**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ**

Кафедра «Многоканальные телекоммуникационные системы»

СОГЛАСОВАНО:
Руководитель ЦЗОПБ _____ Е.А. Шиховец

# Учебно – методическое пособие

по дисциплине

# МИКРОПРОЦЕССОРНАЯ ТЕХНИКА В ИНФОКОММУНИКАЦИЯХ

для направления 11.03.02 «Инфокоммуникационные технологии и системы связи»
Центра заочного обучения по программам бакалавриата

Составитель:
С.С. Шаврин _____

Рассмотрено и одобрено
на заседании кафедры МТС
Протокол № _____
Дата _____
Зав. кафедрой _____ Н.А. Шишова

Москва 2022

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**
Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования
**Московский технический университет связи и информатики**

Кафедра многоканальных телекоммуникационных систем

# Учебно-методическое пособие по дисциплине «Микропроцессорная техника в инфокоммуникациях»

## для направления 11.03.02

Шаврин С.С.

Москва 2022

# УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

по дисциплине

# МИКРОПРОЦЕССОРНАЯ ТЕХНИКА В ИНФОКОММУНИКАЦИЯХ

Автор-составитель: д.т.н., профессор каф. МТС С.С. Шаврин

Дисциплина «Микропроцессорная техника в инфокоммуникациях» реализуется в соответствии с требованиями ФГОС, ОПОП ВО и Учебного плана по направлению 11.03.02 «Инфокоммуникационные технологии и системы связи». Дисциплина «Микропроцессорная техника в инфокоммуникациях» изучается студентами в 7 семестре по профилю бакалавриата «Транспортные сети и системы связи». В учебно-методическое пособие включены материалы, необходимые для выполнения курсовой работы, и лабораторных работ, которые предусмотрены учебным планом. Содержательная часть пособия посвящена использованию цифровых технологий в реализации фильтров, при этом изучение материала опирается на элементную базу сигнальных процессоров, и, в частности, на процессор ADSP-2181, архитектура и системы команд которого ориентированы именно на разработку цифровых фильтров. На примерах рассматривается порядок реализации цифровых фильтров в среде разработки VisualDSP++, который может помочь получить положительный результат максимально эффективно при выполнении курсовой работы по дисциплине.

## СОДЕРЖАНИЕ

**Введение** ..... **5**

**1. Рекомендуемая литература для студентов……..……6**

**2. Задание на курсовую работу…….…….7**

**3. Порядок выполнения курсовой работы……..……7**

**4. Требования к оформлению курсовой работы…….9**

**5. Цифровые фильтры и их реализация средствами микропроцессорной техники……..9**

**5.1. Частотная фильтрация сигналов** ..... **11**

**5.2. Фильтры с конечной импульсной характеристикой**……...……**13**

**5.2.1 Расчет значений отсчетов импульсной характеристики КИХ-фильтра** ..... **15**
**5.2.2 Разработка алгоритма работы фильтра и отладочной программы на языке ассемблера процессора ADSP-2181** ..... **21**
**5.2.3 Отладка программы** ..... **33**
**5.2.4 Итоговая программа фильтра** ..... **36**

**5.3. Фильтры с бесконечной импульсной характеристикой**…….……**43**

**5.3.1 Биквадратное звено**..... **45**
**5.3.2 Особенности реализации БИХ-фильтров на сигнальных процессорах** ..... **46**

**5.4. Специальные виды фильтров…….……...…53**
**5.4.1 Фильтр Герцеля**…….……..……..**54**
**5. 4.2 Фильтр Найквиста**..... **….58**

**ПРИЛОЖЕНИЕ А……...62**
**ПРИЛОЖЕНИЕ Б……..……63**

**Список использованной литературы** ..... **64**

**Введение**

Целями освоения дисциплины является формирование способностей применять современную микропроцессорную элементную базу, системы команд сигнального процессора и средства отладки программ для реализации алгоритмов обработки информации в инфокоммуникационных приложениях, предлагая новые идеи и подходы к решению задач.

Рассматриваются основные методы и алгоритмы цифровой обработки сигналов в инфокоммуникациях средствами микропроцессорной техники. Анализируются вычислительная сложность алгоритмов обработки информации и возможности эффективного использования современных микроконтроллеров, сигнальных и специализированных микропроцессоров в инфокоммуникационных сетях, системах и устройствах. Изучаются принципы построения, технические характеристики и конструктивные особенности микропроцессорных систем.

Дисциплина «Микропроцессорная техника в инфокоммуникациях» включена в часть, формируемую участниками образовательных отношений блока дисциплин учебного плана (блок Б1.В.20). Дисциплина «Микропроцессорная техника в инфокоммуникациях» реализуется в соответствии с требованиями ФГОС, ОПОП ВО и Учебного плана по направлению 11.03.02 «Инфокоммуникационные технологии и системы связи».

Дисциплина «Микропроцессорная техника в инфокоммуникациях» изучается студентами в 7 семестре по профилю бакалавриата «Транспортные сети и системы связи». Дисциплина включает лекции, практические занятия и лабораторные работы. Предусмотрена самостоятельная работа, в том числе значительный объем – на выполнение курсовой работы. Промежуточная аттестация предусматривает курсовую работу и экзамен в 7 семестре при заочной форме обучения.

Для успешного усвоения данной дисциплины необходимо, чтобы студент владел знаниями, умениями и компетенциями, сформированными в процессе изучения дисциплин: «Общая теория связи», «Схемотехника», «Вычислительная техника», «Дискретная математика», «Инфокоммуникационные системы и сети».

## 1. Рекомендуемая литература для студентов

### Основная литература


1. *Шаврин С.С., Зуйкова Т.Н., Мусатова О.Ю. Техника микропроцессорных систем в инфокоммуникационных приложениях. Реализация цифровых фильтров на базе сигнальных процессоров. Учебное пособие // М.: МТУСИ, 2021.*
2. *Лайонс Р. Цифровая обработка сигналов: Второе издание. Пер. с англ. – М.: ООО «Бином – Пресс», 2011. – 656 с.*
3. *Руководство пользователя по сигнальным микропроцессорам семейства ADSP-2100. Под ред. А.Д.Викторова. Пер. с англ. – М.: СПбГЭТУ, 1997.– 520 с.*
4. *Шаврин С.С., Мельник С.В. Учебное пособие по курсу «Технологии микропроцессорных систем в инфокоммуникациях» для магистров по направлению 11.04.02 «Инфокоммуникационные технологии и системы связи» // М.: МТУСИ, 2020.*
5. *Шаврин С.С., Климов Д.А. Реализация базовых операций защиты информации на сигнальных процессорах (часть 1) Учебное пособие.– М.: МТУСИ, 2012.*


### Дополнительная литература


1. *Шаврин С.С., Мельник С.В. Учебное пособие по курсу «Цифровые системы передачи и методы их защиты» для бакалавров по направлению 11.03.02 «Инфокоммуникационные технологии и системы связи», профиль «Защищенные системы связи» // М.: МТУСИ, 2020.*
2. *Шаврин С.С. Реализация базовых операций защиты информации на сигнальных процессорах Ч. 2. Учебное пособие по курсу «Многоканальные системы передачи и средства их защиты» для бакалавров по направлению 11.03.02 // М.: МТУСИ, 2016.*
3. *Гордиенко В.Н., Тверецкий М.С. Многоканальные телекоммуникационные системы. Учебник для вузов. – 2-е издание, испр. и доп. // Горячая линия - Телеком, 2013.*
4. *Мельник С.В., Зуйкова Т.Н., Мусатова О.Ю., Шаврин С.С. Исследование эффектов ограничения разрядной сетки при цифровой обработке сигналов. Практикум // М.: МТУСИ, 2017.*
5. *Андреев И.И., Ланнэ А.А. MATLAB для DSP: SPTool - инструмент для расчёта цифровых фильтров и спектрального анализа сигналов // Цифровая обработка сигналов. 2000. № 2. С. 6–13.*

## 2. Задание на курсовую работу

Задание по курсовой работе выдается индивидуально и представляет собой графический вид амплитудно – частотной и фазо – частотной характеристик фильтра с конечной импульсной характеристикой, подлежащего разработке. Вид задания (пример) представлен на рисунке 2.1.

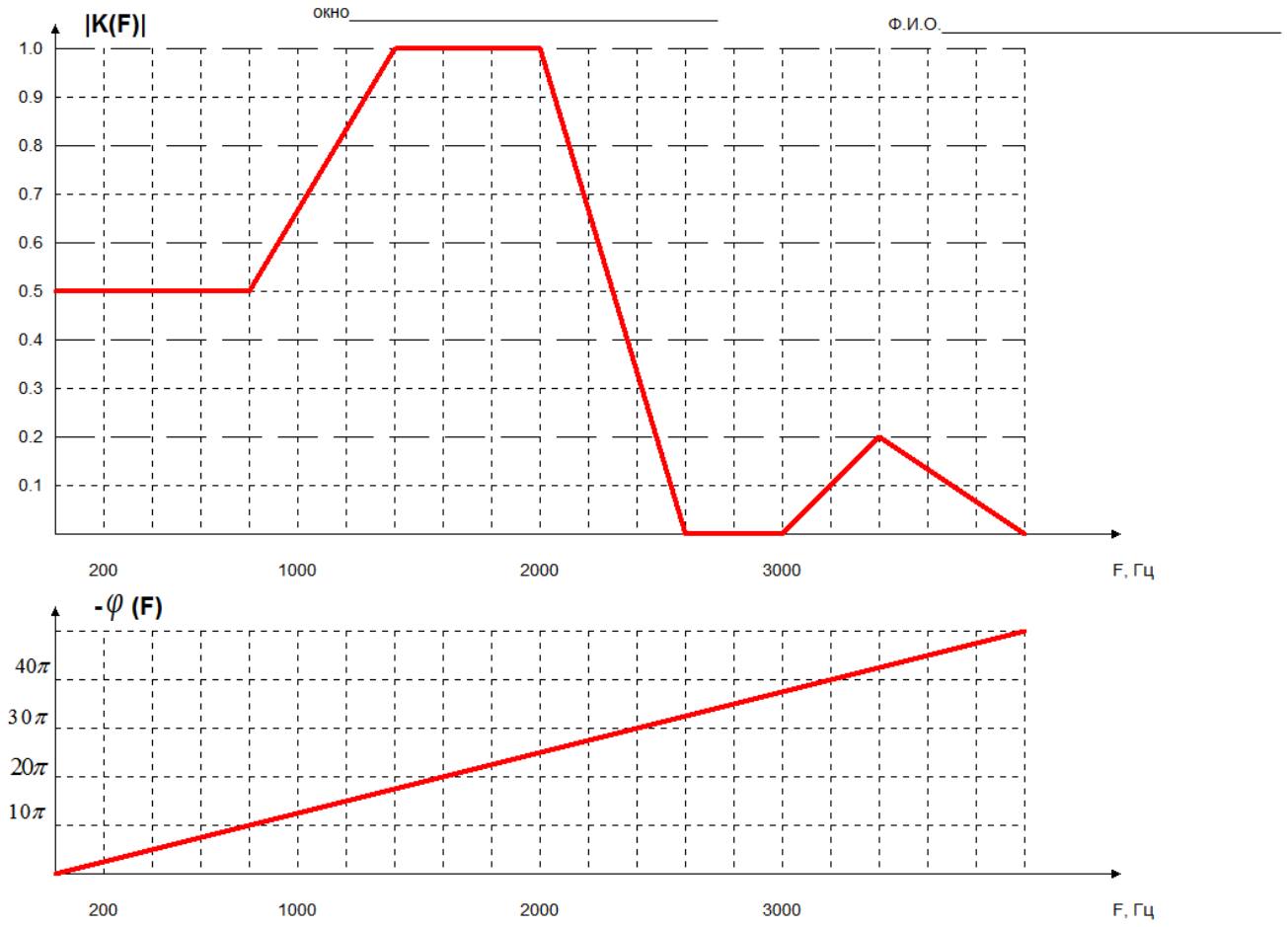



Рисунок 2.1. Вид примера индивидуального задания на курсовую работу

Как правило, вид характеристик задается отрезками прямых линий.

## 3. Порядок выполнения курсовой работы

1) Повторить материалы теоретической части курса; презентации и описание процессора МП.pdf выложены в папке «Методические материалы» в Электронном университете МТУСИ.

2) По представленному в Электронном университете МТУСИ индивидуальному заданию (элемент «Задания на КР» там же), включающему амплитудно – частотную (АЧХ) и фазо – частотную (ФЧХ) характеристики,

вычислить значения отсчетов импульсной характеристики проектируемого цифрового фильтра.

Обратить особое внимание на то, что ФЧХ представлена на графике с обратным знаком, на оси ординат указано –φ (минус фи).

Методика расчета представлена в файле filters.pdf в папке «Методические материалы» в Электронном университете МТУСИ.

3) Разработать программу работы фильтра в циклическом отладочном варианте (без привязки к таймерному прерыванию). Программа должна быть разработана на языке ассемблера процессора ADSP-2181.

4) Для проверки соответствия характеристик разработанного фильтра требованиям технического задания подготовить файлы, содержащие отрезки синусоидальных сигналов характерных частот, соответствующих точкам перегиба и серединам прямых отрезков АЧХ. Амплитуда сигналов разных частот должны быть одинаковыми и соответствовать 16-разрядному представлению. Для простоты желательно амплитуду сигнала выбрать в пределах 20000…30000.

Для формирования отрезков сигнала может быть использован звуковой редактор CoolEdit96, размещенный на сайте электронного университета в числе другого полезного программного обеспечения. Инструкция по работе с CoolEdit96 размещена в файле 110302_Исследование…pdf в папке «Методические материалы» в Электронном университете МТУСИ.

5) Провести моделирование работы созданной программы на подготовленных отрезках сигнала. Сохранить скриншоты сигналов на входе и выходе фильтра для каждого эксперимента.

6) Подготовить отрезок сигнала с частотой, значение которой отличается от одной из ранее подготовленных для максимального коэффициента передачи на 50 Гц.

7) Провести моделирование работы созданной программы на этом отрезке сигнала и оценить влияние эффекта Гиббса.

8) Дополнить разработанную программу привязкой к таймерному прерыванию с интервалом дискретизации 125 мкс и проверить ее работу.

## 4. Требования к оформлению курсовой работы

Оформить отчет по курсовой работе. Отчет должен содержать титульный лист, оглавление, описательную часть работы (включая блок-схемы алгоритмов и тексты программ, а также результаты моделирования – скриншоты сигналов на входе и выходе фильтра), выводы и список использованных литературных источников.

Пример титульного листа пояснительной записки к контрольной работе представлен в Приложении А.

Блок-схемы алгоритмов оформляются в соответствии с ГОСТ 19.701-90. Правила оформления блок-схем алгоритмов представлены в Приложении Б.

## 5. Цифровые фильтры и их реализация средствами микропроцессорной техники

Микропроцессорная элементная база является современным средством реализации цифровых технологий в инфокоммуникационных приложениях. Цифровые технологии в инфокоммуникациях реализуют все основные функции, на основе которых строятся цифровые системы передачи – от цифровой фильтрации и спектроанализа до поддержки сетевых протоколов и оценки качества передачи информации.

Разработка конкурентоспособных средств техники инфокоммуникаций ориентирована в настоящее время на использование элементной базы сверхбольшой степени интеграции – сверхбольших интеграционных схем (СБИС). СБИС развиваются параллельно в двух основных направлениях – программируемых логических интегральных схем (ПЛИС) и микропроцессоров.

ПЛИС представляют собой некоторый набор базовых элементов, на основе которых могут быть реализованы цифровые системы с различной архитектурой и различной разрядностью данных на разных этапах обработки, что позволяет разработчику оптимизировать процесс обработки под решение конкретной задачи. В совокупности с возможностью распараллеливания процесса обработки это обеспечивает возможность достижения рекордно высокой производительности и эффективности обработки информации.

Микропроцессоры предполагают наличие некоторой архитектуры, которая, с одной стороны, ограничивает возможности разработчика в плане

разрядности обработки и системы команд, а с другой – в значительной степени облегчает и ускоряет процесс разработки.

Различают три основных направления разработки микропроцессорных средств – микроконтроллеры (МК), цифровые процессоры для обработки сигнала или сигнальные процессоры (СП), и различные специализированные процессоры; архитектура этих средств ориентирована на решение характерного круга задач. Широко распространены два типа архитектуры вычислительного ядра микропроцессоров – фон-Неймановская и гарвардская. Фон-Неймановская архитектура, используя общее адресное пространство запоминающего устройства для хранения и команд и операндов, ориентирована на универсальность решаемых задач за счет свободно распределяемых ресурсов, и предназначена главным образом для построения компьютеров.

Гарвардская архитектура ориентирована на обеспечение высокого быстродействия. Память программ и память данных в процессорах гарвардской архитектуры обычно логически и физически разделены и часто имеют разную разрядность. Сигнальные процессоры обычно строятся в соответствии с гарвардской архитектурой и ее модификациями [1, 7, 8].

Настоящее учебное пособие посвящено одной из главных проблем цифровой обработки сигналов – цифровой фильтрации. В соответствии с особенностями решения этой проблемы средствами микропроцессорной техники в качестве элементной базы выбран сигнальный процессор ADSP-2181, построенный по модифицированной пятишинной гарвардской архитектуре.

Основными методическими предпосылками выбора этого типа процессора явились:

- архитектура, специально ориентированная на цифровую фильтрацию;

- интуитивно понятная система команд и язык Ассемблера, не требующие изучения и запоминания большого количества аббревиатур команд процессора и позволяющая студентам сосредоточить усилия на процесс обработки информации;

- удобный набор отладочных средств, включая аппаратный набор EZ-KIT, на базе которого могут быть реализованы конечные изделия;

- простота дальнейшей реализации готовых изделий вплоть до серийного выпуска за счет отсутствия специальных требований к трассировке печатных плат, поскольку все скоростные цепи сосредоточены внутри процессора.

В качестве языка программирования выбран язык Ассемблера процессоров семейства ADSP – 21XX. Такой выбор с методической точки зрения обеспечивает более подробное осознание и исследование студентами преобразований сигнала в процессе обработки, а с технической – достижение наиболее высокой эффективности решений.

В пособии рассмотрены основные типы частотных фильтров, особенности их реализации на сигнальных процессорах, представлены алгоритмы реализации и тексты программ с необходимыми комментариями.

Особые надежды авторы возлагают на комментарии, приведенные в текстах программ. Их внимательное прочтение на взгляд авторов, в значительной степени облегчит студентам разработку прикладных программ.

### 5.1. Частотная фильтрация сигналов.

Частотная фильтрация является едва ли не самой распространенной задачей обработки сигналов в инфокоммуникационных приложениях. Устройства, реализующие частотную фильтрацию сигналов, называются фильтрами. Как известно, фильтры относятся к линейным четырехполюсникам, характеристики передачи которых имеют значимую зависимость от частоты сигнала в пределах рабочего диапазона.

Для описания характеристик фильтров обычно используют частотное представление в виде комплексного коэффициента передачи $K(j\omega)$ либо временное представление в форме импульсной характеристики $g(t)$. Для описания дискретных фильтров чаще всего используют передаточную характеристику $H(z)$. Комплексный коэффициент передачи устанавливает связь между спектром сигнала на входе $X(j\omega)$ и выходе $Y(j\omega)$ фильтра.

$$Y(j\omega) = X(j\omega) * K(j\omega)$$

Во многих приложениях вместо комплексного коэффициента передачи используют амплитудно – частотную (АЧХ) $|K(\omega)|$ и фазо - частотную (ФЧХ) $\varphi(\omega)$ характеристики, совокупность которых можно рассматривать как отражение в полярных координатах комплексного коэффициента передачи, действительная и мнимая части которого определены в декартовой системе координат:

$$\begin{cases} \Re[K(j\omega)] = |K(\omega)| * cos\big(\varphi(\omega)\big) \\ \Im[K(j\omega)] = |K(\omega)| * sin\big(\varphi(\omega)\big) \end{cases} \quad (1)$$

Импульсная характеристика дает основание определить значение сигнала $y(t)$ в момент времени $t$ на выходе фильтра по текущему и предшествующим значениям входного сигнала $x(t)$:

$$y(t) = \int_{-\infty}^{t} x(\tau) * g(t-\tau)d\tau \qquad (2)$$

Передаточная характеристика дискретных фильтров задается в Z-плоскости и описывает влияние коэффициентов передачи задержанных сигналов $b_j$ прямой (нули) и $a_i$ обратной (полюса) связей:

$$H(z) = \frac{\sum_{j=0}^{M-1} b_j * z^{-j}}{1 - \sum_{i=0}^{N-1} a_i * z^{-i}}$$

Поскольку все эти описания относятся к одному и тому же объекту, их взаимосвязь определяется следующими соотношениями.

Комплексный коэффициент передачи и импульсная характеристика фильтра связаны между собой через прямое и обратное преобразование Фурье:

$$K(j\omega) = \int_{-\infty}^{\infty} g(t) * e^{-j\omega t}dt$$

$$g(t) = \frac{1}{4\pi}\int_{-\infty}^{\infty} K(j\omega) * e^{j\omega t}d\omega$$

Комплексный коэффициент передачи может быть вычислен на основе передаточной характеристики $H(z)$ подстановкой

$$e^{-j\omega T_д} \rightarrow z^{-j}$$

где $T_д = \frac{1}{F_д}$- период дискретизации, $F_Д$- частота дискретизации.

В технике связи для реализации фильтров в настоящее время используются как аналоговые, так и цифровые технологии. Как правило, аналоговые технологии используются на высоких частотах - там, где цифровые не в состоянии обеспечить необходимое быстродействие. При этом аналоговые технологии накладывают известные ограничения на функциональные возможности фильтров, что связано с самой природой аналоговых технологий, и, в частности, невозможности мгновенного изменения энергии, накопленной в реактивных элементах. Особенности этих ограничений в инфокоммуникационных приложениях будут рассмотрены далее.

Настоящее учебное пособие посвящено использованию цифровых технологий в реализации фильтров, при этом изучение материала опирается на элементную базу сигнальных процессоров. Альтернативным решением может

явиться реализация фильтров на программируемых логических интегральных схемах (ПЛИС). Каждая из этих видов элементной базы имеет свои особенности и область предпочтительного применения. Вопрос о выборе типа элементной базы выходит за рамки настоящего учебного пособия.

Цифровые технологии обеспечивают обход тех ограничений, которые характерны для аналоговых, открывая тем самым новые возможности частотной фильтрации сигналов. Одной из таких возможностей является разрыв взаимосвязи АЧХ и ФЧХ.

### 5.2. Фильтры с конечной импульсной характеристикой.

Фильтры с конечной импульсной характеристикой (КИХ-фильтры) базируются непосредственно на выражении (2) и реализуют вычисление свертки входного сигнала с импульсной характеристикой фильтра. Применительно к представлению в дискретном времени эти выражения трансформируются в:

$$y_i = \sum_{j=0}^{N-1} x_{i-j} * g_j \qquad (3)$$

Функциональная схема фильтра, реализующая выражение (3), представлена на рис. 5.1, где $z^{-1}$ символизирует задержку обрабатываемого сигнала на 1 период дискретизации.

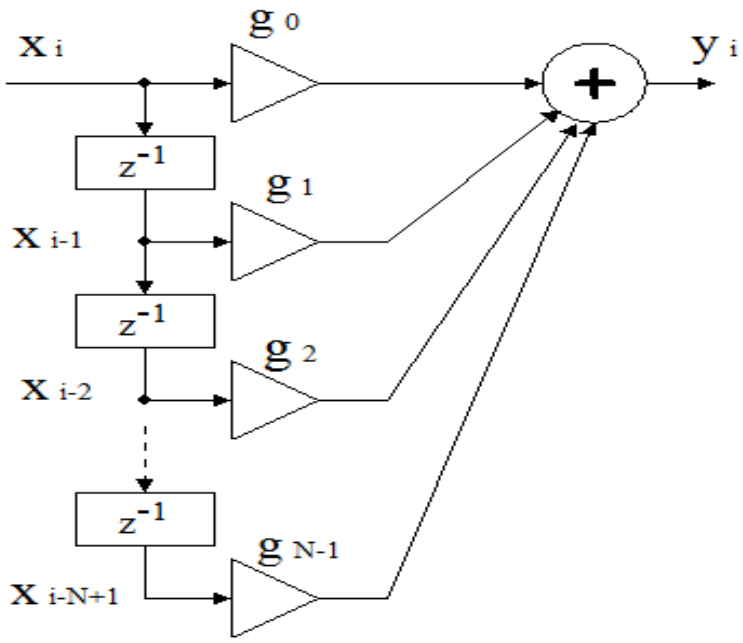



Рисунок 5.1. Функциональная схема КИХ - фильтра

Представленная архитектура косвенно иллюстрирует возможность занесения в коэффициенты $\mathbf{g_i}$ значений импульсной характеристики любой формы, а не только характерной для использующих реактивные элементы аналоговых фильтров. По этой причине КИХ архитектура дает возможности реализации фильтра с произвольной ФЧХ при произвольной АЧХ.

Указанные свойства архитектуры КИХ имеют весьма важные области применения в технике инфокоммуникаций.

Прежде всего, открывается возможность таким образом построить любой фильтр с линейной ФЧХ, минимизируя искажения сложных сигналов. Так, например, реализация на архитектуре КИХ фильтра Найквиста, обеспечивает важное для систем радиосвязи сужение излучаемой передатчиком полосы частот сложного фазо-манипулированного сигнала; при этом интермодуляционные искажения, возникающие за счет расширения символов во времени, нивелируются прохождением через ноль в отсчетные моменты времени соседних символов.

Еще одним важным свойством архитектуры КИХ является возможность ее использования для построения фильтров, согласованных с сигналом. При ограниченной мощности передатчика энергия символа может быть распределена во времени на $N$ символов. Прием такого сигнала на фильтр с согласованной импульсной характеристикой может повысить помехоустойчивость приема в $\sqrt{N}$ раз. В частности, подобный принцип используется для так называемого кодового разделения сигналов от разных источников при их одновременной передаче в одной полосе частот.

Важным достоинством КИХ-фильтров является их принципиальная устойчивость и жестко ограниченная длительность переходных процессов. Представленная рис. 5.1 архитектура КИХ далеко не единственная, с ее модификациями можно ознакомиться, например, в [2, 4, 6]. В то же время она проста в реализации и вполне жизнеспособна. Далее на примерах рассматривается порядок разработки цифровых КИХ-фильтров на базе сигнальных процессоров, который может помочь получить положительный результат максимально эффективно для реализации курсового задания по дисциплине «Технологии микропроцессорных систем в инфокоммуникациях».

### 5.2.1 Расчет значений отсчетов импульсной характеристики КИХ-фильтра.

В качестве элементной базы во всех рассматриваемых примерах выбран сигнальный процессор ADSP-2181, подробное описание которого представлено в [1, 8]. Для простоты понимания значение частоты дискретизации $F_д$ выбрано равным 8 кГц - привычным для студентов направлений 11.03.02 и 11.04.02. Алгоритм дискретного преобразования Фурье представлен почти на уровне псевдокода максимально просто с максимальным приближением к нему на языке Бейсик, и, таким образом, может быть перетранслировано на любой другой язык. Для наглядности выбран фильтр 79 порядка, $(N = 80)$, что обеспечивает разрешающую способность фильтра по частоте равной $\Delta f = F_д/N = 100$Гц. Кроме того, задание на проектирование представлено в привычной форме амплитудно-частотной и фазо-частотной характеристик, представленных, соответственно, на рис. 5.2 и рис. 5.3 сплошными линиями.

Проектирование целесообразно начинать с расчета комплексного коэффициента передачи в соответствии с выражением (1).

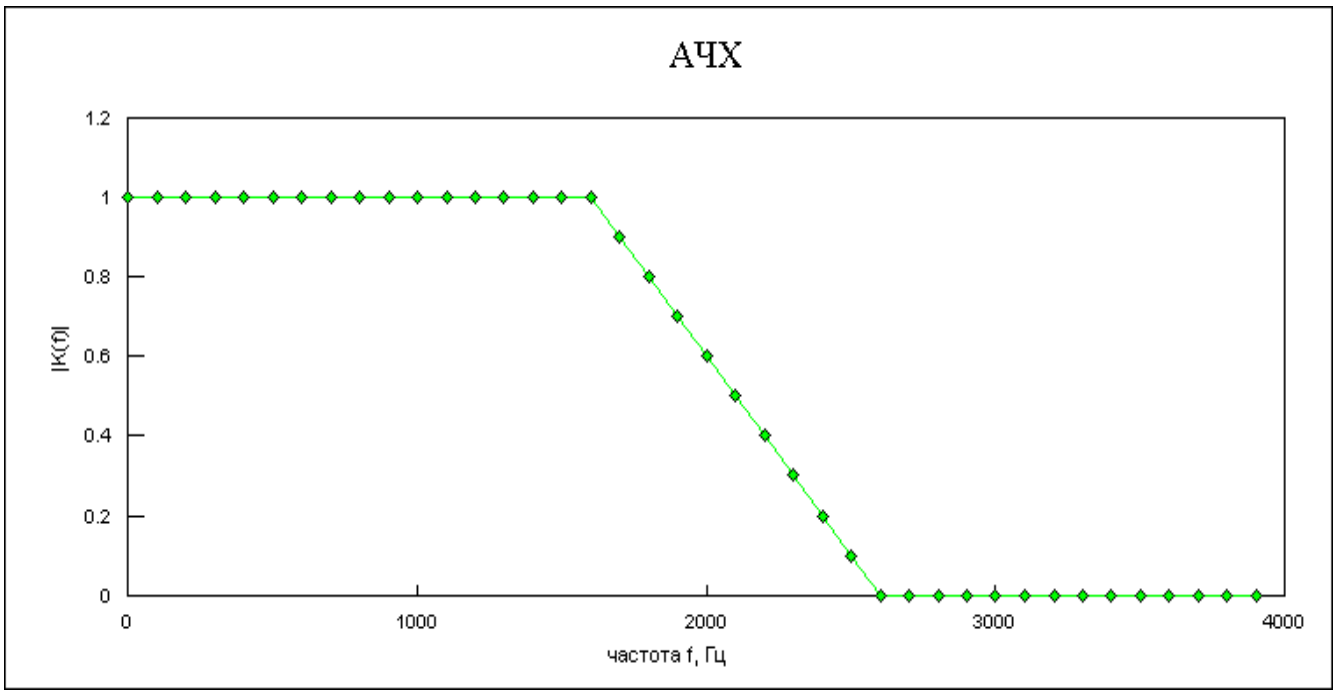



Рисунок 5.2. АЧХ фильтра.

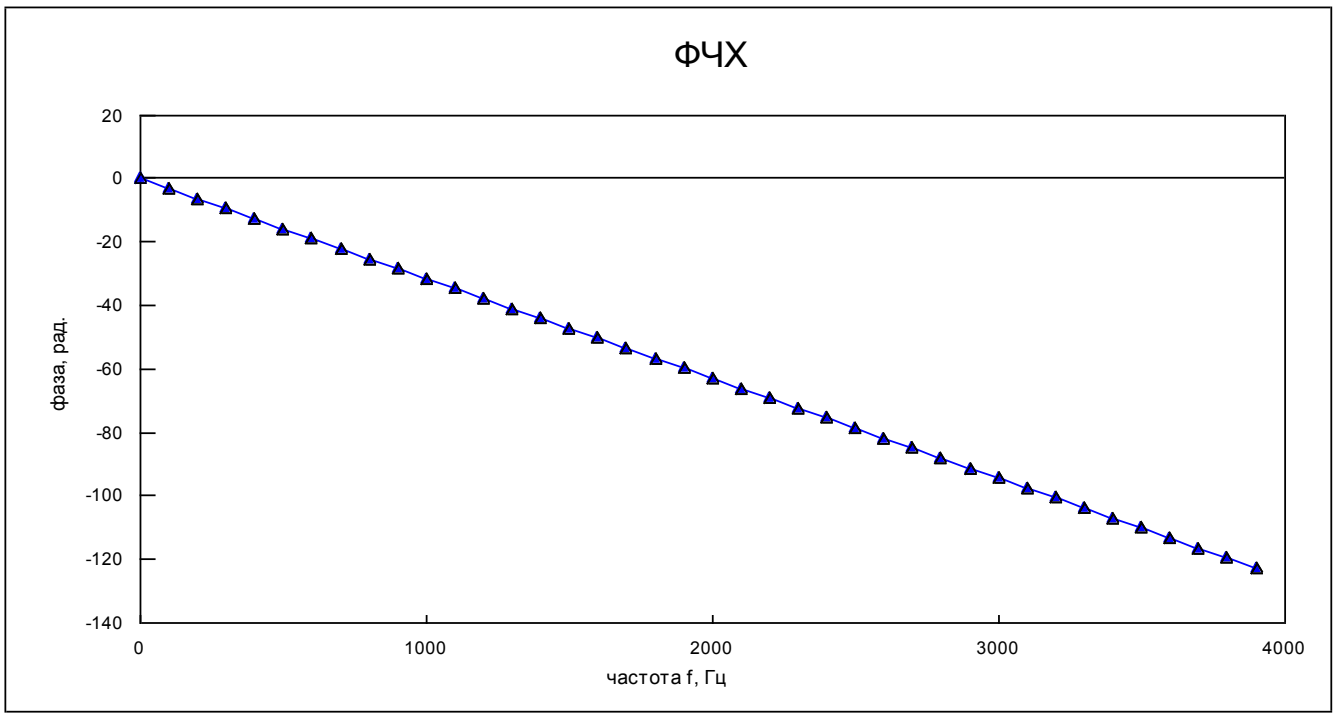



Рисунок 5.3. ФЧХ фильтра.

Забегая немного вперед, следует отметить, что вид ФЧХ выбран не случайно. Представленная форма ФЧХ является во многом типовой и соответствует постоянному для всех частот времени задержки сигнала при прохождении через фильтр (т.е. фактически линии задержки), равного

$\tau = \frac{T_д * N}{2} = \frac{N}{2 * F_д}$, что позволяет обеспечить минимальный характер искажений сложных сигналов, связанных с эффектом Гиббса.

Для автоматизации процессов расчета заданные АЧХ и ФЧХ целесообразно аппроксимировать:

$$|K(f)| = \begin{cases} 1{,}0 \le f < 1600 \\ 2.6 - f/1000, 1600 \le f < 2600 \\ 0{,}2600 \le f < 4000 \end{cases}$$

$$\varphi(f) = \frac{-\pi}{100} * f$$

Переходя к дискретным значениям частоты с шагом $\Delta f = F_д/N = 100$Гц, номера частот будут определены как: $n = f * N/F_д$, тогда

$$|K(n)| = \begin{cases} 1{,}0 \le n < 16 \\ 2.6 - f/100, 16 \le n < 26 \\ 0{,}26 \le n \le 39 \end{cases}$$

$$\varphi(n) = -\pi * n$$

Комплексный коэффициент передачи должен быть определен в диапазоне частот $0 \le f < F_д$ таким образом, чтобы в диапазоне $0 \le f < F_д/2$ его значения были определены по АЧХ и ФЧХ как:

$$\begin{cases} \Re[K(n)] = |K(n)| * cos[\varphi(n)] \\ \Im[K(n)] = |K(n)| * sin[\varphi(n)] \end{cases}, 0 \le n \le 39 \qquad (4)$$

На частоте, равной половине частоты дискретизации, $\Re[K(40)] = \Im[K(40)] = 0$.

В диапазоне $F_д/2 < f < F_д$ должны быть зеркально отражены значения (4) так, что:

$$\begin{cases} \Re[K(n)] = \Re[K(80-n)] \\ \Im[K(n)] = -\Im[K(80-n)] \end{cases}, 41 \le n \le 79 \qquad (5)$$

Последнее выражение (5) иллюстрирует в спектральном представлении отрицательные частоты в математическом понимании или нижнюю боковую полосу дискретного сигнала в понимании техники многоканальной связи.

Выражения (4) и (5) отражены метками на рис. 5.2 и рис. 5.3, а значения комплексного коэффициента передачи – на рис. 5.4: в диапазонах $0 \le f < F_д/2$ (а) и $0 \le f < F_д$ (б).

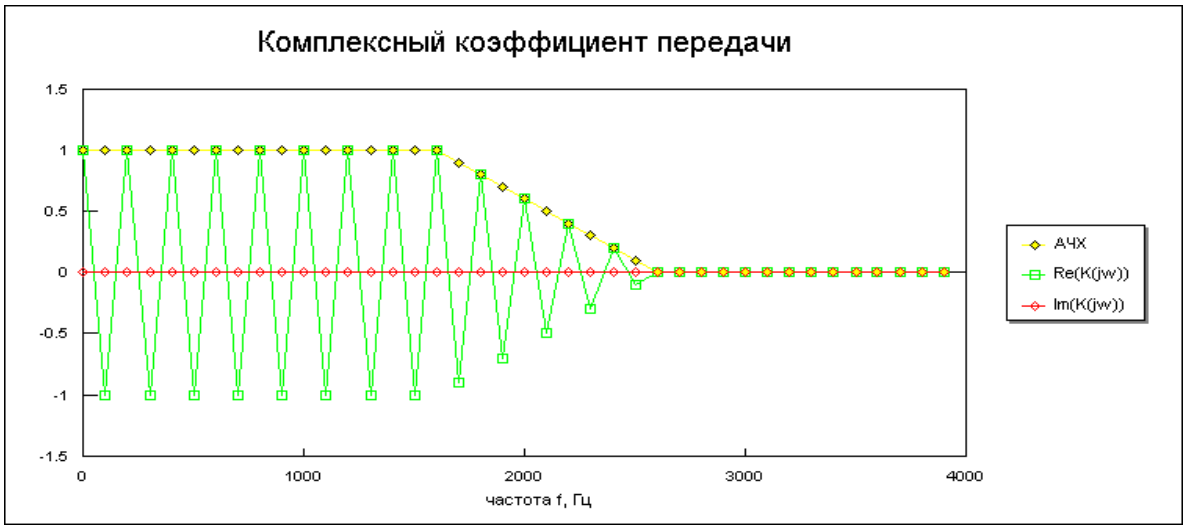



Рисунок 5.4 а

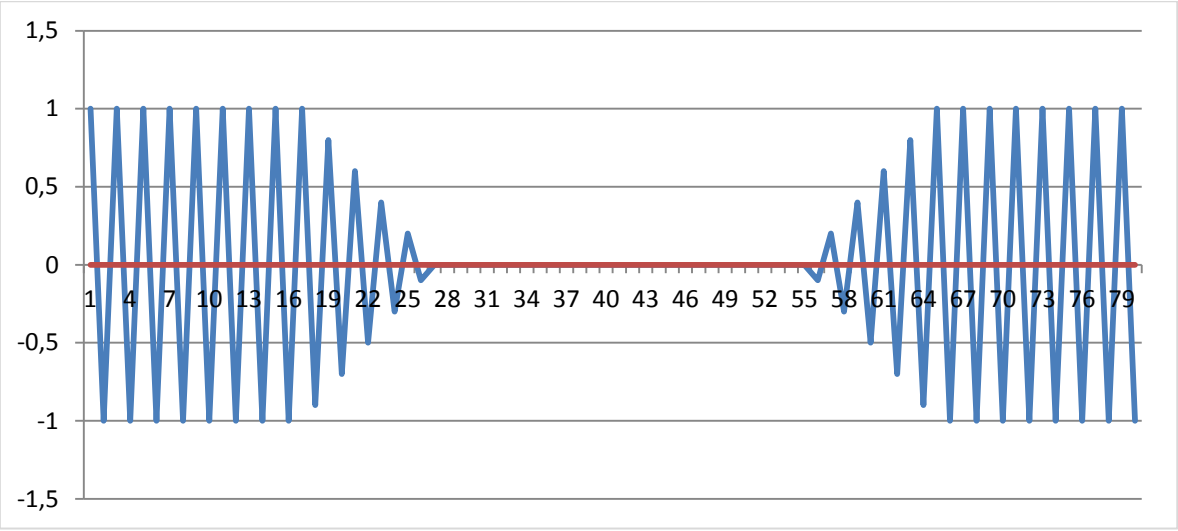



Рисунок 5.4 б. Комплексный коэффициент передачи.

Массивы значений результатов расчета комплексного коэффициента передачи в соответствии с рассмотренными правилами могут быть использованы для расчета импульсной характеристики фильтра $G(k)$; эта операция осуществима с помощью дискретного обратного преобразования Фурье. Это преобразование может быть описано выражением:

$$G(k) = \frac{1}{N}\sum_{n=0}^{N-1} K(n) * e^{j2\pi nk/N}$$

К сожалению, использование каких-то готовых стандартных средств вычисления дискретного преобразования Фурье может оказаться весьма затруднительным, поскольку в стандартных языковых пакетах обычно реализуются быстрые алгоритмы его вычисления [9]. Главной особенностью таких алгоритмов является кратность размера блока преобразования степени двойки или даже степени четверки. Переход к таким размерам массивов при образовательном ознакомительном подходе к проектированию фильтров чреват потерей наглядности и в рамках настоящего учебного пособия не представляется оправданным.

Алгоритм дискретного преобразования Фурье, который предлагается использовать в рассматриваемом примере, представлен на рис. 5.5 в форме программы на языке Матлаб, изучаемом в рамках дисциплины «Вычислительная техника» на младших курсах.

```
%*********************************************
%   Программа вычисления 80 значений отсчетов
%   импульсной характеристики цифрового
%   КИХ-фильтра 79 порядка по заданным
%   амплитудно - частотной и фазо - частотной
%   характеристикам, которые должны быть до
%   выполнения программы сформированы в виде
%   массивов (матриц):
%   ach(40,1) - амплитудно - частотная;
%   fch(40,1) - фазо - частотная;
%---------------------------------------------
%
%   формирование отсчетов
%   комплексного коэффициента передачи
%   в диапазоне от 0 до Fд/2
%
re_sig=zeros(80,1);
im_sig=zeros(80,1);
for k=1:40
    re_sig(k)=ach(k).*cos(fch(k));
    im_sig(k)=ach(k).*sin(fch(k));
end
%
%   формирование отсчетов
%   комплексного коэффициента передачи
%   в диапазоне от Fд/2 до Fд
%
for k=2:40
    re_sig(82-k)=re_sig(k);
    im_sig(82-k)=-im_sig(k);
end
%
%   графическое построение
%   комплексного коэффициента передачи
%   на графиках красные линии соответствуют
%   действительной части, а синие - мнимой
%
k=1:80;
plot(k,re_sig,'r')
hold on
plot(k,im_sig,'b')
%---------------------------------------------
%
%   преобразование Фурье
%   с приведением результатов вычисления
%   отсчетов импульсной характеристики
%   к формату, предназначенному для подключения
%   к программе реализации цифрового фильтра
```

Рисунок 5.5 начало. Обратное преобразование Фурье (программа).

```
%   на базе 16-разрядных процессоров (ADSP-2181)
%
re_spec=zeros(80,1);
im_spec=zeros(80,1);
for k=1:80 %n
    w=2.*pi.*(k-1)./80;
    x=0;
    y=0;
    for j=1:80
        x=x+re_sig(j,1).*cos(w.*(j-1))+im_sig(j,1).*sin(w.*(j-1));
        y=y-re_sig(j,1).*sin(w.*(j-1))+im_sig(j,1).*cos(w.*(j-1));
    end
    re_spec(k,1)=x./80.*32768;
    im_spec(k,1)=y./80.*32768;
end
%-----------------------------------------------
%
%   графическое построение
%   отсчетов импульсной характеристики
%
y1=figure;
jjj=1:80;
    plot(jjj-1,re_spec(jjj),'r')
    hold on
    plot(jjj-1,im_spec(jjj),'b')
hold on
grid on
xlabel i
ylabel g(i)
%
%   вывод результатов вычисления
%   отсчетов импульсной характеристики
%   в формате, предназначенном для подключения
%   к программе реализации цифрового фильтра
%   на базе 16-разрядных процессоров (ADSP-2181)
%
q='g_re.dat';   % имя действительной части
[F1, mes]=fopen(q,'w');
fprintf(F1,'%d\n ',round(re_spec));
fclose(F1);
q='g_im.dat';   % имя мнимой части
[F2, mes]=fopen(q,'w');
fprintf(F2,'%d\n ',round(im_spec));
fclose(F2);
%***********************************************
```

Рисунок 5.5 окончание.

В соответствии с правилами умножения комплексных чисел:

$$(a + jb) * (c + jd) = (ac - bd) + j(ad + bc)$$

во внутреннем цикле для каждой частоты $k$ вычисляется действительная $\Re[G(k)]$ и мнимая $\Im[G(k)]$ части импульсной характеристики фильтра, соответственно, как:

$$\begin{cases} \Re[G(k)] = \sum_{n=0}^{79} \left[ \Re[K(n)] * cos\frac{2\pi nk}{80} + \Im[K(n)] * sin\frac{2\pi nk}{80} \right] \\ \Im[G(k)] = \sum_{n=0}^{79} \left[ \Im[K(n)] * cos\frac{2\pi nk}{80} - \Re[K(n)] * sin\frac{2\pi nk}{80} \right] \end{cases}$$

Внешний цикл повторяет эти вычисления для $k = 0 \dots 79$.

Выходной информацией результата преобразования являются файлы g_re.dat отсчетов действительной и g_im.dat мнимой частей импульсной характеристики фильтра, отсчеты которой масштабируются в диапазоне +32767…-32768 и переводятся в целочисленный формат. В представленном примере отсчеты импульсной характеристики фильтра выводятся в файл g_re.dat для дальнейшего использования в программе фильтра для сигнального процессора.

Помимо действительной, в результате выполнения программы формируется файл мнимой части импульсной характеристики фильтра. При правильном выполнении расчетов значения ее отсчетов не должны превышать 0, в этом следует убедиться перед дальнейшими расчетами.

Результаты расчета отсчетов импульсной характеристики фильтра по рассматриваемому примеру представлены в виде графика на рис. 5.6.

Следующим этапом проектирования фильтра является разработка алгоритма его работы на сигнальном процессоре ADSP-2181 с учетом возможностей последнего и составление на языке ассемблера ADSP-2181 так называемой отладочной программы.

5.2.2. Разработка алгоритма работы фильтра и отладочной программы на языке ассемблера процессора ADSP-2181.

Для облегчения восприятия последующего материала целесообразно восстановить в памяти основные сведения по архитектуре и системе команд сигнального процессора ADSP-2181, и, в частности, такие узлы, как арифметико-логическое устройство, умножитель, сдвигатель, кольцевая косвенная поставтомодифицирующая адресация, тему прерываний, а также правила составления программ на языке ассемблера процессора ADSP-2181.

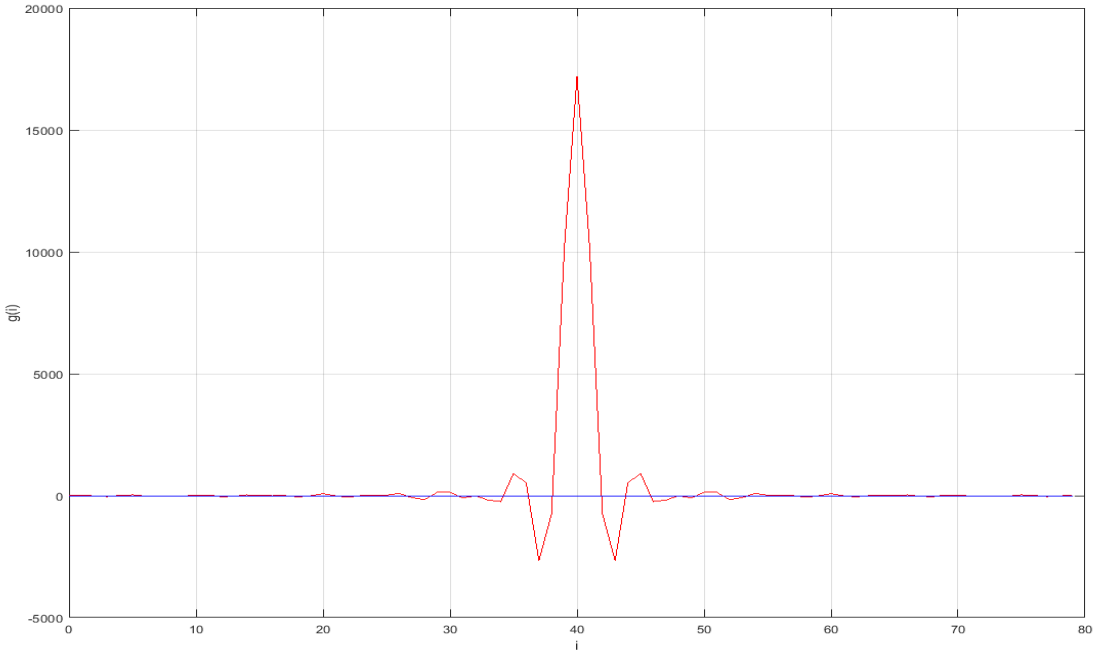



Рисунок 5.6. Импульсная характеристика фильтра (теоретический расчет).

Программа, создаваемая на языке ассемблера, обычно компонуется из модулей, - функционально законченных блоков, допускающих индивидуальную отладку. Файлы, содержащие модули программы или всю программу целиком, написанные на языке Ассемблера ADSP-2181 VisualDSP++, должны иметь расширение .asm.

В рамках настоящего пособия при работе в среде VisualDSP++ программа на языке ассемблера создается как единое целое (без деления на модули), она требует обязательного соответствующего оформления, а также последующих трансляции и линковки.

Программа должна содержать псевдоинструкции ассемблера и команды процессора; каждая из этих конструкций должна заканчиваться символом ";" точки с запятой.

Псевдоинструкции ассемблера используются программой - ассемблером на этапе трансляции для правильной компоновки модуля; они не создают исполняемых процессором команд и в большинстве случаев направлены на создание удобств при программировании. Псевдоинструкции ассемблера начинаются с символа "." точки.

Ввиду большого объема внутренней (резидентной) памяти и высокой производительности процессора, позволяющих создавать программы весьма

высокой сложности, не представляется реальной с психофизической точки зрения возможность успешного отслеживания размещения исходных данных и результатов промежуточных вычислений в памяти процессора. Для облегчения работы с константами и ячейками памяти (переменными) им могут быть приписаны символические имена подобно тому, как это делается в языках высокого уровня. Символическое имя представляет собой набор символов, содержащий заглавные и строчные буквы латинского алфавита, цифры и символы подчеркивания (Shift -), начинающийся не с цифры и имеющий длину до 32 символов. Набор символов, составляющий символическое имя, может быть произвольным, однако существует некоторый набор зарезервированных слов, не допускающих использования в качестве символических имен. К ним относятся имена регистров, мнемоника операций и псевдоинструкций ассемблера и ряд других слов, использование которых в программе в качестве символических имен будет вызывать сообщение об ошибках трансляции.

Константа и символическое имя задаются в программе псевдоинструкцией

#define имя_константы значение;

#define символическое_имя значение;

Значение константы может быть задано как число в десятичной, двоичной или шестнадцатеричной системах счисления или в виде арифметического или логического выражения. Запись числа в двоичной системе счисления должна дополняться префиксом B#, а в шестнадцатеричной - префиксом 0x. Числа, записанные без префикса, интерпретируются ассемблером как десятичные; они могут быть записаны без знака или со знаком минус.

По умолчанию заглавные и строчные буквы символических имен и мнемокодов команд воспринимаются транслятором одинаково, однако при желании можно заставить транслятор воспринимать их как разные символы. Обращение к константе в программе осуществляется непосредственно по имени. Например

```
#define koefficient_1 = - 36  +  1024;
#define vhod  =  0xA3C;
      .
      .
      ax0  =  koefficient_1;
      ar  =  dm(vhod);
```

Переменные могут размещаться как в памяти данных процессора, так и в памяти его программ. При разработке программы в среде VisualDsp++ целесообразно озаботиться проблемой распределения памяти, особенно памяти программ под размещение отдельных модулей. За распределение памяти отвечает файл с расширением .ldf, формируемый линковщиком в соответствие с инструкцией по программированию. При необходимости такой файл можно сформировать вручную объявлением соответствующих секций, следует только помнить, что программа начинает свою работу с нулевого адреса. Пример такого файла представлен на рисунке 5.7.

```
ARCHITECTURE(ADSP-2181)
SEARCH_DIR( $ADI_DSP\218x\lib )
$OBJECTS = $COMMAND_LINE_OBJECTS ;
MEMORY
{
seg_vars       { TYPE(DM RAM) START(0x000000) END(0x002fff) WIDTH(16)  }
seg_program          { TYPE(PM RAM) START(0x000000) END(0x000fff) WIDTH(24) }
seg_varsp     { TYPE(PM RAM) START(0x001000) END(0x001fff) WIDTH(24)}
}
PROCESSOR p0
{
   LINK_AGAINST( $COMMAND_LINE_LINK_AGAINST)
   OUTPUT( $COMMAND_LINE_OUTPUT_FILE )

   SECTIONS
   {
   sec_program
   {
    INPUT_SECTIONS( $OBJECTS(program) )
    } >seg_program
        sec_vars
      {
         INPUT_SECTIONS( $OBJECTS(vars) )
      } >seg_vars
        sec_varsp
      {
         INPUT_SECTIONS( $OBJECTS(varsp) )
      } >seg_varsp
   }
}
```

Рисунок 5.7 Пример файла линковки

Представленный файл организует три секции:

- program – в памяти программ TYPE (PM RAM) с нулевого адреса до 0x000fff – естественно, для размещения программы;

- varsp – также в памяти программ со следующего адреса 0x001000 до 0x001fff – для размещения данных (если до конца адресного пространства – то до 0x3fff);

- vars – в памяти данных TYPE(DM RAM) с нулевого адреса до 0x002fff.

Имя переменной (ячейки памяти) задается в соответствующей секции псевдоинструкцией .var имя_переменной, например:

- в ОЗУ данных:

```
.section/dm vars;
.var koef2;
.var delta;
…
```

- в ОЗУ программ:

```
.section/pm varsp;
.var sinx;
.var destb;
```

Каждой переменной при объявлении может быть присвоено начальное значение, которое она принимает при каждом рестарте процессора:

.var имя_переменной = значение;

В ряде случаев бывает удобным представление переменных в виде массивов из нескольких ячеек памяти; в этом случае при объявлении имени переменной должен быть в квадратных скобках указан размер массива, а начальные значения его ячеек задаются через запятые, например:

```
.var  massiv_1[5] = -3,  B#0000101001011110, 0x287C, 0, 0;
```

Кольцевые массивы (или кольцевые буферы) организуются в резидентной памяти процессора псевдоинструкцией

.var/circ имя_буфера [размер_буфера];

На работу с элементами кольцевых массивов распространяются те же правила, что и для обычных линейных массивов. Отличие заключается в правилах доступа по системе косвенной поставтомодифицирующей адресации, организованной через ГАД1 или ГАД2; правила использования такой адресации будут рассмотрены далее.

В ряде случаев представляется удобной инициализация массива

(занесение начальных значений в его ячейки) из файла данных, хранящегося на носителе компьютера; такая инициализация может быть осуществлена на этапе объявления переменных:

.var имя_массива = <имя_файла>;

при этом начальные значения, подлежащие загрузке в массив, должны быть записаны в файле "в столбик", после последнего значения должен быть помещен код конца строки (нажать клавишу Enter).

Из внешнего файла может быть также включен в программу любой фрагмент, например, список констант или переменных; такое включение реализуется псевдоинструкцией.

#include <имя_файла>;

Псевдоинструкция #include включает текст указанного в угловых скобках файла в текст основного файла при трансляции модуля.

В целях повышения наглядности и удобочитаемости программы в ее текст полезно включать комментарии. Комментарии не транслируются и не оказывают влияния на характеристики (объем и быстродействие) загружаемого модуля программы. Комментарии размещаются в тексте программы между командами или между псевдоинструкциями ассемблера и должны либо предваряться двумя знаками дроби - слэшами (//), либо в символы / и * в виде:

// комментарий

/* комментарий */

В тексте комментариев допускается использование символов кириллицы. Вложенность комментариев одного типа не допускается. Пример использования комментариев

```
/*
Эта программа реализует функцию....
*/
ar  =  dm(koef_beta);  /* чтение значения бета */

ar  =  ar  +  1;     // увеличение его значения
```

Одной из главных особенностей сигнальных процессоров, и, в частности, ADSP-2181, является ориентация их архитектуры и системы команд именно на разработку цифровых фильтров. В состав архитектуры процессоров семейства ADSP-21XX входит аппаратный умножитель - накопитель, обеспечивающий, в частности, возможность за один машинный цикл вычислить значение произведения двух 16-разрядных операндов и сложение полученного произведения с содержимым 40-разрядного поля. Произведение может быть

вычислено как в знаковом, так и в беззнаковом формате и выровнено либо в сторону старших разрядов (по правилу умножения чисел в дробном формате), либо в сторону младших (по правилам действия над целыми числами).

Основным форматом представления операндов при обработке сигналов является дробный знаковый формат 1.15, при котором старший разряд интерпретируется как знак (1 = минус!), а следующий за ним разряд имеет вес $2^{-1}$ и т.д. по нисходящей. В этом формате числа представлены в диапазоне от минус 1 до +1 - $2^{-15}$, т.е. удобном для представления виде $\approx \pm 1$. При этом вес знакового разряда равен минус $2^0$, т.е. минус 1.

Дробный беззнаковый формат 0.16 отличается тем, что вес старшего разряда интерпретируют как +1; таким образом, двоичные комбинации интерпретируются как числа в диапазоне от 0 до 2 - $2^{-15}$. Тип формата должен быть указан в команде – знаковый как `(s)`, беззнаковый – как `(u)`.Форматы представления целых чисел будут рассмотрены далее. При вычислении произведения один из сомножителей должен быть записан в регистр типа `xop`, например, в регистр `MX0`, а второй – в регистр типа `yop`, например, `MY0`. Приемником результата операции могут быть либо 16-разрядный регистр `MF`, в который записываются только старшие разряды результата, либо 40-разрядный регистр `MR`, составленный из 8-разрядного регистра `MR2` и двух 16-разрядных `MR1` и `MR0`; в этой связке регистр `MR2` содержит старшие разряды результата. Поскольку размерность одиночного произведения не превышает 32-х разрядов, для его записи достаточно регистров `MR1` и `MR0`. Регистр `MR2` служит для накопления суммы произведений, с его помощью можно без потери информации накопить сумму до 256 произведений.

Ранее было показано, что каждый отсчет сигнала на выходе фильтра формируется вычислением суммы восьмидесяти произведений отсчетов входного сигнала на отсчеты импульсной характеристики фильтра.

Формат выходного сигнала фильтра должен соответствовать формату входного, поэтому вычисление произведений осуществляется в дробном формате как `(ss)` - знаковый на знаковый, при этом как знаковый должен интерпретироваться старший разряд регистра `MR1`.

Для реализации цифровых фильтров в ADSP-2181 предназначена многофункциональная операция

```
mr = mr + xop * yop (ss), xop = dm(Ii, Mj), yop = pm(Ik, Mn);
```

которая позволяет за один машинный цикл не только накопить в регистре MR очередное произведение, но и обновить операнды к выполнению нового накопления в следующем цикле.

Алгоритм реализации процесса вычислений значения очередного отсчета на выходе фильтра иллюстрируется рис. 5.8. В памяти программ организуется кольцевой буфер impr на 80 ячеек, в который записываются 80 отсчетов импульсной характеристики фильтра. Обращение к отсчетам, начиная с **g0**, осуществляется по принципу кольцевой косвенной поставтомодифицирующей адресации через регистр I4 генератора адреса данных ГАД 2, в качестве модификатора может использоваться любой из ГАД 2, например, M4. Последовательное считывание отсчетов будет обеспечено установкой M4 = 1.

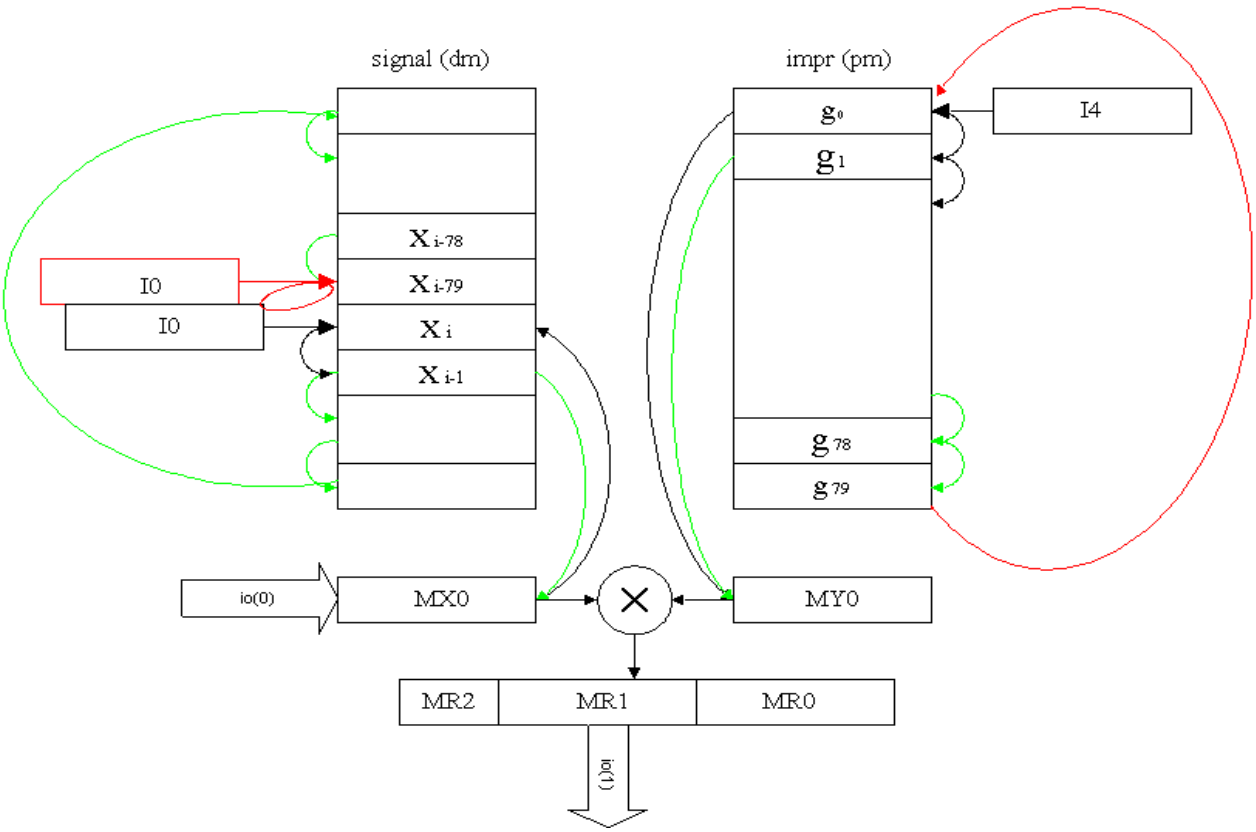



Рисунок 5.8. Алгоритм реализации КИХ-фильтра.

В памяти данных организуется кольцевой «оконный» буфер signal на 80 ячеек, содержащий 80 последних отсчетов сигнала со входа фильтра. Обращение к ячейкам буфера организуется также в режиме косвенной кольцевой поставтомодифицирующей адресации через указатель I0 из генератора адреса данных ГАД 1. Обращение к ячейкам потребует использования двух значений модификатора; для ускорения выполнения

программы целесообразно использовать два регистра M0 = 0 и M1 = 1. Начальная установка регистра I0 в пределах буфера некритична.

Исходное положение регистров на момент обработки отсчета выходного сигнала $y_i$ показано черными стрелками. Очередной отсчет входного сигнала $x_i$ принимается в регистр MX0, откуда он записывается в буфер signal в ячейку, на которую указывает регистр I0 и в который ранее был записан самый ранний отсчет (в рассматриваемом случае $x_{i-80}$), а указатель I0 переставляется на следующую ячейку, в которую ранее был записан отсчет входного сигнала $x_{i-1}$, с помощью модификатора M1. В регистр MY0 из буфера impr считывается отсчет импульсной характеристики **g0**, а указатель I4 с помощью модификатора M4 переставляется на следующую ячейку, в которой записан отсчет **g1**.

Первое выполнение команды (с предварительным обнулением регистра MR командой MR = 0)

```
mr = mr + mx0 * my0 (ss), mx0 = dm(i0, m1), my0 = pm(i4, m4);
```

вычисляет произведение $x_i * g_0$, в регистры MX0 и MY0 будут записаны, соответственно, $x_{i-1}$ и $g_1$, а регистры I0 и I4 будут указывать на ячейки, содержащие, соответственно, $x_{i-2}$ и $g_2$.

Таким образом, для вычисления значения отсчета $y_i$ эту операцию необходимо повторить еще 78 раз, что на рис. 5.8 показано зелеными стрелками. При последнем обращении должны быть подготовлены условия для приема отсчета $x_{i+1}$ (чтобы можно было далее вычислять значение отсчета $y_{i+1}$); для этого указатель I4 должен переместиться на начало буфера с помощью модификатора M4=1, а указатель I0 должен остановиться на самом старом отсчете $x_{i-79}$, чтобы по его месту записать будущий отсчет $x_{i+1}$. Это может быть выполнено с помощью модификатора M0=0; завершающий этап обращений показан красными стрелками.

Вычисленное значение $y_i$ в результате выполнения программы размещено в 40-разрядном поле регистра MR. Чтобы привести результат в рамки 16-разрядной сетки MR1 рекомендуется сначала округлить результат по младшим разрядам (содержимому регистра MR0) командой mr = mr(rnd), и затем, во избежание появления шумов (искажений) перегрузки, ограничить содержимое регистра MR по переполнению командой if mv sat mr.

Текст отладочной программы на языке ассемблера процессора ADSP-2181 с подробными комментариями, реализующей рассмотренные алгоритмы, приведен на рисунке 5.9.

```
//*************************************************************
//
// область объявления переменных в памяти данных
//
.SECTION/DM vars;
// объявление переменных
// кольцевой буфер хранения 80 отсчетов сигнала в памяти данных
// Содержимое ячеек буфера должно быть инициализировано,
// например, обнулено, поскольку симулятор будет обращаться
// к ним для чтения раньше, чем для записи.
// По правилам написания программ на языке Ассемблера
// записываемые в буфер значения отсчетов должны быть
// разделены запятыми, а после последнего значения должен
// быть записан символ точки с запятой.
.var/circ signal[80] =   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
//*************************************************************
//
//=============================================================
//
// область объявления переменных в памяти программ
//
.SECTION/PM varsp;
// кольцевой буфер хранения отсчетов импульсной характеристики
// в памяти программ
// Заполнение буфера impr значениями отсчетов импульсной
// характеристики из файла g_re.dat, который должен быть
// размещен в рабочей директории.
// Размещение отсчетов в буфере должно соответствовать
// порядку их следования, т.е. в ячейке impr(0) должен быть
// записан отсчет g(0), а в ячейке impr(79) - отсчет g(79).
// Правила записи значений отсчетов в файле аналогичны правилам
// записи в тексте программы.
// Адрес ячейки g(0) в памяти программ может быть определен как
// impr, а длина буфера - как length(impr).
.var/circ impr[80]="g_re.dat";

//=============================================================
//
//#############################################################

                                        //
```

Рисунок 5.9 начало. Отладочная программа КИХ-фильтра.

```
// область размещения программы
//
.SECTION/PM program;

//   Собственно программа фильтра с конечной импульсной
//   характеристикой:
//   вычисление значения отсчета сигнала на выходе фильтра как
//   80-ти произведений для j от 0 до 79 (i-j)-того отсчета сигнала
//   на j-тый отсчет импульсной характеристики фильтра.
//   До начала циклической части программы указатель I4
//   устанавливается
//   на начало буфера impr,   а I0 - на начало буфера signal
//   (это необязательно, можно I0 установить на любую ячейку
//    буфера).
     i4=impr;
     l4=length(impr);
     i0=signal;
     l0=length(signal);
//
//   модификатор M4 указателя I4 предполагает поочередное чтение
//   всех отсчетов импульсной характеристики фильтра из буфера impr
//   и возвращение в начало буфера в конце цикла
     m4=1;
//
//   для доступа к буферу отсчетов сигнала выделены 2 модификатора
//   M1=1 - для последовательного чтения предшествующих
//   по времени отсчетов
//   M0=0 - для того, чтобы в конце циклической части
//   обработки указатель
//  I0 остановить на отсчете i-79, чтобы отсчет
//   i+1 записать по его месту
     m0=0;
     m1=1;
//
//   эта команда для экономии одной команды внутри цикла
//   в дальнейшем она выполняется в конце цикла одновременно с
//   вычислением последнего произведения
     my0=pm(i4,m4); //чтение g0 из памяти программ
//
//   бесконечный цикл непрерывной обработки;
//   он допустИм только для отладочной программы.
     do vivod until forever;
//
//        ввод очередного отсчета входного сигнала
//        из порта IO(0)
          mx0=io(0);

//
```

Рисунок 5.9 продолжение

```
//        его запись в буфер signal по месту самого старого
//        и обнуление умножителя - регистров MR2, MR1 и MR0
                                    .
          dm(i0,m1)=mx0, mr=0;
//
//        внутренний цикл: вычисление суммы 78 произведений
//        с последовательным чтением отсчетов сигнала
//        из буфера signal
//        и отсчетов импульсной характеристики из буфера impr


               cntr=78;
               do filt until ce;
filt:          mr=mr+mx0*my0(ss), mx0=dm(i0,m1), my0=pm(i4,m4);
//
//        вычисление и суммирование 79-го произведения,
//        чтение (i-80) -го - самого "старого" - отсчета
//        сигнала из буфера signal,
//        остановка движения указателя I0 на ячейке
//        самого "старого" отсчета,
//        чтение 80-го отсчета импульсной характеристики
//        из буфера impr,
//        перевод указателя I4 на начало буфера impr
          mr=mr+mx0*my0(ss), mx0=dm(i0,m0), my0=pm(i4,m4);
//
//        вычисление и суммирование 80-го произведения
//        и чтение отсчета g0 импульсной характеристики
//        из буфера impr
//        для следующего цикла (для экономии времени выполнения)
          mr=mr+mx0*my0(ss), my0=pm(i4,m4);
//
//        формирование выходного отсчета в 16-разрядной сетке
//        из 40-разрядного поля регистра MR
//        округлением по младшим разрядам регистра MR0
//        и ограничением по переполнению в регистр MR2
          mr=mr(rnd);
          if mv sat mr;
//
//   вывод результата в порт IO(1)
vivod: io(1)=mr1;
//*************************************************************
```

Рисунок 5.9 окончание.

Составленная таким образом программа должна быть введена в компьютер, например, в редакторе UltraEdit или встроенном редакторе симулятора, сохранена в виде файла с расширением .asm, подвергнута сначала трансляции, а затем редакции связей (линковке) в соответствии с рисунке 5.10. В результате трансляции будут созданы листинговый и объектный файлы, а в

результате линковки – карта распределения памяти и файл, предназначенный для загрузки в симулятор.

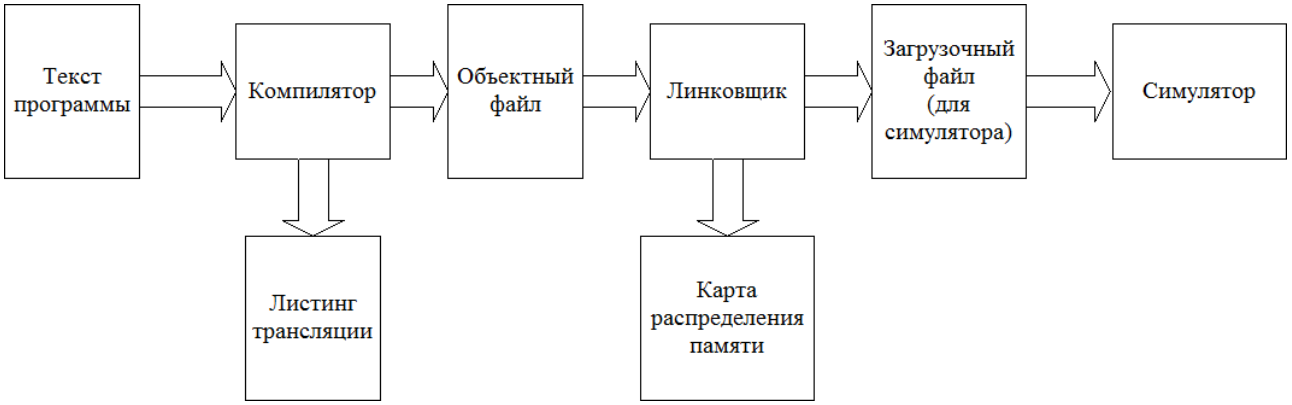



Рисунок 5.10. Подготовка файла для отладки программы.

Отладка разработанной программы может быть выполнена с использованием симулятора.

5.2.3. Отладка программы.

Следующим этапом разработки КИХ-фильтра является оценка соответствующего вида амплитудно-частотной характеристики требованиям технического задания. Эта оценка должна выполняться на синусоидальных сигналах соответствующих частот, предварительно подготовленных в любой языковой системе или с использованием звукового редактора, например, Cool Edit.

Однако предварительно целесообразно измерить импульсную характеристику разработанного фильтра и оценить ее соответствие расчетной (рис. 6). Для этого следует создать файл с записью импульса дельта-функции и подключить его в режиме симуляции к разработанной программе в качестве входного сигнала. Выходной сигнал в процессе симуляции следует записать в отдельный файл; он будет отражать вид импульсной характеристики фильтра.

Входной файл с импульсом в виде дельта-функции может быть создан вручную, например, в том же редакторе UltraEdit. Он должен содержать не менее 80 строк. В первой строке должно быть записано число 32767 (самое большое положительное, но на 1 меньше эквивалента дельта-импульса), либо число минус 32768 (отрицательный эквивалент дельта-импульса). Остальные строки должны содержать число 0.

Подключение входного и выходного файлов к симулятору иллюстрируется, соответственно, рис. 5.11 и рис. 5.12; вызов поля Streams – движением по меню Settings/ Streams.

Результаты работы программы на дельта-импульс амплитудой 32767 шагов квантования представлен на рис. 5.13; он хорошо согласуется с расчетными данными с рис. 5.6, масштаб составит 32767/32768.

Соответствие расчетных данных и экспериментальных результатов свидетельствует только о правильности работы программы фильтра. Правильность же расчетных данных необходимо проверить на синусоидальных сигналах на критических частотах. Процедуры формирования и анализа синусоидальных сигналов с помощью звукового редактора Cool Edit подробно рассмотрены в [3, 5], там же можно более подробно ознакомиться с правилами записи программ на языке ассемблера процессора ADSP-2181.

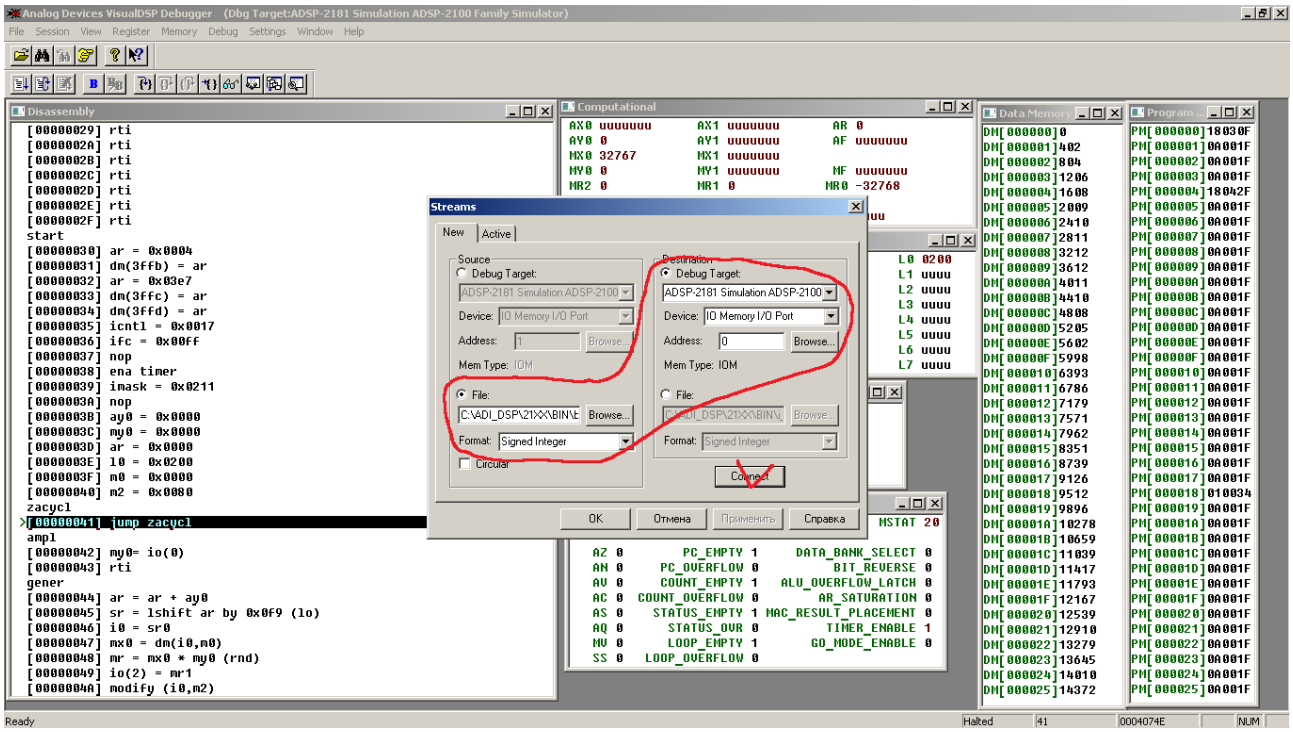



Рисунок 5.11. Подключение файла к порту ввода.

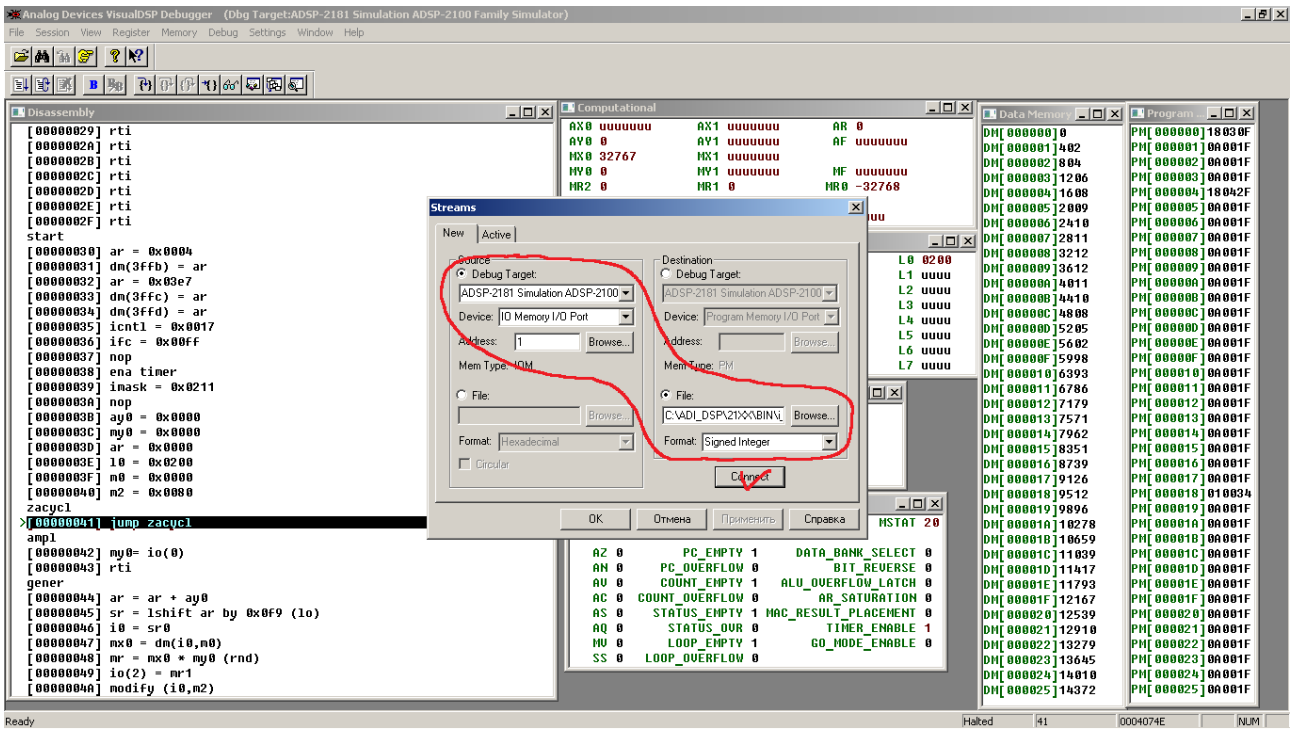



Рисунок 5.12. Подключение файла к порту вывода.

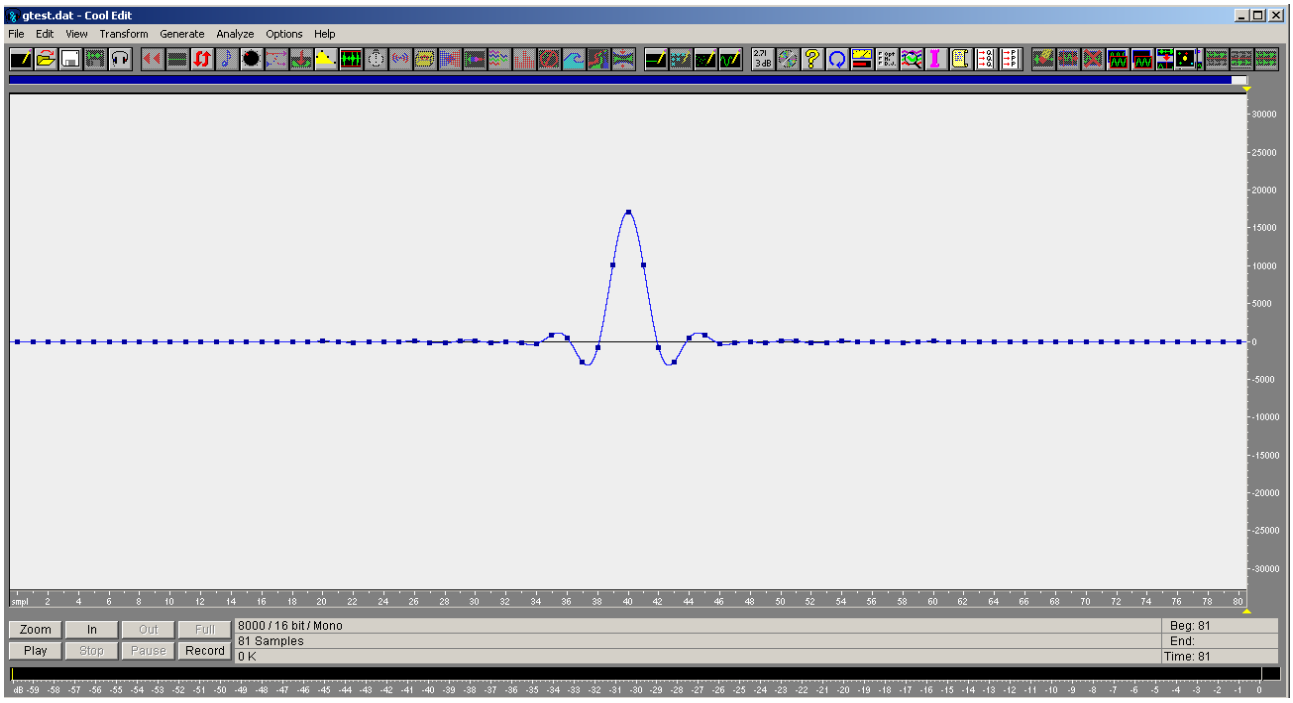



Рисунок 5.13. Импульсная характеристика фильтра (результат отладки программы).

5.2.4. Итоговая программа фильтра.

Рассмотренная ранее отладочная программа не рассчитана на обработку сигналов в реальном времени; ее основное назначение, как было указано ранее, заключается в оценке правильности расчета значений отсчетов импульсной характеристики фильтра и правильности реализации алгоритма обработки сигнала сигнальным процессором. Несоответствие разработанной программы условиям работы в реальном времени обусловлено несоблюдением значения частоты дискретизации сигнала.

В соответствии с требованиями технического задания на разработку, значение частоты дискретизации должно быть равно 8 кГц; это означает, что период циклического обращения программы к портам ввода/вывода IO(0) и IO(1) должен соответствовать 125 микросекундам.

В отладочной программе выполнение циклической части включает 87 машинных циклов, что при типовом для процессора ADSP-2181 значении длительности машинного цикла 25 наносекунд составит 2,175 микросекунды. Таким образом, частота дискретизации сигнала, заданная программой фильтра, будет соответствовать 495,77 кГц, что приведет к растяжению АЧХ и ФЧХ по шкале частот в 57,471 раз.

При проектировании реальных устройств обработки сигналов, в том числе фильтров, частоту дискретизации (частоту ввода/вывода сигнала) стремятся обычно привязать к какому-то опорному источнику. В цифровых системах передачи таким источником обычно является цикловой синхросигнал. Привязка частоты обращения к портам ввода/вывода к цикловому синхросигналу цифровой системы передачи на уровне первичного цифрового потока подробно рассмотрена в [1].

При автономной работе фильтра процессор сам может выполнять функции генерации частоты дискретизации с заданным значением; для этой цели в составе архитектуры ADSP-2181 имеется таймер. Устройство таймера и принципы организации таймерных программ подробно рассмотрены в [1].

Реализация программы фильтра, работающего в реальном времени, потребует ликвидации процедуры зацикливания программы и разбиения программы на 3 части. Одна ее часть, представленная на рис. 5.14, должна быть включена в общую программу в раздел объявления и инициализации переменных. Вторая часть, приведенная на рис. 5.15 – в основную программу, а

третья часть, представленная на рис. 5.16, должна быть оформлена как подпрограмма, ассоциированная с таймерным прерыванием.

```
//*****************************************************
//        Программа fir_init.asm
//   объявление и инициализация переменных КИХ-фильтра
//
//=====================================================
// область объявления переменных в памяти данных
//
.global signal;
.global impr;
.SECTION/DM vars;
// сигнал
.var/circ signal[80] = 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
          0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
          0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
          0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
//=====================================================
//
// область объявления переменных в памяти программ
//
.SECTION/PM varsp;
// импульсная характеристика
.var/circ impr[80]="g_re.dat";
//*****************************************************
```

Рисунок 5.14. Часть для включения в инициализацию `fir_init.asm`.

```
//*************************************************************
//        Программа fir_main.asm
//   Установка регистров указателей и модификаторов
//
.SECTION/PM program;
.extern impr;
.extern signal;
.global fir_ini;
//   импульсная характеристика
fir_ini:
     i4=impr;
     l4=length(impr);
     m4=1;
//   сигнал
     i0=signal;
     l0=length(signal);
     m0=0;
```

Рисунок 5.15 начало. Часть для включения в основную программу `fir_main.asm`.

```
m1=1;

//    Фильтр,
      my0=pm(i4,m4);
      rts;
fir_ini.end:
//*************************************************************
```

Рисунок 5.15 окончание

Таймер должен быть запрограммирован на коэффициент деления, равный отношению периода дискретизации сигнала к длительности машинного цикла процессора. Если длительность машинного цикла равна 25 нс, то значение частоты дискретизации, равное 8 кГц, будет обеспечено при коэффициенте деления, равном 5000.

Коэффициент деления таймера устанавливается записью в регистры TSCALE (8 разрядов) и TPERIOD (16 разрядов) значения коэффициента деления таким образом, что

Кт = (TPERIOD + 1) * (TSCALE + 1)

Кроме этих регистров в таймер входит счетчик TCOUNT, перезагружаемый по обратному переполнению из регистра TPERIOD; запрос на первое прерывание сформируется спустя (TCOUNT + 1) * (TSCALE + 1) машинных циклов после запуска таймера.

Регистры TCOUNT, TPERIOD и TSCALE адресуются как ячейки памяти данных, соответственно, `dm(0x3FFC)`, `dm(0x3FFd)` и `dm(0x3FFB)`. Естественно, в регистр TSCALE можно записывать числа не более 255, а в остальные – не более 65535.

```
//*************************************************************
//         Программа fir_int.dsp
//         циклическая часть программы фильтра,
//         выполняемая по таймерному прерыванию
//
//
// область размещения программы
//
.SECTION/PM program;
.global filtr;
//
//
```

Рисунок 5.16 начало. Часть для включения в прерывающую подпрограмму `fir_int.asm`.

```
filtr:     mx0=io(0);
      dm(i0,m1)=mx0, mr=0;
//
      cntr=78;
      do filt until ce;
filt:      mr=mr+mx0*my0(ss), mx0=dm(i0,m1), my0=pm(i4,m4);
      mr=mr+mx0*my0(ss), mx0=dm(i0,m0), my0=pm(i4,m4);
      mr=mr+mx0*my0(ss), my0=pm(i4,m4);

      mr=mr(rnd);
      if mv sat mr;
      io(1)=mr1;
      rts;
filtr.end:
//*************************************************************
```

Рисунок 5.16 окончание. Часть для включения в прерывающую подпрограмму fir_int.asm.

Помимо установки значения коэффициента деления, необходимо открыть таймерное прерывание и стартовать работу таймера.

Итоговая программа КИХ-фильтра представлена на рис. 5.17. Коэффициент деления таймера, равный 5000, устанавливается записью значения 4 в регистр TSCALE и 999 в регистры TCOUNT и TPERIOD.

Рассмотренный пример иллюстрирует два важных для техники инфокоммуникаций свойства КИХ-фильтров:

- возможность реализации произвольной ФЧХ при произвольной АЧХ;

- возможность обеспечения нулевого коэффициента передачи на частотах, отличных от нуля.

```
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
//
//   Итоговая программа КИХ-фильтра
//
//   начало программы с нулевого адреса
//
//   включение в программу части объявления и инициализации
//   переменных фильтра (рис. 14)
//
//#############################################################
```

Рисунок 5.17 начало. Итоговая программа КИХ-фильтра.

```
//
// область размещения программы
//
.SECTION/PM program;
.extern filtr;
//
//   таблица векторов прерываний
//   с нулевого адреса начинается
//   заполнение таблицы векторов прерываний
//
interr:
     jump start; rti; rti; rti; // RESET
     rti; rti; rti; rti; // IRQ2
     rti; rti; rti; rti; // IRQL1
     rti; rti; rti; rti; // IRQL0
     rti; rti; rti; rti; // SPORT0 Tx
     rti; rti; rti; rti; // SPORT0 Rx
     rti; rti; rti; rti; // IRQE
     rti; rti; rti; rti; // BDMA
     rti; rti; rti; rti; // SPORT1 Tx
     rti; rti; rti; rti; // SPORT1 Rx
     jump timer_proc; rti; rti; rti; // Timer
     rti; rti; rti; rti; // Power Down
//   ------------------------------------------------
//   конец таблицы векторов прерываний
//   ------------------------------------------------}
//
//        Основная программа

//        $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
//        начало основной программы
//        $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

//
//        программирование таймера и прерываний
//
//        предделитель TSCALE
start:    ar = 4;
          dm(0x3ffb) = ar;
//
//        счетчик периода TPERIOD и первого прерывания TCOUNT
          ar = 999;
          dm(0x3ffc) = ar;
          dm(0x3ffd) = ar;
//
//        включение в основную программу части программы обработки,
//        относящейся к фильтру (рис. 15)
//-------------------------------------------------------------
//
//        сброс прерываний, стоящих в очереди
```

Рисунок 5.17 продолжение.

```
          ifc = 0x00ff;
//
//        холостая операция
          nop;
//
//        запуск таймера
          ena timer;
//
//        маскирование прерываний
          imask = B#00000000001;
//                              +------ таймерное прерывание
//                                      разрешено
//        холостая операция
          nop;
//
//        зацикливание основной программы
//
zacycl:       jump zacycl;
//
//        $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
//        конец основной программы
//        $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
//-------------------------------------------------------
//              >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
//              таймерная подпрограмма
//              >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
timer_proc:
//              сохранение значений используемых регистров
//              (при необходимости)
//
//              ****************************************
//              здесь должна быть размещена
//              программа, исполняемая периодически
//              1 раз в 125 мкс
//              +++++++++++++++
//              включение в прерывающую подпрограмму части программы
//              обработки, относящейся к фильтру (рис. 16)
              call filtr;
//              ****************************************
//
//              восстановление значений используемых регистров
//              (если ранее были сохранены)
//
//              возврат из подпрограммы
              rti;
interr.end:
//
//              >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
//              конец таймерной подпрограммы
//              >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

Рисунок 5.17. окончание.

Эти свойства не могут быть обеспечены аналоговыми фильтрами, построенными на реактивных элементах.

В заключение следует отметить одну важную особенность архитектуры КИХ, связанную с представлением импульсной характеристики фильтра на ограниченном количестве отсчетов. Так, в рассмотренном примере длительность импульсной характеристики фильтра ограничена количеством отсчетов $N = 80$, что приводит к линейчатому представлению АЧХ и ФЧХ на сетке частот с шагом $\Delta f = F_д/N = 100$Гц. Характерно, что в буфер `signal` при этом может быть записан ровно один период синусоидального сигнала частотой 100 Гц.

Анализируя рассмотренную в примере процедуру вычисления импульсной характеристики фильтра, можно с известной степенью абстракции утверждать, что вычисление свертки входного сигнала с импульсной характеристикой фильтра сородственно преобразованию Фурье исходного сигнала для наложения на его спектр характеристики комплексного коэффициента передачи. При этом буфером `signal` из входного сигнала «выгрызается» используемый для формирования отсчета выходного сигнала отрезок.

Проблема заключается в том, что преобразование Фурье интерпретирует этот отрезок как периодический сигнал, поскольку в силу физических принципов линейчатый (дискретный) спектр соответствует периодическому процессу.

Между тем, как было показано, строго периодическим для рассмотренного фильтра может быть только процесс, частота которого кратна 100 Гц. Таким образом, на промежуточных частотах будет возникать погрешность, которая характеризуется как пульсации Гиббса или эффект Гиббса.

Для снижения погрешностей, связанных с эффектом Гиббса, используют умножение импульсной характеристики фильтра на некоторые сглаживающие кривые, которые получили название «окна». Широко используется некоторый набор окон, де-факто получивших статус стандарта, обладающих разными свойствами и используемых в разных приложениях. Вопрос о свойствах окон и рекомендации по их использованию выходят за рамки настоящего учебного пособия, подробное их описание можно найти в [2]. Тем не менее, реализация КИХ-фильтров в конкретных инфокоммуникационных приложениях обычно требует использования высоких порядков, что, в свою очередь, требует значительных вычислительных ресурсов.

В тех приложениях, в которых не требуется использования специальных свойств КИХ-фильтров, используют структуры с бесконечной импульсной характеристикой; этим структурам посвящена следующая глава.

### 5.3. Фильтры с бесконечной импульсной характеристикой.

Фильтры с бесконечной импульсной характеристикой (БИХ-фильтры) обеспечивают при равных порядках более высокую по сравнению с КИХ-фильтрами добротность в традиционном для аналоговых технологий понимании, т.е. более высокую крутизну нарастания АЧХ.

Для обеспечения более высокой добротности в БИХ-фильтрах используется механизм положительной обратной связи. Как известно, коэффициент передачи системы с обратной связью $K_{ОС}$ зависит от глубины обратной связи и может быть описан выражением $K_{ОС} = K_{безОС}/F$, где $K_{безОС}$ - коэффициент передачи системы при разомкнутой петле обратной связи, $F = 1 - T$ - возвратная разность, характеризующая глубину обратной связи, Т – возвратное отношение, характеризующее коэффициент передачи по петле обратной связи. С учетом инерционности цепей распространения сигнала и возможной его инверсии знак возвратного отношения на заданной частоте определяет знак обратной связи.

Как и любое другое явление, обратная связь вносит свои коррективы в работу фильтров. Главными моментами, которые необходимо учитывать при использовании обратной связи, в том числе при проектировании цифровых фильтров, как известно, являются:

- система, охваченная обратной связью, может вносить усиление в обрабатываемый сигнал, выводя значение вычисляемых в процессе обработки отсчетов за пределы имеющейся разрядной сетки;

- при известных условиях цепь, охваченная обратной связью, может терять устойчивость (в этом случае в цепи возникают автоколебания); поэтому коэффициенты передачи в цепи обратной связи следует устанавливать с максимально возможной точностью и по возможности избегать округления при вычислениях, связанных с обработкой;

- процессы в системе с обратной связью будут обладать некоторой предсказуемой, но неуправляемой инерционностью, формирующей переходные процессы значительной длительности.

Механизм обратной связи в БИХ – фильтрах реализуется в форме так называемого полюса, представленного на рис. 5.18.

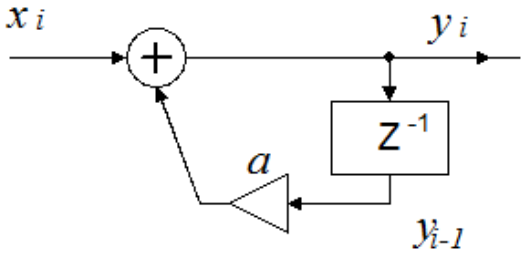



Рисунок 5.18. Одиночный полюс.

Сигнал $y_i$ на выходе одиночного полюса зависит не только от входного сигнала $x_i$, но и от предшествующих значений выходного. Значение отсчета выходного сигнала $y_i$ может быть определено как:

$$y_i = x_i + a * y_{i-1}.$$

Из этого выражения следует, что форма импульсной характеристики одиночного полюса имеет асимптотически (экспоненциально) убывающий характер – равномерно убывающий $y_n = a^n$ при $0 < a < 1$ и знакопеременный характер с экспоненциальным снижением амплитуды $y_n = (-|a|)^n$ при $-1 < a < 0$. Такое поведение системы, соответствуя изменению энергии в реактивных элементах аналоговых фильтров, связывает амплитудно-частотную характеристику БИХ-фильтров с фазо-частотной.

Передаточная характеристика одиночного полюса описывается выражением:

$$H(z) = \frac{1}{1 - a * z^{-1}}$$

Значение $|a| = 1$ обращает в 0 знаменатель $H(z)$. При $a = 1$ одиночный полюс превращается в интегратор, а при $a = -1$ - в генератор, поэтому при проектировании БИХ-фильтров всегда следует помнить о возможной потере системой устойчивости. Генератор можно рассматривать как фильтр с бесконечной добротностью, а при $|a| \to 1$ поведение одиночного полюса будет соответствовать поведению фильтра тем более высокого порядка, чем ближе $a$

к 1. Причем положительное значение $a$ обеспечивает функции фильтра нижних частот (интегратора) а отрицательное – полосового фильтра (резонатора).

5.3.1. Биквадратное звено.

Помимо понятия полюса, в технике разработки как КИХ-, так и БИХ-фильтров используется понятие нуля, являющегося элементом ранее рассмотренной архитектуры КИХ. Нулями называют корни числителя передаточной характеристики фильтра, значения которых могут на каких-то частотах обращать в ноль значения коэффициента передачи фильтра.

В технике инфокоммуникаций чаще других БИХ-структур для реализации фильтров используют так называемое биквадратное звено, схема которого представлена на рис. 5.19.

Учитывая необходимость оценки влияния времени задержки сигнала в структуре фильтра на знак и глубину обратной связи, для описания свойств фильтра обычно используется передаточная характеристика, которая для биквадратного звена, показанного на рис. 5.19, может быть представлена в виде:

$$H(z) = \frac{b_0 + b_1 * z^{-1} + b_2 * z^{-2}}{1 - a_1 * z^{-1} - a_2 * z^{-2}}$$

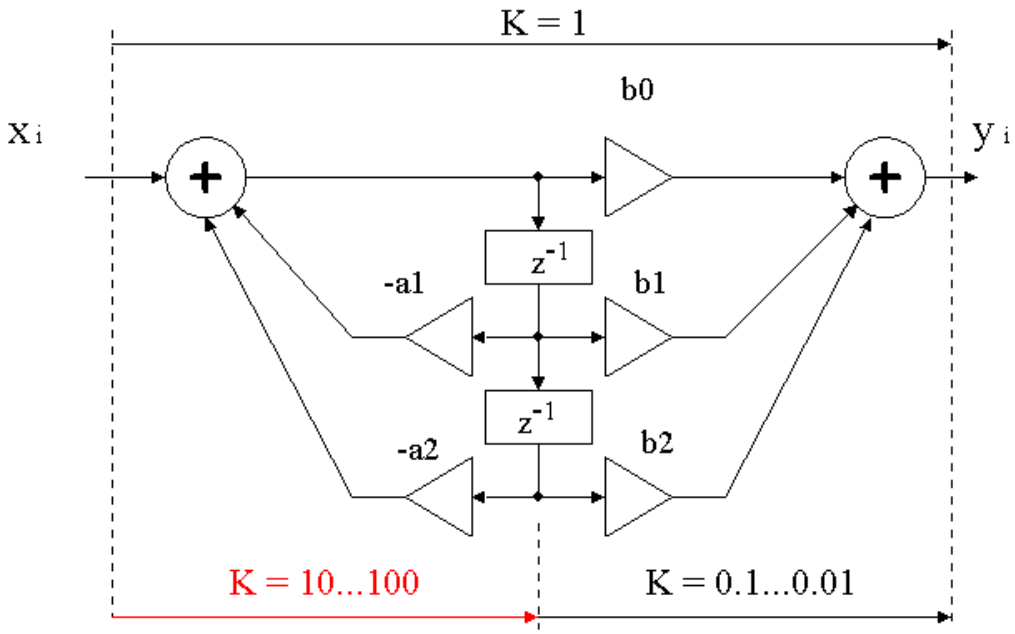



Рисунок 5.19. Биквадратное звено.

Собственно полюсами называются корни знаменателя, обращающие его значение в 0. Комплексно сопряженный характер полюсов обеспечивает идентичность характеристик АЧХ (а также и ФЧХ) в области положительных и отрицательных частот.

5.3.2. Особенности реализации БИХ-фильтров на сигнальных процессорах.

Для облегчения расчетов БИХ-фильтров используется множество программных продуктов. Эти продукты ориентированы на расчет по заданным типу и виду АЧХ значений коэффициентов $a1, a2, b0, b1 и b2$ биквадратного звена рис. 19 на комплексно сопряженных полюсах. Результаты типового расчета БИХ-фильтра шестого порядка представлены на рис. 5.20 – рис. 5.22. Рис. 5.20 отражает результаты расчета коэффициентов $a1, a2, b0, b1 и b2$ для трех каскадов фильтра, а рис. 5.21 и рис. 5.22 иллюстрируют обеспечиваемые для расчетных коэффициентов амплитудно-частотную и импульсную характеристики.

```
                INFINITE IMPULSE RESPONSE (IIR)
                  ELLIPTIC    BANDPASS FILTER
                16-BIT QUANTIZED COEFFICIENTS

                FILTER ORDER =     6
                SAMPLING FREQUENCY =    8.000  KILOHERTZ

I         A(I,1)         A(I,2)         B(I,0)         B(I,1)         B(I,2)

1      -1.676453       .964752        .019773        .000000       -.019773
2      -1.650818       .982910        .205811       -.318451        .205811
3      -1.730042       .984863        .433044       -.785065        .433044
```

Рисунок 5.20. Результаты расчета БИХ-фильтра.

Сопоставляя приведенные на рис. 5.20 результаты со схемой рис. 5.19, нетрудно видеть, что цепь КИХ-звена, сформированная коэффициентами $b0, b1 и b2$ для первого каскада фильтра, обеспечивает коэффициент передачи в полосе пропускания порядка 0,02. Поскольку общий коэффициент передачи каскада должен быть близок к единице (в полосе прозрачности), следует ожидать, что цепь БИХ-звена, сформированная коэффициентами $a1 и a2$, будет вносить в обрабатываемый сигнал усиление, примерно в 50 раз. К сожалению, такая картина при разработке фильтров является типовой.

Во избежание возникновения искажений перегрузки (иногда ошибочно называемых шумами перегрузки) на вход фильтра не следует подавать сигнал, амплитуда которого превышает 0,02. Результатом такого ограничения является снижение (в 50 раз в рассматриваемом случае) отношения сигнал/шум

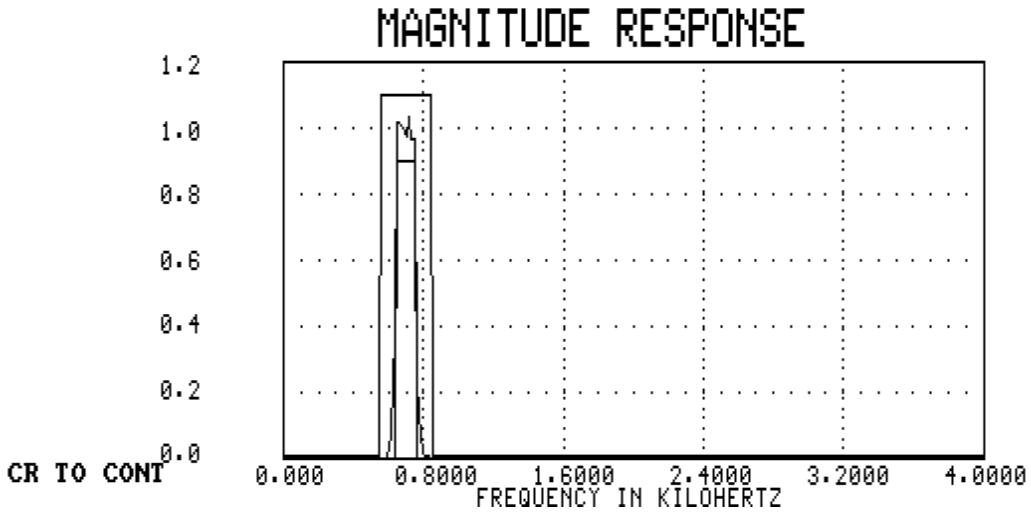



Рисунок 5.21. АЧХ фильтра (результат вычисления).

квантования, что эквивалентно снижению разрядности аналого-цифрового преобразования (на 6 бит). Таким образом, ввиду возможной перегрузки разрядной сетки БИХ-звена при построении алгоритмов цифровой обработки сигналов, включающей частотную фильтрацию БИХ-структурами, следует использовать масштабирование сигнала на входах БИХ-структур.

Еще одной особенностью реализации БИХ-фильтров в условиях ограниченной разрядной сетки является диапазон представления коэффициентов $a1 и a2$.

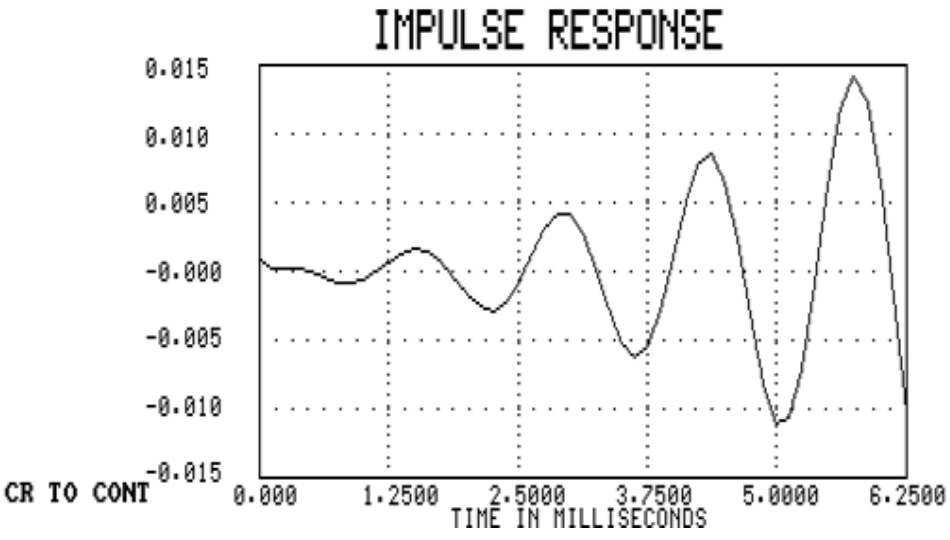



Рисунок 5.22. Импульсная характеристика фильтра (результат вычисления).

В рассматриваемом примере рис. 5.19 коэффициенты передачи $a1$ не могут быть представлены в дробном знаковом (s) формате, в рамках которого числа могут быть представлены в диапазоне $-1\ldots+1-2^{-15}$. В таких случаях модуль коэффициента $a1$ может быть представлен в дробном беззнаковом (u) формате в диапазоне $0\ldots2-2^{-15}$, а знак перенесен в команду суммирования произведений как

```
MR = MR – xop * yop (SU);,
```

где `xop` – отсчет сигнала, а `yop` – коэффициент $a1$.

Текст отладочной программы, реализующей первый каскад БИХ-фильтра, с необходимыми комментариями представлен на рис. 5.23.

```
//*********************************************************
//
// область объявления переменных в памяти данных
//
.SECTION/DM vars;
// объявление переменных
// сигнал
.var/circ z[2] = 0,0;
//*********************************************************
//
//=========================================================
//
// область объявления переменных в памяти программ
//
.SECTION/PM varsp;
//
//   значения коэффициентов передачи a1, a2, b0, b1 b2
//   вводятся в ячейки памяти программ из файла ab.dat
//   коэффициенты a1, a2, b0, b1, b2
.var/circ ab[5] = "ab.dat" ;
//=========================================================
//
//#########################################################
//
// область размещения программы
//
.SECTION/PM program;
//---------------------------------------------------------
//        Основная программа
filter:
//
```

Рисунок 5.23 начало. Отладочная программа БИХ-фильтра.

```
//   фильтр, установка регистров указателей и модификаторов
//
//
//   коэффициенты a1, a2, b0, b1, b2
     i4=ab;
     l4=length(ab);
     m4=1;
//   сигнал
     i0=z;
     l0=length(z);
     m0=0;
     m1=1;
//
//   внешний бесконечный цикл, допустимый только в
//   отладочной программе
//
     do vivod until forever;
//
//   обнуление регистра MR и подготовка к вычислению
//   первого произведения
//   (обновление z1,a1)
     mr=0, mx0=dm(i0,m1), my0=pm(i4,m4);
//
//   ввод очередного отсчета входного сигнала из порта IO(0)
     mr1=io(0);
//
//   вычисление отсчета сигнала на выходе БИХ-ветви
//   ВНИМАНИЕ
//   модуль значения коэффициента a1 в регистре MY0 превышает 1,
//   поэтому при вычислении произведения формат представления
//   информации в этом регистре указан как БЕЗЗНАКОВЫЙ (U),
//   а знак следует учитывать в действии (сложение\вычитание)
//   с произведением
//

// {вычисление xi+z1a1, обновление z2,a2}
     mr=mr+mx0*my0(su),mx1=dm(i0,m0),my0=pm(i4,m4);
//
// {вычисление +z2a2,обновление b0}
     mr=mr-mx1*my0(su), my0=pm(i4,m4);
//
//   БИХ-структура может вносить в обрабатываемый сигнал усиление,
//   поэтому во избежание искажений перегрузки разрядной сетки
//   следует использовать операцию ограничения по переполнению
//
     if mv sat mr;
     mr=mr(rnd), my1=pm(i4,m4); // {вычисление z0, обновление b1}
//
//   вычисление значения выходного отсчета
```

Рисунок 5.23 продолжение.

```
//
// {вычисление z0*b0}
     dm(i0,m0)=mr1, mr=mr1*my0(ss);
// {вычисление +z1*b1,обновление b2}
     mr=mr+mx0*my1(ss), my0=pm(i4,m4);
// {вычисление +z2*b2}
     mr=mr+mx1*my0(rnd);
//
//   и его вывод в порт IO(1)
//
vivod:    io(1)= mr1;
//
filter.end:    ;
//*************************************************************
```

Рисунок 5.23 окончание.

Поскольку точное значение усиления, вносимого в сигнал за счет эффекта обратной связи, заранее предсказать трудно, часто используют альтернативное представление биквадратного звена в соответствии с рис. 5.24, в котором переставлен порядок обработки сигнала – сначала КИХ-звеном, вносящим затухание, а потом БИХ-звеном, вносящим требуемое усиление.

Такая схема не требует предварительного масштабирования сигнала для предотвращения перегрузки разрядной сетки, однако повышение уровня шумов квантования, связанного с эффектом усиления БИХ-структурой, в этой схеме сохраняется на уровне, аналогичном схеме рис. 5.19. Текст программы на языке ассемблера процессора ADSP-2181, реализующей первый каскад БИХ-фильтра по схеме рис. 5.24, представлен с необходимыми комментариями на рис. 5.25.

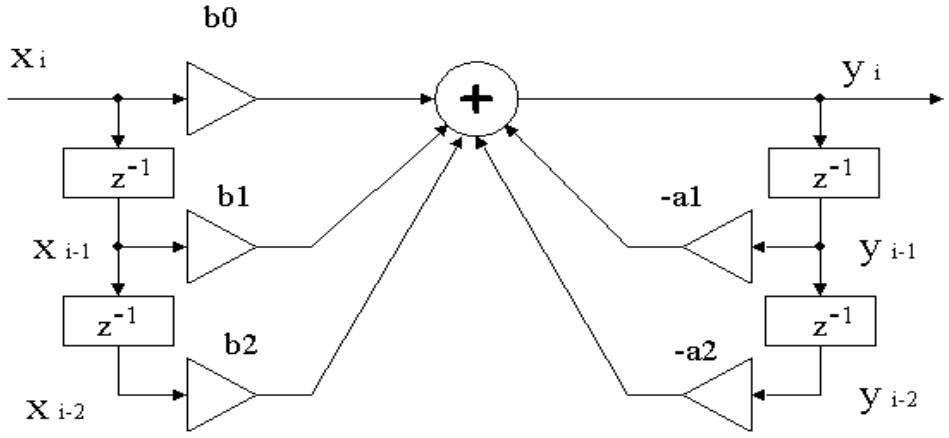



Рисунок 5.24. Вариант схемы биквадратного звена.

```
//*********************************************************
//
// область объявления переменных в памяти данных
//
.SECTION/DM vars;
// объявление переменных
// сигнал
// эта архитектура требует использования двух линий задержки
.var/circ z[2] = 0,0;
.var/circ zz[2]= 0,0;
//*********************************************************
//
//=========================================================
//
// область объявления переменных в памяти программ
//
.SECTION/PM varsp;
//
//   значения коэффициентов передачи a1, a2, b0, b1 b2
//   вводятся в ячейки памяти программ из файла ab.dat
//   коэффициенты a1, a2, b0, b1, b2
.var/circ ab[5] = "ab.dat" ;
//=========================================================
//
//#########################################################
//
// область размещения программы
//
.SECTION/PM program;
//---------------------------------------------------------
//        Основная программа
filter:
//
//   фильтр, установка регистров указателей и модификаторов
//
//   коэффициенты b0, b1, b2, a1, a2
     i4=ab;
     l4=length(ab);
     m4=1;
```

Рисунок 5.25 начало. Программа реализации биквадратного звена в соответствии со схемой рис. 5.24

```
//   сигнал
i0=z;
     l0=length(z);
     m0=0;
     m1=1;
     i1=zz;
     l1=length(zz);
//
//   {b0}
     mr=0, my0=pm(i4,m4);
//
//   внешний бесконечный цикл, допустимый только в
//   отладочной программе
//
     do vivod until forever;
//
//   сначала выполняются ввод очередного отсчета из порта IO(0)
//   и вычисления в КИХ-ветви
//
//   {xi}
     mx1=io(0);
//   {xi*b0,z1,b1}
     mr=mx1*my0(ss), mx0=dm(i0,m1), my0=pm(i4,m4);
//   {+z1*b1,z2,b2}
     mr=mr+mx0*my0(ss),mx0=dm(i0,m0), my0=pm(i4,m4);
//   {+z2*b2,xi->z2}
     mr=mr+mx0*my0(ss), dm(i0,m0)=mx1;
//
//   а затем в БИХ-ветви
//
//   {zz1,a1}
     mx0=dm(i1,m1), my0=pm(i4,m4);
//   {+zz1*a1,z2,a2}
     mr=mr+mx0*my0(su),mx0=dm(i1,m0),my0=pm(i4,m4);
//   {+zz2*a2, b0}
     mr=mr-mx0*my0(su), my0=pm(i4,m4);
     mr=mr(rnd);
     if mv sat mr;
     dm(i1,m0)=mr1;
//
//   вывод в порт IO(1)
vivod:    io(1)= mr1;
filter.end:
//*********************************************************
```

Рисунок 5.25 окончание.

Первичную проверку правильности работы БИХ-фильтра целесообразно проводить аналогично проверке КИХ-фильтра – измерением импульсной характеристики при воздействии на вход дельта-импульса. Характерно, что амплитуда дельта-импульса может быть выбрана аналогично КИХ-фильтру равной 32767 или минус 32768 шагов квантования, поскольку на его распространение не будет оказывать влияние эффект усиления сигнала БИХ-звеном ($y_{i-1} = y_{i-2} = 0$).

Форма сигнала на выходе рассматриваемого БИХ-фильтра при воздействии на его вход дельта-импульса амплитудой 32767 шагов квантования, отражающего вид импульсной характеристики фильтра, представлена на рис. 5.26.

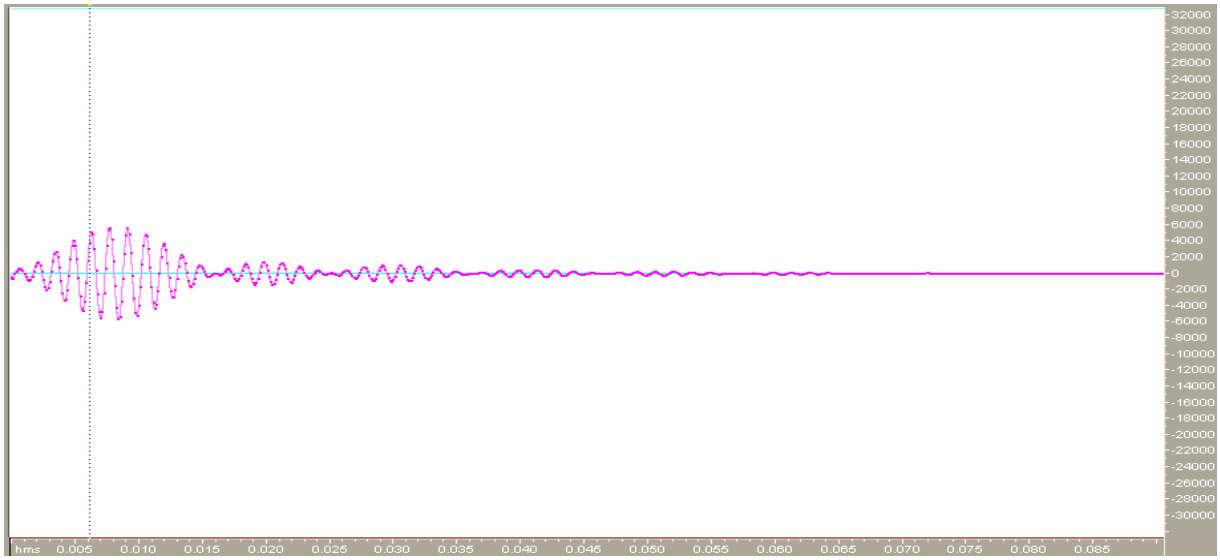



Рисунок 5.26. Импульсная характеристика БИХ-фильтра (результат отладки).

Характерно, что эффективная длительность импульсной характеристики, определяющая в конечном итоге добротность фильтра, значительно превышает длительность импульсной характеристики ранее рассмотренного КИХ-фильтра 79 порядка.

### 5.4. Специальные виды фильтров

Настоящий раздел посвящен разработке некоторых видов фильтров, часто встречающихся в разработке оборудования инфокоммуникационной техники. Ничего принципиально нового по отношению к КИХ и БИХ структурам, рассмотренным в предшествующих разделах, эти фильтры не имеют, однако их

реализация имеет ряд особенностей, которые целесообразно учитывать при разработке практических устройств.

5.4.1 Фильтр Герцеля.

В различных инфокоммуникационных приложениях часто используется операция узкополосной частотной фильтрации. Обычно целью такой фильтрации является оценка мощности сигнала в исследуемой узкой полосе частот, например, в задачах спектрального анализа, измерения затухания пилот - сигналов, фазовой автоподстройки частоты, тонального набора и др.

Одним из эффективных (в смысле использования вычислительных ресурсов) алгоритмов цифровой реализации узкополосной частотной фильтрации является алгоритм Герцеля. Этот алгоритм фактически реализует вычисления дискретного преобразования Фурье для одной частоты и базируется на двух основных моментах.

1) Центральная частота полосы пропускания фильтра $f_{Ц}$ должна быть выражена в соотношении простой дроби с частотой дискретизации $F_{Д}$:

$$f_{Ц} = F_{Д} * k/N = k = f_{Ц} * N/F_{Д}$$

Эффективная полоса пропускания фильтра при этом будет определяться выражением:

$\Delta f = F_{Д}/N.$

2) Во избежание лишних вычисления алгоритм Герцеля снижает частоту дискретизации выходного сигнала в $N$ раз. Правомерность такого подхода следует из корректной формулировки теоремы Котельникова, в соответствии с которой частота дискретизации сигнала определяется не значением самой высокой из составляющих его спектра, а эффективной шириной его полосы:

$F_{Д} \geq 2 * \Delta F,$

где $\Delta F$ – эффективная ширина полосы частот дискретизируемого сигнала.

Алгоритм Герцеля иллюстрируется схемой обработки сигнала рис. 5.27.

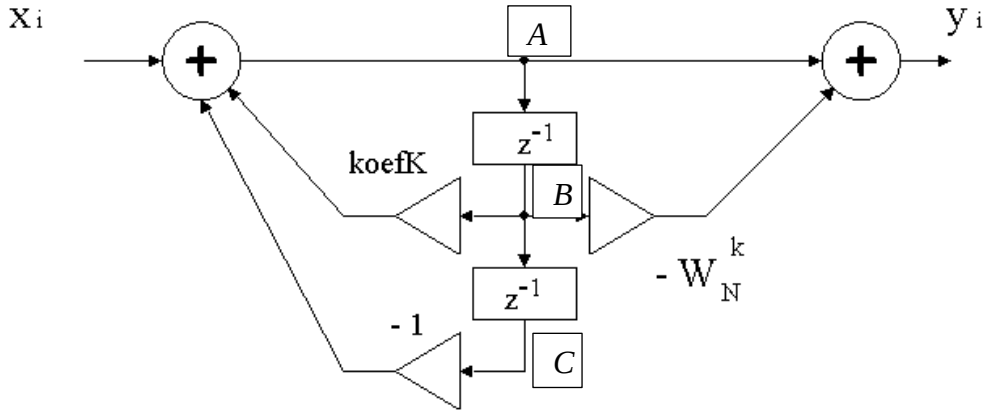



Рисунок 5.27. Фильтр Герцеля.

Аналогично ранее рассмотренной схеме фильтра с бесконечной импульсной характеристикой, - схеме биквадратного звена,- фильтр Герцеля содержит цепь рекуррентных вычислений (обратной связи) и цепь прямой связи.

Значение коэффициента $koefK$ передачи в цепи обратной связи определяется в соответствии с алгоритмом дискретного преобразования Фурье:

$$koefK = 2 * cos\left(\frac{2\pi k}{N}\right).$$

Вычисления в рекурсивной цепи проводятся на каждом такте обработки, соответствующем периоду дискретизации, в следующем порядке:

1) вычисление $A = x_i + B * koefK - C$

2) движение сигнала в линии задержки $B \rightarrow C$

3) движение сигнала в линии задержки $A \rightarrow B$

Комплексное значение отсчета выходного сигнала $y_k$, определяющего фактически спектральную линию $f_{Ц}$, вычисляется один раз в $N$ периодов дискретизации как:

$$y_k = A - B * W_N^k.$$

Мощность сигнала в полосе пропускания фильтра Герцеля может быть оценена как квадрат отсчета $y_k$, поскольку его значение представляет собой усредненную на $N$ отсчетах входного сигнала спектральную составляющую частоты:

$$P_k = y_k^2 = \left(A - B * W_N^k\right)^2.$$

Поскольку $W_N^k = e^{-j2\pi k/N} = cos(2\,\pi k/N) - j * sin(2\,\pi k/N)$,

искомая мощность может быть вычислена как:

$$P_k = [A - B * cos(2\,\pi k/N)]^2 + B^2 * sin^2\left(\frac{2\pi k}{N}\right)$$
$$= A^2 - 2 * A * B * cos\left(\frac{2\pi k}{N}\right) + B^2 * cos^2\left(\frac{2\pi k}{N}\right) + B^2 * sin^2\left(\frac{2\pi k}{N}\right)$$
$$= A^2 + B^2 - 2 * A * B * cos\left(\frac{2\pi k}{N}\right) = A^2 + B^2 - A * B * koefK.$$

Текст отладочной программы на языке Ассемблера процессора ADSP-2181, реализующей алгоритм фильтра Герцеля, представлен с необходимыми комментариями на рис. 5.28.

```
//#########################################################
//
// область размещения программы
//
.SECTION/PM program;
//
//   Расчет значения коэффициента koefK:
//   значение центральной частоты = 900 Гц
//   значение частоты дискретизации = 8000 Гц
//   при N = 80, k = 900*80/8000 = 9
//   koefK = 2*cos(2*pi*9/80)= 1.520812
//
//   !ВНИМАНИЕ!
//   значение модуля koefK превышает 1, значит
//   далее в операциях умножения koefK будет фигурировать
//   в БЕЗЗНАКОВОМ (u) формате.
//
//   (В качестве напоминания: в 16-разрядной сетке в знаковом (s)
//   формате могут быть представлены целые числа в диапазоне
//   от -32768 до 32767 и соответствующие им дробные представления
//   - в диапазоне от минус 1 до +1-2^-15.
//   Соответственно, беззнаковые (u) числа могут быть представлены
//   в диапазоне от 0 до 65535 как целые, их дробные
//   представления - от 0 до 2-2^-15.)
//
//   Для представления в целочисленном формате значение koefK
```

Рисунок 5.28 начало. Программа реализации фильтра Герцеля.

```
//   следует умножить на 32768, т.е.


//   koefK = 1.520812*32768 = 49834

#define koefk 49834;
          my0=koefk;

//   0x8000 в знаковом формате соответствует дробному числу минус 1

     my1=0x8000;    // my1 = -1
     mx0=0;
     mx1=0;
     do vivod until forever;
//-----------------------------------------------------------------
//   Фаза рекурсивной обработки,
//   выполняется с частотой дискретизации сигнала
//   Размещение информации:
//   mx0 - значение B
//   mx1 - значение C
//   my0 = koefK
//   my1 = -1
//
     cntr=80;
     do faz1 until ce;
          mr=0;
//
//        Ввод очередного отсчета
          mr1=io(0);
//
//        Xi+B*koefK
//        ! koefK обрабатывается в БЕЗЗНАКОВОМ формате
          mr=mr+mx0*my0(su);
//
//        A=Xi+B*koefK+(-C)
//        -C = C*(-1)
//   Поскольку C и (-1) обрабатываются в знаковом формате,
//   допустимо совместить вычисление итоговой суммы произведений
//   с округлением результата (rnd)
//   и плюс к этому - с перемещением (сдвигом) в линии задержки B?C
          mr=mr+mx1*my1(rnd), mx1=mx0;

//   ограничение результата по переполнению для предотвращения
//   искажений перегрузки, что в совокупности с ранее выполненным
```

Рисунок 5.28 продолжение.

```
//   округлением ограничивает результат 16-разрядной сеткой с
//   минимальными отрицательными последствиями
          if mv sat mr;

//
//              перемещение (сдвиг) по линии задержки A?B
faz1:         mx0=mr1;
//-------------------------------------------------------------
//
//   Вычисление мощности сигнала в полосе прозрачности фильтра с
//   усреднением по N отсчетам; выполняется 1 раз за
//   N периодов дискретизации
//
//         {вычисление A^2}
          mr=mx0*mx0(ss);
//
//         {вычисление A^2 + B^2}
          mr=mr+mx1*mx1(ss);
//
//         {вычисление B * koefK}
          mf=mx1*my0(su);
//
//         {вычисление P}
          mr=mr-mx0*mf(rnd);
//
//         вывод резтата
vivod:    io(1)=mr1;
//*****************************************************
```

Рисунок 5.28 окончанннне.

### 5.4.2 Фильтр Найквиста

Цифровые инфокоммуникационные системы, оперируя двумя логическими уровнями 0 и 1, обычно ассоциируются с передачей прямоугольных импульсов, и в значительном количестве случаев – передаче по симметричным и коаксиальным кабелям, оптическим волокнам без спектрального или частотного уплотнения – использование прямоугольных импульсов как носителей информации имеет место. Однако в радиосистемах и оптических системах со спектральным и частотным уплотнением, в которых действуют жесткие ограничения на ширину используемого спектра частот, передача прямоугольных импульсов недопустима. Спектр прямоугольного импульса, как

известно, бесконечен, и даже при форме импульса, близкой к трапеции (реальный случай) занимает неоправданно широкую полосу частот и способен создавать помехи в соседние частотные диапазоны.

Ограничение спектра изменяет форму импульса и приводит к изменению его длительности (уширению), в результате которого часть его энергии попадает на позиции соседних импульсов, повышая вероятность ошибки их приема. Этот эффект – межсимвольная интерференция – широко известен в кабельных системах, в которых он обусловлен влиянием линейных искажений в направляющих системах [10].

Фильтр Найквиста решает проблему межсимвольной интерференции при ограничении спектра импульсов приближением их формы к виду функции $sin\,(x)/x$. Как известно, эта функция имеет единичное значение в точке $x = 0$, соответствующей позиции середины передаваемого импульса, а в точках $\pm\pi \cdot n$, соответствующих позициям середин соседних импульсов (точки принятия решения), проходит через 0, не создавая межсимвольной помехи (интерференции). Эффект воздействия фильтра Найквиста на сигналы прямоугольной формы для трех импульсов иллюстрируется рис. 5.29, демонстрируя отсутствие их взаимного влияния.

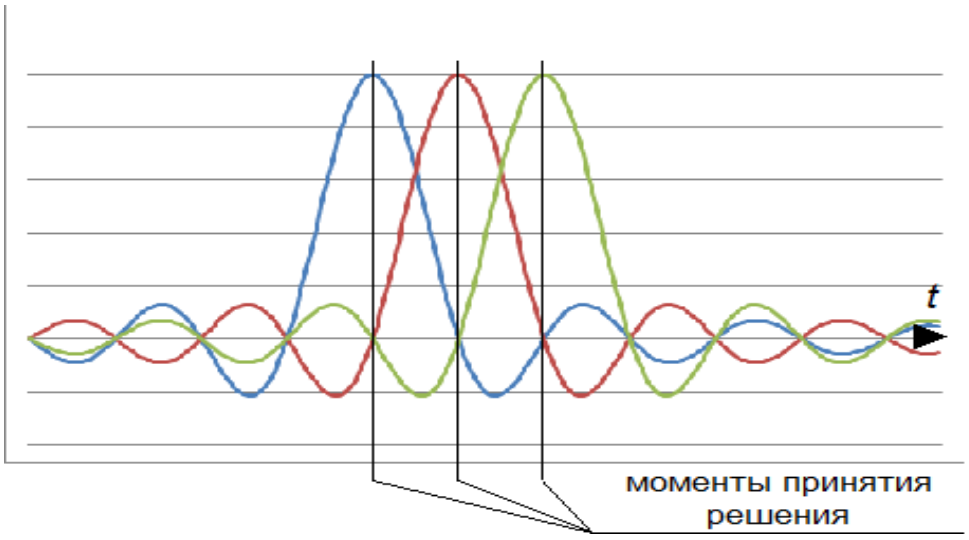



Рисунок 5.29. Устранение межсимвольной интерференции фильтром Найквиста

Амплитудно - частотная характеристика фильтра Найквиста представлена на рис. 5.30; она содержит участок с единичным коэффициентом передачи, участок с нулевым, а также переходной участок, описываемый функцией приподнятого косинуса.

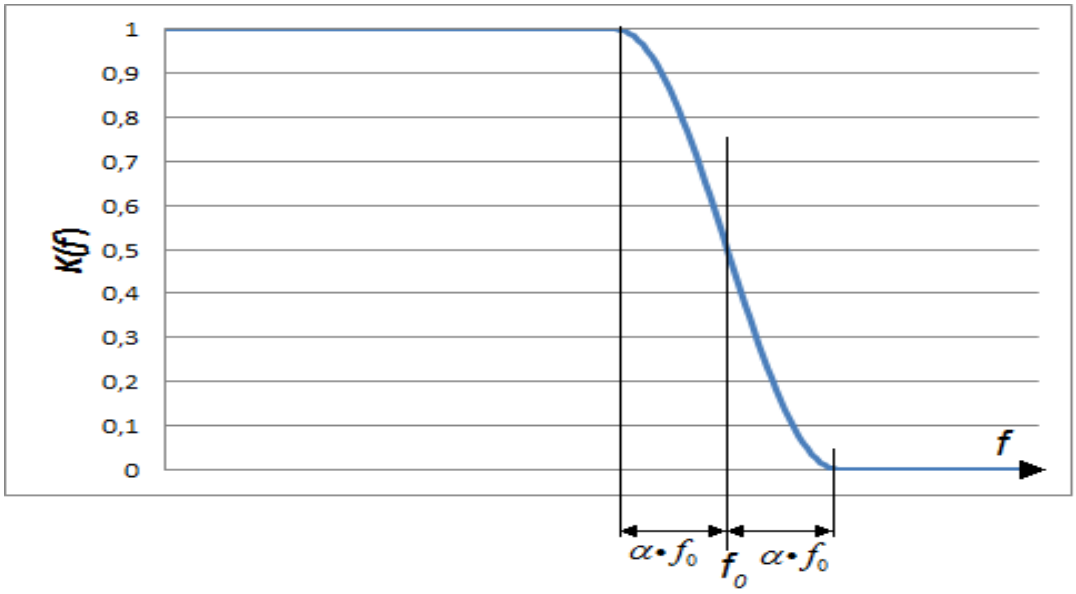



Рисунок 5.30. АЧХ фильтра Найквиста.

$$K(f) = \begin{cases} 1, f < f_0(1-\alpha) \\ \frac{1}{2}\left\{1 + cos\frac{\pi \cdot (|f| - f_0(1-\alpha))}{2f_0\alpha}\right\}, f_0(1-\alpha) \le f \le f_0(1+\alpha) \\ 0, f > f_0(1+\alpha) \end{cases}$$

По этому участку АЧХ фильтра Найквиста часто называют функцией приподнятого косинуса, или RC (Raised Cosine). Коэффициент $\alpha$ называется относительным уширением, его значение обычно устанавливается в пределах 0.25…0.35.

Значение $f_0$ определяется обеспечиваемой скоростью передачи информации. Фазо-частотная характеристика фильтра Найквиста должна быть линейной, поэтому обычно фильтр Найквиста реализуется на КИХ-архитектуре, а ФЧХ рассчитывается в соответствии с рекомендациями раздела 2.

В радиосистемах, помимо необходимости ограничения спектра передаваемого сигнала, возникает необходимость ограничения спектра входного для снижения влияния шумов и помех от сигналов соседних диапазонов. В этом случае фильтр Найквиста разделяется на две части - одна устанавливается на передаче, а другая – на приеме. Во избежание лишних потерь на частоте $f_0$ (в 2 раза), характеристика переходного участка формируется как квадратный корень из приподнятого косинуса, таким образом, что каскадное включение двух частей формирует полный фильтр Найквиста.

Характеристика каждого из таких фильтров называется функцией корня из приподнятого косинуса или RRC (Root Raised Cosine).

Практическая реализация фильтра Найквиста должна проводиться по ранее рассмотренной в разделе 1.1 методике как обычного КИХ-фильтра с линейной ФЧХ.

**ПРИЛОЖЕНИЕ А**

Пример титульного листа

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ**

---

Кафедра многоканальных телекоммуникационных систем

# КУРСОВАЯ РАБОТА

по дисциплине

**Микропроцессорная техника в инфокоммуникациях**

Выполнил:

студент гр. БЗС1801

Петров П.П.

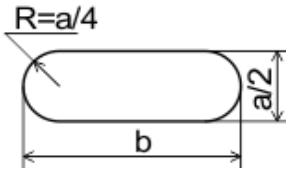
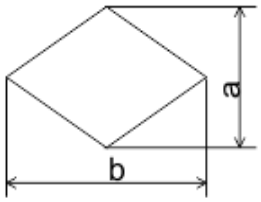
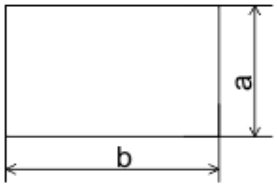
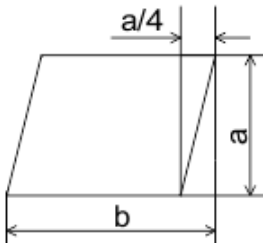
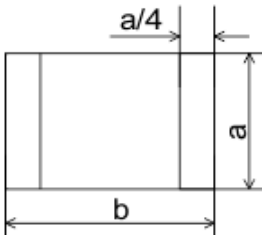
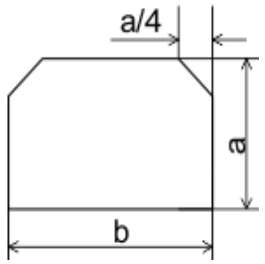
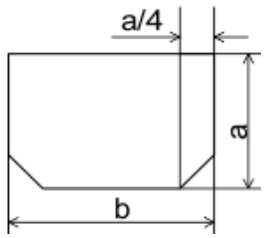
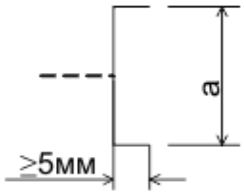
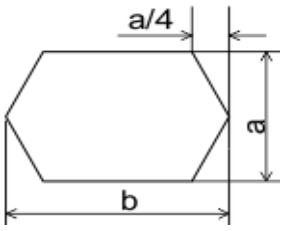
Проверил:

Доцент А.А. Иванов.

Москва 2022

**ПРИЛОЖЕНИЕ Б**

Правила оформления блок-схем алгоритмов

| | | | |
|---|---|---|---|
| Начало-конец |  | Условный переход |  |
| Процесс |  | Ввод-вывод |  |
| Подпрограмма |  | Начало цикла (пред-условие) |  |
| Соединитель |  | Конец цикла (пост-условие) |  |
| Комментарий |  | Цикл с параметром<br>[For-To-Do] |  |

## СПИСОК ЛИТЕРАТУРЫ

1. Шаврин С.С., Климов Д.А. Реализация базовых операций защиты информации на сигнальных процессорах (часть 1) Учебное пособие.– М.: МТУСИ, 2012.– 67 с.

2. Лайонс Р. Цифровая обработка сигналов: Второе издание. Пер. с англ. – М.: ООО «Бином – Пресс», 2011. – 656 с.

3. Мельник С.В., Зуйкова Т.Н., Мусатова О.Ю., Шаврин С.С. Исследование эффектов ограничения разрядной сетки при цифровой обработке сигналов. Практикум.– М.: МТУСИ, 2017. – 22 с.

4. Digital signal processing applications using the ADSP-2100 Family. Volume 1. Prentice Hall, Englewood Cliffs, NJ 07632 – 591 p.

5. Шаврин С.С. Защита информации в многоканальных телекоммуникационных системах. Часть 2. Учебное пособие.– М.: МТУСИ, 2003. – 112 с.

6. Digital signal processing applications using the ADSP-2100 Family. Volume 2. Prentice Hall, Englewood Cliffs, NJ 07632 – 657 p.

7. Руководство пользователя по сигнальным микропроцессорам семейства ADSP-2100. Под ред. А.Д.Викторова. Пер. с англ. – М.: СПбГЭТУ, 1997.– 520 с.

8. Марков С. Цифровые сигнальные процессоры. Книга 1.– М.: фирма МИКРОАРТ, 1996.

9. Андреев И.И., Ланнэ А.А. MATLAB для DSP: SPTool - инструмент для расчёта цифровых фильтров и спектрального анализа сигналов // Цифровая обработка сигналов. 2000. № 2. С. 6–13.

10. Гордиенко В.Н., Тверецкий М.С. Многоканальные телекоммуникационные системы. Учебник для вузов. – 2-е издание, испр. и доп. – Горячая линия - Телеком, 2013. – 396 с.