БИБЛИОТЕКА ЭЛЕМЕНТАРНЫХ МАТЕМАТИЧЕСКИХ ФУНКЦИЙ LibNVCom02EMF

РУКОВОДСТВО ПРОГРАММИСТА

01.06.2012

СОДЕРЖАНИЕ

1. Описание библиотеки

1.1. Математические функции для формата 24E8

В состав прикладной библиотеки входит 21 математическая функция для формата чисел с плавающей точкой (24E8). Каждая функция содержится в отдельном файле.

1.1.1. Перечень функций

- abs – функция вычисления модуля числа с плавающей точкой;
- asin – функция вычисления арксинуса числа;
- atan – функция вычисления арктангенса числа;
- atan2 – функция вычисления арктангенса частного от двух чисел;
- ceil – округление к ближайшему большему;
- cos – функция вычисления косинуса;
- ctan – функция вычисления котангенса;
- exp – вычисление экспоненты;
- exp2 – вычисление значения степени двойки;
- floor – округление к ближайшему меньшему;
- frexp – разделение числа с плавающей точкой на порядок и дробную часть (мантиссу);
- ldexp – объединение порядка и дробной части в число с плавающей точкой;
- log – функция вычисления натурального логарифма;
- modf – функция вычисления дробной части числа с плавающей точкой;
- pow – возведение в степень;
- recip – функция вычисления обратного от числа;
- sin – функция вычисления синуса числа;
- sqrt – функция извлечения корня квадратного из числа;
- sqrt_recip – вычисление обратного от корня квадратного из числа;
- tan – функция вычисления тангенса.

1.1.2. abs

Функция abs вычисляет модуль числа с плавающей точкой.
Аргументы: R0.L – число в формате с плавающей точкой A.
Возврат: R0.L – модуль A.

1.1.3. asin

Функция asin вычисляет арксинус числа с плавающей точкой. Вычисление осуществляется с помощью аппроксимационного полинома.
Использование памяти: для вычисления арксинуса в памяти XYRAM размещается массив коэффициентов аппроксимационного полинома *asin_f_data* размером в 16 32-разрядных слов;
Аргументы: R0.L – число в формате с плавающей точкой A.
Возврат: R0.L – арксинус числа A.

1.1.4. atan

Функция atan вычисляет арктангенс числа с плавающей точкой. Вычисление осуществляется с помощью аппроксимационного полинома.
Использование памяти: для вычисления арктангенса в памяти XYRAM размещается массив коэффициентов аппроксимационного полинома ***atan_f_data*** размером в 9 32-разрядных слов;
Аргументы: R0.L – число в формате с плавающей точкой A.
Возврат: R0.L – арктангенс числа A.

1.1.5. atan2

Функция atan2 вычисляет арктангенс отношения чисел A и B в формате с плавающей точкой. Вычисление арктангенса осуществляется с помощью аппроксимационного полинома.
Использование памяти: для вычисления арктангенса в памяти XYRAM размещается массив коэффициентов аппроксимационного полинома ***atan2_f_data*** размером в 9 32-разрядных слов;
Аргументы: R0.L – число в формате с плавающей точкой A.
R2.L – число в формате с плавающей точкой B.
Возврат: R0.L – арктангенс числа A/B.

1.1.6. ceil

Функция ceil округляет число с плавающей точкой в большую сторону.
Аргументы: R0.L – число с плавающей точкой A.
Возврат: R0.L – округление числа A.

1.1.7. cos

Функция cos вычисляет косинус числа с плавающей точкой. Вычисление осуществляется с помощью аппроксимационного полинома.
Использование памяти: для вычисления косинуса в памяти XYRAM размещается массив коэффициентов аппроксимационного полинома ***cos_f_data*** размером в 8 32-разрядных слов;
Аргументы: R0.L – число в формате с плавающей точкой A.
Возврат: R0.L – косинус числа A.

1.1.8. ctan

Функция ctan вычисляет котангенс числа с плавающей точкой. Вычисление осуществляется с помощью аппроксимационного полинома.
Использование памяти: для вычисления котангенса в памяти XYRAM размещается массив коэффициентов аппроксимационного полинома ***ctan_f_data*** размером в 8 32-разрядных слов;
Аргументы: R0.L – число в формате с плавающей точкой A.
Возврат: R0.L – котангенс числа A.

1.1.9. exp

Функция exp вычисляет экспоненту числа с плавающей точкой. Вычисление осуществляется с помощью аппроксимационного полинома.
Использование памяти: для вычисления экспоненты в памяти XYRAM размещается массив коэффициентов аппроксимационного полинома exp***_f_data*** размером в 14 32-разрядных слов;
Аргументы: R0.L – число в формате с плавающей точкой A.
Возврат: R0.L – экспонента числа A.

1.1.10. exp2

Функция exp2 вычисляет значение от числа 2, возведенного в степень A, где A - число с плавающей точкой. Вычисление осуществляется с помощью аппроксимационного полинома.
Использование памяти: для вычисления экспоненты в памяти XYRAM размещается массив коэффициентов аппроксимационного полинома exp2*_f_data* размером в 7 32-разрядных слов;
Аргументы: R0.L – число в формате с плавающей точкой A.
Возврат: R0.L – 2^A.

1.1.11. floor

Функция floor округляет число с плавающей точкой в меньшую сторону.
Аргументы: R0.L – число с плавающей точкой A.
Возврат: R0.L – округление числа A.

1.1.12. frexp

Функция frexp разделяет число с плавающей точкой на порядок и дробную часть (мантиссу).
Аргументы: R0.L – число с плавающей точкой A.
Возврат: R0.L – порядок числа A;
R2.L – дробная часть числа A.

1.1.13. ldexp

Функция ldexp формирует число с плавающей точкой по заданному порядку E и дробной части F.
Аргументы: R0.L – порядок E (8 младших разрядов).
R2.L – дробная часть F (23 младших разряда и знак в 31-м разряде).
Возврат: R0.L – сформированное число в формате с плавающей точкой.

1.1.14. log

Функция log вычисляет логарифм натуральный от числа с плавающей точкой. Вычисление осуществляется с помощью аппроксимационного полинома.
Использование памяти: для вычисления логарифма в памяти XYRAM размещается массив коэффициентов аппроксимационного полинома *log_f_data* размером в 15 32-разрядных слов;
Аргументы: R0.L – число в формате с плавающей точкой A.
Возврат: R0.L – логарифм натуральный от числа A.

1.1.15. modf

Функция modf вычисляет дробную часть числа с плавающей точкой.
Аргументы: R0.L – число в формате с плавающей точкой A.
Возврат: R0.L – дробная часть A.

1.1.16. pow

Функция pow возводит заданное число A в степень B. Возведение в степень осуществляется с помощью аппроксимационного полинома.
Использование памяти: для вычисления в памяти XYRAM размещается массив коэффициентов аппроксимационного полинома *pow_f_data* размером в 12 32-разрядных слов;
Аргументы: R0.L – число в формате с плавающей точкой A.
R2.L – число в формате с плавающей точкой B.
Возврат: R0.L – A^B.

1.1.17. recip

Функция recip вычисляет обратное от числа с плавающей точкой. Для вычисления используется операция первого приближения к обратному FIN, а затем – последовательное увеличение точности по формуле Ньютона-Рафсона.
Аргументы: R0.L – число с плавающей точкой A.
Возврат: R0.L – 1/A.

1.1.18. sin

Функция sin вычисляет синус числа с плавающей точкой. Вычисление осуществляется с помощью аппроксимационного полинома.
Использование памяти: для вычисления синуса в памяти XYRAM размещается массив коэффициентов аппроксимационного полинома ***sin_f_data*** размером в 8 32-разрядных слов;
Аргументы: R0.L – число в формате с плавающей точкой A.
Возврат: R0.L – синус числа A.

1.1.19. sqrt

Функция sqrt вычисляет корень квадратный из числа с плавающей точкой. Для вычисления используется операция первого приближения к обратному от корня FINR, а затем – последовательное увеличение точности по формуле Ньютона-Рафсона.
Аргументы: R0.L – число с плавающей точкой A.
Возврат: R0.L – $A^{½}$.

1.1.20. sqrt_recip

Функция sqrt_recip вычисляет обратное от корня квадратного из числа с плавающей точкой. Для вычисления используется операция первого приближения к обратному от корня FINR, а затем – последовательное увеличение точности по формуле Ньютона-Рафсона.
Аргументы: R0.L – число с плавающей точкой A.
Возврат: R0.L – $A^{-½}$.

1.1.21. tan

Функция tan вычисляет тангенс числа с плавающей точкой. Вычисление осуществляется с помощью аппроксимационного полинома.
Использование памяти: для вычисления тангенса в памяти XYRAM размещается массив коэффициентов аппроксимационного полинома ***tan_f_data*** размером в 8 32-разрядных слов;
Аргументы: R0.L – число в формате с плавающей точкой A.
Возврат: R0.L – тангенс числа A.

2. Использование библиотеки

2.1. Порядок вызова библиотечных функций

Для обращения к любой библиотечной функции необходимо:
- передать аргументы функции в регистр R0.L и в R2.L (если нужно);
- вызвать функцию с помощью инструкции BS;
- считать результат (возврат) функции из регистра R0.L.

2.2. Соглашения о работе функций

При разработке библиотеки функций для стандартизации вызовов использовались следующие соглашения о вызовах:
- в работе функций задействуются только регистры A7, M7, а также R0 – R24, включительно;
- аргументы функции и ее возврат всегда передаются через регистры R0.L и R2.L.

2.3. Имена функций

Для использования библиотеки в каждом из двух DSP-ядер NVCom02, каждая функция имеет по два символических имени с постфиксами «_0» и «_1». На разных DSP-ядрах необходимо использовать наборы имен с разными постфиксами. Список имен функций приведен ниже:

Имя функции	**Входные параметры**	**Назначение**
abs_f_0, abs_f_1	R0	Вычисляет модуль числа: abs(R0) **à** R0
asin_f_0, asin_f_1	R0	Выполняет операцию arcsin(R0) **à** R0
atan_f_0, atan_f_1	R0	Выполняет операцию arctan(R0) **à** R0
atan2_f_0, atan2_f_1	R0,R2	Выполняет операцию arctan(R0/R2) **à** R0
ceil_f_0, ceil_f_1	R0	Возвращает значение, представляющее наименьшее целое, которое больше или равно аргументу R0. Результат располагается в R0
cos_f_0, cos_f_1	R0	Выполняет операцию cos(R0) **à** R0
ctan_f_0, ctan_f_1	R0	Выполняет операцию ctan(R0) **à** R0
div_f_0, div_f_1	R0,R2	Выполняет операцию (R0/R2) **à** R0
exp_f_0, exp_f_1	R0	Выполняет операцию e^{R0} **à** R0
exp2_f_0,exp2_f_1	R0	Выполняет операцию 2^{R0} **à** R0
fixsfsi_0, fixsfsi_1	R0	Выполняет преобразование R0 **à** (int)R0, где аргумент имеет тип float
fixunssfsi_0, fixunssfsi_1	R0	Выполняет преобразование R0 **à** (unsigned int)R0, где аргумент имеет тип float
floor_f_0, floor_f_1	R0	Возвращает значение, представляющее наибольшее целое, которое меньше или равно аргументу R0. Результат помещается в R0
frexp_f_0, frexp_f_1	R0	Возвращает значения m,e, такие,что arg = m * 2^E .m помещается в R0,E – в R2
ldexp_f_0, frexp_f_1	R0,R2	Выполняет операцию R0×2^{R2} **à** R0

Имя функции	Входные параметры	Назначение
log_f_0, log_f_1	R0	Выполняет операцию log(R0) **à** R0
modf_f_0, modf_f_1	R0	Возвращает x = floor(R0), y = R0-floor(R0). x помещается в R0,y в R2
pow_f_0, pow_f_1	R0,R2	Выполняет операцию $R0^{R2}$ **à** R0
recip_f_0, recip_f_1	R0	Выполняет операцию 1/R0 **à** R0
sin_f_0, sin_f_1	R0	Выполняет операцию sin(R0) **à** R0
sqrt_f_0, sqrt_f_1	R0	Выполняет операцию sin(R0) **à** R0
sqrt_recip_f_0, sqrt_recip_f_1	R0	Выполняет операцию 1/sqrt(R0) **à** R0
tan_f_0, tan_f_1	R0	Выполняет операцию tan(R0) **à** R0

Для вызова функции в программе используется команда обращения к подпрограмме:

```
BS   <имя функции>
```

2.4. Подключение библиотеки

В настройках проекта необходимо в строке вызова линкера DSP-модуля добавить параметр «-L» с указанием пути к библиотеке и параметр «-l» с указанием имени библиотеки (стандартное начало имени файла «lib» в этом случае сокращается до «-l»).

Для подключения файла библиотеки LibNVCom02EFM, расположенного в корневой директории проекта, эта строка будет выглядеть так:

```
   elcore-elvis-elf-ld -e 0 -t -N -T %Unit.xl %Files.o -o %Unit.o –L .
–lNVCom02EFM.a
```

Это значит, что библиотека с именем «libNVCom02EFM.a» находится в текущей директории (символ «.» в файловой системе обозначает текущую директорию).

После этого необходимо скорректировать скрипт линковщика. Нужно скопировать автоматически созданный средой разработки скрипт в файл с видоизмененным названием (например, файл «DSP0.xl» – в «_DSP0.xl») – и редактировать уже эту копию, так как автоматически созданный скрипт пересоздается при каждой сборке среды. Вызов линковщика будет выглядеть для приведенного примера так:

```
   elcore-elvis-elf-ld -e 0 -t -N -T _%Unit.xl %Files.o -o %Unit.o –L
. –lemf12.a
```

В копию скрипта надо вписать имена файлов функций библиотеки. Пример скорректированного скрипта приведен ниже. testrun_dsp0 и testrun_dsp1 – это файлы проекта, в которых осуществляется запуск функций библиотеки. efm0 и efm1 – это содержащиеся в библиотеке файлы, в которых собраны все библиотечные функции для ядер 0 и 1 соответственно.

```
SECTIONS {
 _dsp_LMA_ = . ;

 .DSP0_text_text 0x00000000 : AT(_dsp_LMA_ )
  {
    testrun_dsp0.*(.text);
    efm0.*(.text);
  }

 .DSP1_text_text 0x00000000 : AT(_dsp_LMA_  + SIZEOF(.DSP0_text_text))
  {
    testrun_dsp1.*(.text);
    efm1.*(.text);
  }

 .DSP0_data_data 0x00000000 : AT(_dsp_LMA_  + SIZEOF(.DSP0_text_text) +
SIZEOF(.DSP1_text_text))
  {
    testrun_dsp0.*(.data);
    efm0.*(.data);
  }
 .DSP0_data_bss :
  {
    *(COMMOM)
  }

 .DSP1_data_data 0x00000000 : AT(_dsp_LMA_  + SIZEOF(.DSP0_text_text) +
SIZEOF(.DSP1_text_text) + SIZEOF(.DSP0_data_data ))
  {
    testrun_dsp1.*(.data);
    efm1.*(.data);
  }
 .DSP1_data_bss :
  {
    *(COMMOM)
  }
 }
```