УТВЕРЖДЕН

РАЯЖ.00169-01 13 01-ЛУ

МИКРОСХЕМА ИНТЕГРАЛЬНАЯ 1892ВМ10Я

БИБЛИОТЕКА ЦОС

Описание программы

РАЯЖ.00169-01 13 01

CD-R

Листов 25

2011

Литера

Инв. № подл.	Подп. и дата	Взам.инв.№	Инв.№ дубл.	Подп. и дата

АННОТАЦИЯ

В документе «Микросхема интегральная 1892ВМ10Я. Библиотека ЦОС. Описание программы» РАЯЖ.00169-01 13 01 приводится описание разработанных для библиотеки ЦОС (цифровая обработка сигналов) функций:

- спектрального анализа;
- КИХ-фильтрации;
- арифметических и тригонометрических операций над векторами.

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

1.1. Обозначение и наименование программы

РАЯЖ.00169-01

Библиотека ЦОС (цифровая обработка сигналов).

1.2. Программное обеспечение

1.2.1. Для использования библиотеки ЦОС необходимо иметь ПК с установленной средой разработки MultiCore Studio 3M (РАЯЖ.00167-01).

1.3. Языки программирования

1.3.1. Библиотека прикладных программ сигнальной обработки написана на языке С для исполнения в RISC-ядре и на языке ассемблера для исполнения в ядрах ELCORE-09.

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

2.1. Назначение программы

2.1.1. Библиотека прикладных программ цифровой сигнальной обработки предназначена для использования на ИМС 1892ВМ10Я. Набор функций позволяет осуществлять процедуры:

- спектрального анализа;
- КИХ-фильтрации;
- арифметические векторные;
- тригонометрические векторные;
- преобразования форматов векторные.

3. СТРУКТУРА ПРОГРАММЫ

3.1. Функции прикладной библиотеки

3.1.1. Перечень функций:

1) *fir_ff* – действительный КИХ-фильтр в формате float (здесь и далее float – формат с плавающей точкой 24E8 IEEE 754);

2) *fir_ii* – действительный КИХ-фильтр в формате short (здесь и далее short – 16-разрядный целый формат);

3) *fir_ff_dec* – действительный децимирующий КИХ-фильтр в формате float;

4) *fir_ii_dec* – действительный децимирующий КИХ-фильтр в формате short;

5) *fir_ff_int* – действительный интерполирующий КИХ-фильтр в формате float;

6) *fir_ii_int* – действительный интерполирующий КИХ-фильтр в формате short;

7) *fir_ff_slow* – действительный КИХ-фильтр в формате float без использования режима с отключенными блокировками конвейера;

8) *fir_ff_dec_slow* – действительный децимирующий КИХ-фильтр в формате float без использования режима с отключенными блокировками конвейера;

9) *fft* – комплексное прямое быстрое преобразование Фурье (БПФ) в формате float;

10) *fft_fr* – комплексное прямое быстрое преобразование Фурье (БПФ) в формате short;

11) *ifft* – комплексное обратное БПФ в формате float;

12) *ifft_fr* – комплексное обратное БПФ в формате short;

13) *convol* – быстрая свертка комплексного сигнала в формате float;

14) *convol_fr* – быстрая свертка комплексного сигнала в формате short;

15) *add_f* – сложение двух векторов в формате float;

16) *add_s* – сложение двух векторов в формате short;

17) *sub_f* – вычитание двух векторов в формате float;

18) *sub_s* – вычитание двух векторов в формате short;

19) *mlt_f* – умножение двух векторов в формате float;

20) *mlt_s* – умножение двух векторов в формате short;

21) *maxval_f* – поиск максимального элемента вектора в формате float;

22) *maxval_s* – поиск максимального элемента вектора в формате short;

23) *minval_f* – поиск минимального элемента вектора в формате float;

24) *minval_s* – поиск минимального элемента вектора в формате short;

25) *max_index_f* – поиск индекса максимального элемента вектора в формате float;

26) *max_index_s* – поиск индекса максимального элемента вектора в формате short;

27) *min_index_f* – поиск индекса минимального элемента вектора в формате float;

28) *min_index_s* – поиск индекса минимального элемента вектора в формате short;

29) *neg_f* – вектор отрицательных элементов в формате float;

30) *neg_s* – вектор отрицательных элементов в формате short;

31) *neg_i* – вектор отрицательных элементов в формате int (32-разрядное целое);

32) *sqrt_f* – квадратный корень в формате float;

33) *sqrt_s* – квадратный корень в формате short;

34) *sin_f* – синус в формате float;

35) *sin_s* – синус в формате short;

36) *cos_f* – косинус в формате float;

37) *cos_s* – косинус в формате short;

38) *atan_f* – арктангенс в формате float;

39) *atan_s* – арктангенс в формате short;

40) *short_to_float* – преобразование вектора из формата float в формат short;

41) *float_to_short* – преобразование вектора из формата short в формат float.

3.2. КИХ-фильтрация

3.2.1. Для осуществления КИХ-фильтрации в библиотеке имеются следующие функции:

1) fir_ff(float *dst, const float *src, int len, const float *filter, int f_len) – для фильтрации в формате с плавающей точкой;

2) fir_ff_slow(float *dst, const float *src, int len, const float *filter, int f_len) – для фильтрации в формате с плавающей точкой без отключения блокировок конвейера;

3) fir_ff_dec(float *dst, const float *src, int len, const float *filter, int f_len, int dec) – для децимирующей фильтрации в формате с плавающей точкой;

4) fir_ff_dec_slow(float *dst, const float *src, int len, const float *filter, int f_len, int dec)– для децимирующей фильтрации в формате с плавающей точкой без отключения блокировок конвейера;

5) fir_ff_int(float *dst, const float *src, int len, const float *filter, int f_len, int interp) – для фильтрации в формате с плавающей точкой с интерполяцией;

6) fir_ii (short *dst, const short *src, int len, const short *filter, int f_len) – для фильтрации в 16-разрядном целом формате;

7) fir_ii_dec(short *dst, const short *src, int len, const short *filter, int f_len, int dec) – для децимирующей фильтрации в 16-разрядном целом формате;

8) fir_ii_int(short *dst, const short *src, int len, const short *filter, int f_len, int interp) – для фильтрации в 16-разрядном целом формате с интерполяцией.

3.2.2. Описание аргументов:

- dst – указатель на буфер для размещения результата;
- src – указатель на буфер с входными данными;
- len – длина входного вектора;
- filter – указатель на характеристику фильтра;
- f_len – длина характеристики фильтра;
- dbuf – указатель на массив сигнального перекрытия;
- dec – коэффициент децимации (len кратна dec);
- interp – коэффициент интерполяции (f_len кратна interp).

3.2.3. Алгоритм вычисления КИХ-фильтра иллюстрируется схемой на рис. 1.

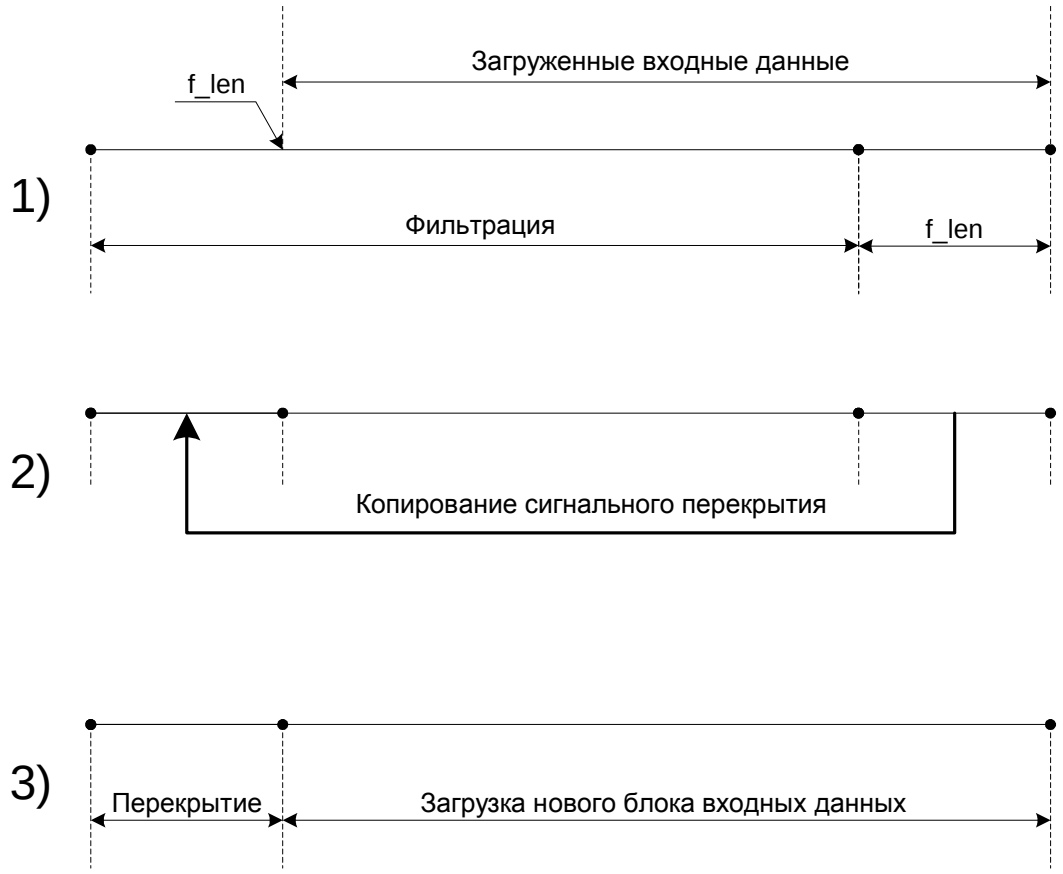

Рисунок 1

3.2.3.1. На первом этапе данные загружаются, начиная с адреса src[f_len], как показано на рис.1. Первые f_len значений буфера src заполнены нулями. Фильтрация же происходит по массиву src[0..len-1]. После фильтрации на этапе 2) последние f_len значений, образующие сигнальное перекрытие, копируются в начало массива src, то есть они будут отфильтрованы на следующем запуске фильтрации. На этапе 3) новый блок данных загружается с адреса src[f_len], продолжая перекрытие, загруженное в начало массива. Таким образом, достигается непрерывная фильтрация поступающих данных с задержкой всего на f_len, то есть на длину характеристики фильтра.

3.2.4. Ограничения при использовании функции:

1) адреса буферов src, dst, d_buf и filter должны быть выровнены в памяти по четыре 32-разрядных слова;

2) буфер сигнального перекрытия d_buf должен быть расположен в памяти непосредственно перед буфером src так, чтобы выполнялось правило &src = &d_buf + f_len;

3) размеры буферов f_len и len должны быть кратны четырем;

4) функции fir_ff и fir_ff_dec используют в своей работе режим с отключением блокировок конвейера. Поэтому на их работу накладываются следующие ограничения:

– во время работы функций все прерывания блокируются;

– все данные должны быть расположены в ближней памяти XYRAM;

– буферы входного сигнала и характеристики фильтра должны быть расположены на разных страницах памяти XYRAM. Например: 0x0000 и 0x1000; 0x1000 и 0x2000; 0x2000 и 0x3000;

5) в библиотеке реализованы версии функций fir_ff и fir_ff_dec без отключения блокировок конвейера под названиями fir_ff_slow и fir_ff_dec_slow соответственно. Скорость их работы в два с половиной раза меньше fir_ff и fir_ff_dec, но на них не распространяются ограничения, связанные с отключениями блокировок.

3.3. Прямое, обратное БПФ, быстрая свертка

3.3.1. Прямое БПФ

3.3.1.1. Комплексный прямой вход, комплексный прямой выход “не на месте”, $N = 4^S$. Формат данных (входных, выходных и поворачивающих коэффициентов) – стандартная плавающая точка (IEEE - 754). Реализация алгоритма состоит из двух функций инициализирующей и главной:

```
void fft_init(float *temp, int n, float *w);
void fft_run(float *in, float *out, int n, float *w);
```

3.3.2. Обратное БПФ

3.3.2.1. Комплексный прямой вход, комплексный прямой выход “не на месте”, $N = 4^S$. Формат данных (входных, выходных и поворачивающих коэффициентов) – стандартная плавающая точка (IEEE - 754). Реализация алгоритма состоит из двух функций инициализирующей и главной:

```
void ifft_init(float *temp, int n, float *w);
void ifft_run(float *in, float *out, int n, float *w);
```

3.3.3. Быстрая свертка комплексного сигнала

3.3.3.1. Формат данных (входных, выходных и поворачивающих коэффициентов) – стандартная плавающая точка (IEEE - 754). Реализация алгоритма состоит из двух функций инициализирующей и главной:

void convol_init(float* temp, int len2, float *w);

void convol(float * dst, float * src1, float * src2, float *temp, int len1, int len2);

3.3.4. Описание параметров, передаваемых в функции

3.3.4.1. В функции БПФ и ОБПФ передаются следующие параметры:

1) in – указатель на входной буфер;

2) out – указатель на выходной буфер;

3) w – указатель на массив поворачивающих коэффициентов;

4) temp – вспомогательный массив, предназначенный для генерации поворачивающих векторов;

5) n – размер преобразования.

3.3.4.2. В функцию быстрой свертки передаются следующие параметры:

1) dst – указатель на выходной буфер;

2) src1 – указатель на первый сворачиваемый сигнал;

3) src2 – указатель на второй сворачиваемый сигнал;

4) temp – указатель на вспомогательный буфер свернутого сигнала;

5) len1 – длина первого сигнала;

6) len2 – длина второго сигнала.

Примечание. len1 кратно len2.

3.3.5. Описание функций БПФ и ОБПФ

3.3.5.1. Функция инициализации генерирует массив поворачивающих векторов с использованием дополнительного массива. В дополнительный массив в начале функции генерируется синусоидальный сигнал с частотой «1», из него впоследствии, по определенному закону, формируется массив поворачивающих векторов.

Перед использованием главной функции необходимо во входной массив передать отсчеты исходного сигнала.

Быстрое преобразование Фурье реализовано с использованием новых векторных команд микропроцессора 1892ВМ10Я, оптимизированных для вычислений с комплексными числами. Векторные команды за один командный цикл выполняют до четырех арифметических операций с числами, представленными в формате с плавающей точкой. В качестве базовой операции используется БПФ размером «4». Такой выбор обусловлен оптимальным соотношением использования вычислительной мощности ядра для сокращения времени вычислений и организацией работы с памятью параллельно с вычислениями.

3.3.6. Описание функций быстрой свертки

3.3.6.1. Функция инициализации генерирует массив поворачивающих векторов с использованием дополнительного массива. В дополнительный массив в начале функции генерируется синусоидальный сигнал с частотой «1», из него впоследствии, по определенному закону, формируется массив поворачивающих векторов.

Перед использованием главной функции необходимо в оба входных массива передать отсчеты исходных сигналов.

Быстрая свертка двух сигналов реализована с использованием новых векторных команд микропроцессора 1892ВМ10Я, оптимизированных для вычислений с комплексными числами. Векторные команды за один командный цикл выполняют до четырех арифметических операций с числами, представленными в формате с плавающей точкой. В качестве базовой операции для вычисления БПФ используется операция «бабочка» размером «4». Такой выбор обусловлен оптимальным соотношением использования вычислительной мощности ядра для сокращения времени вычислений и организацией работы с памятью параллельно с вычислениями.

3.3.7. Распределение памяти для выполнения БПФ

3.3.7.1. Данные входного и выходного массивов в памяти должны располагаться по выровненному начальному адресу. Первые $k = \log_2 N$ разрядов начального адреса должны быть равны «0». Размеры массивов приведены в таблице 1.

Таблица 1 - Размеры массивов для БПФ

Массив	Дополнительный массив	Входной массив	Выходной массив	Массив поворачивающих векторов
Размер в 32-разрядных словах	$2 \cdot N$	$2 \cdot N$	$2 \cdot N$	$\frac{3}{2} \cdot N$

3.3.8. Распределение памяти для выполнения быстрой свертки

3.3.8.1. Данные входных массивов и выходного массива в памяти должны располагаться по выровненному начальному адресу. Первые $k = \log_2 len1$ и $i = \log_2 len2$ разрядов начального адреса должны быть равны «0». Размеры массивов приведены в таблице 2.

Таблица 2 - Размеры массивов для быстрой свертки

Массив	Дополнительный массив temp	Входной массив src1	Входной массив src2	Выходной массив dst	Массив поворачивающих векторов
Размер в 32-разрядных словах	$2 \cdot len2$	$2 \cdot len1$	$4 \cdot len2$	$2 \cdot len1$	$\frac{3}{2} \cdot len2$

3.3.9. Ограничения при использовании функции

3.3.9.1. Существуют следующие ограничения:

1) размеры буферов in и out должны быть степенью «4»;

2) все функции прямого и обратного БПФ используют в своей работе режим с отключением блокировок конвейера. Поэтому на их работу накладываются ограничения:

– во время работы функций все прерывания блокируются;

– все данные должны быть расположены в ближней памяти XYRAM;

– буферы входного сигнала и поворачивающих коэффициентов должны быть расположены на разных страницах памяти XYRAM. Например: 0x0000 и 0x1000; 0x1000 и 0x2000; 0x2000 и 0x3000.

3.3.10. Назначение функций

3.3.10.1. Для ускорения циклической обработки нескольких массивов входных данных каждый из алгоритмов БПФ быстрой свертки, входящих в библиотеку, состоит из двух функций.

Первая функция – инициализирующая. Функция запускает DSP на исполнение инициализирующей части программы DSP:

- генерация вспомогательного массива синусоидального сигнала;

- генерация поворачивающих векторов для выполнения преобразования.

Вторая функция – главная. В ней осуществляется запуск DSP на выполнение программы БПФ или ОБПФ.

3.3.11. Характеристики быстродействия функций БПФ

3.3.11.1. В таблице 3 приведены характеристики быстродействия функций БПФ

Таблица 3

Размер БПФ в элементах	64	256	1024
Тактов DSP на выполнение	484	2336	11244

3.4. Векторные операции

3.4.1. Вычитание векторов

3.4.1.1. Вычитание векторов в формате с плавающей точкой:

*sub (float *x, float *y, float *r, int len)*

3.4.1.2. Вычитание векторов (16-разрядное):

*sub (short * x, short *y, short *r, int len, int scale)*

Описание: for (i = 0; i < len; i++) r(i) = x(i) – y(i)

Аргументы:

– *x,y* – указатели на входные вектора; x – вектор уменьшаемых, y – вектор вычитаемых;

– *r* – указатель на вектор результата;

– *len* – длина векторов;

– *scale* – флаг масштабирования; scale может принимать значения “0”…“3”, соответственно масштабирование будет происходить на *scale* разрядов.

Обработка переполнений: при установленном флаге scale производится масштабирование на один, два или три разряда.

Ограничения: длина векторов должна быть кратна четырем для формата float и восьми для формата short.

3.4.2. Сложение векторов

3.4.2.1. Сложение векторов в формате с плавающей точкой:

*add (float *x, float *y, float *r, int len)*

3.4.2.2. Сложение векторов (16-разрядное):

*add (short *x, short *y, short *r, int len, int scale)*

Описание: for (i = 0; i < len; i++) r(i) = x(i) + y(i)

Аргументы:

– *x,y* – указатели на входные вектора;

– *r* – указатель на вектор результата;

– *len* – длина векторов;

– *scale* – флаг масштабирования; scale может принимать значения “0”...“3”, соответственно масштабирование будет происходить на *scale* разрядов.

Обработка переполнений: при установленном флаге scale производится масштабирование на один, два или три разряда.

Ограничения: длина векторов должна быть кратна четырем для формата float и восьми для формата short.

3.4.3. Умножение векторов

3.4.3.1. Поэлементное умножение векторов в формате с плавающей точкой:

*mlt (float *x, float *y, float *r, int len)*

3.4.3.2. Поэлементное умножение векторов (16-разрядное):

*mlt (short *x, short *y, short *r, int len, int scale)*

Описание: for (i = 0; i < len; i++) r(i) = x(i) * y(i)

Аргументы:

– *x*,*y* – указатели на входные вектора;

– *r* – указатель на вектор результата;

– *len* – длина векторов;

– *scale* – флаг масштабирования; scale может принимать значения “0”...“3”, соответственно масштабирование будет происходить на *scale* разрядов.

Обработка переполнений: при установленном флаге scale производится масштабирование на один, два или три разряда.

Обработка переполнений: при не равном нулю scale производится масштабирование на scale разрядов.

Ограничения: длина векторов должна быть кратна четырем.

3.4.4. Поиск максимального/ минимального элемента

3.4.4.1. Поиск максимального элемента в формате с плавающей точкой:

*float r = maxval (float *x, int len)*

3.4.4.2. Поиск максимального элемента (16-разрядный):

*short r = maxval (short *x, int len)*

3.4.4.3. Поиск минимального элемента в формате с плавающей точкой:

*float r = minval (float *x, int len)*

3.4.4.4. Поиск минимального элемента (16-разрядный):

*short r = minval (short *x, int len)*

3.4.4.5. Поиск индекса максимального элемента в формате с плавающей точкой:

*int ind = max_ index (float *x, int len)*

3.4.4.6. Поиск индекса максимального элемента (16-разрядный):

*int ind = max_index (short *x, int len)*

3.4.4.7. Поиск индекса минимального элемента в формате с плавающей точкой:

*min_index (float *x, int len)*

3.4.4.8. Поиск индекса минимального элемента (16-разрядный):

*int ind = min_index (short *x, int len)*

Аргументы:

– *x* – указатель на входной вектор;
– *len* – длина вектора.

Ограничения: длина векторов для операций поиска минимума/максимума должна быть кратна восьми для формата float. Длина векторов для операций поиска индекса минимума/максимума в формате short должна быть кратна 16.

3.4.5. Вектор отрицательных элементов

3.4.5.1. Вектор отрицательных элементов в формате с плавающей точкой:

*neg (float *x, float *r, int len)*

3.4.5.2. Вектор отрицательных элементов (16-разрядный):

*neg (short *x, short *r, int len)*

3.4.5.3. Вектор отрицательных элементов (32-разрядный):

*int oflag = neg (int32_t *x, int32_t *r, int len)*

Описание: for (i = 0; i < len; i++) r(i) = -x(i)

Аргументы:

– *x* – указатель на входной вектор;
– *r* – указатель на вектор результата;
– *len* – длина векторов.

Ограничения: длина векторов должна быть кратна четырем для форматов float и int; длина должна быть кратна восьми для формата short.

3.4.6. Квадратный корень

3.4.6.1. Квадратный корень в формате с плавающей точкой:

*sqrt (float *x, float *r, int len)*

3.4.6.2. Квадратный корень (16-разрядный):

*sqrt (short *x, short *r, int len)*

Описание: for (i = 0; i < len; i++) r(i) = sqrt(x(i))

Аргументы:

– *x* – указатель на входной вектор;
– *r* – указатель на вектор результата;
– *len* – длина векторов.

Корень вычисляется с округлением.

Ограничения: длина векторов должна быть кратна четырем для формата short.

3.4.7. Синус

3.4.7.1. Синус в формате с плавающей точкой:

*sin (float *x, float *r, int len)*

3.4.7.2. Синус (16-разрядный):

*sin (short *x, short *r, int len)*

Описание: for (i = 0; i < len; i++) r(i) = sin(x(i))

Аргументы:

– *x* – указатель на входной вектор;
– *r* – указатель на вектор результата;
– *len* – длина векторов.

Ограничения: длина векторов должна быть кратна четырем для формата short.

3.4.8. Косинус

3.4.8.1. Косинус в формате с плавающей точкой:

*cos (float *x, float *r, int len)*

3.4.8.2. Косинус (16-разрядный):

*cos (short *x, short *r, int len)*

Описание: for (i = 0; i < len; i++) r(i) = cos(x(i))

Аргументы:

– *x* – указатель на входной вектор;
– *r* – указатель на вектор результата;
– *len* – длина векторов.

Ограничения: длина векторов должна быть кратна четырем для формата short.

3.4.9. Арктангенс

3.4.9.1. Арктангенс в формате с плавающей точкой:

*arc (float *x, float *r, int len)*

3.4.9.2. Арктангенс (16-разрядный):

*arc (short *x, short *r, int len)*

Описание: for (i = 0; i < len; i++) r(i) = atan(x(i))

Аргументы:

– *x* – указатель на входной вектор;
– *r* – указатель на вектор результата;
– *len* – длина векторов.

Ограничения: длина векторов должна быть кратна четырем для формата short.

3.4.10. Преобразование типа

3.4.10.1. Из 16-разрядного в формат с плавающей точкой:

*short_to_float (short *x, float *r, int len)*

3.4.10.2. Из формата с плавающей точкой в 16-разрядный:

*float_to_short(float *x, short *r, int len)*

Аргументы:

– *x* – указатель на входной вектор;

– *r* – указатель на вектор результата;

– *len* – длина векторов.

Обработка переполнений: возврат максимального числа при переполнении.

Ограничения: длина векторов должна быть кратна двум для функции short_to_float и четырем для функции float_to_short.

4. ОБРАЩЕНИЕ К ПРОГРАММЕ

4.1. Оформление функций

4.1.1. Все библиотечные функции – это Си-функции, осуществляющие вызов соответствующих DSP-функций, передачу им аргументов, ожидание возврата из DSP-программы и по необходимости – возврат результата. Функции не привязаны к тому или иному ядру DSP.

4.2. Входные, выходные и промежуточные данные

4.2.1. Все адреса – байтовые. При использовании функций подразумевается, что все данные уже загружены в ближнюю XY-память DSP.

На размещение данных в DSP для всех функций накладываются следующие ограничения:

– адреса должны быть выровнены по границе слова;

– размеры всех буферов должны быть четными.

Кроме того, на работу некоторых функций накладываются дополнительные ограничения, приведенные выше в описании этих функций.

4.3. Размещение программы DSP

4.3.1. DSP-часть функций может быть размещена в программной памяти в любом из двух имеющихся ядер. Поскольку в «Навикоме» программная память для ядер DSP раздельная, внутри функции номер ядра определяется по адресу одной из меток программы DSP.

4.4. Передача аргументов программе DSP и возврат

4.4.1. Передача аргументов программе DSP осуществляется через четные 32-разрядные регистры, начиная с R0 (т.е. R0.L, R2.L, R4.L, R6.L и т.д.) Указатели на буферы передаются через регистры A0, A1, A2 и т.д.

Если программа DSP должна возвращать результат, он возвращается в вызывающую Си-функцию через регистр R0.L.

Ограничений на использование регистров внутри программы нет.

4.5. Вызов функции

4.5.1. Для вызова библиотечной функции необходимо выполнить следующие действия:

1) загрузить входные данные в ближнюю XY-память DSP;

2) вызвать Си-функцию, передав все необходимые входные данные.

Пример.

*fir_ff(0xb8400400, (0xb8400000+32*4), 256, 0xb8408000, 32, 0xb8400000);*

Здесь:

- dst – адрес буфера для сохранения результатов (0xb8400400);
- src – адрес входного буфера (0xb8400000 + 32*4);
- len – длина входного буфера (256);
- filter – адрес буфера с характеристикой фильтра (0xb8408000);
- f_len – длина характеристики фильтра (32);
- d_buf – буфер сигнального перекрытия (0xb8400000).

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

БИХ – бесконечная импульсная характеристика.

БПФ – быстрое преобразование Фурье.

КИХ – конечная импульсная характеристика.

МП – микропроцессор.

ОБПФ – обратное быстрое преобразование Фурье.

ПК – персональный компьютер.

СБИС – сверхбольшая интегральная схема.

ЦОС – цифровая обработка сигналов.

DSP – Digital Signal Processor.

RISC – Reduced Instruction Set Computer.

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	№ документа	Входящий № сопроводи-тельного документа и дата	Подпись	Дата
	изме-ненных	заменен-ных	новых	аннули-рован-ных					