?
2. Какого типа АЦП используется в микроконтроллере Atmega 8535?
3. Сколько разрядов имеет АЦП МК?
4. Что такое опорное напряжение?
5. Что такое разрешающая способность АЦП?
6. Как рассчитать разрешающую способность АЦП (минимальное напряжение)?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Назначение битов регистров **ADCH** и **ADCL**
2. Ответы на контрольные вопросы
3. Вывод по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Практическое занятие №12.

Изучение работы сегментного и ЖК индикаторов под управлением МК Atmega 8535

Цель работы: изучить работу семисегментного и жидкокристаллического индикаторов при работе под управлением МК.

Выполнив работу, Вы будете:

уметь:

- применять нормативные документы, определяющие требования к оформлению программного кода;
- использовать возможности имеющейся технической и/или программной архитектуры
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

Не требуется

Теоретические сведения

В настоящее время подавляющее число промышленных приборов оснащаются цифровыми индикаторами для отображения различных величин. Простейший семисегментный индикатор (рис. 1) представляет набор отдельных сегментов (светодиодов), при зажигании которых в определенной последовательности можно получить набор цифр и определенных символов.

Каждый сегмент индикатора имеет унифицированное буквенное обозначение в соответствии с латинским алфавитом. Верхний горизонтальный сегмент имеет обозначение «А», все последующие сегменты по часовой стрелке имеют обозначения «В», «С», «D», «Е», «F», «G», «H» (рис. 1).

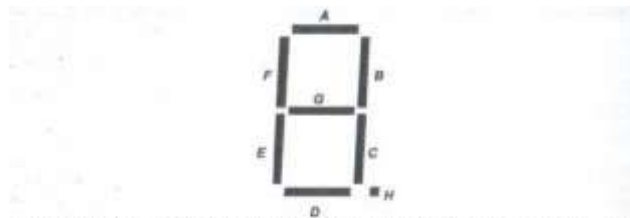


Рис. 1. Внешний вид и структура семисегментного индикатора

Для того, чтобы зажечь сегмент индикатора, необходимо подать напряжение на соответствующий светодиод А ... Н. Семисегментные индикаторы выпускаются двух типов - с общим анодом (рис. 2, а) и с общим катодом (рис. 2, б). Это делается для упрощения работы с индикатором и уменьшения количества выводов его микросхемы. Действительно, при использовании схемы с общим анодом аноды всех светодиодов объединяются, на них подается положительное напряжение.

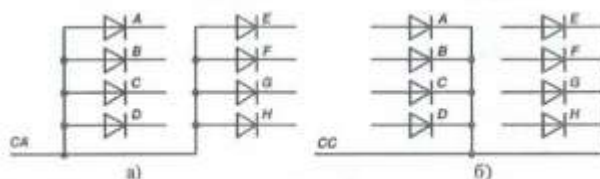


Рис. 2. Семисегментные индикаторы с общим анодом и общим катодом

Для того, чтобы зажечь, например, сегмент «С», необходимо катод соответствующего светодиода через токоограничивающий резистор присоединить к общему проводу.

При использовании индикаторов с общим катодом на него подключается шина с нулевым потенциалом, а на аноды светодиодов через токоограничивающие элементы подключается напряжение питания.

Таким образом, для того, чтобы зажечь, например, цифру 3, необходимо осуществить подачу напряжения на светодиоды А, В, С, D, G.

На практике для упрощения работы с индикаторами применяют схемы промежуточного усиления, предназначенные для усиления сигналов управления индикаторами и удобного управления их сегментами. Пример такой схемы приведен на рис. 3.

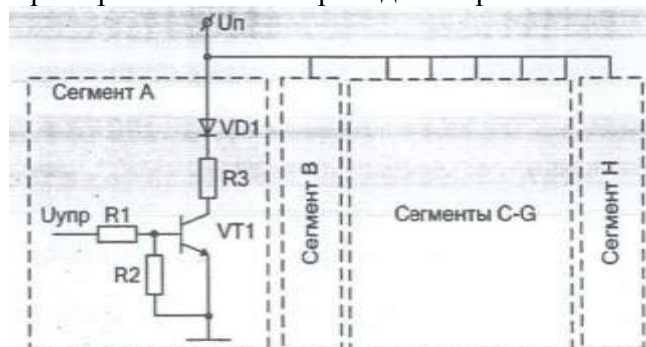


Рис. 3. Схема для управления индикатором

На общие аноды сегментов подается напряжение электропитания $U_{п}$. Катоды сегментов через токоограничивающий резистор и транзистор подключаются к общей шине. Очевидно, что для зажигания сегмента необходимо включить транзистор (для сегмента А - транзистор VT1).

Включение транзистора VT1 осуществляется подачей на его базу тока управления, который появляется при подаче напряжения управления $U_{упр}$ на токоограничивающий резистор R1. Таким образом, при подаче от микроконтроллера сигнала логической «1» транзистор VT1 открывается и светодиод VD1 начинает светиться. Применение рассмотренной схемы позволяет включать сегменты сигналами логической «1», а не логического «0».

Динамическая индикация символов

Часто требуется осуществлять индикацию больших чисел, т.е. обращаться не к одному индикатору, а к нескольким.

Рассмотрим варианты индикации числа 1234 на четырех семисегментных индикаторах. В примере будет использован индикатор с общим анодом (рис. 2, а).

В простейшем случае к общим анодам разрядов индикатора необходимо подключить напряжение питания, а на выводы А...Н каждого разряда подать соответствующую кодовую комбинацию (рис. 4). В случае подобной реализации многоразрядного индикатора возникает существенная проблема - при использовании четырех разрядов количество выводов микроконтроллера, используемых для индикации, составляет 32 шт, при этом всего рабочих выводов у микроконтроллера Atmega8535 - 32, то есть ресурсы контроллера будут использованы полностью.

Для того, чтобы минимизировать ресурсоемкость процесса индикации, предлагается использовать метод динамической индикации. Этот метод основан на свойстве инерции человеческого зрения, при котором глаз не воспринимает разницу между быстро сменяющимися картинками, если они меняются с частотой, превышающей 25 Гц. Схема динамической индикации представлена на рис. 5.



Рис. 5. Схема реализации динамической индикации символов

При динамической индикации соответствующие катоды всех индикаторов объединяются, то есть катоды сегмента А индикаторов HG1...HG4 объединяются в шину А, катоды сегмента В объединяются в шину В и т.д. Общие аноды индикаторов HG1..HG2, наоборот, разъединяются.

Если индикатор реализован подобным образом, то последовательность динамической индикации следующая:

- микроконтроллер выдает код числа 4 на катоды всех индикаторов, при этом напряжение питания подается только на разряд HG4. В результате цифра 4 светится только на индикаторе HG4;

- спустя время, не большее 1/100 секунды, на все сегменты выдается код числа 3, при этом напряжение питания подается только на разряд HG3. В результате цифра 3 светится только на индикаторе HG3;

- на следующем интервале времени на все индикаторы выдается код числа 2, при этом напряжение питания подается только на разряд HG2. В результате цифра 2 светится только на индикаторе HG2;

- в конце цикла на индикаторы подается код числа 1, а напряжение питания - на индикатор HG1. В результате цифра 1 светится только на разряд HG1. Далее процесс повторяется.

Поскольку каждый разряд индикатора обновляется с частотой как минимум 25 Гц, человеческий глаз не замечает мерцания индикатора и процесс отображения числа 1234 кажется постоянным, хотя в один момент времени всегда светится только один разряд индикатора. С увеличением частоты обновления цифр качество индикации улучшается.

Контрольные вопросы

1. Каково назначение семисегментного индикатора?
2. Как конструктивно можно реализовать семисегментный индикатор?
3. В чем смысл динамической индикации?
4. С какой частотой необходимо подавать сигналы управления на индикатор при реализации динамической индикации?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Схемы подключения и управления индикатором

3. Ответы на контрольные вопросы
4. Вывод по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.