Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра дискретной
математики и
информационных
технологий

«Разработка учебного пособия по программированию микроконтроллеров

полное наименование темы выпускной квалификационной работы в кавычках

семейства HCS12 для преподавания курса “ЭВМ и периферийные устройства”»

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студента 4 курса 421 группы
направления 09.03.01 – Информатика и вычислительная техника
факультета КНиИТ
Смагина Владислава Дмитриевича

Научный руководитель
к. ф.-м.н., доцент _____ А.Н. Савин

подпись, дата

Зав. кафедрой
к. ф.-м.н., доцент _____ Л.Б. Тяпаев

подпись, дата

Саратов 2017 год

ВВЕДЕНИЕ

Микроконтроллеры используются во всех сферах жизнедеятельности человека, устройствах, которые окружают его. Их можно встретить в огромном количестве современных промышленных и бытовых приборов: станках, автомобилях, телефонах, телевизорах, холодильниках, стиральных машинах и даже кофеварках. На данный момент популярна разработка микропроцессорных систем управления и соответственно их программирование, поэтому актуальным также является разработка специального учебного материала, обеспечивающего эффективное обучение студентов программированию и разработке микропроцессорных систем управления на базе микроконтроллера реального времени.

В лаборатории теоретических проблем информатики и её приложений кафедры дискретной математики и информационных технологий имеется оборудование компаний National Instruments и FreeScale Semiconductor:

- лабораторный стенд ELVIS II;
- плата CSMB12C128;
- плата PBMCUSLK.

Имеющийся материал по программированию данного оборудования предполагает обучение в течение нескольких семестров, поэтому возникла задача разработки методического пособия способного эффективно обучить программированию микроконтроллеров семейства HCS12 в течение одного семестра в курсе “ЭВМ и периферийные устройства” и далее изучать разработку микропроцессорных систем управления для курса “Системы реального времени”.

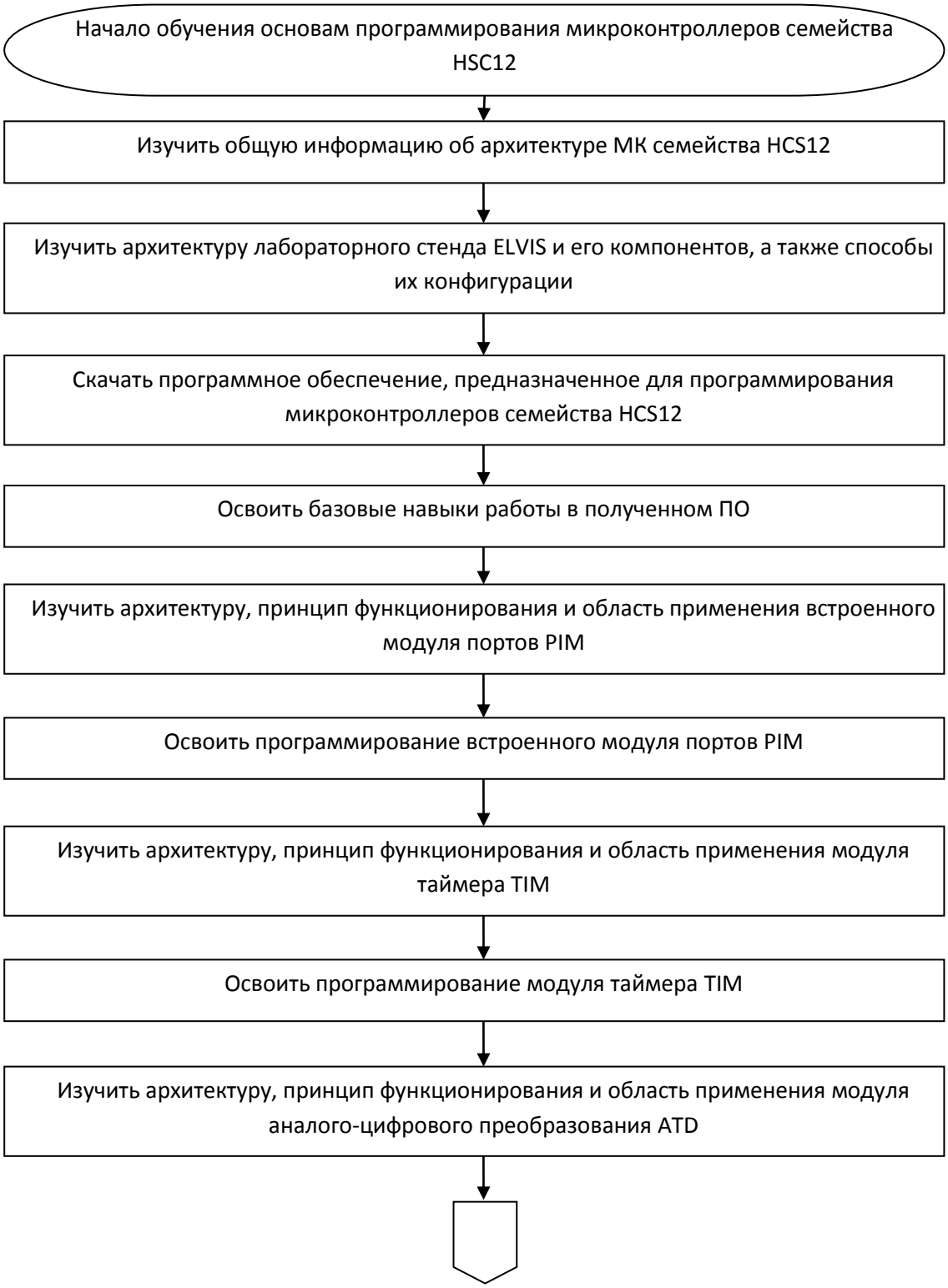
Таким образом, целью бакалаврской работы является разработка данного пособия. Для достижения этой цели были сформулированы следующие задачи:

- изучить архитектуру микроконтроллеров FreeScale семейства HCS12X и способы применения их функциональных компонентов по имеющейся технической информации;
- изучить архитектуру лабораторного стенда NI Elvis II и плат фирмы FreeScale, предназначенных для работы с микроконтроллерами;
- освоить базовые навыки работы в среде разработке CodeWarrior;
- сформулировать задания для лабораторных работ;
- подготовить примеры выполненных заданий;
- подготовить теорию для выполнения лабораторных работ;
- устранить выявленные в ходе апробации недостатки;
- подготовить разрабатываемое учебное пособие к изданию.

Основное содержание работы

1 Описание хода выполнения лабораторных работ

Для разработки учебного пособия была выявлена последовательность действий, с помощью которой можно описать ход выполнения лабораторных работ. Данная последовательность представлена в виде блок-схемы на рисунке 1.



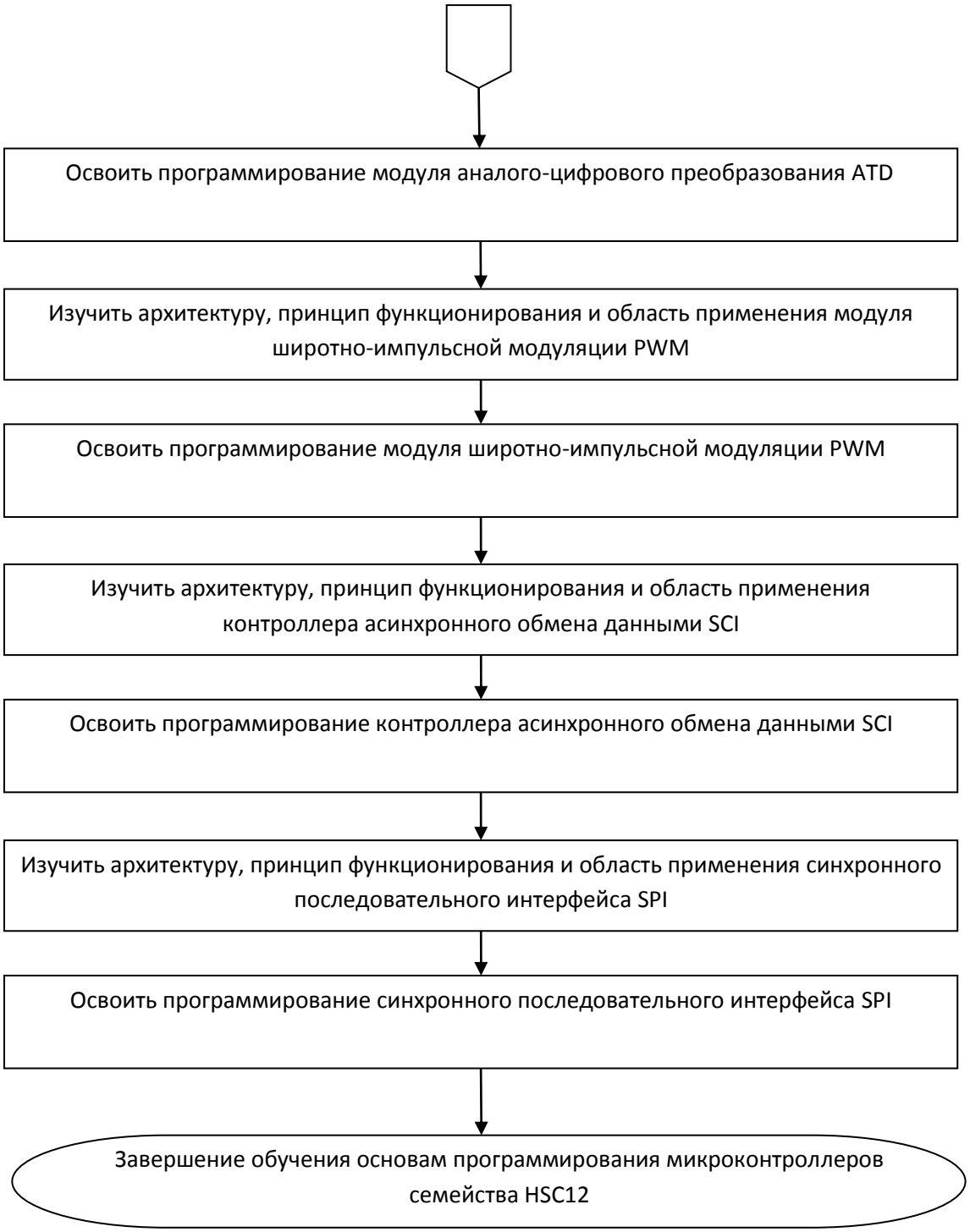


Рисунок 1 – Блок-схема процессам обучения основам программирования микроконтроллеров семейства HCS12

В следующих пунктах будут описаны процессы блок-схемы, указанной на рисунке 1.

1.1 Описание среды разработки CodeWarrior

CodeWarrior – это интегрированная среда разработки (IDE – Integrated Development Environment) программного обеспечения для встраиваемых микропроцессорных систем. Пакет CodeWarrior содержит в себе несколько связанных друг с другом программных модулей: менеджер проектов, редактор с интерактивной коррекцией ошибок, модуль управления компиляцией и созданием программы (Build system), символьный отладчик и др. Оболочка IDE CodeWarrior – это стандартное Windows-приложение. Она позволяет создавать файлы с текстом программы и организовывать их в проект, определяющий конечное приложение. В дополнение к этому среда разработки обеспечивает автоматическую компиляцию, ассемблирование и линковку/загрузку всех необходимых исходных данных для получения готового абсолютного модуля программы для микроконтроллера. Управляющая оболочка CodeWarrior имеет два типовых режима работы:

- Режим «построения» программы (Build) позволяет транслировать все файлы приложения и получать готовый запускаемый модуль программы.
- Режим отладки (Debug) обеспечивает отладку и испытание приложения с помощью встроенного отладчика или симулятора.

1.1.1 Пример программы на языке С в среде разработки CodeWarrior

Текст, расположенный после символов “//”, называется комментарием и при компиляции программы не учитывается. Используйте комментарии для пояснения выполняемых вами смысловых действий прикладной программы.

В PORTB всего 8 бит, первые 4 бита PB0-PB3 подключены к переключателям SW. Остальные же 4 бита PB7-PB4 подключены к 4 светодиодам, которые мы включаем и выключаем с помощью прерывания таймера. Поэтому мы определили старшие биты на вывод, где расположены наши лампочки, чтобы эти данные порта B отправлялись на ножки светодиодов.

А младшие биты – на ввод, чтобы получать данные с выводов переключателей на порте B.

1.1.2 Лабораторная работа “Создание простейших программ на языке C в среде разработки CodeWarrior”

Цели работы:

- Научиться создавать проект и файлы проекта в среде разработки CodeWarrior.
- Научиться создавать простейшие программы на языке C в среде разработки CodeWarrior.
- Получить навыки работы в режиме Debug для освоения процесса отладки программ.

1.2 Особенности программирования встроенного модуля портов PIM и мультиплексированного интерфейса внешней шины MEBI

Для программирования портов модуля PIM можно использовать язык Ассемблера, встроенный в среду разработки CodeWarrior. Это позволит обучающимся в лучшей степени понять принцип функционирования портов и правила работы с ними, так как в языке C работа с портами упрощена до минимума, путем использования библиотек.

При программировании микроконтроллера на языке Ассемблера в отличие от языка C приходится решать следующие дополнительные задачи:

- Осуществить распределение адресов карты памяти. Это необходимо, так как у каждого типа МК своя карта памяти.
- Изучить команды языка Ассемблера данного микроконтроллера. Как говорилось ранее, возникают случаи, когда использовать команды языка Ассемблера удобнее при написании какого-то фрагмента кода, да и самый оптимальный вариант при написании программы, это использование обоих языков, поэтому знание хотя бы самых базовых команд и структуры программы на языке Ассемблера необходимо. Полной список команд языка

Ассемблера на английском языке представлен в среде программирования Freescale CodeWarrior – Help/CodeWarrior Help/HC(S)12(X)/Assembler Directives.

После ознакомления с картой памяти, стало понятно, что у нее есть особенность: адреса разных блоков памяти накладываются друг на друга. В момент написания кода это может быть неудобно, и вы должны правильно организовать структуру карты памяти.

С помощью команд EQU именам присваиваются значения. Потому что это псевдокоманды, и они не выполняют реальных действий в программе. При ассемблировании первая команда будет размещена в ячейке памяти с адресом $E000.

Ранее было сказано, что у каждого есть свой адрес в памяти МК, поэтому во время программирование МК нужно правильно устанавливать адреса портов, т.к. регистры данных портов расположены по строго определенным в техническом описании физическим адресам. Если вы укажете неверный адрес, то информация попадет в другой регистр, а может и в другой модуль.

Подробнее рассмотрим направление передачи данных порта. Установка направления передачи порта на вывод означает, что все данные, которые лежат в порте будут отправлены в модуль PIM, и действия с ними буду производиться в зависимости от назначения ножек порта. В обратном случае, все данные будут записываться в порт, и оставаться по адресу регистра данных этого порта.

Для того чтобы не занимать память промежуточными значениями, в МК MC9S12C128 существуют регистры аккумуляторы A и B, и индексные регистры X и Y. С помощью них можно сохранить любой результат, которые вам потребуется в дальнейшем, например: придержать значение порта для его инверсии или выполнения некоторой логической операции.

1.2.1 Лабораторная работа “Порты модуля PIM и работа с ними”

Цель работы:

- Изучить устройство и принцип функционирования модулей PIM и MEBI микроконтроллеров семейства HCS12.
- Освоить программирование модулей PIM и MEBI микроконтроллеров семейства HCS12.
- Освоить работу с подсистемой памяти микроконтроллеров семейства HCS12.
- Освоить использование базовых команд языка Ассемблера в среде разработки CodeWarrior.

1.3 Примеры программирование модуля таймера TIM и подсистемы прерываний реального времени RTI

Познакомившись с основными подсистемами модуля таймера, детально разберем процессы его инициализации и программирования. Прежде чем приступить к процессу инициализации, нужно детально ознакомиться с выполняемой задачей, потому что от нее напрямую зависит выбор настраиваемых регистров. Мною были выделены два типа задач при работе с таймером:

- ожидание изменения фронта на канале таймера;
- ожидание завершения некоторого периода времени.

В основном эти два типа различаются в выборе прерываний, по которому будем срабатывать таймер, но о них поговорим чуть позже. Особенность первого типа заключается в подаче сигнала на порт таймера, где срабатывает режим входного захвата в момент изменения логической 1 на логический 0 или наоборот в последовательности сигнала, представленной в двоичном виде. Второй тип задач используется в тех случаях, когда необходимо выполнять некоторое действие через одинаковый промежуток времени, например: установить дату или выполнить какую-либо проверку. Именно второй тип

задач идеально описывает работу подсистемы прерываний реального времени.

После определения типа задач необходимо выбрать тип прерывания и разрешить микроконтроллеру их использование командой EnableInterrupts. Прерывания бывают 4 типов:

- по активному событию в канале 0-7(C[7:0]F). Таймер будет реагировать на нарастающий фронт полученных данных с канала 0-7 (например – переход от 0 к 1 полученного сигнала);
- по активному событию входных данных импульсного аккумулятора (PAOVI). Таймер будет реагировать на нарастающий фронт полученных данных с импульсного аккумулятора;
- по переполнению импульсного аккумулятора (PAOVF);

по переполнению счетчика таймера (TOF), крайним числом является 65536, т.е. в момент, когда счетчик дойдет до этого числа, он сброситься до 0.

При написании программы необходимо следить за флагами таймера (TFLG1, TFLG2 и PAFLG), то есть при выполнении прерывания его необходимо сбрасывать, так как если этого не сделать, то после обслуживания прерывания сгенерируется новый запрос по флагу и микроконтроллер снова попадёт на тот же фрагмент кода.

1.3.1 Лабораторная работа «Программирование модуля таймера TIM и подсистемы реального времени RTI в составе HCS12»

Цель работы:

- Изучить устройство и принципы функционирования модуля таймера TIM.
- Освоить программирование модуля таймера TIM с использованием подпрограмм прерываний.
- Изучить устройство и принципы функционирования подсистемы прерываний реального времени RTI.
- Освоить программирование подсистемы прерываний реального времени RTI.

− Освоить все возможные режимы работы модуля таймера TIM.

1.4 Примеры программирования модуля аналого-цифрового преобразования ATD

При программировании модуля ATD нужно понимать, что принципом работы АЦП является преобразование сигналов, т.о. устройство, генерирующее аналоговый сигнал должно быть подключено к входам AN0…AN7. Важным фактом является, что напряжение измеряемых сигналов должно находиться в диапазоне 0 до 5,0 В. На стенде таким прибором является потенциометр.

Как и в работе с таймером, во время программирование используются прерывания, но в данном случае их выбор ограничивается одним прерыванием. Оно срабатывает по окончанию вычислений модуля ATD. Прерывания разрешаются посредством установки бита ASCIE регистра ATDCTL2 в 1. При этом запрос на прерывание генерируется, если устанавливается бит ASCIF в том же регистре. Этот же бит следует сбросить в подпрограмме прерывания, чтобы по завершению следующей измерительной последовательности был сгенерирован новый запрос. Но есть и другой способ узнать о завершении вычислений - метод программного опроса триггера завершения измерительной последовательности SCF в регистре состояния ATDSTAT. Данный способ заключается в опросе флагов CCF7…CCF0 регистра ATDSTAT, которые устанавливаются в 1 или 0 при записи в регистр результата, т.е. мы можем программно узнать о завершении того или иного преобразования. Это удобно при работе с динамичными объектами, когда нужно знать о завершении не всей преобразованной последовательности, а только об её части. Также в программе должно быть разрешение прерываний, только на этот раз мы воспользуемся командой: asm CLI, где asm означает, что это команда языка Ассемблера.

Важной особенностью также является необходимость установки задержки, как минимум в 40мкс, после включения модуля АЦП, это обуславливается тем, что аналоговые компоненты АЦП должны прийти в

рабочее состояние после его включения. Инициализацию модуля АЦП нужно производить последовательно, т.е. сначала произвести запись в регистр ATDCTL2, затем – в ATDCTL3 и т.д. по возрастанию имён. В противном случае, даже если все команды верны, конфигурируемое устройство может не включиться или заработать неправильно.

Модуль ATD может работать как непрерывно, так и осуществлять однократные преобразования. Для того чтобы подробнее разобраться в работе модуля ATD, мы будем реализовывать второй режим. Его особенностью является тот факт, что начинать новое преобразование можно только тогда, когда данные от предыдущего уже обработаны, поэтому наблюдать за его работой будет удобнее.

Рассмотрим одну особенность. Как говорилось выше, при использовании прерывания, необходимо сбрасывать значение флага в его подпрограмме, чего мы не сделали. Это объясняется тем, что сброс флага окончания преобразования (SCF, бит 7 в регистре ATDSTAT0) осуществляется автоматически при записи данных в регистр ATDCTL5. Это действие одновременно запустит АЦП на новое преобразование. То есть при использовании данного режима, после каждого вычисления необходимо заново запускать последовательность преобразования.

1.4.1 Лабораторная работа “Программирование модуля аналого-цифрового преобразования ATD”

Цель работы:

- Изучить устройство и принципы функционирования модуля аналого-цифрового преобразования ATD.
- Освоить программирование модуля аналого-цифрового преобразования ATD.
- Освоить все возможные режимы работы модуля аналого-цифрового преобразования ATD.

1.5 Особенности программирования модуля широтно-импульсной модуляции PWM

Модуль PWM требует инициализации большого количества регистров для определения параметров работы модуля, но после выполнения инициализации, программирование модуля в основном заключается в смене коэффициента заполнения и периода следования. Важным аспектом при работе с модулем является правильный подбор параметров, таких как:

- разрешающая способность генерируемого ШИМ-сигнала, т.е. число дискретных отсчетов частоты тактирования канала в периоде и длительности импульса выходного сигнала канала (8-разрядном или 16-разрядном);
- частота генерируемого сигнала;
- структуру подсистемы тактирования;
- выбор режима работы ШИМ (центрированной или фронтовой).

Также следует отметить: чтобы наблюдать работу модуля, что крайне полезно для изучения принципов его работы, необходима графическая визуализация. В лабораторном стенде присутствует такой элемент как Scope, позволяющий визуально наблюдать за выходным сигналом. Также можно воспользоваться внешними устройствами, но в таком случае перед началом работы необходимо настроить стенд, чтобы выходы порта P были соединены с таким устройством, и линии порта были настроены на вывод.

Список областей применения ШИМ огромен, но в основном на микроконтроллерах они используются для запуска работы двигателей. Для того чтобы визуально наблюдать работу широтно-импульсной модуляции на микроконтроллере семейства HCS12 можно воспользоваться светодиодами, только в данном случае, мы будем управлять яркостью.

1.5.1 Лабораторная работа “Программирование модуля широтно-импульсной модуляции PWM”

Цели работы:

- Изучить устройство и принципы функционирования модуля широтно-импульсной модуляции PWM.
- Освоить программирование модуля широтно-импульсной модуляции PWM.
- Освоить все возможные режимы работы модуля широтно-импульсной модуляции PWM.

1.6 Примеры программирование контроллера последовательного обмена SCI

В процессе программирования контроллера SCI следует соблюдать выполнение трех последовательностей действий, каждую из которых далее рассмотрим подробно:

- инициализацию приемника и передатчика контроллера;
- управление процессом передачи информации;
- управление процессом приема информации.

При инициализации контроллера рекомендуется выполнить следующие настройки:

- настройка порта S (0 и 1 биты отвечают на модуль SCI);
- установить скорость обмена;
- выбрать формат кадра обмена: 8 или 9 бит данных;
- назначить параметры приема и передачи в регистрах управления SCxCR1 и SCxCR2.

Стоит отметить, что аппаратные средства модуля SCI предполагают, что и прием, и передача информации могут происходить только с одинаковой скоростью, с одинаковым форматом кадра и одинаковой логикой паритета.

Также следует отметить важную особенность, прежде чем переходить к следующей задаче. Во время инициализации модуля необходимо всегда

очищать флаги, отвечающие за передачу или прием информации, так как ваши данные могут не передаться, или вместо них будут передаваться совсем иные данные. Эти флаги подробно описаны в предыдущем пункте, и отвечают они за готовность к отправке/приему данных.

Следующие две последовательности действий очень похожи, за исключением того, что первая передает символы, а вторая – принимает. Ввиду этого, будем рассматривать процесс передачи символов. Начинать процесс передачи одного символа следует с проверки флага TDRE. Если он равен единице, то регистр данных передатчика пуст и готов к приему следующего байта. Далее, если у нас обмен настроен на 8-битный формат, то загружаем данные в регистр SCIDRL, а если 9-битный формат, то в регистры SCIDRH: SCIDRL. После загрузки данных в регистр, их передача начинается автоматически и нам необходимо следить за флагом TC, символизирующим о завершении операции.

1.6.1 Лабораторная работа “Программирование контроллера последовательного обмена SCI”

Цель работы:

- Изучить устройство и принципы функционирования контроллера последовательного обмена SCI.
- Освоить программирование контроллера последовательного обмена SCI.
- Освоить режим передачи данных контроллера последовательного обмена SCI.

1.7 Примеры программирование синхронного последовательного интерфейса SPI

Как вы могли заметить, в данном примере мы установили активный уровень SS в самом начале и более его не трогали. Это обуславливается тем, что в нашем примере не предусматривается второе устройство с контроллером SPI, и выходы порта SS отключены установлением в 0 MODFEN в регистре

управления SPICR2.

В предыдущем примере мы рассматривали простейшую пересылку данных, используя только интерфейс SPI микроконтроллера. Это лишь часть возможностей этого интерфейса. Для того чтобы полностью освоить режимы его работы, необходимо использовать как минимум два таких интерфейса. В нашем случае вторым интерфейсом является ЖКИ (жидкокристаллический индикатор), имеющий в своем составе интерфейс SPI. Он располагается на плате PBMCUSLK, установленной в лабораторном стенде ELVIS II. Этот пример также является шаблоном, в котором МК с помощью синхронного последовательного интерфейса может обмениваться данными с ЖКИ. Для этого в регистры данных SPIDR необходимо записать число для передачи. МК работает в режиме Master с частотой тактирования 12 МГц, причем бит SPISWAI установлен в 1 для корректной работы модулей. Интерфейс SPI ЖКИ по умолчанию настроен в режиме Slave, причем в библиотеке LCD.h описаны методы работы с этим устройством.

1.7.1 Лабораторная работа “Программирование синхронного последовательного интерфейса SPI”

Цели работы:

- Изучить устройство и принципы функционирования синхронного последовательного интерфейса SPI.
- Освоить программирование синхронного последовательного интерфейса SPI.
- Освоить режим обмена данных между двумя устройствами с помощью синхронного последовательного интерфейса SPI.

2 Апробация лабораторных работ

В ходе выполнения лабораторных работ была произведена их апробация на занятиях заочного отделения 421 группы факультета КНиИТ кафедры ДМиИТ курса “ЭВМ и периферийные устройства”. Студентам были выданы задания с необходимой для них теорией, чтобы выявить недостатки предложенного материала, которые в последствие устранить, а также оценить, насколько удобен для восприятия написанный материал. Во время проведения апробации были выявлены и устранены следующие недостатки:

- ошибки в текстах примеров программы;
- недостаточное количество информации в комментариях к примерам;
- отсутствие блок-схем алгоритмов к некоторым лабораторным работам;
- недостаточное количество ссылок на предоставленную теорию

ЗАКЛЮЧЕНИЕ

В ходе бакалаврской работы было разработано методическое пособие по программированию микроконтроллеров семейства HCS12, а также проведена его апробация. Для достижения этой цели были выполнены следующие задачи:

- изучена архитектура микроконтроллеров семейства HCS12X и способы применения их функциональных компонентов по имеющейся технической информации;
- изучена архитектура лабораторного стенда NI Elvis II и плат фирмы FreeScale, предназначенных для работы с микроконтроллерами;
- освоены базовые навыки работы в среде разработке CodeWarrior;
- сформулирован текст заданий для лабораторных работ;
- подготовлены примеры выполненных заданий;
- подготовлена теория для выполнения лабораторных работ;
- устранены выявленные в ходе апробации недостатки;
- разработанное учебное пособие подготовлено к печати в соответствии с требованиями издательства.

Таким образом, все поставленные в бакалаврской работе задачи выполнены.