МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

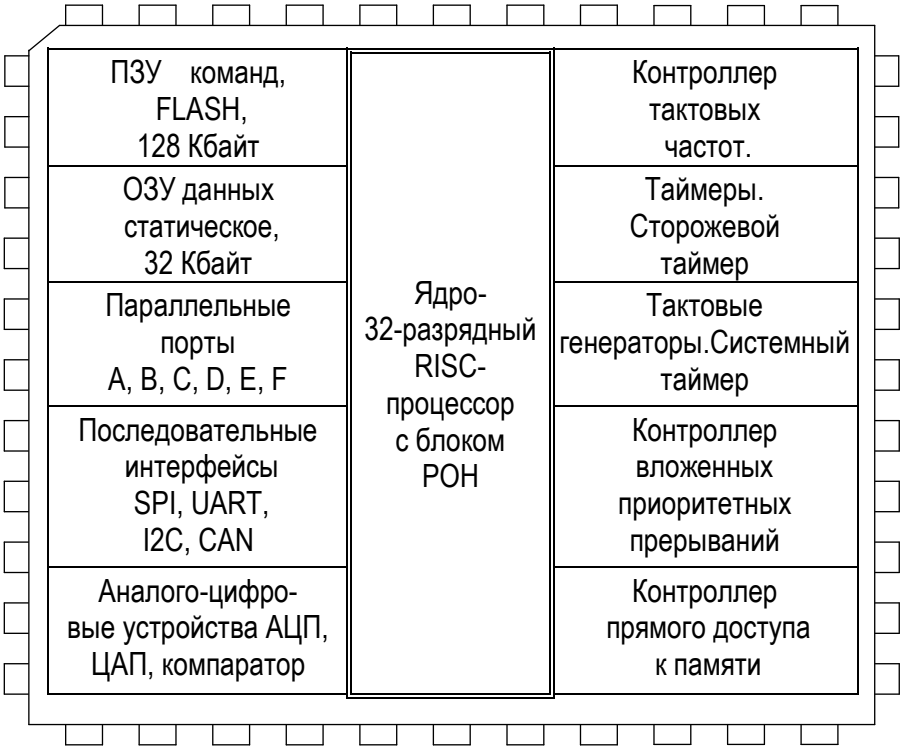
КАФЕДРА ЭВМ

А. В. Кистрин

Микропроцессорные системы и интерфейсы

Методические указания к лабораторным работам

Часть 1



Рязань 2019

Работа 1. Изучение система команд микроконтроллера ARM Cortex-M3. Проектирование программ в среде Keil µVision

Цель работы: получение навыков проектирования в среде Keil µVision, изучение системы команд микроконтроллера ARM Cortex-M3, примеров записи команд, методов анализа результатов выполнения программы.

Микроконтроллер ARM Cortex-M3 – 32-разрядный быстродействующий МК нового поколения семейства STM32, разработанный на основе процессорного ядра ARM компании Advanced RISC Machine. Применяется при разработках современных интеллектуальных электронных устройств. В лабораторных работах исследуются микроконтроллеры и отладочные платы, выпускаемые отечественной фирмой Миландр (г. Зеленоград, сайт www.milandr.ru). В лабораторных работах изучаются система команд микроконтроллеров Cortex-M3, и основы языка ассемблер, приемы написания программ, отладка программ в среде разработки *Kejl* , исследование работы программ с использованием отладочных плат фирмы Миландр серии *1986BExx.*

Процессорное ядро Cortex-M3 содержит 32-разрядное АЛУ, соединенное с блоком 32-разрядных регистров r0 – r15 различного назначения.

r0 – r12 -я регистрs общего назначения (РОН), предназначенные для хранения данных; и адресов ячеек памяти. Результаты выполнения основных операций представляются в РОН в виде 32-разрядных слов.

r13 - SP - Stack Pointer - указатель стека, содержит адрес вершины стека, который уменьшается при записи в стек;

r14 – LR - Link Register - регистр связи, используется для сохранения адреса возврата при переходе на подпрограмму.

r15 - PC - Program Counter - счетчик команд, содержит адрес выполняемой команды. Бит 0 всегда 0, так как все коды команд содержат 2, или 4 байта.

Процессор имеет отдельный регистр состояния программы xPSR –Extended Program Status Register, в старших разрядах которого (31..27) содержатся признаки результата выполненной операции: n (Negative) – признак отрицательного результата; z (Zero) – признак нулевого результата; c (Carry) – признак переноса при сдвигах или сложении, или заема при вычитании; v (Overflow) – признак переполнения ; q – признак накопления.

Память процессора Cortex-M3 имеет разрядность данных - 1 байт, разрядность адреса - 32 бита. В 16-ричной системе адрес определяет 8 цифр. Совокупность всех возможных адресов образует адресное пространство. Это максимальное количество адресуемых ячеек (4 Гбайт). Общее адресное пространство разделено на секции, доступ к которым осуществляется через шины, работающие независимо друг от друга, подобно гарвардской архитектуре.

System – секция адресов системных регистров ядра и системной периферии.

External RAM – секции адресов для доступа к внешней системной шине, предназначенной для подключения устройств памяти или периферийных устройств.

Peripheral - адреса регистров периферийных устройств м портов ввода – вывода.

Data – адреса ячеек ОЗУ данных.

Code секция, предназначенная для хранения программ.

Физическая реализация устройств памяти зависит от конкретной модели, при этом используется только часть адресов общего адресного пространства. Области

адресного пространства, используемые в МК Cortex, и содержащие адреса физически существующих ячеек имеют следующие параметры.

В младшие адресах памяти программ расположена область BOOT ROM, которая предназначена для хранения программы запуска микроконтроллера.

Область ПЗУ команд, начинающаяся с адреса 0x0800_0000, содержащая 128 Кбайт, предназначена для хранения основной рабочей программы пользователя.

Область ОЗУ данных начинается с адреса 0x2000_0000, содержит 32 Кбайт. В этой область также располагаются стек.

Область Peripheral, начинающаяся с адреса 0x4000_0000, содержит адреса регистров периферийных устройств.

Ассемблер является аппаратно - зависимым языком. Он зависит от структуры и аппаратной реализации конкретного процессора, поэтому ассемблерные программы для различных процессоров и микроконтроллеров существенно различаются. Ассемблерные программы эффективнее эквивалентных программ, транслированных с языков высокого уровня. Они занимают меньший объем памяти и выполняются быстрее.

Разработка программ на ассемблере выполняется в несколько этапов.

Формулируется постановка задачи и составляется алгоритм.

Для всех переменных, используемых в задаче, выбираются форматы данных и регистры или ячейки памяти для размещения данных. для сложных программ целесообразно составить таблицу размещения данных.

Программа должна содержать комментарий, поясняющий назначение и цель выполняемых действий.

Отличительной особенностью ассемблера микроконтроллера Cortex и процессоров ARM является возможность использования суффиксов совместно с мнемониками команд и большое количество модификаций форматов команд, что существенно расширяет функциональные возможности системы команд. Заметим, системы команд процессоров предыдущих поколений таких возможности не имеют.

Строка ассемблерной программы содержит несколько полей, разделенных табуляцией или пробелами.

1) Поле метки. Начинается обязательно с первой позиции строки. Двоеточие после метки (как в ассемблере Intel или AVR) не ставится. Имя метки должно начинаться с буквы. Метка определяет адрес команды для перехода на эту команду.

2) Поле мнемоник команд или директив ассемблера. Мнемоники команд и директивы ни в коем случае не должны записываться с первой позиции строки.

3) Поле операндов. Указано место нахождения операндов в соответствии с использованным методом адресации. Методом адресации называют способы доступа к командам программы и данным.

4) Поле комментария. Признак начала комментария - точка с запятой. В комментариях допускается русский шрифт. Символ начала комментария (;) позволяет при отладке временно исключить из компиляции строку программы («закомментировать»).

Обобщенный формат команд пересылки, арифметических и логических операций, позволяет получить большое количество вариантов записи команд.

op {s}{cond} {rd}, rm, operfnd2. Обозначения:

op - мнемоника операции, **{s}, {cond}**- необязательные суффиксы; **{rd}, rm, operand2** - операнды.

суффикс {cond}	приз-нак	результат предыду-щей операции…-
eq (equal)	z = 1	-равен нулю
ne (not equal)	z = 0	-не равен нулю
cs (c set)	c = 1	был перенос
cc (c clear)	c = 0	не было переноса
mi (minus)	n = 1	-отрицательный
pl (plus)	n = 0	-положительный
vs (v set)	v = 1	есть переполнение
vc (v clear)	v = 0	нет переполнения

Суффикс {s} обеспечивает запись признаков результата данной команды в регистр состояния, при его отсутствии признаки не изменяются.

Суффикс {cond} (condition – условие), обеспечивает выполнение команды,. если условие выполняется, иначе команда будет пропущена. Назначение операндов:

{rd} – необязательный регистр-получатель результата (register destination).

rm – регистр первого операнда (register master);

operand2 - второй операнд, его способ записи определяет метод адресации.

Обобщенный формат позволяет получить шесть форматов для каждой команды.

Форматы команд обработки данных

№	Метод адресации	Формат команды
Двухадресные команды обработки данных, результат в *rm*		
1	непосредственная	*op{s}{cond} rm, #imm8.*
2	Регистровая	*op{s}{cond} rm, rs.*
3	регистровая с масштабированием	*op{s}{cond} rm, rs, shift #n*
Трехадресные команды обработки данных, результат в *rd*		
4	непосредственная	*op{s}{cond} rd, rm, #imm8*
5	регистровая	*op{s}{cond} rd, rm, rs*
6	регистровая с масштабированием	*op{s}{cond} rd, rm, rs, shift #n.*

В двухадресных командах регистр *rd* отсутствует и указаны два операнда: первый – *rm* и второй - *Operand_2* в вде *#imm8, rs,* или *rs shift #n*. Операция выполняется над первым и вторым операндами, а ее результат записывается в *rm* вместо первого операнда, исходное значение которого теряется.

В трехадресной команде указаны *rd* - получатель, *rm* –первый операнд, *Operand_2* – второй операнд. Операция выполняется над первым и вторым операндами, а ее результат записывается в *rd*. Исходное значение операнда, содержащееся в *rm,* после выполнения команды не изменяются. Это важное достоинство трехадресных команд.

Вид второго операнда *Operand_2* определяет метод адресации.

Мнемоники операций сдвига ***shift***	
asr	Арифметический сдвиг вправо
lsr	Логический сдвиг вправо
lsl	Логический сдвиг влево
ror	Циклический сдвиг вправо
rrx	Циклический сдвиг вправо на один бит через перенос

Непосредственная адресация.

Второй операнд – 8-разрядная константа, которая может быть сдвинута влево на любое количество разрядов в пределах 32-разрядного слова путем записи нулей в младшие разряды. Систему счисления, в которой записана константа, указывают префиксом: 0x - 16-ричная; 0b -

двоичная; префикса нет - десятичная.

Регистровая адресация . Операнды находятся в регистрах: первый - в **rm,** второй в **rs** (register source). Результат в rd , или в **rm –** вместо первого операнда.

Регистровая адресация с масштабированием: Операция выполняется над содержимым **rm** и сдвинутым на n разрядов содержимым **rs**. После выполнения операции содержимое регистра rs сохраняется. Мнемоники операции сдвига *shift* приведены в таблице. Результатом сдвига является масштабное преобразование второго операнда – умножение, или деление на 2^n.

Форматы 1..6 используют в Командах пересылки, арифметических и логических операций.

Для операций пересылки и сложения предусмотрены форматы команд с непосредственной адресацией и расширенной разрядностью константы**: movw - п**ересылка 16-разрядной константы в младшее полуслово 32-разрядного регистра и заполнение нулями старшего полуслова; **movt- п**ересылка 16-разрядной константы в старшее полуслово и сохранение без изменения младшего полуслова: Запись в регистр 32-разрядных слов можно выполнить командами movw, а затем movt, либо посредством директивы ассемблера **ldr rm, = imm32.**

Команды обращения к памяти

Обращение к памяти - это загрузка регистра из памяти (мнемоника ldr от слов Load Register), или запись из регистра в память (str – Store Register) – выполняют команды с косвенной адресацией различных модификаций.

Результаты операций формируется в виде 32-разрядных слов, содержащихся в РОН,независимо от разрядности аргументов. Дданные, используемые впоследствии, и хранящиеся в памяти могут иметь разрядность 1, 2, 4, или 8 байт, поэтому в командах обращения к памяти указываю разрядность данных.

Форматы команд обращения к памяти

№	Метод адресации	Формат команды
7	косвенная	*op{type}{cond} rt, [rn]*
8	косвенная со смещением	*op{type}{cond} rt, [rn, #offset]*
9	косвенная с пост-индексированием	*op{type}{cond} rt, [rn], #offset*
10	косвенная с пре-индексированием	*op{type}{cond} rt, [rn, #offset]!*
11	косвенная с адресами в двух регистрах	*op{type}{cond} rt, [rn, rm, #offset]*

op – одна из мнемоник операции: *str (Store Register)* - запись содержимого регистра в память; *ldr (Load Register)* - загрузка регистра из памяти;

{type} – суффикс, определяющий разрядность данных: *b* –байт, *h* – 16-разрядное полуслово, отсутствие суффикса – 32-разрядное слово, *d* – 64-разрядное двойное слово.

sb или ***sh*** – суффиксы для команды *ldr* , необходимые при загрузке 8- или 16-разрядных чисел со знаком в 32-разрядный регистр *rt* для выполнения операции расширения знака - заполнения старших разрядов слова значениями, соответствующими знаку числа (0 для положительных чисел, или 1 – для отрицательных). В обозначениях суффиксов *содержитсяя* буква s (от слова signed),

{cond} - суффикс условного выполнения команды;

rt – регистр для временного хранения данных, участвующих в пересылке;

[*rn*] - **указатель адреса,** содержащий адрес ячейки памяти, в которой хранится операнд, записывается в квадратных скобках; **[*rm*]** – дополнительный указатель адреса ячейки памяти;

offset – смещение, целое число со знаком. Адрес ячейки памяти равен сумме содержимого регистров rm, rn и числа offset.

Назначение команд различных форматов:

7 - обращение к одной ячейке памяти, адрес которой был записан в указатель *rn:*

8 - обращение к элементам массива в произвольном порядке адреса которых представляют виде суммы базового адреса, записанного в регистр *rn*, и смещения offset.

9 - обращение к смежным элементам массива в циклических программах. (Смещение записывается за пределами квадратных скобок) Адрес ячейки памяти определяется содержимым регистра *rn*. После обращения к элементу к содержимому регистра *rn* прибавляется смещение *#offset* (равное 1, 2, 4, или -1, -2, -4), и новый адрес записывается в регистр *rn*. Это адрес следующего элемента массива, подготовленный для повторения цикла.

10 – обращение к ячейке, аадрес которой вычисляется как сумма содержимого регистра и смещения, и записывается в регистр *rn* перед обращением к ячейке памяти. Предварительное индексирование обозначает восклицательный знак в конце записи. Команды форматов 10 и 11 используют совместно при организации стековой памяти.

11. Адрес ячейки памяти равен сумме содержимого регистров *rn, rm* и числа *offset*. После обращения содержимое регистров остается неизменным. Данный метод адресации позволяет создавать сложные структуры данных.

Описание программы PR_1.

1 Настройка конфигурации. Программа для микроконтроллера вначале должна настроить его конфигурацию. Эту функцию выполняют директивы инициализации процессора. Директивы не транслируются в машинные коды, а используются в служебных целях для разметки программы, присвоение константам символьных имён, определения размеров секций памяти. В сложных программах директивы и команды инициализации выделяют в отдельный файл StartUp.

Строка 1 – комментарий. Начало комментария – точка с запятой. Указано имя программы. Символ «;» позволяет временно исключить выполнение строки программы при отладке (закомментировать).

Директивы area разбивают программу на отдельные секции.

В строке 2 определена секция с именем stack (стек), тип секции noinit (не определен), атрибут, определяющий доступ к данным - readwrite (чтение и запись).

В строке 3 указан размер этой секции, а в строке 4 установлена метка, адрес которой (0x20000400) будет определять вершину стека. Метка записана с первой позиции строки!

В строке 5 определена секция с именем reset (запуск), тип секции data (данные), атрибут, определяющий доступ к данным - readonly (только чтение).

В строке 6 определена секция с именем program (программа), тип секции code (код), атрибут, определяющий доступ к данным - readonly (только чтение).

В строках 6 и 7 записаны директивы dcd (объявить двойное слово), резервирующие ячейки памяти для записи адресов вершины стека и начала программы, для которых в программе записаны метки.

Директива entry (ввод) и метка start определяют начало исполняемой программы.

Строки 2-10 выполняющие инициализацию микроконтроллера, будут являться впоследствии началом всех программ.

Программа пользователя, начало которой отмечено меткой *start*, содержит примеры команд для изучения правил записи и для анализа результатов выполнения, составлена из подпрограмм, представляет собой бесконечны1 цикл.

```
1   ;    PR_1
2        area stack, noinit, readwrite
3        space 0x400
4   stack_top
5        area reset, data, readonly
6        area program,code,readonly
7        dcd stack_top
8        dcd start
9        entry
10  start
11           bl    pp1
12  ;        bl    pp2
13  ;        bl    pp2
14           b     start
15  pp1
16       movs   r0,  #0
17       moveq  r1,  #32
18       movne  r1, #150
19       movs   r2, #-1
20       movs   r3, #255
21       movw r5,    #0xcdef
22       movt    r5, #0x89ab
23       ldr r6, = 0xfedcba98
24       mov    r4,  r0
25       adds   r4,      r1
26       add    r4,      r2
27       movs   r7,  r4, lsr # 2
28       add    r8,  r4,  #25
29       add    r9,  r7,  r4
30       add     r10, r4, r7, lsr # 2
31       push   {r4}
32       push   {r5-r7}
33       pop{r1, r0}
34       pop    {r3, r2}
35       mov    r0, #0x20000000
36       str  r5, [r0],#4
37       str  r6, [r0],#4
38       str  r7, [r0],#4
39       ldr  r8, [r0,#-4]!
40       ldr  r9, [r0,#-4]!
41       ldr  r10, [r0,#-4]!
42       bx  lr
43  pp2
44       ldr r0, = 0x12345678
45       rbit r1,  r0
46       rev r2,  r0
47       rev16   r3,  r0
48       revsh   r4,  r0
49       bx  lr
50  pp3
51       bx  lr
52       end
```

2. Использование подпрограмм.

Сложные программы разбивают на подпрограммы с именами, например, pp1, pp2, pp3. В начале подпрограммы записывается метка (имя подпрограммы), которая начинается с первой позиции строки.

При вызове подпрограмм для сохранения адреса возврата используется регистр связи LR (R14). Для вызова подпрограммы используется команда перехода на метку (строки 11, 12). Мнемоника bl – Branch Link Register означает ветвление с запоминанием адреса возврата в регистре связи. В конце подпрограммы необходимо записать команду возврата bx lr. (В ассемблерах микроконтроллеров Cortex не

используются команды процессоров старых типов для вызова подпрограмм и возврата call и ret.) При запуске приведенной программы будет выполняться только вызов первой подпрограммы, вызов остальных (строки 12, 13) временно закомментирован, в начале строки поставлена точка с запятой. После исследования первой подпрограммы необходимо запретить ее вызов и разрешить работу другой подпрограммы.

Программа пользователя обычно содержит бесконечный цикл, который можно многократно повторять при отладке. Команда b – branch – ветвление (строка 14) выполняет безусловный переход на начало пользовательской программы, отмеченное меткой start.

3. Пересылка данных. Используются для определенного размещения исходных данных в регистрах микроконтроллера, в ОЗУ данных, а также в регистрах периферийных устройств и для инициализации вычислительных процессов.

Примеры команд пересылки с непосредственной адресацией в строках 16-24.

Строка 16. Пересылка байта в регистр (формат 1). Загружает в регистр r0 константу 0. Содержит суффикс s, поэтому после выполнения команды будет установлен признак z = 1.

Строка 17. Условная пересылка байта в регистр r1. Ммнемоника содержит суффикс eq. Эта команда не изменяет признаки. Константа записана в десятичной системе счисления.Команда будет выполняться, так как в регистре признаков содержится z = 1.

Строка 18. Условная пересылка при выполнении противоположного условия (мнемоника содержит суффикс ne). Эта команда не будет выполняться, так как в регистре признаков содержится z = 1.

Строки 16,18 иллюстрируют выполнение на ассемблере Cortex алгоритмов, содержащих условные переходы вида «Если – То - Иначе».

Строка 19. Пересылка в r2 отрицательного числа. Число будет записано в дополнительном коде, в старших разрядах которого будет установлена 1.

Строка 20. Пересылка максимального положительного 8-разрядного числа.

Строка 21. Пересылка младшего полуслова (два байта) в r5 с обнулением старшего полуслова. Строка 22 Пересылка старшего полуслова в r3 с сохранением младшего полуслова. Для записи в регистр 32-разрядного слова необходимо вначале командой movw записать младшее полуслово, а затем – командой movt – старшее. Порядок важен!

Строка 23. Директива ассемблера для пересылки 32-разрядного слова в r4, упрощает запись, выполняется за два такта.

Строки 24 - 26 Команды пересылки и суммирования с регистровой адресацией формат 2.

Строка 27. Команда пересылки с регистровой адресацией и масштабированием (формат 3) позволяют за один машинный цикл переслать содержимое одного регистра в другой и выполнить сдвиг. При сдвиге влево (или вправо) происходит умножение (или деление) исходного числа на 2^n, где n – количество разрядов, на которое выполняется сдвиг. Последний выдвигаемый разряд записывается в регистр состояния xPSR как признак «C».При логических сдвигах освобождающиеся разряды заполняются нулями. При арифметическом сдвиге вправо старшие разряды заполняют значения знакового бита.

Строки 29-30 содержат трехадресные команды форматов 4 – 6.

Обращение к стеку (строки 31 - 34)**.** Стек реализован как участок системной памяти, адресуемый указателем SP, работающий как буфер типа LIFO. Стек используется для сохранения и последующего восстановления содержимого регистров при выполнении подпрограмм и операций обработки данных.

В процессоре Cortex для работы со стеком предусмотрены команды push (загрузка регистров в стек) и pop (извлечение регистров из стека). После мнемоники команды **в фигурных скобках** записывается список регистров (reglist), который может содержать один регистр, или перечисление нескольких регистров через запятую, или через тире. В строке 32 в стеке сохраняется адрес указателя, который впоследствии в строке 36 восстанавливается. Список регистров записывается **обязательно в фигурных скобках**, даже если регистр в команде всего один!

В стеке для каждого операнда отводится 4 байта. При записи данных в стек вначале выполняется уменьшение содержимого SP на 4, а затем запись содержимого регистров в стек . При чтении выполняются противоположные действия: вначале - чтение, а затем увеличение содержимого SP на 4.

5. Обращение к памяти. В строке 35 (заблаговременно !) выполняется пересылка 32-разрядного адреса (2000_0000), соответствующего началу физически реализованного ОЗУ, в регистр r0, который будет использован как указатель адреса при косвенной адресации. Использована команда mov, предназначенная для пересылки одного байта (формат 1). Константа записана со сдвигом, заданным нулями в младших разрядах числа.

Строка 36. Запись 32-разрядной константы (4 байта) из регистра r5 по адресу, который был записан в r0, с последующим увеличением адреса на 4 (это пост-индексирование). Правило записи многобайтных чисел: **младший байт числа записывается по младшему адресу,** который содержится в r0.

В строках 37 – 38 выполняется запись в память 32-разрядных слов из регистров r6, r7 в память в виде элементов, содержащих 4 байта.

В строках 39 – 41 выполняется загрузка в регистры r7, r8, r9 из памяти.

В строках 43 – 49 приведена подпрограмма, содержащая команды для изменения порядка байт и полуслов в слове.

В строках 50 – 51 заготовлены команды для начала и завершения подпрограммы пользователя pp3.

Проектирование и отладка программ в системе Keil µVision

В лабораторных работах будет использоваться система Keil µVision, версии 5.16a. Для установки программы запустите файл MDK516a.exe. Дополнительно запустите файл mdr_spl_v1.4.1(для Keil 5).pack для установки параметров конфигурации микроконтроллеров ARM Cortex от фирмы Миландр. Также перепишите папку milfbdr в каталог ARM установленной программы Keil_v5.

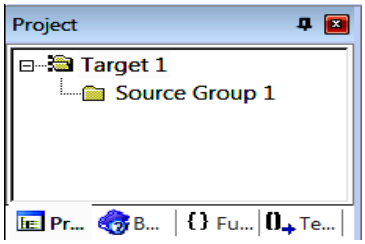


Рис. 1.1. Менеджер проекта

Этапы работы в системе Keil µVision

1) Создайте новый проект. Для размещения файлов проекта необходимо создать папку, например, D:\ USERS\ 545\LR_Cortex. Имена папки, каталогов и путей не должны содержать русских букв и пробелов. Запустите Keil. Из

меню Project/ New µVision Project, откройте созданную для проекта папку, введите имя проекта, сохраните проект.

В открывшемся окне Select Device for Target ‘Target 1’ укажите устройство MDR32F9Q2I, нажмите ОК. На вопрос «Copy «startup MDR32F9.s» to Project Folder and Add File to Project?» необходимо ответить «нет». В программах на ассемблере этот файл не требуется.

Откройте окно проекта (менеджер проекта) из меню View > Project Window и закладку Project для отображения структуры проекта.

2) Создайте новый файл. Командой File->New создайте новый файл. Командой Save As запишите этот файл в каталог проекта с именем, например, PR_1.s. Расширение *.s определяет тип файла – ассемблерный. (По умолчанию выбирается расширение *.c, соответствующее файлу на языке Си). В окне проекта установите курсор на папку Source Group 1, нажмите правую кнопку, укажите в контекстном меню Add Existing Files to Group 'Source Group 1'. В открывшемся окне обязательно укажите тип и имя файла.

Для создания файла для новой работы откройте файл предыдущей работы (например, PR_1) и сохраните его с новым именем (PR_2.s). В окне (менеджере) проекта укажите Source Group 1, нажмите правую кнопку мыши, выберите из меню Add Existing Files To Group ‘Source Group 1‘, укажите тип файла – ассемблерный, укажите PR_2.s, нажмите Add. Удалите старый файл PR_1 из списка. Откройте вновь созданный файл PR_2 для редактирования и ввода новой программы.

3) Настройте конфигурацию системы отладки Keil µVision. Окно для настройки параметров *Option for Target ‘Target 1’* открывается из меню Project, содержит закладки для установки параметров отладки программы.

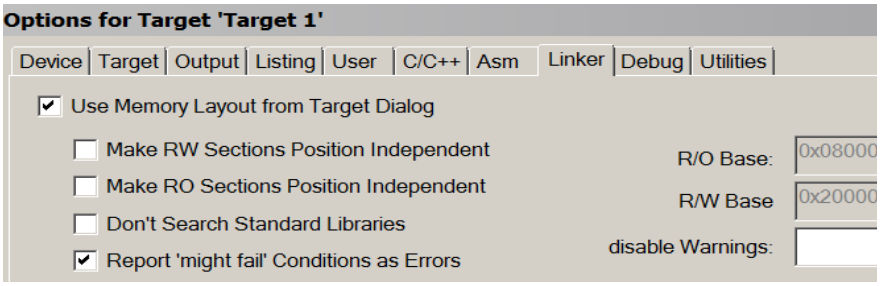


Рис. 1.2. Настройка конфигурации микроконтроллера

Device. Должно быть выбрано устройство *MDR1986BE92*

Target. В строке System Viewer File укажите путь к файлу системного отладчика, например, C:\ProgramFiles\Keil_v5\ARM\SFD\ MDR32F9Qx.SFR.

Asm - снимите галочку в строке Thumb Mode, если она установлена.

Linker. установите галочку в строке *Use Memory Layout from Target Dialog*;.

Debuger - Укажите Use Simulator, а также путь к файлу memoryinit.ini. Нажимая Edit, убедитесь в доступности этого файла в системе.

Закладки Output, Listing, User, C/C++ не требуют коррекции.

4) Введите программу.

5) Выполните компиляцию, нажимая F7, убедитесь в отсутствии ошибок (рис. 2.1). Наличие предупреждений не препятствует продолжению работы

6) Запустите компиляцию (F7) и отладку программы (Ctrl-F5) в режиме *Simulator*. Настройте режим отладки. Откройте закладку Registers менеджера проекта (рис. 4.4). Нажмите не квадратик со знаком плюс, расположенный в начале строки

xPSR для отображения признаков (*N, Z, C, V*) на экране. Из меню *View/Memory Windows* откройте два окна памяти. Для каждого окна из меню, вызываемого по правой кнопке можно установить тип отображаемых данных, например, Unsigned Char.

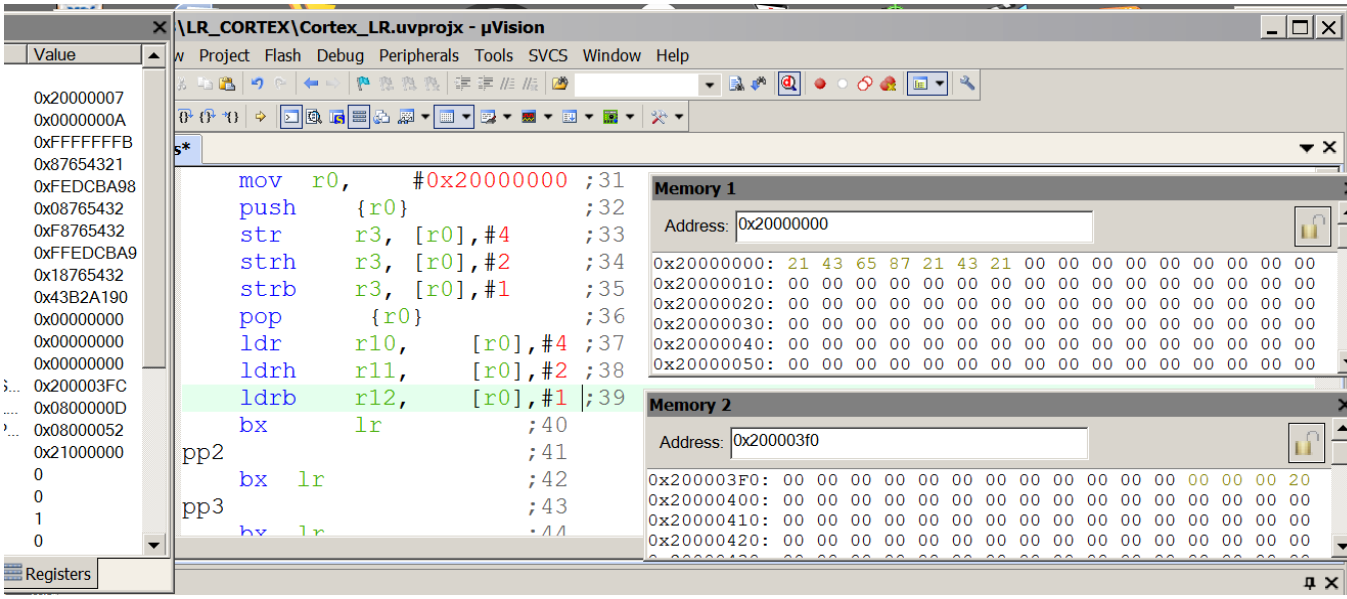


Рис. 1.3. Отладка программы

Изменяя размер окна, можно выбрать количество отображаемых столбцов, например, 16. Адрес окна *Memory1* установлен на начало ОЗУ - 0x20000000, а окна *Memory2* – на область, принадлежащую стеку- 0x200003e0. Клавиша F11 позволяет выполнить программу по шагам и проверить результаты выполнения команд.

Задание 1.1. Изучите программу *PR_1*. Выполните запуск подпрограмм pp1 и pp2/

Задание 1.2. Выполните программу PR_1 по шагам, нажимая клавишу F11. Представьте результаты выполнения подпрограмм в виде таблиц, содержащих номер строки, команду, признаки результата, выполняемое действие и результат – модифицируемый параметр – содержимое регистра или ячейки памяти. В таблице ниже показаны примеры для некоторых строк.

№	Команда	NZCV	Действие	Результат
16	movs r0, #0	0 1 0 0	r0:=0	r0=0
26	movs r5, r3, lsr#4	0 0 0 0	r5:=r3/4	r5=0x08765432
32	push {r0}	0 0 0 0	M(SP):=r0 Sp:=Sp-4	M(200003ff)=00 00 00 20 Sp = 200003fc
33	str r3, [r0], #4	0 0 0 0	M(r0):=r3; r0:=r0+4	M(20000000)=21 43 65 87 r0=20000004

Символы «:=» означают присваивание; запись M(20000000) означает элемент памяти, младший байт которого имеет адрес 20000000; M(SP) означает содержимое памяти по адресам указателя стека.

Задание 1.3. Заданы числа A,B,C,D в десятичной системе счисления.:

Бригада	1	2	3	4	5	6	7	8	9	10	11	12
A	160	212	30	45	570	65	85	99	110	130	125	725

B	125	30	210	780	100	300	65	301	500	360	225	400
C	360	270	221	45	300	200	400	80	180	80	450	55
D	500	770	450	21	450	100	200	220	720	20	80	35

Составьте подпрограмму pp2, которая выполнит запись заданных чисел в ячейки памяти с адресами 0x20000010, 0x20000014, 0x20000018, 0x2000001c. Для чисел, заданных в десятичной системе счисления приведите 16-ричные значения.

Также запишите команды для сохранения заданных чисел в стеке и извлчения их из стека в заданные регистры (в соответствии с ваоиантом).

Бригада	1	2	3	4	5	6	7	8	9	10	11	12
A	r0	r1	r5	r9	r5	r3	r5	r4	r6	r9	r9	r9
B	r1	r2	r3	r9	r6	r4	r3	r5	r7	r6	r7	r8
C	r2	r3	r2	r8	r7	r5	r2	r3	r8	r8	r8	r7
D	r3	r4	r7	r7	r8	r6	r1	r2	r9	r7	r6	r6

Задание 1.4. Составьте подпрограмму pp3, содержащую все возможные форматы заданных команд.

Бригада	1	2	3	4	5	6	7	8	9	10	11	12
lsr	rsr	asr	ror	rrx	lsr	rsr	asr	lsr	ror	rsr	asr	rrx
rrx	as	ror	rrx	lsr	rsr	rrx	rsr	ror	lsr	lsr	lsr	asr

Контрольные вопросы

1. Перечислите элементы архитектуры, функциональный состав и основные параметры микроконтроллеров Cortex-M3 фирмы Миландр.
2. Как определяется максимальный размер адресного пространства МК?
3. Какие функции выполняют регистры общего назначения и системные регистры - указатель стека SP, регистр связи LR, счетчик команд PC?
4. С какой целью для связи блока РОН с устройствами памяти используют несколько шин. Какие свойства процессора зависят от количества шин связи?
5. Сколько шин используется для обмена данными между АЛУ и РОН? Какие свойства процессора зависят от этого количества?
6. Формат строки ассемблерной программы и его особенности.
7. Какие команды выполняют обработку данных в МК? Запишите форматы.
8. Запишите форматы команд обращения к памяти. Поясните обозначения.
9. Назначение и особенности применения суффиксов в форматах команд.
10. Перечислите признаки результата, используемые в МК Cortex-M3.
11. Какие значения может принимать суффикс *cond* в конкретных командах?
12. Типы сдвигов, применение операций сдвигов.
13. Назначение команд пересылки.
14. Приведите примеры команд с заданным методом адресации.
15. Базовый адрес и смещение в массиве данных.
16. Каким образом записать в память массив, элементы которого содержат заданное
17. количество байт – 1, 2, 4?
18. Какие команды обеспечивают использование подпрограмм?
19. Какие команды обеспечивают использование подпрограмм?
20. Приведите примеры командобращения к стеку.

Работа 2. Логические операции в микроконтроллерах *ARM Cortex-M3*

Цель работы: Получение начальных знаний об основных приемах обработки данных с применением условного выполнения команд в микроконтроллерах ARM Cortex-M3.

Программа *PR_2,* содержит примеры использования команд логических операций и рекомендации по их применению. Выполните отладку программы и анализ результатов.

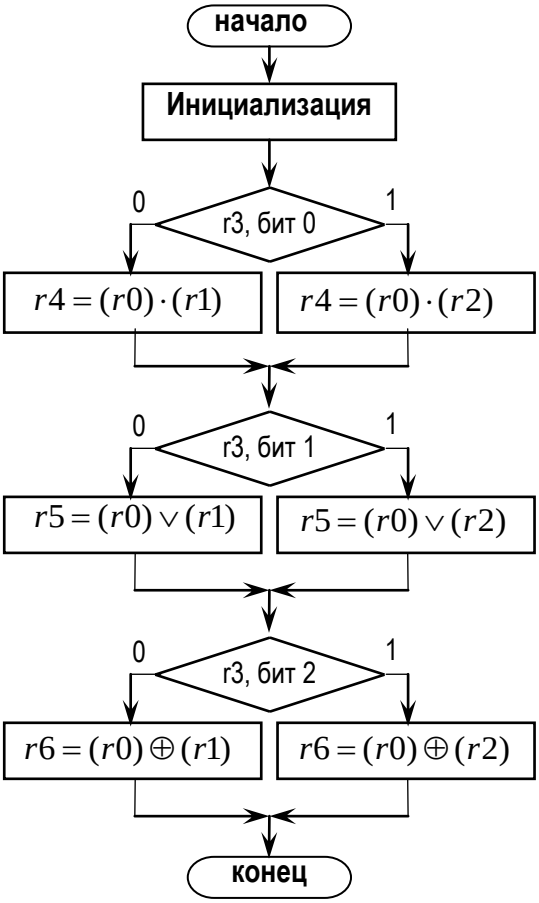


Рис. 2.1. Алгоритм условного выполнения операций с маской

Условное выполнение логических операций иллюстрируется в строках 16 - 28. Задан алгоритм (рис. 2.1), в соответствии с которым должны выполняться логические операции с данными и с одной из масок в зависимости от разрядов 2..0 управляющего кода. Бит 0 управляет выполнением операции И, если он равен 0, то используется маска 1, а если равен 1 – то маска 2. Аналогично биты 1 и 2 управляют выполнением операций ИЛИ и исключающее ИЛИ. Назначение регистров: r0 - данные, *r1* - маска 1, *r2* – маска 2, *r3* - управляющие логические сигналы, r4..r6 - результаты логических операций. Отдельные биты кода в регистре r3 выбирают вариант маски для определенных операций: бит 0 – для операции И; бит - для 1 ИЛИ; бит 2 – для операции исключающее ИЛИ.

Начало подпрограммы – инициализация. В строке 16 выполнена загрузка 32-разрядного слова данных в *r*0; в строках 17, 18 – загрузка масок 0x0000ffff и 0xffff0000 в регистры *r*1 и *r*2; в строке r19 – загрузка одного из вариантов управляющего кода в *r*3. В сроке 20 выполняется команда тестирования tst, которая выполняет побитовую операцию логического И между содержимым регистра r3 и константой 1 (маска на младший бит) и по результату устанавливает признаки *z* и *n*, но результат операции не сохраняется. Данная команда установит признак *z*=1, если младший бит в регистре *r*3 равен 0. В зависимости от признака *z* выполняется одна и команд, записанных в строках 21 и 22.

В сроке 23 выполняется команда тестирования *tst* с содержимым регистра r3 и константой 2 (маска на бит 1) и по результату выполняется одна из команд ИЛИ с маской (строки 24, 25). В строках 26 – 28 условное выполнение операции исключающее ИЛИ (eor) с маской.

Для анализа результатов выполнения логических операций с маской выбраны трехадресные команды, позволяющие получить результаты в отдельных регистрах и сохранить при этом исходные данные. Рекомендуется проверить работу программы, при других значениях управляющего кода, содержащегося в регистре *r*3.

Операция распаковки. В строках 29 – 35 выполняется выделение отдельных байт слова, содержащегося в регистре r0 и их запись в регистры *r*7, *r*8, r9, r10. Для выполнения этих операций используют операции И, ИЛИ и сдвиги.

```
1   ;       PR_2
2       area stack, noinit, readwrite
3       space 0x400
4   stack_top
5       area reset, data, readonly
6       dcd stack_top
7       dcd start
8       area program,code,readonly
9       entry
10  start
11  ;   bl      pp1
12      bl      pp2
13  ;   bl      pp3
14      b   start
15  pp1
16      ldr r0, =0x12345678
17      movw  r1, #0xffff
18      mov   r2, r1, ror #16
19      mov   r3, #0x05
20      tsts r3, #1
21      andne  r4, r0, r1
22      andeq  r4, r0, r2
23      tsts r3, #2
24      orrne  r5, r0, r1
25      orreq  r5, r0, r2
26      tsts r3, #4
27      eorne  r6, r0, r1
28      eoreq  r6, r0, r2
29      and     r7, r0,#0xff
30      and     r8, r0,#0xff00
31      lsr r8, r8, #8
32      and     r9, r0, #0xff0000
33      lsr  r9 ,r9, #16
34      and r10, r0, #0xff000000
35      lsr  r10,r10, #24
36      bx  lr
37  pp2
38      mov     r0, #0x20000000
39      mov r1, #0x22000000
40      mov r2, #0
41      mov r3, #1
42      str r2, [r0]
43      str r3, [r1]
44      str r3, [r1,#0x04]
45      str r3, [r1,#0x08]
46      str r3, [r1,#0x0c]
47      ldr r5, [r1,#0x08]
48      ldr r6, [r0]
49      mov r7, #0x0a
50      str r7, [r0,#0x03]
51      str r2, [r1,#0x64]
52      str r2, [r1,#0x6c]
53      ldr r8, [r0,#0x03]
54      bx  lr
55  pp3
56      bx  lr
57      end
```

Доступ к битам ОЗУ поясняет подпрограмма *pp2*. Строки 38 – 42 – инициализация. Назначение регистров: r0 – базовый адрес ОЗУ данных; r1 – базовый адрес области адресов доступа к битам (ОАДБ); *r2, r3* – константы 0 и 1, предназначенные для записи в отдельные биты. В строке 42 выполняется очистка байта в ОЗУ по адресу 20000000, в отдельные биты которого предполагается записывать единицы.

Строки 43 – 46 - установка в 1 отдельных бит младшего байта ОЗУ. При отладке результат отображается в окне *Memory*. Строка 47 – пример чтения в r5 бита 2 младшего байта ОЗУ. Получим 1 в младшем разряде. Строка 48 – чтение младшего байта ОЗУ полностью (для сравнения) в *r6*. В строках 49 – 53 в ячейку с адресом 0x20000003 вначале записывается константа 0x0a, а затем отдельные биты этой ячейки сбрасываются в 0 .

Запустите компиляцию (F7) и отладку программы (Ctrl-F5). Настройте режим отладки. Откройте закладку *Registers* менеджера проекта (рис. 4.4). Нажмите не квадратик со знаком плюс, расположенный в начале строки *xPSR* для отображения признаков (*N, Z, C, V*) на экране. Из меню *View/Memory Windows* откройте два окна памяти. Для каждого окна из меню, вызываемого по правой кнопке можно установить тип отображаемых данных, например, *Unsigned Char*. Изменяя размер окна, целесообразно выбрать количество отображаемых столбцов, например, 16. Адрес окна *Memory1* установлен на начало ОЗУ - 0x20000000, а окна *Memory2* – на область, принадлежащую стеку- 0x200003e0.

В строках 55 – 57 команды для начала и завершения подпрограммы, используемой при самостоятельной работе.

Задание 2.1. Выполните программу PR_2 по шагам, нажимая клавишу F11. Представьте результат выполнения подпрограммы в виде таблиц, содержащих номер строки, команду, признаки результата, выполняемое действие и результат – модифицируемый параметр – содержимое регистра или ячейки памяти. В таблице ниже показаны примеры для некоторых строк.

Задание 2.2. Составьте подпрограмму , для исследования результатов выполнения всех возможных операций сдвига для заданных чисел A,B,C,D: Представьте результаты выполнения подпрограмм в виде таблицы.

Бригада	1	2	3	4	5	6	7	8	9	10	11	12
A	160	212	30	-45	570	65	85	99	-110	130	125	725
B	-125	-30	-210	780	-100	300	65	301	500	360	225	-400
C	360	270	221	45	300	200	400	-80	180	80	450	55
D	500	770	450	21	450	-100	-200	220	720	-20	-80	35

Задание 2.3. Составьте подпрограмму , для вычисления:

1) алгебраической суммы чисел A,B,C,D;

2) суммы по модулю 2 чисел A,B,C,D.

Представьте результат выполнения подпрограммы в виде таблицы.

Задание 2.4. Составьте подпрограмму, содержащую упаковку, при которой 32-разрядное слово в регистре r12 составляется из 8-разрядных слов, содержащихся в младших разрядах заданных регистров.

(r0) = 0x05; (r1) = 0x16; (r2) = 0x27; (r3) = 0x38

r12	1	2	3	4	5	6	7	8	9	10	11	12
Байт 0	r0	r1	R2	r3	r3	r2	r2	r3	r3	2	r1	3
Байт 1	r1	r2	r3	r0	r2	r3	r3	r1	r2	0	r3	1
Байт 2	r2	r3	r0	r1	r1	r1	r0	r2	r0	1	r0	0
Байт 3	r3	r0	r1	r2	r0	r0	r1	r0	r1	3	r2	2

Задание 2.5. Составьте программу, содержащую команды, формирующие признаки C и Z.

Задание 2.6. Составьте программу, содержащую команды умножения и деления на числа, являющиеся степенью числа 2.

Задание 2.7. Составьте программу, содержащую примеры использования логических операций для формирования признаков результата.

Контрольные вопросы

1. Опишите использование логических операций И, ИЛИ, исключающее ИЛИ для формирования признаков.
2. Использование операции тестирования для формирования признаков.
3. Типы сдвигов, применение команд сдвигов.
4. Формирование признаков результата командами различных типов.
5. Составьте программу, формирующую признак, заданный преподавателем.
6. Составьте программу, иллюстрирующую упаковку и распаковку данных.
7. Поясните способ обращения к битам.
8. Поясните выполнение команд изменения порядка полей в слове.
9. Составьте программу, содержащую условно выполняемые команды.
10. Изобразите алгоритм с структурой «Если – то – иначе», напишите программу.
11. Перечислите признаки результата, используемые в МК Cortex.
12. Что такое условное выполнение команды и как оно используется?
13. Поясните работу команд перестановки полей в слове.

Работа 3. Арифметические операции в микроконтроллерах *ARM Cortex-M3*

Подпрограмма pp3 содержит примеры использования команд арифметических операций над данными различной разрядности, представленных в виде чисел без знака и со знаком..

Введите программу PR_3, запустите компиляцию (F7) и отладку программы (Ctrl-F5). Откройте закладку Registers менеджера проекта. Из меню *View/Memory Windows* откройте два окна памяти, для отображения содержимого одних и тех же ячеек памяти с различным типом данных. Установите одинаковые адреса в этих окнах памяти, соответствующие началу ОЗУ: 0x20000000. Установите тип отображаемых данных, используя меню, вызываемое правой кнопкой, для окна *Memory1 - Unsigned Char*, а для окна *Memory2 – Signed Char, Decimal*. Такая настройка позволит для одних и тех же ячеек анализировать двоичные коды и соответствующие им десятичные числа со знаком при отладке подпрограммы *pp1*. Признака начала комментария (;) в строке 11 не должно быть.

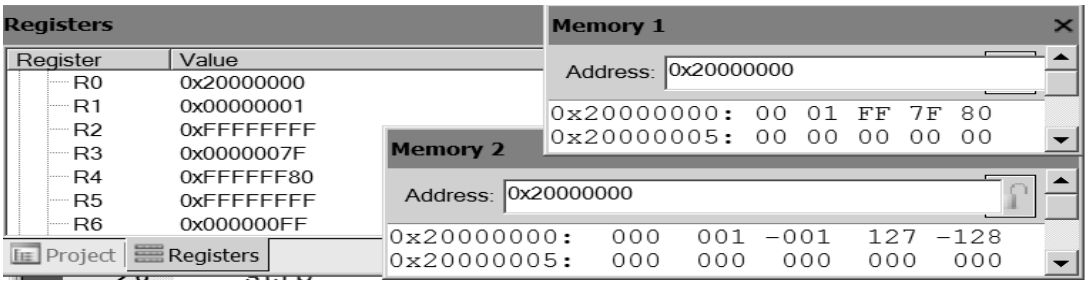


Рис. 5.3. Отладка подпрограммы программы *pp1* из PR_5

Регистр r0 будет использоваться как указатель адреса, в строке 16 в этот регистр записан адрес начала ОЗУ. В регистры *r1 – r4* записаны константы, соответствующие разрядности 1 байт (строки 17 - 20). Анализ содержимого регистров *r1 - r4* иллюстрирует представление чисел со знаком в виде слов. В программах, содержащих арифметические операции, для ввода данных и анализа результатов при отладке целесообразно использовать десятичную систему счисления.

В строках 21 – 25 содержатся команды записи байтов данных из регистров в память, при этом в память записываются младшие байты слов, а остальные байты теряются. В строке 26 записана команда чтения байта данных со знаком в регистр r5, в результате в регистре имеем дополнительный код числа -1, в котором старшие разряды заполнены единицами. Для сравнения строке 27 записана команда чтения того же байта данных регистр *r6* как числа без знака, при выполнении которой старшие разряды слова заполняются нулями. Результаты отладки подпрограммы *pp1* поясняет рисунок 5.3.

```
1   ;       PR_3
2       area stack, noinit, readwrite
3       space 0x400
4       stack_top
5       area  reset,  data,  readonly
6       area program,code,readonly
7       dcd stack_top
8       dcd start
9       entry
10  start
11      bl  pp1
12  ;   bl  pp2
13  ;   bl  pp3
14      b   start
15  pp1
16      mov r0, #0x20000000
17      mov r1,    #1
18      mov r2,    #-1
19      mov r3,    #127
20      mov r4,    #-128
21      strb    r0, [r0]
22      strb    r1, [r0,#1]
23      strb    r2, [r0,#2]
24      strb    r3, [r0,#3]
25      strb    r4, [r0,#4]
26      ldrsb   r5, [r0,#2]
27      ldrb    r6, [r0,#2]
28      bx  lr
```

```
29  pp2
30      ldr r8, = -2
31      ldr r9, = 3
32      ldr r10, =-4
33      ldr r11, =5
34      smull r0, r1, r8, r9
35      smull r2, r3, r10, r11
36      adds    r4, r0, r2
37      adcs    r5,  r1, r3
38      rrx     r5, r5
39      rrx     r4, r4
40      bx  lr
41  pp3
42      mov r0, #0x20000000
43      mov r1, #50
44      mov r2, #-100
45      mov r3, #-250
46      mov r4, #100
47      add r5, r1, r2
48      add r5, r3
48      add r5, r4
50      asr r6, r5, #2
51      strb r6, [r0]
52      ldrsb r7, [r0]
53      ldrb r8, [r0]
54      bx  lr
55      end
```

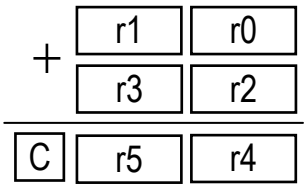
В подпрограмме *pp2* приведен пример программы, в которой при использовании команд учитываются биты переноса. При отладке *pp2* не должно быть признака начала комментария (;) в строке 12.

Задано вычислить сумму двух произведений 32-разрядных чисел со знаком по формуле: y = (a*b + c*d), а также среднее значение z = y/2.

При назначении регистров необходимо учитывать, что для результата команд умножения необходимо выбирать регистры *r0 – r7*, поэтому для сомножителей *a, b, c,*

d выбираем регистры *r8, r9, r10, r11*; для произведения *a*b* - регистры *r1*:*r0*; для произведения *c*d* - регистры *r3*:*r4*; для суммы y и среднее значения *z* - регистры *r5*:*r4*.

В строках 30 – 33 в регистры r8 – r11 вводятся исходные данные. В строках 34, 35 вычисляются произведения в виде 64-разрядных чисел со знаком. Каждое произведение содержит два слова, старшее и младшее. В строке 36 выполняется суммирование младших слов, и устанавливаются признаки. В данном случае появляется признак c = 1, который учитывается при суммировании старших слов (строка 37),в этой операции также устанавливается c = 1. После выполнения строки 37 в регистрах *r5*:*r4* получим сумму произведений. Для деления суммы на 2 используются команды сдвига через перенос, вначале сдвигается старшее слово, затем младшее (строки 38, 39). Команда сдвига вправо через перенос *rrx* выполняет деление числа на 2 и при этом записывает значение признака «*c*» в старший разряд регистра, а в качестве нового значения признака с записывает младший бит слова. Приведенные команды вычисляют полусумму при максимально возможной разрядности слагаемых (до 32 разрядов), при этом допускается возникновение переноса.



В подпрограмме *pp3* вычисляется среднее значение чисел, содержащихся в регистрах *r1, r2, r3, r4* и запись результата в ячейку памяти с адресом 0x20000000.. Для накопления суммы чисел используется регистр *r6*, а для вычисления среднего значения – регистр r6. После записи в память выполняется проверка результата – чтение содержимого ячейки памяти в *r7* как числа со знаком, а также чтение в *r8* как беззнакового числа.

Задание 3.1. Выполните программу PR_3 по шагам, нажимая клавишу F11. Представьте результат выполнения подпрограммы в виде таблиц, содержащих номер строки, команду, признаки результата, выполняемое действие и результат – модифицируемый параметр – содержимое регистра или ячейки памяти. В таблице ниже показаны примеры для некоторых строк.

Задание 3.2. Составьте подпрограмму для исследования результатов выполнения операций умножения с суммированием. Рассчитайте теоретические значения, представьте результат выполнения подпрограммы в виде таблицы.

Задание 3.3. Составьте подпрограмму для исследования результатов выполнения операций умножения с накоплением. Рассчитайте теоретические значения, представьте результат выполнения подпрограммы в виде таблицы.

Задание 3.4. Составьте программу, содержащую примеры выполнения арифметических операций над числами со знаком и без знака.

Контрольные вопросы

1. Перечислите форматы данных, используемые в микроконтроллерах *Cortex*.
2. Укажите границы диапазонов чисел без знака и со знаком, представленных: 8-разрядными, 16-разрядными кодами, 32-разрядными кодами.
3. Перечислите команды, при выполнении которых учитывается перенос *C*.
4. Как определить разрядность суммы при заданной разрядности слагаемых?
5. Как определить разрядность произведения при заданной разрядности сомножителей, заданных числами без знака и со знаком?
6. Как выполняется умножение на числа, являющиеся степенью числа 2?

7. Как выполняется деление на числа, являющиеся степенью числа 2?
8. Поясните назначение операции суммирования с учетом переноса.
9. Поясните использование в pp1 операции сдвига через перенос.
10. Поясните назначение команд обращения к памяти всех возможных форматов.

Работа 4. Циклические программы в микроконтроллерах ARM Cortex-M3

Программа *PR_4,* содержащую примеры использования циклических программ.

Подпрограмма pp1 содержит пример циклической программы со счетчиком для формирования двух массивов, каждый из которых содержит 16 элементов разрядностью 8 бит, вычисленных по заданным формулам.

Массив 1. Xk = 5*k+7. Базовый адрес 20000000.

Массив 2. Yk = $(k-3)^2$. Базовый адрес 20000010.

Массив 1 – арифметическая прогрессия, для которой первый элемент равен 7, а разность равна 5. Каждый элемент (начиная со второго) равен сумме предыдущего элемента и разности. Массив 2 – геометрическая прогрессия, для вычисления элемента необходимо использовать индекс. Для вычисления элементов массивов могут быть использованы отдельные циклы для каждого массива (при этом упрощается отладка) или общий цикл (короче программа). Выбран вариант с общим циклом. **Выбор регистров для размещения данных**.

r0, r1 – косвенные адреса массивов;

r2 – счетчик повторений цикла;

r3 – номер элемента в массиве (индекс k);

r4, r6 – регистры для текущих значений Xk, Yk.

r7 - r9 – вспомогательные регистры.

Начало подпрограммы – заголовок, инициализация процесса вычислений. В регистры r0, r1 записывается базовые адреса массивов. В регистр *r2* записывается заданное количество повторений цикла, в *r3* – исходное значение индекса k. В регистры *r4, r5* ,записываются значения элементов x(0), y7(0).

Начало цикла. Запись в память значений элементов, полученных в предыдущем цикле (или в заголовке), посредством команд, использующих косвенную адресацию с пост – индексированием. К мнемонике команды добавлен суффикс *b*, означающий запись байта. После выполнения команды адрес увеличивается на 1 и записывается в указатель (строки 22, 23). Далее выполняется вычисление индекса текущего элемента (строка 24), вычисление текущего элемента первого массива, результат в *r4* (строка 25), вычисление текущего элемента второго массива, результат в *r5* (строки 26 - 28). Завершение тела цикла – декремент счетчика циклов, формирование признака одолжения повторений (строка 29) и переход на начало цикла.

В подпрограмме *pp2* содержится пример вычисления чисел Фибоначчи.

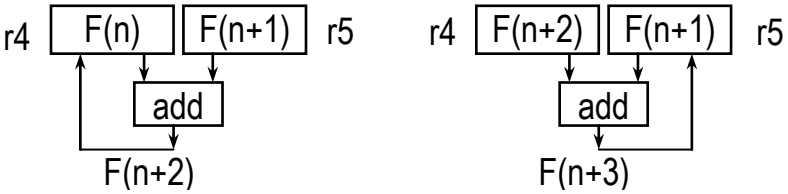


Рис. 6.3. Вычисление чисел Фибоначчи

Последовательности чисел Фибоначчи имеет первые два числа, равные 1, а каждое последующее равно сумме двух предыдущих. Последовательность

задается **рекуррентной формулой**, в которой *n*-ый член, при *n*>1 выражается через предыдущие: $a_0 = 1;\quad a_1 = 1;\quad a_n = a_{n-1} + a_{n-2}\quad для\quad n > 1.$

В память, начиная с адреса 0x20000000, записываются младшие байты чисел. Ограничение разрядности полученных слагаемых позволяет использовать числа массива как псевдослучайные.

Инициализация подпрограммы выполняется в строках 33 – 38, это запись начального адреса массива чисел в регистр *r1*, а также запись первых двух чисел последовательности, равных единице в регистры *r4, r5* и в первые две ячейки массива. Для записи чисел в память использованы команды записи байт с копенной адресацией с пост-индексированием, в которых после записи числа в память адрес в указателе увеличивается на 1 и сохраняется как новое значение.

Вычисление последующих чисел последовательности выполняется в цикле, с получением результата попеременно в регистрах *r4*, или *r5* (строки 39 - 43), как показано на рис. 6.3. Результат суммирования записывается на место слагаемого, которое в дальнейшем не потребуется.

Числа, получаемые в регистрах, непрерывно возрастают до появления переноса из старшего разряда регистра, но при ограничении разрядности закон изменения чисел становится псевдослучайным.

Числа, записанные в массив, могут быть использованы впоследствии как 8-разрядные целые без знака, или со знаком, что показано на рис. 5.3, где открыто два окна *Memory* (рис. 5.3) и из меню, открывающегося по правой кнопке, установлены различные параметры для одного и того же массива.

Для *Memory1* выбраны параметры *Decimal, Unsigned, Char,* а для *Memory2* - *Decimal, Signed, Char*.

Подпрограмма pp3 содержит пример пользования циклической программы для вычисления заданных параметров данных, содержащихся в массиве. Для массива данных, полученного в *pp2*, или в *pp1*, имеющего 32 элемента разрядностью 1 байт, определить:

- количество элементов n1, удовлетворяющих условию X≤X1;
- количество элементов n2, удовлетворяющих условию X≤X2;
- количество элементов n3, удовлетворяющих условию X1<X≤X2. Записать значения *n1, n2, n3* в ячейки памяти с адресами

В подпрограмме принято следующее назначение регистров: *r0* – базовый адрес массива; *r1* – счетчик количества повторений цикла; *r2* – *r4* – регистры для размещения чисел *n1, n2, n3*.

Начало подпрограммы *pp3* – инициализация (строки 47 - 60). В цикле (строки 51 - 57) выполняется подсчет чисел n1 и n2, а после выхода из цикла определяется n3 (строка 58) и записываются числа *n1, n2, n3* в ячейки с заданными адресами (строки 69 - 62). Заметим, при отладке подпрограммы pp3 необходимо включить одну из подпрограмм - *pp1*, или *pp2*.

```
1        PR_4
2        area stack, noinit, readwrite
3        space  0x400
4  stack_top
5        area reset, data, readonly
```

```
33 pp2
34     ldr r1, = 0x20000020
35     mov   r2, #16
36     mov   r4, #1
37     mov   r5, #1
```

```
6         area program,code,readonly
7         dcd stack_top
8         dcd start
9         entry
10  start
11        bl       pp1
12  ;     bl       pp2
13  ;     bl       pp3
14  m     b        m
15  pp1
16        mov      r0, #0x20000000
17        ldr  r1, = 0x20000010
18        mov      r2, #16
19        mov      r3, #0
20        mov      r4, #7
21        mov      r5, #9
22  m1
23         strb    r4, [r0],    #1
24        strb  r5,[r1],    #1
25        add r3, #1
26        add r4, #5
27        mov      r5, r3
28        add r5, #-3
29        mul r5, r5
30        subs     r2, #1
31        bne m1
32        bx   lr
```

```
38      strb     r4, [r1],    #1
39      strb     r5, [r1],    #1
40  m2  add      r4, r5
41      strb     r4, [r1],    #1
42      add      r5, r4
43      strb     r5, [r1],#1
44      subs     r2, #2
45      bpl m2
46      bx  lr
47  pp3
48      mov      r0, #0x20000000
49      mov      r1, #32
50      mov      r2, #0
51      mov      r3, #0
52  m3  ldrb     r5, [r0],#1
53      cmps     r5, #20
54      addls    r2, #1
55      cmps     r5, #40
56      addls    r3, #1
57      subs     r1, #1
58      bpl m3
59      sub      r4, r3, r2
60      strb     r2, [r0],#1
61      strb     r3, [r0],#1
62      strb     r4, [r0],#1
63      bx  lr
64      end
```

Задание 4.1. Выполните программу PR_4 по шагам, представьте результат выполнения подпрограммы в виде таблиц, содержащих теоретические значения.

Задание 4.2. Разработайте программу, выполняющую заданные действия.

1) Формирование массива Y(n), содержащего 16 элементов, разрядностью 1 байт, определяемых по формуле в соответствии с вариантом.

Вариант	Зависимость	Вариант	Зависимость
1	$2*n^2+2*n-1$	7	$2*n^3+4*n+1$
2	$3*n^3+12*n+1$	8	$10*n^2+4*n+5$
3	$5*n^2+3*n+2$	9	$n^2+8*n-10$
4	$2*n^3+n+1$	10	$4*n^3+8*n+1$
5	$3*n^3+5*n+15$	11	$3*n^2+12*n+12$
6	$4*n^2+5*n-3$	12	$2*n^3+2*n+20$

2) Определение среднего значения элементов в массиве.

3) Вычисления количества элементов, принадлежащих заданным диапазонам:

Вариант	Диапазон
1-3	0x00 – 0x3f
4-6	0x40 – 0x7f
7-9	0x80 – 0xbf

10-12	b0xc0 – 0xff

Приведите текст программы и таблицу результатов выполнения команд.

Контрольные вопросы

1. Перечислите форматы и границы диапазонов чисел, используемые в микроконтроллерах *Cortex*.
2. Перечислите команды, при выполнении которых модифицируются признаки.
3. Как выполняется умножение на числа, являющиеся степенью числа 2?
4. Как выполняется деление на числа, являющиеся степенью числа 2?
5. Укажите диапазон адресов физически реализованного в *Cortex* ОЗУ данных.
6. Запишите формат команды с косвенной адресацией.
7. Запишите формат команды с косвенной адресацией со смещением.
8. Запишите формат команды с косвенной адресацией с пост-
9. индексированием.
   Поясните структуру циклических программ.
   Какие параметры описывают массив данных?

Работа 5. Программирование параллельных портов МК Cortex-M3

Цель работы. Получение практических навыков программирования периферийных устройств микроконтроллера ARM Cortex-M3.

Программа PR_5 содержит .пример программирования контроллера тактовых частот и параллельных портов МК Cortex-M3

Периферийные устройства микроконтроллеров необходимо программировать путем записи в регистры выбранных устройств определенных управляющих кодов. Для включения в работу периферийных устройств микроконтроллеров Cortex-M3 необходимо на выбранные устройства подключить тактовые сигналы, формируемые контроллером тактовых частот. В исходном состоянии (по умолчанию) все периферийные устройства не работают, они отключены от системы синхронизации для уменьшения потребления энергии. Подключение определенных устройств выполняют биты регистра **управления тактированием периферийных блоков PER_CLOCK,** адрес которого 0x4002_001C.

При работе с периферийными устройствами необходимо подать тактовые сигналы на используемые устройства, а также на модуль питания ядра (батарейный домен) BKP и контроллер тактовых частот RST_CLK. Для включения в работу параллельных портов необходимо учитывать подключение устройств к выводам микроконтроллера на отладочной плате. В соответствии со схемой подключения кнопок и светодиодов (рис. 4.1) необходимо подать тактовые частоты на порты C, B, E. Для указанных устройств необходимо записать «1» в соответствующие разряды регистра. Программирование регистра PER_CLOCK выполненов строках 15 - 17.

Параллельные порты обеспечивают обмен данными МК и внешнего устройства параллельными кодами. Микроконтроллеры **Cortex** могут содержать 6 портов,

обозначаемых A...F, разрядности 16. В исследуемой модели порты A, B, C и E 8-разрядные.

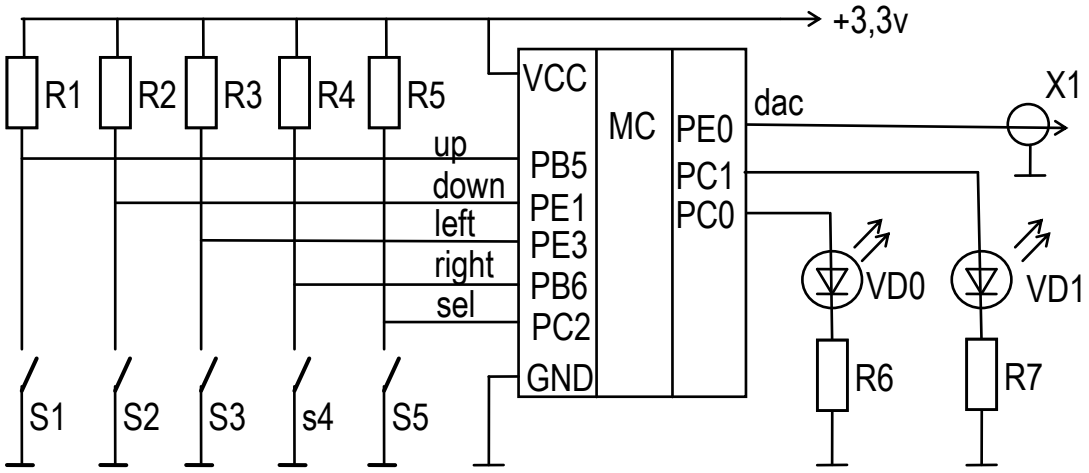


Рис.5.1 Схема подключения кнопок и светодиодов к МК

Для программирования каждого порта используется 8 регистров. Адрес регистра представляют в виде суммы базового адреса порта и смещения, задающего определенный регистр. Для программирования регистров порта используется косвенная адресация со смещением.

1) **Регистр данных порта RXTX**[15:0], смещение 0x00, предназначен для хранения данных при вводе и при выводе. Его адрес совпадает с базовым адресом порта.

2) **Регистр направления передачи данных OE**[15:0], смещение 0x04. Каждый разряд порта программируется независимо от остальных, сигнал 1 устанавливает режим вывода, а 0 - ввода.

3) **Функция разряда порта FUNC**[31:0], смещение 0x08. Каждый разряд (контакт или вывод) порта можно использовать в нескольких режимах. Режим одного бита 8-разрядного порта задают 2 бита 16-разрядного регистра: 00 – порт (выбирается по умолчанию); 01 – основная функция; 10 – альтернативная функция.

4) **Аналоговый режим работы ANALOG**[15:0]. Смещение 0x0C. 0 - аналоговый режим (выбирается по умолчанию); 1 - цифровой – 1. В строке 21 для порта A задан цифровой режим работы.

5) **Регистр включения подтягивающих резисторов PULL**[31:0]. Смещение 0x10. Используется, если в схеме не предусмотрены подтягивающие резисторы при подключении контактных датчиков. Два 16-разрядных регистра. Разряды 31..16 - Pull up – подтяжка к питанию, разряды 15..0 - Pull down – подтяжка к земле: 0 – выключена (выбирается по умолчанию); 1 – включен резистор R = 50 кОм.

6) **Режим работы выходного драйвера PD**[31:0]. Смещение0x14. Два 16-разрядных регистра. Разряды 31..16 подключают триггер Шмидта в устройстве ввода. По умолчанию - 0 – триггер Шмидта выключен гистерезис 200 мВ; 1 – триггер Шмидта включен гистерезис 400 мВ.

7) **Режим мощности передатчика PWR**[31:0]. Смещение0x18. Режим одного бита 8-разрядного порта задают 2 бита 16-разрядного регистра: 00 – зарезервировано (передатчик отключен) 01 – медленный фронт (порядка 100 нс) строки 18,19; 10 – быстрый фронт (порядка 20 нс) 11 – максимально быстрый фронт (порядка 10 нс)

8) **Режим работы входного фильтра GFEN**[15:0]. Смещение0x1C. 0 – фильтр выключен; 1 – фильтр включен (фильтрация импульсов до 10 нс).

Строки 18 - 23 – инициализация порта A, а строки 24 - 30 – инициализация порта E, используемого для ввода цифровых сигналов от кнопок и для вывода аналогового сигнала с выхода ЦАП. Строки 31 – 33 – инициализация ЦАП. Выход ЦАП подключен к выводу порта E0 и к гнезду DAC на отладочной плате.

Формирование и вывод заданного сигнала выполняет основная программа (строки 11 - 14), представляющая собой бесконечный цикл, в котором вызываются подпрограммы.

В подпрограмме pp2 (строки 35 - 43) формируются линейно-изменяющиеся коды, в строке 38 выполняется чтение порта E, к разрядам 1 и 3 которого подключены кнопки (см. таблицу 4 в раздаточном материале). Операция И с маской (строка 39) выделяет разряд E1 и при результате равном нулю формирует признак z=1, который используется для условного вывода сигналов.

Сложные функции в микроконтроллерах получают, используя таблицы в ПЗУ команд. Таблица должна начинаться с метки. Для записи массивов в ПЗУ предназначены следующие директивы: dcb – объявление байт, dcw – объявление полуслов, dcd – объявление слов. После директивы перечисляются данные, разделенные запятой. Загрузку в регистр rd адреса, соответствующего метке в программе, выполняет команда: ldr rd, = метка.

В подпрограмме используется таблица функции: $N(n) = 1000 \cdot \sin\frac{360}{16}n + 1000$, где n – индекс (номер) элемента в массиве. Количество элементов – 16, разрядность – 16. Функция в таблице представлена в десятичной системе счисления, значения функции положительные, не превышают 2000. Для представления знакопеременной функции использован смещенный код, более удобный для преобразования сигнала в аналоговую форму, чем дополнительный.

```
1  ;        PR_5
2      area reset, data, readonly
3      area stack, noinit, readwrite
4      space 0x400
5  stack_top
6      area program,code,readonly
7      dcd stack_top
8      dcd start
9      entry
10 start
11             bl      pp1
12 cykle       bl      pp2
13 ;           bl      pp3
14             b       cykle
15 pp1    ldr      r0, =0x4002001c
16     ldr      r1, =0x0aa40010
17     str      r1, [r0]
18     ldr      r0, =0x400a8000
19     mov      r1, #0xff
29     mov      r2, #0x44
30     str      r2, [r1, #0x18]
31     ldr r2, =0x40090000
32     mov        r3, #0x08
33     str      r3, [r2]
34     bx       lr
35 pp2  add     r3, #1
36     sub      r4, #1
37     ldr r5, [r1]
38     ands     r6, r5, #0x02
39     streq    r3, [r2,#0x08]
40     streq    r3, [r0]   ;41
41     strne    r4, [r2,#0x08]
42     strne    r4, [r0]
43     bx       lr
44 pp3    ldr r7, =tab
45     add      r8, #0x02
46     and      r8, #0x1f
47     add      r9, r7, r8
```

```
    str     r1, [r0,#0x04]
    str     r1, [r0,#0x0c]
    movw    r1, #0x5555
    str     r1, [r0,#0x18]
    ldr r1, =0x400c8000
    mov     r2, #0x01
    str     r2, [r1, #0x04]
    mov         r2, #0x0a
    str     r2, [r1, #0x0c]
```

```
    ldrh    r10, [r9]
    str r10, [r0]
    str     r10, [r2,#0x08]
    bx      lr
tab dcw 1000,1382,1708,1924
    dcw 2000,1924,1708,1382
    dcw 1000,618,292,76
    dcw 0,76,292,618
    end
```

Начало подпрограммы pp3 - инициализация. В регистр r7 записывается базовый адрес массива, соответствующий метке «tab». Регистр r8 будет использован как счетчик, код которого будет увеличиваться с шагом 2 до максимального значения 0x1f.

Отладка программы в режиме симулятора.

При отладке программы целесообразно использовать системный вьювер, для работы которого необходимо указать путь гк файлу с расширением .SFR, который содержит конфигурацию данного микроконтроллера. Откройте окно Options for Target / Debug, выберите Use Simulator. На закладке Target в строке System Viewer File укажите файл системного отладчика C:\ Keil / ARM / SFD / MDR32F9Qx.sfr. Системный отладчик можно вызвать только в режиме отладки из меню: View/System Viewer/ Port/Port A. Для отображения другого устройства необходим новый вызов View/System Viewer/ Port/Port E.

После удачной компиляции запустите отладчик. В режиме отладки (Debug) из меню View / System Viewer / Port / MDR_PORTA откройте окно с отображением всех регистров порта A. Также откройте окно с отображением регистров порта E При нажатой клавише F11 код в регистре RXTX должен увеличиваться.

Настройка конфигурации микроконтроллера

Откройте окно Options for Target / Debug, выберите Use J-LINK / J-TRACE Cortex. Тип программатора выбирайте из списка. Нажмите кнопку Settings. В открывшемся окне Cortex JLink должны появиться параметры программатора и процессора. Откройте закладку Download, укажите файл с алгоритмом программирования.

Выполните компиляцию. Запустите программу отображения сигналов логического анализатора Saleae LLC / Logic. Наличие слова Connected в заголовке окна указывает на наличие подключения анализатора к компьютеру.

Установите частоту дискретизации 24 МГц, а интервал измерения 1 мкс из меню открывающегося кнопкой с изображением треугольников, которая расположена рядом с кнопкой Start. Масштаб временных диаграмм изменяют клавиши управления курсором, или колесико мыши.

При установке курсора на диаграмму выводятся параметры: duty – коэффициент заполнения; τ – период повторения; w – ширина (длительность) интервала времени, на который установлен курсор - импульса, или паузы; f – частота повторения импульсов.

Полученные диаграммы позволяют сделать вывод, что изменение кодов происходит с интервалом 1 мкс, при этом младший разряд кода изменяется от 0 к 1, и наоборот с частотой около 1 МГц. Полный период изменения младшего разряда содержит два значения выходного кода, различающиеся младшим разрядом.

Изменяя масштаб, и устанавливая курсоры A1 и A2 на начало и окончание периода изменения кода, можно получить частоту изменения сигнала старшего разряда, период f ≈ 4 кГц, что в 256 раз меньше частоты изменения младшего разряда.

Практическая часть

Задание 5.1. Для приведенной программы, используя системный отладчик, определите параметры формируемого сигнала – амплитуду и период. Определите адрес, соответствующий метке «tab». Приведите фрагмент содержимого ПЗУ, содержащий таблицу функции, опишите способ размещения данных в таблице.

З**адание 5.2.** Изучение подпрограммы инициализации pp1. Опишите способ выбора адресов регистров при инициализации устройств. Укажите буферные ячейки, содержимое которых будет использоваться в других подпрограммах.

Задание 5.3. Изучение подпрограммы pp2 – формирования и вывода линейно-изменяющихся кодов и ввода сигналов от кнопки.

Задание 5.4. Изучение подпрограммы формирования гармонического сигнала pp3.

Задание 5.5. Составьте подпрограмму, выполняющую вывод в порт A последовательность кодов, отображающих зависимость вида: $y = k \cdot \sin^2 x + m$. Подберите константы *k* и *m*, обеспечивающие отсутствие ограничения сигнала.

Задание 5.6. Составьте подпрограмму, обеспечивающую режим работы счетчика, при котором направление счета изменяется на противоположное после каждого нулевого состояния.

Контрольные вопросы

1. Как программируется контроллер тактовых частот для периферийных устройств?
2. Какую функцию выполняют подтягивающие резисторы?
3. Как программируется направление передачи данных?
4. В каких случаях используется аналоговый, или цифровой режим
5. В каких случаях используется двухтактный выход порта?
6. Какие команды используют при обращении к таблицам?
7. Структура циклических программ управления объектами.
8. Варианты формирования признаков выхода из цикла.
9. Поясните структуру циклических программ.
10. Какие параметры описывают массив данных?

Работа 6. Исследование фильтров

Теоретическая часть

Электрическим фильтром называется четырехполюсник, пропускающий без заметного ослабления колебания определенных частот, принадлежащих полосе пропускания, и подавляющий колебания других частот , принадлежащих полосе затухания.

Передача сигнала через фильтр определяется комплексным коэффициентом передачи (ККП), модуль которого равен отношению амплитуды выходного сигнала к

амплитуде входного, а фаза - фазовому сдвигу выходного сигнала относительно входного. Зависимость модуля м фазы ККП от частоты отображают амплитудно-частотная и фазочастотная характеристики (АЧХ и ФЧХ).

$$\dot{K}(j\omega)=\frac{u_{вых}}{u_{вх}}=\frac{U_{m\,вых}\cdot e^{j\varphi вых}\cdot e^{j\omega t}}{U_{m\,вх}\cdot e^{j\varphi вх}\cdot e^{j\omega t}}=\frac{U_{m\,вых}}{U_{m\,вх}}\cdot e^{j(\varphi вых-\varphi вх)}$$

Полосы частот пропускания и затухания на характеристиках разделяет частота среза АЧХ, или граничная частота ω_C, для которой модуль коэффициента передачи равен 0,707.

В зависимости от формы АЧХ различают следующие типы фильтров (рис. 6.1).

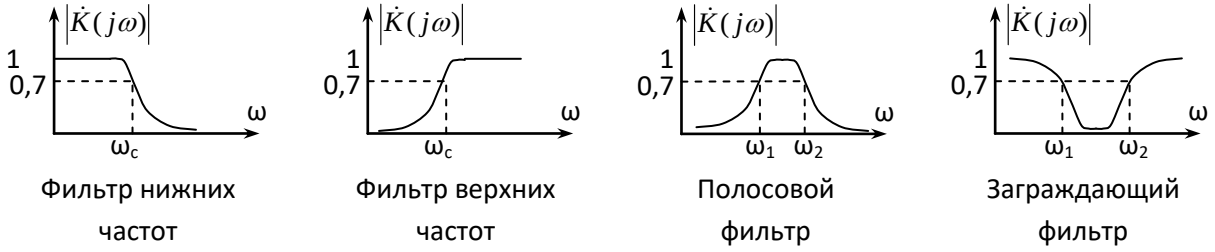


Рис. 6.1. Амплитудно-частотные характеристики фильтров

Фильтр нижних частот имеет полосу пропускания для колебаний, частота которых принимает значения от 0 (ω=0 - это постоянная составляющая, которая пропускается без ослабления) до частоты среза АЧХ ω_C.

Фильтр верхних частот имеет полосу пропускания для колебаний, частота которых выше граничной ω_C. Полоса пропускания полосовых фильтров лежит между двумя граничными частотами. Заграждающие фильтры, пропускающие в основном колебания всех частот, кроме определенной полосы затухания.

В зависимости от формы сигналов различают аналоговые и цифровые фильтры.

В аналоговых фильтрах обрабатываемые сигналы – напряжения. Пассивные аналоговые фильтры строятся на элементах R, L, C, а активные содержат дополнительно операционные усилители. Существуют аналоговые фильтры, использующие явления механического резонанса. Это фильтры электромеханические, пьезоэлектрические, кварцевые, фильтры на поверхностных акустических волнах.

В цифровых фильтрах обрабатываемые сигналы – двоичные коды. Современная элементная база позволяет реализовать задачи цифровой фильтрации несколькими способами, выбор которых определяют условия задачи.

Программная реализация с использованием персональных компбютеорв.

программная реализация на базе специализированной микропроцессиорной системы на кристалле, содержащей синтезированный процессор

Аппаратная реализация на ПЛИС, разработанная в виде проекта в САПР.

Анализ аналогового фильтра нижних частот первого порядка .

Комплексный коэффициент передачи $\dot{K}(j\omega)$ вычисляют по схеме ФНЧ (рис.6.2), учитывая, что резистор R и конденсатор с комплексным сопротивлением $1/j\omega C$ включенные последовательно, и образуют делитель напряжения, выходом которого является напряжение на конденсаторе. Модуль комплексного числа $\dot{K}(\omega)$ равен отношению модуля числителя (это 1) к модулю знаменателя, вычисляемого по теореме Пифагора.

$$\dot{K}(j\omega)=\frac{\frac{1}{j\cdot\omega\cdot C}}{R+\frac{1}{j\cdot\omega\cdot C}}=\frac{1}{1+j\cdot\omega\cdot R\cdot C}=\frac{1}{1+j\cdot\omega\cdot\tau};\quad K(\omega)=\frac{1}{\sqrt{1^2+(\omega\cdot\tau)^2}}=\frac{1}{\sqrt{1^2+(\frac{\omega}{\omega_C})^2}};\quad \omega_C=\frac{1}{\tau}.$$

Амплитудно-частотная характеристика (АЧХ), отображающая зависимость модуля от частоты показана на рис. 6.1. Произведение RC=τ имеет размерность времени, называется «постоянная времени фильтра». Частота $1/\tau = \omega_C$ является частотой среза АЧХ. Она является границей, разделяющей полосы пропускания и задерживания, если $\omega = 1/\tau$, то $K(\omega)=\frac{1}{\sqrt{2}}$.

При моделировании и экспериментах для вычисления модуля ККП в формулы подставляют частоту среза измеряемую в Герцах или период этой частоты в секундах.

$$;K(F)=\frac{1}{\sqrt{1^2+(\frac{F}{F_C})^2}};K(\mathrm{T})=\frac{1}{\sqrt{1^2+(\frac{T_C}{T})^2}};F_C=\frac{1}{2\cdot\pi\cdot\tau};T_C=2\cdot\pi\cdot\tau.$$

Пример. Для ФНЧ с параметрами R = 1 кОм, C = 1 мкФ получим:

$$\tau=\mathrm{R}\cdot\mathrm{C}=10^3\cdot10^{-6}=10^{-3}c.;\quad \omega_C=\frac{1}{\tau}=10^3\frac{1}{c};\quad F_C=\frac{1}{2\cdot\pi}\cdot\omega_C=159Гц;\quad \mathrm{T_c}=2\cdot\pi\cdot\tau=6{,}28\cdot10^{-3}c.$$

$$\text{При F}=159\ \text{Гц}\ K(F)=\frac{1}{\sqrt{2}};\ \text{при F}=1590\ \text{ГГ}\ K(\mathrm{F})=\frac{1}{\sqrt{1^2+(100)^2}}=\frac{1}{10};\ \text{при F}=15{,}9\ \text{Г}\quad K(\mathrm{F})=1.$$

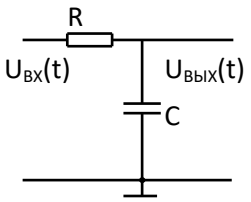


Рис. 6.2. Схема ФНЧ

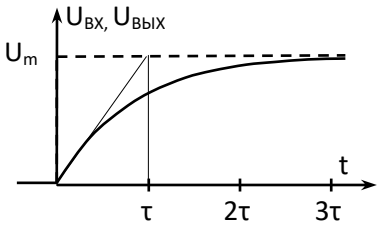


Рис. 6 .3. Переходная характеристика ФНЧ

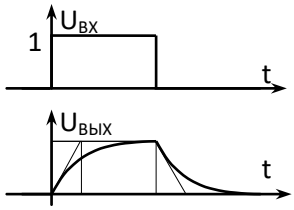


Рис. 6.4. Реакция ФНЧ на импульс

При передаче через фильтр импульсных сигналов возникают переходные процессы, которые описывает дифференциальное уравнение, составленное по схеме ФНЧ (рис. 6.2).

$$I_R=I_C;\qquad \frac{U_{ВХ}(t)-U_{ВЫХ}(t)}{R}=C\cdot\frac{dU_{ВЫХ}(t)}{dt};\qquad \frac{dU_{ВЫХ}(t)}{dt}=\frac{U_{ВХ}(t)-U_{ВЫХ}(t)}{RC}$$

В результате решения этого уравнения определяется переходная характеристика, которая отображает реакцию фильтра (вид выходного сигнала) при подаче на вход скачка напряжения с амплитудой Um $U_{ВЫХ}(t)=\mathrm{U_m}(1-e^{-t/RC})$ (рис. 6.3). Реакция на импульс определяется суммой реакций на скачки (рис. 6.4).

Выходное напряжение, снимаемое с конденсатора, не может измениться скачком, процесс его изменения сопровождается инерцией.

Отрезок, равный постоянной времени τ= RC определяет длину подкасательной, которая неизменна для любой точки экспоненты Uвых(t). Практически (с погрешностью 5 %) переходной процесс заканчивается за время, равное 3τ.

Постоянная составляющая сигнала и нижние частоты спектра передаются на выход фильтра без ослабления.

Дифференциальное уравнение показывает, что скорость изменения выходного напряжения зависит не только о входного напряжения, но и от выходного напряжения. Это означает, что в фильтре действует внутренняя обратная связь, при которой выходной сигнал влияет на вход. Аналоговые фильтры с обратной связью называют рекурсивными.

Теоретически переходные процессы заканчиваются в бесконечности, поэтому аналоговые фильтры имеют бесконечную импульсную характеристику, их называют БИХ - фильтры.

Анализ аналогового фильтра верхних частот первого порядка.

В схеме ФВЧ (рис. 6.5) конденсатор включен последовательно между входом и выходом. Он хорошо пропускает высокочастотные составляющие спектра и не пропускает нулевую гармонику.

По схеме, учитывая, что выходом делителя, построенного на элементах R и C, является напряжение на резисторе, можно записать выражение для комплексного

$$\dot{K}(j\omega) = \frac{R}{R + \frac{1}{j\cdot\omega\cdot C}} = \frac{1}{1 + \frac{1}{j\cdot\omega\cdot R\cdot C}} = \frac{1}{1 - \mathrm{j}\frac{1}{\omega\cdot\tau}}; \quad K(\omega) = \frac{1}{\sqrt{1^2 + (\frac{\omega_C}{\omega})^2}}$$

коэффициента передачи и его модуля:

Модуль комплексного числа $\dot{K}(j\omega)$ (с использованием теоремы Пифагора) равен отношению модуля числителя к модулю знаменателя. АЧХ для ФВЧ показана на рис. 6.1. Полученное выражение позволяет получить значения модуля K(ω) для различных ω. Если ω = 0, то K(ω) = 01; если ω = ∞, то K(ω)=1; если ω = 1π/τ, то $K(\omega) = \frac{1}{\sqrt{2}}$.

При измерении частоты в Герцах используют формулы, описывающие ФВЧ:

$$K(F) = \frac{1}{\sqrt{1^2 + (\frac{F_C}{F})^2}}; \quad K(\mathrm{T}) = \frac{1}{\sqrt{1^2 + (\frac{T}{T_C})^2}}; \quad F_C = \frac{1}{2\cdot\pi\cdot\tau}; \quad T_C = 2\cdot\pi\cdot\tau.$$

Пример. Для ФВЧ R = 1 кОм, C = 1 мкФ получим: Fc = 159 Гц.

$$\text{При } \mathrm{F} = 159 \text{ Гц } K(F) = \frac{1}{\sqrt{2}}; \text{ при } \mathrm{F} = 15{,}9 \text{ ГГ } K(\mathrm{F}) = \frac{1}{\sqrt{1^2 + (100)^2}} = \frac{1}{10}; \text{ при } \mathrm{F} = 1590 \text{ Г } \quad K(\mathrm{F}) = 1.$$

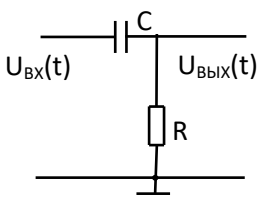


Рис. 6.5. Схема ФВЧ

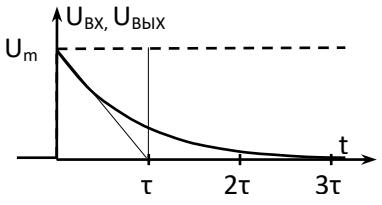


Рис. 6 .6. Переходная характеристика ФВЧ

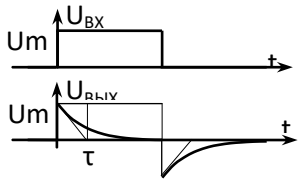


Рис. 6.7. Реакция ФВЧ на импульс

Для анализа передачи через ФВЧ импульсных сигналов используется дифференциальное уравнение, составленное по схеме ФВЧ (рис. 6.5).

$$I_R = I_C; \qquad C \cdot \frac{d[U_{ВЫХ}(t) - U_{ВХ}(t)]}{dt} = \frac{U_{ВЫХ}(t)}{R}; \qquad U_{ВЫХ}(t) = \mathrm{U_m} \cdot e^{-t/RC}$$

Решение дифференциального уравнения позволяете получить переходную характеристику Um $U_{ВЫХ}(t) = \mathrm{U_m} e^{-t/RC}$. (рис.6.6), описывающую реакцию на скачок напряжения. Реакция на импульс определяется как сумма реакций на скачки (рис. 6.7).

Напряжение на конденсаторе, который соединяет выход со входом, не может измениться скачком, процесс его изменения сопровождается инерцией. Однако, резкие изменения входного напряжения передаются на выход. Также передаются без ослабления верхние частоты спектра, для которых сопротивление конденсатора мало. ФВЧ не передает на выход нулевую гармонику (постоянную составляющую сигнала). Он выполняет функцию разделительной цепи. Теоретически переходные процессы заканчиваются в бесконечности, поэтому аналоговый ФВЧ имеет бесконечную импульсную характеристику, его называют рекурсивным БИХ - фильтром.

Цифровые рекурсивные фильтры с бесконечной импульсной характеристикой (БИХ – фильтры).

Анализ цифрового ФНЧ первого порядка.

При цифровой обработке сигналов используют аппаратные или программные фильтры в которых непрерывные функции времени Uвх(t) и Uвых(t) заменяются на последовательности дискретных отсчетов X(n) и Y(n)), выдаваемых в определенные моменты времени с шагом h и номером отсчета n - .

Разностное уравнение для цифрового ФНЧ можно записать , используя дифференциальное уравнение, описывающее аналоговый фильтр. При этом производная заменяется отношением конечных разностей. и выполняя замену переменных.

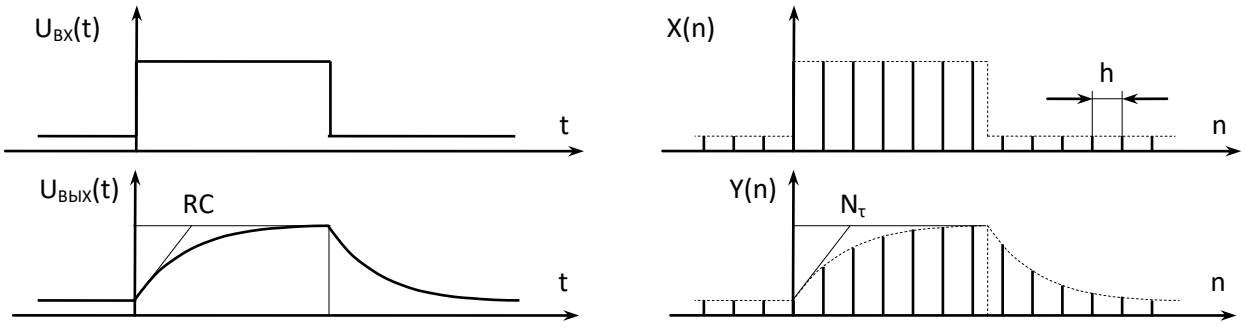
$$U_{ВХ}(t) \rightarrow X(n); \qquad U_{ВЫХ}(t) \rightarrow Y(n); \qquad \frac{dU_{ВЫХ}(t)}{dt} \rightarrow \frac{Y(n+1) - Y(n)}{h}$$

$$\frac{dU_{ВЫХ}(t)}{dt} = \frac{U_{ВХ}(t) - U_{ВЫХ}(t)}{RC} \rightarrow \frac{Y(n+1) - Y(n)}{h} = \frac{X(n) - Y(n)}{RC}; \qquad \mathrm{N}_\tau = \frac{RC}{h}.$$

ОтношениеN_τ = RC/h называется «машинная постоянная времени». Это безразмерная величина, равная количеству машинных циклов h, соответствующему интервалу времени, равному постоянной времени фильтра RC. Разностное уравнение для ФНЧ имеет вид:

$$Y(n+1) = Y(n) + \frac{X(n) - Y(n)}{\mathrm{N}_\tau};$$

Разностное уравнение фильтра нижних частот показывает, что новое значение выходного сигнала равно предыдущему «старому» значению плюс часть разности между входным и выходным сигналами. При увеличении машинной постоянной времени замедляется скорость изменения выходного сигнала и уменьшается граничная частота.



Рим. 6.8. Временные диаграммы для аналогового и цифрового фильтров

Реакция цифрового фильтра на последовательности отсчетов, имеющие огибающую вида прямоугольных импульсов имеет огибающую, подобную реакции аналогового фильтра, для которой:

подкасательная равна N_τ ;

переходной процесс практически заканчивается за интервал $3N_\tau$.

Импульсные сигналы используются при моделировании цифрового фильтра, они могут поступать в реальном времени от датчиков, а также могут быть представлены в виде таблицы данных.

Выражение для модуля коэффициента передачи можно получить из формулы аналогового фильтра, которая содержит отношение периода гармоники, соответствующей частоте среза T_c к периоду гармоники входного сигнала T.

Для цифрового фильтра период огибающей определяет количество отсчетов N_x , а период гармоники, соответствующей частоте среза - количество отсчетов $2 \cdot \pi \cdot N_\tau$.

$$K(T) = \frac{1}{\sqrt{1^2 + (\frac{T_C}{T})^2}}; \quad K(n) = \frac{1}{\sqrt{1^2 + (\frac{2 \cdot \pi \cdot N_\tau}{N_x})^2}}.$$

Пример. Входной сигнал имеет амплитуду $X_{max} = 100$, период $N_x = 16$ отсчетов, машинная постоянная времени 4. Определим K(n) и амплитуду выходного сигнала.

$$K(n) = \frac{1}{\sqrt{1^2 + (\frac{2 \cdot \pi \cdot 4}{16})^2}} = \frac{1}{\sqrt{1^2 + (1{,}57)^2}} = \frac{1}{\sqrt{3{,}46}} = 0{,}54; \quad Y_m = 54.$$

Для входного сигнала с периодом период $N_x = 4$ отсчета получим:

$$K(n) = \frac{1}{\sqrt{1^2 + (\frac{2 \cdot \pi \cdot 4}{4})^2}} = \frac{1}{\sqrt{1^2 + (6{,}28)^2}} = \frac{1}{\sqrt{40{,}44}} = 0{,}157; \quad Y_m = 16.$$

Анализ цифрового ФвЧ первого порядка.

Дифференциальное уравнение, описывающее аналоговый ФВЧ, позволяет записать соответствующее разностное уравнение для цифрового фильтра.

$$\frac{X(n+1) - X(n)}{h} - \frac{Y(n+1) - Y(n)}{h} = \frac{Y(n)}{RC}; \qquad Y(n+1) = X(n+1) - X(n) + Y(n) - \cdot \frac{Y(n)}{N_\tau}; \quad N_\tau = \frac{RC}{h}.$$

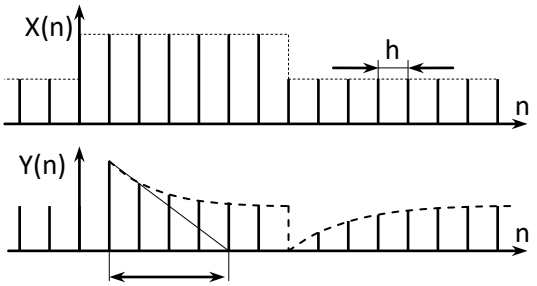


Рис. 6.9. Работа цифрового ФВЧ

Разностное уравнение цифрового фильтра верхних частот показывает, что на выход полностью передается приращение входного сигнала X(n+1) – X(n). При увеличении машинной

постоянной времени замедляется скорость изменения выходного сигнала и уменьшается граничная частота.

Работа ФВЧ. ФВЧ пропускает на выход резкие изменения входного сигнала (рис 6.9). При подаче на вход импульса, на выходе формируются выбросы напряжения разного знака. Эту особенность необходимо учитывать при моделировании, используя дополнительный, или смещенный код. Использование смещенного кода дает более наглядный результат.

Выражение для модуля коэффициента передачи можно получить из формулы аналогового фильтра.

$$K(T) = \frac{1}{\sqrt{1^2 + (\frac{T}{T_C})^2}}; \quad K(n) = \frac{1}{\sqrt{1^2 + (\frac{N_x}{2 \cdot \pi \cdot N_\tau})^2}}.$$

Пример. Входной сигнал имеет амплитуду $X_{max} = 100$, период $N_x = 16$ отсчетов, машинная постоянная времени 4. Определим K(n) и амплитуду выходного сигнала.

$$K(n) = \frac{1}{\sqrt{1^2 + (\frac{16}{2 \cdot \pi \cdot 4})^2}} = \frac{1}{\sqrt{1^2 + (0{,}64)^2}} = \frac{1}{\sqrt{1{,}4}} = 0{,}85; \quad Y_m = 85.$$

Для входного сигнала с периодом период $N_x = 4$ отсчета получим:

$$K(n) = \frac{1}{\sqrt{1^2 + (\frac{4}{2 \cdot \pi \cdot 4})^2}} = \frac{1}{\sqrt{1^2 + (0{,}16)^2}} = \frac{1}{\sqrt{1{,}025}} = 0{,}98; \quad Y_m = 98.$$

Цифровой фильтр с конечной импульсной характеристикой (КИХ-фильтр)

не имеет внутренних обратных связей, поэтому его называют не рекурсивным. Для формирования выходного сигнала используются только отсчеты входных сигналов, взятые с определенными коэффициентами. Совокупность отсчетов, используемых для определения текущего значения выходного сигнала, образует «скользящее окно». Переходные процессы полностью заканчиваются за конечный интервал времени, поэтому импульсная характеристика фильтра конечна. Аналоговых фильтров данного типа не существует.

$$y(n) = \frac{1}{N} \sum_{k=1}^{N-1} b_k \cdot x(n-k).$$

Выходной сигнал y(n) определяется суммой входных сигналов x(n-k) с весовыми коэффициентами. h(k). Число N в уравнении представляет собой число звеньев и определяет эффективность фильтра, как было сказано выше.

Наиболее простой КИХ - фильтр - скользящего среднего (moving average) - имеет весовые коэффициенты, равные единице.

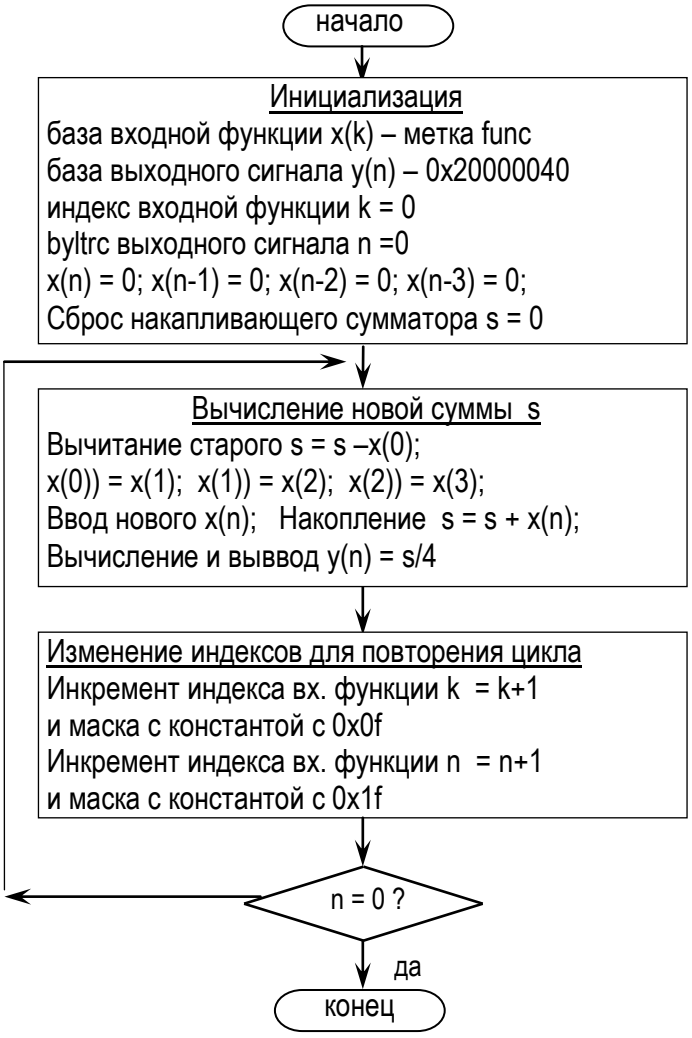


Рис. 6.10. Алгоритм работы КИХ - фильтра

Выходной сигнал для 4 -точечного ФНЧ определяется по формуле.

$y(n) = ¼ [x(n)+x(n–1)+x(n–2)+x(n–3)]$

Алгоритм работы 4-точечного КИХ - фильтра скользящего среднего приведен на рис. 6.10. При программной реализации выборка из четырех последних отсчетов входных данных всегда сохраняется в регистрах процессора, которые используются вместе как буфер FIFO. Сумма входных сигналов, записанная в квадратных скобках, содержится в накапливающем сумматоре.

Для вычисления следующего отсчета выходного сигнала y(n+1) необходимо переместить окно и выполнить вычисления:

Новое содержимое накапливающего сумматора проще всего получить, если от старого содержимого вначале отнять самый старый элемент x(n-3), а затем переместить элементы в буфере и прибавить новый элемент x(n+1) в сумматор и буфер. Программа содержится в подпрограмме pp3.

Реализация цифровых фильтров на основе ПЛИС.

Для выполнении алгоритмов цифровой фильтрации используются аппаратные средства реализуемые в ПЛИС. В этом случае схему фильтра составляют, используя следующие элементы.

1. Элемент задержки сигнала на один такт. Это элемент памяти – регистр с динамическим управлением -, который сохраняет на выходе значение сигнала, поступившее на его вход в предыдущем такте. Элемент задержки обозначают как z^{-1} в соответствии с его представлением при z-преобразовании.

2.Сумматор – элемент, построенный аз комбинационных схем, содержащий два входа и один выход. На схемах изображают кружком.

3. Устройство умножения на коэффициент.

Синтез фильтров позволяет определить веса коэффициентов и число звеньев для реализации практически любой частотной характеристики. Методика синтеза с использованием MathLab, подробно описанная в литературе, позволяет получить оптимальный фильтр, обеспечивающий наилучшее отношение сигнал / шум для заданных сигналов. При синтезе используют математический аппарат Z – преобразования, в котором оператор Z^{-1} , использованный в семах фильтров (рис. 6.10 и 6.12) выполняет задержку сигнала на один машинный такт.

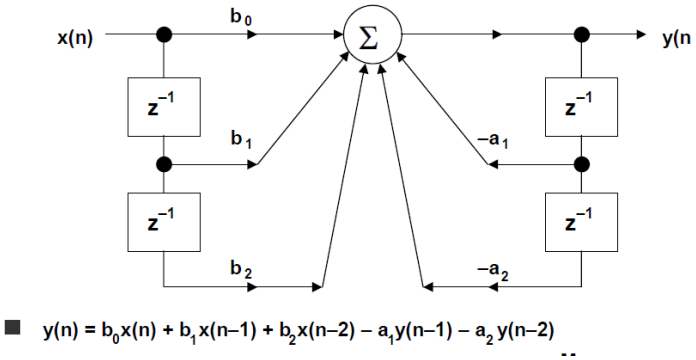
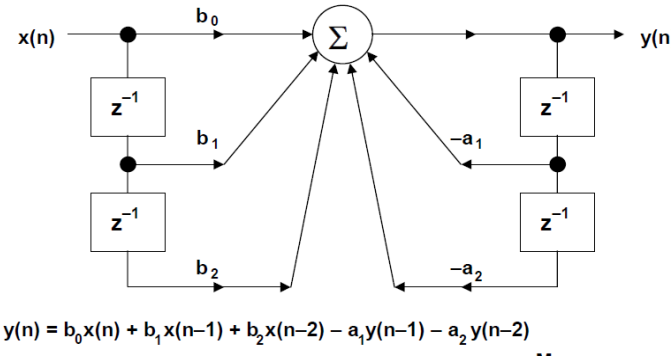


Рис. 6.11. Схема рекурсивного БИХ - фильтра



Рекурсивный БИХ - фильтр

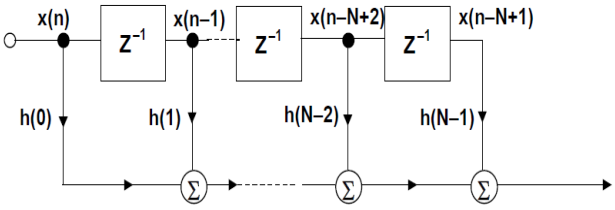
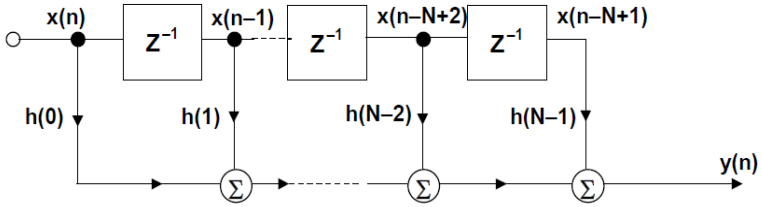


Рис. 6.12. Схема нерекурсивного КИХ - фильтра



Нерекурсивного КИХ - фильтра

Пример схемы рекурсивного фильтра используют для представления любого типа фильтров, показанных на рис. 6.1. Используя методику синтеза, получают весовые коэффициенты для требуемого типа фильтра с заданными параметрами. БИХ – фильтры позволяют при небольших аппаратных затратах получить необходимое качество.

Цифровой КИХ - фильтр имеет незначительную неравномерность характеристики в полосе пропускания, линейную фазовую характеристику и значительно более крутой спад частотной характеристики. Таких показателей невозможно достичь аналоговыми методами.

Однако, высокоэффективные КИХ-фильтры строятся с большим числом операций умножения с накоплением и поэтому требуют использования быстрых и эффективных процессоров.

Экспериментальная часть

Задание 6.1. Исследование цифровых БИХ-фильтров при импульсных входных сигналах.

Используется программа PR6, содержащая две подпрограммы для исследования ФНЧ (pp1) и ФВЧ (pp2), которые могут работать независимо. В каждой подпрограмме вначале выполняется инициализация указателей памяти для обращения к массивам командами с косвенной адресацией, в которых адрес содержится в двух регистрах в виде базы и индекса. Входной сигнал X(k) записан в ПЗУ команд в виде нескольких вариантов таблиц функций func(k), содержащих 16 отсчетов. Выходной сигнал Y(n) записывается в ОЗУ данных в виде массивjd из 32 отсчетов. Для

выходных сигналов ФНЧ базовый адрес массива равен 0x20000000, затем, начиная с адреса 0x20000020 записывается массив выходных сигналов ФВЧ.

В программе фильтра выбрана машинная постоянная времени N_τ =4, , кратная степени числа 2, это позволяет выполнить деление посредством операций сдвига вправо. Выбраны команды арифметического сдвига, при выполнении которых старший знаковый разряд сохраняется. Амплитуда импульсов равной 1ю0, для удобства анализа результатов. Такое значение исключает переполнение при использовании данных в виде 8-разрядных чисел со знаком.

Введите и запустите программу.

1)Запишите в регистр r0 метку, соответствующую базовому адресу таблицы входных сигналов func1. Настройте параметры окна памяти (рис. 6.12).

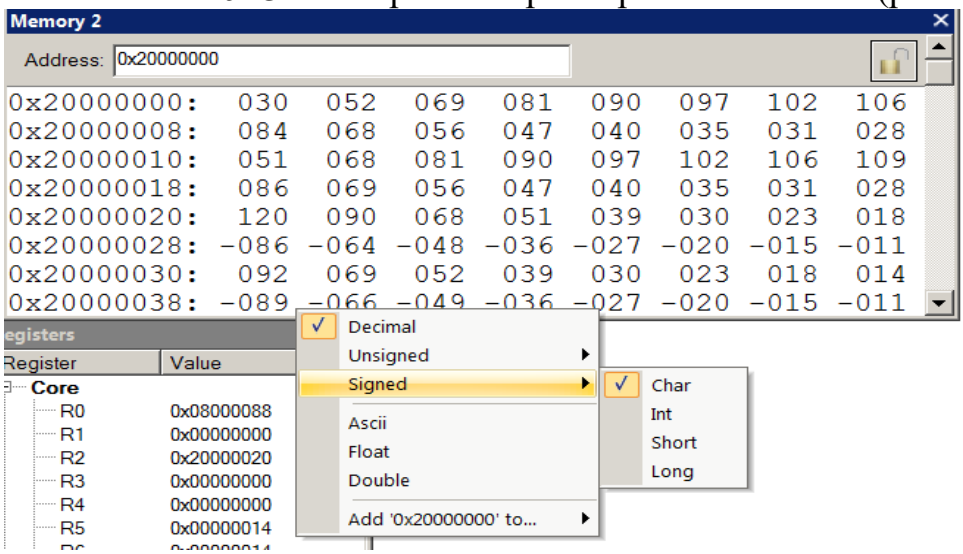


Рис. 6.12. Настройка отображения содержимого памяти и

2) Постройте в масштабе для ФНЧ и для ФВЧ графики, на каждом из которых отобразите огибающую входного сигнала и отсчеты выходного сигнала. Постройте подкасательную, определите экспериментально машинную постоянную времени. Определите отношение длительности входного импульса к машинной постоянной времени.

3) Опишите, какие изменения формы импульсов выполняют данные фильтры.

4) Поясните причины появления погрешностей.

Задание 6.2. Исследование работы цифровых БИХ-фильтра при входных сигналах малой длительности.

1)Запишите в регистр r0 метку, соответствующую базовому адресу таблицы входных сигналов func2.

2) Постройте в масштабе для ФНЧ и для ФВЧ графики, на каждом из которых отобразите огибающую входного сигнала и отсчеты выходного сигнала. Определите экспериментально машинную постоянную времени фильтров. Укажите отношение длительности входного импульса к машинной постоянной времени.

3)Постройте графики изменения постоянной составляющей импульсов, определяя ее для каждого периода сигналов.

4) Определите максимальный размах колебаний для входного сигнала и для каждого фильтра.

5) Опишите, какие свойства имеют данные фильтры при передаче на выход постоянной составляющей сигнала и высокочастотных гармоник.

Задание 6.3. Исследование работы цифровых БИХ-фильтров при гармоническом входном сигнале.

1) Запишите в регистр r0 метку, соответствующую базовому адресу таблицы гармонического сигнала func3. Запустите программу.

2) Постройте в масштабе два графика (для ФНЧ и для ФВЧ), на каждом из которых отобразите огибающую входного сигнала и отсчеты выходного сигнала.

3) определите отношение периода входного сигнала к машинной постоянной времени.

4) Запишите формулы, рассчитайте теоретическое значение модуля коэффициента передачи, определите экспериментальное значение, сравните результаты.

5) Запишите в регистр r0 метку, соответствующую базовому адресу таблицы гармонического сигнала func4. Укажите отличие этого сигнала от предыдущего. Выполните пункты 2 – 4 для данного варианта входного сигнала, поясните результаты.

6) Состабте таблицу гармонического сигнала, период гармоники должен составляьб 8 отсчетов (0, 71, 100, 71, 0, -71, -100, -71). Запишите таблицу в программу с меткой базового адреса func5.

Укажите отличие этого сигнала от предыдущего. Выполните пункты 2 – 4 для данного варианта входного сигнала, поясните результаты.

Задание 6.4. Исследование работы КИХ - фильтра нижних частот.

В программе PR 6 включите выполнение подпрограммы pp3 (знак «;»в начале строки bl pp3 должен отсутствовать). Выполнение подпрограмм pp1 и pp2 отключите.

1) Запишите в регистр r0 метку, соответствующую базовому адресу таблицы func1, содержавшей сигналы с огибающей вида прямоугольных импульсов, имеющих длительности импульса и паузы, содержащие 8 отсчетов. По результатам анализа постройте в масштабе два графика, на каждом из которых отобразите огибающую входного сигнала и отсчеты выходного сигнала. По графику определите продолжительность переходного процесса. Докажите, что длительность процесса конечна.

2) Запишите в регистр r0 метку, соответствующую базовому адресу таблицы func2, содержавшей импульсные сигналы, имеющие длительность импульса и паузу, содержащие 2 отсчета. По результатам анализа постройте в масштабе два графика, на каждом из которых отобразите огибающую входного сигнала и отсчеты выходного сигнала. По графику определите постоянные составляющие вхдного и выходного сигналов.

3) Запишите в регистр r0 метку, соответствующую базовому адресу таблицы func3, содержавшей гармонический сигнал. По результатам анализа постройте в масштабе графики, на которых отобразите огибающие входного и выходного сигналов. Определите модуль коэффициента передачи.

4)) Запишите в регистр r0 метку, соответствующую базовому адресу таблицы гармонического сигнала func4. Укажите отличие этого сигнала от предыдущего. Выполните моделирование. Опишите результат..

5) Исследуйте работу фильтра при гармоническом сигнале, период которого составляет 8 отсчетов (он записан в предыдущем пункте в таблицу с меткой func5). Укажите отличие этого сигнала от предыдущего. Выполните анализ, поясните результаты.

Задание 6.5. Исследование АЧХ аналоговых фильтров

Для ФНЧ и ФВЧ заданы одинаковые параметры : C = 1 мкФ, R -в в таблице, в Омах, в соответствии с вариантом.

Вариант	1	2	3	4	5	6	7	8	9	10	11	12
R	50	60	70	80	90	100	110	120	130	140	150	160

Вариант	13	14	15	16	17	18	19	20	21	22	23	24
R	170	180	190	200	210	220	230	240	250	260	270	280

1. Рассчитайте постоянную времени фильтров, и частоту среза АЧХ в радианах/с. и в Гц.

2. Рассчитайте модуль комплексного коэффициента передачи на частотах 0,1Fc, 05 Fc,fc, 2Fc, 10Fc, постройте теоретические АЧХ для ФНЧ и ФВЧ.

3. В режиме AC получите экспериментальные АЧХ для ФНЧ и ФВЧ. Проверьте экспериментально выполненные расчеты.

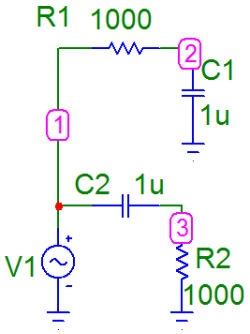
Пример. Заданы параметры C = 1мкф, R = 100 Ом.

Расчет. Постоянная времени фильтров: $\tau = R*C = 1000 * 10^{-6} = 10^{-3}$ 1/с.

Частота среза АЧХ: $\omega_c = 1/\tau = 10^3$ 1/с.

Частота среза АЧХ, Гц: $Fc = \omega_c/2\pi = 1/(2\pi\tau) = 159$Гц.

В схемах RC- фильтров нижних частот (ФНЧ) и верхних частот (ФВЧ) параметры R и C которых одинаковы, а входы подключены к одному общему источнику сигнала (рис. 6.13).



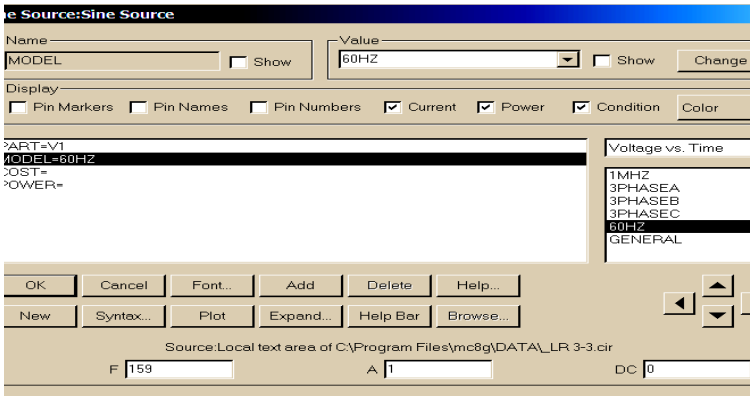


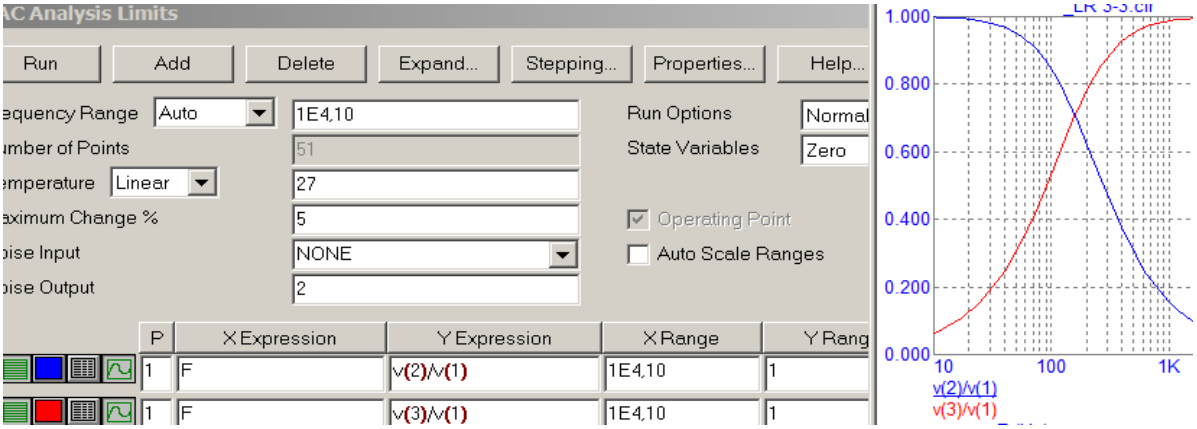
Рис. 6.13 Схема

Постоянные времени фильтров равны, поэтому граничные частоты фильтров совпадают. Для ввода схемы откройте окно редактора схем (File / New / Schematic) откройте палитру № 1 (Ctrl+1), установите параметры источника Sine Source как показано на рис. 6.14. Для размещения компонента на схеме: переместите курсор в

область размещения, затем при нажатой левой кнопке, не отпуская ее, нажмите правую кнопку несколько раз для вращения символа компонента.

Запустите анализ в режиме AC. После выбора метода анализа появляется диалоговое окно задания параметров (пределов) моделирования . В окне AC Limits (рис. 6.12) установите диапазон частот. Первой указывается верхняя частота диапазона, затем, через запятую - нижняя. Для шкалы частот используется логарифмический масштаб. С учетом граничной частоты фильтров (160 герц) указан диапазон от 10 килогерц до 10 герц.

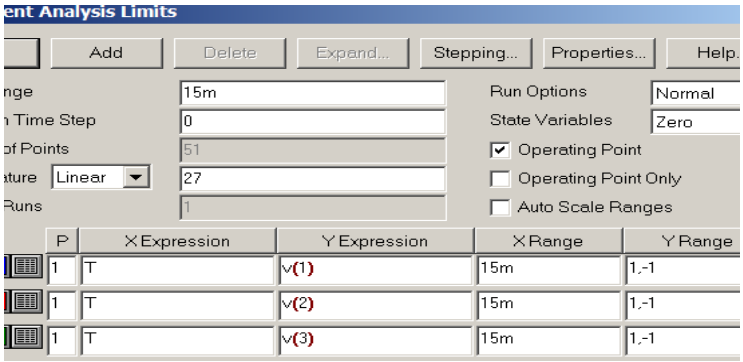
В таблице установки параметров графиков в столбце таблицы P указывается номера отдельных графиков от 1 до 9. В одном окне можно разместить несколько графиков разного цвета. В графе переменная X (X Expression) указывается имя переменной - частота F. В графе Y Expression записывается математическое выражение для переменной Y(рис. 6.15).

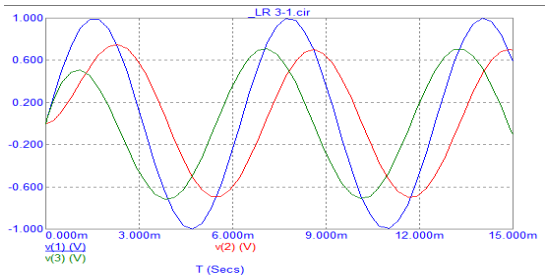


Выполните анализ результата.

1) Выделите область пересечения графиков (f7), включите режим курсоров (A8), определите координаты точки пересечения, поясните физический смысл этих координат.

2) Определите экспериментально модуль комплексного коэффициента передачи на частотах 0,1Fc, 05 Fc,fc, 2Fc, 10Fc.





Задание 6.6. Исследование аналоговых фильтров при гармоническом входном сигнале.

Получите временные диаграммы в режиме анализа Transient для гармонических сигналов. на входах фильтров: с частотой $F_{BX}=F_{C.}$

Сравните теоретические значения с результатами моделирования.

При установке параметров (Рис. 6.16) в строке Time Range необходимо указать время моделирования с учетом периода входного сигнала схемы. В рассматриваемом примере частота источника равна граничной частоте пропускания фильтров – 160 Гц, (см. рис. 2.2), а период составляет 6,25 мс. Выбираем время моделирования 15 мс, при этом на временной диаграмме получим несколько периодов.

По графику (Рис. 6.17 определите параметры выходных сигналов фильтров.

Задание 6.7. Исследование аналоговых фильтров первого порядка при импульсных входных сигналах.

Длительность импульса 5мс., период 10 мс.

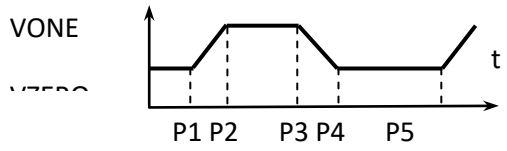


Рис. 6.18. Модель импульса

При моделировании в схеме фильтров (рис. 6.8) вместо источника синусоидального сигнала включите источник импульсного сигнала Pulse Source.Для любого источника используется модель импульса (рис. 6.18), в которой можно установить заданные временные параметры.

После запуска режима Transient в окне параметров в строке Time Range (рис. 6.17) указано время моделирования 15 мс, чтобы иметь на графике почти 2 периода импульсов и удобную для отсчета шкалу. Флажок Auto Scale Ranges – отключен, диапазоны переменных установлены в соответствии с амплитудой входных импульсов. В столбце P задано построение двух отдельных графиков.

1) Поясните, какие функции отображены на временных диаграммах, с какой целью используется два отдельных графика?

2) На графиках (рис. 6.17 и рис. 6.18) сделайте геометрические построения для определения постоянных времени фильтров по экспериментальным данным.

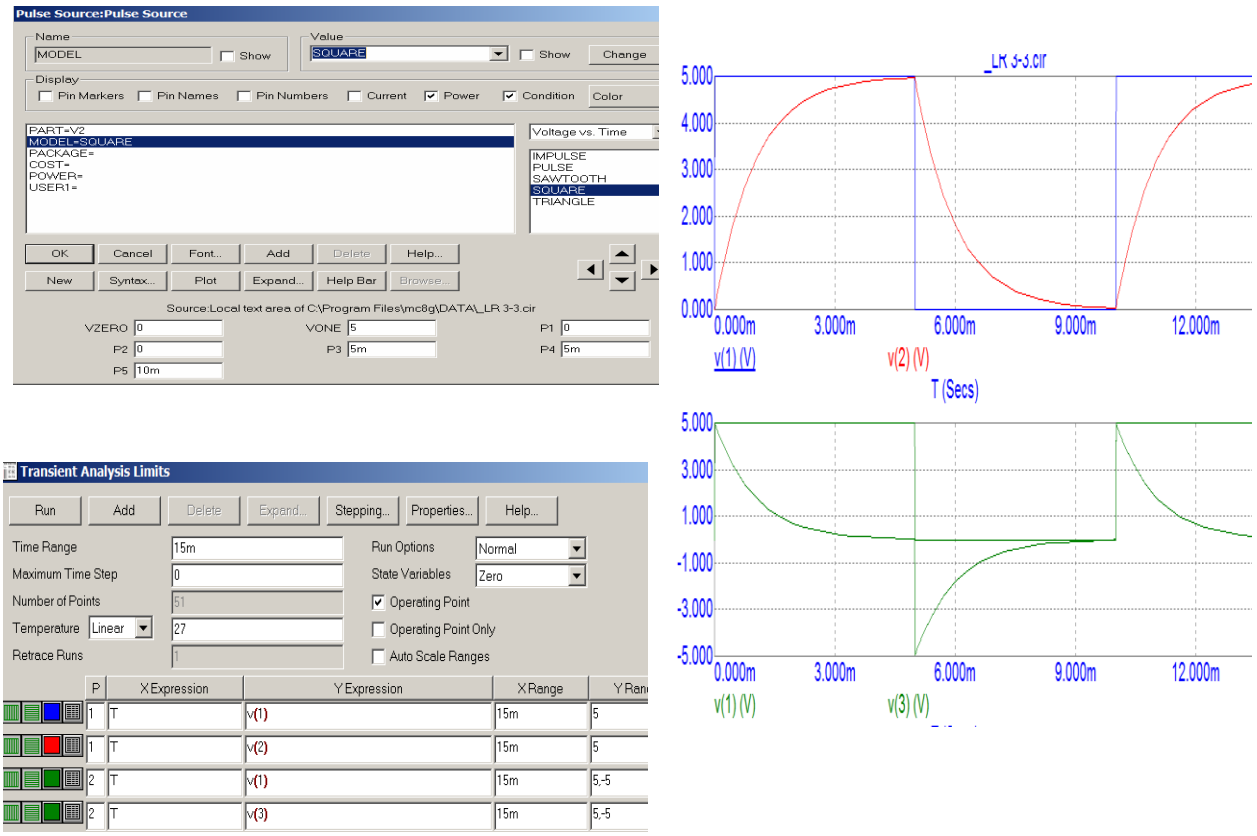


Рис. 6.121. Результат анализа

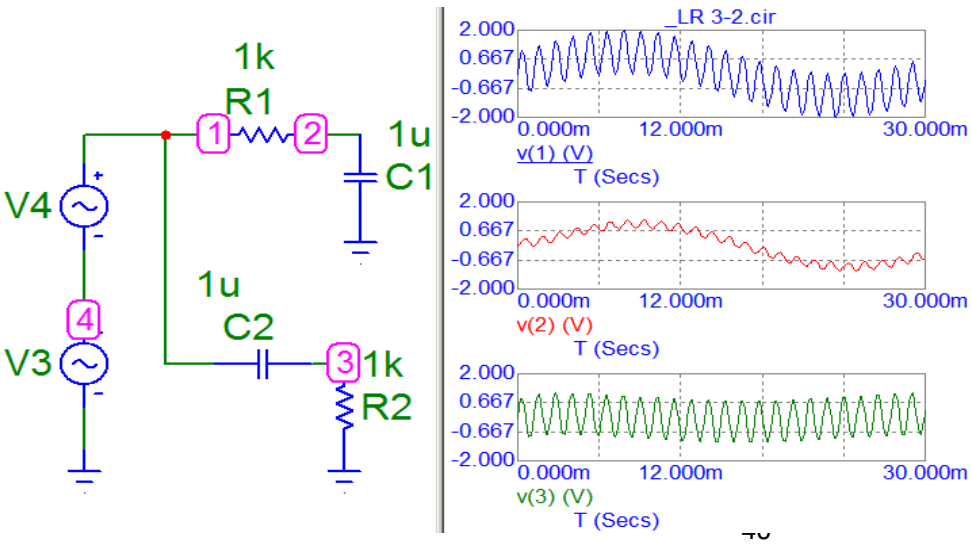
3) Запишите выражение для реакции ФНЧ и ФВЧ на единичный скачок, изобразите графики.

4) Постройте теоретические временные диаграммы для выходных напряжений фильтров при подаче на вход импульсов различной длительности с коэффициентом заполнения 0,25:

- длительность импульса равна постоянной времени фильтра,
- длительность импульса в 10 раз меньше постоянной времени фильтра,
- длительность импульса в 10 раз больше постоянной времени фильтра,

5)Выполните моделирование в режиме Transient – анализ переходных процессов. Сравните экспериментальные и теоретические характеристики.

Задание 6.8. Исследование аналоговых фильтров при входном сигнале, содержащем несколько гарьоник.



Получите временные диаграммы в режиме анализа Transient для сигналов. на входах и выходах фильтров, если входной сигнал равен сумме двух гармоник с амплитудой 1 В и частотами $F_{BX}1=5*F_C$, $F_{BX}2=0,2*F_C$.

Рассчитайте коэффициенты передачи фильтров для данных гармоник.
Определите экспериментально амплитуды помех на выходах фильтров.

Контрольные вопросы

1. Классификация фильтров по виду амплитудно– частотной характеристики.
2. Изобразите схемы , запишите выражения для ККП.
3. Запишите выражения для вычисления модуля ККП аналоговых ФНЧ и ФИЧ.
4. Запишите выражения для вычисления модуля ККП цифровых ФНЧ и ФИЧ.
5. Как определяется машинная постоянная времени цифрового фильтра?
6. 4. Какие этапы содержит процесс составления разностного уравнения?
7. Как экспериментально определить машинную постоянную времени?
8. Как теоретически определить амплитуду огибающей гармонического сигнала на выходе цифрового фильтра?
9. Как практически определить амплитуду огибающей гармонического сигнала на
10. выходе цифрового фильтра?
    Составьте программу, формирующую заданную преподавателем
11. последовательность дискретных значений, запишите последовательность в
12. таблицу, исследуйте работу фильтра
13. Поясните причины появления погшрешностей при вычислениях параметров цифровых фильтров.
14. Запишите формулу дл\я КИХ-фильтра НЧ.

Программа для исследования цифровых фильтров

```
    area stack, noinit, readwrite
    space 0x400
stack_top
        area reset, data, readonly
    dcd stack_top
    dcd start
    area program,code,readonly
    entry
start
;   bl   pp1     ;
;   bl  pp2      ;
    bl  pp3      ;
m   b   m
pp1 ldr  r0, = func3     ;baza X(k)
    mov  r1, #0          ;undex k
    mov  r2, #0x20000000;baza Y(n)
    mov  r3, #0          ;undex n
    mov  r6, #0          ;y(n)
m1  ldrsb  r5, [r0,r1]   ;x(n)
    sub R7,  R5, r6      ;x(n)-y(n)
    add  r8, r6, r7, asr # 2;y(n+1)
    strb  r8, [r2,r3]    ;write y(n+1)
    mov  r6, r8          ;new y(n)
    add  r1,  #1         ;k = k+1
    and  r1,   #0x0f     ;cout mod 16
    add  r3,  #1         ;n:= n+1
    ands  r3,  #0x01f    ;cout mod 32
    cmp  r3, #0x00
    bne   m1
    bx  lr
pp2 ldr  r0, = func3;baza X(k)
    mov  r1, #0          ;undex k
    ldr  r2,= 0x20000020; baza Y(n)
    mov  r3, #0          ;undex n
    mov  r5, #0          ;x(n)
    mov  r6, #0          ;y(n)
m2  ldrsb  r7, [r0,r1]   ;x(n+1)
    mov R8,  r7      ;r8:= x(n+1)
    sub r8, r5  ;-x(n)
    add r8, r6  ;+ y(n)
    sub r8, r6, asr # 2 ;r8=y(n+1)
    strb  r8, [r2,r3] ;write y(n+1)
    mov r5,r7        ;new x(n)
    mov r6,r8        ;new y(n)
    add  r1,  #1     ;k =k + 1
    and  r1,   #0x0f
    add  r3,  #1     ;n:= n+1
    ands  r3,  #0x01f
    cmp  r3, #0x00
    bne   m2
    bx  lr
pp3 ldr  r0, = func1;baza X(n)
    mov  r1, #0      ;undex n
    ldr  r2,= 0x20000040; baza Y(k)
    mov  r3, #0          ;undex k
    mov  r4, #0          ;x(n)
    mov  r5, #0          ;x(n-1)
    mov  r6, #0          ;x(n12)
    mov  r7, #0          ;x(n-3)
m3  sub r8, r7  ;s:=s-x(n-3)
    mov R7,  r6      ;nev x(n-3)
    mov R6,  r5      ;nev x(n-2)
    mov R5,  r4      ;nev x(n-1)
    ldrsb  r4, [r0,r1];nev x(n)
    add r8, r4  ;s:=s+ y(n)
    mov r9, r8, asr # 2 ;r8=s/4
    strb  r9 [r2,r3] ;write y(n+1)
    add  r1,  #1     ;n =n + 1
    and  r1,   #0x0f
    add  r3,  #1     ;k:= k+1
    ands  r3,  #0x01f
    cmp  r3, #0x00
    bne   m3
    bx  lr
func1
    dcb 120,120,120,120
    dcb 120,120,120,120
    dcb 20,20,20,20
    dcb 20,20,20,20
func2
    dcb 120,120,20,20
    dcb 120,120,20,20
    dcb 120,120,20,20
    dcb 120,120,20,20
func3
    dcb 0,38,71,92
    dcb 100,92,71,38
    dcb 0,-38,-71,-92
    dcb -100,-92,-71,-38
func4
    dcb  0,100,0,-100
    dcb  0,100,0,-100
    dcb  0,100,0,-100
    dcb  0,100,0,-100
    END
```

Работа 7. Устройство дл вычисления коэффициентов ряда Фурье

Теоретическая часть

Электрические сигналы, используемые для передачи данных, имеют сложную фору. При анализе электрических цепей исследуют реакцию цепи на элементарные функции времени. Сложные сигналы можно представить состоящими из элементарных сигналов. В основе спектрального анализа представление сигнала сложной формы в виде суммы гармонических сигналов. Спектр (от лат. spectrum – представление, образ) – совокупность простых гармонических колебаний, являющихся составляющими сложного колебательного процесса.

Спектральный анализ периодических функций заключается в нахождении коэффициентов ряда Фурье. Если задана периодическая функция u(t) на интервале $-T/2 < t < T/2$ с конечным значением интеграла и конечным числом разрывов за период (условия Дирихле), то она разлагается в ряд Фурье, содержащий гармонические составляющие, частоты которых кратны основной частоте $\omega_1 = 2\pi / T$.

$$u(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty}(a_n \cos n \cdot \omega_1 \cdot t + b_n \sin n \cdot \omega_1 \cdot t)$$

Коэффициенты ряда Фурье определяются по формулам:

$$a_0 = \frac{2}{T}\int_{-T/2}^{T/2} u(t)dt; \quad a_n = \frac{2}{T}\int_{-T/2}^{T/2} u(t)\cdot\cos n\cdot\omega_1\cdot t\cdot dt; \quad b_n = \frac{2}{T}\int_{-T/2}^{T/2} u(t)\cdot\sin n\cdot\omega_1\cdot t\cdot dt;$$

Все гармоники ряда Фурье с частотами, кратными частоте первой гармоники ω_1: $\omega_n = n * \omega_1$, где n – номер гармоники..

Заменяя две гармоники одинаковой частоты с нулевыми начальными фазами на одну с ненулевой фазой, получим второй вариант записи ряда Фурье в тригонометрической форме:

$$u(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} C_n(\cos n\cdot\omega_1\cdot t + \varphi_n); \quad C_n = \sqrt{a_n^2 + b_n^2}; \quad \varphi_n = arctg\frac{b_n}{a_n}$$

Эта форма ряда Фурье определяет спектр сигнала – упорядоченное множество (совокупность) амплитуд C_n и фаз $ф_n$. Наибольший интерес при проектировании электронных устройств представляет амплитудный спектр, содержащий модули амплитуд всех гармоник в виде положительных чисел C_n.

При цифровой обработке выполняют дискретизацию и квантование непрерывных сигналов. Дискретизация – приближенное представление

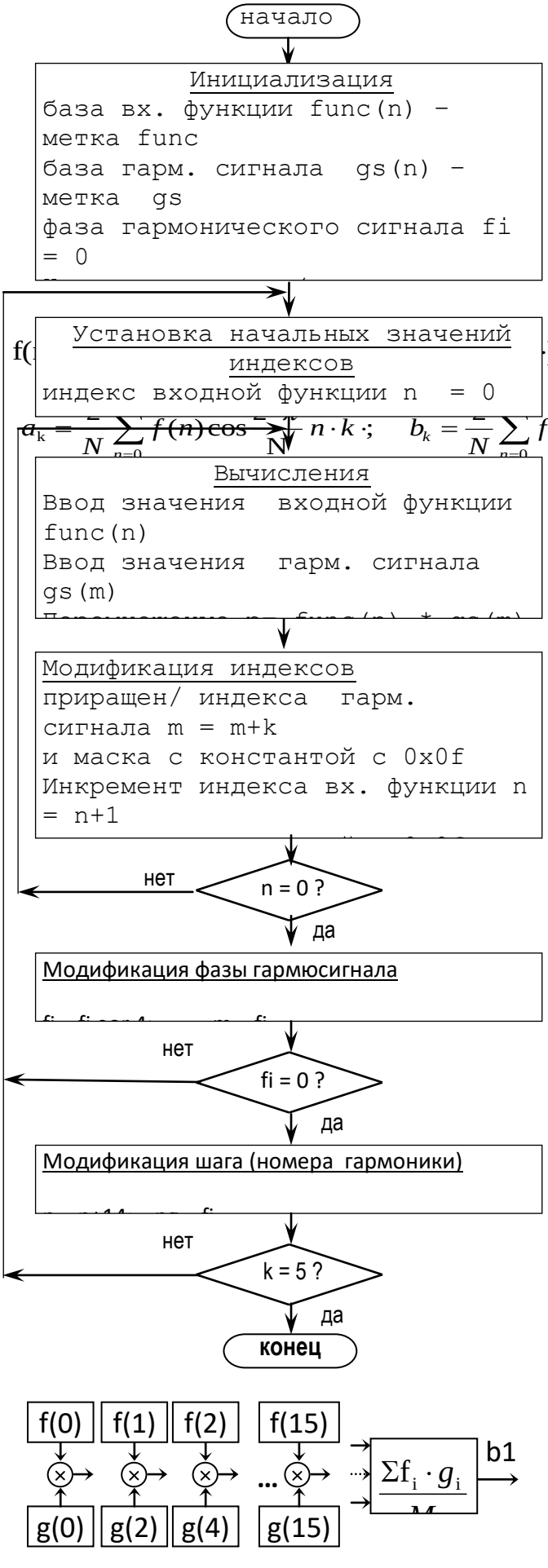


Рис. 7.3. Схема вычисления b2

непрерывного сигнала цифровым n-разрядным кодом с относительной погрешностью $1 / 2^n$.

Квантование - представление непрерывного сигнала в виде отдельных отсчетов (значений). Периоду непрерывного сигнала T будет соответствовать N дискретных отсчетов, а аргументу непрерывного сигнала t - номер отсчета n. Заменяя в приведенной выше формуле для ряда Фурье частоту $\omega_1 = 2\pi / T$ на $2\pi / N$, а также аргумент t на номер отсчета n, получим формулы для представления дискретной функции в виде ряда Фурье для для определения коэффициентов гармоник, в которых: . N – количество отсчетов функции; n – номер отсчета функции; k – номер гармоники.

Разработка алгоритма вычисления амплитуд гармоник ak, bk для k =1..4 для периодической функции, заданной в виде таблицы из 16 чисел со знаком в дополнительном коде разрядностью 8 бит.

Гармонический сигнал gs(m), соответствующий первой гармонике и входная сигнал func(n) представлены в программе в виде таблиц, размещенных в виде массива чисел в ПЗУ команд. Каждая тааблица содержит N=16 отсчетов. Начало каждой таблицы отмечают меткой. В программе формируется базовый адрес массива, соответствующий этой метке. Номер отсчета функции задается индексом массива.

$$b1 = \frac{1}{\mathrm{N} \cdot U_m} \cdot \sum_{n=0}^{N-1} f(n) \cdot \mathrm{gs(n)}$$

Для вычисления амплитуды первой гармоники b1 суммируются произведения дискретных значений входной функции функции f(n) и гармонической

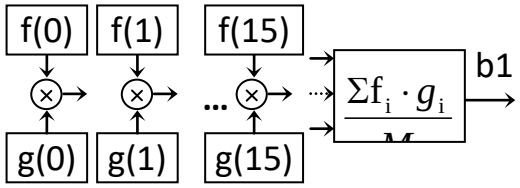


Рис. 7.1. Схема вычисления b1

функции gs(m) для отсчетов с одинаковыми номерами n = m, как показано на рис. 7.1.

$$b1 = \frac{1}{\mathrm{N} \cdot U_m} \cdot \sum_{n=0}^{N-1} f(n) \cdot \mathrm{gs(n+4)}$$

Для вычисления амплитуды первой гармоники a1 требуется функция косинус, которая получается из таблицы gs(n) с учетом начальной фазы cos 0 = sin 90. Для функции с периодом 16 отсчетов начальной фпзе 90° соответствует отсчет 4(рис. 7.2)..

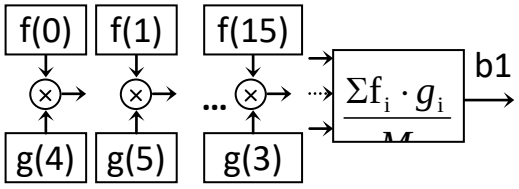


Рис. 7.2. Схема вычисления a1

При вычислении амплитуды второй

$$b2 = \frac{1}{\mathrm{N} \cdot U_m} \cdot \sum_{n=0}^{N-1} f(n) \cdot \mathrm{gs(2*n)}$$

гармоники b2 используется гармоническая функция, частота которой вдвое выше частоты первой гармоники, а период вдвое меньше. Дискретные отсчеты для этой функции получаются из исходной таблицы при изменении индекса с шагом 2 (рис. 7.3),.при этом будет использовано N/2 отсчетов. Интервалу из 16 отсчетов, для которого задана входная функция, будет соответствовать два периода гармонического сигнала.

Максимальное количество гармоник, которые можно определить для функции , заданной N отсчетами ограничено и составляет N/2. В данном случае максимальный номер гармоники равен 8. На интервале, содержащем 16 отсчетов, будет отображаться 8 периодов гармонического сигнала. Каждый период гармонического сигнала будут определять всего два отсчета. Минимальное количество отсчетов за период, , при котором функция может быть восстановлена равно двум. Это теоретический предел, определяемый теоремой Котельникова.

Алгоритм программы – циклический. Периодически изменяются вид гармонического сигнала, определяющий начальную фазу fi и номер ггарионики k. Результат вычислений будет представлен в ОЗУ в виде последовательности значений

коэффициентов:

b1, a1, b2, a2, b3, a3, b4, a4.

Программа для вычисления коэффициентов Фурье.

```
1       area stack, noinit, readwrite
2       space 0x400             ;pr_7
3   stack_top
4       area reset, data, readonly
5       dcd stack_top
6       dcd start
7       area program,code,readonly
8       entry
9   start
10      ldr   r0, = func4;baza func
11      mov   r1, #0      ;index n
12      ldr   r2, = gs    ;baza gs(
13      mov   r6, #0x20000000;bazaout
14      mov   r4, #0      ;faza fi
15      mov   r5, #1      ;k-num_garm
16  k   mov   r3, r4      ;m = fi
17      mov   r9, #0
18      mov   r10, #0
19      mov   r11, #0     ;accumulators
20  m   ldrsb  r7, [r0, r1] ;func(n)
21      ldrsb  r8, [r2, r3] ;gs(m)
22      smull  r10, r9, r7,  r8
23      add  R11,  R10
24      add  r3,  r5 ;m =m+k
25      and  r3, #0x0f
26      add  r1, #1
27      ands  r1, #0x0f
28      ;cmps  r1, #0x00
29      bne m
30      asr  r11,#10
31      str  r11, [r6],#4
32      eors r4, 0x4
33      bne  k
34      add  r5, #1
35      cmps r5, #5
36      bne k
37  bc  b  bc
38  gs dcb 0,49,91,118
39      dcb 127,118,81,49
40      dcb 0,-49,-91,-118
41      dcb -127,-118,-81,-49
42  func1 dcb 0,38,71,92
43      dcb 100,92,71,38
44      dcb 0,-38,-71,-92
45      dcb -100,-92,-71,-38
46  func2  dcb 100,92,71,38
47      dcb 0,-38,-71,-92
48      dcb -100,-92,-71,-38
49      dcb 0,38,71,92
50  func3 dcb 0,71,100,71
51      dcb 0,-71,-100,-71
52      dcb 0,71,100,71
53      dcb 0,-71,-100,-71
54  func4 dcb 10,10,10,10
55      dcb 10,10,10,10
56      dcb -10,-10,-10,-10
57      dcb -10,-10,-10,-10
58      END
```

В программе в виде таблиц в ПЗУ представлены гармонический сигнал gs(n) и различные варианты входных сигналов. Массивы содержат по 16 отсчетов разрядностью 1 байт.

Значения функции gs(m) вычислялись по формуле gsn(m) = 128*sin(360 / 16)*m, где m-индекс. Шаг дискретизации τ равен: τ = 360/16 = 22,5°. Отсчеты приведены в виде десятичных чисел со знаком, это удобно при вводе. При отладке данные в регистрах отображаются в 16-ричном коде, поэтому в таблицах приведите 16-ричные значения отсчетов в дополнительном коде, необходимые при отладке.

Таблица значений входного сигнала func1 func1(n) = 100*sin(360 / 16)*n предназначена для тестирования программы. При спектральном анализе которой должен быть получен коэффициент b1, не равный 0, а остальные коэффициенты - равные 0.

Входной сигнал func2 содержит дискретные отсчеты, огибающая которых имеет вид прямоугольного импульса длительность которого 8 отсчетов, а период – 16 отсчетов.

Спектры импульсных сигналов содержат большое количество гармоник.

При моделировании необходимо указать метку определенного иассива как базовый адрес.

Таблица функции gs(m) = 128*sin(360 / 16)*m

m	0	1	2	3	4	5	6	7
n*τ	0	22,5	45	67,5	90	112,5	135	157,5
sin	0	0,38	0,71	0,92	1	0,92	0,71	0,308
gs(n)	0	49	91	118	127	118	91	49
m	8	9	10	11	12	13	14	15
n*τ	180	202,5	225	247,5	270	292,5	315	337,5
sin	0	-0,38	-0,71	-0,92	-1	-0,92	-0,71	-0,38
gs(n)	0	-49	-91	-118	-128	-118	-91	-49

Таблица функции func1(n) = 100*sin(360 / 16)*n

n	0	1	2	3	4	5	6	7
n*τ	0	22,5	45	67,5	90	112,5	135	157,5
sin	0	0,38	0,71	0,92	1	0,92	0,71	0,308
Д.к.	0	38	71	92	100	92	71	38
n	8	9	10	11	12	13	14	15
n*τ	180	202?5	225	247,5	270	292,5	315	337,5
sin	0	-0,38	-0,71	-0,92	-100	-0,92	-0,71	-0,38
Д.к.	0	218	185	164	156	164	185	218

Таблица функции func2(n) – импульсный сигнал

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
f	10	10	10	10	10	10	10	10	-10	-10	-10	-10	-10	-10	-10	-10
Д.к.	10	10	10	10	10	10	10	10	246	246	246	246	246	246	246	246
Hex	0A	0A	0A	0A	0A	0A	0A	0A	F6	F6	F6	F6	F6	F6	F6	F6

Аппаратная реализация устройства для вычисления коэффициентов ряда Фурье, подобного предыдущему, на основе ПЛИС.

Теоретическая часть

Аппаратная реализация устройства разрабатывается как проект с иерархической структурой в САПР Quartus с использованием приведенного ранее алгоритма.

Регистровая модель вычислительного устройства содержит отдельные устройства памяти для хранения исходных данных, гармонического сигнала и результатов. Это позволяет использовать раздельные адресные пространства для вех переменных и упрощает адресацию. Рассмотрим назначение элементов устройства.

Входной сигнал записывается в ПЗУ ROM_1 в виде таблицы из 16 чисел со знаком в дополнительном коде разрядностью 8 бит с использованием *.mif - файда.

Гармоническая функция времени gs(m) представлена в виде массива в ПЗУ (ROM_1), также содержащего N=16 отсчетов с шагом дискретизации, равным $2\pi/N = 360/16 = 22{,}5°$. После подачи адреса гармоники m в регистре гармоники Reg_gs фиксируется отсчет гармоники. Отсчеты приведены в дополнительном коде. Для ввода данных в файл инициализации (*.mif) необходтиа 16-ричная система.

Устройство должно вычислить и записать в буферное ОЗУ значения амплитуд гармоник ai, bi для i =1..4. При исследовании результатов вычислений, содержащихся в таблице содержимого ОЗУ (RAM) целесообразна десятичная система счисления.

Обозначения переменных.

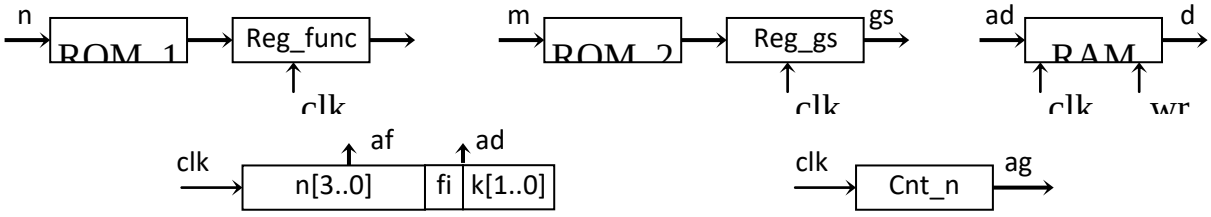
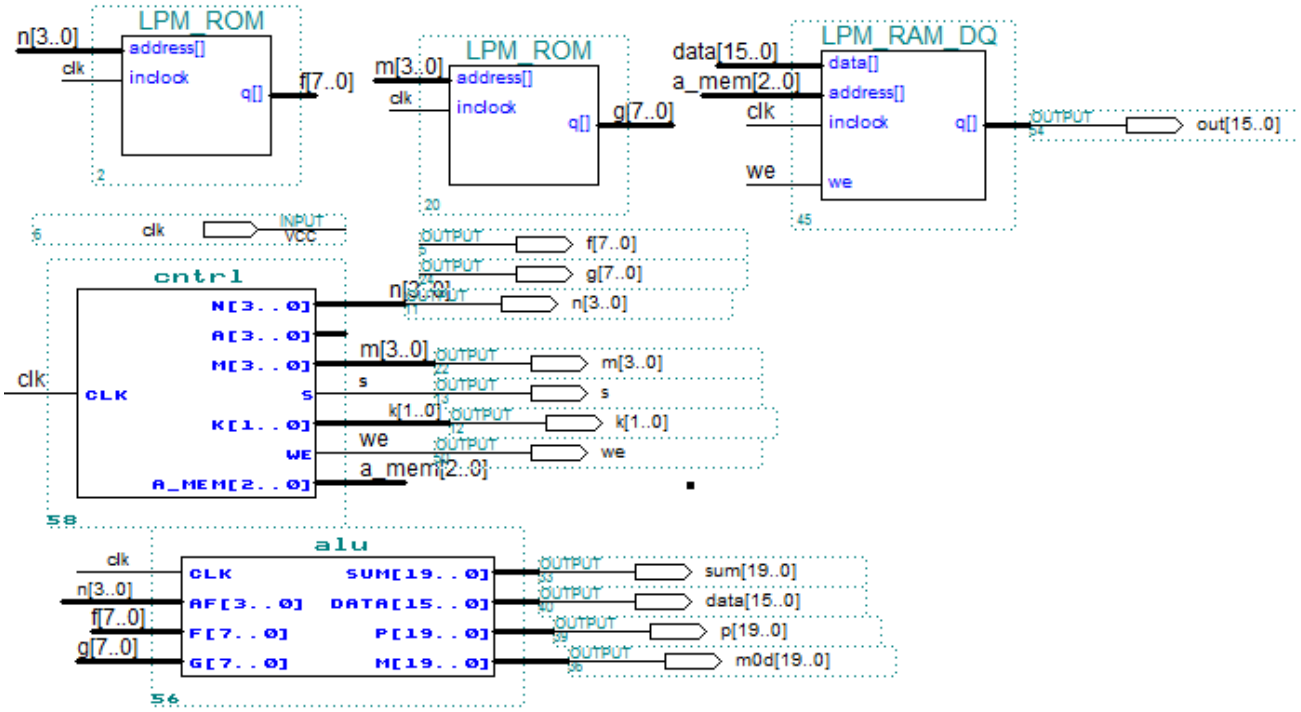


Рис.7.4. Регистровая модель вычислительного устройства

m[3..0] – номер отсчета гармоники для ПЗУ ROM_2;

gs[7..0] – отсчет гармоники с выхода ПЗУ; n[3..0] – номер отсчета функции для ПЗУ ROM_1; func[7..0] - отсчет входной функции; a_mem - адрес ОЗУ для записи результата;

data – входные данные ОЗУ; we - разрешенгие записи в ОЗУ.



Функции устройства управления (модуль control) выполняет счетчик, содержащий несколько полей. Такое включение существенно упрощает выполнение алгоритма, содержащего три вложенных цикла. В поле n, содержащем 4 разряда, формируется адрес отсчета функции, в поле fi содержится бит признака начальной фазы анализируемой гармоники – (0 - синус, 1 - косинус).

Двухразрядное поле k определяет номер гармоники. Код 0, устанавливаемый по умолчанию при включении ПЛИС, должен соответствовать 1-й гармонике; код 1 – 2-й; код 2 – 3-й; код 3 – 4-й гармонике.

В начале моделирования по умолчанию fi = 0, n = 0, будет вычисляться коэффициент b1 (амплитуда первой гармоники функции sin), после переполнения счетчика адреса n установится fi = 1 и бдет вычисляться a1 (амплитуда первой гармоники cos). Впоследствии, при значениях n, равных 1, 2, 3 будут вычисляться коэффициенты b2, a2, b3, a3, b4, a4. Признак fi используется при формировании адреса при обращении к ПЗУ гармоники. При fi = 0 на адресный вход ПЗУ подается адрес, полученный в регистре m, а при fi = 1 к адресу m прибавляется 4. Это ¼ периода, содержащего 16 отсчетов. Таким образом, с выхода ПЗУ будет получена функция cos.

Код, содержащийся в поле k определяет номер гармоники и индексирование счетчика адреса гармоник - прибавление к адресу числа k. В результате для первой гармоники (k = 1) соответствует приращение адреса на 1 шаг, второй гармонике соответствует приращение адреса на 2 шага, и т. д.

В модуле АЛУ вычисляются произведения кодов входной функции и гармоники. Переменная mod - модуль произведения определяется как произведение модулей сомножителей, которые представлены в дополнительном кое. Переменная p - произведение вычисляется в дополнительном коде. Переменна sum – накапливающий сумматор, его разрядность выбрана с учетом суммирования нескольких 16-разряднвх слагаемых. Выходной сигнал АЛУ – data определяется делением суммы (sum) на 16.

При моделировании необходимо выбрать шаг сетки 400 нс. и время моделирования 51 мкс.

```
module alu (c,af,f,g,sum,data,p,m);
input c;
input [3:0] af;
input [7:0] f, g;
output [15:0] data;
output [19:0] sum;
reg [19:0] sum;
output [15:0] p;
output [15:0] m;
assign m =(f[7]? 256-f :f)*(g[7]? 256-g:g);
```

```
module cntrl
(clk,n, a,m, s,k, we,a_mem);
input clk;
output [3:0] n;     reg [3:0] n;
output [3:0] a;     reg [3:0] a;
output [3:0] m;    //reg [3:0] m;
output [2:0] a_mem;
output s, we;            reg s ;
output [1:0] k;          reg [1:0] k;
always @ (negedge clk)
```