Министерство образования Республики Беларусь
Учреждение образования
«Полоцкий государственный университет»

В. М. Чертков
С. В. Мальцев
Д. Л. Коц

ВЫЧИСЛИТЕЛЬНЫЕ И МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА

Методические указания
к выполнению курсового проекта
для студентов специальности 1-39 01 01 «Радиотехника»

Новополоцк
ПГУ
2010

УДК 004.9(075.8)

Одобрено и рекомендовано к изданию
методической комиссией радиотехнического факультета
в качестве методических указаний
(протокол № 2 от 24.11.2009)

Кафедра радиоэлектроники

РЕЦЕНЗЕНТЫ:
канд. техн. наук, доц., зав. каф. конструирования РЭС Ю. Г. ГРОЗБЕРГ;
канд. техн. наук, доц., зав. каф. вычислительных сетей Р. П. БОГУШ

ВВЕДЕНИЕ

Дисциплина «Вычислительные и микропроцессорные устройства» занимает ведущее место в подготовке специалистов в области проектирования специализированных электронных вычислительных средств (ЭВС). Цель дисциплины – усвоение фундаментальных основ вычислительной и микропроцессорной техники, основ построения микропроцессорных систем (МПС), получение знаний по вопросам проектирования радиоэлектронных устройств, реализующих цифровые методы управления, формирования и обработки сигналов, а также навыков разработки аппаратных и программных средств МПС.

Интенсивное развитие микроэлектроники привело к созданию различных специализированных микропроцессоров с малым уровнем потребления энергии, широкими функциональными возможностями и низкой стоимостью, которые можно встраивать в системы цифровой обработки сигналов (ЦОС). В результате стало реальным решение задачи выпуска дешевой продукции с учетом индивидуальных интересов потребителей. На базе таких микропроцессорных средств возможно создание различных устройств и приборов – систем промышленной автоматики и управления, обработки аудио- и видеосигналов, связи, устройств автомобильной электроники и т.п. Именно поэтому необходимо готовить таких специалистов, которые умеют применять универсальные и специализированные микропроцессорные средства и разрабатывать для них соответствующее программное обеспечение.

1. ЦЕЛИ И ЗАДАЧИ КУРСОВОГО ПРОЕКТИРОВАНИЯ

Выполнение курсового проекта по дисциплине «Вычислительные и микропроцессорные устройства» имеет следующие цели и задачи:

– закрепить, углубить и систематизировать теоретические знания, полученные ранее при изучении схемотехнических дисциплин;

– приобрести практические навыки самостоятельного решения комплекса задач, связанных с проектированием вычислительных систем с использованием современной элементной базы, путем выполнения самостоятельной творческой разработки по заданному индивидуальному заданию;

– научить пользоваться специальной, справочной и другой нормативно-технической литературой, действующими стандартами;

– подготовить студента к дипломному проектированию и последующей самостоятельной работе по специальности.

2. ОСНОВНЫЕ ТРЕБОВАНИЯ К КУРСОВОМУ ПРОЕКТУ

2.1. Тематика курсового проектирования

Курсовой проект по дисциплине «Вычислительные и микропроцессорные устройства» должен представлять собой самостоятельное решение комплексной инженерно-технической задачи, связанной с проектированием специализированных вычислительных систем для цифровой обработки сигналов и их программного обеспечения. Темы курсовых проектов должны быть посвящены проектированию микропроцессорных систем различного назначения, а также их системного и прикладного программного обеспечения. В качестве курсовых проектов могут выполняться научно-исследовательские работы, связанные с данной тематикой, в рамках госбюджетных и хоздоговорных НИР, заказов предприятий и организаций. Научно-исследовательские проекты должны содержать аналитический материал по решаемой проблеме, теоретический и экспериментальный разделы.

Рекомендуются следующие типовые темы курсового проекта:

1. Разработка генератора сигналов специальной формы с цифровым управлением и индикацией параметров на базе микропроцессора.
2. Разработка цифрового измерителя параметров сигнала.
3. Разработка специализированного устройства для цифровой фильтрации или спектрального анализа.
4. Разработка специализированного устройства ввода и обработки сигналов.
5. Разработка специализированного устройства для компрессии, декомпрессии, кодирования и декодирования данных.
6. Цифровой измеритель параметров физического процесса.
7. Система охранной сигнализации на базе микроконтроллера.
8. Устройство управления аппаратом или прибором.

2.2. Задание на курсовое проектирование и исходные данные

Индивидуальное задание на курсовое проектирование выдается студенту в течение первой недели текущего семестра преподавателем – руководителем проекта. Задание оформляется на специальном бланке. Заданием на курсовое проектирование предусматривается разработка структурной организации специализированной вычислительной системы, разработка алгоритма работы, функциональной схемы вычислительной системы, ее программного обеспечения. В задание включаются:

1) тема проекта;
2) сроки сдачи студентом законченного проекта;
3) исходные данные к проекту;
4) содержание пояснительной записки (перечень вопросов, подлежащих разработке);
5) перечень графического материала;
6) календарный график работы над проектом.

В качестве исходных данных для выполнения курсового проекта задаются:

– назначение разрабатываемого устройства;
– требования пользователя (выполняемые функции, устройства ввода/вывода, быстродействие, частотный диапазон, разрядность, погрешность, параметры входных и выходных сигналов и т.п.).

2.3. Содержание и объем курсового проекта

Курсовой проект должен содержать: проектировочную часть, помещаемую в пояснительной записке, и графическую часть, оформленную в виде комплекта схем. При изложении материала в пояснительной записке рекомендуется придерживаться следующего расположения разделов:

– титульный лист;
– задание на курсовое проектирование;
– содержание;
– введение;
– анализ задачи;
– проектирование аппаратных средств системы;
– проектирование программного обеспечения;
– выводы и заключение;
– список литературы;
– приложения.

Общий объем пояснительной записки должен составлять 30 – 40 страниц.

2.4. Организация курсового проектирования и защита проекта

На выполнение курсового проекта по дисциплине «Вычислительные и микропроцессорные устройства» отводится примерно 13 учебных недель. Работа над курсовым проектом является самостоятельной работой

студента, проводимой под контролем руководителя проектирования. В установленное расписанием время студент консультируется у своего руководителя. Объем выполненной студентом работы по каждому этапу оценивается руководителем проектирования в процентах от общего объема проектирования. **Явка студентов к преподавателю для контроля за выполнением курсового проекта строго обязательна.** Оформленный курсовой проект сдается студентом руководителю на проверку не позднее, чем за неделю до назначенного срока защиты, и после проверки может быть представлен к защите.

Защита курсового проекта проводится перед комиссией и включает в себя доклад в течение 4 – 6 минут и ответы на вопросы членов комиссии. В докладе студент должен изложить задачи проектирования и способ их реализации, дать обоснование принятых технических решений, кратко охарактеризовать каждый лист графического материала. Принятые в курсовом проекте технические решения должны сопровождаться выводами. В заключении студент должен отразить степень соответствия разработанного проекта требованиям задания на проектирование. Студенту на защите могут задаваться любые вопросы по теме курсового проекта. По выступлению студента и его ответам на вопросы комиссия судит о его способностях правильно и доходчиво излагать результаты своей работы, поэтому выступление рекомендуется подготовить заранее. Общая продолжительность защиты с учетом ответов на вопросы составляет 10 – 15 минут.

3. МИКРОПРОССОРНЫЙ КОМПЛЕКС TMS320VC5510 ДЛЯ ВЫПОЛНЕНИЯ КУРСОВОГО ПРОЕКТА

3.1. Структура процессора

В курсовом проектировании используется цифровой сигнальный процессор (ЦСП) с фиксированной точкой TMS320VC5510/5510A, базирующийся на ядре TMS320C55x. Это связано с тем, что данная архитектура обладает высокой производительностью при низком энергопотреблении благодаря повышению параллелизма и уделению особого внимания вопросам снижения потребляемой мощности. ЦПУ обладает внутренней шинной структурой, состоящей из программной шины, трех шин чтения данных, двух шин записи данных и вспомогательных шин для периферии и контроллера прямого доступа к памяти (DMA). Это позволяет выполнять до трех операций чтения данных и двух операций записи данных за один цикл, при этом контроллер DMA может осуществить до двух операций перемещения данных без задействования ЦПУ.

Ядро ЦСП C55x обладает двумя модулями умножения-накопления (MAC), каждый из которых способен выполнять операции типа «умножение 17-бит x 17-бит» за один цикл. Центральное 40-битное арифметическо-логическое устройство (АЛУ) сопровождается вспомогательным 16-битным АЛУ. Алгоритм совместного использования двух АЛУ определяется набором инструкций, обеспечивая оптимальную параллельную работу и снижение энергопотребления. Распределение ресурсов возложено на адресное устройство (АУ) и устройство данных (УД) ядра ЦСП C55x.

Семейством цифровых сигнальных процессоров C55x поддерживаются инструкции с переменным числом байт, что позволяет увеличить плотность кода. Модуль инструкций (МИ) осуществляет 32-битную выборку инструкций из внутренней либо внешней памяти и определяет очередь инструкций для программного модуля (ПМ). В свою очередь, ПМ декодирует инструкции, определяет задачи для АУ и УД и управляет защищенным конвейером. Для предотвращения переполнения конвейера при выполнении условных переходов используется предсказание переходов. Процессоры 5510/5510A также имеют встроенный кэш инструкций размером 24 кбайт для снижения числа операций доступа к внешней памяти, повышения производительности и снижения энергопотребления.

Набор периферийных устройств процессоров 5510/5510A включает в себя интерфейс внешней памяти (EMIF), обеспечивающий непосредственное подключение различных типов асинхронной памяти, таких, как EPROM и SRAM, а также высокоскоростных запоминающих устройств высокой плотности, таких, как синхронные DRAM и SRAM с пакетной выборкой. Три полнодуплексных многоканальных буферизованных последовательных порта (McBSP) обеспечивают непосредственное подключение большого ряда устройств со стандартными последовательными интерфейсами и многоканальный обмен (до 128 каналов с индивидуальным запретом). Порт управляющего контроллера (EHPI) представляет собой 16-битный параллельный интерфейс, обеспечивающий внешнему управляющему процессору доступ к встроенной памяти процессоров 5510/5510A, и может быть сконфигурирован как в мультиплексный, так и в немультиплексный режим, что позволяет использовать его совместно с самыми различными управляющими процессорами. Контроллер прямого доступа к памяти (DMA) обеспечивает перемещение данных по шести независимым каналам без вмешательства ЦПУ, его суммарная пропускная способность составляет до двух 16-битных слов за цикл. Кроме этого, набор периферийных устройств содержит еще два таймера общего назначения, восемь выводов общего назначения (GPIO) и генератор с ФАПЧ и цифровым управлением (DPLL).

Процессоры 5510/5510A поддерживаются программным обеспечением eXpressDSP™, которое состоит из интегрированной среды разработки (IDE) Code Composer Studio™, РТОС DSP/BIOS™ и документации TMS320™ DSP Algorithm Standard. Вообще, существует большое количество разработок сторонних производителей. IDE Code Composer Studio состоит из компилятора языка «С», линкера Visual Linker, симулятора, поддержки обмена данными в реальном режиме времени Real-Time Data Exchange (RTDX™), драйверов эмулятора XDS510™ и библиотек Chip Support Libraries (CSL). DSP/BIOS представляет собой расширяемое модульное программное обеспечение, бесплатно доступное пользователям цифровых сигнальных процессоров фирмы Texas Instruments и состоящее в основном из планировщика задач и средств поддержки систем реального времени с весьма экономичным расходованием памяти и других ресурсов процессора. TMS320 DSP Algorithm Standard – это набор спецификаций, выполнение которых обеспечивает совместимость кода, созданного различными разработчиками, что значительно облегчает его дальнейшую интеграцию в проекты. Сеть сторонних разработчиков фирмы Texas Instruments объединяет более 400 участников, предлагающих пользователям массу готовых комплексных и компетентных решений.

Ядро процессоров TMS320C55x базируется на открытой архитектуре с добавлением специфических модулей, позволяющих значительно ускорить обработку некоторых алгоритмов. Данные модули обеспечивают процессорам 5510/5510A оптимальное соотношение производительности, присущей процессорам с фиксированной точкой, низкого энергопотребления и цены, что является уникальным сочетанием на рынке видеопроцессоров. Дополнительные модули обеспечивают процессорам 5510/5510A высокую производительность видеокодека, в то же время оставляя незадействованными более половины ресурсов ядра, что позволяет реализовывать параллельно такие функции, как преобразование цветовых схем, организация пользовательского интерфейса, функции безопасности, стек TCP/IP, распознавание голоса, преобразование текста в голос и т.д. В результате цифровой сигнальный процессор 5510/5510A способен в одиночку решить практически все задачи, стоящие перед системами фильтрации.

3.2. Отличительные особенности микропроцессорного комплекса

1. Высокопроизводительное ядро для операций с фиксированной точкой с низким энергопотреблением TMS320C55™, внешний вид которого представлен на рис. 1:

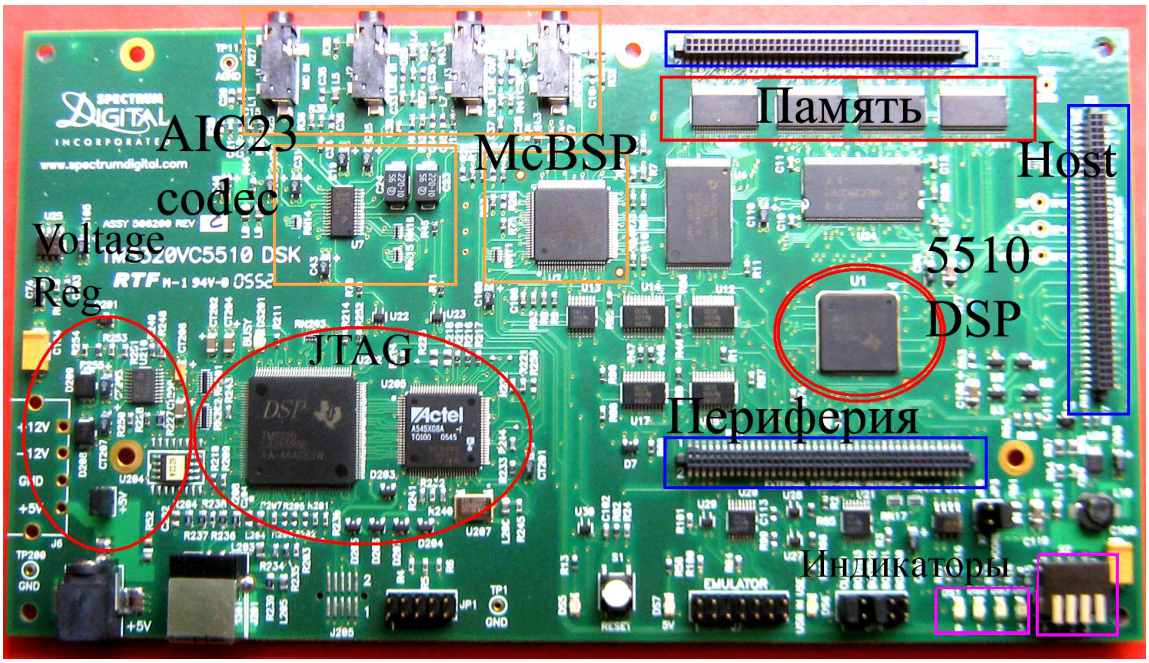

Рис. 1. **Внешний вид TMS320VC5510**

а) время цикла 6,25 / 5 нс;
б) тактовая частота 160 / 200 МГц;
в) выполнение одной или двух операций за такт;
г) два умножителя (производительность до 400 млн умножений с накоплением в секунду (MMACS));
д) два арифметическо-логических устройства (АЛУ);
е) внутренняя программная шина;
ж) три внутренних шины чтения данных/операндов;
з) две внутренних шины записи данных/операндов.

2. Кэш инструкций (24 кбайта).
3. Встроенное ОЗУ 160 кбайт x 16 бит, состоящее:
   а) из 8 блоков по 4 кбайт x 16 бит двухпортовой ОЗУ (DARAM);
   б) 32 блоков по 4 кбайт x 16 бит однопортовой ОЗУ (SARAM).
4. Встроенное ПЗУ 16 кбайт x 16 бит (32 кбайт).
5. Общий объем адресуемой памяти 8 Мбайт x 16 бит.
6. 32-битный интерфейс внешней памяти (EMIF) с возможностью подключения памяти типа:
   а) асинхронное статическое ОЗУ (SRAM);
   б) асинхронное EEPROM;
   в) синхронное динамическое ОЗУ (SDRAM);
   г) синхронное статическое ОЗУ с пакетной выборкой (SBSRAM).

7. Программный контроль энергопотребления шести функциональных блоков из внутренних устройств.

8. Встроенные периферийные устройства:
   а) два 20-битных таймера;
   б) шестиканальный контроллер прямого доступа к памяти (DMA);
   в) три многоканальных буферизованных последовательных порта (McBSP);
   г) 16-битный параллельный порт управляющего контроллера (EHPI);
   д) программируемый тактовый генератор с ФАПЧ;
   е) восемь портов ввода-вывода общего назначения (GPIO) и выход общего назначения (XF);
   ж) встроенный эмулятор;
   з) поддержка периферийного сканирования по стандарту JTAG[1].

9. Варианты корпусов:
   а) 240-выводный корпус MicroStar BGA™ (суффикс GCW);
   б) 240-выводный корпус MicroStar BGA™ (суффикс ZGW) без содержания свинца (Pb-free).

10. Питание ядра 1,6 В.

11. Питание портов ввода-вывода 3,3 В.

На рис. 2 представлена блок-схема процессора TMS320VC5510.

Память процессора состоит из адресов регистров и адресов данных программы, содержащей константы и тело программы. Обращение к памяти процессора может осуществляться по адресам. Аппаратно доступ к блокам, описанным в табл. 1, может осуществляться параллельно, т.е. два доступа за один цикл (два чтения, две записи или чтение и запись). DRAM – *Dual-Access RAM (Память с двойным доступом).*

Таблица 1

Разбиение на блоки областей памяти с двойным доступом

Шестнадцатеричный адрес	Память
000000h − 001 FFFh	DARAM 0
002000h − 003FFFh	DARAM 1
004000h − 005FFFh	DARAM 2
006000h − 007FFFh	DARAM 3

Внутри кристалла также имеются области памяти с одиночным доступом (SDRAM).

[1] Стандарт IEEE 1149.1-1990.

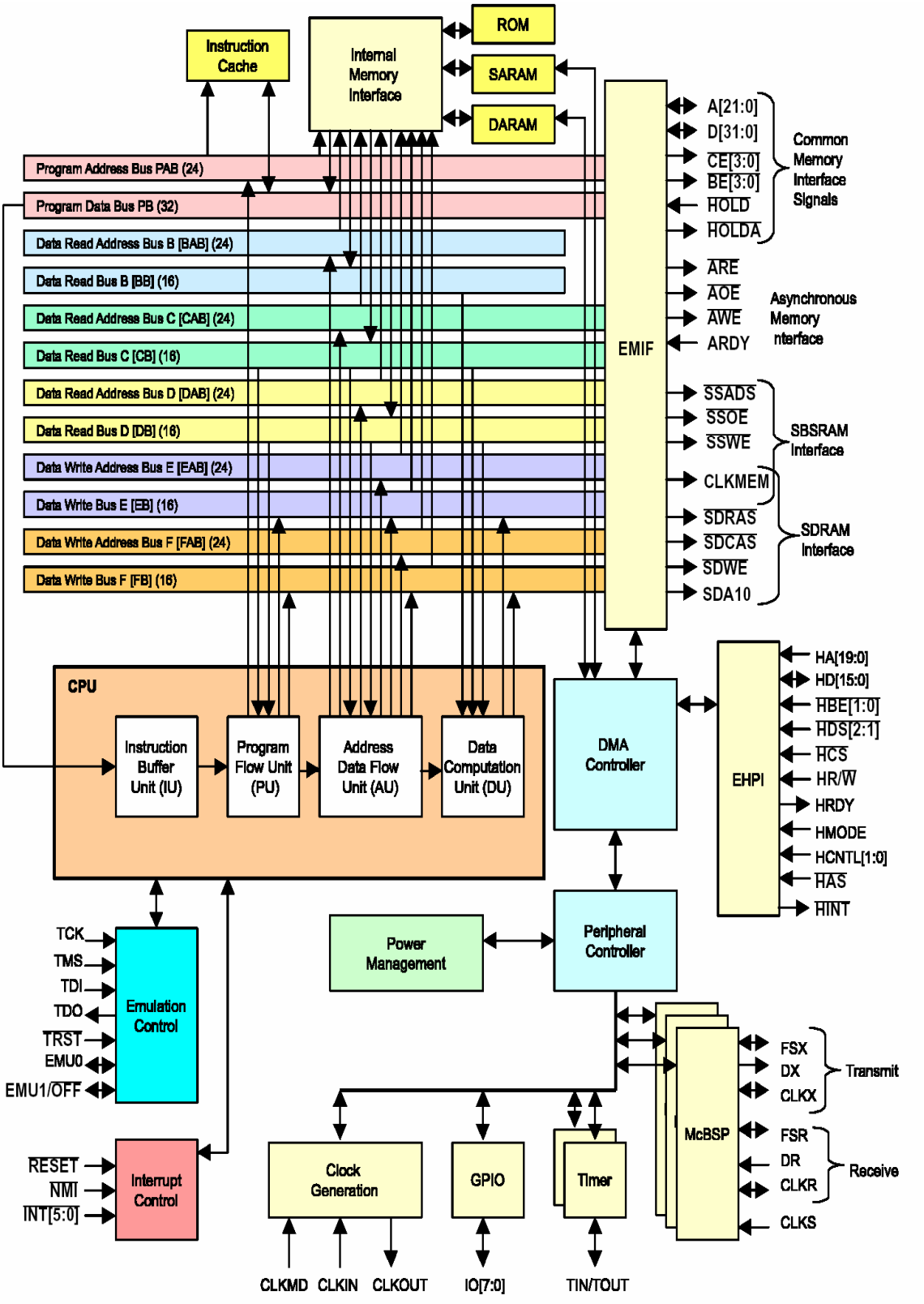

Рис. 2. **Блок-схема внутрикристалльных компонентов с функциональными выводами**

В табл. 2 обозначено деление этих областей по адресам.

Таблица 2

Области памяти с одиночным доступом

Шестнадцатеричный адрес	Область памяти	Шестнадцатеричный адрес	Область памяти
010000h−011 FFFh	SARAM 0	030000h−031 FFFh	SARAM 16
012000h−013FFFh	SARAM 1	032000h − 033FFFh	SARAM 17
014000h−015FFFh	SARAM 2	034000h − 035FFFh	SARAM 18
016000h−017FFFh	SARAM 3	036000h − 037FFFh	SARAM 19
018000h−019FFFh	SARAM 4	038000h − 039FFFh	SARAM 20
01A000h−01BFFFh	SARAM 5	03A000h − 03BFFFh	SARAM 21
01C000h−01DFFFh	SARAM 6	03C000h − 03DFFFh	SARAM 22
01E000h−01FFFFh	SARAM 7	03E000h − 03FFFFh	SARAM 23
020000h − 021 FFFh	SARAM 8	040000h − 041 FFFh	SARAM 24
022000h − 023FFFh	SARAM 9	042000h − 043FFFh	SARAM 25
024000h − 025FFFh	SARAM 10	044000h − 045FFFh	SARAM 26
026000h − 027FFFh	SARAM 11	046000h − 047FFFh	SARAM 27
028000h − 029FFFh	SARAM 12	048000h − 049FFFh	SARAM 28
02A000h − 02BFFFh	SARAM 13	04A000h − 04BFFFh	SARAM 29
02C000h − 02DFFFh	SARAM 14	04C000h − 04DFFFh	SARAM 30
02E000h − 02FFFFh	SARAM 15	04E000h − 04FFFFh	SARAM 31

Регистры управления процессора:

SYSR – регистр системы обеспечивает управление над определенными для устройства функциями. SYSR расположен в 07FDh (адреса порта). В табл. 3 представлено его побитовое представление.

Таблица 3

Системный регистр (побитное представление)

Резерв	HPE	BH	HBH	BOOTM 3	Резерв	Резерв	Резерв	CLKDIV
15 10	9	8	7	6	5	4	3	2 0
R-0000	R/W-1	R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-000

R – чтение; *W* – запись; *n* – значение после рестарта

Расшифровка битов системного регистра представлен в прил. А. Регистры процессора **VC5510** – прил. Б.

DMA_CCR – регистр контроля прямого доступа к памяти позволяет синхронизировать каналы передачи данных по отобранным событиям. Поддерживается 14 отдельных синхронизирующих событий, и каждый канал может быть привязан, чтобы отделить синхронизирующие события, независимые от других каналов. Синхронизирующие события могут быть

отобраны путем программирования области SYNC в определенном для канала регистре контроля – канала прямого доступа к памяти (DMA_CCR).

EHPI – регистр управления обеспечивает 16-битный параллельный интерфейс удаленного управления со следующими особенностями:

1) 20-битовую шину адреса хоста;
2) 16-битовую шину данных хоста;
3) мультиплексированные и демультиплексированные способы передачи по шинам;
4) доступ хоста к памяти SARAM, DARAM и внешней памяти;
5) 20-битовый регистр адреса (в мультиплексном способе) со способностью автоприращения к более быстрым передачам;
6) HRDY используют для подтверждения связи с ведущим.

IODATA, IODIR − регистры управления портами ввода/вывода интерфейса GPIO. Этот интерфейс обеспечивает восемь точек ввода/вывода общего назначения IO0−IO7. Каждая точка может независимо формироваться как ввод или вывод, используя регистр руководства ввода/вывода (IODIR). Регистр данных ввода/вывода (IODATA) используется для контроля логического состояния точек, формируемых как входы, и управления логическим состоянием точек, формируемых как выходы. IODIR и IODATA доступны для центрального процессора и диспетчера прямого доступа к памяти в адресах в месте ввода/вывода.

3.3. Реализация базовых процедур ЦОС на TMS 320VC55

Фирма Texas Instruments предлагает пользователям ЦСП 5510/5510A программное обеспечение C55x DSP Library (DSPLIB), представляющее собой набор из более 50 подпрограмм, вызываемых из программ на языке «С» и реализующих типовые задачи цифровой обработки сигнала, таких, как БИХ/КИХ фильтры, БПФ и многие другие. Библиотека обработки изображений DSP Image/Video Processing Library (IMGLIB) содержит более 20 подпрограмм, оптимизированных для ядра C55x и скомпилированных с помощью последней версии ПО для ЦСП C55x. Библиотека включает в себя стандартные функции обработки изображений, такие, как сжатие, обработка видеосигнала, машинное зрение и медицинские задачи обработки изображений.

Для инициализации базовых функций необходимо подключить следующие компоненты:

1) файл заголовка для среды программирования «С»: *dsplib.h*;

2) библиотеку объектов: *55xdsp.lib*;

3) библиотеку объектов: *TMS320.lib*;

4) исходную библиотеку для осуществления конечных настроек: *55xdsp.src*;

5) необходимо описать области памяти программы в файле с расширением *.cmd.*

В прил. В приведен список базовых процедур, описанных разработчиками.

Благодаря такому широкому выбору базовых функций даже при большой вычислительной сложности практически любая задача легко решается с высоким быстродействием.

3.4. Реализация процедуры фильтрации на TMS 320VC5510

Из исходных данных курсового задания следует, что необходимо реализовать фильтр Чебышева со следующими параметрами: частота дискретизации 44 кГц; полоса пропускания 3 400 Гц.

Самый простой и надежный способ заключается в преобразовании входных отсчетов во временной области, т.е. реализация КИХ-фильтра. Для этого необходимо иметь входные отсчеты и коэффициенты фильтра. Т.к фильтр будет с КИХ, то это в свою очередь предполагает самый простой вариант – нерекурсивный фильтр.

Для расчета коэффициентов фильтра воспользуемся программой Matlab. Расчет удобно выполнять с помощью графической программы (GUI)sptool, входящей в библиотеку *signal*. Загрузим эту программу с помощью инструкции *sptool*, на появившейся панели нажмем кнопку *New Design*, находящуюся под окном *Filters*. Вид появившегося окна – на рис. 3.

После этого на экране появится панель *Filter Designer*. Чтобы вычислить фильтр, необходимо ввести данные в поля *Sampling Frequency*: *Fp* (верхнее граничное значение полосы пропускания); *Rp* (максимально допустимое значение пульсации в полосе пропускания); *Fs* (нижнее граничное значение полосы задерживания); *Rs* (максимально допустимое значение пульсации в полосе задерживания). Необходимо выбрать параметр расчета с наименьшим числом коэффициентов. В поле типа фильтра выбираем фильтр Чебышева – именно этот фильтр дает максимально лучшие результаты. Вид окна разработки фильтра с введенными параметрами представлен на рис. 4.

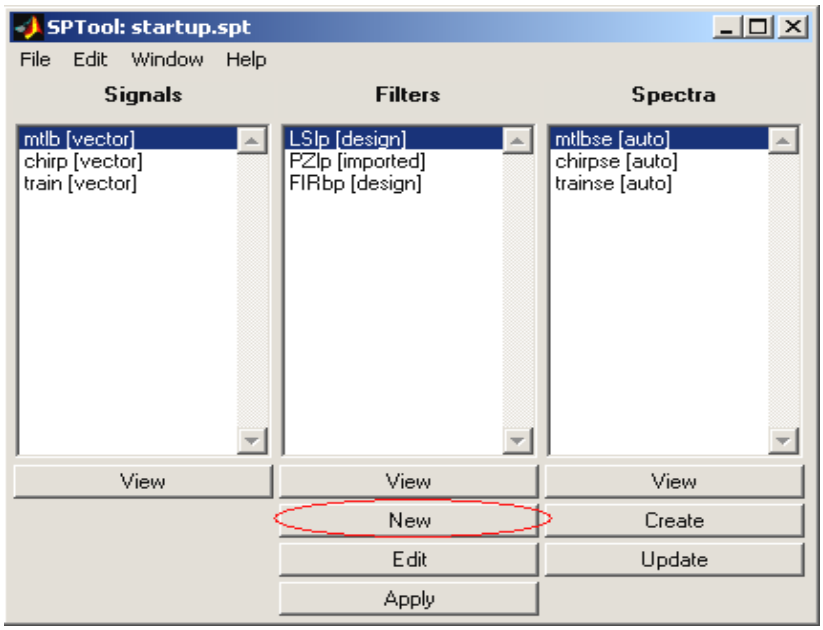

Рис. 3. **Меню GUI**

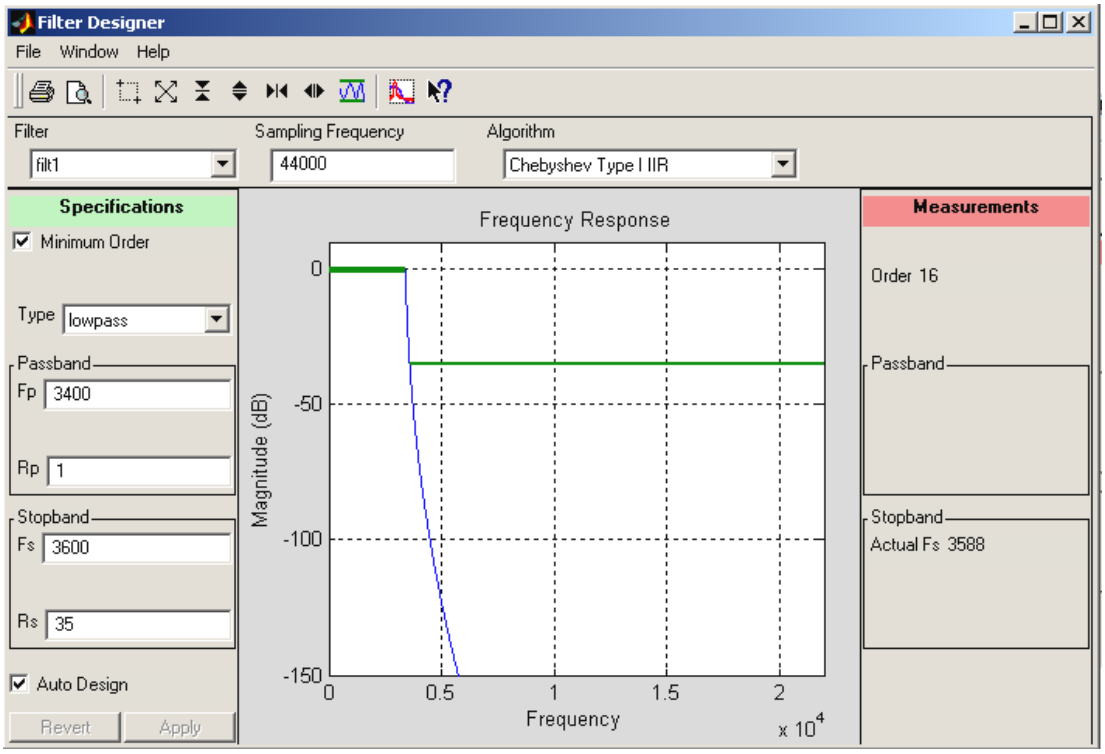

Рис. 4. **Окно разработки фильтра**

Чтобы коэффициенты спроектированного фильтра стали доступны для дальнейшего использования, их надо экспортировать в рабочее пространство Matlab. Для этого необходимо открыть меню *File* главной панели *sptool* и выбрать раздел *Export*… Если экспортированная структура

имеет имя filt1, извлечь соответствующие коэффициенты можно с помощью команды b1=filt1.tf.num;.

Полученные коэффициенты:

```
0, 0.0005,
0.0034, 0.0149,
0.0446, 0.0982,
0.1637, 0.2104,
0.2104, 0.1637,
0.0982, 0.0446,
0.0149, 0.0034,
0.0005, 0
```

При цифровой фильтрации необходимо реализовать алгоритм обработки сигнала. Алгоритм программы написан на языке программирования «С» с комментариями и представлен в прил. Г.

3.5. Правила написания и отладки приложений в программной среде Code Composer Studio

Основная задача радиоинженера, работающего с цифровыми сигнальными процессорами, – быстрое и качественное описание программы на языке низкого уровня. В большинстве случаев сам код программы представляет собой набор индивидуальных команд для каждого типа и даже вида процессоров. Работа с этими командами достаточно сложна. Для удобства программистов были разработаны адаптивные оболочки, которые могут переводить команды процессора в язык более высокого уровня, а также моделировать сам процессор. Имея программу эмулятора процессора и адаптивную оболочку, при наличии компьютера возможно написание и отладка кода программы. Записанный код программы также может переводиться обратно в команды процессора и возможна запись этого кода в память процессорного блока для получения автономного микропроцессорного блока, выполняющего заложенный программой алгоритм.

Рассмотрим работу программы *Code Composer Studio 3.1*.

Для начала работы необходимо настроить саму программу – указать, какой вид процессора далее будет использоваться, и подключить его библиотеки команд.

Для этого, после установки, необходимо запустить *Code Composer Studio Setup (Setup CCStudio v3.1)* на рабочем столе. На рис. 5 представлен внешний вид меню. Для реализации курсового проекта необходимо доба-

вить *C5510 Device Simulator*. В меню, изображенном на рис. 6, подтверждаем нужный нам вид микропроцессорного блока двойным щелчком. Появляется интерфейс самого симулятора (рис. 7), в котором возможно написание непосредственно кода программы и его отладка.

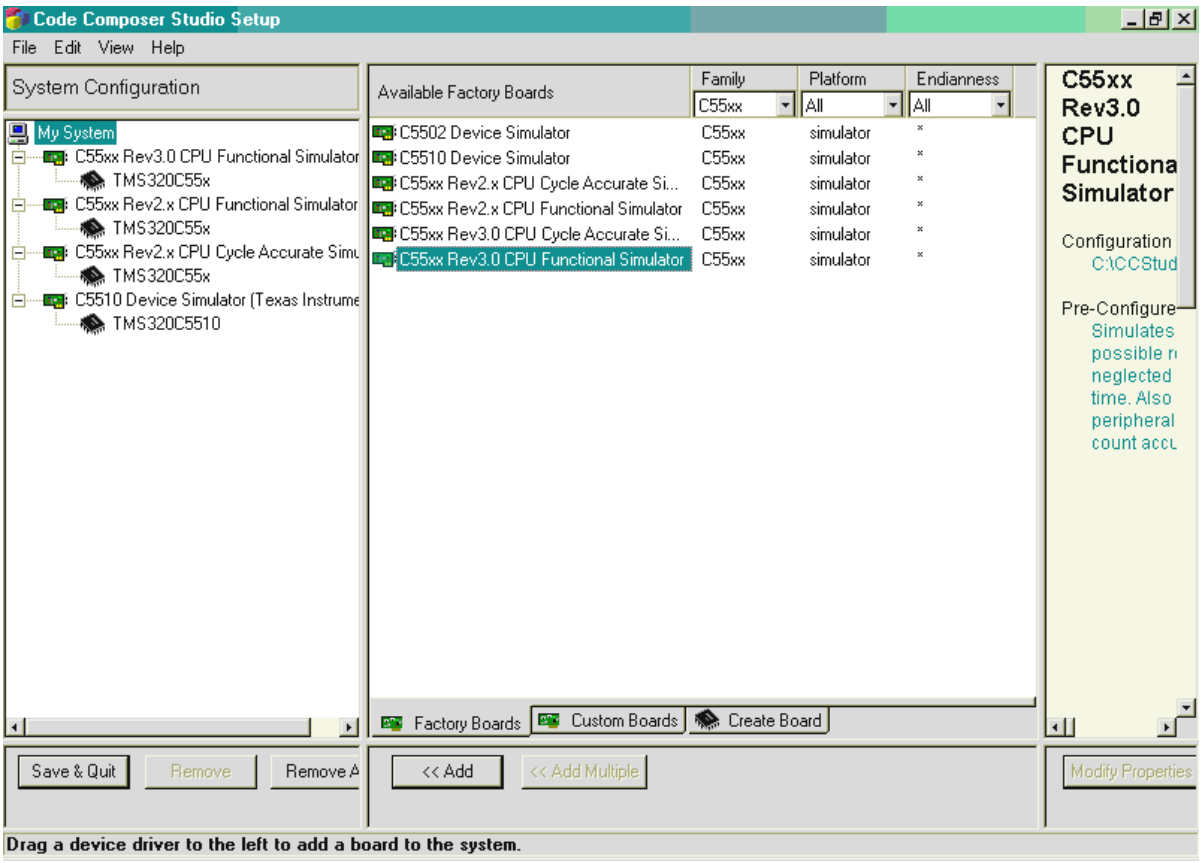

Рис. 5. **Меню *Code Composer Studio Setup***

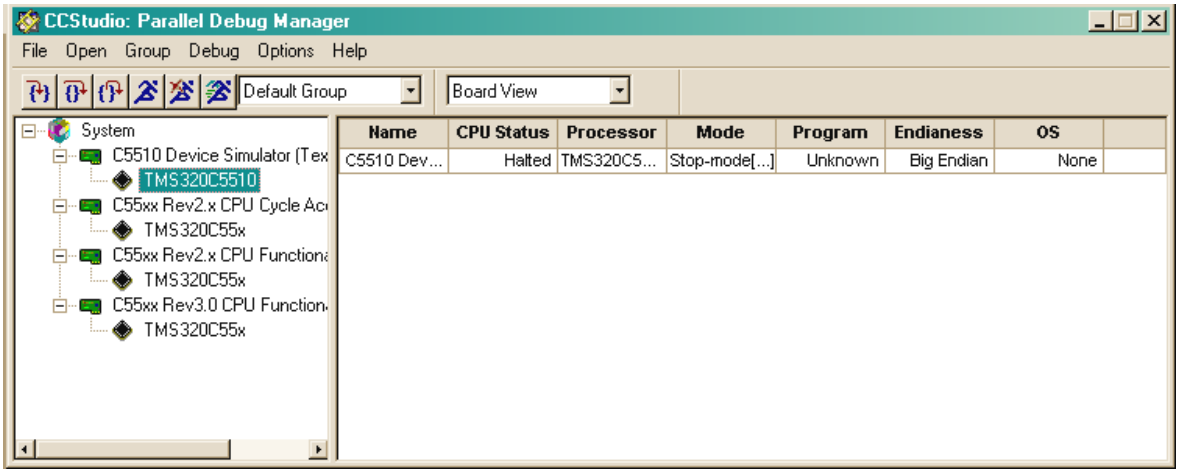

Рис. 6. **Меню *Parallel Debug Manager***

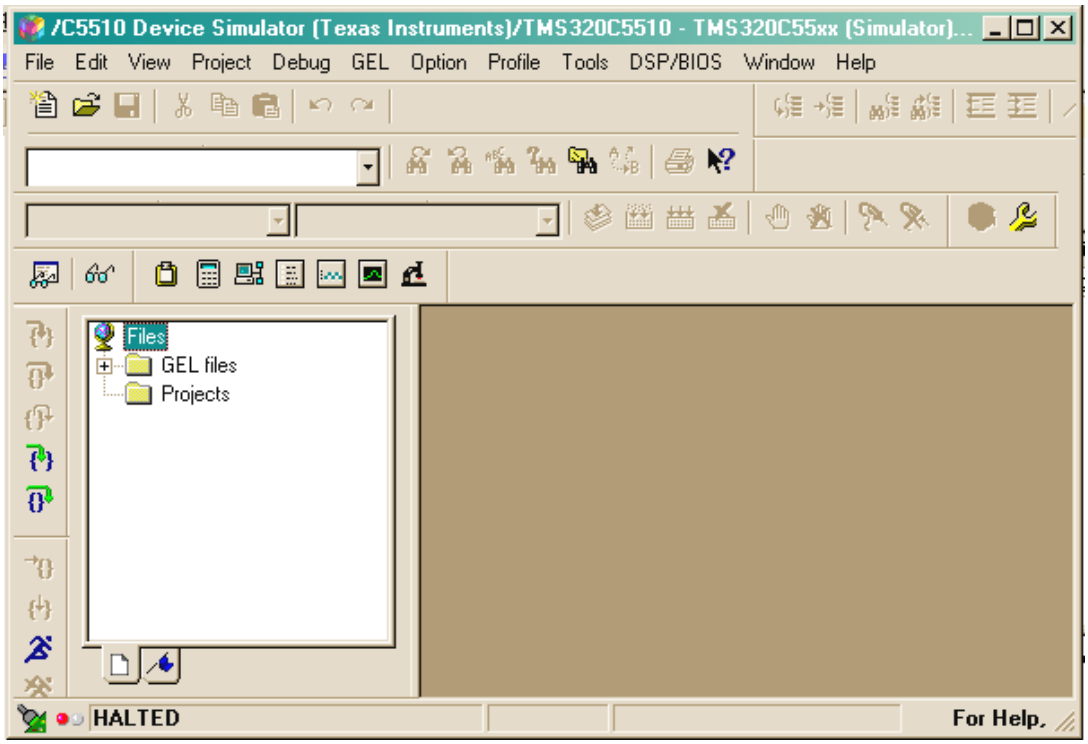

Рис. 7. **Интерфейс *Code Composer Studio* (симулятор)**

На рис. 7 справа видна панель, в заглавии которой написано *Files*. Это дерево менеджера проекта. Здесь будут отображаться файлы компонентов будущего приложения.

Для написания программы будут использоваться файлы со следующими расширениями:

– .lib – это библиотека, которая обеспечивает поддержку во время выполнения целевому чипу цифрового обработчика сигналов;

– .c – этот файл содержит исходный текст, который обеспечивает главные функциональные возможности этого проекта;

– .h – этот файл объявляет буферную С-структуру и позволяет определять любые необходимые константы;

– .pjt – этот файл содержит весь построенный проект и варианты конфигурации;

– .asm – этот файл содержит инструкции ассемблера;

– .cmd – этот файл наносит на карту секции в памяти.

Для создания нового проекта выберите из вкладки *Project* пункт *New* (рис. 8). На экране появится меню, изображенное на рис. 9.

Project type – этот пункт предназначен для описания функции самого проекта: при выборе *executable(.out)* проект будет полноценной программой; если будет выбрана функция *library(.lib)*, то проект после компиляции

будет считаться библиотекой. Библиотека является набором функций и процедур, заключенных в одном файле. Просмотреть содержимое этого файла как других, содержащихся в проекте, не представляется возможным. Итак, выбираем *executable(.out)*.

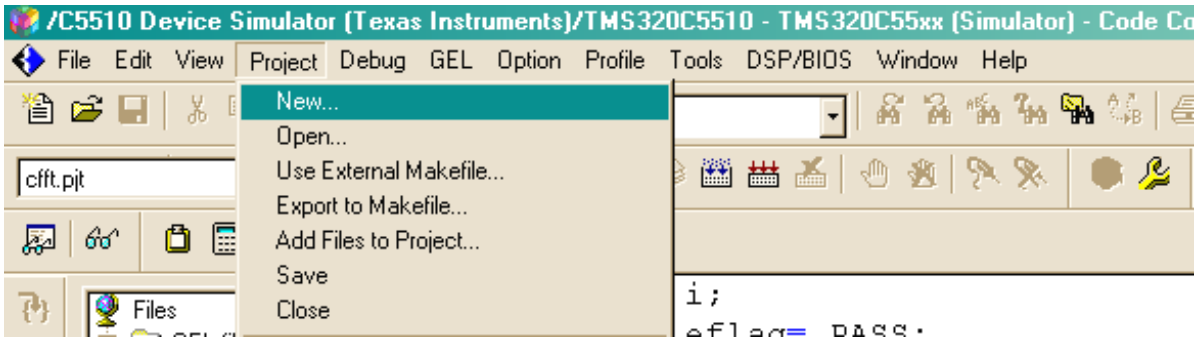

Рис. 8. **Путь создания нового проекта**

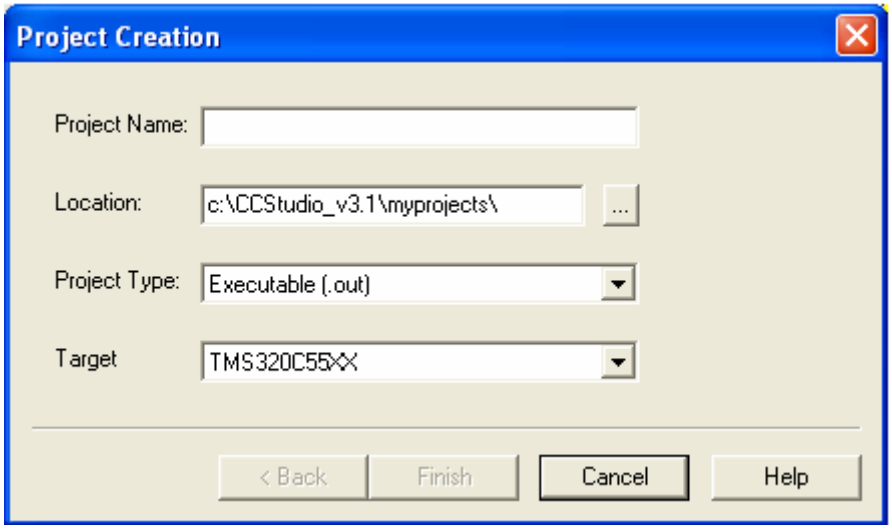

Рис. 9. **Меню создания нового проекта**

В разделе *Project Name* вводим название проекта. После этого эмулятор создаст в указанной вами директории (раздел *Location*) папку с названием проекта.

На рис. 10 видно, что менеджер проекта изменился. В нем появились папки с названиями компонентов, которые еще не подключены или не написаны. Для начала подключим библиотеку для нашего процессора: (…CCStudio_v3.1\C5500\cgtools\lib\rts55.lib). Для этого необходимо нажать на одной из папок правой кнопкой мыши и выбрать вкладку *Add files to project.*

Для полноценной работы программы напишем основной алгоритм.

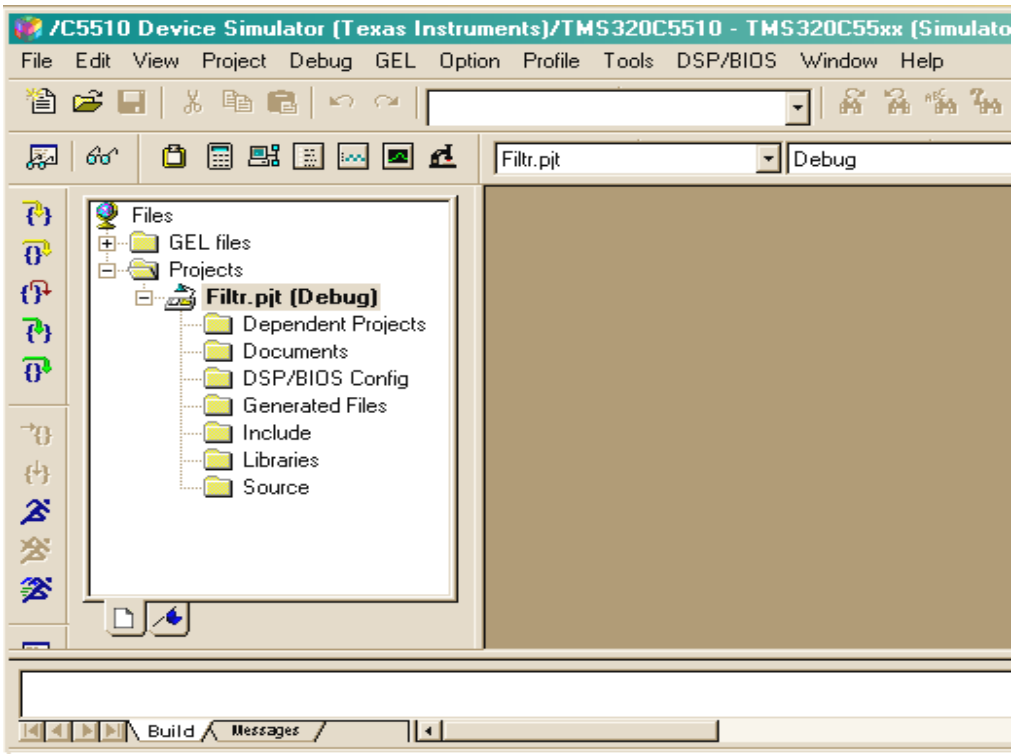

Рис. 10. **Меню менеджера проекта**

Прежде всего, нужно создать файл, в котором будут указываться команды на языке программирования «С». Для этого выберем на панели меню *File / New / Source File*. Чтобы наш файл стал главным файлом проекта, его необходимо сохранить под расширением *.c*, затем добавить его тем же способом, что и библиотеку, но из той директории, где находится проект. В папке *Source* появится добавленный файл. В последующем каждый добавленный файл будет появляться в своей папке согласно его назначению автоматически.

Структура написания программ идентична работе в среде программирования «С».

Для добавления файлов описаний (*.h*) необходимо сослаться на них в основном алгоритме, написав *#include "Fil_h.h"*, нажав правой кнопкой на названии проекта в менеджере и выбрав *Scan All File Dependencies*. При следующей компиляции проекта этот файл будет добавлен автоматически в проект. Запуск составленного проекта на компиляцию изображен на рис. 11.

При наличии ошибок после компиляции они будут обозначены в окне статуса компиляции. Вид и местоположение этого окна обозначен на рис. 12.

Каждая ошибка выделяется красным цветом, и при двойном нажатии на надписи ошибки курсор автоматически становится на ее место (это в случае синтаксической ошибки. Если ошибка в нарушении правила напи-

сания, допустим, неверное написание адреса той или иной переменной, то двойной щелчок ничего не даст, зато к каждой ошибке будет дано небольшое пояснение, по которому можно догадаться в чем суть ошибки).

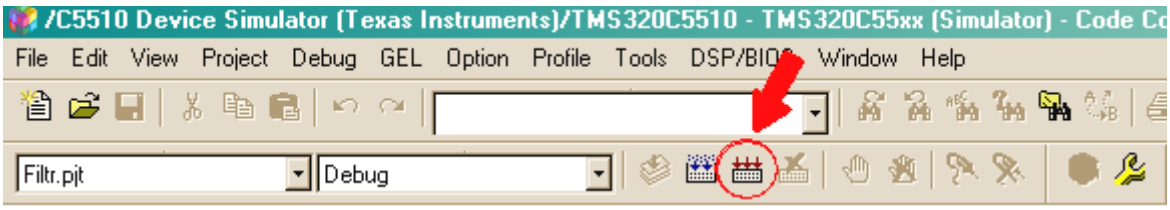

Рис. 11. **Запуск компилятора**

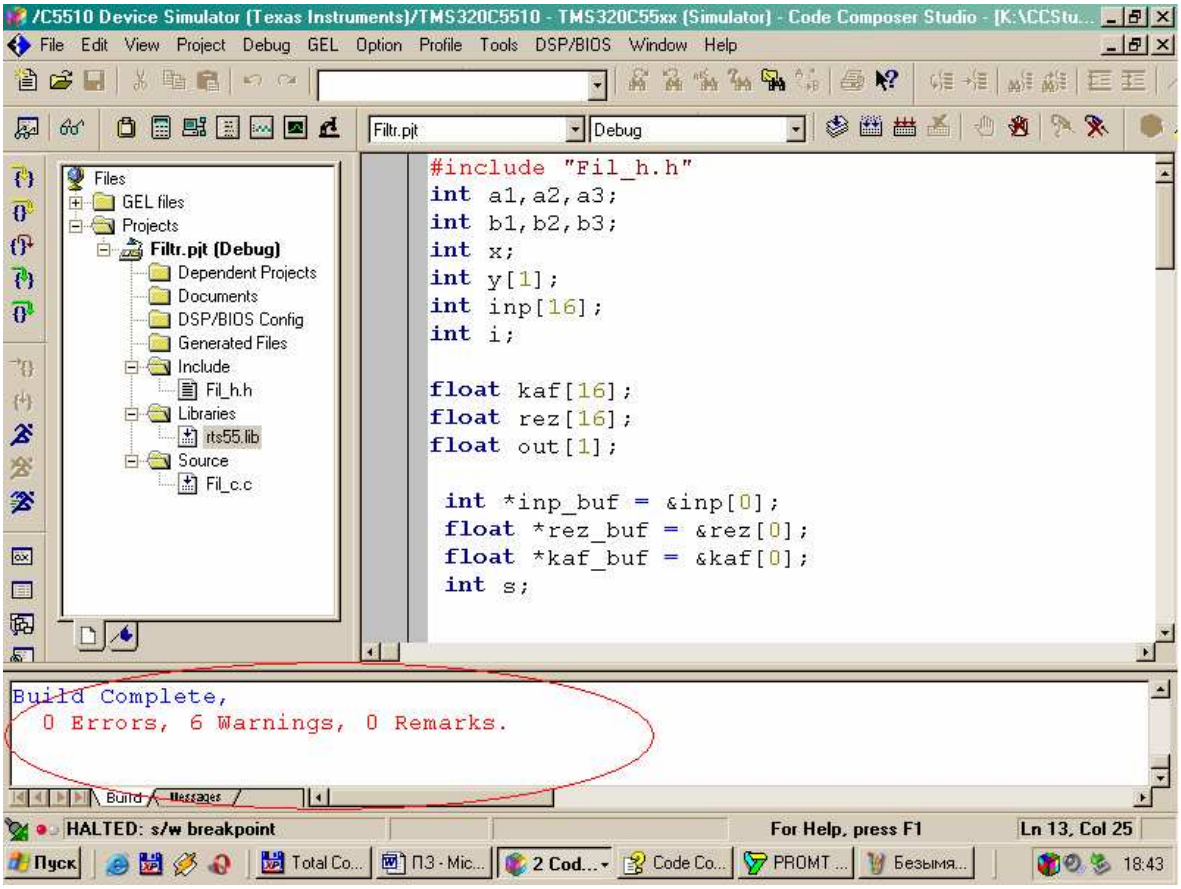

Рис. 12. **Окно статуса компилятора**

Для того чтобы эмулятор мог работать с переменными внутри программы или измененными данными, необходимо после компиляции в память записать код программы (имитировать перезапись памяти процессора). Для этого необходимо добавим следующий файл из папки *Debug* с помощью пункта *Load Program* из меню *File*. Название файла идентично названию проекта, только с расширением *.out*.

Теперь посмотрим алгоритм основной части проекта, содержащийся в файле *Fil_c.c*. Листинг программы приведен ниже.

```
#include "Fil_h.h"
int a, x, y[1], inp[16], i, s;
float rez[16], out[1];
int *inp_buf = &inp[0];
float *rez_buf = &rez[0];
float *kaf_buf = &kaf[0];

main()
{
 while (TRUE)
  {float *rez_buf = &rez[0];
  a=16;
  x=0;
  while (a--){*rez_buf++ = 0;}
  while (x!=16)
   {a=15;
    while (a!=-1)
     {inp[a]=inp[a-1];
     a--;}
    inp[0]=y[0];
    s=15;
    while (s!=0)
     {rez[x]=rez[x]+inp[s]*kaf[s];
      s--;}
    out[0]=rez[x];
    x++;}
  }
}
```

Тело программы состоит из двух частей: в первой части описываются переменные, вернее, указывается их тип (int, long, short и т.д.), во второй – описывается основная функция *main*, которая содержит в себе непосредственно весь алгоритм программы.

В папке с именем проекта хранится одноименный файл с расширением *.pjt*. Если открыть этот файл с помощью текстового редактора, то можно увидеть, что этот файл содержит имена файлов, включенных в проект, и другую информацию о настройках. Содержание файла *Filtr.pjt* представлено ниже.

```
; Code Composer Project File, Version 2.0

[Project Settings]
ProjectDir="C:\CCStudio_v3.1\MyProjects\Filtr\"
ProjectType=Executable
CPUFamily=TMS320C55XX
Tool="Compiler"
Tool="CustomBuilder"
Tool="DspBiosBuilder"
Tool="Linker"
Config="Debug"
Config="Release"

[Source Files]
Source="..\..\C5500\cgtools\lib\rts55.lib"
Source="K:\CCStudio_v3.1\MyProjects\Filtr\Fil_c.c"
Source="K:\CCStudio_v3.1\MyProjects\Filtr\Fil_h.h"
Source="..\..\tutorial\sim55xx\volume1\volume.cmd"

["Compiler" Settings: "Debug"]
Options=-g -fr"$(Proj_dir)\Debug" -d"_DEBUG"
["Compiler" Settings: "Release"]
Options=-o2 -fr"$(Proj_dir)\Release"

["Linker" Settings: "Debug"]
Options=-c -m".\Debug\Filtr.map" -o".\Debug\Filtr.out" -w -x

["Linker" Settings: "Release"]
Options=-c -m".\Release\Filtr.map" -o".\Release\Filtr.out" -w -x

["K:\CCStudio_v3.1\MyProjects\Filtr\Fil_h.h" Settings: "Debug"]
ExcludeFromBuild=true

["K:\CCStudio_v3.1\MyProjects\Filtr\Fil_h.h" Settings: "Release"]
ExcludeFromBuild=true

["..\..\tutorial\sim55xx\volume1\volume.cmd" Settings: "Debug"]
LinkOrder=1

["..\..\tutorial\sim55xx\volume1\volume.cmd" Settings: "Release"]
LinkOrder=1
```

Немаловажной особенностью эмулятора является то, что в нем можно подключать внешние файлы данных.

Принцип подключения заключается в том, что указывается часть кода, в момент выполнения которого должно осуществляться чтение из файла в определенный *массив*. Момент чтения может задаваться и другими событиями, а не только переходом указателя выполнения программы через определенный ее участок. Для того чтобы определить место в коде программы, в котором в массив должны записаться данные из внешнего файла, необходимо поставить там точку прерывания, которую можно увидеть на рис. 13.

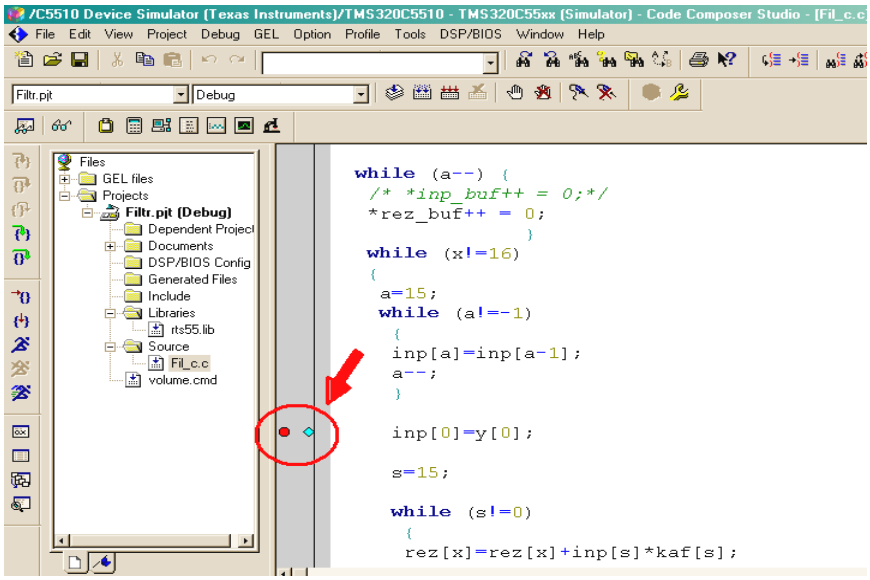

Рис. 13. **Точки прерывания**

Точки прерывания ставятся напротив той строки, где должно осуществляться присваивание внешней переменной внутреннему массиву. Для подключения файла необходимо во вкладке *File* выбрать пункт *File I/O*. На экране появится меню, представленное на рис. 14.

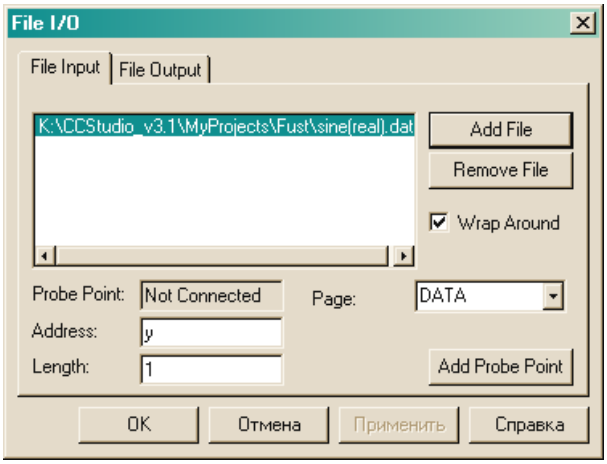

Рис. 14. **Меню файлов ввода/вывода**

Нажимаем *Add File* и выбираем файл данных, например, *sine.dat.*

Address – указатель массива, в котором происходит чтение.

Length – количество читаемых в массив единиц данных (отсчетов). В нашем случае чтение происходит по одному отсчету.

Далее нажимаем *Add Probe Point* для установки связи между точкой прерывания и файлом, из которого происходит чтение. На экране появится меню (рис. 15).

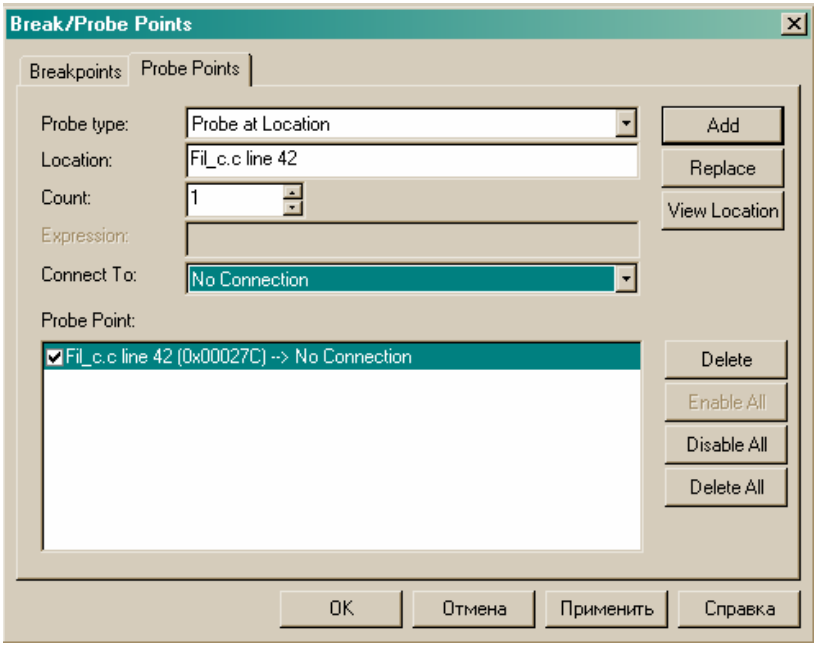

Рис. 15. **Меню указателя связывающего файл с событием**

Выделяем пункт с линией в программе в окне *Probe Point*, выбираем из меню *Connection To* название файла данных и нажимаем *Replace*. Подтверждаем.

Рассмотрим теперь возможность визуализации программы. Для этой цели в эмуляторе предусмотрен осциллограф.

Для его подключения на вкладке *Viev* выбираем *Graph*, затем *Time/Frequency*. На экране появляются настройки изображения (рис. 16).

Необходимо указать *Start Address* – указатель массива, из которого будут поступать данные в граф при переходе точки прерывания. Также указываем количество переменных, читаемых одновременно (*Acquisition Buffer Size*), и тип отображаемых данных (*DSP Data Type)*. На экране появляется граф, представленный на рис. 17.

Не забывайте обновлять код в памяти, добавляя следующий файл из папки *Debug* с помощью пункта *Load Program* из меню *File*. Название файла идентично названию проекта, только с расширением *.out.*

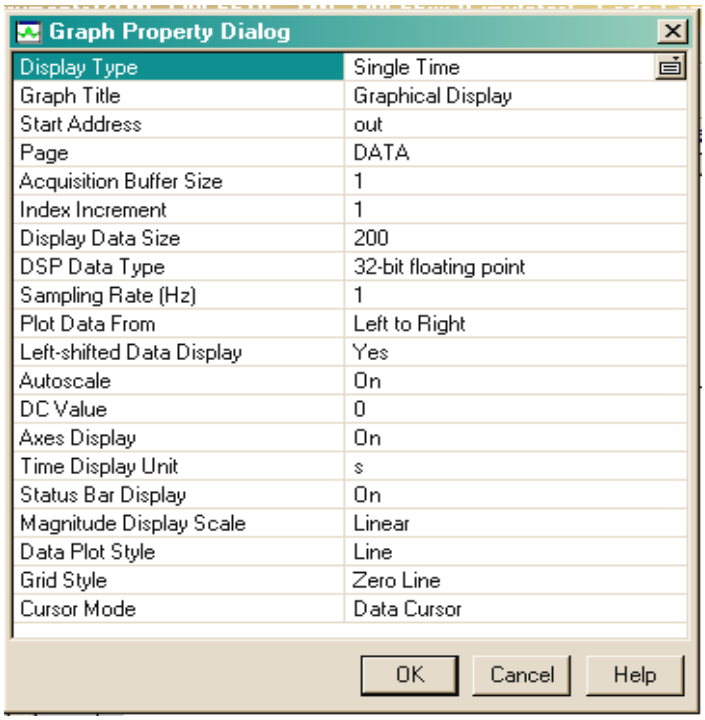

Рис. 16. **Меню настроек графика**

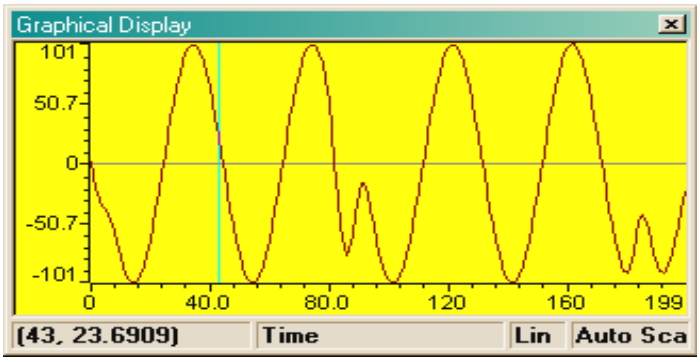

Рис. 17. **Вид графического окна**

Теперь демонстрация готова и осталось только запустить программу. Для этого нажимаем *Run* (чтобы программа проработала до точки прерывания) и *Animate* (для многократного повторения (зацикливания)).

Удобным инструментом является отображение значений тех или иных переменных в определенный момент времени. Для этого необходимо нажать в панели сверху *Watch Window* (рис. 18) и написать в ней название переменной для обзора или, нажав правой клавишей мыши на нужной переменной в программе, выбрать пункт *Add to Watch Window*.

Как видно, *Code Composer Studio* обладает широкими возможностями для написания и отладки приложений, что позволяет достигнуть высокого качества разрабатываемого продукта.

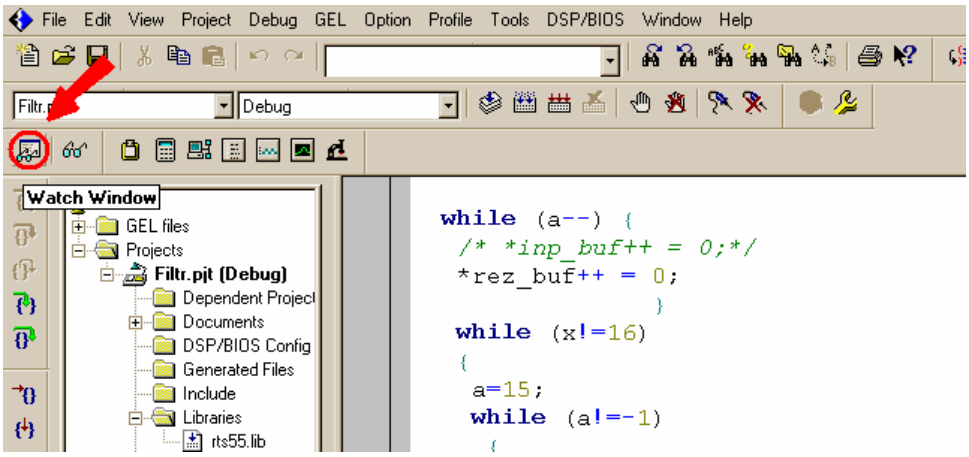

Рис. 18. **Вызов окна *Watch Window***

3.6. Аппаратная реализация

В данном пункте описана последовательность действий по аппаратной реализации программы, написанной в симуляторе.

Для начала работы с процессором обеспечим соединение платы процессора с компьютером. Для этого в плате разработчиками предусмотрен порт USB. Сначала необходимо подключить плату процессора к компьютеру с помощью шнура и не забыть подключить питание. При обнаружении компьютером платы, вернее, устройства USB, операционная оболочка Windows XP SP2 потребует установки драйверов. Для корректной установки необходимо выбрать ручной режим установки и показать путь, по которому находиться драйвер (….\CCStudio_v3.1\specdig\xds510usb\).

В программе *CCStudio_v3.1* для совмещения работы симулятора и реального процессорного блока предусмотрен достаточно гибкий переход. Он заключается в подключении необходимых модулей для работы с тем или иным процессором. Подключение обеспечивается в программе *Setup CCStudio_v3.1*.

Окно подключения модуля и сам модуль представлены на рис. 19.

Выделяем нужный нам модуль (C5510 DSK-USB) мышью и нажимаем кнопку *Add*. Теперь используем установки кнопкой *Save&Quit* и запускаем *CCStudio_v3.1*.

На экране монитора появится меню менеджера параллельной загрузки (рис. 20).

На рисунке видно, что плата процессора не подключена. Не обращая на это внимание, запускаем окно редактора кода двойным щелчком по иконке процессора *DSP_C55xx*. На экране появится окно редактора с предупреждением о том, что плата процессора не подключена (рис. 21).

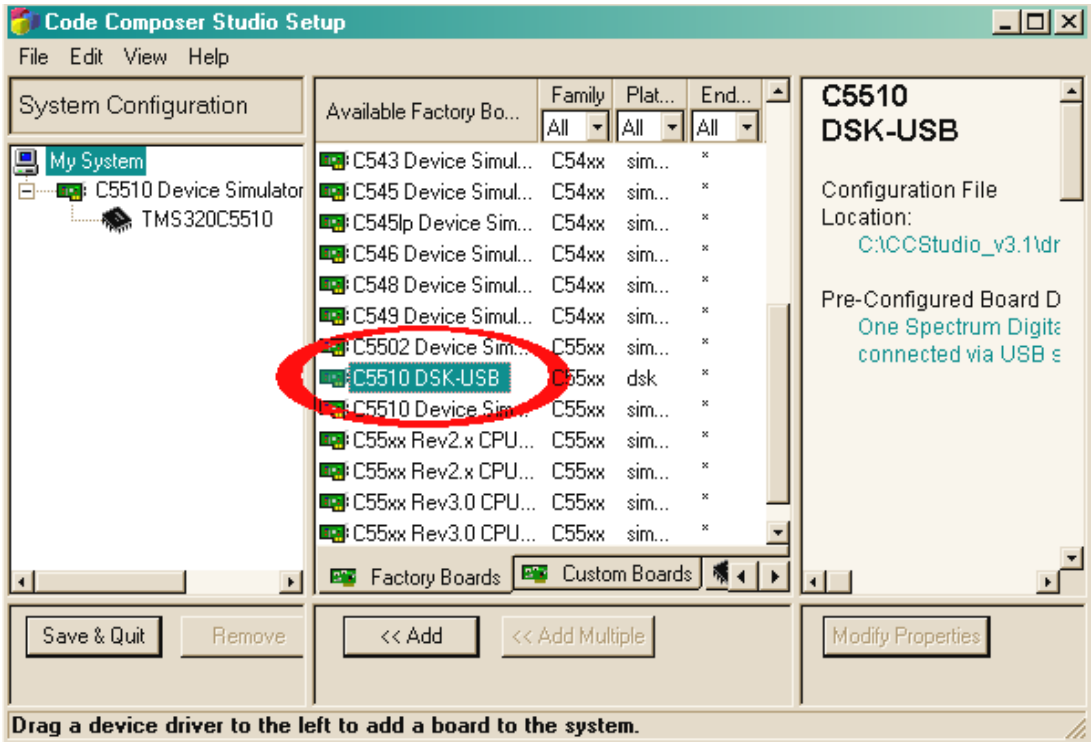

Рис. 19. **Модуль процессорного блока**

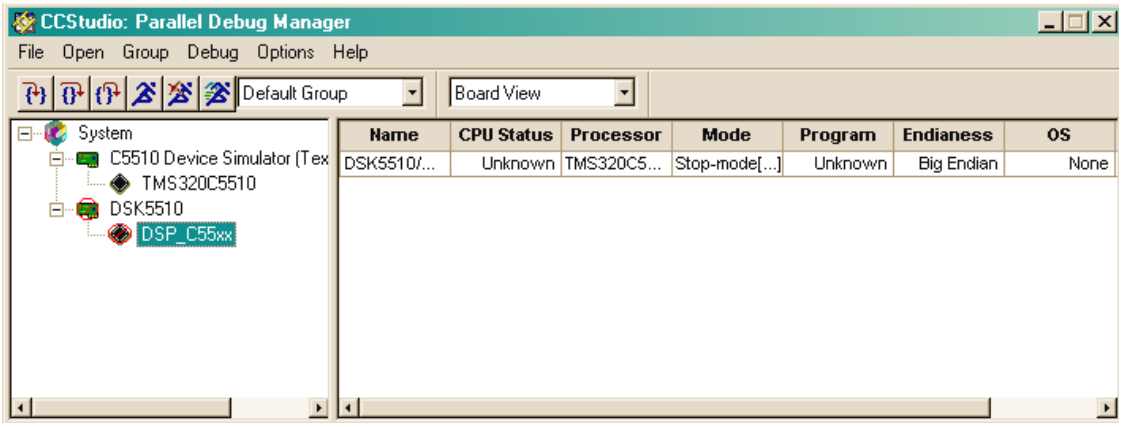

Рис. 20. **Менеджер параллельной загрузки**

На рис. 21 стрелкой показано место предупреждения.

Для подключения процессора необходимо выбрать в главной панели меню *Debug*, далее *Connect* или нажать сочетание клавиш Alt + C. Окно редактора изменится. Изменение окна представлено на рис. 22.

Такой способ подключения не единственный, можно просто в окне менеджера параллельной загрузки правой клавишей мыши нажать на иконке *DSP_C55xx* и выбрать в меню пункт *Connect Device.*

Далее создание и редактирование проекта происходит также, как в симуляторе.

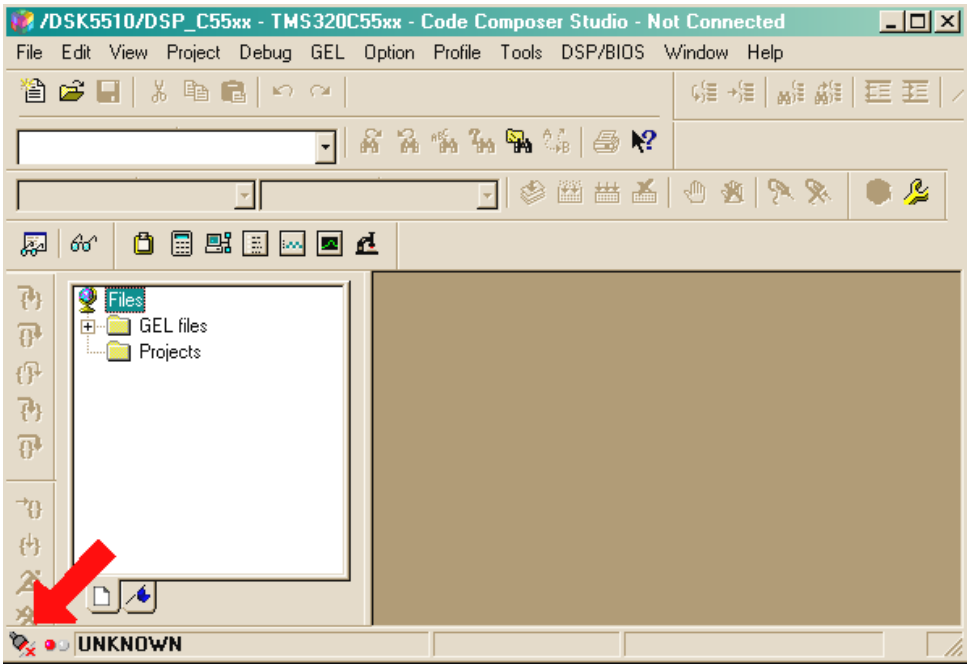

Рис. 21. **Окно редактора**

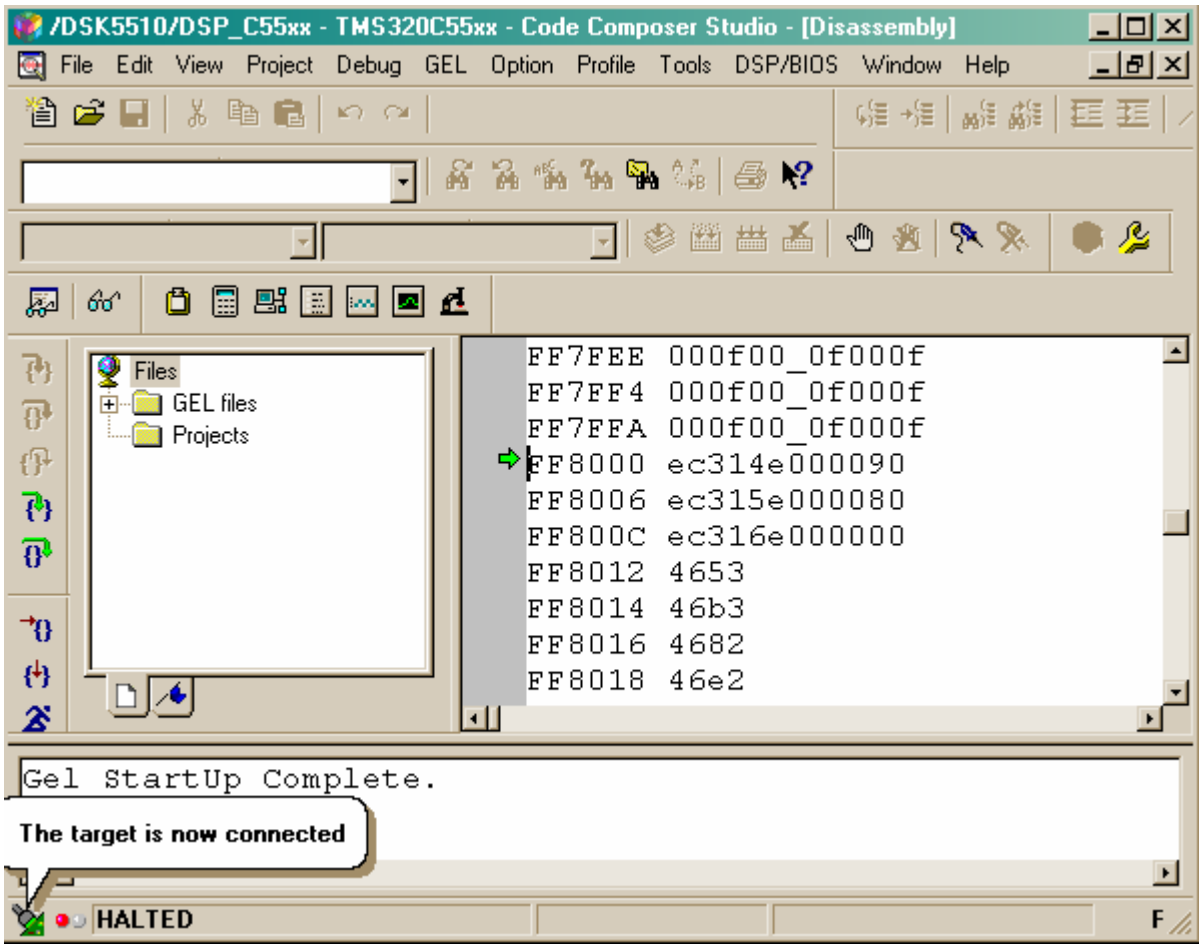

Рис. 22. **Изменение окна редактора**

Для реализации типовой темы курсового проекта («Организация фильтра нижних частот») необходимо подключить линейный вход и линейный выход процессорной платы. Для этого можно воспользоваться уже готовой программой *tone*, расположенной в библиотеке по пути (…*CCStudio_v3.1\examples\dsk5510\bsl\tone*\). Вид окна редактора *CCStudio_v3.1* представлен на рис. 23.

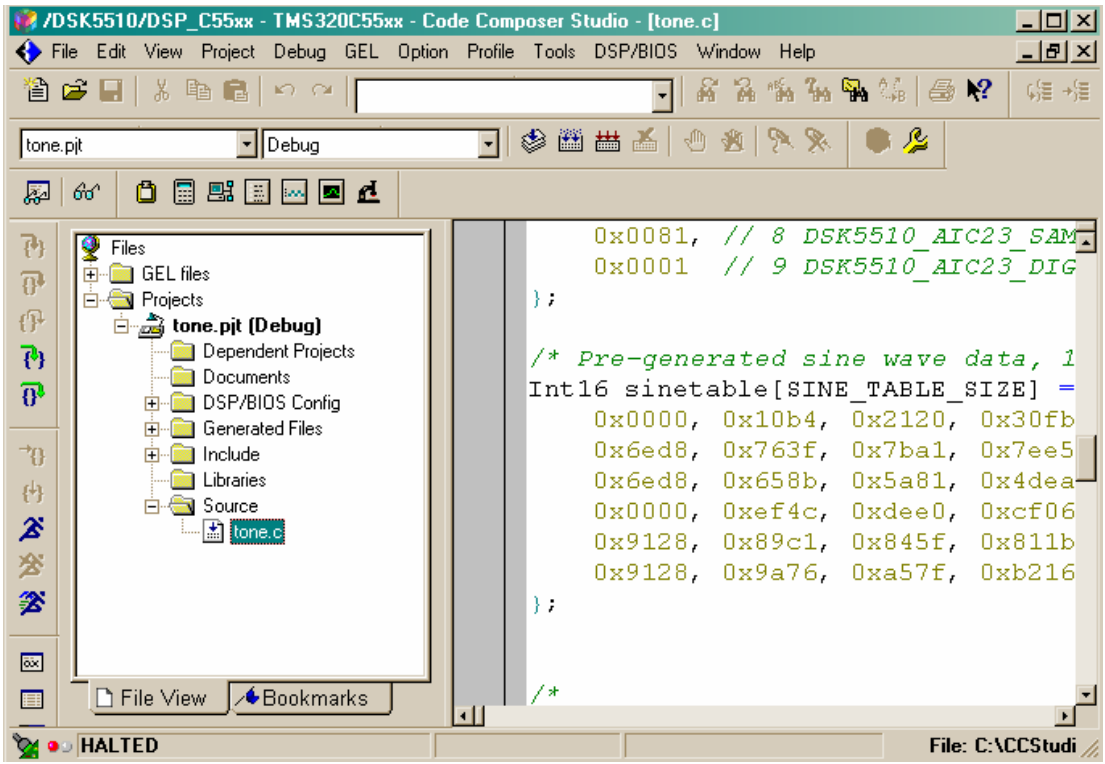

Рис. 23. **Окна редактора с проектом *tone***

Этот пример использует AIC23 кодек модуля библиотеки 5510, поддерживаемый платой процессора, чтобы генерировать синус с частотой в 1 кГц, и выполняется в течение 5 с.

Плюс этой программы в том, что не надо прописывать множество настроек процессора вручную. Самые главные настройки заключаются в файле dsk5510_aic.h. Там же содержится описание самого кодека.

Листинг файла dsk5510_aic.h представлен в прил. Д.

Процедуры и функции настроек описаны с помощью макросов. Это значительно облегчает задачу программиста. Поэтому стоит уделить внимание процедурам и функциям, поддерживаемым кодеком (табл. 4).

Для выполнения курсового проекта необходимо установить значение частоты дискретизации 44,1 кГц, а также записать обработанные отсчеты в регистр вывода.

Таблица 4

Процедуры и функции, поддерживаемые кодеком AIC23

Функция	Описание
DSK5510_AIC23_rset*(DSK5510_AIC23_CodecHandle hCodec, Uint16 regnum, Uint16 regval)*	Процедура, при помощи которой можно установить значение необходимого регистра, задав его название и значение целым 16-битным числом
DSK5510_AIC23_rget*(DSK5510_AIC23_CodecHandle hCodec, Uint16 regnum)*	Возвращает значение регистра по его названию
DSK5510_AIC23_openCodec*(**int** id, DSK5510_AIC23_Config *Config)*	Обозначает открытие кодека с номером и возвращает указатель открытия
DSK5510_AIC23_closeCodec*(DSK5510_AIC23_CodecHandle hCodec)*	Процедура закрытия кодека по указателю
DSK5510_AIC23_config*(DSK5510_AIC23_CodecHandle hCodec, DSK5510_AIC23_Config *Config)*	Процедура применения настроек регистров обязательных для корректной работы кодека
DSK5510_AIC23_write16*(DSK5510_AIC23_CodecHandle hCodec, Int16 val)*	Функция записи в регистр вывода 16-битной величины, возвращает булевское значение (true-false)
DSK5510_AIC23_write32*(DSK5510_AIC23_CodecHandle hCodec, Int32 val)*	Функция записи в регистр вывода 32-битной величины, возвращает булевское значение (true-false)
DSK5510_AIC23_read16*(DSK5510_AIC23_CodecHandle hCodec, Int16 *val)*	Читает из регистра ввода информацию в 16-разрядную величину, возвращает булевское значение (true-false)
DSK5510_AIC23_read32*(DSK5510_AIC23_CodecHandle hCodec, Int32 *val)*	Читает из регистра ввода информацию в 32-разрядную величину, возвращает булевское значение (true-false)
DSK5510_AIC23_outGain*(DSK5510_AIC23_CodecHandle hCodec, Uint16 outGain)*	Процедура, устанавливающая увеличение значение выходных величин (усиление)
DSK5510_AIC23_loopback*(DSK5510_AIC23_CodecHandle hCodec, CSLBool mode)*	Устанавливает состояние перемычки
DSK5510_AIC23_mute*(DSK5510_AIC23_CodecHandle hCodec, CSLBool mode)*	Отключает или включает звук
DSK5510_AIC23_powerDown*(DSK5510_AIC23_CodecHandle hCodec, Uint16 sect)*	Включает или отключает аттенюатор аналого-цифрового и цифро-аналогового преобразователя
DSK5510_AIC23_setFreq*(DSK5510_AIC23_CodecHandle hCodec, Uint32 freq)*	Устанавливает величину частоты дискретизации

Файл тела программы, исправленный под требования рассматриваемого фильтра (*tone.c*), описан ниже. Также в программу был добавлен файл коэффициентов *Fil_h.h*.

Были введены следующие процедуры и функции:

– **DSK5510_AIC23_setFreq** (hCodec, 5) – установка частоты дискретизации. В файле dsk5510_aic.h под цифрой 5 подразумевается 44,1 кГц;

– **while** (**!DSK5510_AIC23_read16** (hCodec, adr)) – цикл выполняет чтение в указатель адреса переменной *x* (см. листинг ***tone.c***), пока значение функции равно true.

ВАЖНО. *При разработке программы были использованы коэффициенты фильтра типа float. Это неприемлемо для данного типа процессора, т.к. выполнение операции умножения целого числа на дробное с большим числом знаков после запятой занимает много процессорного времени. Поэтому при демонстрации могут быть слышны щелчки или небольшой треск. При использовании целых чисел такого эффекта не возникает.*

Во избежание выхода из строя порта линейного входа ни в коем случае нельзя подавать на него сигнал большой мощности, предназначенный для воспроизведения на колонках без усилителей, либо сигнал, прошедший усиление, – ТОЛЬКО СИГНАЛ С ЛИНЕЙНОГО ВЫХОДА АУДИО УСТРОЙСТВ.

ЛИТЕРАТУРА

1. Проектирование проблемно-ориентированных вычислительных средств: учеб. пособие по курсовому проектированию для студ. спец. I-40 02 02 «Электронные вычислительные средства» дневной формы обуч. / А.А. Петровский [и др.]. – Минск: БГУИР, 2005. – 51 с.
2. Айфичер, Эммануил С. Цифровая обработка сигналов: практический подход / Эммануил С. Айфичер, Барри У. Джервис: пер. с англ. – 2-е изд. – М.: Издательский дом «Вильямс», 2004. – 992 с.: ил.
3. Саломатин, С.Б. Цифровая обработка сигналов в радиоэлектронных системах: учеб. пособие для студентов специальности 39 01 02 «Радиоэлектронные системы» / С.Б. Саломатин. – Минск, 2002.
4. Глинченко, А.С. Цифровая обработка сигналов: в 2 ч. / А.С. Глинченко. – Красноярск: Изд-во КГТУ. – Ч. 1. – 2001. – 199 с.
5. Петько, В.И. Цифровая фильтрация и обработка сигналов: учеб. пособие / В.И. Петько, В.Е. Куконин, Н.Б. Шихов. – Минск: Універсітэцкае, 1995. – 168 с.
6. TMS320C55x DSP Peripherals Overview Reference Guide. Chapter 4. Direct Memory Access (DMA) Controller (Literature number: SPRU587). – Режим доступа: www.ti.com.
7. TMS320C55x DSP Peripherals Overview Reference Guide. Chapter 7. Host Port Interface (HPI) (Literature number SPRU588). – Режим доступа: www.ti.com.
8. TMS320C55x DSP Peripherals Overview Reference Guide (Literature number: SPRU317G). – Режим доступа: www.ti.com.
9. TMS320C55x DSP Library Programmer's Reference (Literature Number: SPRU422). – Режим доступа: www.ti.com.

ПРИЛОЖЕНИЯ

Приложение А

Таблица А.1

Системный регистр (расшифровка битов)

№ бита	Название бита	Значение по умолчанию	Функции
15 − 10	Резерв	000000	Эти биты сохранены от записи
9	HPE	1	**EHPI** возможность pullup/pulldown. Позволяет повысить напряжение EHPI в контрольных точках HCS, HAS, HR/W, HMODE, HCNTL0, и HA1/HCNTL1. Позволяет понизить напряжение EHPI в контрольных точках HBE0 и HBE1. HPE = 0. Повышение и понижение напряжения отключается HPE = 1
8	BH	0	**EMIF** позволяет установить задержку. Устанавливает внутрикристалльное прерывание D[31:0]. BH = 0, EMIF порта отключается. BH = 1, EMIF порта включается
7	HBH	0	**EHPI** позволяет установить возможность задержки порта данных. Устанавливает задержку HD[15:0]. HBH = 0, EHPI порта отключается. HBH = 1, EHPI порта включается
6	BOOTM3	0	**Статус BOOTM3.** Этот бит представляется только для чтения и указывает на состояние BOOTM3 контакта
5	Резерв	0	Этот бит защищен от записи
4	Резерв	0	Этот бит защищен от записи и должен содержать 0
3	Резерв	0	Этот бит защищен от записи
2 − 0	CLKDIV	000	**CLKOUT** разделяющий фактор. Позволяет представить CLKOUT как делитель основной частоты процессора. Эта область не затрагивает программирование PLL. CLKDIV = 000, CLKOUT представляет деление частоты процессора на 1; CLKDIV = 001, CLKOUT представляет деление на 2; CLKDIV = 010, CLKOUT – на 4; CLKDIV = 011, CLKOUT – на 6; CLKDIV = 100, CLKOUT – на 8; CLKDIV = 101, CLKOUT – на 10; CLKDIV = 110, CLKOUT – на 12; CLKDIV = 111, CLKOUT представляет деление частоты процессора на 14

Приложение Б

Таблица Б.1

Список регистров процессора

Регистры **VC5510**	Шестнадцатеричный адрес **(HEX)**	Описание
1	*2*	*3*
IER0	00	Регистр маски прерывания 0
I FR0	01	Регистр флага прерывания 0
ST0_55	02	Регистр статуса 0 для C55x
ST1_55	03	Регистр статуса 1 для C55x
ST3_55	04	Регистр статуса 3 для C55x
–	05	Зарезервировано
ST0	06	Регистр статуса ST0 (для совместимости с 54x)
ST1	07	Регистр статуса ST1 (для совместимости с 54x)
AC0L	08	Аккумулятор 0 (эквивалент аккумулятору A в C54x)
AC0H	09	
AC0G	0A	
AC1L	0B	Аккумулятор 1 (эквивалент аккумулятору A в C54x)
AC1H	0C	
AC1G	0D	
T3	0E	Временный регистр
TRN0	0F	Регистр перехода
AR0	10	Вспомогательный регистр 0
AR1	11	Вспомогательный регистр 1
AR2	12	Вспомогательный регистр 2
AR3	13	Вспомогательный регистр 3
AR4	14	Вспомогательный регистр 4
AR5	15	Вспомогательный регистр 5
AR6	16	Вспомогательный регистр 6
AR7	17	Вспомогательный регистр 7
SP	18	Регистр указателя стека
BK03	19	Циркулярный буферный регистр размера
BRC0	1A	Граница повторения блока
RSA0L	1B	Адрес начала блока повторения
REA0L	1C	Адрес конца блока повторения
PMST	1D	Регистр статуса состояния процессора
XPC	1E	Регистр расширения границы программы
–	1F	Зарезервировано
T0	20	Временный регистр данных 0
T1	21	Временный регистр данных 1
T2	22	Временный регистр данных 2
T3	23	Временный регистр данных 3

Окончание табл. Б.1

1	*2*	*3*
AC2L	24	Аккумулятор 2
AC2H	25	
AC2G	26	
CDP	27	Коэффициент указателя данных
AC3L	28	Аккумулятор 3
AC3H	29	
AC3G	2A	
DPH	2B	Расширенный указатель страницы данных
MDP05	2C	Зарезервировано
MDP67	2D	Зарезервировано
DP	2E	Начальный адрес страницы данных
PDP	2F	Начальный адрес периферийной страницы данных
BK47	30	Регистр размера циркулярного буфера для AR[4 – 7]
BKC	31	Регистр размера циркулярного буфера для CDP
BSA01	32	Регистр начального адреса циркулярного буфера для AR[0 – 1]
BSA23	33	Регистр начального адреса циркулярного буфера для AR[2 – 3]
BSA45	34	Регистр начального адреса циркулярного буфера для AR[4 – 5]
BSA67	35	Регистр начального адреса циркулярного буфера для AR[6 – 7]
BSAC	36	Регистр начального адреса коэффициента циркулярного буфера
BIOS	37	Местоположение хранения указателя страницы данных для 128-word Data Table
TRN1	38	Регистр перехода1
BRS1	3A	Сохранение блока повторения 1
CSR	3B	Вычисление единственного повторения
RSA0H	3C	Адрес начала повторения 0
RSA0L	3D	
REA0H	3E	Адрес конца повторения 0
REA0L	3F	
RSA1H	40	Адрес начала повторения 1
RSA1L	41	
REA1H	42	Адрес конца повторения 1
REA1L	43	
RPTC	44	Граница повторения
IER1	45	Регистр маски прерывания 1
IFR1	46	Регистр флага прерывания 1
DBIER0	47	Отладка IER0
DBIER1	48	Отладка IER1
IVPD	49	Указатель вектора прерывания DSP
IVPH	4A	Указатель вектора прерывания HOST
ST2_55	4B	Регистр статуса 2 для C55x
SSP	4C	Указатель системного стека
SP	4D	Указатель пользовательского стека
SPH	4E	Расширенный указатель страницы данных для SP и для SSP

Приложение В

Таблица В.1

Базовые функции цифровой обработки сигналов

Функции	Описание
1	*2*
БПФ	
void cfft (DATA *x, ushort nx, type)	Прямое комплексное преобразование Фурье
void cfft32 (LDATA *x, ushort nx, type);	Прямое комплексное 32-битное преобразование Фурье
void cifft (DATA *x, ushort nx, type)	Обратное комплексное преобразование Фурье
void cifft32 (LDATA *x, ushort nx, type)	Обратное комплексное 32-битное преобразование Фурье
void cbrev (DATA *x, DATA *r, ushort n)	Комплексная бит-риверсия
void cbrev32 (LDATA *a, LDATA *r, ushort)	32-битная бит-риверсия
void rfft (DATA *x, ushort nx, type)	Вещественное прямое преобразование Фурье
void rifft (DATA *x, ushort nx, type)	Вещественное обратное преобразование Фурье
Фильтрация и свертка	
ushort fir (DATA *x, DATA *h, DATA *r, DATA *dbuffer, ushort nx, ushort nh)	Фильтрация прямая форма
ushort fir2 (DATA *x, DATA *h, DATA *r, DATA *dbuffer, ushort nx, ushort nh)	Фильтрация прямая форма (опритимизированный алгоритм)
ushort firs (DATA *x, DATA *h, DATA *r,)	Симметричный фильтр
ushort cfir (DATA *x, DATA *h, DATA *r, DATA *dbuffer, ushort nx, ushort nh)	Комплексный фильтр
ushort convol (DATA *x, DATA *h, DATA *r, ushort nr, ushort nh)	Свертка
ushort convol1 (DATA *x, DATA *h, DATA *r, ushort nr, ushort nh)	Свертка (с применением двойного сдвига с накоплением)
ushort convol2 (DATA *x, DATA *h, DATA *r, ushort nr, ushort nh)	Свертка (с применением двойного сдвига с накоплением)
ushort iircas4 (DATA *x, DATA *h, DATA *r, DATA *dbuffer, ushort nbiq, ushort nx)	Фильтр IIR, прямая форма II с использованием 4-х коэффициентов Бигвуда
ushort iircas5 (DATA *x, DATA *h, DATA *r, DATA *dbuffer, ushort nbiq, ushort nx)	Фильтр IIR, прямая форма II с использованием 5-ти коэффициентов Бигвуда
ushort iircas51 (DATA *x, DATA *h, DATA *r, DATA *dbuffer, ushort nbiq, ushort nx)	Каскадный фильтр IIR, прямая форма I с использованием 4-х коэффициентов Бигвуда
ushort iirlat (DATA *x, DATA *h, DATA *r, DATA *pbuffer, int nx, int nh)	Инверсия решетки фильтра IIR
ushort firlat (DATA *x, DATA *h, DATA *r, DATA *pbuffer, int nx, int nh)	Прямая решетка фильтра FIR
ushort firdec (DATA *x, DATA *h, DATA *r, DATA *dbuffer, ushort nh, ushort nx, ushort D)	Фильтр дециматор
ushort firinterp (DATA *x, DATA *h, DATA *r, DATA *dbuffer, ushort nh, ushort nx, ushort I)	Фильтр интерполятор

Продолжение таблицы В.1

1	*2*
ushort hilb16 (DATA *x, DATA *h, DATA *r, DATA *dbuffer, ushort nx, ushort nh)	Фильтр трансформатор Хиберта
ushort iir32 (DATA *x, LDATA *h, DATA *r, LDATA *dbuffer, ushort nbiq, ushort nr)	Фильтр с двойной точностью
Адаптивная фильтрация	
ushort dlms (DATA *x, DATA *h, DATA *r, DATA *des, DATA *dbuffer, DATA step, ushort nh, ushort nx)	LMS фильтр (с задержками)
ushort oflag = dlmsfast (DATA *x, DATA *h, DATA *r, DATA*des, DATA *dbuffer, DATA step, ushort nh, ushort nx)	Адаптивный фильтр с задержками
Корреляция	
ushort acorr (DATA *x, DATA *r, ushort nx, ushort nr, type)	Автокорелляция
ushort corr (DATA *x, DATA *y, DATA *r, ushort nx, ushort ny, type)	Корелляция
Тригонометрия	
ushort sine (DATA *x, DATA *r, ushort nx)	Синус вектора
ushort atan2_16 (DATA *i, DATA *q, DATA *r, ushort nx)	Инверсия тангенса четвертого квадранта вектора
ushort atan16 (DATA *x, DATA *r, ushort nx)	Арктангенс
Математические функции	
ushort add (DATA *x, DATA *y, DATA *r, ushort nx, ushort scale)	Оптимизированный вектор суммы
ushort expn (DATA *x, DATA *r, ushort nx)	Вектор экспоненты
short bexp (DATA *x, ushort nx) ushort logn (DATA *x, LDATA *r, ushort nx)	Экспонента всех величин вектора
ushort log_2 (DATA *x, LDATA *r, ushort nx)	Логарифм по основанию 2
ushort log_10 (DATA *x, LDATA *r, ushort nx)	Логарифм по основанию 10
short maxidx (DATA *x, ushort ng, ushort ng_size)	Индекс максимальной величины вектора
short maxidx34 (DATA *x, ushort nx)	Индекс максимального элемента вектора ≤ 34
short maxval (DATA *x, ushort nx)	Максимальный элемент вектора
void maxvec (DATA *x, ushort nx, DATA *r_val, DATA *r_idx)	Индекс и величина максимального элемента в векторе
short minidx (DATA *x, ushort nx)	Индекс минимальной величины в векторе
short minval (DATA *x, ushort nx)	Минимальная величина вектора
void minvec (DATA *x, ushort nx, DATA *r_val, DATA *r_idx)	Индекс и величина минимального элемента в векторе
ushort mul32 (LDATA *x, LDATA *y, LDATA *r, ushort nx)	32-битный вектор умножения
short neg (DATA *x, DATA *r, ushort nx)	16-битный вектор отрицания

Окончание таблицы В.1

short neg32 (LDATA *x, LDATA *r, ushort nx)	32-битный вектор отрицания
short power (DATA *x, LDATA *r, ushort nx)	Сумма квадратов (мощность)
void recip16 (DATA *x, DATA *r, DATA *rexp, ushort nx)	Вектор аналог
void ldiv16 (LDATA *x, DATA *y, DATA *r, DATA *rexp, ushort nx)	32-битный с 16-битным разделением
ushort sqrt_16 (DATA *x, DATA *r, short nx)	Квадратный корень
short sub (DATA *x, DATA *y, DATA *r, ushort nx, ushort scale)	Вектор вычитания
Работа с матрицами	
ushort mmul (DATA *x1, short row1, short col1, DATA *x2, short row2, short col2, DATA *r)	Матричное умножение
ushort mtrans (DATA *x, short row, short col, DATA *r)	Транспонирование

Приложение Г

Алгоритм фильтрации

```
float kaf[16]={
0, 0.0005,
0.0034, 0.0149,
0.0446, 0.0982,
0.1637, 0.2104,
0.2104, 0.1637,
0.0982, 0.0446,
0.0149, 0.0034,
0.0005, 0,};
int a;
int x;
int y[1];
int inp[16]; /*объявление буфера входных данных. Он равен размеру окна фильтра */
int i;

float rez[16]; /*объявление буфера выходных данных. Он равен размеру окна фильтра */
float out[1];  /*буфер выходных отсчетов*/
int s;

main(){
  while (1) { /*бесконечный цикл*/
  float *rez_buf = &rez[0]; /*присвоение адреса одной переменной другой переменной */
  a=16; /*устанавливаем размер окна*/
  x=0;
   while (a--) { /*обнуляем память по адресам выходного буфера*/
   *rez_buf++ = 0;}
   while (x!=16)/*цикл формирования выходного буфера данных*/
   {a=15;
    while (a!=-1) /*цикл сдвига данных в буфере на 1 адрес вверх*/
     { inp[a]=inp[a-1];
     a--;}
     inp[0]=y[0]; /*чтение данных из внешнего файла*/
     s=15;
     while (s!=-1) /*перемножение буфера входных данных на коэффициенты и формиро-
                     вание одного элемента буфера выходных данных*/
      { rez[x]=rez[x]+inp[s]*kaf[s];
      s--;}
      out[0]=rez[x]; /*запись результата по адресу выхода*/
      x++;
   }
  }
}
```

Приложение Д

Листинг файла dsk5510_aic.h

```
 *  ======== dsk5510_aic23.h ========
 *  Codec interface for AIC23 on the DSK5510 board
#ifndef DSK5510_AIC23_
#define DSK5510_AIC23_
#ifdef __cplusplus
extern "C" {
#endif
#include <csl.h>
#include <csl_mcbsp.h>
/* AIC23 McBSP defines */
#define DSK5510_AIC23_DATAHANDLE    C55XX_DMA_MCBSP_hMcbsp
#define DSK5510_AIC23_CONTROLHANDLE C55XX_CONTROLHANDLE_hMcbsp
/* McBSP handles */
extern MCBSP_Handle DSK5510_AIC23_DATAHANDLE,
DSK5510_AIC23_CONTROLHANDLE;
/* Codec module definitions */
#define DSK5510_AIC23_NUMREGS        10
#define DSK5510_AIC23_LEFTINVOL      0
#define DSK5510_AIC23_RIGHTINVOL     1
#define DSK5510_AIC23_LEFTHPVOL      2
#define DSK5510_AIC23_RIGHTHPVOL     3
#define DSK5510_AIC23_ANAPATH        4
#define DSK5510_AIC23_DIGPATH        5
#define DSK5510_AIC23_POWERDOWN      6
#define DSK5510_AIC23_DIGIF          7
#define DSK5510_AIC23_SAMPLERATE     8
#define DSK5510_AIC23_DIGACT         9
#define DSK5510_AIC23_RESET          15
/* Frequency Definitions */
#define DSK5510_AIC23_FREQ_8KHZ      1
#define DSK5510_AIC23_FREQ_16KHZ     2
#define DSK5510_AIC23_FREQ_24KHZ     3
#define DSK5510_AIC23_FREQ_32KHZ     4
#define DSK5510_AIC23_FREQ_44KHZ     5
#define DSK5510_AIC23_FREQ_48KHZ     6
#define DSK5510_AIC23_FREQ_96KHZ     7
/* Codec Handle */
typedef int DSK5510_AIC23_CodecHandle;
/* Parameter Structure for the DSK5510 AIC23 Codec */
typedef struct DSK5510_AIC23_Config {
   int regs[DSK5510_AIC23_NUMREGS];
} DSK5510_AIC23_Config;
#define DSK5510_AIC23_DEFAULTCONFIG { \
```

```
0x0017, /* Set-Up Reg 0       Left line input channel volume control */
         /* LRS     0         simultaneous left/right volume: disabled */
         /* LIM     0         left line input mute: disabled */
        /* XX      00        reserved */\
        /* LIV     10111     left line input volume: 0 dB */\
0x0017, /* Set-Up Reg 1       Right line input channel volume control */ \
         /* RLS     0         simultaneous right/left volume: disabled */\
         /* RIM     0         right line input mute: disabled */        \
         /* XX      00        reserved */                               \
        /* RIV     10111     right line input volume: 0 dB */
0x008d, /* Set-Up Reg 2       Left channel headphone volume control */  \
         /* LRS     0         simultaneous left/right volume: enabled */ \
         /* LZC     1         left channel zero-cross detect: enabled */ \
        /* LHV     1011000    left headphone volume: 0 dB */           \
0x008d, /* Set-Up Reg 3       Right channel headphone volume control */ \
        /* RLS     0      simultaneous right/left volume: enabled */ \
        /* RZC     1     right channel zero-cross detect: enabled */\
       /* RHV     1011000    right headphone volume: 0 dB */          \
0x0011, /* Set-Up Reg 4       Analog audio path control */             \
         /* X       0        reserved */                              \
         /* STA     00        sidetone attenuation: -6 dB */          \
         /* STE     0        sidetone: disabled */                    \
        /* DAC     1         DAC: selected */                         \
        /* BYP     0        bypass: off */                            \
        /* INSEL   0         input select for ADC: line */           \
        /* MICM    0         microphone mute: disabled */            \
       /* MICB    1         microphone boost: enabled */             \
0x0000, /* Set-Up Reg 5       Digital audio path control */           \
         /* XXXXX   00000      reserved */                           \
         /* DACM    0         DAC soft mute: disabled */             \
         /* DEEMP   00        deemphasis control: disabled */        \
        /* ADCHP   0         ADC high-pass filter: disabled */       \
0x0000, /* Set-Up Reg 6       Power down control */                   \
         /* X       0        reserved */                             \
         /* OFF     0         device power: on (i.e. not off) */     \
         /* CLK     0         clock: on */                           \
        /* OSC     0         oscillator: on */                       \
        /* OUT     0         outputs: on */                          \
        /* DAC     0         DAC: on */                              \
        /* ADC     0         ADC: on */                              \
        /* MIC     0         microphone: on */                       \
        /* LINE    0         line input: on */                       \
0x0043, /* Set-Up Reg 7       Digital audio interface format */      \
         /* XX      00        reserved */                            \
         /* MS      1         master/slave mode: master */           \
        /* LRSWAP  0         DAC left/right swap: disabled */       \
        /* LRP     0         DAC lrp: MSB on 1st BCLK */            \
        /* IWL     00        input bit length: 16 bit */            \
```

```
            /* FOR     11        data format: DSP format */              \
    0x0081, /* Set-Up Reg 8      Sample rate control */                  \
            /* X       0         reserved */                            \
            /* CLKOUT  1         clock output divider: 2 (MCLK/2) */     \
            /* CLKIN   0         clock input divider: 2 (MCLK/2) */      \
            /* SR,BOSR 00000     sampling rate: ADC  48 kHz DAC  48 kHz */ \
            /* USB/N   1         clock mode select (USB/normal): USB */  \
    0x0001  /* Set-Up Reg 9      Digital interface activation */         \
            /* XX..X   00000000  reserved */                           \
            /* ACT     1         active */                             \
}
/* Set codec register regnum to value regval */
void DSK5510_AIC23_rset(DSK5510_AIC23_CodecHandle hCodec, Uint16 regnum,
Uint16 regval);
/* Return value of codec register regnum */
Uint16 DSK5510_AIC23_rget(DSK5510_AIC23_CodecHandle hCodec, Uint16 regnum);
/* Open the codec with id and return handle */
DSK5510_AIC23_CodecHandle DSK5510_AIC23_openCodec(int id,
DSK5510_AIC23_Config *Config);
/* Close the codec */
void DSK5510_AIC23_closeCodec(DSK5510_AIC23_CodecHandle hCodec);
/* Configure the codec register values */
void DSK5510_AIC23_config(DSK5510_AIC23_CodecHandle hCodec,
DSK5510_AIC23_Config *Config);
/* Write a 16-bit value to the codec */
CSLBool DSK5510_AIC23_write16(DSK5510_AIC23_CodecHandle hCodec, Int16 val);
/* Write a 32-bit value to the codec */
CSLBool DSK5510_AIC23_write32(DSK5510_AIC23_CodecHandle hCodec, Int32 val);
/* Read a 16-bit value from the codec */
CSLBool DSK5510_AIC23_read16(DSK5510_AIC23_CodecHandle hCodec, Int16 *val);
/* Read a 32-bit value from the codec */
CSLBool DSK5510_AIC23_read32(DSK5510_AIC23_CodecHandle hCodec, Int32 *val);
/* Set the codec output gain */
void DSK5510_AIC23_outGain(DSK5510_AIC23_CodecHandle hCodec, Uint16 outGain);
/* Set the codec loopback mode */
void DSK5510_AIC23_loopback(DSK5510_AIC23_CodecHandle hCodec, CSLBool
mode);
/* Enable/disable codec mute mode */
void DSK5510_AIC23_mute(DSK5510_AIC23_CodecHandle hCodec, CSLBool mode);
/* Enable/disable codec powerdown modes for DAC, ADC */
void DSK5510_AIC23_powerDown(DSK5510_AIC23_CodecHandle hCodec, Uint16 sect);
/* Set the codec sample rate frequency */
void DSK5510_AIC23_setFreq(DSK5510_AIC23_CodecHandle hCodec, Uint32 freq);
#ifdef __cplusplus
}
#endif
#endif
 *  ======== tone.c ========
```

```
 */
/*
 *  DSP/BIOS is configured using the DSP/BIOS configuration tool.  Settings
 *  for this example are stored in a configuration file called tone.cdb.  At
 *  compile time, Code Composer will auto-generate DSP/BIOS related files
 *  based on these settings.  A header file called tonecfg.h contains the
 *  results of the autogeneration and must be included for proper operation.
 *  The name of the file is taken from tone.cdb and adding cfg.h.
 */
#include "tonecfg.h"
#include "Fil_h.h"
#include "math.h"
#include "tms320.h"
#include "dsplib.h"
/*
 *  The 5510 DSK Board Support Library is divided into several modules, each
 *  of which has its own include file.  The file dsk5510.h must be included
 *  in every program that uses the BSL.  This example also includes
 *  dsk5510_aic23.h because it uses the AIC23 codec module.
 */
#include "dsk5510.h"
#include "dsk5510_aic23.h"

/* Length of sine wave table */

#define TRUE 1
/* Codec configuration settings */
DSK5510_AIC23_Config config = {
   0x0017, // 0 DSK5510_AIC23_LEFTINVOL  Left line input channel volume
   0x0017, // 1 DSK5510_AIC23_RIGHTINVOL Right line input channel volume
   0x008d, // 2 DSK5510_AIC23_LEFTHPVOL  Left channel headphone volume
   0x008d, // 3 DSK5510_AIC23_RIGHTHPVOL Right channel headphone volume
   0x0011, // 4 DSK5510_AIC23_ANAPATH    Analog audio path control
   0x0000, // 5 DSK5510_AIC23_DIGPATH    Digital audio path control
   0x0000, // 6 DSK5510_AIC23_POWERDOWN  Power down control
   0x0043, // 7 DSK5510_AIC23_DIGIF      Digital audio interface format
   0x0081, // 8 DSK5510_AIC23_SAMPLERATE Sample rate control
   0x0001  // 9 DSK5510_AIC23_DIGACT     Digital interface activation
};
/*
 *  main() - Main code routine, initializes BSL and generates tone
 */
int i,a;
float out;
Int16 x;
Int16 inp[16];
void main()
{
```

```
   DSK5510_AIC23_CodecHandle hCodec;


Int16 *adr=&x;

    /* Initialize the board support library, must be called first */
   DSK5510_init();

   /* Start the codec */
   hCodec = DSK5510_AIC23_openCodec(0, &config);

  DSK5510_AIC23_setFreq(hCodec, 5);

   while (TRUE)
    {

     while(!DSK5510_AIC23_read16(hCodec, adr));

       a=15;
           while (a!=-1) /*цикл сдвига данных в буфере на 1 адрес вверх*/
      {
       inp[a]=inp[a-1];
       a--;
      }
            inp[0]=*adr;
            i=14;
      out=0;
     while (i!=0) /* перемножение буфера входных данных
     на коэффициенты и формирование одного элемента
     буфера выходных данных*/
      {
      out=out+inp[i]*kaf[i];
      i--;
      }
          x=out;
/*                  x=*adr;*/
          while (!DSK5510_AIC23_write16(hCodec, x));
          while (!DSK5510_AIC23_write16(hCodec, x));
    }
    /* Close the codec */
    DSK5510_AIC23_closeCodec(hCodec);
}
```

СОДЕРЖАНИЕ

Учебное издание

ЧЕРТКОВ Виктор Михайлович
МАЛЬЦЕВ Сергей Васильевич
КОЦ Дмитрий Леонидович

ВЫЧИСЛИТЕЛЬНЫЕ И МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА

Методические указания
к выполнению курсового проекта
для студентов специальности 1-39 01 01 «Радиотехника»

Редактор *Т. А. Дарьянова*

Подписано в печать 10.02.10. Формат 60x84 1/16. Гарнитура Таймс. Бумага офсетная. Ризография. Усл. печ. л. 2,55. Уч.-изд. л. 2,18. Тираж 55 экз. Заказ 263.

Издатель и полиграфическое исполнение –
учреждение образования «Полоцкий государственный университет»

ЛИ № 02330/0548568 от 26.06.09 ЛП № 02330/0494256 от 27.05.09

211440, г. Новополоцк, ул. Блохина, 29