МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

Радиотехнические комплексы контроля полёта и управления микро/наноспутников

Электронные методические указания к лабораторному практикуму

САМАРА

2010

Составители: КУДРЯВЦЕВ Илья Александрович
КОРНИЛИН Дмитрий Владимирович

Методические указания к лабораторному практикуму по дисциплине «Радиотехнические комплексы контроля полёта и управления микро/наноспутников» включает комплекс лабораторных работ, в котором рассмотрены вопросы составления и отладки программ на языке Си для микроконтроллеров семейства ARM7 применительно к продукции фирмы NXP, Рассмотрены вопросы составления и отладки программ на языке ассемблера и Си для процессоров цифровой обработки сигналов. Описаны приемы работы со средой программирования Visual DSP. Приведен пример программы реализующей цифровой фильтр; рассмотрены вопросы разработки импульсных преобразователей без гальванической развязки на базе пакета LTspiceIV. Рассмотрены вопросы разработки программ для микроконтроллеров MSP430 на языках Си и ассемблер. Продемонстрирована работа со средой программирования IAR Embedded Workbench и основные приемы отладки программ. Приведены краткие справочные сведения по системе команд и архитектуре MSP430, необходимые для выполнения лабораторной работы. Рассмотрены вопросы разработки цифровых устройств на базе FPGA и CPLD фирмы XILINX в графическом режиме и с помощью VHDL. Приведены примеры разработки устройств с применением различных приемов работы с ПЛИС, описана работа с отладочной платой на базе FPGA SPARTAN-3.
Методические указания предназначены для магистрантов, обучающихся по магистерской программе «Космические информационные системы и наноспутники. Навигация и дистанционное зондирование» по направлению 010900.68 «Прикладные математика и физика».
Методические указания разработаны на межвузовской кафедре космических исследований.

Лабораторная работа №1 «Разработка программного обеспечения для микроконтроллеров семейства ARM7 фирмы NXP»

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Микроконтроллеры семейства LPC21XX с RISC ядром ARM7TDMI являются высокопроизводительными устройствами фоннеймановской архитектуры с богатым набором периферии. Высокая тактовая частота, тридцатидвухразрядные шины делают эти контроллеры перспективными для разработки систем управления микроспутниками.

Микроконтроллеры обладают значительным объемом FLASH-памяти на кристалле и имеют модуль ускорения доступа, увеличивающий эффективность кода. Микроконтроллеры снабжены системой внутрисхемного программирования на базе интерфейса JTAG и специальным загрузчиком, позволяющим загружать программу через асинхронный приемопередатчик. Средства разработки включают в себя поддержку языка Си, использование которого рассматривается в данных методических указаниях.

Среда разработки IAR Embedded Workbench позволяет работать в режиме симулятора, причем способность использовать макросы расширяет возможности отладчика, несмотря на отсутствие прямой поддержки периферийных устройств. Большая часть работы посвящена изучению особенностей микроконтроллера посредством отладочной платы с микроконтроллером LPC2148. Установленные модули (LCD дисплей, матричный индикатор, шаговый двигатель) позволяют изучить основные пути применения микроконтроллера в системах управления.

Методические указания позволяют студентам изучить основы применения языка Си для разработки программ микроконтроллеров с ядром ARM. Указания не претендуют на полноту описания особенностей ядра ARM7TDMI, микроконтроллеров семейства LPC21XX фирмы NXP или среды разработки IAR Embedded Workbench, приводятся лишь краткие пояснения, необходимые для понимания приведенных фрагментов кода, в приложении приводится фрагменты перевода технической документации на микроконтроллер.

1 Создание нового проекта

Написание и отладку программ для микроконтроллеров будем производить с помощью среды интегрированной разработки IAR EMBEDDED WORKBENCH IDE, разработанной фирмой IAR SYSTEMS.

Для работы необходимо создать директорию с проектом, в которой могут быть размещены необходимые файлы с текстом исходной программы, служебные файлы и выходной файл. Рекомендуется создать папку с именем, набранным латинскими буквами, для избежания проблем с интерпретацией имени в ряде случаев.

Для создания нового проекта удобно использовать меню «Project/Create New Project». После активизации этого меню (рис. 1) программа предложит выбрать тип проекта, который предполагается создать. Из предлагаемого списка выберем «Empty project», в следующем диалоговом окне укажем место размещения (в предварительно созданной папке) и имя проекта (также латинскими буквами). После этого, в любое время можно добавить дополнительные файлы или удалить уже включенные в состав проекта.

Для экономии времени мы будем использовать заранее подготовленный набор файлов, находящийся в каталоге Samples (точное местоположение указывается преподавателем). По мере необходимости мы будем копировать оттуда файлы и вставлять их в проект. **ВНИМАНИЕ ! Запрещается удалять и изменять файлы в каталоге Samples, а также добавлять свои файлы. Все изменения производить только с копиями этих файлов в созданном собственном каталоге.**

Для начала работы над первым проектом скопируйте из каталога Samples в свою папку каталог Configurations, а также файлы Main.cpp, lcd.cpp

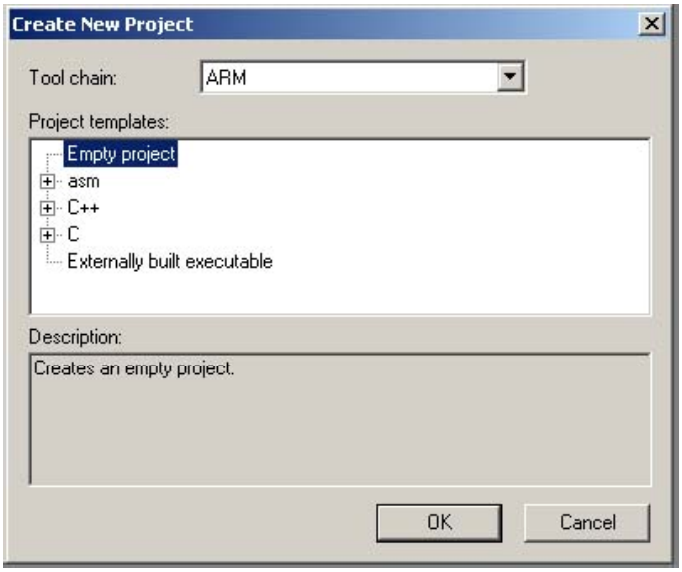


Рисунок 1 – Создание проекта

Обратите внимание, что в окне Workspace слева создано по умолчанию две конфигурации проекта: Debug и Release. Мы можем добавить и свои конфигурации, например, полезно иметь конфигурацию для загрузки кода программы в ОЗУ контроллера. В целом общепринятым является использование конфигурации Debug для отладки проекта, а конфигурации Release – для размещения готового проекта с применением оптимизаций компилятора. В нашей работе мы настроим конфигурацию Debug для отладки проекта с использованием симулятора, конфигурацию Release – для размещения кода проекта в энергонезависимой памяти контроллера, и создадим конфигурацию RAM (для экспериментов с размещением кода в ОЗУ контроллера).

Теперь необходимо настроить свойства конфигураций проекта. Начнем с конфигурации Debug. Для этого выберем эту конфигурацию, щелкнем правой кнопкой мыши по названию проекта и выберем вкладку «Options…», после чего откроется окно, показанное на рис. 2. Здесь выбираем вкладку «General Options» и далее тип устройства – LPC2148 фирмы NXP.

Для того, чтобы использовать синтаксис C++ нужно выбрать вкладку C/C++ compiler и указать в группе «Language» «Automatic» или «Embedded C++»

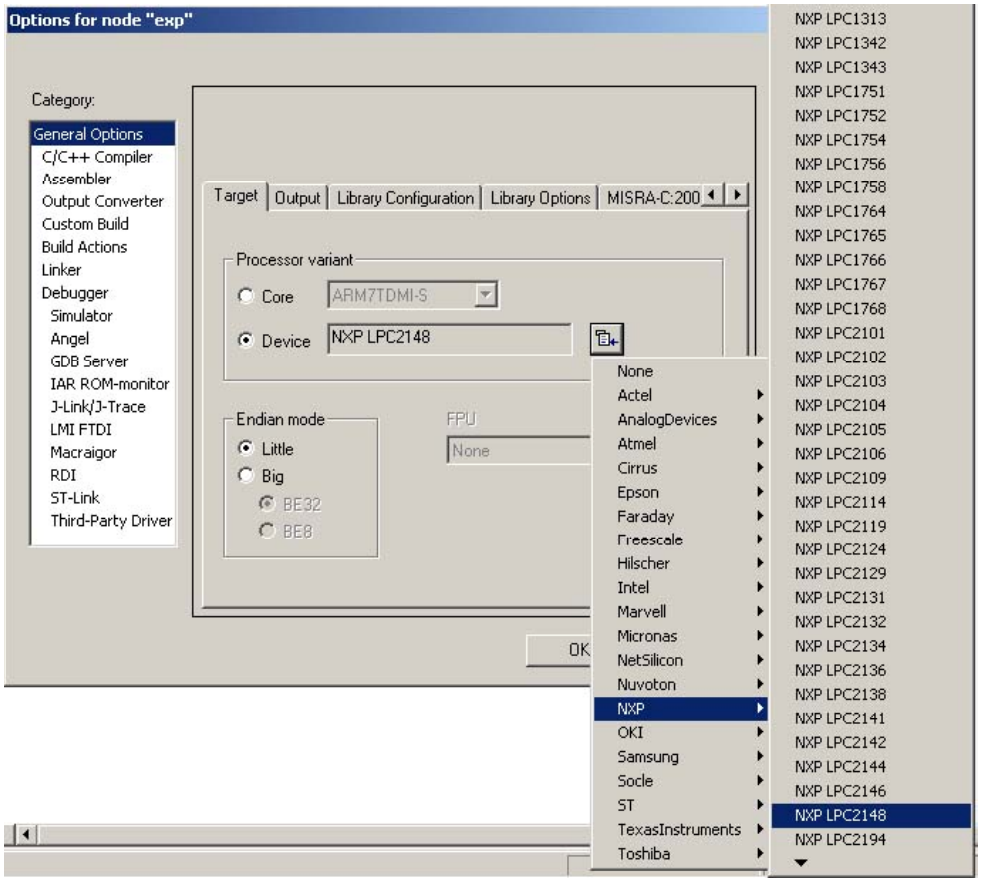


Рисунок 2 – Выбор типа устройства

Далее выберем пункт «Debugger» и в окне «Driver» установим «Simulator». Остальные опции оставим в том виде, как это предлагает программа.

Для настройки отладочной конфигурации с размещением кода в энергонезависимой памяти (конфигурация «Release») аналогичным образом установим тип микроконтроллера, далее выберем пункт Linker, поставим галочку в окне «Override default» и укажем путь к файлу LPC2148_flash.icf, находящемуся в подкаталоге Configurations. Выберем вновь пункт «Debugger» и в окне «Driver» установим J-Link/J-Trace.

Для создания отладочной конфигурации с размещением кода в ОЗУ воспользуемся меню «Project\Edit Configurations», далее в диалоговом окне выберем «New» и создадим новую конфигурацию RAM. Далее необходимо соответствующим образом настроить эту конфигурацию аналогично конфигурации Release, для чего в окне Categories выберем пункт Debugger и в окне Driver справа укажем отладчик, как показано на рис. 3. Установите галочку в окне «Use macro file(s)» и укажите путь к файлу LPC2148_RAM.mac, находящемуся в подкаталоге «Configurations». Теперь выберем пункт Linker, поставим галочку в окне «Override default» и укажем путь к файлу LPC2148_RAM.icf, находящемуся в том же подкаталоге.

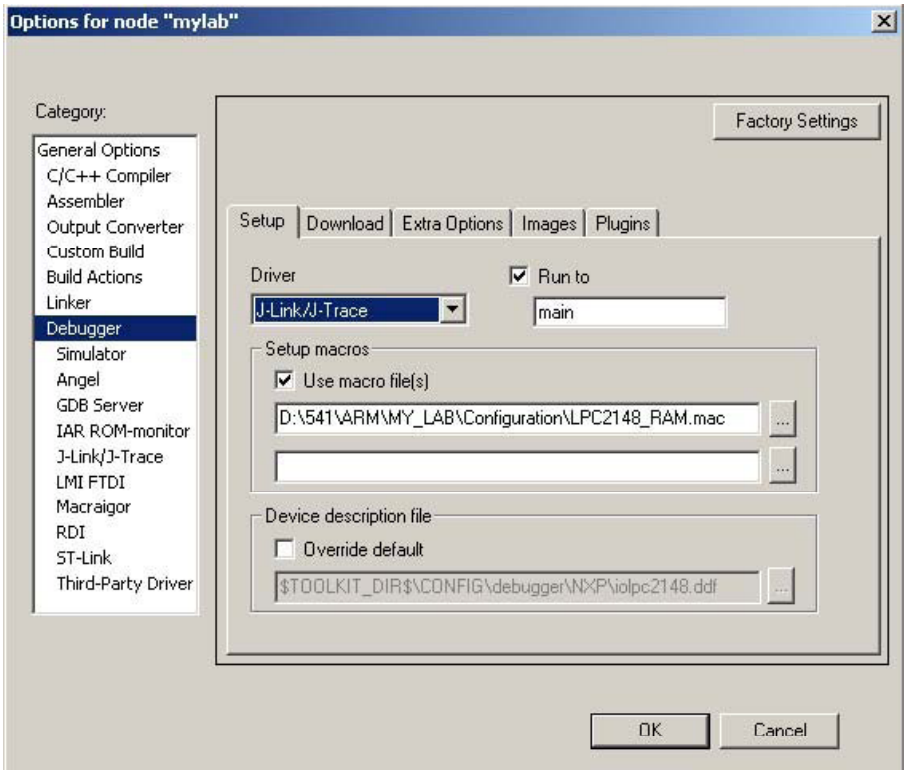


Рисунок 3 – Настройка отладчика

Теперь щелкнем правой кнопкой мыши по названию проекта, выберем пункт «Add Files» и добавим в проект файлы «Main.cpp» и «lcd.cpp». Теперь проект готов к компиляции и отладке.

2 Компиляция и отладка программы

2.1 Компиляция проекта

Для ассемблирования и сборки объектного файла программы можно воспользоваться меню PROJECT, где выбрать вкладку MAKE или COMPILE. После завершения компиляции при наличии ошибок или предупреждений автоматически открывается окно MESSAGES с результатами выполнения.

2.2 Отладка проекта в режиме симулятора

Для дальнейшей работы выберем конфигурацию Debug, предусматривающую отладку в режиме симулятора. Для запуска отладчика необходимо выбрать в меню PROJECT пункт «Download and Debug», после чего автоматически переходим в режим отладчика, а курсор устанавливается на функцию «main». Интерфейс отладчика идентичен рассмотренному ранее при работе с микроконтроллерами MSP430 [1], поэтому детали здесь повторять не будем. Изучите возможности анализа хода программы (отладка всей программы в пошаговом режиме), профилировщика (View/Profiling) при оптимизации по размеру кода и быстродействию, а также анализатора кода (View/Code Coverage).

2.3 Отладка проекта с применением аппаратных средств

После отработки в режиме симулятора целесообразно посмотреть, как будет работать созданная программа на «живом» контроллере отладочной платы. Подключите модуль J-Link к разъему JTAG отладочной платы и в разъем

USB персонального компьютера, далее аккуратно соедините кабелем мини-разъем USB отладочной платы (рядом с разъемом питания слева от разъема JTAG) со вторым USB разъемом компьютера. Светодиод Power должен подтвердить наличие напряжения питания на отладочной плате. Убедитесь в свечении (мигании) светодиодов на корпусе модуля J-Link.

Переключите конфигурацию проекта в режим RAM и запустите отладчик аналогично п. 2.2, после чего запустите программу на выполнение «Debug/Go» или F5. Убедитесь в работоспособности проекта.

3 Изучение особенностей микроконтроллера

3.1 Изучение базовых функций ввода-вывода

При запуске проекта можно видеть надпись на дисплее и мигание светодиодов внизу платы. Работа с дисплеем заключается в настройке контроллера дисплея и записи кодов символов в соответствующее знакоместо. Особенности контроллера дисплея описаны в специальном файле, находящемся в подкаталоге Doc папки Samples. Для работы со строками язык Си предусматривает довольно большой набор функций, прототипы которых описаны в подключаемом заголовочном файле STDIO.H. Общие сведения о возможностях этих функций можно почерпнуть в справочных файлах в подкаталоге Doc и встроенной справочной системе среды разработки IAR Embedded Workbench IDE.

В проекте также демонстрируется работа с системой ввода/вывода микроконтроллера LPC2148 на примере включения/выключения светодиодов и опросе состояния кнопки. На отладочной плате указано, к каким выводам микроконтроллера подключены светодиоды и кнопка, кроме того, в подкаталоге Doc имеется файл со схемой отладочной платы.

3.2 Изучение модуля MAM

Модуль MAM обеспечивает ускорение работы с FLASH памятью путем предварительной выборки команд из памяти. Ознакомьтесь с деталями работы в справочном руководстве, также находящемся в подкаталоге Doc. В проекте модуль MAM включается и выключается в зависимости от нажатия кнопки, подключенной к выводу P0.14.

Выберите конфигурацию Release, загрузите код в устройство и запустите программу. Нажимая кнопку, наблюдайте изменение скорости мигания светодиодов, обусловленное включением и выключением MAM.

3.3 Изучение модуля PLL

При выключенном модуле PLL тактовая частота микроконтроллера равна частоте кварцевого резонатора, для получения максимальной производительности необходимо настроить и включить PLL. Последовательность действий здесь такова: Записать в регистр PLLCFG коэффициенты M и P, включить PLL, дождаться захвата, включить выход PLL в качестве тактового сигнала микроконтроллера.

Удалите из проекта файл «main.cpp» (щелчок правой клавишей по его имени в списке, далее выбрать «Remove» из контекстного меню) и включите файл «main_pll.cpp». Выберите конфигурацию RAM, откомпилируйте проект и

запустите программу на выполнение. Сравните частоту мигания светодиодов до и после нажатия кнопки P0.14. Выясните из кода программы, какова теперь частота тактового сигнала контроллера.

3.4 Изучение системы прерываний

Удалите из проекта файлы «main_pll.cpp» и «lcd.cpp» и добавьте файлы «main_VIC.cpp» и «lpc2xxx_startup.s». Последний файл – это стандартный файл, включающий в себя стартовый код микроконтроллера LPC2148. Файл создан на языке ассемблера и необходим нам для того, чтобы инициализировать векторы прерываний микроконтроллера. Обычной практикой считается использование в своем проекте копии этого файла и изменение его кода в зависимости от необходимости. В файле «main_VIC.cpp» описываются обработчики прерываний IRQ и FIQ. Прерывание IRQ привязывается к таймеру 0, а FIQ – к внешнему прерыванию, активизируемому кнопкой P0.14. Ознакомьтесь с деталями управления контроллером VIC и таймером по документации контроллера (файл в подкаталоге Doc).

Установите конфигурацию RAM, откомпилируйте программу и запустите на выполнение. Откройте окно вывода сообщений «Debug\Terminal I/O». Наблюдайте вывод информации, инициируемый обработчиком прерывания IRQ.

Нажмите кнопку (P0.14) и наблюдайте результат в окне терминала.

Удалите из проекта файл «main_VIC.cpp» и добавьте файл «main_VIC_UART.cpp». Аккуратно переставьте кабель питания из разъема USB в разъем UART (слева). Установите конфигурацию RAM, откомпилируйте программу и запустите на выполнение. Запустите программу RS-232, установите скорость 19200 и соответствующий COM порт. Наблюдайте вывод информации в рабочее окно программы. Перестройте скорость передачи порта на величину 9600 бод и проверьте работоспособность.

3.5 Работа с модулем SPI

Удалите из проекта файлs «main_VIC.cpp» и «lpc2xxx_startup.s», и добавьте файлы «main_SPI.cpp», «spi.cpp». Скопируйте в каталог своего проекта файл «spi.h». Установите конфигурацию RAM, откомпилируйте программу и запустите на выполнение. С помощью джойстика (в правом нижнем углу платы) управляйте положением светящегося светодиода в матрице. Измените программу так, чтобы обеспечить передвижение по диагонали. Создайте свой собственный световой эффект. Схема подключения джойстика приведена в приложении Ж (рис. Ж.1).

3.6 Работа с модулем АЦП

Удалите из проекта файлы «main_SPI.cpp», «spi.cpp» и «spi.h», «lcd.cpp» и добавьте файлы «main_ADC.cpp», «adc.cpp», «lpc2xxx_startup.s». Также скопируйте в директорию проекта файл «main_ADC.h». Замените файл «lpc2xxx_startup.s» на файл «lpc2xxx_startup_no_fiq.s», в нем заблокированы строчки, описывающие обработчик fiq_handler отказавшись от использования прерывания FIQ.

Установите конфигурацию RAM, откомпилируйте программу и запустите на выполнение. С помощью переменного резистора AIN1 регулируйте скорость

вращения пропеллера, одновременно наблюдая изменение скорости на дисплее. Анализируя код программы, установите истинную скорость вращения двигателя и получите выражение, связывающее число на дисплее со скоростью вращения. Измените программу, так, чтобы на дисплей выводилась скорость вращения.

4 Задания для самостоятельного выполнения

1. Используя имеющийся на плате динамик и модуль ЦАП микроконтроллера, создайте генератор звуковой частоты с изменением частоты по команде компьютера, принимаемой через UART.
2. Используя имеющийся на плате динамик и модуль ЦАП микроконтроллера, создайте генератор звуковой частоты с изменением частоты, управляемый потенциометром AIN2.
3. Создайте программу, записывающую блок данных объемом 100 байт во внешнюю EEPROM через интерфейс I^2C.
4. Создайте программу, изображающую ползущую змею на матричном индикаторе (направление по собственному выбору, длина 6 пикселей).
5. Создайте программу, отображающую текущую температуру (используя датчик на плате) на LCD дисплее.

5 Список использованных источников

1. Разработка программного обеспечния микроконтроллеров семейства MSP430: Методические указания для выполнения лабораторной работы / Самарский гос. аэрокосмический ун-т. Сост. И.А. Кудрявцев, Д.В. Корнилин. Самара 2010 20стр.
2. Тревор Мартин Микроконтроллеры ARM7. Семейство LPC2000 компании Philips. Вводный курс. – М.:Издательский дом «Додэка-XXI», 2006. – 240с.
3. UM10139 Volume 1: LPC214X User Manual Rev.01 – 15 August 2005.
4. ARM7 TDMI Rev.3 Technical Reference Manual

Приложение А

Таблица А.1 Регистры управления портами ввода-вывода

Наименование	Описание	Доступ	Состояние при сбросе
IOXPIN	Предназначен для считывания состояния вывода	R/W	-
IOXSET	Запись единицы приводит к установке высокого логического уровня на выводе	R/W	0x00000000
IOXCLR	Запись единицы приводит к установке низкого логического уровня на выводе	R/W	0x00000000
IOXDIR	Направление передачи. Запись единицы конфигурирует вывод в режим вывода информации	R/W	0x00000000

Приложение Б

Таблица Б.1 Регистры UART0

Name	Description	Bit functions and addresses								Access	Reset value[1]	Address
		MSB							LSB			
		BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0			
U0RBR	Receiver Buffer Register	8-bit Read Data								RO	NA	0xE000 C000 (DLAB=0)
U0THR	Transmit Holding Register	8-bit Write Data								WO	NA	0xE000 C000 (DLAB=0)
U0DLL	Divisor Latch LSB	8-bit Data								R/W	0x01	0xE000 C000 (DLAB=1)
U0DLM	Divisor Latch MSB	8-bit Data								R/W	0x00	0xE000 C004 (DLAB=1)
U0IER	Interrupt Enable Register	-	-	-	-	-	-	En.ABTO	En.ABEO	R/W	0x00	0xE000 C004 (DLAB=0)
		-	-	-	-	-	En.RX Lin.St.Int	Enable THRE Int	En.RX Dat.Av.Int			
U0IIR	Interrupt ID Reg.	-	-	-	-	-	-	ABTO Int	ABEO Int	RO	0x01	0xE000 C008
		FIFOs Enabled		-	-	IIR3	IIR2	IIR1	IIR0			
U0FCR	FIFO Control Register	RX Trigger		-	-	-	TX FIFO Reset	RX FIFO Reset	FIFO Enable	WO	0x00	0xE000 C008
U0LCR	Line Control Register	DLAB	Set Break	Stick Parity	Even Par.Selct.	Parity Enable	No. of Stop Bits	Word Length Select		R/W	0x00	0xE000 C00C
U0LSR	Line Status Register	RX FIFO Error	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60	0xE000 C014
U0SCR	Scratch Pad Reg.	8-bit Data								R/W	0x00	0xE000 C01C
U0ACR	Auto-baud Control Register	-	-	-	-	-	-	ABTO Int.Clr	ABEO Int.Clr	R/W	0x00	0xE000 C020
		-	-	-	-	-	Aut.Rstrt.	Mode	Start			
U0FDR	Fractional Divider Register	Reserved[31:8]									0x10	0xE000 C028
		MulVal				DivAddVal						
U0TER	TX. Enable Reg.	TXEN	-	-	-	-	-	-	-	R/W	0x80	0xE000 C030

Формула для расчета скорости передачи

$$UART0_{baudrate} = \frac{PCLK}{16 \times (16 \times U0DLM + U0DLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

Основные регистры:
U0RSR – регистр принятых данных;
U0THR – регистр данных для передачи;
U0IER – регистр разрешения прерываний UART:
Бит 0 - 1- разрешить прерывание при наличии принятых данных;
Бит 1 - 1 - разрешить прерывание при опустошении буфера передачи;
Бит 2 – 1 – разрешить прерывание при определенном состоянии линии RX
U0LCR – регистр управления состоянием линии: Конфигурирует формат посылки [3];
U0LSR – регистр состояния линии: Текущее состояние порта (ошибки)

Приложение В

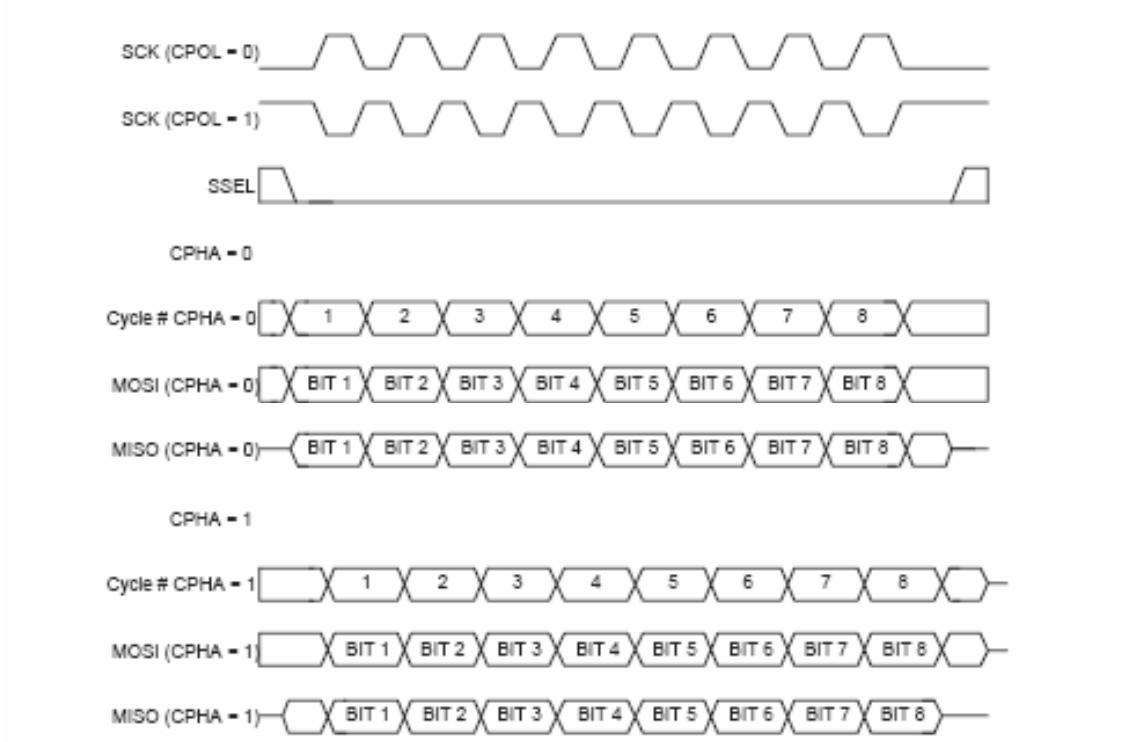


Рисунок В.1 – Форматы передачи в модуле SPI

Таблица В.1 Регистры модуля SPI

Name	Description	Access	Reset value[1]	Address
S0SPCR	SPI Control Register. This register controls the operation of the SPI.	R/W	0x00	0xE002 0000
S0SPSR	SPI Status Register. This register shows the status of the SPI.	RO	0x00	0xE002 0004
S0SPDR	SPI Data Register. This bi-directional register provides the transmit and receive data for the SPI. Transmit data is provided to the SPI0 by writing to this register. Data received by the SPI0 can be read from this register.	R/W	0x00	0xE002 0008
S0SPCCR	SPI Clock Counter Register. This register controls the frequency of a master's SCK0.	R/W	0x00	0xE002 000C
S0SPINT	SPI Interrupt Flag. This register contains the interrupt flag for the SPI interface.	R/W	0x00	0xE002 001C

Важнейшие регистры:
S0SPCR – управляющий регистр, подробный формат см. в [3];
S0SPSR – регистр статуса, отражает текущее состояние (ошибки);
S0SPDR – регистр, содержащий передаваемые и принимаемые данные;
S0SPCCR –регистр управления частотой передачи в режиме MASTER.

Приложение Г

Таблица Г.1 Регистры таймеров

Generic Name	Description	Access	Reset value[1]	TIMER/ COUNTER0 Address & Name	TIMER/ COUNTER1 Address & Name
IR	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	R/W	0	0xE000 4000 T0IR	0xE000 8000 T1IR
TCR	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	R/W	0	0xE000 4004 T0TCR	0xE000 8004 T1TCR
TC	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	R/W	0	0xE000 4008 T0TC	0xE000 8008 T1TC
PR	Prescale Register. The Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	R/W	0	0xE000 400C T0PR	0xE000 800C T1PR
PC	Prescale Counter. The 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	R/W	0	0xE000 4010 T0PC	0xE000 8010 T1PC
MCR	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	R/W	0	0xE0004014 T0MCR	0xE000 8014 T1MCR
MR0	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	R/W	0	0xE000 4018 T0MR0	0xE000 8018 T1MR0
MR1	Match Register 1. See MR0 description.	R/W	0	0xE000 401C T0MR1	0xE000 801C T1MR1
MR2	Match Register 2. See MR0 description.	R/W	0	0xE000 4020 T0MR2	0xE000 8020 T1MR2
MR3	Match Register 3. See MR0 description.	R/W	0	0xE000 4024 T0MR3	0xE000 8024 T1MR3
CCR	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	R/W	0	0xE000 4028 T0CCR	0xE000 8028 T1CCR
CR0	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CAPn.0(CAP0.0 or CAP1.0 respectively) input.	RO	0	0xE000 402C T0CR0	0xE000 802C T1CR0
CR1	Capture Register 1. See CR0 description.	RO	0	0xE000 4030 T0CR1	0xE000 8030 T1CR1
CR2	Capture Register 2. See CR0 description.	RO	0	0xE000 4034 T0CR2	0xE000 8034 T1CR2
CR3	Capture Register 3. See CR0 description.	RO	0	0xE000 4038 T0CR3	0xE000 8038 T1CR3
EMR	External Match Register. The EMR controls the external match pins MATn.0-3 (MAT0.0-3 and MAT1.0-3 respectively).	R/W	0	0xE000 403C T0EMR	0xE000 803C T1EMR
CTCR	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	R/W	0	0xE000 4070 T0CTCR	0xE000 8070 T1CTCR

Основные регистры: (X – 0 или 1 в зависимости от таймера)
TxTCR – управляющий регистр;
TxTC – регистр счетчика;
TxPR – регистр предделителя;
TxMR0 – регистр, содержащий величину, при совпадении с которой генерируется прерывание и счетчик сбрасывается;
TxMCR – регистр управления режимом совпадения;
TxIR – регистр управления прерываниями таймера.

Приложение Д

Таблица Д.1 Регистры АЦП

Generic Name	Description	Access	Reset value[1]	AD0 Address & Name	AD1 Address & Name
ADCR	A/D Control Register. The ADCR register must be written to select the operating mode before A/D conversion can occur.	R/W	0x0000 0001	0xE003 4000 AD0CR	0xE006 0000 AD1CR
ADGDR	A/D Global Data Register. This register contains the ADC's DONE bit and the result of the most recent A/D conversion.	R/W	NA	0xE003 4004 AD0GDR	0xE006 0004 AD1GDR
ADSTAT	A/D Status Register. This register contains DONE and OVERRUN flags for all of the A/D channels, as well as the A/D interrupt flag.	RO	0x0000 0000	0xE003 4030 AD0STAT	0xE006 0030 AD1STAT
ADGSR	A/D Global Start Register. This address can be written (in the AD0 address range) to start conversions in both A/D converters simultaneously.	WO	0x00	0xE003 4008 ADGSR	
ADINTEN	A/D Interrupt Enable Register. This register contains enable bits that allow the DONE flag of each A/D channel to be included or excluded from contributing to the generation of an A/D interrupt.	R/W	0x0000 0100	0xE003 400C AD0INTEN	0xE006 000C AD1INTEN
ADDR0	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 0.	RO	NA	0xE003 4010 AD0DR0	0xE006 0010 AD1DR0
ADDR1	A/D Channel 1 Data Register. This register contains the result of the most recent conversion completed on channel 1.	RO	NA	0xE003 4014 AD0DR1	0xE006 0014 AD1DR1
ADDR2	A/D Channel 2 Data Register. This register contains the result of the most recent conversion completed on channel 2.	RO	NA	0xE003 4018 AD0DR2	0xE006 0018 AD1DR2
ADDR3	A/D Channel 3 Data Register. This register contains the result of the most recent conversion completed on channel 3.	RO	NA	0xE003 401C AD0DR3	0xE006 001C AD1DR3
ADDR4	A/D Channel 4 Data Register. This register contains the result of the most recent conversion completed on channel 4.	RO	NA	0xE003 4020 AD0DR4	0xE006 0020 AD1DR4
ADDR5	A/D Channel 5 Data Register. This register contains the result of the most recent conversion completed on channel 5.	RO	NA	0xE003 4024 AD0DR5	0xE006 0024 AD1DR5
ADDR6	A/D Channel 6 Data Register. This register contains the result of the most recent conversion completed on channel 6.	RO	NA	0xE003 4028 AD0DR6	0xE006 0028 AD1DR6
ADDR7	A/D Channel 7 Data Register. This register contains the result of the most recent conversion completed on channel 7.	RO	NA	0xE003 402C AD0DR7	0xE006 002C AD1DR7

Основные регистры:
ADCR – управляющий регистр;
ADGDR – регистр, содержащий последний результат и бит готовности;
ADSTAT – регистр статуса АЦП (всех каналов);
ADDRX – регистр результата соответствующего канала.

Таблица Д.2 Формат регистра DACR (0xE006C000) управления ЦАП

Bit	Symbol	Value	Description	Reset value
5:0	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:6	VALUE		After the selected settling time after this field is written with a new VALUE, the voltage on the A_{OUT} pin (with respect to V_{SSA}) is VALUE/1024 * V_{REF}.	0
16	BIAS	0	The settling time of the DAC is 1 µs max, and the maximum current is 700 uA.	0
		1	The settling time of the DAC is 2.5 µs and the maximum current is 350 µA.	
31:17	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Для активизации ЦАП необходимо установить биты 19:18 регистра PINSEL1 в состояние «10».

Приложение Е

Таблица Е.1 Регистры контроллера прерываний

Name	Description	Access	Reset value[1]	Address
VICIRQStatus	IRQ Status Register. This register reads out the state of those interrupt requests that are enabled and classified as IRQ.	RO	0	0xFFFF F000
VICFIQStatus	FIQ Status Requests. This register reads out the state of those interrupt requests that are enabled and classified as FIQ.	RO	0	0xFFFF F004
VICRawIntr	Raw Interrupt Status Register. This register reads out the state of the 32 interrupt requests / software interrupts, regardless of enabling or classification.	RO	0	0xFFFF F008
VICIntSelect	Interrupt Select Register. This register classifies each of the 32 interrupt requests as contributing to FIQ or IRQ.	R/W	0	0xFFFF F00C
VICIntEnable	Interrupt Enable Register. This register controls which of the 32 interrupt requests and software interrupts are enabled to contribute to FIQ or IRQ.	R/W	0	0xFFFF F010
VICIntEnClr	Interrupt Enable Clear Register. This register allows software to clear one or more bits in the Interrupt Enable register.	WO	0	0xFFFF F014
VICSoftInt	Software Interrupt Register. The contents of this register are ORed with the 32 interrupt requests from various peripheral functions.	R/W	0	0xFFFF F018
VICSoftIntClear	Software Interrupt Clear Register. This register allows software to clear one or more bits in the Software Interrupt register.	WO	0	0xFFFF F01C
VICProtection	Protection enable register. This register allows limiting access to the VIC registers by software running in privileged mode.	R/W	0	0xFFFF F020
VICVectAddr	Vector Address Register. When an IRQ interrupt occurs, the IRQ service routine can read this register and jump to the value read.	R/W	0	0xFFFF F030
VICDefVectAddr	Default Vector Address Register. This register holds the address of the Interrupt Service routine (ISR) for non-vectored IRQs.	R/W	0	0xFFFF F034

Основные регистры:
VicSoftInt – регистр биты в котором соответствуют имеющимся на данный момент запросам прерываний;
VicSoftIntClear – регистр сброса VicSoftInt;
VicIntEnable – регистр индивидуального разрешения/запрета прерываний;
VicIntEnClear – регистр сброса VicIntEnable;
VicIntSelect – регистр выбора привязки (IRQ или FIQ) для каждого прерывания;
VicVectCntl0-15 – регистры слотов IRQ (содержат бит разрешения и номер слота для каждого из 16 векторных прерываний IRQ);
VicVectAddr0-15 – регистры адресов векторов IRQ;
VicVectDefAddr – регистр адреса обработчика невекторного IRQ;
VicVectAddr – регистр, содержащий адрес обрабатываемого IRQ;
VicProtection – регистр разрешения доступа к регистрам VIC.

Таблица E.2 Источники прерываний

Block	Flag(s)	VIC Channel # and Hex Mask	
WDT	Watchdog Interrupt (WDINT)	0	0x0000 0001
-	Reserved for Software Interrupts only	1	0x0000 0002
ARM Core	Embedded ICE, DbgCommRx	2	0x0000 0004
ARM Core	Embedded ICE, DbgCommTX	3	0x0000 0008
TIMER0	Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3)	4	0x0000 0010
TIMER1	Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3)	5	0x0000 0020
UART0	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI)	6	0x0000 0040
UART1	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) Modem Status Interrupt (MSI)[1]	7	0x0000 0080
PWM0	Match 0 - 6 (MR0, MR1, MR2, MR3, MR4, MR5, MR6)	8	0x0000 0100
I^2C0	SI (state change)	9	0x0000 0200
SPI0	SPI Interrupt Flag (SPIF) Mode Fault (MODF)	10	0x0000 0400
SPI1 (SSP)	TX FIFO at least half empty (TXRIS) Rx FIFO at least half full (RXRIS) Receive Timeout condition (RTRIS) Receive overrun (RORRIS)	11	0x0000 0800
PLL	PLL Lock (PLOCK)	12	0x0000 1000
RTC	Counter Increment (RTCCIF) Alarm (RTCALF)	13	0x0000 2000
System Control	External Interrupt 0 (EINT0)	14	0x0000 4000
	External Interrupt 1 (EINT1)	15	0x0000 8000
	External Interrupt 2 (EINT2)	16	0x0001 0000
	External Interrupt 3 (EINT3)	17	0x0002 0000
ADC0	A/D Converter 0 end of conversion	18	0x0004 0000
I^2C1	SI (state change)	19	0x0008 0000

Приложение Ж

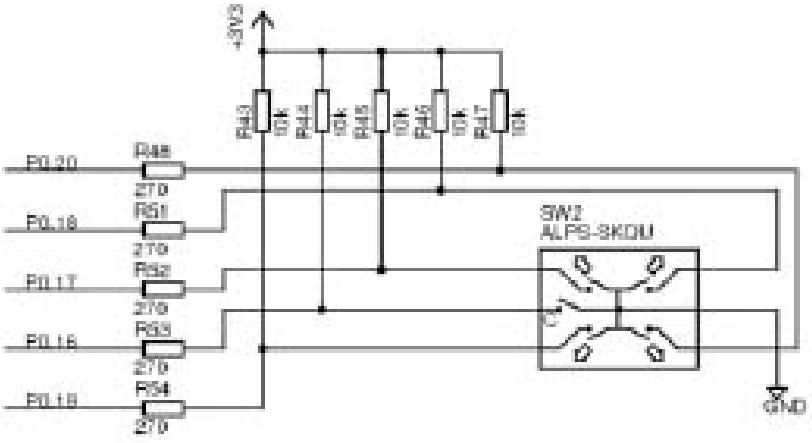


Рисунок Ж.1 – Схема подключения джойстика на отладочной плате

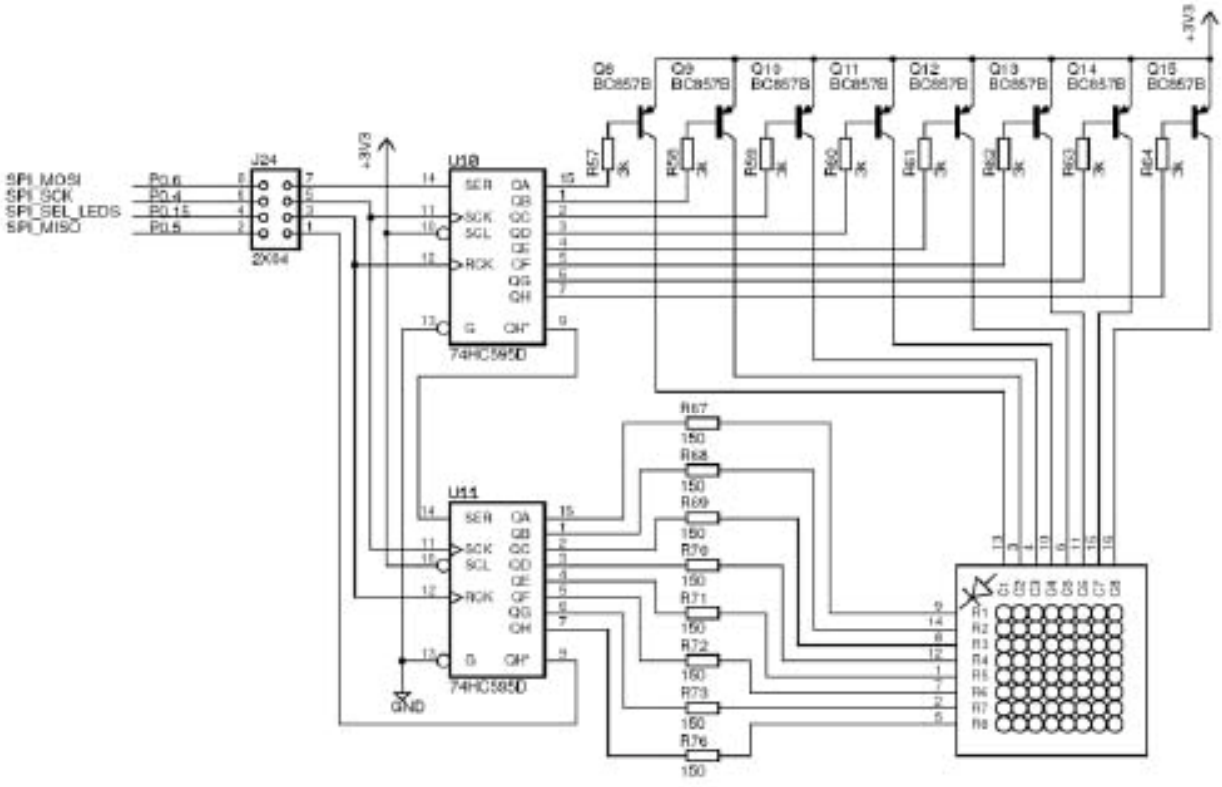


Рисунок Ж.2 – Схема подключения матричного индикатора на отладочной плате

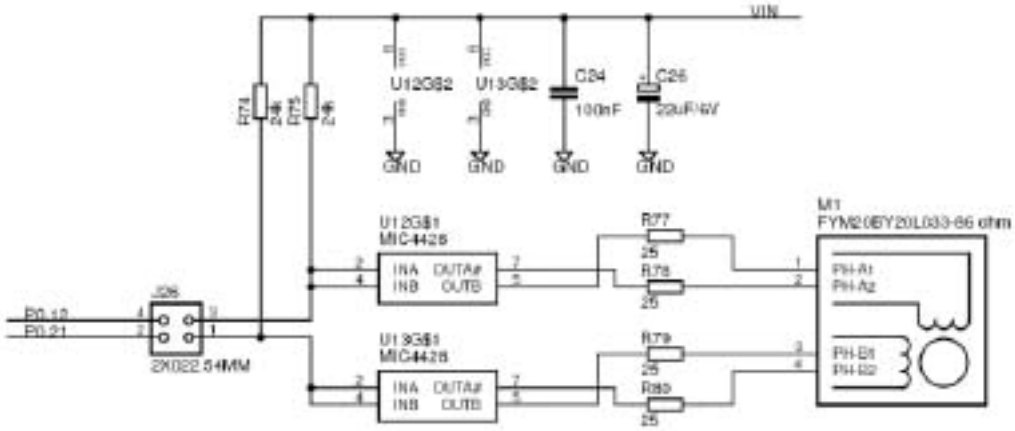


Рисунок Ж.3 – Схема подключения шагового двигателя на отладочной плате

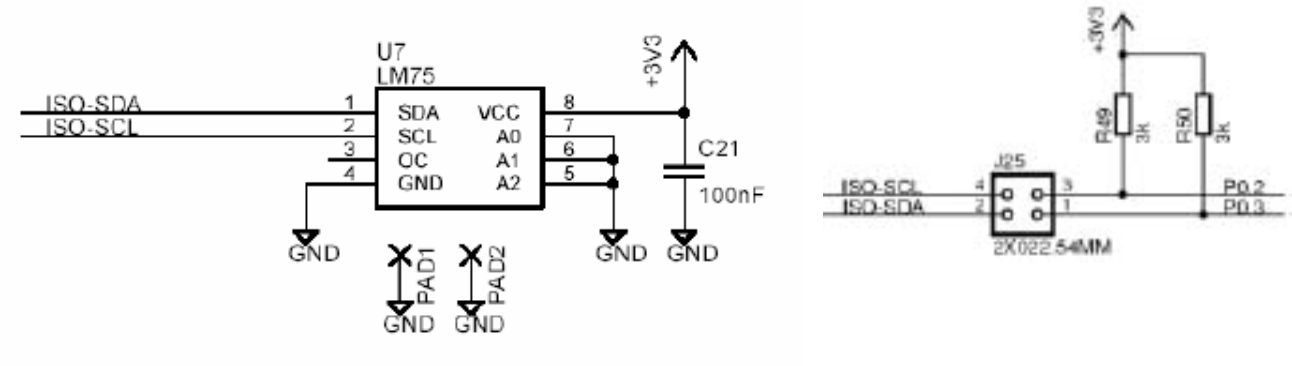


Рисунок Ж.4 – Схема подключения датчика температуры на отладочной плате

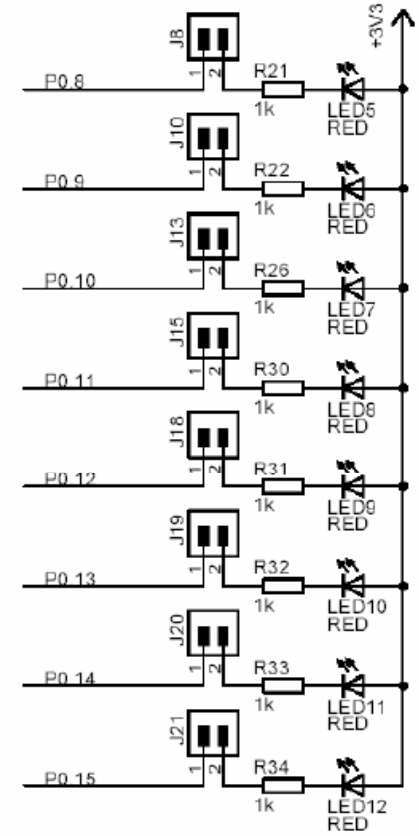


Рисунок Ж.5 – Подключение светодиодов на отладочной плате

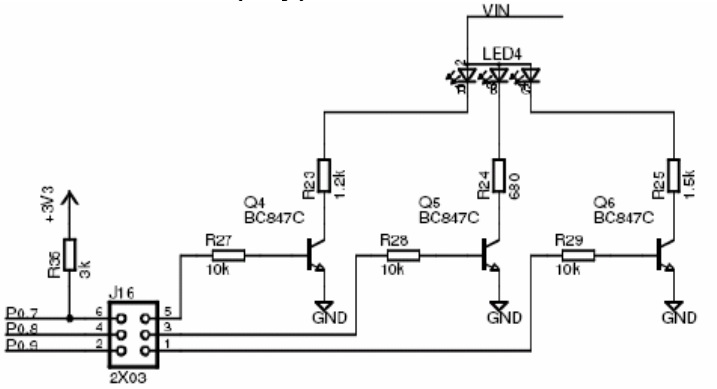


Рисунок Ж.6 – Подключение трехцветного индикатора на отладочной плате

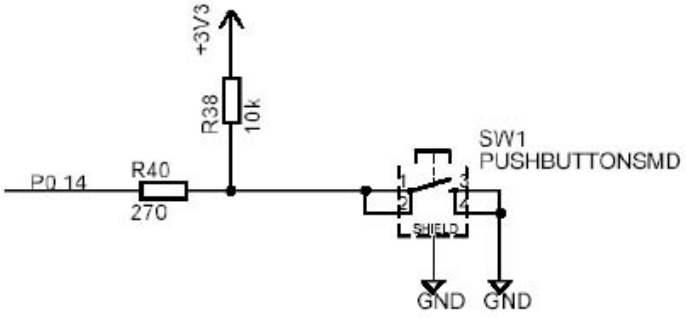


Рисунок Ж.7 – Подключение кнопки на отладочной плате

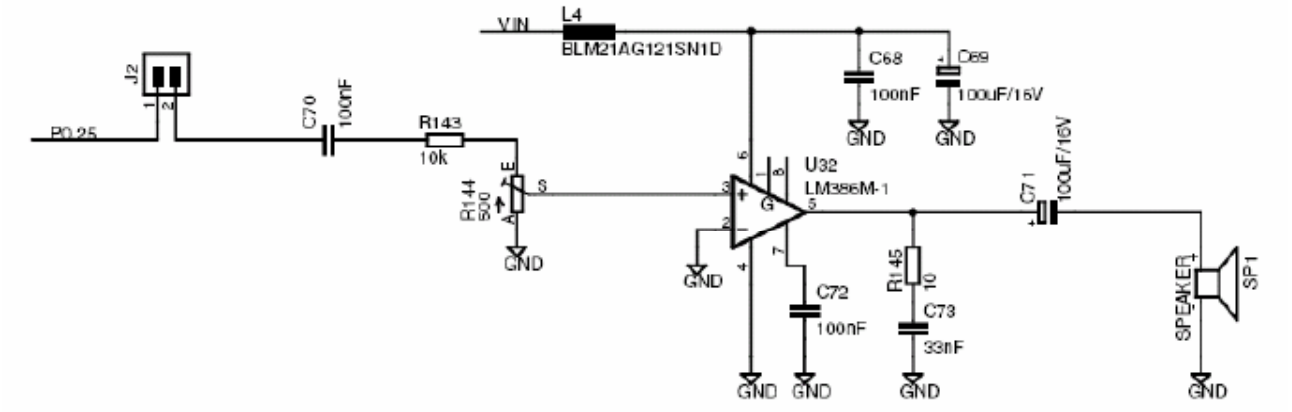


Рисунок Ж.8 – Подключение динамика на отладочной плате

Лабораторная работа №2 «Разработка программного обеспечения цифровых сигнальных процессоров»

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цифровые сигнальные процессоры Blackfin фирмы Analog Devices являются одними из самых производительных в своем классе. Высокая тактовая частота (до 750МГц), возможность выполнения параллельных операций, мощная поддержка ПДП обеспечивают высокую эффективность применения данных процессоров для реализации скоростных алгоритмов обработки сигналов в режиме реального времени. Следует отметить их высокую популярность для разработки приложений универсального характера, сочетающих реализацию алгоритмов управления и алгоритмов цифровой обработки сигналов в реальном времени, необходимых для обработки и сжатия изображений, решения задач коммуникационного характера и т.п.

Шестнадцати и тридцатидвухразрядные операции, развитая система команд делают эффективным применение С-компиляторов для разработки программного обеспечения, что также сокращает затраты времени программиста на реализацию целого ряда рутинных операций. Вместе с тем для создания особенно критичных участков кода и эффективного использования архитектурных возможностей процессора программист должен владеть системой команд на уровне языка ассемблера.

При разработке сложных алгоритмов программист вынужден затрачивать значительные усилия на реализацию рутинных операций, например, организацию операций ввода-вывода, манипуляции с памятью, обработку прерываний и т.п., что требует длительного времени. Между тем, современные процессоры обработки сигналов обладают возможностью сократить «посторонние» усилия разработчика за счет использования операционной системы, автоматически решающей указанные задачи, и позволяющей сосредоточиться на разработке главного алгоритма, реализация которого составляет смысл проекта. Применительно к процессорам Blackfin такой операционной системой является VDK (Visual DSP Kernel), тесно интегрированная в одноименную среду разработки.

Процессор обладает целым рядом встроенных средств, облегчающих отладку программного обеспечения, в том числе режим эмуляции на базе интерфейса JTAG, что позволяет сократить сроки разработки конечного устройства при условии грамотного использования разработчиком имеющихся средств САПР.

Для создания окончательного варианта устройства на базе DSP Blackfin от разработчика требуются довольно значительные усилия по разработке схемотехнической и конструкторской частей проекта, причем стоимость и сроки разработки конструкции, включая печатную плату, могут быть весьма значительными. Сократить указанные затраты, распараллелить работу над проектом, а также снизить риск неоправданных потерь времени и средств на переработку проекта может использование отладочной платы типа EZ-LITE, на которой установлен процессор и наиболее популярные периферийные устройства. При использовании подобных средств «макетную часть» проекта удается провести без длительной и дорогостоящей стадии разработки (и изготовления) топологии печатной платы.

Методические указания позволяют студентам изучить основы программирования процессоров Blackfin в среде разработки Visual DSP++ с использо-

ванием такой отладочной платы и операционной системы VDK. Указания не претендуют на полноту описания особенностей процессора, приводятся лишь краткие пояснения, необходимые для понимания приведенных фрагментов.

1 Создание проекта

Среда программирования Visual DSP предусматривает несколько режимов работы, включая работу с аппаратными отладчиками и симуляцию проекта. При запуске среды программирования обычно загружается последний проект в том режиме, с которым велась работа. При появлении диалогового окна, предлагающего выбрать режим работы, необходимо выбрать режим симуляции и тип процессора – ADSP-BF533. Впоследствии изменить режим работы достаточно легко с помощью переключения сессии (меню «Session»).

Для создания нового проекта целесообразно воспользоваться предлагаемым мастером создания проекта «Project wizard», вызвать который можно через меню «File/New/Project». В появившемся диалоговом окне укажите имя проекта (латинскими символами), путь к директории проекта и его тип. В качестве типа укажите «Standard Application». Попутно отметим, что можно также выбрать «Multi-threaded application using VDK», что понадобится нам в дальнейшем.

Далее щелкните левой кнопкой мыши по значку «Output type» и в появившемся окне выберите процессор ADSP BF533, в окне «Project output type» нужно указать «Executable file».

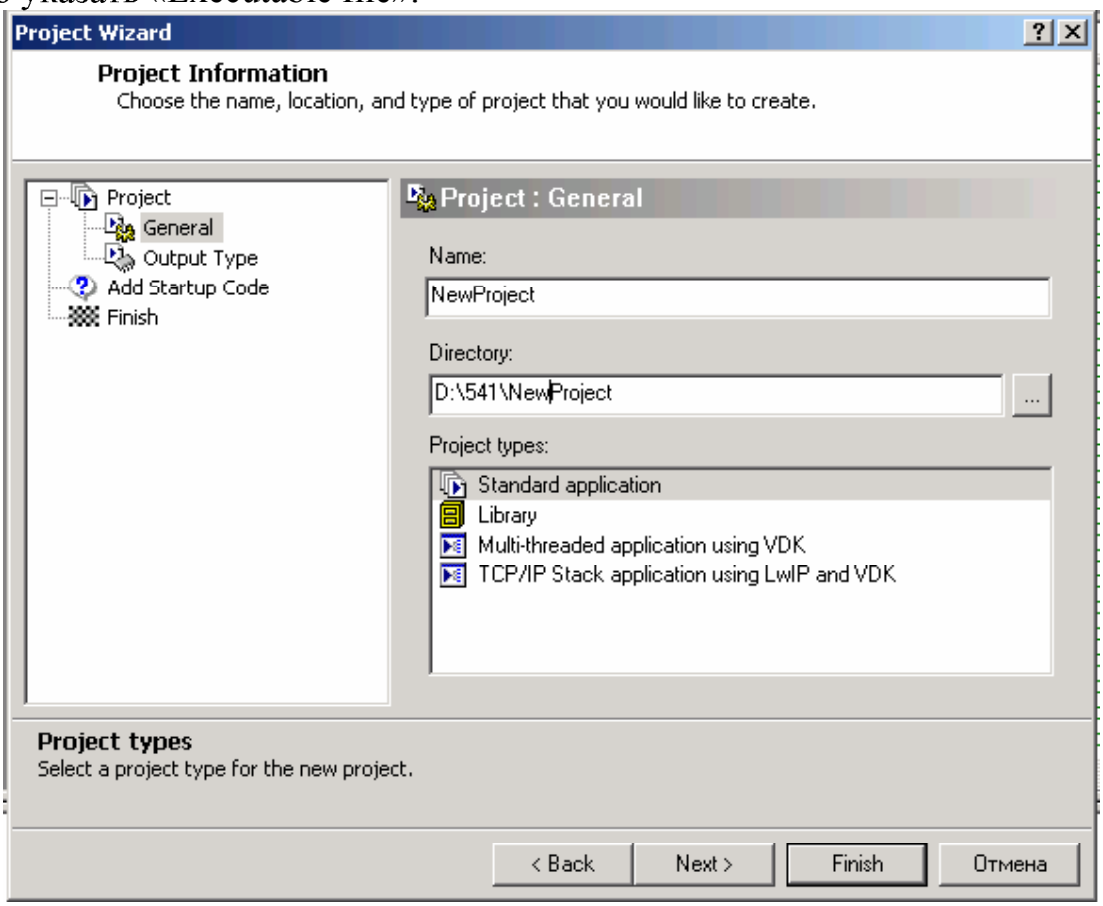


Рисунок 1 – Окно мастера создания проекта

После этого щелкните левой кнопкой мыши по значку «Add startup code» и далее откажитесь от внедрения в проект типового стартового кода (выбор «No»). По завершению всех этих операций нажмите кнопку «Finish».

В левой части окна появилось окно, в котором отображается структура проекта, но пока ничего нет, так как мы еще не включили в состав проекта никаких файлов с исходным кодом.

1.1 Описание исследуемой программы

Программа представляет собой реализацию алгоритма несложного цифрового фильтра с конечной импульсной характеристикой (КИХ). Смысл обработки данных в таком фильтре (на примере фильтра 9-го порядка) представлен на рисунке 2. Здесь z^{-1} означает задержку сигнала на 1 такт, $x(n)$ - отсчеты входного сигнала, $y(n)$ - выходного. $h(n)$ - отсчеты импульсной характеристики фильтра, или его коэффициенты.

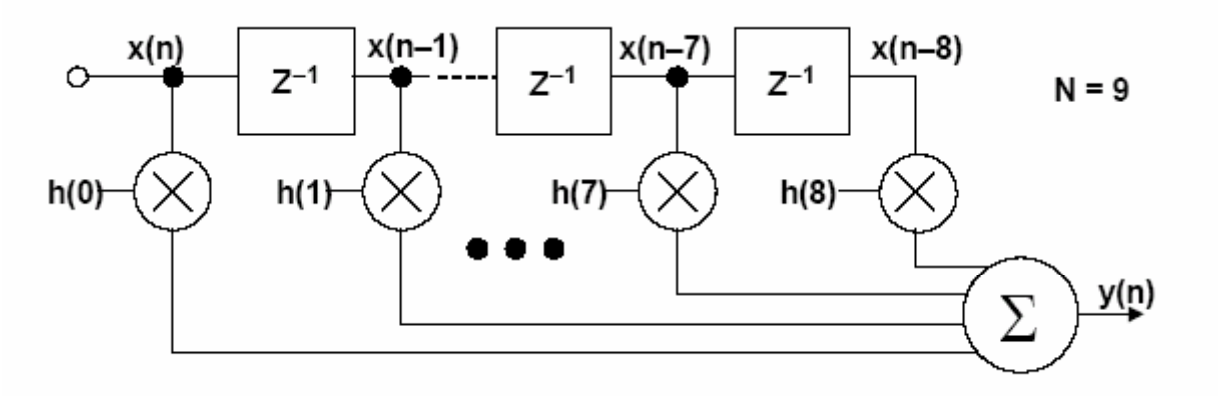


Рисунок 2 – Структурная схема КИХ фильтра

Не вдаваясь в подробности алгоритма, отметим, что фильтры с КИХ нашли довольно широкое распространение в технике цифровой обработки сигналов и их параметры определяются набором коэффициентов. Коэффициенты нашего фильтра (h(0), h(1)…) описываются в файле «fir_coeff.h».

Процедура обработки сигнала реализована на языке ассемблера и находится в файле «fir.asm». Программа, с помощью которой тестируется изучаемый фильтр, написана на языке Си и находится в файле «fir_test.c», который сопровождается заголовочным файлом «filter.h». Массив входных данных, представляющий собой набор чисел типа «short» и имитирующий входной сигнал, содержится в файле «fir_input.h».

Программа на языке ассемблера вызывается из основной программы на языке Си и принимает аргументы с помощью стандартных соглашений, подробно описываемых в файле помощи по работе с компилятором C/C++. Данные, обрабатываемые программой и коэффициенты описаны как «short», что естественно для реализации на базе 16-разрядного процессора.

Отметим также использование структуры, что характерно, в основном, для программ на языке Си, однако может применяться и в программах на ассемблере. Структура представляет собой фрагмент памяти, состоящий из отдельных полей, с точки зрения языка Си, структура является объектом, инкапсулирующим некоторый набор данных, связанных замыслом программиста. К отдельным полям структуры можно обращаться, указывая имя структуры и имя поля, разделенные точкой или символом «->». Подробнее с возможностями работы со структурами можно ознакомиться в руководстве по компилятору. В нашем случае описание структуры и макроса, инициализирующего ее, содержится в файле «filter.h».

В файле на языке ассемблера, помимо самого текста, содержатся некоторые директивы:

.section program - указывает, что данный фрагмент должен размещаться в памяти программ;

.global _fir– указывает на глобальную видимость идентификатора _fir, что необходимо для работы линковщика;

Все эти файлы находятся на жестком диске компьютера (путь указывается преподавателем). Включите файлы fir_test.c и fir.asm в состав проекта, для чего щелкните правой кнопкой мыши по имени проекта в окне проекта и из контекстного меню выберите пункт «Add File(s) to project».

2 Компиляция и отладка программы

Для компиляции проекта выберите из меню команду «Project/Build project». Результаты компиляции будут отображены в окне «Output». После успешного завершения компиляции среда разработки автоматически перейдет в режим отладки, точка останова будет установлена на первый выполняемый оператор программы. Справа появится окно дизассемблера, в котором отображаются инструкции ассемблера в том виде, как они выполняются процессором.

Отладчик предусматривает режим пошагового выполнения программы, запуска, работу с точками останова, имитацию прерываний и т.п. Все эти особенности аналогичны стандартным возможностям отладчиков, изученных ранее.

2.1 Графическое представление результатов выполнения программы

Объектом операций исследуемой программы является файл данных, представляющий собой массив значений отсчетов входного сигнала фильтра ($x(n)$). Результатом выполнения программы является массив выходных значений сигнала $y(n)$. Среда разработки Visual DSP позволяет просмотреть графическое представление этих данных во временной и частотной области, что позволяет разработчику наглядно оценить результаты своей работы.

В меню «View» выберите пункт «Debug Windows/Plot/New», после чего появится окно выбора графических данных.

Введите название («Title») и укажите имя набора «In», далее с помощью кнопки «Browse» выберите объект «IN». В окне «Count» введите значение 128 и укажите тип данных «short», после чего нажмите «Add». В окне «Data Sets» должно появиться имя набора данных. Повторите эту же процедуру для набора «Out», после чего окно должно принять вид, представленный на рисунке 3.

После нажатия кнопки «ОК» появится окно с изображением содержимого обоих массивов. Если программа еще не запускалась, массив «Out» содержит нулевые значения. Запустите программу (нажатием F5) и в окне отобразится новое содержимое массива «Out» (рисунок 4). Смысл полученного результата очевиден: богатый гармониками входной сигнал после фильтрации представляет собой практически единственную гармонику.

С помощью курсора можно выделить фрагмент изображения для детального исследования, с помощью кнопок-стрелок на клавиатуре можно просмат-

ривать значения сигналов по одной выборке. Для этого необходимо выбрать в контекстном меню пункт «Data cursor».

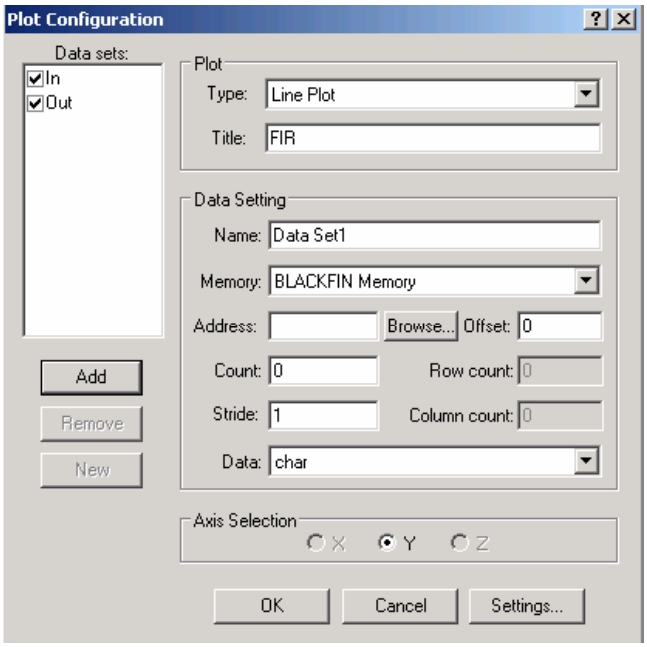


Рисунок 3 – Окно выбора графических данных

Можно перейти и в частотную область, где можно видеть спектральную диаграмму сигналов. Для этого в контекстном меню выберите пункт «Modify Settings» и вкладку «Data processing». Далее выделите имя набора в окне «Data Sets» и в окне «Data Process» укажите «FFT magnitude». В окне «Sample Rate» введите значение 10000 и нажмите «ОК».

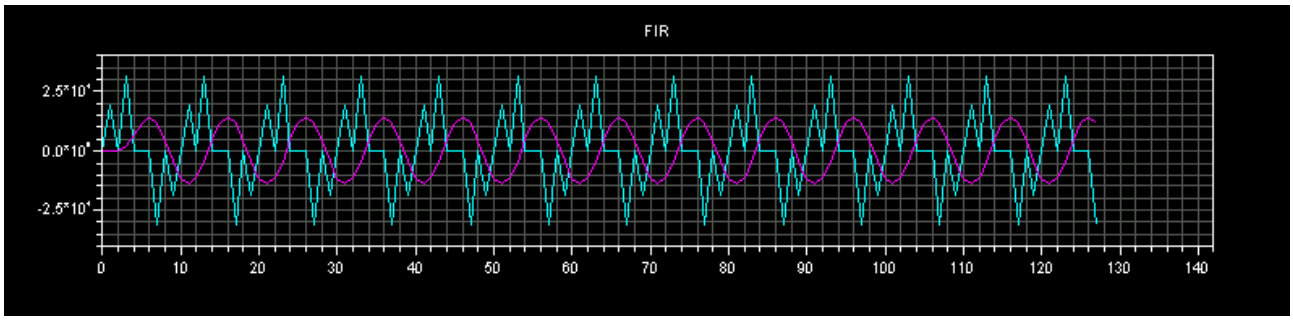


Рисунок 4 – Графическая интерпретация входных и выходных сигналов фильтра

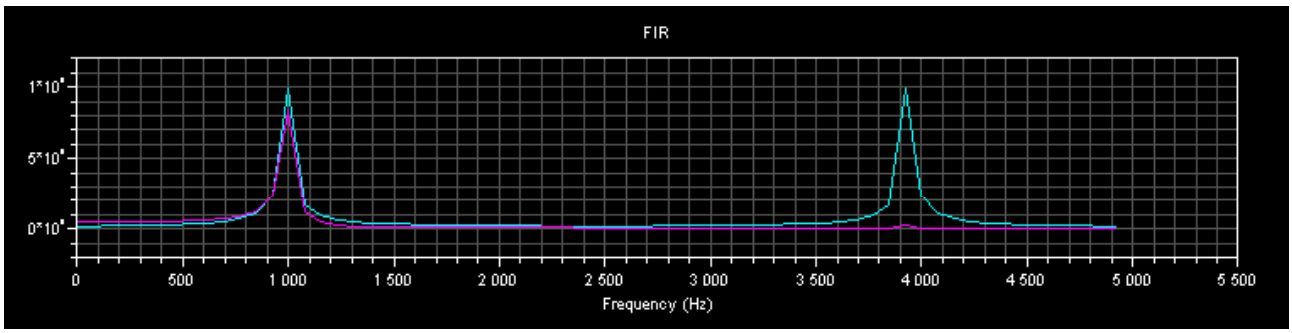


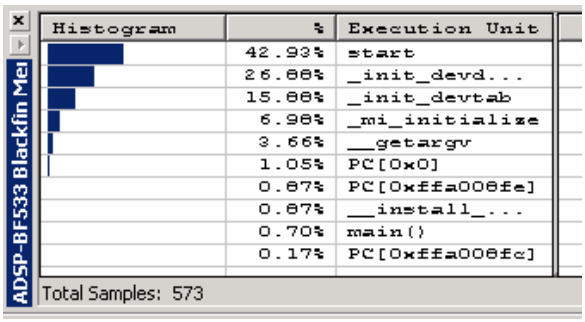
Рисунок 5 – Спектральная диаграмма сигналов на входе и выходе фильтра

Обратите внимание на задержку выходного сигнала во временной области и подавление высших гармоник сигнала на спектральной диаграмме.

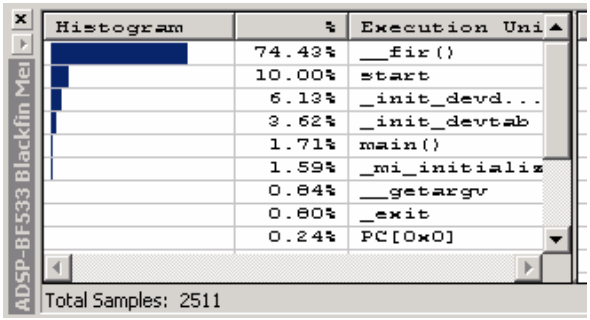
Можно также просмотреть АЧХ фильтра, если в качестве набора данных выбрать массив коэффициентов фильтра h (длиной 8) и режим отображения в частотной области.

2.2 Анализ программы

Выберите в меню пункт «Tools/Linear profiling/New» и откройте новое окно результатов профилировки. Далее перезагрузим программу «File/Load program» (необходимо указать имя исполняемого модуля с расширением «dxe»). Обратите внимание, что хотя программа еще не начала выполняться, в окне профилировки уже есть какие-то результаты. В данном случае это код, автоматически «приписанный» компилятором и инициализирующий процессор в рамках стандартной процедуры, свойственной Си-программам.



До запуска программы



После запуска

Рисунок 6 – Окно профилировщика

Очистим окно профилировки командой «Clear profile» из контекстного меню и запустим программу на выполнение нажатием кнопки F5. Гистограмма и процентные значения затраченного времени покажут эффективность созданного кода.

3 Контрольные вопросы

1) Поясните основные требования к программе на языке ассемблера, вызываемой из Си-программы. Как передаются аргументы функции fir ?
2) Зачем в программе на языке ассемблера используются команды nop ? Попробуйте убрать их из исходной программы.
3) Что содержится в регистре R1 в начале выполнения программы fir ?
4) По какому адресу находится точка входа в создаваемую программу? Каковы адреса точек входа в функции main и fir ?
5) Где размещены исходные данные (массивы IN и OUT) в памяти процессора ?
6) Выясните, какой код предшествует выполняемой программе ? Определите его основные функции.
7) Как узнать точное время выполнения программы fir ?
8) Какие ограничения накладываются на инструкции выполняемые параллельно ? Сравните инструкции в исходном файле с дизассемблированным кодом.
9) Замените в исходном тексте на языке ассемблера параллельные инструкции на отдельные (закомментируйте строку с параллельной инструкцией и раскомментируйте соответствующие строки ниже). Исследуйте, как из-

менилось поведение программы, результаты профилировки, время выполнения.

10) Поясните принципы организации циклов с нулевыми непроизводительными затратами. Определите длительность выполнения процессором циклов, входящих в состав исследуемой программы.
11) Как использовать циклическую адресацию в процессорах Blackfin ?
12) Что такое исключения, чем они вызываются и как обрабатываются ?
13) Поясните основные принципы обслуживания прерываний в DSP Blackfin

4 Приложение

Структура процессора Blackfin

Ядро и периферийные устройства, а также взаимодействие между ними представлены на рисунке 7.

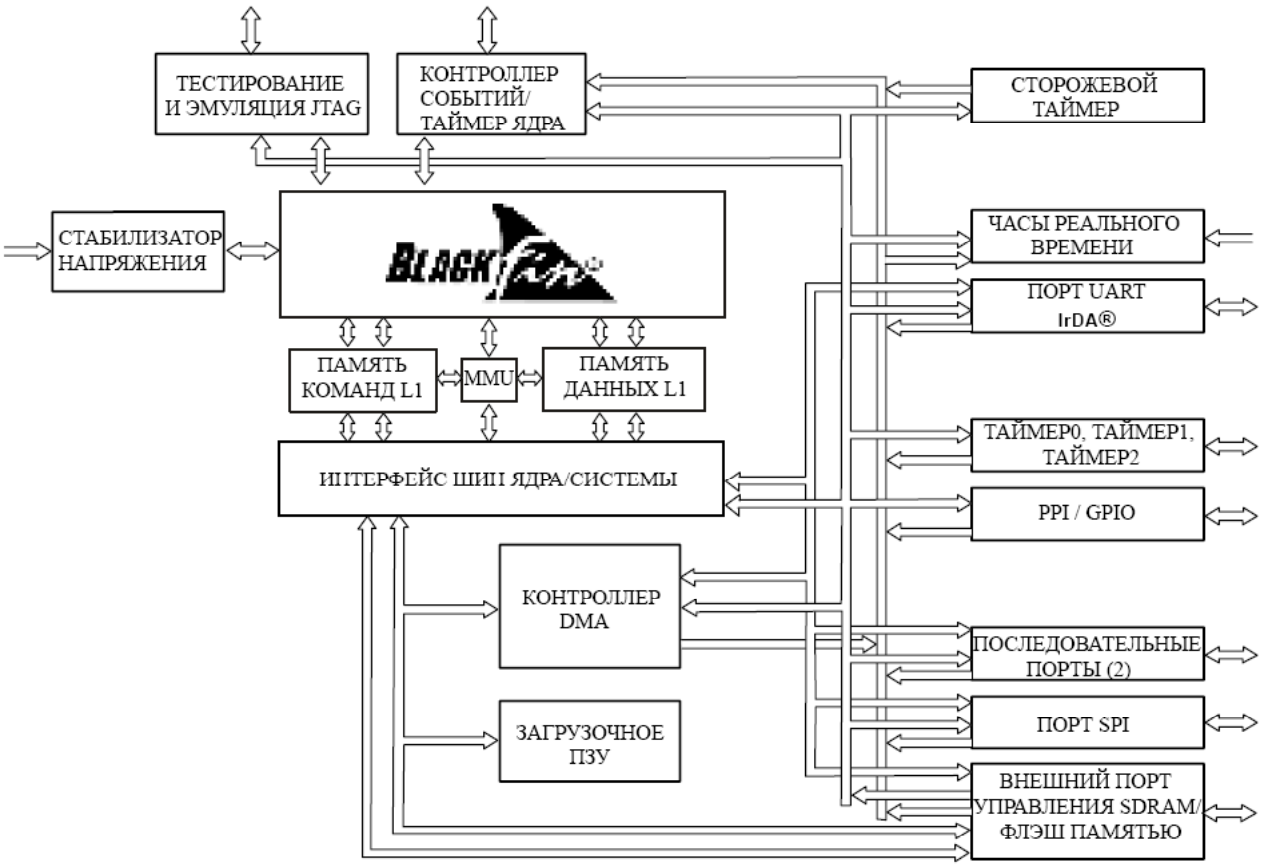


Рисунок 7 – Блок-схема процессора

Архитектура ядра процессора Blackfin является архитектурой с единым набором команд, включающей ядро обработки сигналов со сдвоенным блоком умножения-накопления, имеющей ортогональный набор команд, характерный для RISC-микропроцессоров, обладающей гибкостью команд типа SIMD и мультимедийными возможностями. Особенностью продуктов семейства Blackfin является динамическое управление питанием. Возможность изменения как напряжения питания, так и рабочей частоты позволяет оптимизировать потребление мощности в соответствии с конкретной задачей.

Периферийные устройства системы процессора включают:

- Параллельный периферийный интерфейс (PPI).
- Последовательные порты (SPORT)
- Последовательный периферийный интерфейс (SPI)
- Таймеры общего назначения
- Универсальный асинхронный приёмник-передатчик (UART)
- Часы реального времени (RTC)
- Сторожевой таймер

• Порт ввода/вывода общего назначения (программируемые флаги)

Эти периферийные устройства соединены с ядром несколькими шинами с высокой пропускной способностью, как показано на рисунке 7.

Параллельный периферийный интерфейс (PPI, Parallel Peripheral Interface) – это полудуплексный двунаправленный порт, поддерживающий передачу данных разрядностью до 16 бит. Он имеет выделенный вывод тактовой синхронизации, три мультиплексируемых вывода кадровой синхронизации и четыре выделенных вывода данных.

Последовательные порты (SPORT0 и SPORT1) обеспечивают интерфейс ввода/вывода с различными последовательными периферийными устройствами. Они поддерживают разнообразные протоколы последовательной передачи данных и могут обеспечивать прямое соединение процессоров в многопроцессорной системе.

Порт последовательного периферийного интерфейса (SPI, Serial Peripheral Interface) представляет собой четырехпроводной интерфейс с двумя выводами данных, выводом выбора устройства и выводом сигнала синхронизации. SPI является полнодуплексным синхронным последовательным интерфейсом, поддерживающим режимы ведущего и ведомого устройств и режим с несколькими ведущими устройствами.

Процессор имеет три идентичных 32-разрядных таймера общего назначения, таймер ядра и сторожевой таймер. Для каждого из таймеров общего назначения может быть индивидуально задан один из трёх режимов работы:

• режим широтно-импульсной модуляции (PWM_OUT),

• режим измерения периода и длительности импульса (WDTH_CAP),

• режим счётчика внешних воздействий (EXT_CLK).

Таймер ядра генерирует периодические прерывания, используемые в задачах синхронизации системы.

Универсальный асинхронный приёмопередатчик (UART, Universal Asynchronous Receiver Transmitter) – это полнодуплексное периферийное устройство, совместимое с интерфейсами UART индустриального стандарта. UART преобразует данные между последовательным и параллельным форматами. Последовательная передача (приём) выполняется по асинхронному протоколу, поддерживающему различные длины слов данных, различное количество стоп-битов и возможность формирования битов проверки чётности.

Блок **часов реального времени** (RTC, Real-Time Clock) процессора обеспечивает набор свойств цифровых часов, включающий функции будильника, секундомера и индикации текущего времени.

Процессор имеет 16 двунаправленных программируемых флагов (PFx) или выводов I/O общего назначения, PF[15:0]. Каждый вывод может индивидуально настраиваться на вход или на выход при помощи регистра направления флагов (FIO_DIR)

Лабораторная работа №3 «Исследование импульсных преобразователей»

СОДЕРЖАНИЕ

Введение

В настоящее время импульсная техника имеет высокую популярность в связи с появлением на рынке новых компонентов, позволяющих разрабатывать высокоэффективные устройства и ростом требованиям к КПД и коэффициенту мощности силовых устройств. Особое значение эффективные преобразователи имеют в системах управления и энергопитания автономными объектами, в т.ч. микро- и наноспутниками.

Важным элементом любого электронного устройства является система энергопитания, во многом определяющая энергоэффективность, массогабаритные показатели, надежность и способность к работе в условиях нестабильного внешнего энергопитания. На данный момент большинство систем энергопитания строится на базе импульсных преобразователей, обеспечивающих максимальный КПД при минимальных массогабаритных показателях. Недостатком импульсных преобразователей следует признать повышенный уровень пульсаций и необходимость в специальных средствах фильтрации для подавления собственных помех.

Основной элементной базой для построения импульсных преобразователей в диапазоне напряжений до 100-300В являются МДП транзисторы, имеющие низкое сопротивление канала во включенном состоянии. Основными схемами построения преобразователей без гальванической развязки, в зависимости от требований, являются чоппер (chopper), бустер (booster), инвертор (inverter) и схема SEPIC.

Наиболее популярным практическим путем построения таких преобразователей является применение интегральных микросхем различных производителей, включающих минимальное количество внешних дополнительных компонентов и реализующих заданные требования параллельно с обеспечением ряда защитных и сервисных функций. Основными производителями таких устройств являются такие фирмы, как Linear Technology, National Semiconductor, Maxim, Texas Instruments, Power Integrations и др.

Для облегчения работы разработчика фирмы производители предлагают различный инструментарий, включая онлайн средства разработки. К числу таких инструментов относится пакет LTspice IV, предлагаемый фирмой Linear Technology. Используя его возможности, разработчик может построить модель преобразователя, исследовать его основные характеристики при изменении внешних условий, определить энергетические показатели. Лабораторная работа посвящена освоению базовых навыков работы с пакетом и исследованию характеристик основных схем импульсных преобразователей без гальванической развязки.

1 Основные сведения о работе с LTspice IV

Пакет LTspice IV является мощным средством моделирования работы электрических схем с довольно простым интерфейсом пользователя. Пакет позволяет моделировать сложные электрические схемы, получая разнообразные характеристики (АЧХ, переходные характеристики, статистический анализ и т.п.). Для наших целей мы будем использовать небольшую часть возможностей пакета, основные необходимые функции перечислены ниже.

Запустить программу компьютерного моделирования «LTspice IV» из меню «Пуск» или с помощью ярлыка 

на рабочем столе.

1.1 Открытие файлов

Для открытия файлов необходимо нажать на кнопку  («File/Open…») и выбрать необходимый файл. Разработчики программы снабдили свой продукт большим количеством примеров, в основном основанных на продукции фирмы Linear Technology. Примеры расположены в папке «Examples». Во вложенной в неё папке «Educational» можно найти примеры типовых аналоговых и цифровых электронных схем, в папке «jigs» - примеры устройств на базе конкретных микросхем, выпускаемых фирмой Linear Technology.

1.2 Создание схемы

Для создания схемы можно воспользоваться меню «File/New Schematic», после чего добавлять на поле схемы разнообразные элементы из библиотеки, соединять их проводниками аналогично известному пакету ORCAD. Другой вариант действий – выбрать готовый пример из описанной выше папки, сохранить его под другим именем, отредактировав при необходимости. Для выполнения лабораторной работы воспользуемся готовыми схемами, находящимися в папке «Training», также расположенной в главной папке программы. Для выполнения лабораторной работы скопируйте содержимое нужных файлов в свою папку. ВНИМАНИЕ! Запрещается изменять содержимое файлов в папке «Training».

1.3 Изменение параметров компонентов схемы

Для изменения параметров компонента достаточно щелкнуть правой кнопкой мыши на выбранном компоненте, после чего в диалоговом окне изменить необходимый атрибут. Например, для резистора отобразится окно, представленное на рис. 1. При изменении величины сопротивления, емкости и т.п. необходимы дробные единицы, для указания которых используются приставки m (10^{-3}), u (10^{-6}), n (10^{-9}), p (10^{-12}), k (10^{3}), M (10^{6}). Обратите внимание на кнопку «Select», использование которой помогает выбрать стандартный компонент с параметрами, близкими к приведенным на схеме. Можно также получить список элементов с номиналами, используя меню «View/Bill of materials», при этом можно привести его прямо на схеме или скопировать в буфер для последующего включения в другие документы.

Рисунок 1 – Диалоговое окно свойств компонента

1.4 Анализ работы схем

Для выбора и изменения параметров моделирования необходимо открыть окно редактора «Simulate/Edit Simulation Cmd» (рис. 2). Здесь можно указать параметры моделирования, определяющие поведение имитатора: время анализа, тип интересующих данных и т.п. Для данной лабораторной работы ограничимся анализом «Transient». Длительность моделирования можно изменить в окне «Stop time», указав конкретное время (на рис. 3 -10мс). Полезно также включить отмечаемый блок «Stop simulating, if steady state is detected», указывающий программе остановить моделирование при завершении переходного процесса и переходе в стационарный режим. Эта опция необходима, если требуется получить отчёт об энергетических показателях («View/Efficiency report»).

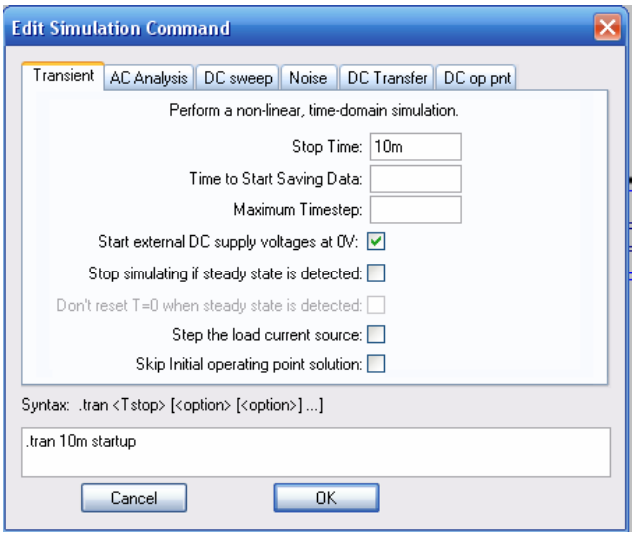
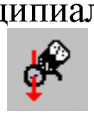


Рисунок 2 – Диалоговое окно выбора параметров моделирования

Запустить моделирование можно с помощью пиктограммы  или меню («Simulate/Run»).

1.5 Представление и анализ результатов

По завершении моделирования открывается окно, где можно просмотреть результаты. Правым щелчком мыши по этому окну можно открыть контекстное меню, откуда выбрать необходимые функции. Можно выбрать необходимый график из списка, можно использовать свои функции для операций над первичными токами и напряжениями, также можно выбрать нужный график, указав нужный компонент на схеме.

Для получения графиков зависимости напряжения в точке схемы от времени, необходимо, в окне принципиальной схемы, навести курсор на необходимый узел до появления символа  и нажать на левую кнопку мыши. Для формирования дополнительной панели с результатами моделирования можно воспользоваться меню «Plot settings/Add Plot Pane». Для получения графиков зависимости тока, протекающего через элемент, от времени, необходимо в окне принципиальной схемы, навести курсор на необходимый элемент до появления символа  и так же нажать на левую кнопку мыши.

Изображение на графиках можно масштабировать, имеется функция автоматического подбора масштаба изображения, можно также включать и выключать координатную сетку.

Получить числовые характеристики схемы можно с помощью меню «View/Efficiency report».

2 Изучение работы схемы типа «Booster»

Для исследования этой схемы откройте файл «booster.asc». В рабочем окне программы должна отобразиться схема, показанная на рис. 3

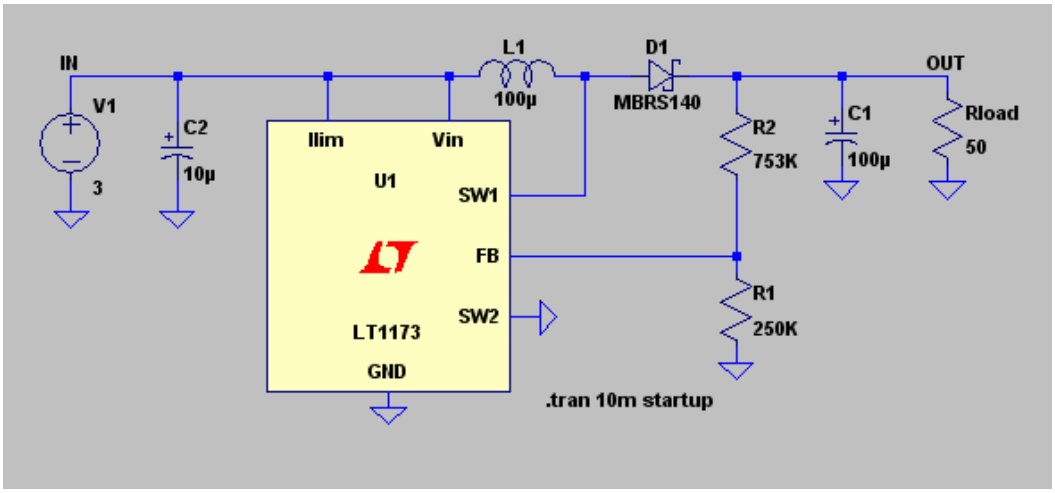


Рисунок 3 – Схема исследуемого преобразователя типа booster

Исследование схемы будет состоять из следующих этапов:

- Изучите нагрузочную способность схемы, изменяя величину сопротивления нагрузки и контролируя выходное напряжение, ток дросселя и напряжение на выводе SW1;
- Изменяя коэффициент передачи резистивного делителя, изучите регулировочные способности схемы;
- Изменяя величины емкости конденсаторов C1 и C2, изучите изменение величины пульсации выходного напряжения и влияние на входной ток;
- Изменяя величину индуктивности дросселя, изучите изменение параметров схемы и сделайте вывод о выборе величины индуктивности в зависимости от величины сопротивления нагрузки и входного напряжения;

– Включите другой диод в схему, сделайте вывод о влиянии параметров диода на работу бустера;

– Получите отчет об энергетических показателях схемы и КПД в номинальном режиме.

3 Изучение принципа работы схемы типа «Chopper»

Выберите для моделирования файл «chopper.asc». Должна отобразиться схема, приведенная ниже.

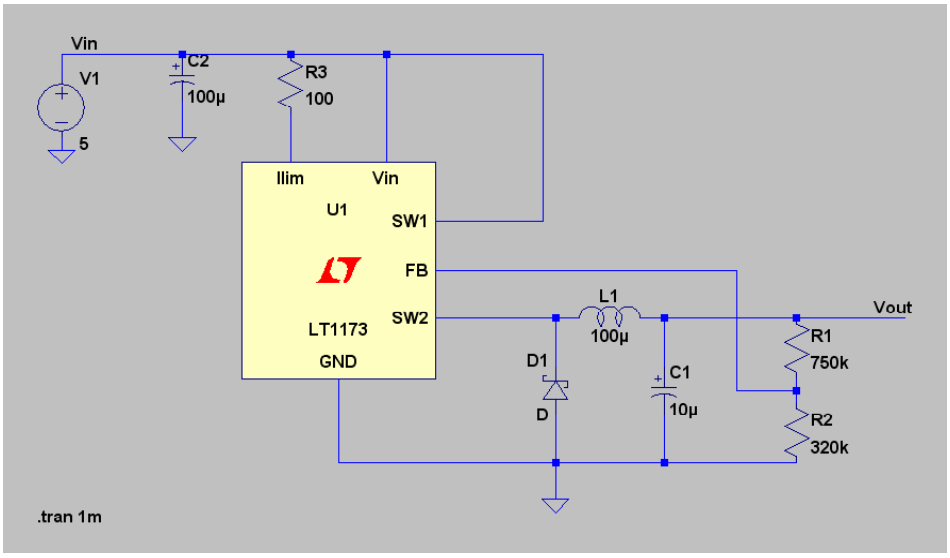


Рисунок 4 – Схема преобразователя типа chopper

Исследуйте работу схемы аналогично исследованию бустера. Дополнительно изучите влияние величины входного напряжения на энергетические показатели схемы.

4 Изучение принципа работы схемы типа «Inverter»

Выберите для моделирования файл «inverter.asc». Должна отобразиться схема, приведенная ниже.

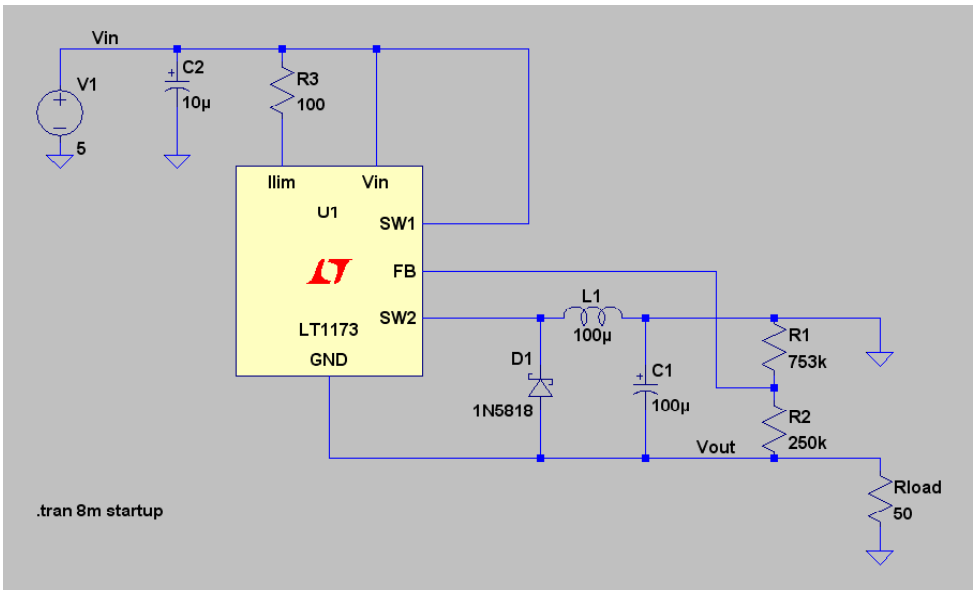


Рисунок 5 – Схема преобразователя типа inverter

Аналогично предыдущим случаям исследуйте возможности преобразователя и его параметры.

Приложение 1

Список индивидуальных заданий

№	Задание			
	Входное напряжение, В	Выходное напряжение, В	Сопротивление нагрузки, Ом	КПД, не менее, %
1.	10	5	1	88
2.	5	10	5	85
3.	3.3	1.8	0.1	80
4.	3.3	1.2	0.1	85
5.	27	12	100	95

Требования по выполнению индивидуального задания

Задание выполняется студентом индивидуально, результат оформляется в виде отчета по лабораторной работе. При выполнении задания необходимо подобрать подходящую микросхему стабилизатора, внешние элементы, провести моделирование, оценить энергетические показатели.

В отчет должны быть включены:

1. Принципиальная схема стабилизатора с заданными параметрами;
2. Результаты моделирования в виде отчета с энергетическими показателями, графиков выходного напряжения, токов через дроссель и диод;
3. Перечень элементов, сформированный программой.

Лабораторная работа №4 «Разработка программного обеспечения микроконтроллеров MSP430»

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Шестнадцатиразрядные микроконтроллеры семейства MSP430 занимают промежуточное положение между сигнальными процессорами и обычными микроконтроллерами, благодаря наличию как универсального набора периферийных модулей, характерных для универсальных устройств, так и 12-разрядного АЦП со встроенным источником опорного напряжения, что, позволяет реализовывать алгоритмы цифровой обработки сигналов. Отметим также наличие аппаратного модуля умножения и гибкий выбор режимов пониженного энергопотребления, что позволяет эффективно использовать данный контроллер в устройствах, критичных к величине потребляемой мощности, что является немаловажным для разработки экономичных систем управления.

Микроконтроллер обладает значительным объемом памяти на кристалле и снабжен системой внутрисхемного программирования на базе интерфейса JTAG, что также сокращает сроки разработки устройств на его основе. Средства разработки включают в себя высокоуровневый язык Си, использование которого позволяет облегчить программисту реализацию ряда рутинных операций, вроде обслуживания интерфейсов и реализации типовых вычислительных алгоритмов, характерных для систем управления.

Следует отметить, что в настоящее время разработчик микропроцессорных систем должен одинаково хорошо владеть как техникой программирования на ассемблере, так и языками высокого уровня, из которых наиболее широкое распространение получил язык Си.

Данные методические указания позволяют студентам изучить основы применения обоих языков для разработки программ современных микроконтроллеров. Указания не претендуют на полноту описания особенностей MSP430, приводятся лишь краткие пояснения, необходимые для понимания приведенных фрагментов, в приложении приводится таблица с системой команд.

1 Создание нового проекта

Написание и отладку программ для микроконтроллеров будем производить с помощью среды интегрированной разработки IAR EMBEDDED WORKBENCH, разработанной фирмой IAR SYSTEMS.

Для работы необходимо создать директорию с проектом, в которой могут быть размещены необходимые файлы с текстом исходной программы, служебные файлы и выходной файл. Рекомендуется создать папку с именем, набранным латинскими буквами, для избежания проблем с интерпретацией имени в ряде случаев.

Для создания нового проекта удобно использовать меню «Project/Create New Project». После активизации этого меню программа предложит выбрать тип проекта, который предполагается создать. Из предлагаемого списка выберем тип «asm», укажем место размещения (в предварительно созданной папке) и в диалоговом окне введем имя проекта (также латинскими буквами). Программа автоматически создаст заготовку файла, в котором будет находиться программа на языке ассемблера и включит этот файл в проект. После этого, в

любое время можно добавить дополнительные файлы или удалить уже включенные в состав проекта.

2 Создание исходных файлов

2.1 Создание файла на языке ассемблера

В состав используемой среды интегрированной разработки входит текстовый редактор, который удобно использовать для создания текстового файла исходного текста.

При создании файла следует придерживаться следующих правил:

- *С первой позиции набираются метки и директивы ассемблера;*
- *Команды микроконтроллера набираются в следующей колонке (рекомендуется использовать табуляцию);*
- *Для лучшей читаемости программы директивы ассемблера набираются заглавными буквами, а команды строчными;*
- *Комментарии начинаются символом* «;»;
- *Числа в шестнадцатеричном формате записываются с суффиксом h;*
- *Числа в двоичном формате записываются с суффиксом b.*

Создайте файл, содержащий следующий текст:

```
    NAME    main
        #include        "msp430x14x.h"                  ; Стандартные определения
result  EQU   0200h                                     ; Начало таблицы
        PUBLIC  main

        ORG     0FFFEh
        DC16    main
        ORG     0FFEEh                                  ; Прерывание от ADC12
        DW      ADC12ISR

        RSEG    CODE
main    mov     #0A00h,SP                               ; Инициализация стека
        mov     #WDTPW+WDTHOLD,&WDTCTL                  ; Стоп WATCHDOG
        mov     #ADC12ON+REFON+REF2_5V+SHT0_6,&ADC12CTL0
; Включение АЦП, установка опорного напряжения 2.5В, установка режима УВХ
        mov     #SHP,&ADC12CTL1                         ; Режим пуска
        mov.b   #INCH_0+SREF_1,&ADC12MCTL0              ; Выбор канала
        bis     #BIT0,&ADC12IE                          ; Разрешить прерывание
        mov     #03600h,R15                             ; Задержка для завершения
L$1     dec     R15                                     ; переходных процессов
        jnz     L$1
        bis     #ENC,&ADC12CTL0                         ; Запуск преобразования
        clr     R6
        eint                                            ; Разрешение прерываний
Moop    bis     #ADC12SC,&ADC12CTL0                     ; Пуск преобразования
        bis     #CPUOFF,SR                              ; Переход в режим малого
                                                        ; потребления для ожидания
                                                        ; завершения преобразования
        jmp     Moop                                    ; Организация цикла.
;-------------------------------------------------------------------------------
ADC12ISR    ; Подпрограмма обработки прерывания для ADC12
```

```
;-------------------------------------------------------------------------
      mov    &ADC12MEM0,result(R6)          ; Сохранить результат
      incd   R6                             ; Инкремент указателя
      and    #0FEh,R6                          ; Модификация буфера
      bic    #CPUOFF,0(SP)                ; Возврат в активный режим
      reti
      END    main
```

3 Компиляция программы

Для ассемблирования и сборки объектного файла программы можно воспользоваться меню PROJECT, где выбрать вкладку MAKE. Так как мы воспользовались автоматическим созданием проекта и нам не требуются особые настройки компилятора или линковщика, настройки по умолчанию будут вполне соответствовать нашим требованиям. Изменить эти настройки можно с помощью меню «Project/Options».

После завершения компиляции автоматически открывается окно MESSAGES с результатами выполнения.

3.1 Запуск отладчика и режимы выполнения

Для запуска отладчика необходимо выбрать в меню PROJECT вкладку DEBUG, после чего открывается окно отладчика.

Для пошагового выполнения набранной программы можно воспользоваться клавишей F10 (STEP) или воспользоваться соответствующим значком панели инструментов. При каждом нажатии указанной клавиши выполняется одна машинная команда. При этом можно наблюдать результаты выполнения в окнах регистров микроконтроллера и памяти. Необходимо отметить, что нажатие F10 отличается от F11 (STEP INTO), тем, что подпрограммы выполняются, как одна команда без захода в тело процедуры.

Кроме пошаговой отладки возможен прямой запуск программы (необходимо предварительно установить точку останова), выполнение с задержкой (AUTOSTEP), при котором команды выполняются с устанавливаемой задержкой. Кроме этого, возможен запуск с остановкой в позиции курсора, выполнение до момента выхода из подпрограммы.

3.2 Модификация и просмотр ячеек памяти, переменных и регистров

Для модификации регистров необходимо включить окно «Register» (вкладка Register меню View). Для модификации регистра можно непосредственно в окне ввести необходимое значение и нажать ENTER.

Для просмотра и модификации переменных удобно использовать окно «Watch». Используя щелчок правой клавишей мыши в этом окне можно добавить (ADD), удалить (REMOVE) или изменить формат отображения переменной. В окне «Quick watch» можно также просматривать значение переменных, значение которых не обязательно инспектировать постоянно. При необходимости можно перенести временную переменную в постоянное окно WATCH. Необходимо иметь в виду, что для отображения значения переменной в этих окнах необходимо первым символом имени записывать «#», например, для отображения содержимого регистра ADC12MEM0 в окне нужно ввести #ADC12MEM0.

Для просмотра и модификации ячеек памяти можно использовать вкладку MEMORY. При этом открывается диалоговое окно, в котором можно просматривать и модифицировать требуемые ячейки. Для удобства можно отображать отдельно области памяти, относящиеся к ОЗУ, FLASH EEPROM и.т.д. С помощью этого окна можно также установить точки останова на доступ к отдельным ячейкам памяти или целому диапазону.

3.3 Установка точек останова

Используемая среда программирования предполагает возможность применения трех типов точек останова: на конкретной инструкции кода, на доступ к ячейке памяти и точка, не останавливающая выполнения программы (Immediate).

– Для редактирования параметров точек останова можно использовать вкладку «Edit breakpoints», при этом открывается специальное диалоговое окно.

К точке останова удобно привязать выполнение макроса, который будет автоматически модифицировать память или выводить какую-нибудь информацию. Для этого служит редактируемое поле «Action», в котором необходимо указать имя макрофункции, которую требуется выполнить.

3.4 Использование макрофункций

Макрофункции записываются в отдельном файле с расширением «mac» с использованием Си-подобного языка, описание которого приводится во встроенной системе подсказок. Применение макросов позволяет существенно расширить возможности эмуляции внешних событий. Для использования макросов необходимо воспользоваться меню «Debug/Macros», далее указать имя файла, добавить его к списку («Add») и зарегистрировать («Register»). После этого макросы станут доступны отладчику. Создайте макрос следующего содержания:

```
Mymac()
{
__writeMemory16(100,0x140,"Memory");
}
```

Зарегистрируйте его в системе и привяжите к точке останова типа Immediate, которую установите на доступ по чтению к регистру ADC12MEM0 (адрес 0x140). При этом автоматически при чтении регистра результата преобразования АЦП в нем будет записываться число 100.

3.5 Эмуляция прерываний

Для эмуляции прерывания необходимо использовать вкладку «Interrupts» меню «Simulator». При этом требуется установить флажок «Enable», выбрать требуемый вектор прерывания, указать число тактов до момента активации «First Activation», период повторения «Repeat interval», длительность запроса прерывания «Hold time» (укажите «Infinite»), вероятность появления прерывания в указанный период в процентах и возможное отклонение момента возникновения запроса относительно указанного периода повторения в процентах.

Существует также возможность принудительной активации прерывания в любой момент с помощью меню «Simulator/Forced Interrupts». В списке необходимо выделить нужный вектор и нажать кнопку «Trigger».

4 Работа с программами на языке Си

Использование языка Си может значительно ускорить работу над проектом и облегчить работу программиста за счет использования богатого набора библиотек и стандартных функций. Вместе с тем, компилятор, конечно, уступает изобретательности человеческого разума в части поиска нестандартных решений, и это обстоятельство может сказаться на результатах работы. В частности, полученный код может оказаться больше по размеру и медленнее, чем код, полученный компиляцией исходного текста программы на ассемблере.

4.1 Создание текста программы

Создайте новый проект, указав тип исходного файла C++, и наберите в файле следующую программу:

```
#include        "msp430x14x.h"                  // Стандартные определения
#define   Num_of_Results   128
static unsigned int results[Num_of_Results];    // Массив для хранения результатов

void main(void)
{
WDTCTL = WDTPW+WDTHOLD;                         // Стоп WATCHDOG
 ADC12CTL0 = ADC12ON+REFON+REF2_5V+SHT0_6;
  // Включение АЦП, установка опорного напряжения, установка режима УВХ
  ADC12CTL1 = SHP;                              // Режим пуска
   ADC12MCTL0 = INCH_10+SREF_1;                 // Выбор канала
    ADC12IE = 0x01;                             // Разрешение прерывания
// Задержка для завершения переходных процессов
  for (int i=0; i<0x3600; i++)  { }
   ADC12CTL0 |= ENC;                             // Пуск преобразования
     _EINT();                                   // Разрешение прерывания

          while(1)
            {
    ADC12CTL0 |= ADC12SC;                       // Пуск преобразования
    _BIS_SR(LPM0_bits);                         // Переход в режим LPM0
          _NOP();                               // Для точки останова
            }
}

#pragma vector=ADC_VECTOR
__interrupt void ADC12_ISR ()
{
static unsigned int index = 0;
  results[index] = ADC12MEM0;
   index = (index+1)%Num_of_Results;            // Increment results index, modulo
    _BIC_SR_IRQ(LPM0_bits);                     // Переход  в активный режим
}
```

4.2 Компиляция исходного текста и отладка программы

Процесс компиляции не особенно отличается от процесса компиляции файла на языке ассемблера, однако, просмотр результатов в окне «Disassembly» приводит к заключению, что размер кода значительно увеличился по сравнению с размером кода, полученного трансляцией файла на языке ассемблера.

Вернитесь в режим редактора исходного кода и откройте диалоговое окно настроек компилятора («Project/Options/C++ Compiler»). Выбрав вкладку «Code», поэкспериментируйте с настройками компилятора, наблюдая результаты в режиме отладчика.

После выбора оптимального, с вашей точки зрения, кода проделайте отладочные процедуры аналогично работе с ассемблерным кодом и убедитесь в идентичности программ с точки зрения функциональности.

4.3 Анализ программы

Для анализа эффективности программы имеется два инструмента: профилировщик и анализатор выполнения кода. Профилировщик запускается из меню «View/Profiling». Далее нужно активизировать его из собственной панели инструментов и установить удобный режим отображения. В окне профилировщика приводятся числовые данные (в циклах) и относительные (в процентах) чистого времени выполнения функций и общего времени с учетом выполнения дочерних и библиотечных функций.

Анализируя эти данные, можно сделать выводы об узких местах программы и направлении путей оптимизации кода.

Анализатор выполнения предназначен для поиска частей программы, не получающих управления по каким-то причинам. В частности, красным цветом выделены функции, не получающие управления, зеленым - выполняемые целиком, красно-зеленым - выполняемые частично. Внутри «красных» функций можно увидеть отдельные переходы «желтые», не получающие управления.

Поэкспериментируйте с профилировщиком и анализатором, меняя способ оптимизации кода и периодичность генерируемых прерываний.

5 Порядок выполнения работы

1. Исследуйте работу программы на языке ассемблера, для чего выполните всю программу в пошаговом режиме с имитацией прерывания.
2. Исследуйте работу программы на языке Си, для чего выполните всю программу в пошаговом режиме с имитацией прерывания.
3. Исследуйте производительность Си-программы с помощью профилировщика и анализатора кода, сделайте выводы о целесообразности оптимизации и ее направленности.
4. Измените Си-программу так, чтобы происходило накопление серии из 200 результатов преобразования, получаемых по двум каналам АЦП с сохранением в памяти.

6 Приложение

6.1 Архитектура модуля АЦП микроконтроллера MSP430

Модуль АЦП MSP430 позволяет производить сбор данных с 8 внешних источников, причем результаты автоматически сохраняются в отдельных регистрах. В качестве источника опорного напряжения могут быть использованы:

– Напряжение питания;
– Внутренний источник опорного напряжения (1.5В или 2.5В);
– Внешний источник опорного напряжения.

Необходимо отметить, что диапазон преобразования задается двумя величинами (VR+ и VR-), программируемыми индивидуально для каждого канала преобразования. Результат преобразования определяется по формуле:

$$N = \frac{V_{IN} - V_{R-}}{V_{R+} - V_{R-}} \times 4095 .$$

Частота преобразования может также задаваться от различных источников (до 5 МГц). Имеется гибкая система запуска преобразования, позволяющая учитывать индивидуальные особенности отдельных источников сигнала. MSP430 оснащен внутренним датчиком температуры, облегчающих построение систем с алгоритмами термокомпенсации.

Модуль АЦП может работать в режиме однократного и автоматического многократного преобразования, обеспечивая многоканальный сбор данных.

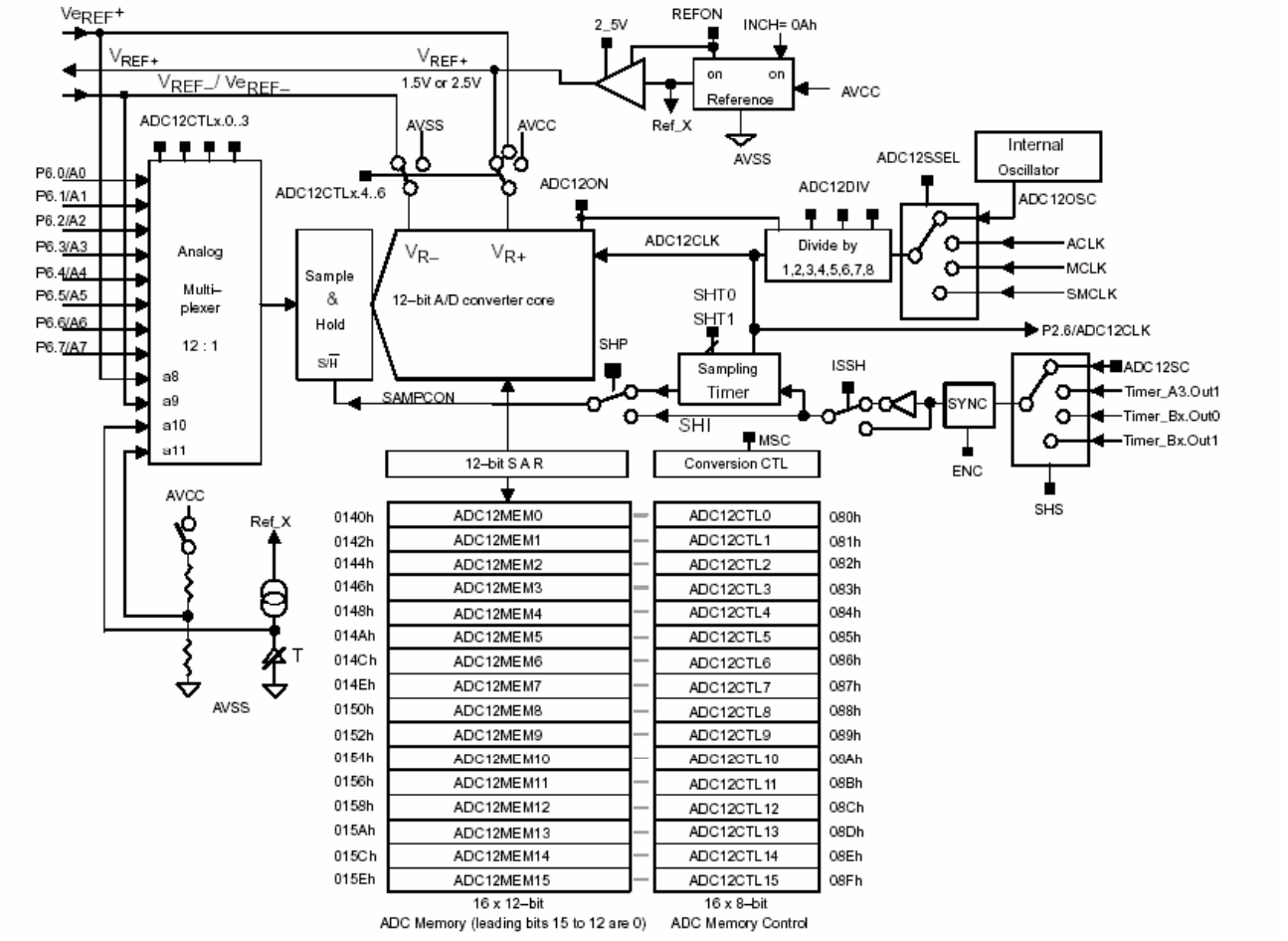


Рисунок 6.1 - Структура модуля АЦП

Для управления модулем 12 - разрядного АЦП используются следующие регистры:

- ADC12CTL0, ADC12CTL1 - управляющие регистры;
- ADC12IFG - регистр флагов прерывания;
- ADC12IE - регистр масок прерывания;
- ADC12IV - регистр векторов прерывания.

Структура ADC12CTL0, ADC12CTL1 приведена на рис. 6.2. Серым цветом выделены биты, модификация которых возможна только при ENC=0.

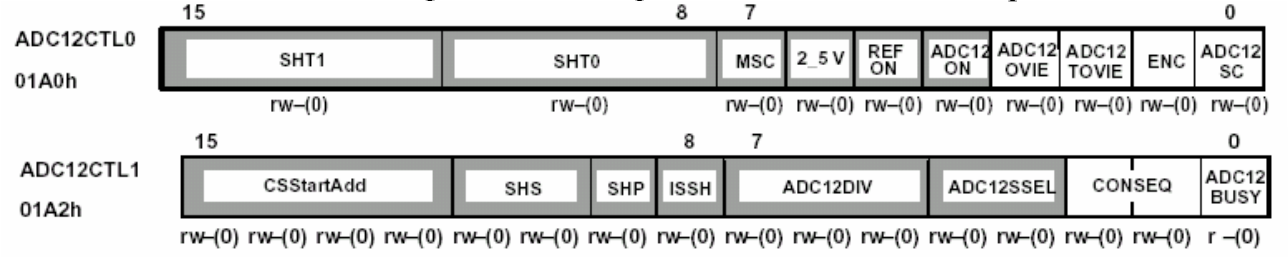


Рисунок 6.2 - Управляющие регистры модуля АЦП

SHTX - 4 бита, определяющих длительность интервала выборки УВХ в циклах сигнала ADC12CLK. SHT1 и SHT0 определяют длительность сигнала выборки соответственно для регистров ADC12MEM15-ADC12MEM8 и ADC12MEM7-ADC12MEM0 (табл. 1).

Таблица 1

SHTX	0	1	2	3	4	5	6	7	8	9	10	11	12-15
n	4	8	16	32	64	96	128	192	256	384	512	768	1024

MSC - 0 - Таймер УВХ требует фронта сигнала SHI для запуска каждого преобразования;
Таймер УВХ требует фронта сигнала SHI для запуска первого преобразования, далее процесс автоматический.

REF2_5V - Напряжение источника опорного напряжения (0 - 1.5В, 1 - 2.5В).

REFON Включение источника опорного напряжения (1 - включен).

ADC12ON - Включение модуля АЦП (1 - включен).

ADC12OVIE - Разрешение прерывания по переполнению (перезаписи) регистров ADC12MEMX (1 - разрешено).

ADC12TOVIE - Разрешение прерывания по перезаписи таймера УВХ (1 - разрешено).

ENC - 1 - преобразование разрешено.

ADC12SC - Старт преобразования (1 запускает таймер УВХ).

CSTARDADDX - 4 бита, указывающих адрес регистра, в который будет записан результат преобразования, или адрес первого регистра в группе в режиме группового преобразования.

SHSX - 2 бита, указывающих на источник сигнала управления УВХ

00	Бит ADC12SC
01	Таймер A (OUT1)
10	Таймер B (OUT0)
11	Таймер B (OUT1)

SHP - режим запуска УВХ (0- непосредственно сигналом запуска, 1 - таймером)

ISSH - Инверсия сигнала управления УВХ (0 - не инвертирован)

ADC12DIVX - 3 бита, определяющих коэффициент делителя частоты

000	1
001	2
010	3
011	4
100	5
101	6
110	7
111	8

ADC12SSELX - 2 бита, выбора источника синхросигнала АЦП

00	ADC12OSC
01	ACLK
10	MCLK
11	SMCLK

CONSEQX - 2 бита, определяющих режим работы модуля АЦП

00	Одноканальный однократный
01	Многоканальный
10	Одноканальный многократный
11	Многоканальный многократный

ADC12BUSY - 1 - идет преобразование, 0 - модуль неактивен.

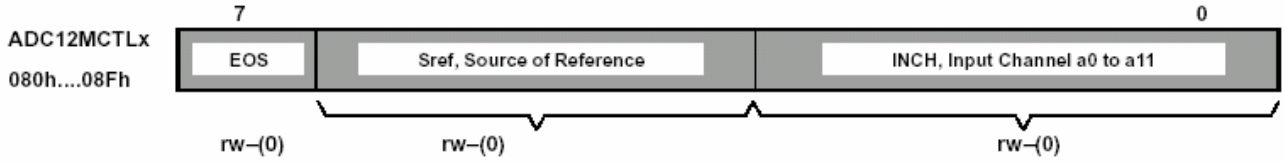


Рисунок 6.3 - Регистр управления преобразованием

Регистры управления преобразованием ADC12MCTL15…0 позволяют индивидуально настраивать 16 каналов, каждый из которых может быть подключен к определенному выводу MSP430, а также иметь индивидуально настроенный источник опорного напряжения. Структура регистров приведена н7а рис. 6.3.

EOS - 1, когда данный регистр - последний в группе, иначе 0.

SREFX 3 - бита, определяющих опорное напряжение.

000	VR+=AVCC, VR-=AVSS
001	VR+=VREF+, VR-=AVSS
010	VR+=VeREF+, VR-=AVSS
011	VR+=VeREF+, VR-=AVSS
100	VR+=AVCC, VR-=VREF-/VeREF-
101	VR+=VREF+, VR-=VREF-/VeREF-
110	VR+=VeREF+, VR-=VREF-/VeREF-
111	VR+=VeREF+, VR-=VREF-/VeREF-

INCHX - 4 бита, определяющих номер канала

0000-0111	Каналы A0 - A7
1000	VeREF+
1001	VREF-/VeREF-
1010	Датчик температуры
1011-1111	(AVCC-AVSS)/2

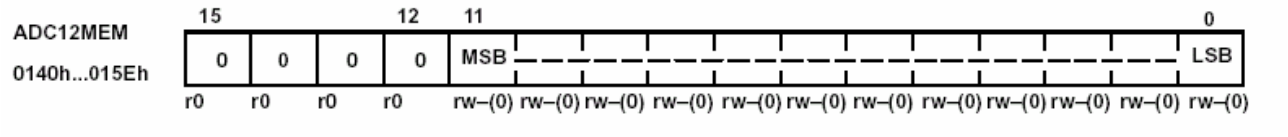


Рисунок 6.4 - Регистры хранения результатов преобразования

Регистры ADC12IE, ADC12IFG - 16-битные регистры, содержащие соответственно биты маски прерываний от соответствующих каналов и флаги прерывания, устанавливаемые, когда АЦП производит запись результата.

Регистр ADC12IV (доступный только для чтения) содержит векторы прерывания, используя которые, можно легко определить номер канала, вызвавшего прерывание.

Содержимое ADC12IV	Источник прерывания	Флаг прерывания
000h	Нет прерывания	-
002h	Перезапись ADC12MEMX	-
004h	Перезапуск АЦП	-
006h+2*n	ADC12MEMn	ADC12IFGn

Микроконтроллер MSP430 имеет векторную систему прерываний, в частности прерывания от АЦП имеют общий вектор прерывания, расположенный по адресу 0xFFEEh (для MSP430x13X, MSP430x14X). Адрес подпрограммы обработки прерывания должен находиться в ячейке памяти по этому адресу. Вектор сброса располагается в ячейке с адресом 0xFFFEh, здесь должен находиться адрес начала основной программы. При возникновении любого запроса прерывания от АЦП (при условии, что оно не замаскировано), MSP430 передает управление по фиксированному адресу. Выяснение конкретного источника запроса возлагается на подпрограмму обработки. Рекомендуется следующий алгоритм реализации подпрограммы обработки прерываний от АЦП:

```
INT_ADC12  ADD &ADC12IV,PC    ; Переход по вектору ADC12IV
           RETI               ; нет запроса прерывания
           JMP ADCOV          ; перезапись регистра
           JMP ADCTOV         ; перезапуск АЦП
           JMP ADCMEM0        ; запрос от ADCMEM0
           …
           JMP ADCMEM14       ; запрос от ADCMEM14
ADCMEM15 …                    ; обработка запроса от ADCMEM15
           JMP INT_ADC12      ; проверка других запросов
ADCMEM14 …                    ; обработка запроса от ADCMEM14
ADCOV      …                  ; обработка запроса от перезаписи
           JMP INT_ADC12
ADCTOV     …                  ; обработка запроса от перезапуска
           JMP INT_ADC12
```

6.2 Система команд MSP430

Система команд семейства MSP430 содержит 24 основные команды и 27 эмулируемых. Перечень команд приводится в таблице 2

Можно выделить команды с одним операндом, двумя операндами, и команды перехода. Все команды с операндами могут оперировать байтами или словами, при этом байтовые команды имеют окончание ".b", а команды, работающие со словами - ".w". По умолчанию предполагаются команды, работающие со словами.

Обозначения, используемые при описании команд.

src - операнд - источник;
dst - операнд - приемник;

Таблица 2

Мнемоника		Описание		V	N	Z	C
ADC(.B)*	dst	Сложение C с dst	dst + C → dst	*	*	*	*
ADD(.B)	src,dst	Сложение src с dst	dst + src → dst	*	*	*	*
ADDC(.B)	src,dst	Сложение src с C и dst	dst + src +C→ dst	*	*	*	*
AND(.B)	src,dst	Логическое "И" src и dst	dst AND src→ dst	0	*	*	*
BIC(.B)	src,dst	Сброс бита	not src AND dst →dst	-	-	-	-
BIS(.B)	src,dst	Установка бита	not src AND dst →dst	-	-	-	-
BIT(.B)	src,dst	Проверка бита	src AND dst	0	*	*	*
BR*	dst	Безусловный переход	dst→PC	-	-	-	-
CALL*	dst	Вызов подпрограммы	PC+2→Stack, dst→PC	-	-	-	-
CLR(.B)*	dst	Сброс приемника	0→dst	-	-	-	-
CLRC*		Сброс C	0→C	-	-	-	0
CLRN*		Сброс N	0→N	-	0	-	-
CLRZ*		Сброс Z	0→Z	-	-	0	-
CMP(.B)	src,dst	Сравнение src и dst	dst-src	*	*	*	*
DADC(.B)*	dst	Дес. сложение C и dst	dst+C→dst	*	*	*	*
DADD(.B)	src,dst	Дес. сложение src и dst	src+dst+C→dst	*	*	*	*
DEC(.B)*	dst	Декремент приемника	dst-1→dst	*	*	*	*
DECD(.B)*	dst	Двойной декремент	dst-2→dst	*	*	*	*
DINT*		Запрет прерываний	0→GIE	-	-	-	-
EINT*		Разрешение прерываний	1→GIE	-	-	-	-
INC(.B)*	dst	Инкремент приемника	dst+1→dst	*	*	*	*
INCD(.B)*	dst	Двойной инкремент	dst+2→dst	*	*	*	*
INV(.B)*	dst	Инверсия приемника	not dst→dst	*	*	*	*
JC/JHE	label	Переход, если C=1		-	-	-	-
JEQ/JZ	label	Переход, если Z=1		-	-	-	-
JGE	label	Переход, если N^V=0		-	-	-	-
JL	label	Переход, если N^V=1		-	-	-	-
JMP	label	Безусловный переход	PC+2*offset→PC	-	-	-	-
JN	label	Переход, если N=1		-	-	-	-

JNC/JLO	label	Переход, если C=0		-	-	-	-
JNE/JNZ	label	Переход, если Z=0		-	-	-	-
MOV(.B)	src,dst	Пересылка	src→dst	-	-	-	-
NOP*		Нет операции		-	-	-	-
POP(.B)*	dst	Выталкивание из стека	@SP→dst, SP+2→SP	-	-	-	-
PUSH(.B)	src	Занесение в стек	SP-2→SP, src→@SP	-	-	-	-
RET*		Выход из подпрограммы	@SP→PC,SP+2→SP	-	-	-	-
RETI		Выход из прерывания	@SP→PC,SP+2→SP	-	-	-	-
RLA(.B)	dst	Арифм. сдвиг влево		*	*	*	*
RLC(.B)	dst	Цикл. сдвиг влево		*	*	*	*
RRA(.B)	dst	Арифм. сдвиг вправо		0	*	*	*
RRC(.B)	dst	Цикл. сдвиг вправо		*	*	*	*
SBC(.B)*	dst	Вычитание $\overline{C}$ из dst	dst+0FFFFh+C→dst	*	*	*	*
SETC*		Установка C	1→C	-	-	-	1
SETN*		Установка N	1→N	-	1	-	-
SETZ*		Установка Z	1→Z	-	-	1	-
SUB(.B)	src,dst	Вычитание src и C из dst	dst+not src+1→dst	*	*	*	*
SUBC(.B)	src,dst	Вычитание src и $\overline{C}$ из dst	dst+not src+C→dst	*	*	*	*
SWPB	dst	Перестановка байт		-	-	-	-
SXT	dst	Знаковое расширение	Bit7→Bit8…Bit15	0	*	*	*
TST(.B)*	dst	Проверка dst	dst+0FFFFh+1	0	*	*	1
XOR(.B)	src,dst	Искл. ИЛИ src и dst	src XOR dst→dst	*	*	*	*

* Эмулируемая команда

MSP430 поддерживает семь режимов адресации источника и четыре режима адресации приемника, представленные в таблице 3

Таблица 3 Режимы адресации

Режим адресации	Синтаксис	Тип	Описание
Регистровая	Rn	src/dst	Операнд - содержимое регистра
Индексная	X(Rn)	src/dst	X+Rn указывает на операнд, хранящийся в памяти
Относительная	ADDR	src/dst	На операнд указывает PC+X
Абсолютная	&ADDR	src/dst	Абсолютный адрес операнда содержится в команде
Косвенная регистровая	@Rn	src	Rn - указатель на операнд (содержит адрес операнда)
Косвенная автоинкрементная	@Rn+	src	Rn - указатель на операнд. После операции содержимое Rn увеличивается на 1, если команда байтовая и на 2, если команда оперирует со словами
Непосредственная	#N	src	Операнд содержится в команде

6.3 Структура памяти MSP430

MSP430 - микроконтроллер фон-Неймановской архитектуры, карта его адресного пространства представлена на рис. 6.5.

Объем FLASH/ROM определяется типом контроллера, верхние 16 слов отведены под вектора прерывания.

RAM (ОЗУ) начинается с адреса 0200h и может использоваться для хранения данных и кода программы.

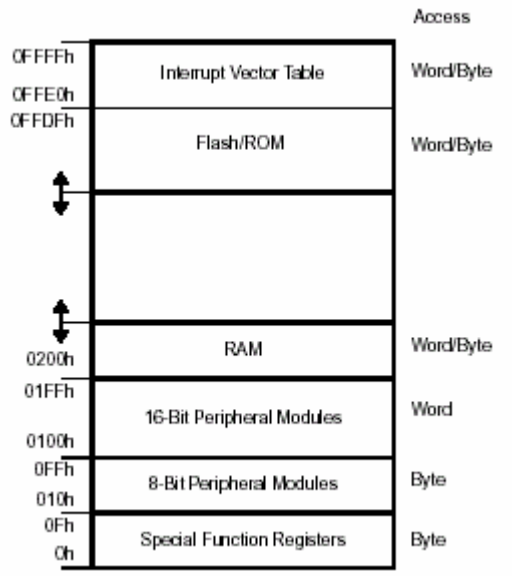
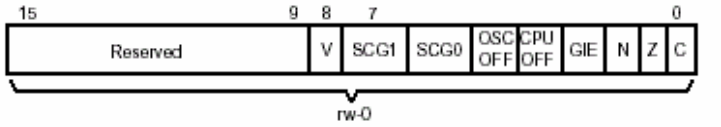


Рисунок 6.5 - Карта памяти MSP430

6.4 Регистры микроконтроллера

1. *Регистр статуса (SR)*



V - Бит переполнения (устанавливается в 1, когда результат операции не укладывается в формат числа со знаком.
SCG1 Системный синхрогенератор 1 SCG1=1 выключает SMCLK.
SCG0 Системный синхрогенератор 0 SCG1=0 выключает DCO.
OSCOFF=1 выключает LFXT1.
CPUOFF=1 выключает процессор.
GIE=0 глобальный запрет прерываний, 1 - разрешение прерываний
N - бит отрицательного результата (N=1, когда результат отрицательный)
Z - бит нулевого результата (Z=1, когда результат равен нулю)
C - бит переноса

2. *Указатель стека (SP/R1)*

16-разрядный регистр, хранящий текущее значение вершины стека.

3. *Генератор константы (R2/R3)*

Хранит в двух регистрах шесть наиболее часто используемых констант, используемых при программировании.

4. *Регистры общего назначения (R4 - R15)*

Лабораторная работа №5 «Разработка цифровых систем управляющих комплексов микроспутников на базе ПЛИС»

СОДЕРЖАНИЕ

Введение

Разработка систем управления автономными объектами в ряде случаев требует создания уникальных высокоскоростных цифровых систем, обладающих высокой надежностью и гибкостью. Для решения таких задач эффективно применение программируемых интегральных схем типа FPGA или CPLD.

В настоящее время программируемые логические интегральные схемы (ПЛИС) получили широкое распространение благодаря универсальности, простоте программирования, сокращению цикла проектирования конечного устройства, гибкости, доступности средств разработки. Применение ПЛИС требует от разработчика знания элементной базы, владения общими технологиями создания цифровых устройств и специальными технологиями реализации этих устройств на базе ПЛИС. Современный разработчик должен также уметь использовать языки описания аппаратуры (HDL) с технологиями структурного и поведенческого описания проектов. Использование HDL для работы с ПЛИС позволяет разрабатывать устройства, легко адаптируемые под различную элементную базу и легко подстраиваемые под изменяющиеся требования заказчика.

Одним из лидеров рынка ПЛИС в настоящее время является фирма XILINX, выпускающая широкий спектр микросхем с различными характеристиками. Для работы с продукцией фирмы XILINX предназначен программный пакет ISE (Integrated Software Environment, позволяющий разработчику создавать устройства различной степени сложности с применением современных технологий. Настоящие методические указания предназначены для освоения навыков проектирования цифровых устройств с использованием ПЛИС фирмы XILINX и вышеуказанного программного обеспечения. Также рассматриваются особенности применения широко распространенного языка VHDL, поддерживаемого большинством современных средств САПР.

Современный разработчик имеет возможность создавать свой проект, как с помощью графического изображения принципиальной схемы устройства, состоящего из традиционных узлов цифровой техники, так и пользуясь HDL, причем, возможно, сочетая преимущества обоих подходов. Рассматриваемая программная среда допускает построение иерархических проектов, включающих различные модули, выполненные с использованием обеих концепций. Например, разработчик может создать графическое изображение схемы устройства, состоящее из отдельных блоков, создаваемых с помощью VHDL и наоборот описать устройство глобально с помощью VHDL и использовать в качестве библиотечных модулей узлы, созданные в графическом редакторе.

В лабораторных работах навыки использования ПЛИС дополнительно закрепляются использованием отладочных плат, включающих в себя микросхемы фирмы XILINX и набор периферийных устройств.

В методических указаниях не приводится подробное описание всех возможных приемов, используемых при проектировании цифровых устройств и не описываются все особенности программного обеспечения. При необходимости можно воспользоваться встроенной документацией ISE, а также литературой из приведенного списка.

Настоящие методические указания предназначены для выполнения двух лабораторных работ, основной целью которых является овладение основными приемами проектирования цифровых устройств с использованием ПЛИС фирмы XILINX в интегрированной среде ISE. В первой лабораторной работе рассматриваются базовые процедуры по созданию проекта в графическом редакторе, моделированию и реализации цифрового устройства на базе ПЛИС. Итогом работы является отлаженный проект и результаты компьютерного моделирования.

Во второй лабораторной работе рассмотрена реализация более сложного устройства с применением иерархического проекта, включающего в себя модули, выполненные с помощью различных способов, эффективно используемых при работе с ПЛИС. Итогом работы является реализация проекта на базе отладочной платы с установленной ПЛИС.

После выполнения этих двух лабораторных работ студент должен получить у преподавателя номер индивидуального задания (список заданий приведен в приложении), при выполнении которого используются полученные навыки.

1 Лабораторная работа №1

Лабораторная работа №1 является вводной, при ее выполнении студент должен создать проект, содержащий принципиальную схему устройства графическом исполнении, произвести моделирование работы устройства и реализовать устройство на базе FPGA. Основной целью этой работы является освоение принципов работы с пакетом ISE и приобретение навыков разработки цифровых устройств на базе ПЛИС.

В качестве первой схемы выберем простое устройство, состоящее из D-триггера со входами асинхронной установки и сброса и конъюнктора, схема которого представлена на рис. 1.

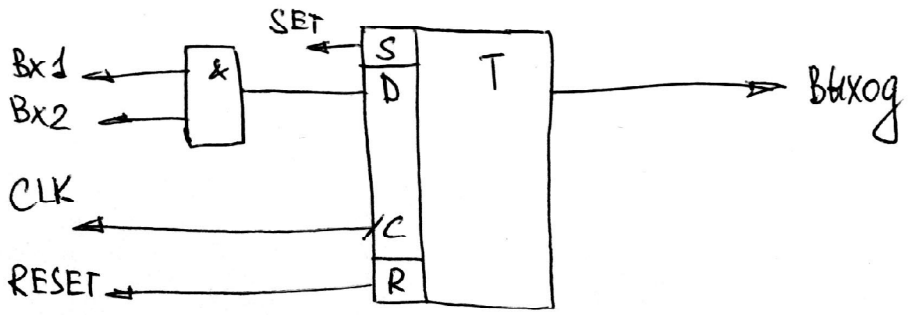


Рисунок 1 – Эскиз принципиальной схемы устройства

1.1 Создание проекта

Запустим навигатор проектов среды ISE (Project Navigator) из меню или с помощью ярлыка на рабочем столе. Обычно при этом автоматически открывается проект, с которым пользователь программы работал в предыдущем сеансе. Закроем этот проект (если, конечно, это чужой проект) командой «File/Close Project». Далее создадим новый проект командой «File/New Project», после чего появится диалоговое окно, в котором нужно указать имя проекта, расположение автоматически создаваемой директории проекта и выбрать тип проекта (HDL при использовании языка описания аппаратуры или Schematic при использовании графического редактора).

Выберем «Schematic». В директории проекта будут размещены необходимые файлы проекта: графические файлы, текстовые файлы проекта на одном из языков программирования, сигнальные файлы и другие. Рекомендуется указать имя, состоящее из латинских букв, во избежание проблем с интерпретацией символов кириллицы в ряде случаев.

По завершении этих манипуляций нажмем «Далее», после чего откроется диалоговое окно, показанное на рис. 2.

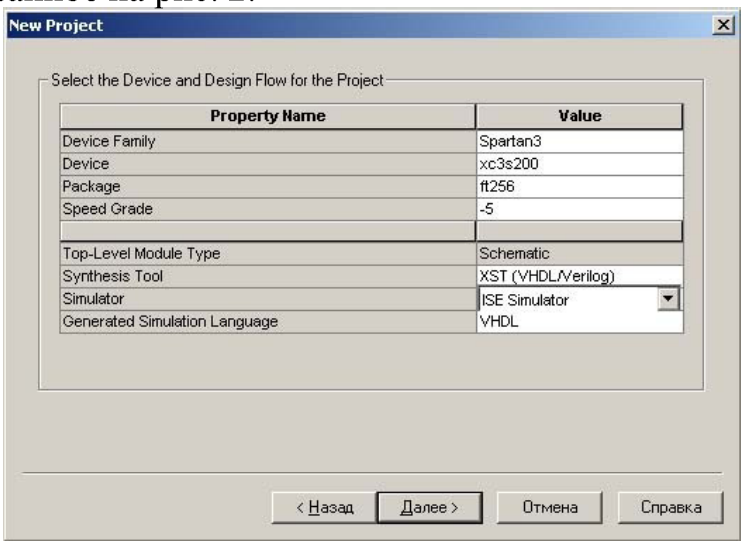


Рисунок 2 Диалоговое окно конфигурации проекта

В этом окне необходимо, как минимум, выбрать семейство и конкретную используемую ПЛИС, а также тип ее корпуса. Выберем FPGA семейства Spartan3, остальные настройки установить согласно рисунку 2. На следующем шаге необходимо указать один вновь создаваемый исходный файл проекта и его тип, при необходимости создания других файлов, их можно добавить позднее. Создадим файл с именем Main типа Schematic (убедитесь, что включена опция «Add to project»). В следующем окне предлагается подключить к проекту уже созданные файлы (например, из других проектов), при этом можно включить опцию их автоматического копирования в директорию проекта. У нас таких файлов пока нет, поэтому просто нажмем «Далее».

По завершению всех описанных шагов откроется окно графического редактора, в котором мы будем создавать наше устройство.

1.2 Создание схемы в графическом редакторе

В левой части окна находится окно «OPTIONS», в котором вы можете выбрать удобный для вас стиль работы с графическим редактором (смысл предлагаемых опции ясен из надписей имеющихся в окне). Рекомендуется поэкспериментировать с выбором предлагаемых особенностей.

При выборе вкладки «Symbols» откроется окно выбора используемых компонентов, смысл которого очевиден. Редактор предлагает ряд удобных особенностей, в частности, фильтр, с помощью которого можно сократить список выбираемых компонентов и возможность просмотра параметров выбираемого компонента «Symbols Info». Обратите внимание на способ реализации компонента в выбранной ПЛИС. В качестве триггера используем элемент «fdcp» из семейства «Flip-flop» и конъюнктор «and2» из семейства «Logic».

Внесите все необходимые компоненты на рабочее поле схемы и соедините их проводниками «Add/wire». Все проводники, которые должны быть подключены к выводам ПЛИС, необходимо снабдить маркерами «Add I/O Marker». После создания схемы целесообразно присвоить созданным маркерам имена в соответствии с назначением выводов устройства, для этого следует щелкнуть правой кнопкой по выбранному маркеру и в контекстном меню выбрать пункт «Rename port». Помните, что нельзя использовать в качестве имен зарезервированные слова, вроде «Integer», «Output», «Out», «In» и др. Если в появляющемся диалоговом окне включена опция «Rename the branch's net», то автоматически будет переименована и подключенная к маркеру цепь. Результат работы должен выглядеть примерно так, как изображено на рис. 3. Сохраните полученный файл перед тем, как перейти к следующему этапу.

После создания схемы целесообразно проверить ее правильность (с точки зрения ISE), для чего выберите процесс «Check Design Rules» из группы «Design Utitlities». Для того, чтобы увидеть процесс и группу, необходимо активировать в окне слева вкладки «Module View» и «Process View». По завершении проверки в нижней части экрана появятся диагностические сообщения либо сообщение о безошибочном завершении проверки.

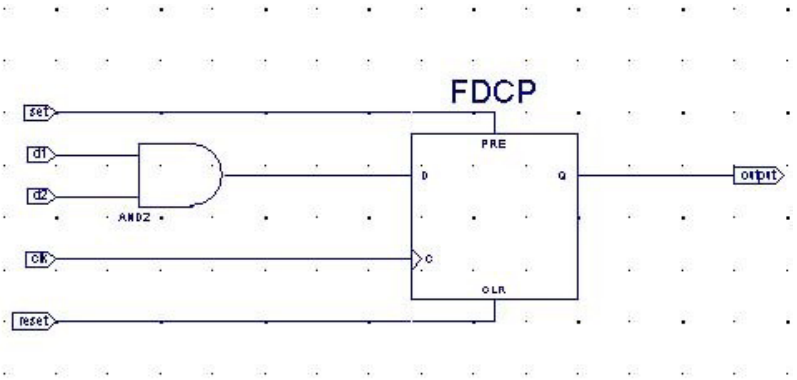


Рисунок 3 – Примерный вид схемы в окне графического редактора

1.3 Моделирование работы устройства

Для моделирования устройства воспользуйтесь командой меню «Project/New source» и выберите из списка «Test bench waveform». Назовем новый файл «Main_test» и нажмем «Далее». Появится диалоговое окно, где нужно указать файл с описанием тестируемого модуля. После этого программа сообщает, что она создает требуемый файл, привязанный к выбранному описанию, и открывает окно, показанное на рис. 4

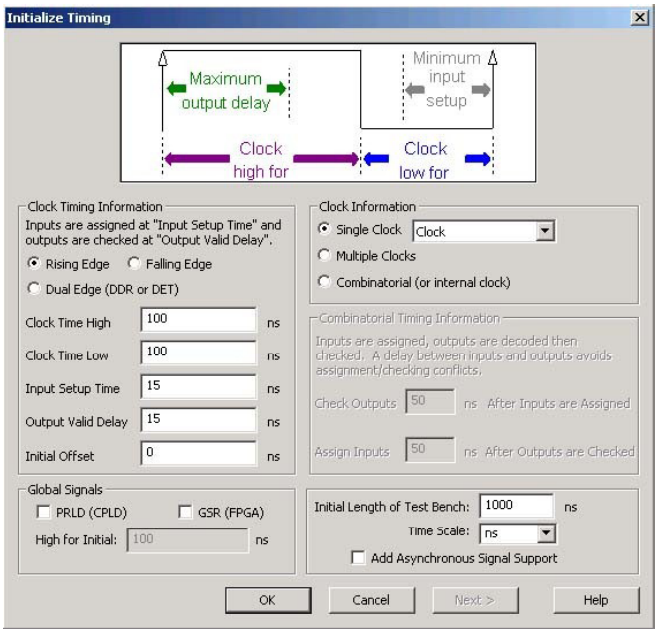


Рисунок 4 – Окно диалога определения параметров тактового сигнала

В этом окне необходимо определить тип и параметры тактового сигнала, используемого для тестирования модуля, обратите внимание также на выбор длительности имитируемого отрезка времени. Тип тактового сигнала определяется блоками «Single clock» и «Rising edge», необходимо также указать, какой сигнал, из определенных в модуле, будет использоваться в качестве тактового. Параметрами тактового сигнала являются длительность высокого и низкого уровней, а также задержки относительно выбранного типа перепада. Для облегчения понимания в верхней части окна имеется графическая подсказка, для более детального ознакомления с настройками воспользуйтесь кнопкой «Help». В окне «Initial Length of Test Bench» укажите величину 10000. (Выбор слишком большой величины замедляет моделирование, слишком малая величина не позволяет подробно анализировать происходящие процессы.)

В нашем случае имеются асинхронные, не привязанные к тактовому, сигналы, для их имитации отметьте блок «Add Asynchronous signal support» и нажмите «Next». В следующем окне нужно выбрать тактовый сигнал (хотя бы один сигнал должен быть обязательно) и далее указать, какие сигналы должны быть привязаны к тактовому сигналу, а какие являются асинхронными. В нашем случае к тактовому сигналу целесообразно привязать только выходной сигнал, остальные подаются асинхронно. Затем нужно указать параметры тактового сигнала (можно оставить параметры, предлагаемые по умолчанию), после чего появится окно с графическим изображением сигналов. Обратите внимание на выделение цветом интервалов времени, соответствующих процессам задержки относительно тактового сигнала.

Отредактированный модуль нужно сохранить, после чего его имя появится в окне проекта. Выделим созданный модуль в окне проекта и в окне «Process View» активизируем процесс «Generate expected simulation results». Проверим соответствие получившихся сигналов нашим ожиданиям. Следует отметить, что результаты моделирования на этом этапе являются оценочными, кроме того, реализация ISE содержит ряд ошибок, в частности, некорректную работу с асинхронными и двунаправленными сигналами. Процесс «Simulate Behavioral Model» обеспечивает поведенческое моделирование устройства, исходя из функциональных особенностей компо-

нентов, и не учитывает особенностей реализации устройства в выбранной ПЛИС. Полное моделирование обеспечивается с помощью «Simulate Post-Place & Route HDL Model». Эту операцию имеет смысл проделать несколько позднее, после того, как мы определимся с реализацией проекта.

1.4 Временные параметры проекта

При создании цифрового устройства на базе ПЛИС разработчик может выдвигать целый ряд требований по быстродействию, энергопотреблению и др. При использовании пакета ISE данные требования могут быть учтены с помощью специальных ограничений («Constraints»), вводимых в процессе описания устройства. Подробно возможные ограничения описаны в //.

При активизации процесса «Create timing constraints» откроется окно редактора временных параметров проекта. На вкладке «Global» можно указать параметры внешнего тактового сигнала (clock). Введите значение 20.0 в поле Period, при необходимости можно активизировать специальное окно, дважды щелкнув мышью на этом поле. При нажатии кнопки «Pad to Pad» откроется диалоговое окно, в котором можно указать максимальную задержку сигналов в проектируемом устройстве при распространения от входа к выходу. Данная возможность актуальна только для комбинационных устройств, так как у нас нет чисто комбинационных путей прохождения между какими-либо контактными площадками, использовать это ограничение бессмысленно.

Далее активизируйте вкладку Port, выделите в списке выходной сигнал, и щелкните правой кнопкой мыши. Выберите в контекстном меню пункт «Clock to Pad» и введите задержку относительно тактового сигнала Clock (10.0нс). Смысл этого ограничения ясен из рисунка: задержка распространения информационного сигнала не должна превышать интервала между активными перепадами тактового сигнала. 10 нс – это половина периода выбранного тактового сигнала.

Аналогично можно указать ограничения на задержку информационного сигнала относительно тактового (Pad to setup). Для этого выделим в списке сигналы d1 и d2, затем введем имя группы в окне «Group Name» и щелкнем левой кнопкой мыши по «Create Group». После этого выберем из списка групп введенное имя и щелкнем по кнопке «Pad to setup». Также зададим здесь 10 нс, при этом информационные сигналы d1 и d2 будут поступать на вход триггера с приемлемым опозданием.

Обратите внимание, что результаты всех проведенных манипуляций отображаются в текстовой форме в нижнем окне редактора. При ошибочном вводе каких-либо ограничений можно удалить соответствующую запись в этом окне и повторить операцию. В дальнейшем можно будет использовать возможность прямого редактирования создаваемого файла в текстовом редакторе.

После внесения всех необходимых изменений сохраните файл и закройте окно редактора.

1.5 Редактирование расположения выводов

Для редактирования расположения выводов активизируйте процесс «Create Area Constraints». При этом откроется окно редактора PACE (Pin-out area constraints), который позволяет управлять процессом размещения проекта на кристалле ПЛИС. Вид главного окна редактора переключается с помощью вкладок, расположенных в нижней части экрана. Выберите вкладку «Package View». На экране возникнет изоб-

ражение выводов выбранной ПЛИС. В левой части экрана имеется список выводов, принадлежащих проекту. Используя метод «drag-and-drop», перетащите требуемые сигналы из этого списка на необходимые выводы ПЛИС. Выберите для тактового сигнала вывод T9 (что соответствует физическому подключению тактового генератора на плате S3), расположение остальных выводов оставим на усмотрение планировщика.

1.6 Синтез проекта

Целью синтеза («Synthesize»)является построение списка цепей с учетом исходных данных (графического изображения схемы, HDL-описания) и введенных ограничений. Полное описание опций процесса синтеза доступно в документации производителя /,/ и подстрочной помощи. Внимания заслуживает опция «Optimization Goal», позволяющая оптимизировать проект с различными приоритетами (скорость или компактность размещения) Для нашего случая целесообразно оставить все опции действующими по умолчанию. Процесс синтеза автоматически выполняется при запуске редактора PACE, поэтому специальных манипуляций с ним не требуется.

1.7 Реализация проекта

Процесс реализации (Implement design) состоит из трех операций, отображаемых в списке процессов: трансляции (Translate), отображения (Map) и размещения на кристалле (Place&Route). Каждый из процессов может выполняться отдельно, или автоматически при запуске операции, требующей в качестве входных данных результатов процесса. Разработчик может влиять на указанные процессы с помощью изменения настроек отдельных процессов и введения ограничений и привязок для всего проекта, а также выполняя отдельные операции вручную.

Процесс трансляции формирует предварительный вариант размещения модулей устройства на кристалле. Результаты работы процесса трансляции можно просмотреть в виде отчета, а также просмотреть предварительное размещение с помощью специального редактора (Floorplan editor). В окне этого редактора имеется список используемых ресурсов ПЛИС и окно планировщика, где можно увидеть размещение этих ресурсов. Используя редактор PACE можно на этом этапе изменить расположение выводов на кристалле FPGA.

Процесс «Map» предназначен для размещения устройства в различных ячейках FPGA. Среди опций процесса размещения следует обратить внимание на пункт «Perform Time-driven Packing and Placement». Включение этой опции означает приоритетное внимание к временным параметрам проекта, что является особенно важным при работе с быстродействующими схемами.

Процесс «Place and Route» предназначен для разводки проекта, т.е. разработки системы связей между отдельными модулями устройства на кристалле ПЛИС. После завершения трассировки целесообразно сформировать отчет (Post-Place & Route Static Timing Report), чтобы проверить соответствие заданным временным ограничениям. Запишите полученные результаты для последующего анализа.

Для просмотра полученного размещения необходимо запустить планировщик (Floorplanner), в окне которого можно не только просмотреть результаты работы трассировщика, но и вручную скорректировать размещение. Попробуйте расположить блоки проекта ближе друг к другу и проверьте, как изменились временные па-

раметры после такой оптимизации (сравните с записанными). Вернитесь к этапу моделирования, запустите процесс «Simulate Post-Place & Route HDL Model». Пользуясь маркерами, определите задержку между фронтом тактового сигнала и изменением сигнала на выходе триггера, а также задержку между подачей асинхронных сброса и установки и изменением тактового сигнала.

Контрольные вопросы к лабораторной работе №1

1. Какие узлы использованы для построения схемы ? Поясните логику их действия.
2. Поясните принцип действия разрабатываемой схемы и докажите по результатам моделирования, что она работает правильно.
3. Какие сигналы целесообразно использовать в качестве тестовых для разрабатываемой схемы ? Все ли возможные сочетания проверены при тестировании ?
4. Какие временные ограничения можно задать для проекта ?
5. Как вручную изменить результаты трассировки автоматического трассировщика ?
6. Какие возможности моделирования разрабатываемого проекта имеются в среде ISE и в чем их различия ?
7. Из каких этапов состоит реализация проекта ? В чем их функции ?
8. Какие ресурсы имеются на кристалле FPGA ?
9. Что такое Look-up Table ?
10. Каким образом разработчик может управлять процессами размещения, трассировки и др. в среде ISE ?
11. Определите, какие выводы назначил планировщик для сигналов Вашего проекта.
12. Какую минимальную задержку между фронтом тактового сигнала и выходным сигналом Вы можете получить в данном проекте ?

2 Лабораторная работа №2

Лабораторная работа №2 предназначена для освоения принципов разработки относительно сложных цифровых устройств на базе ПЛИС. Эффективность работы разработчика и конкурентоспособность разрабатываемого устройства в значительной степени определяются временем от момента получения задания до получения опытного образца, а также гибкостью полученного решения. Применение ПЛИС идеально отвечает упомянутым требованиям. Для сокращения временных затрат на отработку технических деталей, а также экономии материальных средств на создание макета устройства, целесообразно применять так называемые отладочные платы, содержащие ПЛИС и необходимую «обвязку», а также типовые периферийные устройства. После разработки устройства, проведения необходимых испытаний, можно перейти и к разработке печатной платы и конструкции устройства в целом, будучи уже уверенным в работоспособности электрической схемы и ее соответствии условиям технического задания. При выполнении лабораторной работы №2 будем использовать отладочную плату для FPGA Spartan3. Электрические схем и описания отладочных плат содержатся в файлах /,/.

2.1 Стратегия разработки сложных цифровых устройств

При создании сложных устройств полезно использовать иерархическую структуру проекта, которая позволяет распараллелить работу над проектом в целом и отлаживать отдельные узлы, как самостоятельные устройства.

Наше устройство должно реагировать на нажатия кнопки, далее в зависимости от положения переключателя, осуществлять инкремент или декремент счетчика, содержимое которого будем выводить на семисегментный светодиодный индикатор (четыре знакоместа) с использованием динамической индикации. Эскиз структурной схемы устройства представлен на рис. 5. Для простоты на схеме не показаны цепи сигнала сброса, которые целесообразно подключить к автоматам с памятью для корректной инициализации всего устройства.

Блок анти-дребезга контактов обеспечивает подавление лишних импульсов, генерируемых при нажатии кнопки. На входы этого блока подаются: сигнал от кнопки (push button), тактовый сигнал с делителя частоты и сигнал сброса, необходимый для начальной инициализации. Обратите внимание, что кнопки и переключатели показаны условно, практическая реализация на отладочной плате отличается от схемы, при нажатии кнопки и замыкании переключателя (slide switch) активный логический уровень - высокий. Выход схемы, на котором при нажатии кнопки должны появляться одиночные импульсы, подключается ко входу счетчика.

Делитель частоты формирует из входного сигнала (50 МГц) сигналы с частотой, необходимой для работы узла индикации и узла анти-дребезга.

Реверсивный счетчик получает счетные импульсы с выхода блока анти-дребезга и сигнал, определяющий направление счета, от движкового переключателя. Для реверсивного счетчика также необходим сигнал сброса.

Система индикации построена по динамическому принципу, смысл которого состоит в динамическом переключении индикаторов с частотой, незаметной для человеческого глаза (более 100 Гц). При таком способе код символа подается на все индикаторы параллельно, а высвечивается только на нужном индикаторе.

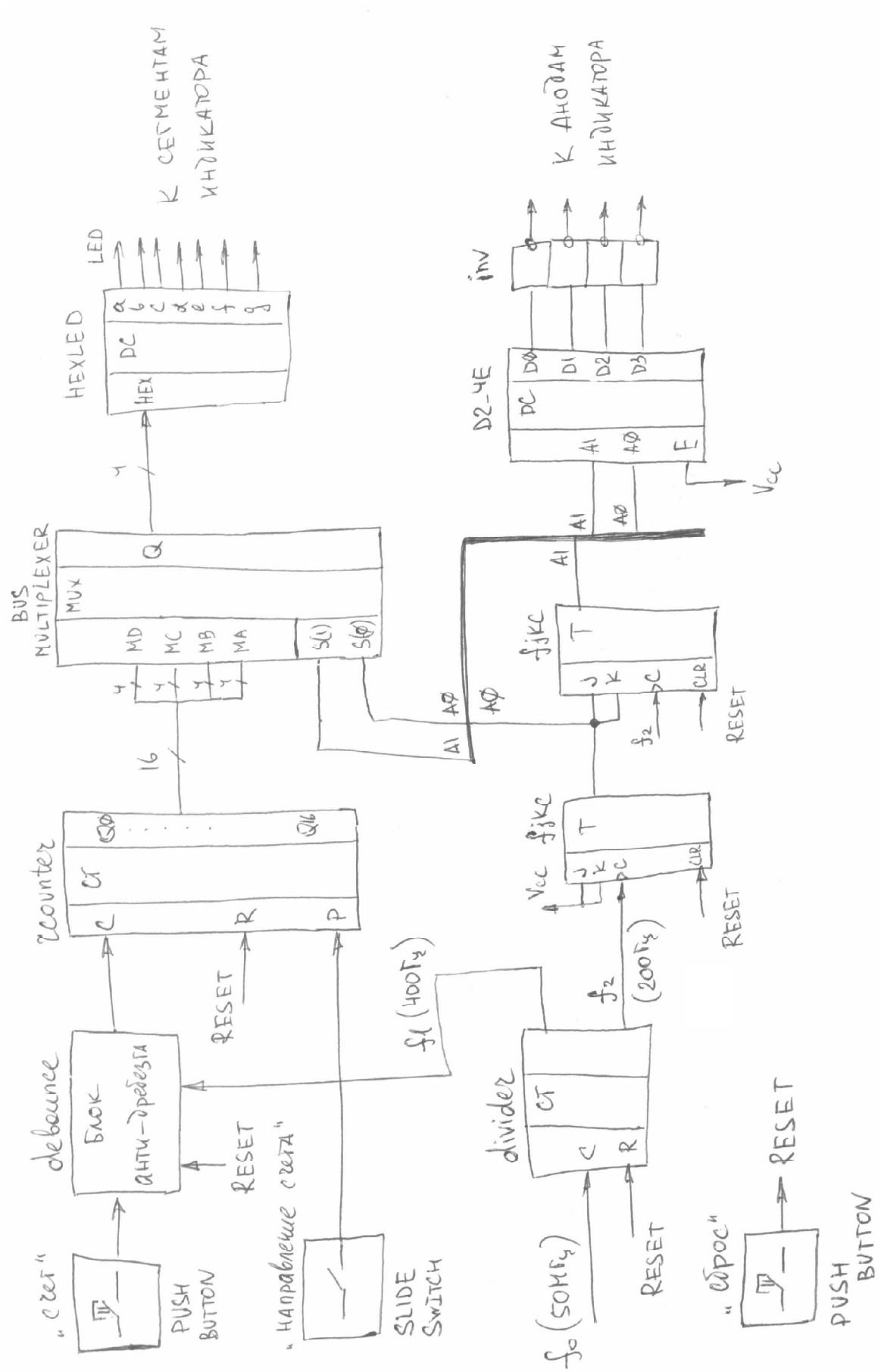


Рисунок 5 – Эскиз разрабатываемой схемы

Таким образом, для включения сегмента одного из индикаторов необходимо установить высокий уровень напряжения на его аноде и низкий – на выводе соответствующего сегмента.

Все одноименные выводы сегментов индикаторов соединены параллельно, и для создания видимости их независимой работы напряжение на аноды подается поочередно. Частота переключения должна быть такой, чтобы глаз наблюдателя не мог заметить мерцания индикаторов (как правило, не менее 100 Гц).

Учитывая послесвечение индикаторов, при синхронном переключении информации и управляющих сигналов, получаем устойчивое изображение на дисплее в целом. Такой способ позволяет сэкономить количество задействованных выводов ПЛИС и существенно упростить топологию печатной платы.

Схема устройства учитывает специфику отладочной платы и поэтому использует имеющиеся в ней узлы. В частности, тактовый сигнал, подаваемый на ПЛИС, имеет фиксированную частоту (50МГц для FPGA Spartan3), а четыре семисегментных индикатора подключены по схеме динамической индикации с общим анодом //.

Для разработки общей схемы устройства будем использовать графический редактор, в котором объединим готовые библиотечные модули ISE и специализированные модули, создаваемые отдельно другими способами.

Создадим проект, аналогично тому, как это было сделано в лабораторной работе №1, также укажем для главного модуля проекта тип Schematic. В окне выбора параметров проекта в качестве симулятора выберем «Modelsim», остальные настройки необходимо установить как показано на рис. 6.

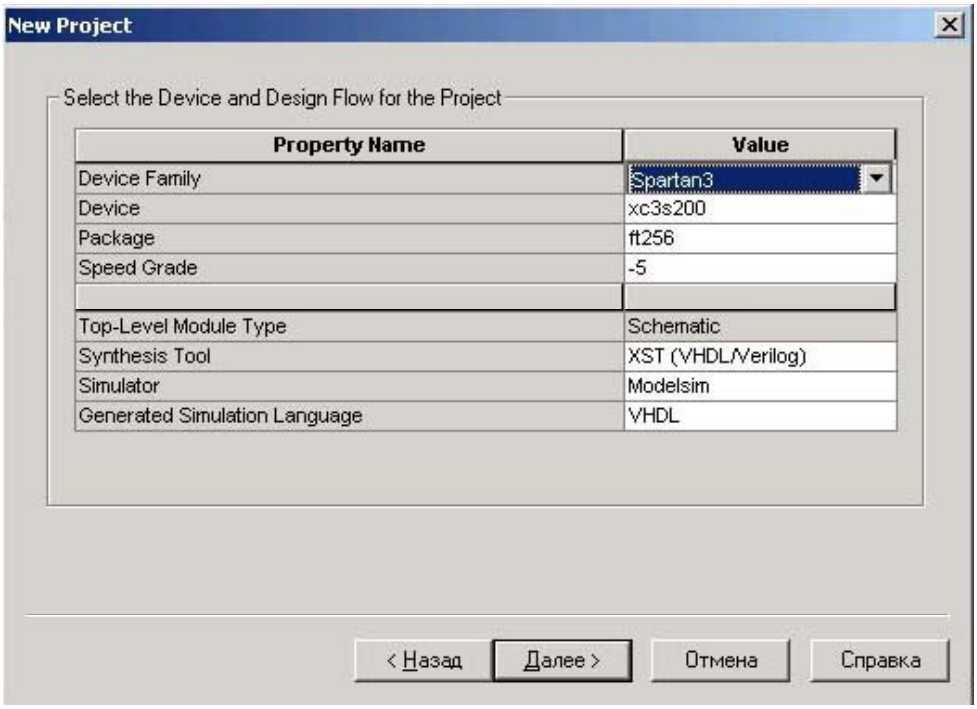


Рисунок 6 Диалоговое окно конфигурации проекта

В окне графического редактора для реализации узла управления индикаторами подключим два JK-триггера (fjkc) и дешифратор (D2_4E) с инверторами (inv). Остальные узлы устройства будем создавать в отдельных модулях.

В отдельных модулях создадим:

— Модуль подавления дребезга контактов (debounce);

— Реверсивный счетчик (rcounter);

— Мультиплексор (mux);
— Делитель частоты (divider);
— Дешифратор семисегментного кода (hexled).

При разработке иерархических проектов, как правило, разработчик сразу полностью представляет себе полную структуру проекта на верхнем уровне иерархии, соответственно количество и типы выводов могут быть заданы еще до разработки архитектуры отдельных узлов.

Все эти модули необходимо добавить в проект. Для этого воспользуемся меню «Project/New source» и выберем в списке тип модуля «VHDL Module». В открывшемся окне можно указать типы выводов модуля и их имена, а также имя архитектуры. Имена выводов и архитектуры логично выбирать из соображений наглядности. Основной принцип – выбор имени должен осуществляться таким образом, чтобы давать представление о функциональном назначении. По завершению процедуры создания файла откроется окно текстового редактора с заготовкой VHDL-описания с пустым разделом архитектуры. Проверьте, что все имена, их тип и направление передачи соответствуют тому, что требуется от модуля

При использовании среды разработки часто используются шаблоны, т.е. «скелеты» наиболее распространенных модулей, которые легко можно модифицировать в соответствии с пожеланиями разработчика. В шаблонах, разумеется, уже имеются имена выводов, в этом случае целесообразно оставить эти имена без изменения, начав разработку с выбора шаблона. В этом случае можно заменить в созданной заготовке раздел entity на приведенный в шаблоне.

2.2 Создание модуля подавления дребезга

Выберите из меню навигатора «Edit/Language Templates», далее из списка выберите «VHDL/Synthesis Constructs/Coding Examples/Misc/Debounce circuits». Воспользуйтесь приведенным шаблоном и включите код в раздел архитектуры (скопировав его через буфер). Проверьте код на отсутствие ошибок, выделив строку «debounce.vhd» в окне «Sources», и щелкнув мышью по строке «Check Syntax» в окне «Processes».

Выделим в окне «Sources» имя вновь созданного модуля, и запустим процесс «Design Utilities/Create Schematic Symbol» в окне «Processes», после чего будет создан схемный символ будущего модуля, который можно включить в графический редактор. Графическое изображение можно редактировать, щелкнув на нем правой кнопкой мыши и выбрав пункт «Edit symbol».

2.3 Создание модуля дешифратора семисегментного кода

Аналогичным образом выберем шаблон дешифратора (7-segment display hex conversion) в том же разделе и скопируем его в раздел описания архитектуры. Следует отметить, что использование библиотеки шаблонов значительно облегчает создание кода, так предлагаемые шаблоны являются проверенными решениями.

2.4 Создание модуля реверсивного счетчика

Так как наш проект носит демонстрационный характер, для создания реверсивного счетчика откажемся от применения готовых решений и воспользуемся языком описания аппаратуры VHDL /„/. Используем поведенческое описание устройства. Счетчик должен инкрементировать свое содержимое или декрементировать (в зависимости от уровня сигнала на управляющем входе.

Создайте пустой модуль rcounter.vhd и наберите в нем следующий текст, используя уже имеющиеся заготовки:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity rcounter is
    Port ( Clk : in std_logic;
           updown : in std_logic;
           reset : in std_logic;
           data : inout std_logic_vector(15 downto 0));
end rcounter;

architecture Behavioral of rcounter is
begin
process (Clk,reset)
begin
  if reset='1' then
     data <= (others => '0');
  elsif Clk='1' and Clk'event then
    if updown ='1' then
          data <= data + 1;
       else
          data <= data - 1;
    end if;
  end if;
end process;
end Behavioral;
```

Сохраните сделанные изменения, проверьте синтаксис (Check Syntax), создайте символ и добавьте его на схему.

2.5 Создание модуля делителя частоты

Для создания модуля делителя частоты воспользуемся структурным стилем описания устройства. В качестве делителя используем синхронный счетчик со сквозным переносом, фрагмент схемы которого представлен на рис. 7. Так как счетчик состоит из одинаковых модулей, с точки зрения создания компактного описания целесообразно использовать конструкцию generate //.

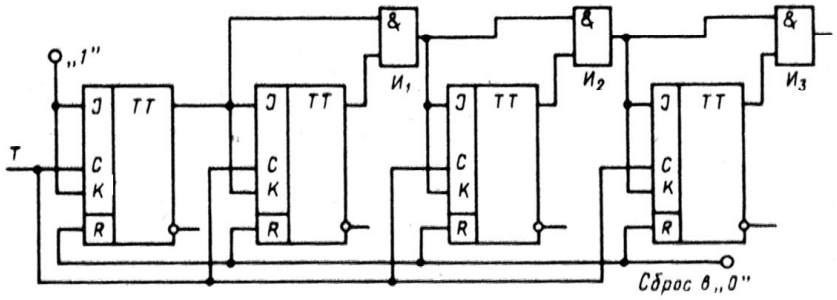


Рисунок 7 - Схема счетчика со сквозным переносом

Создайте файл divider.vhd и создайте в нем описание устройства, показанное ниже.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ANDE is
port (
  X1,X2 : in std_logic;
  Y :out std_logic
);
end ANDE;

architecture ANDA of ANDE is
begin
Y<=X1 and X2;
end ANDA;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity JFF is
port (
  J,C,R : in std_logic;
  Q : inout std_logic
);
end JFF;

architecture JK of JFF is
begin
process (C,R)
begin
  if R='1' then
    Q <= '0';
  elsif (C'event and C='1') then
    if J='1' then Q <= not Q;
          end if;
  end if;
```

```
end process;
end JK;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity divider is
generic (Nd : integer := 24);
   Port ( Clock : in std_logic;
          Clk_led : out std_logic;
          Clk_deb : out std_logic;
          reset : in std_logic);
end divider;

architecture Behavioral of divider is
component JFF
port ( J,C,R : in std_logic;
       Q : inout std_logic);
end component;
component ANDE is
port (
  X1,X2 : in std_logic;
  Y : out std_logic
);
end component;
signal T,V: std_logic_vector(0 to Nd);
begin
T(0)<='1';
Clk_deb<=V(3);
Clk_led<=V(4);
ST0: JFF port map(J=>T(0),C=>Clock,R=>reset,Q=>V(1));
JK1: for i in 1 to Nd-1 generate
    begin
  ST1: ANDE port map(X1=>T(i-1),X2=>V(i),Y=>T(i));
  ST2: JFF port map(J=>T(i),C=>Clock,R=>reset,Q=>V(i+1));
   end generate;
end Behavioral;
```

Обратите внимание на конструкцию generic, позволяющую легко менять разрядность создаваемой структуры, а также на искусственный прием, позволивший компактно описать структуру делителя, несмотря на наличие в ней неоднородных элементов (первый триггер). Выходные сигналы делителя подключены к третьему и четвертому триггерам с целью упрощения контроля работоспособности устройства (небольшой коэффициент деления), в дальнейшем их подключение необходимо скорректировать с учетом требуемого коэффициента деления (необходимые частоты – порядка сотен герц). Сохраните сделанные изменения, добавьте файл в проект и

проверьте синтаксис аналогично тому, как было сделано при создании модуля реверсивного счетчика.

2.6 Создание модуля мультиплексора

Для модуля mux в качестве типа файла (через Project/New source) выберем «IP (CoreGen&Architecture Wizard)» (при работе с отладочной платой с CPLD, а также при работе с «облегченной версией» ISE, такая возможность недоступна и мультиплексор придется реализовать «своими силами» из шаблона VHDL описания или другим способом). Использование IP Core позволяет получить необходимое устройство путем автоматической генерации модуля с требуемыми характеристиками. В появившемся окне выберем «Basic elements/multiplexers/Bus multiplexer», после чего в диалоге создания модуля укажем, что нам требуется 4 входных шины шириной 4 бита. Тип выходного каскада «Non Registered», так как нам требуется чисто комбинационный мультиплексор.

2.7 Создание схемы верхнего уровня

Внесите все необходимые компоненты на рабочее поле схемы и соедините их проводниками «Add/wire». При работе с шинами используйте тот же инструмент, что и для отдельных проводников, разрядность шины будет определена автоматически. При подключении к шине отдельных проводников или шин меньшего размера используется специальный элемент «Add/Bus Tap».

Выводы устройства должны быть снабжены маркерами (команда меню «Add I/O Marker» или кнопка из панели инструментов). Созданным маркерам следует дать осмысленные имена (латинскими символами) для удобства идентификации, при этом соответствующие цепи автоматически получат те же идентификаторы. При желании можно также именовать внутренние цепи («Add Net Name»). При выделении отдельных проводников из созданной шины необходимо указать их имена, например, для шины S имена отдельных проводников могут быть - S(0), S(1). Для переименования можно использовать команду «Rename Selected Net» контекстного меню. Неиспользуемые входы подключим к земле или питанию (стандартные элементы «VCC» и «GND»).

2.8 Отладка устройства

Для отладки всего устройства удобно разбить всю работу на части: сначала отладить работу отдельных модулей, а затем проверить функционирование всего проекта. Для проверки работоспособности на первом этапе будем использовать симулятор ModelSim, который, хотя и не является частью среды разработки, тесно с ней интегрирован. Следует отметить, что наиболее целесообразно отлаживать разрабатываемое устройство в режиме симуляции, а подключение отладочной платы и окончательное тестирование производить только после успешного завершения этапа симуляции.

2.8.1 Отладка модулей устройства

Отладку отдельных модулей проиллюстрируем на примере «самодельного» делителя частоты. Здесь действуем аналогично отладке триггера в лабораторной работе №1, не забывая про наличие асинхронного сигнала сброса. Поведение устрой-

ства исследуем с помощью процесса «Generate expected simulation results». Рекомендуемая тестовая последовательность представлена на рис. 8.

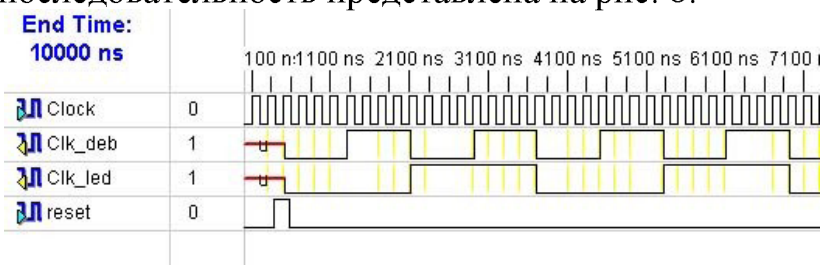


Рисунок 8 – Временные диаграммы тестирования модуля divider

Аналогичным образом можно протестировать и другие модули проекта, хотя, очевидно, что целиком заимствованные узлы сами по себе в тестировании не нуждаются. При тестировании модуля rcounter необходимо иметь в виду, что процесс «Generate expected simulation results» в текущей версии ISE не поддерживает сигналы типа inout, поэтому придется прибегнуть к поведенческому моделированию с помощью Modelsim («Simulate Behavioral Model»).

2.9 Реализация проекта

Реализация проекта в данном случае производится аналогично лабораторной работе №1. Отличия заключаются в необходимости подключения выводов ПЛИС в соответствии с электрической схемой отладочной платы. При работе с CPLD и FPGA имеются некоторые отличия, связанные с архитектурой этих ПЛИС, рекомендации по конкретным деталям содержатся в /,/.

2.10 Работа с отладочной платой

При выполнении лабораторной работы используется отладочная плата S3 с FPGA Spartan-3. На плате имеются необходимые периферийные модули, подключение которых описано в //. Электрические принципиальные схемы платы содержатся в //. При работе с отладочной платой следует помнить, что FPGA не содержат конфигурационного ПЗУ на кристалле и должны программироваться из внешней памяти. Таким образом, при работе с FPGA файл «прошивки» может быть загружен непосредственно в микросхему при включенном питании, либо в конфигурационное ПЗУ (FLASH EEPROM), расположенное на плате. В процессе отладки рекомендуется загружать данные непосредственно в ПЛИС.

Как FPGA, так и CPLD поддерживают различные режимы конфигурации входных/выходных буферов, причем эти режимы задаются программно при компиляции проекта. При выборе режима следует помнить, что все периферийные устройства, имеющиеся на отладочных платах, подключены к напряжению +3.3В, таким образом, не следует выбирать режим с меньшим напряжением. При работе с отладочной платой рекомендуется «оставить все как есть», ничего не указывая в поле «I/O Std.».

2.10.1 Конфигурация FPGA для отладочной платы S3

Для записи конфигурационной информации в микросхему необходимо, чтобы плата была подключена к компьютеру (через параллельный порт), а также подключена к источнику питания (внешний блок питания).

Для записи файла прошивки в FPGA следует выполнить следующие действия:

1. В списке процессов выбрать группу «Generate Programming Files», щелкнуть правой кнопкой мыши и выбрать из контекстного меню «Properties»;
2. В диалоге «Process Properties» выбрать вкладку «Startup Options» и установить значение параметра «FPGA Start-up Clock» - JTAG Clock. Таким образом включается правильный источник синхросигнала для процесса записи конфигурации;
3. Проверить значение параметра «Security» во вкладке «Readback Options», необходимое значение – «Enable Readback and Reconfigure»;
4. Проверить значение параметра «Create Bit File» во вкладке «General Options» - эта опция должна быть отмечена галочкой;
5. Закрыть диалоговое окно и активизировать процесс «Configure Device (Impact)». Если ошибок на предыдущих этапах не обнаружено, то при первом запуске процесса конфигурирования программа предложит указать ряд параметров процесса. В частности в диалоговом окне «Configure Device» нужно отметить пункт «Boundary Scan Mode», а в диалоге «Boundary Scan Mode Selection» отметить пункт «Automatically connect to cable and identify Boundary-Scan chain», после чего откроется окно, вид которого показан на рис. 9. ВНИМАНИЕ ! При наличии ошибок на этом этапе необходимо обратиться к преподавателю;

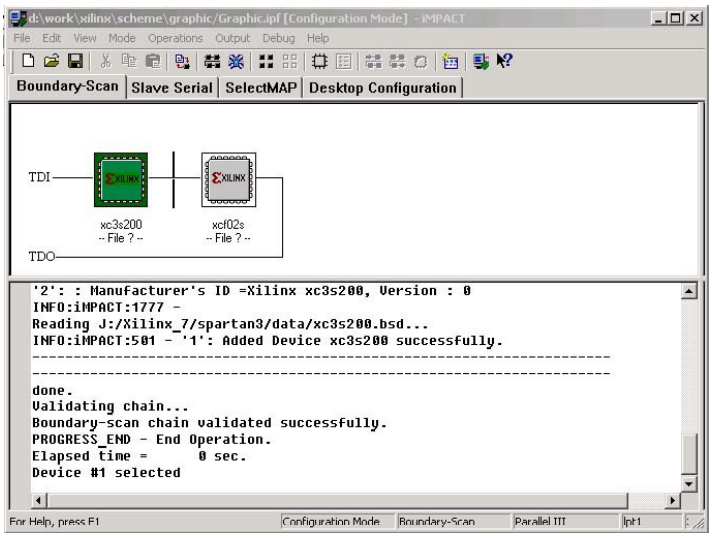


Рисунок 9 – Окно процесса записи конфигурации

6. При первом запуске программа автоматически запросит имя файла конфигурации с расширением «bit», причем запрос будет повторен дважды (для каждого устройства в цепочке). Так как мы собираемся программировать ПЛИС «на лету», откажемся от конфигурирования ПЗУ (xcf02s). Впоследствии с помощью контекстного меню «Assign New Configuration File» можно изменить имя файла конфигурации;
7. В верхней части экрана изображены устройства, программирование которых может производиться посредством интерфейса JTAG. Необходимо выбрать ПЛИС (XC3S200) и активизировать контекстное меню щелчком правой клавиши мыши;
8. В контекстном меню выберем пункт «Program», после чего откроется диалоговое окно программирования. Для ускорения процесса откажемся от проверки

(снять флажок «Verify») и нажмем «ОК». По окончанию процесса программирования конфигурационная информация будет переписана в ПЛИС;

9. При закрытии диалогового окна программа предложит сохранить сделанные настройки с тем, чтобы впоследствии не повторять всех сделанных шагов. Целесообразно подтвердить этот запрос.

Контрольные вопросы к лабораторной работе №2

Как Вы понимаете выражение «иерархическая структура проекта» ?

Для чего служит модуль debounce в разработанной схеме ?

Какие сигналы целесообразно использовать в качестве тестовых для разрабатываемой схемы ?

Чем отличается поведенческий стиль описания устройства от структурного ?

В каких случаях целесообразно использовать конструкцию «generic» ?

Поясните смысл понятия «процесс» в VHDL

Поясните разницу между сигналом и переменной в VHDL

В чем основное отличие FPGA от CPLD ?

Как хранится информация о конфигурации в FPGA и CPLD ?

Что такое интерфейс JTAG, как он используется при разработке цифровых устройств на базе ПЛИС ?

Определите нижнюю и верхнюю границы частоты сигналов clk_deb и clk_led, при которых схема работает удовлетворительно. На что влияют эти частоты ?

Список использованных источников

1. В.В. Соловьев Проектирование цифровых систем на основе ПЛИС. - М.:Горячая линия – Телеком, 2001 – 636с.
2. В.Ю. Зотов Проектирование цифровых устройств на основе ПЛИС фирмы XILINX в САПР WebPack ISE. - М.:Горячая линия – Телеком, 2003 – 624с.
3. И.Е. Тарасов Разработка цифровых устройств на основе ПЛИС XILINX с применением языка VHDL. - М.:Горячая линия – Телеком, 2005 – 252с.
4. П.Н. Бибило Основы языка VHDL. – М.:СОЛОН-Р, 2000 – 200с.
5. Spartan-3 Starter Kit Board User Guide. 2004, Xilinx
6. http://www.digilentinc.com/Data/Products/S3BOARD/S3BOARD-rm.pdf
7. http://www.digilentinc.com/Data/Products/S3BOARD/S3BOARD-sch.pdf
8. http://direct.xilinx.com/bvdocs/publications/ds099.pdf
9. http://direct.xilinx.com/bvdocs/publications/ds090.pdf
10. http://direct.xilinx.com/bvdocs/publications/ds094.pdf
11. http://direct.xilinx.com/bvdocs/publications/DS054.pdf
12. http://direct.xilinx.com/bvdocs/publications/ds057.pdf
13. http://direct.xilinx.com/bvdocs/userguides/ug130.pdf
14. http://www.xilinx.com/support/sw_manuals/xilinx7/download/cgd.zip
15. http://www.xilinx.com/support/sw_manuals/xilinx7/download/dev.zip
16. http://www.xilinx.com/support/sw_manuals/xilinx7/download/qst.zip
17. http://www.xilinx.com/support/sw_manuals/xilinx7/download/sim.zip
18. http://www.xilinx.com/support/sw_manuals/xilinx7/download/hdl.zip

Приложение 1

Список индивидуальных заданий

	Задание
	Бегущая строка (циклический режим) из четырех различных символов (неизменяемых)
	Автомат, выводящий в циклическом режиме на семисегментный индикатор содержимое слов (16бит) памяти (ОЗУ) с адреса 0000 по адрес 00FFH
	Автоматическая индикация последовательности четных чисел с частотой 2Гц на четырехзначном семисегментном индикаторе
	Секундомер с кнопками пуска и останова (4 десятичных цифры на семисегментном индикаторе)
	Автоматическая циклическая индикация последовательности чисел кратных трем с частотой 2Гц на четырехзначном семисегментном индикаторе
	Устройство, суммирующее по нажатию кнопки числа, заданные 8-ю переключателями. Предусмотреть сброс и индикацию суммы на 4-хзначном семисегментном индикаторе
	Автомат «однорукий бандит», управляемый кнопкой «пуск» и четырьмя переключателями, задающими эталон. Индикация цифры на семисегментном индикаторе, частоту и длительность перебора подобрать самостоятельно
	Автомат, формирующий пачку импульсов заданной длины (с индикацией на светодиоде). Предусмотреть задание длины пачки 8-ю переключателями с индикацией на 4-хзначном семисегментном индикаторе
	Устройство, считывающее двухбайтовое число из ОЗУ и отображающее его на семисегментном индикаторе (адрес задается 8-ю переключателями)
	Устройство, считывающее из памяти байт по адресу, заданному 8-ю переключателями и отображающее его в двоичном виде 8-ю светодиодами
	Устройство, отображающее на индикаторе четырехразрядное случайное число (по нажатию кнопки)
	Устройство, переставляющее в случайном порядке цифры числа ABCDH
	Устройство, отображающее 4-хзначное число (в десятичном виде), введенное с клавиатуры
	Устройство «световых эффектов» на 8 светодиодах, автоматически переключающее их по заданной программе с частотой 3Гц
	Часы с индикацией минут и секунд на 4-хзначном семисегментном индикаторе
	Генератор случайных чисел на регистре сдвига с индикацией

		на 4-хзначном семисегментном индикаторе
		Автомат «световых эффектов» на 8 светодиодах, работающий по следующему алгоритму: 1-й диод мигает 1 раз и остается зажженным, 2-й – 2 раза и т.д., после чего цикл повторяется. Частота миганий – 3Гц
		Реверсивный счетчик (с индикацией на 4-хзначном семисегментном индикаторе), определяющий количество щелчков левой и правой кнопками мыши (щелчок левой кнопкой – инкремент, правой – декремент)

Продолжение приложения 1

		Устройство ввода чисел с помощью 3-х кнопок: нажатие 1-й кнопки увеличивает содержимое счетчика на 1, 2-й – на 2, 3-й – на 4. Предусмотреть кнопку сброса и индикацию на 4-хзначном семисегментном индикаторе
		Устройство ввода 2-х байтного числа с помощью 8 переключателей и одной кнопки с индикацией на 4-хзначном семисегментном индикаторе
		Управляемый формирователь меандра (с индикацией на светодиоде) с частотой 1, 2, 3… Гц, задаваемой 8-ю переключателями
		Устройство ввода 4-х разрядного десятичного числа с помощью 4-х кнопок и 2-х переключателей (с индикацией на 4-хзначном семисегментном индикаторе)
		Устройство, индицирующее (на 4-хзначном семисегментном индикаторе) адрес байта в ОЗУ, совпадающего с числом, указанным 8-ю переключателями
		Устройство, считывающее последовательно двухбайтовые числа из памяти с шестнадцатеричной индикацией на 4-хзначном семисегментном индикаторе
		Автомат, отображающий число, указанное переключателями в шестнадцатеричной форме на семисегментном индикаторе и его инверсию в двоичном коде на СИД.
		Десятичная и шестнадцатеричная (режим переключается кнопкой) индикация двоичного числа, заданного 8-ю переключателями
		Устройство, вычисляющее члены ряда арифметической прогрессии (по нажатию кнопки). Предусмотреть сброс и отображение на 4-хзначном семисегментном индикаторе, а также задание разности с помощью 8 переключателей
		Устройство, вычисляющее члены ряда геометрической прогрессии (по нажатию кнопки). Предусмотреть сброс и отображение на 4-хзначном семисегментном индикаторе, а также задание знаменателя с помощью 8 переключателей
		Устройство, считывающее последовательно (частота 2Гц)

		байтовые числа из памяти с шестнадцатеричной индикацией на 4-хзначном семисегментном индикаторе
		Автоматический десятичный счетчик (частота 5Гц) с выводом 4-х цифр результата счета (в шестнадцатеричном виде) на семисегментный индикатор

Требования по выполнению индивидуального задания

Задание выполняется студентом индивидуально, результат оформляется в виде отчета по лабораторной работе. Каждый проект должен быть отлажен в режиме симулятора и, после разрешения преподавателя, проверен с помощью отладочной платы. Рабочее устройство также необходимо продемонстрировать преподавателю.
В отчет должны быть включены:

1. Принципиальная схема всего устройства (при реализации с помощью графического редактора) или VHDL – описание;
2. Результаты моделирования в виде тестовых сигналов и реакции устройства на них;
3. Распечатка отчета, формируемого ISE, с указанием типа и количества используемых ресурсов и т.д.