



Библиотеки для NM6403

Библиотека БПФ Руководство программиста

Версия 1.0



Module® и **NeuroMatrix®** являются зарегистрированными торговыми марками ЗАО НТЦ “Модуль”. Все другие торговые марки являются исключительной собственностью их соответствующих владельцев.

ПРЕДИСЛОВИЕ	1
О СПРАВОЧНОМ РУКОВОДСТВЕ	1
КАК ОРГАНИЗОВАН ДАННЫЙ ДОКУМЕНТ	1
БИБЛИОТЕКА ФУНКЦИЙ ПРЯМОГО И ОБРАТНОГО БПФ	1-1
1.1 ОБЩАЯ ХАРАКТЕРИСТИКА ФУНКЦИЙ БПФ	1-1
1.2 ФУНКЦИИ УПРАВЛЕНИЯ ТОЧНОСТЬЮ БПФ	1-3
1.3 ОПИСАНИЕ ФУНКЦИЙ БИБЛИОТЕКИ БПФ	1-4
1.3.1 NMFFT256	1-4
1.3.2 NMIFFT256	1-5
1.3.3 NMFFT256Set	1-6
1.3.4 NMIFFT256Set	1-7
1.3.5 NMFFT512	1-8
1.3.6 NMIFFT512	1-9
1.3.7 NMFFT512Set	1-10
1.3.8 NMIFFT512Set	1-11
1.3.9 NMFFT1024	1-12
1.3.10 NMIFFT1024	1-13
1.3.11 NMFFT1024Set	1-14
1.3.12 NMIFFT1024Set	1-15
1.3.13 NMFFT2048	1-16
1.3.14 NMIFFT2048	1-17
1.3.15 NMFFT2048Set	1-18
1.3.16 NMIFFT2048Set	1-19
РАЗРАБОТКА ПРИЛОЖЕНИЙ НА ОСНОВЕ БИБЛИОТЕКИ БПФ	2-1
2.1 ПОДКЛЮЧЕНИЕ БИБЛИОТЕКИ БПФ	2-1
2.2 ПРИМЕР СБОРКИ ПРОЕКТА	2-2
ПРИЛОЖЕНИЕ	3-1
3.1 РЕАЛИЗАЦИЯ ПРЯМОГО БПФ-256	3-1
3.1.1 Алгоритм вычисления БПФ-256 на процессоре NM6403	3-1
3.1.2 Схема вычислений в функции NMFFT256	3-2
3.2 РЕАЛИЗАЦИЯ ПРЯМОГО БПФ-512	3-3
3.2.1 Алгоритм вычисления БПФ-512 на процессоре NM6403	3-3
3.2.2 Схема вычислений в функции NMFFT512	3-5
3.3 РЕАЛИЗАЦИЯ ПРЯМОГО БПФ-1024	3-7
3.3.1 Алгоритм вычисления БПФ-1024 на процессоре NM6403	3-7
3.3.2 Схема вычислений в функции NMFFT1024	3-9
3.4 РЕАЛИЗАЦИИ ПРЯМОГО БПФ-2048	3-10

Оглавление

3.4.1 Алгоритм вычисления БПФ-2048 на процессоре NM6403	3-10
3.4.2 Схема вычислений в функции NMFFT2048	3-12
3.5 РЕАЛИЗАЦИЯ ОБРАТНОГО БПФ-256	3-12
3.5.1 Алгоритм вычисления обратного БПФ-256 на процессоре NM6403	3-12
3.5.2 Схема вычислений в функции NMIFFT256	3-13
3.6 РЕАЛИЗАЦИЯ ОБРАТНОГО БПФ-512	3-14
3.6.1 Алгоритм вычисления обратного БПФ-512 на процессоре NM6403	3-14
3.6.2 Схема вычислений в функции NMIFFT512	3-14
3.7 РЕАЛИЗАЦИЯ ОБРАТНОГО БПФ-1024 НА ПРОЦЕССОРЕ NM6403	3-15
3.7.1 Алгоритм вычисления обратного БПФ-1024 для процессора NM6403	3-15
3.7.2 Схема вычислений в функции NMIFFT1024	3-16
3.8 РЕАЛИЗАЦИЯ ОБРАТНОГО БПФ-2048	3-16
3.8.1 Алгоритм вычисления обратного БПФ-2048 на процессоре NM6403	3-16
3.8.2 Схема вычислений в функции NMIFFT1024	3-16

В предисловии описывается назначение и состав документа, приводится краткий обзор разделов и глав, определяется стиль и символьные нотации, используемые в документе.

О справочном руководстве

Данное руководство содержит описание библиотеки функций, используемых для выполнения прямого и обратного быстрого преобразования Фурье. Набор функций оптимизирован для конфигурации модуля МЦ 4.01 и архитектуры процессора NeuroMatrix® NM6403.

Как организован данный документ

Документ разделен на три главы, каждая из которых описывает следующие вопросы:

- | | |
|----------------|---|
| Глава 1 | Библиотека функций прямого и обратного БПФ
Содержит описание функций библиотеки БПФ, предназначенных для выполнения на процессоре NM6403. |
| Глава 2 | Разработка приложений на основе библиотеки БПФ
Содержит пример сборки программы с вызовом функции БПФ на процессоре NM6403 |
| Глава 3 | Приложение
Содержит алгоритмы и структуры вычислений функций БПФ |

Соглашения о нотациях

В данном справочном руководстве используются следующие типографические нотации:

- | | |
|----------------------|--|
| <code>courier</code> | так помечается текст, который может быть набран пользователем с клавиатуры: исходные тексты на языках Си++ и ассемблера. |
|----------------------|--|

<i>Courier</i>	отмечает текст, который должен быть заменен пользовательской информацией, например, реальным значением константы.
Текст или <u>Текст</u>	Так помечается текст, на который необходимо обратить особое внимание.
<i>//Текст</i>	так помечаются комментарии к программам.

Примечание

Данное примечание представляет собой пример того, как оформлены все важные замечания и комментарии, возникающие по ходу описания.

Как оформлять замечания и предложения

По документации

Если при работе с документацией, описывающей процессор и его базовое ПО, у Вас возникли сложности, например: Вы считаете, что материал изложен недостаточно подробно для понимания, пожалуйста, присылайте свои замечания в следующем виде:

- название документа;
- персональный номер документа и номер версии документации;
- номера страниц, на которые приводится ссылка;
- краткое описание замечания.

По работе компонент БПО

Если при работе с какой либо из компонент БПО у Вас возникли сложности, пожалуйста, присылайте свои замечания в следующем виде:

- номер версии базового ПО процессора NM6403, которое Вы используете;
- небольшой фрагмент исходного кода, на котором проявляется ошибка;
- краткое описание ее проявления и возможные предположения о причинах ее возникновения, если таковые имеются.

Данная глава содержит описание библиотеки БПФ. Приводится формат вызовов функций, назначение параметров, а также время выполнения функций, измеренных на цифровом модуле МЦ 4.01 .

Библиотека функций быстрого преобразования Фурье разработана и оптимизирована на языке ассемблера нейропроцессора NM6403. Функции выполняют прямое и обратное дискретное преобразование Фурье над массивами целых комплексных чисел. Функции позволяют обрабатывать массивы длиной 256,512,1024 и 2048 элементов. Разрядность входных и выходных данных выровнена до 32 бит.

1.1 Общая характеристика Функций БПФ

Наименования функций

Библиотека БПФ включает в себя следующие функции прямого и обратного преобразования Фурье:

Длина массива	Функции прямого БПФ	Функции обратного БПФ	Алгоритм БПФ (Комбинация оснований)
256	NMFFT256	NMIFFT256	16-16
512	NMFFT512	NMIFFT512	2-16-16
1024	NMFFT1024	NMIFFT1024	32-32
2048	NMFFT2048	NMIFFT2048	2-32-32

Также библиотека включает набор дополнительных функций для управления точностью вычислений, назначение которых будет описано ниже.

Вычисления

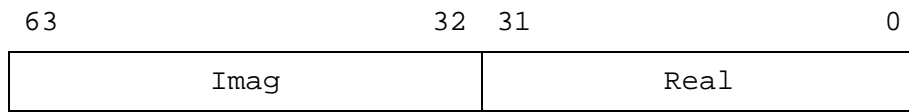
Приведенные выше функции построены на базе алгоритмов БПФ с основанием 16, 32 или с комбинацией оснований: 2,16,32 (алгоритмы вычислений БПФ описаны в приложении). Все вычисления используют целочисленную арифметику (арифметику с фиксированной точкой).

Входные и выходные данные

Входные и выходные данные передаются через массив комплексных чисел типа `CmplxInt` :

```
struct CmplxInt{
    int re;    // Real part;
    int im;    // Imag part;
};
```

В памяти структура `CmplxInt` занимает одно 64-разрядное слово. Старшие 32 бита отводятся под мнимую часть, а младшие 32 – под действительную:



Диапазон допустимых входных значений мнимых и реальных частей массива приведен в **Табл. 1**

На выходе данные располагаются в правильном порядке и по умолчанию являются нормализованными.

Примечание

При объявлении переменных и массивов типа `CmplxInt` следует обратить внимание на то, чтобы они начинались обязательно с четного адреса. Для этого перед объявлением массива можно выделить переменную типа `long`, либо объявлять массив типа `long`, а затем использовать преобразование типов.

Производительность

Ниже приводятся допустимые диапазоны входных данных и временные характеристики функций БПФ, полученные на цифровом модуле МЦ 4.01. Приведенные характеристики дополнительно зависят от выбранной точности вычислений (6 или 7 бит на единицу, подробнее см. п.1.2) и количества этапов нормализации (арифметического сдвига вправо) внутри функций, служащих для предотвращения переполнения в ходе вычислений. Реализованные на данный момент функции прямого БПФ используют один этап нормализации (в конце вычислений), а функции обратного БПФ – два этапа (одна нормализации производится в середине, а вторая в конце вычислений). Промежуточная нормализация необходима в силу более широкого диапазона выходных данных, полученных после прямого БПФ. Однако, при необходимости сравнительно легко, изменяя код функций, можно либо добавить, либо исключить этапы нормализации. С помощью параметров функций также имеется дополнительная возможность менять коэффициент нормализации.

Табл. 1 Производительность функций прямого и обратного БПФ на процессоре NM6403.

Кол-во комплекс отсчетов	Без нормализации			С одной нормализацией (Прямое БПФ)			С двумя нормализациями (Обратное БПФ)		
	Тактов	Время мс	Диапазон вх.данных	Тактов	Время, мс	Диапазон вх.данных	Тактов	Время, мс	Диапазон вх.данных
256	3662	0.092	±512 (7bit) ±2047(6bit)	3994	0.1	±512 (7bit) ±2048 (6bit)	4300	0.11	±2 ¹⁸ (7bit) ±2 ¹⁸ (6bit)
512	8180	0.2	±256 (7bit) ±1023(6bit)	8766	0.22	±256 (7bit) ±1023 (6bit)	9350	0.24	±2 ¹⁸ (7bit) ±2 ¹⁸ (6bit)
1024	18900	0.47	±128 (7bit) ±511 (6bit)	20041	0.5	±128 (7bit) ±511 (6bit)	21200	0.53	±2 ¹⁷ (7bit) ±2 ¹⁷ (6bit)
2048	47624	1.2	±64 (7bit) ±255 (6bit)	49800	1.25	±64 (7bit) ±255 (6bit)	52000	1.3	±2 ¹⁷ (7bit) ±2 ¹⁷ (6bit)

NM6403 cycle time=25 ns (40MHz)

1.2 Функции управления точностью БПФ

В состав библиотеки введены 8 функций для управления точностью вычислений. Эти функции являются необязательными и требуются для изменения точности вычислений в ходе программы. Всего имеется два возможных варианта. По умолчанию (без вызова функций установки точности) функции БПФ производят вычисления по второму варианту.

- В первом варианте для представления коэффициентов преобразования (синусов и косинусов) в 8-разрядной сетке используется условно 6 бит на единицу. Т.е. перевод значений синусов и косинусов в числа с фиксированной точкой осуществляется с помощью масштабирующего множителя- 64.0 по формуле: $W = \text{round}(64.0 * \cos(x))$.
- Во втором способе для этих целей используется условно 7 бит на единицу, а перевод значений синусов и косинусов в числа с фиксированной точкой осуществляется с помощью масштабирующего множителя- 127.0 по формуле: $W = \text{round}(127.0 * \cos(x))$.

Первый способ имеет 65 градаций косинуса в диапазоне 0..1, в то время как второй способ – 128 и следовательно обладает более высокой точностью. Однако, выполняя в ходе вычислений только операции целочисленного умножения и сложения, в конце необходимо провести нормализацию результатов, т.е. каждый элемент выходного массива требуется поделить на соответствующий масштабирующий коэффициент. Для первого случая он равен 64^2 , а для второго - 127^2 . В обоих случаях деление заменяется сдвигом вправо, но в отличие от первого способа замена деления на 127^2 сдвигом на 14 бит вправо привносит небольшую систематическую ошибку.

С помощью функций управления точностью можно выбрать нужную схему вычислений для каждой отдельной функции преобразования Фурье (прямого и обратного).

Длина массива	Функции установки точности для прямого БПФ	Функции установки точности для обратного БПФ
256	NMFFT256Set	NMIFFT256Set
512	NMFFT512Set	NMIFFT512Set
1024	NMFFT1024Set	NMIFFT1024Set
2048	NMFFT2048Set	NMIFFT2048Set

1.3 Описание Функций библиотеки БПФ

1.3.1 NMFFT256

Синтаксис:

```
#include "nmfft.h"
int NMFFT256(
    CmplxInt*   GSrcBuffer, // Source buffer :long[256]
    CmplxInt*   LDstBuffer, // Result FFT   :long[256]
    __int64*    LBuffer,    // Temp buffer  :long[256*3]
    __int64*    GBuffer,    // Temp buffer  :long[256*2]
    const int   ShiftR=-1  // Shift normalization
```

Описание: Функция NMFFT256 выполняет прямое 256-точечное быстрое преобразование Фурье.

CmplxInt* GSrcBuffer
Указатель на исходный массив 256 комплексных чисел. (long[256]).
Диапазон входных данных приведен в **Табл. 1**

CmplxInt *LDstBuffer
Указатель на результирующий массив 256 комплексных чисел.
(long[256]).

__int64* LBuffer
Временный буфер размером long[3*256] для хранения промежуточных вычислений на локальной шине.

__int64* GBuffer
Временный буфер размером long[2*256] для хранения промежуточных вычислений на глобальной шине.

int ShiftR=-1
Коэффициент нормализации, выполняет арифметический сдвиг на ShiftR бит вправо результирующего массива для получения нормализованного массива LDstBuffer.
По умолчанию ShiftR автоматически принимается равным 14 при установленной точности 7-бит или 12 – при точности 6-бит, что соответствует результатам вычислений БПФ с плавающей точкой. (Механизм нормализации более подробно описан в приложении). Точность устанавливается функцией NMFFT256Set. Если эта функция не вызывалась, то по умолчанию принимается 7-битная точность.

Возвращает Функция всегда возвращает 0. (Reserved)

Пример:

```
#include "nmfft.h"
extern CmplxInt   GSrc[];    // Source array
extern CmplxInt   LDst[];    // Result array
extern __int64    LBuffer[]; // Temp array
extern __int64    GBuffer[]; // Temp array
void main ()
{
    NMFFT256(GSrc, LDst, LBuffer, GBuffer);
}
```

Производительность Производительность функции определяется в зависимости от расположения массивов на локальной и глобальной шине.

Максимальное быстродействие достигается при следующей конфигурации:
GSrcBuffer: Global SRAM
LDstBuffer: Local SRAM
LBuffer: Local SRAM
GBuffer: Global SRAM

См. также: NMIFFT256 NMFFT256Set

1.3.2 NMIFFT256

Синтаксис

```
#include "nmfft.h"
int NMIFFT256(
    CmplxInt*  GSrcBuffer, // Source buffer :long[256]
    CmplxInt*  GDstBuffer, // Result FFT   :long[256]
    __int64*   LBuffer,    // Temp buffer  :long[256*3]
    __int64*   GBuffer,    // Temp buffer  :long[256*3]
    int        ShiftR1=8,  // First shift normalization
    int        ShiftR2=-1 // Final shift normalization
```

Описание: Функция NMIFFT256 выполняет обратное 256-точечное быстрое преобразование Фурье.

void* GSrcBuffer
Указатель на исходный массив 256 комплексных чисел. (long[256]).
Диапазон входных данных приведен в **Табл. 1**

void *GDstBuffer
Указатель на результирующий массив 256 комплексных чисел. (long[256]).

void* LBuffer
Временный буфер размером long[256*3] для хранения промежуточных вычислений на локальной шине.

void* GBuffer
Временный буфер размером long[256*3] для хранения промежуточных вычислений на глобальной шине.

int ShiftR1=8
Промежуточный сдвиг результатов на ShiftR1 бит вправо (первая нормализация). По умолчанию равен 8.

int ShiftR2=-1
Заключительный сдвиг результатов на ShiftR2 бит вправо (вторая нормализация) в конце вычисления обратного БПФ. По умолчанию ShiftR2 принимается равным 14 при установленной точности 7-бит или 12 - при точности 6-бит, что соответствует результатам вычислений БПФ с плавающей точкой. (Механизм нормализации более подробно описан в приложении) Точность устанавливается функцией NMIFFT256Set. Если эта функция не вызывалась, то по умолчанию принимается 7-битная точность.

Возвращает Функция всегда возвращает 0. (Reserved)

Пример:

```
#include "nmFFT.h"
extern CmplxInt  GSrc[];    // Source Array
extern CmplxInt  GDst[];   // Result Array
extern __int64   GBuffer[]; // Temp buffer
extern __int64   LBuffer[]; // Temp buffer
void main ()
{
    NMIFFT256(GSrc, GDst, LBuffer, GBuffer);
}
```

Производительность. Производительность функции определяется в зависимости от расположения массивов на локальной и глобальной шине.
Максимальное быстродействие достигается при следующей конфигурации:
GSrcBuffer: Global SRAM
GDstBuffer: Global SRAM
LBuffer: Local SRAM
GBuffer: Global SRAM

См. также NMFFT256, NMIFFT256Set

1.3.3 NMFFT256Set

Синтаксис `#include "nmfft.h"`

`int NMFFT256Set(int Flag) // Type of precision`

Описание: Функция NMFFT256 устанавливает 6-битную или 7-битную точность вычислений для функции NMFFT256.

`int Flag`

Через параметр Flag передается тип устанавливаемой точности с помощью констант **FFT6BIT** или **FFT7BIT**. Установленная точность остается действующей до следующего вызова функции NMFFT256Set. По умолчанию установлена 7-битная точность. Т.е. если до вызова NMFFT256 установка точности функцией NMFFT256Set явно не проводилась, то вычисления будут проходить с 7-битной точностью.

FFT6BIT- устанавливает 6-битную точность вычислений. Коэффициенты Фурье преобразования задаются с помощью масштабирующего множителя 64.0 по формуле:

`W.re=round(64.0*cos(x))`

`W.im=round(64.0*sin(x))`

FFT7BIT- устанавливает 7-битную точность вычислений. Коэффициенты Фурье преобразования задаются с помощью масштабирующего множителя 127.0 по формуле:

`W.re=round(127.0*cos(x))`

`W.im=round(127.0*sin(x))`

Данный масштабирующий множитель следует учитывать при явном задании коэффициента нормализации ShiftR функции NMFFT256.

Возвращает Функция всегда возвращает 0. (Reserved)

Пример:

```
#include "nmfft.h"
extern CmplxInt  LDst[];    // Source array
extern CmplxInt  GSrc[];   // Result array
extern __int64   LBuffer[]; // Temp array
extern __int64   GBuffer[]; // Temp array
void main ()
{
    NMFFT256Set(FFT7BIT);
    NMFFT256(GSrc, LDst, LBuffer, GBuffer);
}
```

См. также NMFFT256

1.3.4 NMIFFT256Set

Синтаксис `#include "nmfft.h"`

`int NMIFFT256Set(int Flag) // Type of precision`

Описание: Функция NMIFFT256 устанавливает 6-битную или 7-битную точность вычислений для функции NMFFT256.

`int Flag`

Через параметр Flag передается тип устанавливаемой точности с помощью констант **FFT6BIT** или **FFT7BIT**. Установленная точность остается действующей до следующего вызова функции NMFFT256Set. По умолчанию устанавливается 7-битная точность. Т.е. если до вызова NMFFT256 установка точности функцией NMFFT256Set явно не проводилась, то вычисления будут проходить с 7-битной точностью.

FFT6BIT- устанавливает 6-битную точность вычислений. Коэффициенты Фурье преобразования задаются с помощью масштабирующего множителя - 64.0 по формуле:

`W.re=round(64.0*cos(x))`

`W.im=round(64.0*sin(x))`

FFT7BIT- устанавливает 7-битную точность вычислений. Коэффициенты Фурье преобразования задаются с помощью масштабирующего множителя - 127.0 по формуле:

`W.re=round(127.0*cos(x))`

`W.im=round(127.0*sin(x))`

Значение масштабирующего множителя следует учитывать при явном задании коэффициентов нормализации ShiftR1 и ShiftR2 функции NMIFFT256.

Возвращает Функция всегда возвращает 0. (Reserved)

Пример:

```
#include "nmFFT.h"
extern CmplxInt  GSrc[];    // Source Array
extern CmplxInt  GDst[];   // Result Array
extern __int64   GBuffer[]; // Temp buffer
extern __int64   LBuffer[]; // Temp buffer
void main ()
{
    NMFFT256Set(FFT6BIT);
    NMIFFT256(GSrc, GDst, LBuffer, GBuffer);
}
```

См. также NMIFFT256

1.3.5 NMFFT512

Синтаксис:

```
#include "nmfft.h"
int NMFFT512(
    CmplxInt  GSrcBuffer, // Source buffer :long[512]
    CmplxInt  GDstBuffer, // Result FFT   :long[512]
    __int64*  LBuffer,    // Temp buffer  :long[512*3]
    __int64*  GBuffer,    // Temp buffer  :long[512*3]
    int       ShiftR=-1) // Right shift normalization
```

Описание: Функция NMFFT512 выполняет прямое 512-точечное быстрое преобразование Фурье.

CmplxInt* GSrcBuffer
Указатель на исходный массив 512 комплексных чисел. (long[512]).
Диапазон входных данных приведен в **Табл. 1**

CmplxInt *GDstBuffer
Указатель на результирующий массив 512 комплексных чисел.
(long[512]).

__int64* LBuffer
Временный буфер размером long[512*3] для хранения промежуточных вычислений на локальной шине.

__int64* GBuffer
Временный буфер размером long[512*3] для хранения промежуточных вычислений на глобальной шине.

int ShiftR=-1
Коэффициент нормализации. Выполняет арифметический сдвиг на ShiftR бит вправо результирующего массива для получения нормализованного массива GDstBuffer.
По умолчанию принимается равным 14 при установленной точности 7-бит или 12 - при точности 6-бит, что соответствует результатам вычислений БПФ с плавающей точкой. (Механизм нормализации более подробно описан в приложении). Точность устанавливается функцией NMFFT512Set, если эта функция не вызывалась, то по умолчанию принимается 7-битная точность.

Возвращает Функция всегда возвращает 0. (Reserved).

Пример:

```
#include "nmfft.h"
extern ComplexInt  GSrc[];    // Source array
extern ComplexInt  GDst[];    // Result array
extern __int64     LBuffer[]; // Temp array
extern __int64     GBuffer[]; // Temp array
void main ()
{
    NMFFT512Set(FFT6BIT);
    NMFFT512(GSrc, GDst, LBuffer, GBuffer);
}
```

Производительность Производительность функции определяется в зависимости от расположения массивов на локальной и глобальной шине.

Максимальное быстродействие достигается при следующей конфигурации:
GSrcBuffer: Global SRAM
GDstBuffer: Global SRAM
LBuffer: Local SRAM
GBuffer: Global SRAM

См. также NMFFT512Set NMIFFT512

1.3.6 NMIFFT512

Синтаксис:

```
#include "nmfft.h"
int NMIFFT512(
    CmplxInt*  GSrcBuffer, // Source buffer :long[512]
    CmplxInt*  LDstBuffer, // Result FFT   :long[512]
    __int64*   LBuffer,    // Temp buffer  :long[512*3]
    __int64*   GBuffer,    // Temp buffer  :long[512*3]
    int        ShiftR1=9,  // First shift normalization
    int        ShiftR2=-1) // Final shift normalization
```

Описание: Функция NMIFFT512 выполняет обратное 512-точечное быстрое преобразование Фурье.

void* GSrcBuffer
Указатель на исходный массив 512 комплексных чисел. (long[512]).
Диапазон входных данных приведен в **Табл. 1**

void *LDstBuffer
Указатель на результирующий массив 512 комплексных чисел (long[512]).

void* LBuffer
Временный буфер размером long[512*3] для хранения промежуточных вычислений на локальной шине.

void* GBuffer
Временный буфер размером long[512*3] для хранения промежуточных вычислений на глобальной шине.

int ShiftR1=9
Промежуточный сдвиг результатов на ShiftR1 бит вправо (первая нормализация). По умолчанию равен 9.

int ShiftR2=-1
Заключительный сдвиг результатов на ShiftR2 бит вправо (вторая нормализация) в конце вычисления обратного БПФ. По умолчанию ShiftR2 принимается равным 14 при установленной точности 7-бит или 12 - при точности 6-бит, что соответствует результатам вычислений БПФ с плавающей точкой. (Механизм нормализации более подробно описан в приложении). Точность устанавливается функцией NMIFFT512Set, если эта функция не вызывалась, то по умолчанию принимается 7-битная точность.

Возвращает Функция всегда возвращает 0. (Reserved)

Пример:

```
include "nmfft.h"
extern ComplexInt LDst[]; // Result array
extern ComplexInt GSrc[]; // Source array
extern __int64 LBuffer[]; // Temp array
extern __int64 GBuffer[]; // Temp array
void main ()
{
    NMIFFT512(GSrc, LDst, LBuffer, GBuffer);
}
```

Производительность Производительность функции определяется в зависимости от расположения массивов на локальной и глобальной шине.
Максимальное быстродействие достигается при следующей конфигурации:
GSrcBuffer: Global SRAM
LDstBuffer: Local SRAM
LBuffer: Local SRAM
GBuffer: Global SRAM

См. также NMFFT512 NMIFFT512Set

1.3.7 NMFFT512Set

Синтаксис `#include "nmfft.h"`

`int NMFFT512Set(int Flag) // Type of precision`

Описание: Функция NMFFT512 устанавливает 6-битную или 7-битную точность вычислений для функции NMFFT512.

`int Flag`

Через параметр Flag передается тип устанавливаемой точности с помощью констант **FFT6BIT** или **FFT7BIT**. Установленная точность остается действующей до следующего вызова функции NMFFT512Set. По умолчанию устанавливается 7-битная точность. Т.е. если до вызова NMFFT512 установка точности функцией NMFFT512Set явно не проводилась, то вычисления будут проходить с 7-битной точностью.

FFT6BIT- устанавливает 6-битную точность вычислений. Коэффициенты Фурье преобразования задаются с помощью масштабирующего множителя 64.0 по формуле:

`W.re=round(64.0*cos(x))`

`W.im=round(64.0*sin(x))`

FFT7BIT- устанавливает 7-битную точность вычислений. Коэффициенты Фурье преобразования задаются с помощью масштабирующего множителя 127.0 по формуле:

`W.re=round(127.0*cos(x))`

`W.im=round(127.0*sin(x))`

Данный масштабирующий множитель следует учитывать при явном задании коэффициента нормализации ShiftR функции NMFFT512.

Возвращает Функция всегда возвращает 0. (Reserved)

Пример:

```
#include "nmfft.h"
extern ComplexInt GSrc[]; // Source array
extern ComplexInt GDst[]; // Result array
extern __int64 LBuffer[]; // Temp array
extern __int64 GBuffer[]; // Temp array
void main ()
{
    NMFFT512Set(FFT6BIT);
    NMFFT512(GSrc, GDst, LBuffer, GBuffer);
}
```

См. также NMFFT512

1.3.8 NMIFFT512Set

Синтаксис `#include "nmfft.h"`

`int NMIFFT512Set(int Flag) // Type of precision`

Описание: Функция NMIFFT512 устанавливает 6-битную или 7-битную точность вычислений для функции NMFFT512.

`int Flag`

Через параметр Flag передается тип устанавливаемой точности с помощью констант **FFT6BIT** или **FFT7BIT**. Установленная точность остается действующей до следующего вызова функции NMFFT512Set. По умолчанию устанавливается 7-битная точность. Т.е. если до вызова NMFFT512 установка точности функцией NMFFT512Set явно не проводилась, то вычисления будут проходить с 7-битной точностью.

FFT6BIT- устанавливает 6-битную точность вычислений. Коэффициенты Фурье преобразования задаются с помощью масштабирующего множителя - 64.0 по формуле:

$W.re = \text{round}(64.0 * \cos(x))$

$W.im = \text{round}(64.0 * \sin(x))$

FFT7BIT- устанавливает 7-битную точность вычислений. Коэффициенты Фурье преобразования задаются с помощью масштабирующего множителя - 127.0 по формуле:

$W.re = \text{round}(127.0 * \cos(x))$

$W.im = \text{round}(127.0 * \sin(x))$

Значение масштабирующего множителя следует учитывать при явном задании коэффициентов нормализации ShiftR1 и ShiftR2 функции NMIFFT512.

Возвращает Функция всегда возвращает 0. (Reserved)

Пример:

```
#include "nmfft.h"
extern ComplexInt GSrc[]; // Source array
extern ComplexInt LDst[]; // Result array
extern __int64 LBuffer[]; // Temp array
extern __int64 GBuffer[]; // Temp array

void main ()
{
    NMIFFT512Set(FFT7BIT);
    NMIFFT512(GSrc, LDst, LBuffer, GBuffer);
}
```

См. также NMIFFT512

1.3.9 NMFFT1024

Синтаксис:

```
#include "nmfft.h"
int NMFFT1024(
    CmplxInt*  GSrcBuffer, // Source buffer:long[1024]
    CmplxInt*  LDstBuffer, // Result FFT :long[1024]
    __int64*   LBuffer,    // Temp buffer :long[1024*3]
    __int64*   GBuffer,    // Temp buffer :long[1024]
    int        ShiftR=-1  // Right shift normalization
);
```

Описание: Функция NMFFT1024 выполняет прямое 1024-точечное быстрое преобразование Фурье.

CmplxInt* GSrcBuffer
Указатель на исходный массив 1024 комплексных чисел. (long[1024]).
Диапазон входных данных приведен в **Табл. 1**

CmplxInt *LDstBuffer
Указатель на результирующий массив 1024 комплексных чисел.
(long[1024]).

__int64* LBuffer
Временный буфер размером long[3*1024] для хранения промежуточных вычислений на локальной шине.

__int64* GBuffer
Временный буфер размером long[1024] для хранения промежуточных вычислений на глобальной шине.

int ShiftR=-1
Коэффициента нормализации. Выполняет арифметический сдвиг на ShiftR бит вправо результирующего массива для получения нормализованного массива GDstBuffer.
По умолчанию принимается равным 14 при установленной точности 7-бит или 12 - при точности 6-бит, что соответствует результатам вычислений БПФ с плавающей точкой. (Механизм нормализации более подробно описан в приложении). Точность устанавливается функцией NMFFT512Set, если эта функция не вызывалась, то по умолчанию принимается 7-битная

Возвращает Функция всегда возвращает 0. (Reserved)

Пример:

```
#include "nmfft.h"
extern ComplexInt  GSrc[];    // Source array
extern ComplexInt  LDst[];    // Result array
extern __int64     LBuffer[]; // Temp array
extern __int64     GBuffer[]; // Temp array
void main ()
{
    NMFFT1024(GSrc,LDst,LBuffer,GBuffer);
}
```

Производительность Производительность функции определяется в зависимости от расположения массивов на локальной и глобальной шине.
Максимальное быстродействие достигается при следующей конфигурации:
GSrcBuffer: Global SRAM
LDstBuffer: Local SRAM
LBuffer: Local SRAM
GBuffer: Global SRAM

См. также NMIFFT1024 NMFFT1024Set

1.3.10 NMIFFT1024

Синтаксис	<pre>#include "nmfft.h" int NMIFFT1024(CmplxInt* GSrcBuffer, // Source buffer :long[1024] CmplxInt* GDstBuffer, // Result FFT :long[1024] __int64* LBuffer, // Temp buffer :long[1024*3] __int64* GBuffer, // Temp buffer :long[1024*3] int ShiftR1=10, // First shift normalization int ShiftR2=-1) // Final shift normalization</pre>
Описание:	<p>Функция NMIFFT1024 выполняет обратное 1024-точечное быстрое преобразование Фурье.</p> <p>void* GSrcBuffer Указатель на исходный массив 1024 комплексных чисел. (long[1024]). Диапазон входных данных приведен в Табл. 1</p> <p>void *LDstBuffer Указатель на результирующий массив 1024 комплексных чисел (long[1024]).</p> <p>void* LBuffer Временный буфер размером long[1024*3] для хранения промежуточных вычислений на локальной шине.</p> <p>void* GBuffer Временный буфер размером long[1024*3] для хранения промежуточных вычислений на глобальной шине.</p> <p>int ShiftR1=10 Промежуточный сдвиг результатов на ShiftR1 бит вправо (первая нормализация). По умолчанию равен 10.</p> <p>int ShiftR2=-1 Заключительный сдвиг результатов на ShiftR2 бит вправо (вторая нормализация) в конце вычисления обратного БПФ. По умолчанию ShiftR2 принимается равным 14 при установленной точности 7-бит или 12 – при точности 6-бит, что соответствует результатам вычислений БПФ с плавающей точкой. (Механизм нормализации более подробно описан в приложении). Точность устанавливается NMIFFT1024Set, если эта функция не вызывалась, то по умолчанию принимается 7-битная точность.</p>
Возвращает	Функция всегда возвращает 0. (Reserved).
Пример:	<pre>#include "nmfft.h" extern ComplexInt GSrc[]; // Source array extern ComplexInt GDst[]; // Result array extern __int64 LBuffer[]; // Temp array extern __int64 GBuffer[]; // Temp array void main () { NMIFFT1024(GSrc, GDst, LBuffer, GBuffer); }</pre>
Производительность	<p>Производительность функции определяется в зависимости от расположения массивов на локальной и глобальной шине.</p> <p>Максимальное быстродействие достигается при следующей конфигурации:</p> <p>GSrcBuffer: Global SRAM GDstBuffer: Global SRAM LBuffer: Local SRAM GBuffer: Local SRAM</p>

См. также NMFFT1024 NMIFFT1024Set

1.3.11 NMFFT1024Set

Синтаксис `#include "nmfft.h"`
`int NMFFT1024Set(int Flag) // Type of precision`

Описание: Функция NMFFT1024 устанавливает 6-битную или 7-битную точность вычислений для функции NMFFT1024.

`int Flag`

Через параметр Flag передается тип устанавливаемой точности с помощью констант **FFT6BIT** или **FFT7BIT**. Установленная точность остается действующей до следующего вызова функции NMFFT1024Set. По умолчанию устанавливается 7-битная точность. Т.е. если до вызова NMFFT1024 установка точности функцией NMFFT1024Set явно не проводилась, то вычисления будут проходить с 7-битной точностью.

FFT6BIT- устанавливает 6-битную точность вычислений. Коэффициенты Фурье преобразования задаются с помощью масштабирующего множителя 64.0 по формуле:

`W.re=round(64.0*cos(x))`

`W.im=round(64.0*sin(x))`

FFT7BIT- устанавливает 7-битную точность вычислений. Коэффициенты Фурье преобразования задаются с помощью масштабирующего множителя 127.0 по формуле:

`W.re=round(127.0*cos(x))`

`W.im=round(127.0*sin(x))`

Данный масштабирующий множитель следует учитывать при явном задании коэффициента нормализации ShiftR функции NMFFT1024.

Возвращает Функция всегда возвращает 0. (Reserved)

Пример:

```
#include "nmfft.h"
extern ComplexInt GSrc[]; // Source array
extern ComplexInt LDst[]; // Result array
extern __int64 LBuffer[]; // Temp array
extern __int64 GBuffer[]; // Temp array
void main ()
{
    NMFFT1024Set(FFT7BIT);
    NMFFT1024(GSrc, LDst, LBuffer, GBuffer);
}
```

См. также NMFFT1024

1.3.12 NMIFFT1024Set

Синтаксис `#include "nmfft.h"`
`int NMIFFT1024Set(int Flag) // Type of precision`
Описание: Функция NMIFFT1024 устанавливает 6-битную или 7-битную точность вычислений для функции NMFFT1024.

`int Flag`
Через параметр Flag передается тип устанавливаемой точности с помощью констант **FFT6BIT** или **FFT7BIT**. Установленная точность остается действующей до следующего вызова функции NMFFT1024Set. По умолчанию устанавливается 7-битная точность. Т.е. если до вызова NMFFT1024 установка точности функцией NMFFT1024Set явно не проводилась, то вычисления будут проходить с 7-битной точностью.

FFT6BIT- устанавливает 6-битную точность вычислений. Коэффициенты Фурье преобразования задаются с помощью масштабирующего множителя - 64.0 по формуле:

$W.re = \text{round}(64.0 * \cos(x))$

$W.im = \text{round}(64.0 * \sin(x))$

FFT7BIT- устанавливает 7-битную точность вычислений. Коэффициенты Фурье преобразования задаются с помощью масштабирующего множителя - 127.0 по формуле:

$W.re = \text{round}(127.0 * \cos(x))$

$W.im = \text{round}(127.0 * \sin(x))$

Значение масштабирующего множителя следует учитывать при явном задании коэффициентов нормализации ShiftR1 и ShiftR2 функции NMIFFT1024.

Возвращает Функция всегда возвращает 0. (Reserved)

Пример:

```
#include "nmfft.h"
extern ComplexInt GSrc[]; // Source array
extern ComplexInt GDst[]; // Result array
extern __int64 LBuffer[]; // Temp array
extern __int64 GBuffer[]; // Temp array

void main ()
{
    NMIFFT1024Set(FFT6BIT);
    NMIFFT1024(GSrc, GDst, LBuffer, GBuffer);
}
```

См. также NMIFFT1024

1.3.13 NMFFT2048

Синтаксис

```
#include "nmfft.h"
int NMFFT2048(
    CmplxInt*  GSrcBuffer, // Source buffer :long[2048]
    CmplxInt*  GDstBuffer, // Result FFT   :long[2048]
    __int64*   LBuffer,    // Temp buffer  :long[2048*4]
    int        ShiftR=-1  // Right shift normalization
);
```

Описание: Функция NMIFFT1024 выполняет прямое 1024-точечное быстрое преобразование Фурье.

`void* GSrcBuffer`
Указатель на исходный массив 1024 комплексных чисел. (`long[1024]`).
Диапазон входных данных приведен в **Табл. 1**

`void *GDstBuffer`
Указатель на результирующий массив 1024 комплексных чисел (`long[1024]`).

`void* LBuffer`
Временный буфер размером `long[2048*4]` для хранения промежуточных вычислений на локальной шине.

`int ShiftR=-1`
Коэффициент нормализации. Выполняет арифметический сдвиг на `ShiftR` бит вправо результирующего массива для получения нормализованного массива `GDstBuffer`.
По умолчанию принимается равным 14 при установленной точности 7-бит или 12 - при точности 6-бит, что соответствует результатам вычислений БПФ с плавающей точкой. (Механизм нормализации более подробно описан в приложении). Точность устанавливается функцией `NMFFT2048Set`, если эта функция не вызывалась, то по умолчанию принимается 7-битная

Возвращает Функция всегда возвращает 0. (Reserved).

Пример:

```
#include "nmfft.h"
extern ComplexInt GSrc[]; // Source array
extern ComplexInt GDst[]; // Result array
extern __int64 LBuffer[]; // Temp buffer
void main ()
{
    NMFFT2048Set(FFT7BIT);
    NMFFT2048(GSrc, GDst, LBuffer);
}
```

Производительность Производительность функции определяется в зависимости от расположения массивов на локальной и глобальной шине.
Максимальное быстродействие достигается при следующей конфигурации:
`GSrcBuffer: Global SRAM`
`GDstBuffer: Local SRAM`
`LBuffer: Local SRAM`

См. также `NMFFT2048Set` `NMIFFT2048`

1.3.14 NMIFFT2048

Синтаксис	<pre>#include "nmfft.h" int NMIFFT2048(CmplxInt* GSrcBuffer, // Source buffer :long[2048] CmplxInt* LDstBuffer, // Result FFT :long[2048] __int64* LBuffer, // Temp buffer :long[2048*4] __int64* GBuffer, // Temp buffer :long[2048*4] int ShiftR1=11, // First shift normalization int ShiftR2=-1 // Final shift normalization);</pre>
Описание:	<p>Функция NMIFFT2048 выполняет обратное 2048-точечное быстрое преобразование Фурье.</p> <p>void* GSrcBuffer Указатель на исходный массив 2048 комплексных чисел. (long[2048]). Диапазон входных данных приведен в Табл. 1</p> <p>void *LDstBuffer Указатель на результирующий массив 2048 комплексных чисел (long[2048]).</p> <p>void* LBuffer Временный буфер размером long[2048*4] для хранения промежуточных вычислений на локальной шине.</p> <p>void* GBuffer Временный буфер размером long[2048*4] для хранения промежуточных вычислений на глобальной шине.</p> <p>int ShiftR1=11 Промежуточный сдвиг результатов на ShiftR1 бит вправо (первая нормализация). По умолчанию равен 11.</p> <p>int ShiftR2=-1 Заключительный сдвиг результатов на ShiftR2 бит вправо (вторая нормализация) в конце вычисления обратного БПФ. По умолчанию ShiftR2 принимается равным 14 при установленной точности 7-бит или 12 - при точности 6-бит, что соответствует результатам вычислений БПФ с плавающей точкой. (Механизм нормализации более подробно описан в приложении). Точность устанавливается NMIFFT2048Set, если эта функция не вызывалась, то по умолчанию принимается 7-битная точность</p>
Возвращает	Функция всегда возвращает (Reserved).
Пример:	<pre>#include "nmfft.h" extern ComplexInt LDst[]; // Source array extern ComplexInt GSrc[]; // Result array extern __int64 LBuffer[]; // Temp array extern __int64 GBuffer[]; // Temp array void main () { NMIFFT2048(GSrc, LDst, LBuffer, GBuffer); }</pre>
Производительность	<p>Производительность функции определяется в зависимости от расположения массивов на локальной и глобальной шине.</p> <p>Максимальное быстродействие достигается при следующей конфигурации:</p> <p>GSrcBuffer: Global SRAM LDstBuffer: Local SRAM LBuffer: Local SRAM GBuffer: Global SRAM</p>
См. также	NMIFFT2048Set NMIFFT2048

1.3.15 NMFFT2048Set

Синтаксис `#include "nmfft.h"`
`int NMFFT2048Set(int Flag) // Type of precision`

Описание: Функция NMFFT2048 устанавливает 6-битную или 7-битную точность вычислений для функции NMFFT2048.

`int Flag`

Через параметр Flag передается тип устанавливаемой точности с помощью констант **FFT6BIT** или **FFT7BIT**. Установленная точность остается действующей до следующего вызова функции NMFFT2048Set. По умолчанию устанавливается 7-битная точность. Т.е. если до вызова NMFFT2048 установка точности функцией NMFFT2048Set явно не проводилась, то вычисления будут проходить с 7-битной точностью.

FFT6BIT- устанавливает 6-битную точность вычислений. Коэффициенты Фурье преобразования задаются с помощью масштабирующего множителя 64.0 по формуле:

`W.re=round(64.0*cos(x))`

`W.im=round(64.0*sin(x))`

FFT7BIT- устанавливает 7-битную точность вычислений. Коэффициенты Фурье преобразования задаются с помощью масштабирующего множителя 127.0 по формуле:

`W.re=round(127.0*cos(x))`

`W.im=round(127.0*sin(x))`

Данный масштабирующий множитель следует учитывать при явном задании коэффициента нормализации функции NMFFT2048.

Возвращает Функция всегда возвращает 0. (Reserved)

Пример:

```
#include "nmfft.h"
extern ComplexInt GSrc[]; // Source array
extern ComplexInt GDst[]; // Result array
extern __int64 LBuffer[]; // Temp buffer
```

```
void main ()
{
    NMFFT2048Set(FFT6BIT);
    NMFFT2048(GSrc, GDst, LBuffer);
}
```

См. также NMFFT2048

1.3.16 NMIFFT2048Set

Синтаксис `#include "nmfft.h"`
`int NMIFFT2048Set(int Flag) // Type of precision`
Описание: Функция NMIFFT2048 устанавливает 6-битную или 7-битную точность вычислений для функции NMFFT2048.

`int Flag`
Через параметр Flag передается тип устанавливаемой точности с помощью констант **FFT6BIT** или **FFT7BIT**. Установленная точность остается действующей до следующего вызова функции NMFFT2048Set. По умолчанию устанавливается 7-битная точность. Т.е. если до вызова NMFFT2048 установка точности функцией NMFFT2048Set явно не проводилась, то вычисления будут проходить с 7-битной точностью.

FFT6BIT- устанавливает 6-битную точность вычислений. Коэффициенты Фурье преобразования задаются с помощью масштабирующего множителя - 64.0 по формуле:

$W.re = \text{round}(64.0 * \cos(x))$

$W.im = \text{round}(64.0 * \sin(x))$

FFT7BIT- устанавливает 7-битную точность вычислений. Коэффициенты Фурье преобразования задаются с помощью масштабирующего множителя - 127.0 по формуле:

$W.re = \text{round}(127.0 * \cos(x))$

$W.im = \text{round}(127.0 * \sin(x))$

Значение масштабирующего множителя следует учитывать при явном задании коэффициентов нормализации функции NMIFFT2048.

Возвращает Функция всегда возвращает 0. (Reserved)

Пример:

```
#include "nmfft.h"
extern ComplexInt LDst[]; // Source array
extern ComplexInt GSrc[]; // Result array
extern __int64 LBuffer[]; // Temp array
extern __int64 GBuffer[]; // Temp array
void main ()
{
    NMIFFT2048Set(FFT6BIT);
    NMIFFT2048(GSrc, LDst, LBuffer, GBuffer);
}
```

См. также NMIFFT2048

Данный раздел содержит пример сборки программы с вызовом функции БПФ, предназначенной для выполнения на процессоре NM6403

2.1 Подключение библиотеки БПФ

Для построения пользовательских программ, использующих функции БПФ, в строку сборки исполняемого файла необходимо включить библиотеку БПФ -**NMFFT.LIB**. В состав этой библиотеки входят: функции прямого БПФ:

NMFFT256, NMFFT512, NMFFT1024, NMFFT2048

функции обратного БПФ:

NMIFFT256, NMIFFT512, NMIFFT1024, NMIFFT2048.

Функции установки точности:

NMFFT256Set, NMFFT512Set, NMFFT1024Set, NMFFT2048Set,
NMIFFT256Set, NMIFFT512Set, NMIFFT1024Set, NMIFFT2048Set.

Указанные функции объявлены в файле NMFFT.H.

Строка для сборки проекта может выглядеть следующим образом:

```
nmcc -g -m -l -offt.abs -cmain.cfg main.cpp mem.asm nmfft.lib
```

FFT.ABS – создаваемый исполняемый файл.

MAIN.CPP – программа с вызовом функций БПФ

MAIN.CFG- файл конфигурации

MEM.ASM- файл выделения и расположения временных, входных и выходных массивов на локальной и глобальной шине.

NMFFT.LIB – библиотека БПФ

2.2 Пример сборки проекта

В данном разделе приводится пример программы для двухпроцессорного модуля МЦ 4.01, вызывающей функцию прямого 256-точечного БПФ. Проект включает в себя четыре файла: main.cpp, main.cfg, mem.asm и make.bat. После запуска сборочного файла make.bat создается исполняемый файл FFT256.ABS.

Main.cpp Пример вызова функции 256-точечного БПФ

```
#include <time.h>
#include "nmfft.h"
extern CmplxInt LDst[]; // Результирующий массив
extern CmplxInt GSrc[]; // Исходный массив
extern __int64 LBuffer[]; // Временный массив
extern __int64 GBuffer[]; // Временный массив

int main ()
{
    clock_t startTime = clock();
    NMFFT256(GSrc,LDst,LBuffer,GBuffer);
    clock_t endTime = clock();
    return (endTime-startTime); // Возврат времени
                                // выполнения функции
}
```

Mem.asm Файл выделения временных, входных и выходных массивов на локальной (LocalSRAM) и глобальной шине (GlobalSRAM)

```
const LONG_LENGTH=256;
/***** Local Bus *****/
data "LocalSRAM"
    global _LDst :long[2*LONG_LENGTH];
    global _LBuffer :long[3*LONG_LENGTH];
end "LocalSRAM";
/***** Global Bus *****/
data «GlobalSRAM»
    global _GSrc :long[LONG_LENGTH];
    global _GBuffer :long[LONG_LENGTH];
end «GlobalSRAM»;
```

Main.cfg Файл конфигурации (для двухпроцессорной платы МЦ 4.01)

```
MEMORY
{
    local0 : at 0x00000080, len = 0x01ff80;
    local1 : at 0x40000000, len = 0x800000;
    global0: at 0x80004000, len = 0x01c000;
    global1: at 0xc0000000, len = 0x020000;
}
SEGMENTS
{
    local: in local0;
    global: in global0;
    locDRAM: in local1;
}
SECTIONS
{
    .init : in local;
    .fini : in local;
    text : in local;
    LocalSRAM : in local;
    GlobalSRAM : in global;
```

```
.stack      : in local;  
.bssLocalSRAM : in local;  
.bssGlobalSRAM: in global;  
}
```

make.bat Сборочный файл

```
nmcc -g -m -l -oFFT256.abs -cmain.cfg main.cpp mem.asm nmfft.lib
```


Данный раздел посвящен описанию алгоритмов вычислений на основе которых были построены функции БПФ. Приводится краткий математический вывод алгоритмов, а так же блок-схемы вычислений функций БПФ.

3.1 Реализация прямого БПФ-256

1.1.1 Алгоритм вычисления БПФ-256 на процессоре NM6403

Дискретное 256-точечное преобразование Фурье определяется формулой:

$$Y(k) = \sum_{n=0}^{255} W_{256}^{k \cdot n} \cdot X(n), k = 0..255 \quad (1.1)$$

$$\text{где } W_{256}^{kn} = \exp\left(-\frac{2\pi \cdot n \cdot k}{256}\right) \quad (1.2)$$

Прямое преобразование Фурье(1.1) по основанию 256, вычисляется через быстрое преобразование Фурье с помощью двух преобразований по основанию 16:

$$Y(k) = \sum_{n=0}^{15} W_{256}^{k \cdot n} \cdot \sum_{i=0}^{15} W_{256}^{16k \cdot i} \cdot X(16 \cdot i + n); k = 0..255 \quad (1.3)$$

Обозначим правую сумму через S(k,n):

$$S(k, n) = \sum_{i=0}^{15} W_{256}^{16k \cdot i} \cdot X(16 \cdot i + n); k = 0..255, n = 0..15 \quad (1.4)$$

Тогда формула(1.3) запишется в виде:

$$Y(k) = \sum_{n=0}^{15} W_{256}^{k \cdot n} \cdot S(k, n); k = 0..255 \quad (1.5)$$

Так как коэффициенты W, входящие в выражение(1.4) для S(k,n), повторяются через каждые k=16, т.е. :

$$W_{256}^{16k \cdot i} = W_{256}^{16(k+16) \cdot i}; k = 0..255 \quad (1.6)$$

то суммы S(k,n) также будут иметь период повторения равный k=16 т.е. переопределив переменную k (k=0..255) на 16p+k (p=0..15, k=0..15), получим:

$$S(16p + k, n) = S(k, n); k = 0..15, p = 0..15, n = 0..15 \quad (1.7)$$

Т.е., формулу для вычисления коэффициентов Фурье(1.5) можно переписать в виде:

$$Y(16p + k) = \sum_{n=0}^{15} W_{256}^{(16p+k) \cdot n} \cdot S(k, n); p = 0..15, k = 0..15 \quad (1.8)$$

Приложение

Таким образом, процедура вычисления БПФ состоит из двух этапов:

1. Нахождение массива S длиной 256 элементов

$$S(k, n) = \sum_{i=0}^{15} W_{256}^{16k \cdot i} \cdot X(16 \cdot i + n); k = 0..15, n = 0..15$$

2. Нахождение результирующего массива Y длиной 256 элементов

$$Y(16p + k) = \sum_{n=0}^{15} W_{256}^{(16p+k) \cdot n} \cdot S(k, n); p = 0..15, k = 0..15$$

3.1.2 Схема вычислений в функции NMFFT256

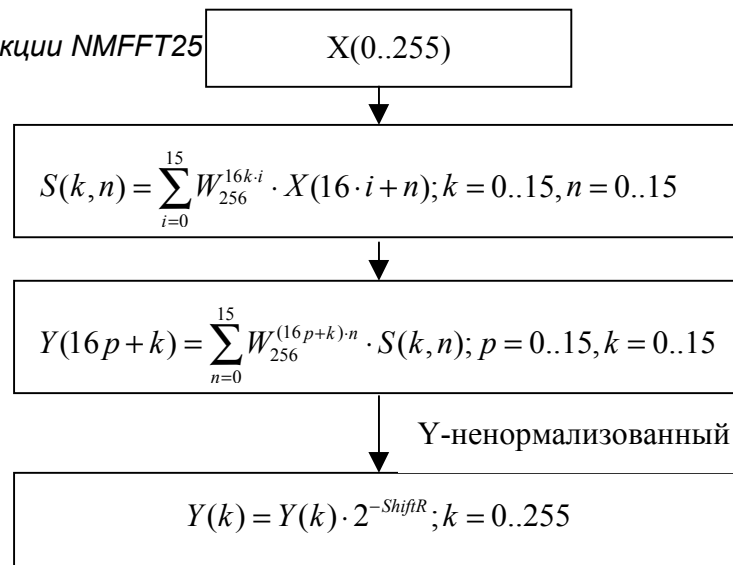
Так как все вычисления БПФ 256 производятся в целочисленной арифметике, то комплексные коэффициенты W_{256}^k представляются в виде пары 8-разрядных чисел (в двух возможных вариантах см. п. 1.2-Функции управления точностью БПФ):

- Вариант 1.
 $\text{Re}(W_{256}^k) = \text{int}(127.0 * \cos(-2\pi \cdot k / 256))$
 $\text{Im}(W_{256}^k) = \text{int}(127.0 * \sin(-2\pi \cdot k / 256))$
- Вариант 2.
 $\text{Re}(W_{256}^k) = \text{int}(64.0 * \cos(-2\pi \cdot k / 256))$
 $\text{Im}(W_{256}^k) = \text{int}(64.0 * \sin(-2\pi \cdot k / 256))$

(**int** - оператор приведения числа с плавающей точкой к целому числу путем отбрасывания дробной части)

В результате после двух этапов умножений полученный массив Y будет превосходить истинный в 127^2 раз (для первого варианта). Нормализацию результирующего массива можно выполнить с помощью арифметического сдвига вправо на число бит задаваемое параметром *ShiftR* функции NMFFT256. По умолчанию параметр *ShiftR*=14, что приблизительно соответствует делению на 127^2

Рис. 1 Блок-схема функции NMFFT256



3.2 Реализация прямого БПФ-512

3.2.1 Алгоритм вычисления БПФ-512 на процессоре NM6403

Дискретное 512-точечное преобразование Фурье определяется формулой:

$$Y(k) = \sum_{n=0}^{512} W_{512}^{k \cdot n} \cdot X(n), k = 0..511 \quad (2.1)$$

$$\text{где } W^k = W_{512}^k = \exp\left(-\frac{2\pi \cdot i \cdot k}{512}\right) \quad (2.2)$$

Преобразуем формулу (2.1) следующим образом:

$$\begin{aligned} Y(k) &= \sum_{n=0}^{512} W_{512}^{k \cdot n} \cdot X(n) = \\ &= \sum_{i=0}^1 W_{512}^{k \cdot i} \cdot \sum_{n=0}^{255} W_{512}^{2k \cdot n} \cdot X(2n+i) = \\ &= \sum_{i=0}^3 W_{512}^{k \cdot i} \cdot \sum_{n=0}^{127} W_{512}^{4k \cdot n} \cdot X(4n+i) = \\ &= \sum_{i=0}^7 W_{512}^{k \cdot i} \cdot \sum_{n=0}^{63} W_{512}^{8k \cdot n} \cdot X(8n+i) = \\ &= \sum_{i=0}^{15} W_{512}^{k \cdot i} \cdot \sum_{n=0}^{31} W_{512}^{16k \cdot n} \cdot X(16n+i) = \\ &= \sum_{j=0}^1 W_{512}^{16k \cdot j} \cdot \sum_{i=0}^{15} W_{512}^{k \cdot i} \cdot \sum_{n=0}^{15} W_{512}^{32k \cdot n} \cdot X(32n+16j+i) = \\ &= \sum_{j=0}^3 W_{512}^{16k \cdot j} \cdot \sum_{i=0}^{15} W_{512}^{k \cdot i} \cdot \sum_{n=0}^7 W_{512}^{64k \cdot n} \cdot X(64n+16j+i) = \\ &= \sum_{j=0}^7 W_{512}^{16k \cdot j} \cdot \sum_{i=0}^{15} W_{512}^{k \cdot i} \cdot \sum_{n=0}^3 W_{512}^{128k \cdot n} \cdot X(128n+16j+i) = \\ &= \sum_{j=0}^{15} W_{512}^{16k \cdot j} \cdot \sum_{i=0}^{15} W_{512}^{k \cdot i} \cdot \sum_{n=0}^1 W_{512}^{256k \cdot n} \cdot X(256n+16j+i) = \\ &= \sum_{i=0}^{15} W_{512}^{k \cdot i} \cdot \sum_{j=0}^{15} W_{512}^{16k \cdot j} \cdot \sum_{n=0}^1 W_{512}^{256k \cdot n} \cdot X(256n+16j+i) \end{aligned} \quad (2.3)$$

Обозначим правую сумму через $S(k,j,i)$:

$$S(k, j, i) = \sum_{n=0}^1 W_{512}^{256k \cdot n} \cdot X(256n+16j+i) \quad (2.4)$$

Приложение

Тогда формула (2.3) запишется в виде:

$$Y(k) = \sum_{i=0}^{15} W_{512}^{k \cdot i} \cdot \sum_{j=0}^{15} W_{512}^{16k \cdot j} \cdot S(k, j, i); k = 0..511 \quad (2.5)$$

Учитывая, что коэффициенты W , входящие в выражение (2.4) для $S(k, j, i)$, повторяются через каждые $k=2$:

$$\begin{aligned} W_{512}^{256(0i)} &= W_{512}^{256(2i)} = W_{512}^{256(4i)} = \dots \\ W_{512}^{256(1i)} &= W_{512}^{256(3i)} = W_{512}^{256(5i)} = \dots \end{aligned} \quad (2.6)$$

то суммы $S(k, j, i)$ также будут иметь период повторения равный 2 т.е. переопределив переменную $k(k=0..511)$ на $2p+k$ ($k=0..1, p=0..255$), получим:

$$S(k, j, i) = S(2p+k, j, i); k = 0..1, p = 0..255 \quad (2.7)$$

Следовательно, формулу для вычисления коэффициентов Фурье можно переписать в виде:

$$Y(2p+k) = \sum_{i=0}^{15} W_{512}^{(2p+k) \cdot i} \cdot \sum_{j=0}^{15} W_{512}^{16(2p+k) \cdot j} \cdot S(k, j, i); k = 0..1, p = 0..255 \quad (2.8)$$

Обозначим правую сумму через $T(p, k, i)$:

$$T(p, k, i) = \sum_{j=0}^{31} W_{512}^{16(2p+k) \cdot j} \cdot S(k, j, i) \quad (2.9)$$

Тогда формула (2.8) запишется в виде:

$$Y(2p+k) = \sum_{i=0}^{15} W_{512}^{16(2p+k) \cdot i} \cdot T(p, k, i); k = 0..1, p = 0..255 \quad (2.10)$$

Учитывая, что коэффициенты W , входящие в выражение (2.9) для $T(p, k, i)$, повторяются через каждые $p=16$, т.е. :

$$W_{512}^{16(2p+k) \cdot i} = W_{512}^{16(2(p+16)+k) \cdot i}; p = 0..255, k = 0..1; i = 0..15 \quad (2.11)$$

то суммы $T(p,k,i)$ также будут иметь период повторения равный $p=16$ т.е. переопределив переменную $p(p=0..255)$ на $16l+p$ ($l=0..15, p=0..15$), получим:

$$T(p, k, i) = T(16l + p, k, i); l = 0..15, p = 0..15, k = 0..1, \quad (2.12)$$

Следовательно, формулу для вычисления коэффициентов Фурье (2.10) можно переписать в виде:

$$Y(32l + 2p + k) = \sum_{i=0}^{15} W_{512}^{(32l+2p+k) \cdot i} \cdot T(p, k, i); l = 0..15, p = 0..15, k = 0..1$$

Таким образом, процедура вычисления БПФ состоит из трех этапов:

1. Нахождение массива S длиной 512 элементов

$$\begin{aligned} S(k, j, i) &= \sum_{n=0}^1 W_{512}^{256k \cdot n} \cdot X(256n + 16j + i) = \\ &= \sum_{n=0}^1 (-1)^{k \cdot n} \cdot X(256n + 16j + i); k = 0..1, i = 0..15, j = 0..15 \end{aligned}$$

2. Нахождение массива T длиной 512 элементов

$$T(p, k, i) = \sum_{j=0}^{31} W_{512}^{16(2p+k) \cdot j} \cdot S(k, j, i); p = 0..15, k = 0..1, i = 0..15$$

3. Нахождение результирующего массива Y длиной 512 элементов

$$Y(32l + 2p + k) = \sum_{i=0}^{15} W_{512}^{(32l+2p+k) \cdot i} \cdot T(p, k, i); l = 0..15, p = 0..15, k = 0..1$$

3.2.2 Схема вычислений в функции NMFFT512

Так как все вычисления БПФ 512 производятся в целочисленной арифметике, то комплексные коэффициенты W_{256}^k представляются в виде пары 8-разрядных чисел (в двух возможных вариантах см. п. 1.2-Функции управления точностью БПФ):

- Вариант 1.
 $\text{Re}(W_{256}^k) = \text{int}(127.0 * \cos(-2\pi \cdot k / 512))$
 $\text{Im}(W_{256}^k) = \text{int}(127.0 * \sin(-2\pi \cdot k / 512))$

- Вариант 2.

$$\operatorname{Re}(W_{256}^k) = \operatorname{int}(64.0 * \cos(-2\pi \cdot k / 512))$$

$$\operatorname{Im}(W_{256}^k) = \operatorname{int}(64.0 * \sin(-2\pi \cdot k / 512))$$

(**int** - оператор приведения числа с плавающей точкой к целому числу путем отбрасывания дробной части)

В результате после двух этапов умножений полученный массив Y будет превосходить истинный в 127^2 раз (для варианта 1).

Нормализацию результирующего массива можно выполнить с помощью арифметического сдвига вправо на число бит задаваемое параметром *ShiftR* функции NMFFT512. По умолчанию параметр *ShiftR*=14, что приближенно соответствует делению на 127^2

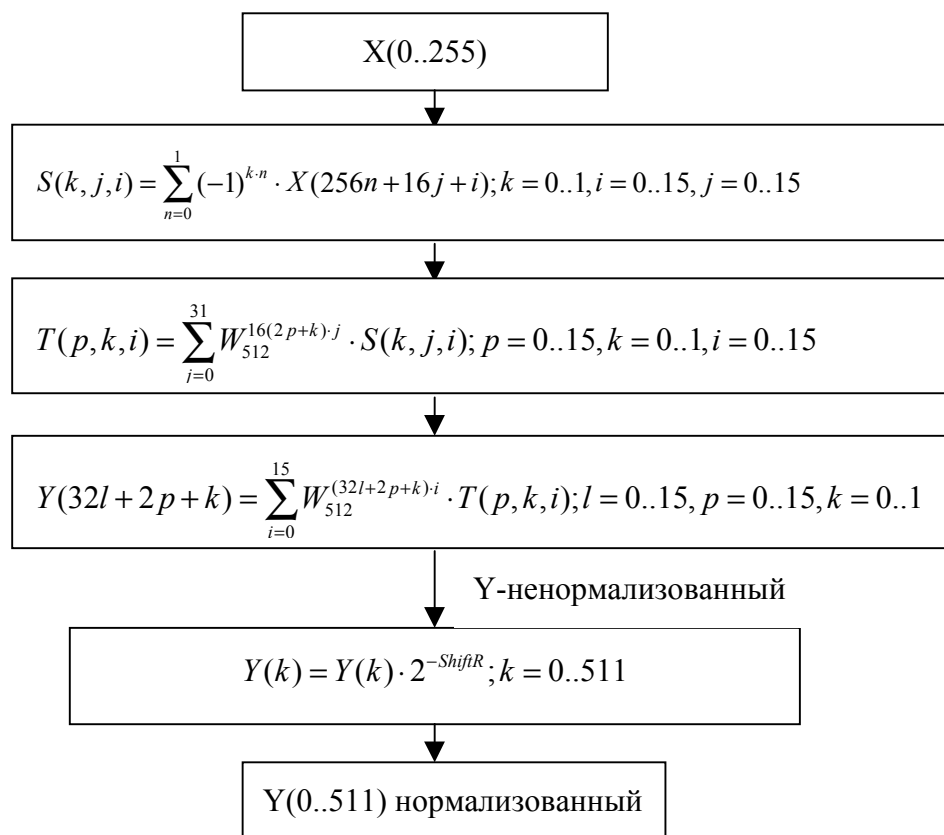


Рис .2 Блок-схема функции NMFFT512

3.3 Реализация прямого БПФ-1024

3.3.1 Алгоритм вычисления БПФ-1024 на процессоре NM6403

Дискретное 1024-точечное преобразование Фурье определяется формулой:

$$Y(k) = \sum_{n=0}^{1023} W_{1024}^{k \cdot n} \cdot X(n), k = 0..1023 \quad (3.1)$$

$$\text{где } W^k = W_{1024}^k = \exp\left(-\frac{2\pi \cdot i \cdot k}{1024}\right) \quad (3.2)$$

Преобразуем формулу (3.1) следующим образом:

$$\begin{aligned} Y(k) &= \sum_{n=0}^{1023} W_{1024}^{k \cdot n} \cdot X(n) = \\ &= \sum_{i=0}^1 W_{1024}^{k \cdot i} \cdot \sum_{n=0}^{511} W_{1024}^{2k \cdot n} \cdot X(2n+i) = \\ &= \sum_{i=0}^3 W_{1024}^{k \cdot i} \cdot \sum_{n=0}^{225} W_{1024}^{4k \cdot n} \cdot X(4n+i) = \\ &= \sum_{i=0}^7 W_{1024}^{k \cdot i} \cdot \sum_{n=0}^{127} W_{1024}^{8k \cdot n} \cdot X(8n+i) = \\ &= \sum_{i=0}^{15} W_{1024}^{k \cdot i} \cdot \sum_{n=0}^{63} W_{1024}^{16k \cdot n} \cdot X(16n+i) = \\ &= \sum_{j=0}^1 W_{1024}^{16k \cdot j} \cdot \sum_{i=0}^{15} W_{1024}^{k \cdot i} \cdot \sum_{n=0}^{31} W_{1024}^{32k \cdot n} \cdot X(32n+16j+i) = \\ &= \sum_{j=0}^3 W_{1024}^{16k \cdot j} \cdot \sum_{i=0}^{15} W_{1024}^{k \cdot i} \cdot \sum_{n=0}^{15} W_{1024}^{64k \cdot n} \cdot X(64n+16j+i) = \\ &= \sum_{j=0}^7 W_{1024}^{16k \cdot j} \cdot \sum_{i=0}^{15} W_{1024}^{k \cdot i} \cdot \sum_{n=0}^7 W_{1024}^{128k \cdot n} \cdot X(128n+16j+i) = \\ &= \sum_{j=0}^{15} W_{1024}^{16k \cdot j} \cdot \sum_{i=0}^{15} W_{1024}^{k \cdot i} \cdot \sum_{n=0}^3 W_{1024}^{256k \cdot n} \cdot X(256n+16j+i) = \\ &= \sum_{j=0}^{31} W_{1024}^{16k \cdot j} \cdot \sum_{i=0}^{15} W_{1024}^{k \cdot i} \cdot \sum_{n=0}^1 W_{1024}^{512k \cdot n} \cdot X(512n+16j+i) = \\ &= \sum_{i=0}^{15} W_{1024}^{k \cdot i} \cdot \sum_{j=0}^{31} W_{1024}^{16k \cdot j} \cdot \sum_{n=0}^1 W_{1024}^{512k \cdot n} \cdot X(512n+16j+i) \end{aligned} \quad (3.3)$$

Обозначим правую сумму через $S(k,j,i)$:

$$S(k, j, i) = \sum_{n=0}^1 W_{1024}^{512k \cdot i} \cdot X(512n + 16j + i) \quad (3.4)$$

Тогда формула (3.3) запишется в виде:

$$Y(k) = \sum_{i=0}^{15} W_{1024}^{k \cdot i} \cdot \sum_{j=0}^{31} W_{1024}^{16k \cdot j} \cdot S(k, j, i); k = 0..1023 \quad (3.5)$$

Учитывая, что коэффициенты W , входящие в выражение (3.4) для $S(k,j,i)$, повторяются через каждые $k=2$:

$$\begin{aligned} W_{1024}^{512(0i)} &= W_{1024}^{512(2i)} = W_{1024}^{512(4i)} = \dots \\ W_{1024}^{512(1i)} &= W_{1024}^{512(3i)} = W_{1024}^{512(5i)} = \dots \end{aligned} \quad (3.6)$$

то суммы $S(k,j,i)$ также будут иметь период повторения равный 2 т.е. переопределив переменную $k(k=0..1023)$ на $2p+k$ ($k=0..1, p=0..511$), получим:

$$S(k, j, i) = S(2p + k, j, i); k = 0..1, p = 0..511 \quad (3.7)$$

Следовательно, формулу для вычисления коэффициентов Фурье можно переписать в виде:

$$Y(2p + k) = \sum_{i=0}^{15} W_{1024}^{(2p+k) \cdot i} \cdot \sum_{j=0}^{31} W_{1024}^{16(2p+k) \cdot j} \cdot S(k, j, i); k = 0..1, p = 0..511 \quad (3.8)$$

Обозначим правую сумму через $T(p,k,i)$:

$$T(p, k, i) = \sum_{j=0}^{31} W_{1024}^{16(2p+k) \cdot j} \cdot S(k, j, i) \quad (3.9)$$

Тогда формула (3.8) запишется в виде:

$$Y(2p + k) = \sum_{i=0}^{15} W_{1024}^{16(2p+k) \cdot i} \cdot T(p, k, i); k = 0..1, p = 0..511 \quad (3.10)$$

Так как коэффициенты W , входящие в выражение (3.9) для $T(p,k,i)$, повторяются через каждые $p=32$:

$$W_{1024}^{16(2p+k)i} = W_{1024}^{16(2(p+32)+k)i}; p = 0..511, k = 0..1, i = 0..15 \quad (3.11)$$

то суммы $T(p,k,i)$ также будут иметь период повторения равный $p=32$ т.е. переопределив переменную $p(p=0..511)$ на $32l+p$ ($l=0..15, p=0..31$) получим:

$$T(p, k, i) = T(32l + p, k, i); l = 0..15, p = 0..31, k = 0..1, \quad (3.12)$$

Следовательно, формулу для вычисления коэффициентов Фурье можно переписать в виде:

$$Y(64l + 2p + k) = \sum_{i=0}^{15} W_{1024}^{(64l+2p+k) \cdot i} \cdot T(p, k, i); l = 0..15, p = 0..31, k = 0..1$$

Таким образом, процедура вычисления БПФ состоит из трех этапов:

1. Нахождение массива S длиной 1024 элемента

$$\begin{aligned} S(k, j, i) &= \sum_{n=0}^1 W_{1024}^{512k \cdot n} \cdot X(512n + 16j + i) = \\ &= \sum_{n=0}^1 (-1)^{k \cdot n} \cdot X(512n + 16j + i); k = 0..1, i = 0..15, j = 0..15 \end{aligned}$$

2. Нахождение массива T длиной 1024 элемента

$$T(p, k, i) = \sum_{j=0}^{31} W_{1024}^{16(2p+k) \cdot j} \cdot S(k, j, i); p = 0..31, k = 0..1, i = 0..15$$

3. Нахождение результирующего массива Y длиной 1024 элемента

$$Y(64l + 2p + k) = \sum_{i=0}^{15} W_{1024}^{(64l+2p+k) \cdot i} \cdot T(p, k, i); l = 0..15, p = 0..31, k = 0..1$$

3.3.2 Схема вычислений в функции NMFFT1024

Схема вычислений аналогична функции NMFFT512, см. стр. 3-5

3.4 Реализации прямого БПФ-2048

1.1.1 Алгоритм вычисления БПФ-2048 на процессоре NM6403

Дискретное 2048-точечное преобразование Фурье определяется формулой:

$$Y(k) = \sum_{n=0}^{2047} W_{2048}^{k \cdot n} \cdot X(n), k = 0..2047 \quad (4.1)$$

$$\text{где } W^k = W_{2048}^k = \exp\left(-\frac{2\pi \cdot i \cdot k}{2048}\right) \quad (4.2)$$

Преобразуем формулу (4.1) следующим образом:

$$\begin{aligned} Y(k) &= \sum_{n=0}^{2047} W_{2048}^{k \cdot n} \cdot X(n) = \\ &= \sum_{i=0}^1 W_{2048}^{k \cdot i} \cdot \sum_{n=0}^{1023} W_{2048}^{2k \cdot n} \cdot X(2n+i) = \\ &= \sum_{i=0}^3 W_{2048}^{k \cdot i} \cdot \sum_{n=0}^{511} W_{2048}^{4k \cdot n} \cdot X(4n+i) = \\ &= \sum_{i=0}^7 W_{2048}^{k \cdot i} \cdot \sum_{n=0}^{255} W_{2048}^{8k \cdot n} \cdot X(8n+i) = \\ &= \sum_{i=0}^{15} W_{2048}^{k \cdot i} \cdot \sum_{n=0}^{127} W_{2048}^{16k \cdot n} \cdot X(16n+i) = \\ &= \sum_{i=0}^{31} W_{2048}^{k \cdot i} \cdot \sum_{n=0}^{63} W_{2048}^{32k \cdot n} \cdot X(32n+i) = \\ &= \sum_{j=0}^1 W_{2048}^{32k \cdot j} \cdot \sum_{i=0}^{31} W_{2048}^{k \cdot i} \cdot \sum_{n=0}^{31} W_{2048}^{64k \cdot n} \cdot X(64n+32j+i) = \\ &= \sum_{j=0}^3 W_{2048}^{32k \cdot j} \cdot \sum_{i=0}^{31} W_{2048}^{k \cdot i} \cdot \sum_{n=0}^{63} W_{2048}^{128k \cdot n} \cdot X(128n+32j+i) = \\ &= \sum_{j=0}^7 W_{2048}^{32k \cdot j} \cdot \sum_{i=0}^{31} W_{2048}^{k \cdot i} \cdot \sum_{n=0}^{127} W_{2048}^{256k \cdot n} \cdot X(256n+32j+i) = \\ &= \sum_{j=0}^{15} W_{2048}^{32k \cdot j} \cdot \sum_{i=0}^{31} W_{2048}^{k \cdot i} \cdot \sum_{n=0}^{255} W_{2048}^{512k \cdot n} \cdot X(512n+32j+i) = \\ &= \sum_{j=0}^{31} W_{2048}^{32k \cdot j} \cdot \sum_{i=0}^{31} W_{2048}^{k \cdot i} \cdot \sum_{n=0}^{511} W_{2048}^{1024k \cdot n} \cdot X(1024n+32j+i) = \\ &= \sum_{i=0}^{31} W_{2048}^{k \cdot i} \cdot \sum_{j=0}^{31} W_{2048}^{32k \cdot j} \cdot \sum_{n=0}^{511} W_{2048}^{1024k \cdot n} \cdot X(1024n+32j+i) = \end{aligned} \quad (4.3)$$

Обозначим правую сумму через $S(k,j,i)$:

$$S(k, j, i) = \sum_{n=0}^1 W_{2048}^{1024k \cdot i} \cdot X(1024n + 32j + i) \quad (4.4)$$

Тогда формула (4.3) запишется в виде:

$$Y(k) = \sum_{i=0}^{31} W_{2048}^{k \cdot i} \cdot \sum_{j=0}^{31} W_{2048}^{32k \cdot j} \cdot S(k, j, i); k = 0..2047 \quad (4.5)$$

Учитывая, что коэффициенты W , входящие в выражение (4.4) для $S(k,j,i)$, повторяются через каждые $k=2$:

$$\begin{aligned} W_{2048}^{1024(0i)} &= W_{2048}^{1024(2i)} = W_{2048}^{1024(4i)} = \dots \\ W_{2048}^{1024(1i)} &= W_{2048}^{1024(3i)} = W_{2048}^{1024(5i)} = \dots \end{aligned} \quad (4.6)$$

то суммы $S(k,j,i)$ также будут иметь период повторения равный 2, т.е. переопределив переменную $k(k=0..2047)$ на $2p+k$ ($k=0..1, p=0..1023$), получим:

$$S(k, j, i) = S(2p + k, j, i); k = 0..1, p = 0..1023 \quad (4.7)$$

Следовательно, формулу для вычисления коэффициентов Фурье можно переписать в виде:

$$Y(2p + k) = \sum_{i=0}^{31} W_{2048}^{(2p+k) \cdot i} \cdot \sum_{j=0}^{31} W_{2046}^{32(2p+k) \cdot j} \cdot S(k, j, i); k = 0..1, p = 0..1023 \quad (4.8)$$

Обозначим правую сумму через $T(p,k,i)$:

$$T(p, k, i) = \sum_{j=0}^{31} W_{1024}^{16(2p+k) \cdot j} \cdot S(k, j, i) \quad (4.9)$$

Тогда формула (4.8) запишется в виде:

$$Y(2p + k) = \sum_{i=0}^{31} W_{2048}^{32(2p+k) \cdot i} \cdot T(p, k, i); k = 0..1, p = 0..1023 \quad (4.10)$$

Так как коэффициенты W , входящие в выражение (4.4) для $T(p,k,i)$, повторяются через каждые $p=32$:

$$W_{2048}^{32(2p+k)i} = W_{1024}^{32(2(p+32)+k)i} \quad (4.11)$$

Приложение

то суммы $T(p,k,i)$ также будут иметь период повторения равный $p=32$, т.е. переопределив переменную $p(p=0..1023)$ на $32l+p(l=0..31, p=0..31)$:

$$T(p, k, i) = T(32l + p, k, i); l = 0..31, p = 0..31, k = 0..1, \quad (4.12)$$

Следовательно, формулу для вычисления коэффициентов Фурье (4.10) можно переписать в виде:

$$Y(64l + 2p + k) = \sum_{i=0}^{31} W_{2048}^{(64l+2p+k) \cdot i} \cdot T(p, k, i); l = 0..31, p = 0..31, k = 0..1$$

Таким образом, процедура вычисления БПФ состоит из трех этапов:

1. Нахождение массива S длиной 2048 элемента

$$\begin{aligned} S(k, j, i) &= \sum_{n=0}^1 W_{2048}^{1024k \cdot n} \cdot X(1024n + 32j + i) = \\ &= \sum_{n=0}^1 (-1)^{k \cdot n} \cdot X(1024n + 32j + i); k = 0..1, i = 0..31, j = 0..31 \end{aligned}$$

2. Нахождение массива T длиной 2048 элемента

$$T(p, k, i) = \sum_{j=0}^{31} W_{2048}^{32(2p+k) \cdot j} \cdot S(k, j, i); p = 0..31, k = 0..1, i = 0..31$$

3. Нахождение результирующего массива Y длиной 2048 элемента

$$Y(64l + 2p + k) = \sum_{i=0}^{31} W_{2048}^{(64l+2p+k) \cdot i} \cdot T(p, k, i); l = 0..31, p = 0..31, k = 0..1$$

3.4.2 Схема вычислений в функции NMFFT2048

Схема вычислений аналогична функции NMFFT512, см. стр. 3-5

3.5 Реализация обратного БПФ-256

3.5.1 Алгоритм вычисления обратного БПФ-256 на процессоре NM6403

Обратное 256-точечное преобразование Фурье определяется формулой:

$$Y(k) = \frac{1}{256} \sum_{n=0}^{255} W_{256}^{k \cdot n} \cdot X(n), k = 0..255 \quad (5.1)$$

где $W_{256}^k = \exp\left(\frac{2\pi \cdot i \cdot k}{256}\right)$ (5.2)

С учетом отличий формул (5.1),(5.2) от (1.1) и (1.2) все преобразования при выводе алгоритма аналогичны приведенным для прямого БПФ-256.

3.5.2 Схема вычислений в функции NMIFFT256

Так как в формулу (5.1) входит коэффициент $1/256$, то для его учета, а так же для исключения возможности переполнения (см. п. 3.1.2), кроме заключительной нормализации можно использовать промежуточную нормализацию. Обе процедуры нормализации выполняют арифметический сдвиг на число бит, указанное с помощью параметров *ShiftR1* и *ShiftR2* функции NMIFFT256. По умолчанию параметры *ShiftR1*=8, *ShiftR2*=14, что приближенно соответствует делению на $256 \cdot 127^2$

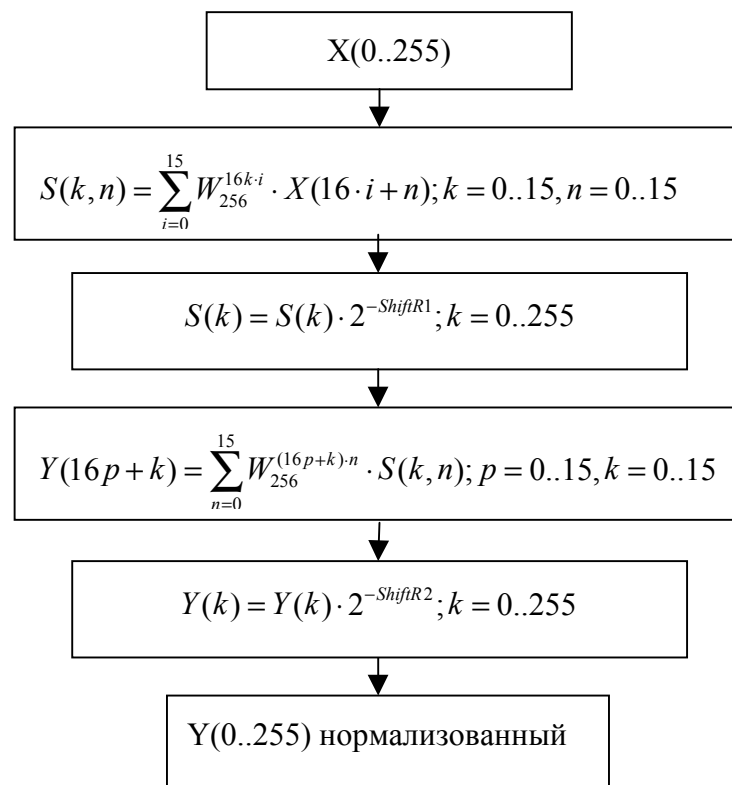


Рис 3 Блок-схема функции NMIFFT256

3.6 Реализация обратного БПФ-512

3.6.1 Алгоритм вычисления обратного БПФ-512 на процессоре NM6403

Обратное 512-точечное преобразование Фурье определяется формулой:

$$Y(k) = \frac{1}{512} \sum_{n=0}^{511} W_{512}^{k \cdot n} \cdot X(n), k = 0..511 \quad (6.1)$$

$$\text{где } W_{512}^k = \exp\left(\frac{2\pi \cdot i \cdot k}{512}\right) \quad (6.2)$$

С учетом отличий формул (6.1), (6.2) от (2.1) и (2.2) все преобразования при выводе алгоритма аналогичны приведенным для прямого БПФ-512.

3.6.2 Схема вычислений в функции NMIFFT512

Так как в формулу (6.1) входит коэффициент $1/512$, то для его учета, а так же для исключения возможности переполнения (см. п.3.2.2), кроме заключительной нормализации можно использовать промежуточную нормализацию. Обе процедуры нормализации выполняют арифметический сдвиг на число бит, указанное с помощью параметров *ShiftR1* и *ShiftR2* функции NMIFFT512. По умолчанию параметры *ShiftR1*=11, *ShiftR2*=25, что приближенно соответствует делению на $512 \cdot 127^2$

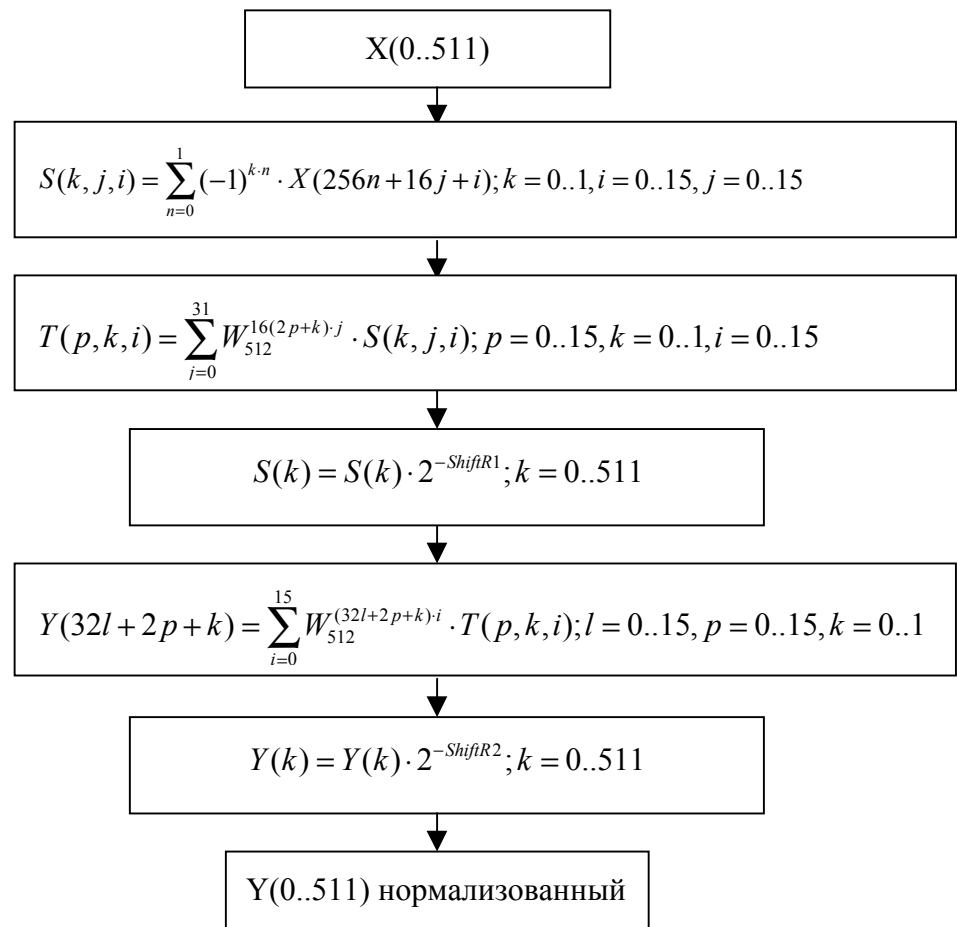


Рис 4 Блок-схема функции NMIFFT512

3.7 Реализация обратного БПФ-1024 на процессоре NM6403

3.7.1 Алгоритм вычисления обратного БПФ-1024 для процессора NM6403

Обратное 1024-точечное преобразование Фурье определяется формулой:

$$Y(k) = \frac{1}{1024} \sum_{n=0}^{1023} W_{1024}^{k \cdot n} \cdot X(n), k = 0..1023 \quad (7.1)$$

$$\text{где } W_{1024}^k = \exp\left(\frac{2\pi \cdot i \cdot k}{1024}\right) \quad (7.2)$$

С учетом отличий формул (6.1), (6.2) от (2.1) и (2.2) все преобразования при выводе алгоритма аналогичны приведенным для прямого БПФ-1024.

3.7.2 Схема вычислений в функции NMIFFT1024

Схема вычислений аналогична схеме функции NMIFFT512 (см. п.3.6.2)

3.8 Реализация обратного БПФ-2048

3.8.1 Алгоритм вычисления обратного БПФ-2048 на процессоре NM6403

Обратное 2048-точечное преобразование Фурье определяется формулой:

$$Y(k) = \frac{1}{2048} \sum_{n=0}^{2047} W_{2048}^{k \cdot n} \cdot X(n), k = 0..2047 \quad (8.1)$$

$$\text{где } W_{2048}^k = \exp\left(\frac{2\pi \cdot i \cdot k}{2048}\right) \quad (8.2)$$

С учетом отличий формул (8.1), (8.2) от (4.1) и (4.2) все преобразования при выводе алгоритма аналогичны приведенным для прямого БПФ-2048.

3.8.2 Схема вычислений в функции NMIFFT1024

Схема вычислений аналогична схеме функции NMIFFT512 (см. п.3.6.2)



**АКЦИОНЕРНОЕ ОБЩЕСТВО
НАУЧНО-ТЕХНИЧЕСКИЙ ЦЕНТР**

Научно-технический центр Модуль

АЯ 166, Москва, 125190, Россия

Тел: +7 (095) 152-9335

Факс: +7 (095) 152-4661

E-Mail: info@module.ru

WWW: <http://www.module.ru>

©НТЦ Модуль, 2000

Все права защищены

Никакая часть информации, приведенная в данном документе, не может быть адаптирована или воспроизведена, кроме как согласно письменному разрешению владельцев авторских прав.

НТЦ Модуль оставляет за собой право производить изменения как в описании, так и в самом продукте без дополнительных уведомлений. НТЦ Модуль не несет ответственности за любой ущерб, причиненный использованием информации в данном описании, ошибками или недосказанностью в описании, а также путем неправильного использования продукта.