C28x Control Law Accelerator

Math Library

(CLAmath)

Module User's Guide (SPRC993)

C28x Foundation Software

V3.00

September 13, 2010

TEXAS INSTRUMENTS

Contents

1. Introduction

The Texas Instruments TMS320C28x Control Law Accelerator math library is a collection of optimized floating-point math functions for controllers with the CLA. This source code library includes CLA assembly macros of selected floating-point math functions. All source code is provided so it can be modified for your particular requirements.

The basic math library is a collection of simple floating-point arithmetic operations designed to allow the first time CLA user to write simple routines quickly and effectively without having to worry about the pipeline architecture. It has arithmetic operations like add, subtract and multiply, type conversion operations from IQ24 to floating point and vice versa, and a conditional statement macro akin to an if-else statement in C. The examples provided with this library illustrate how the macros are used and how users can leverage them in their code.

2. Other Resources

There is a live Wiki page for answers to CLA frequently asked questions (FAQ). Links to other CLA references such as training videos will be posted here as well.

> http://tiexpressdsp.com/index.php/Category:Control_Law_Accelerator_Type0

Also check out the TI Piccolo page:

> http://www.ti.com/piccolo

And don't forget the TI community website:

> http://e2e.ti.com/

Building CLA code requires Codegen Tools V5.2 or later.

Debugging in Code Composer Studio V4:

> Debugging CLA code requires CCS V4.1.3 or later. V4.1.3 can be installed by going to the update manager and upgrading your CCS 4.0 install.

Debugging in Code Composer Studio V3.3:

> C2000 evaluation version 3.3.83.19 or later supports CLA debug.   For updating other versions of CCS, please check the Wiki and community website for updates.

3. Installing the Macro Library

3.1. Where the Files are Located (Directory Structure)

As installed, the *C28x CLAmath and CLAmathBasic Libraries are* partitioned into a well-defined directory structure. By default, the library and source code is installed into the following directory:

C:\TI\controlSUITE\libs\math\CLAmath

Table 1 describes the contents of the main directories used by library:

Table 1. C28x CLAmath Library Directory Structure

Directory	Description
<base>	Base install directory. By default this is C:\TI\controlSUITE\libs\math\CLAmath\v300 For the rest of this document <base> will be omitted from the directory names.
<base>\doc	Documentation including the revision history from any previous release.
<base>\lib	The macro library include file. The macro library header file. Lookup tables used by the macros (assembly file). The macro source files.
<base>\lnk	Custom linker command files
<base>\2803x_examples	Example code for the 2803x family of devices.
<base>\2803x_examples\examples_ccsv4	Code Composer Studio V4 based examples

4. Using the CLAmath and CLAmathBasic Macro Library

The best place to start is to open up and run some of the example programs provided. This section will walk you through the framework the examples use. Open one of the projects and follow along.

4.1. The C28x CPU and the CLA are Friends!

Also known as: How to Use Header Files to Make Sharing Easy

In each of the examples provided, you will find a header file called CLAShared.h. This file includes the header files, variables, and constants that we want both the C28x and the CLA code to know about.

If possible, open one of these examples and follow along in the CLAShared.h file as we go through some suggested content:

- IQmath: If your project uses IQmath, then include it in CLAShared.h. Now you can define variables as _iq and the CLA will also know what your GLOBAL_Q value is.

- C28x Header files: These make accessing peripherals easy and can be used in both C and assembly code. In the header file software downloads we provide a file called DSP28x_Project.h. This will include all of the peripheral header files along with some example header files with useful definitions.

- The CLAmath header file: CLAmath_type0.h. All of the symbols used by the CLAmath library are included in this header file.

- Any variables or constants that both the C28x and CLA should know about. For example, if they will both access a variable called X, you can add:

  ```
  extern float32 X;
  ```

  The variable X will be created in the C28x C environment, but now the CLA will also know about it.

- Symbols in the CLA assembly file. These might include start/end symbols for each task. The main CPU can then use these symbols to calculate the vector address for each task.

4.2. C28x C Code

Now open the main.c file to see the system setup performed by the C28x main CPU. Notice the C28x C code includes the CLAShared.h header file discussed in the previous section.

The C28x C code is responsible for

- Declaring all the C28x and CLA shared variables
- Assigning these variables to linker sections by using the CODE_SECTION #pragma statement. This will be used by the linker file to place the variables in the proper memory block or message RAM.
- Turning on the clock to the CLA
- Initializing the CLA data and program memory

  Note as the examples are provided, the CLA data and program are loaded directly by Code Composer Studio. This is a great first step while debugging CLA code. Later if you move the load address of these sections to flash, the main CPU will need to copy them to the CLA data and program memory.
- Assigning the CLA program and data memory to the CLA module. At reset these memories belong to the C28x CPU so they can be initialized like any other memory. After initialization, they can then be assigned to the CLA for use.
- Enabling CLA interrupts

4.3. CLA Assembly Code

The CLA assembly code for each example is in the file CLA.asm. Open this file and notice the following:

- The CLAShared.h header file discussed previously is included by using the assembly directive:

  ```
  .cdecls   C,LIST,"CLAShared.h"
  ```
- Depending on which macro library you plan to use, it will have to be included in the assembly file. The CLAmath macro library is included as follows

  .

  ```
  .include   "CLAmathLib_type0.inc"
  ```

  The CLAmathBasic library will have the following assembler directive

  ```
  .include   "CLAmathBasicLib_type0.inc"
  ```

  This will allow you to create instances of the macros in the CLA assembly file.
- CLA code is assigned to its own assembly section. In this case the section is called Cla1Prog. Later we will use this section in the linker command file to indicate where in the memory map the CLA code should be loaded and where it will run.

- A symbol indicates the start of CLA program memory and a symbol indicates the start of each task. These can be used by the main CPU to calculate the contents of the vector register (MVECT) for each CLA task.

- In one or more of the tasks, a CLA macro is instanced. The input and output of each macro instance are tied to variables you will create in the C28x code and reference in the CLAShared.h header file. A single task can include one or more macros. The macros themselves do not include the MSTOP instruction. Notice that an “MSTOP” instruction has been inserted to indicate the end of each task.

.

4.4. Math Lookup Tables

Many of the functions in the library use look-up tables to increase performance. These tables are located in the “CLAmathTables” assembly section and are available in the file called CLAsincosTable_type0.asm.

Make sure to add this file to your project if you are using the sin, cos or sincos macros.

4.5. CLA readable and CLA writeable memory

In several of the macros described in the CLAmathBasic library you will notice the use of the term CLA readable and CLA writeable memory. CLA readable memory consists of the CLA data RAMs (0 and 1), *CPU to CLA* message RAM and the *CLA to CPU* message RAM. CLA writeable memory only consists of the CLA data RAMs and the *CLA to CPU* message RAM. Care must be taken to not allocate any return variables to the *CPU to CLA* message RAM as the CLA can only read from this memory and not write to them. Also note that the Data RAMs have to be switched over to the CLA through a bit in a memory configuration register (MMEMCFG). Prior to switching them over, the C28x core has read/write access to them. If you plan to use these RAMs exclusively in the CLA, you should not store C28x variables in them as the main core will not be able to access them once the RAM is switched over. Refer to the CLA reference guide (SPRUGE6B) for more details

4.6. Linker File

The linker file needs to specify the load and run memory locations for the following sections:

- Cla1Prog – this contains the CLA program code. The examples load this code directly into CLA program memory for debug. Later you will want to load this section into flash and copy it to CLA program SARAM just as you would any time critical section of code.

- Cla1ToCpuMsgRAM – The examples place all of the CLA to CPU message traffic in this section using the DATA_SECTION #pragma in the main C28x code. This section must be placed in the CLA to CPU message RAM.

- CpuToCla1MsgRAM – Likewise, the examples place all of the CPU to CLA message traffic in this section using the DATA_SECTION #pragma in the main C28x code. This section must be placed in the CPU to CLA message RAM.

- Cla1Ram1–This section corresponds to the CLA Data RAM 1. Some of the local CLA variables in the basic math library examples are stored in this section using the DATA_SECTION #pragma in the main C28x code. This section must be placed in RAML2.

- CLAmathTables – This table is used by the sin, cos and sincos macros. The contents of this section should be placed in the CLA data memory at runtime. The examples load this code directly into CLA data memory for debug. Later you will want to load this section into flash and copy it to CLA data SARAM just as you would any time critical data table.

5. CLAmath Macro Summary

The following functions are included in this release of the CLAmath Library. All of the macros are provided in the CLAmathLib_type0.inc file and can be modified as required for a particular application.

TRIGONOMETRIC	
CLAcos	CLAsin
CLAsincos	CLAatan
CLAatan2	CLAatan2PU
CLAcosPU	CLAsinePU
CLAacos	CLAasin
LOGARITHIMIC	
CLAln	CLAlog10
CLAlog	
EXPONENTIAL	
CLAexp	CLAexp10
CLAexp2	CLAexpN
MISCELLANEOUS	
CLAdiv	CLAisqrt
CLAsqrt	

CLAcos	*Single-Precision Floating-Point COS (radians)*

Description Returns the cosine of a 32-bit floating-point argument “rad” (in radians) using table look-up and Taylor series expansion between the look-up table entries.

Header File

C28x C code: #include "CLAmathLib_type0.h"
CLA Code: .cdecls C,LIST,"CLAmathLib_type0.h"

Macro Description

```
CLAcos    .macro  y, rad      ; y = cos(rad)

Input:    rad = radians in 32-bit floating-point
Output: y   = cos(rad) in 32-bit floating-point
```

Lookup Tables This function requires the CLAsincosTable located in the file called CLAsincosTable_type0.asm. By default, this table is in the assembly section named CLAmathTables. Make sure this table is loaded into the CLA data space.

CLAsin

Single-Precision Floating-Point SIN (radians)

Description

Returns the sine of a 32-bit floating-point argument “rad” (in radians) using table look-up and Taylor series expansion between the look-up table entries.

Header File

C28x C code: #include "CLAmathLib_type0.h"
CLA Code: .cdecls C,LIST,"CLAmathLib_type0.h"

Macro Description

```
CLAsin   .macro  y, rad      ; y = cos(rad)

Input:   rad = radians in 32-bit floating-point
Output: y   = sin(rad) in 32-bit floating-point
```

Lookup Tables

This function requires the CLAsincosTable located in the file called CLAsincosTable_type0.asm. By default, this table is in the assembly section named CLAmathTables. Make sure this table is loaded into the CLA data space.

CLAsincos

Single-Precision Floating-Point SIN and COS (radians)

Description

Returns the sine and cosine of a 32-bit floating-point argument “rad” (in radians) using table look-up and Taylor series expansion between the look-up table entries.

Header File

C28x C code: #include "CLAmathLib_type0.h"
CLA Code: .cdecls C,LIST,"CLAmathLib_type0.h"

Macro Description

```
CLAcos   .macro  y1, y2, rad, temp1, temp2

Input:   rad = radians in 32-bit floating-point
Output: y1  = sin(rad) in 32-bit floating-point
          2  = cos(rad) in 32-bit floating-point
Temporary:      temp1, temp2.
                Note: temp1 and temp2 locations are used for temporary
                storage during the function execution.
```

Lookup Tables

This function requires the CLAsincosTable located in the file called CLAsincosTable_type0.asm. By default, this table is in the assembly section named CLAmathTables. Make sure this table is loaded into the CLA data space.

CLAatan	*Single-Precision Floating-Point ATAN (radians)*

Description Returns the arc tangent of a floating-point argument x . The return value is an angle in the range [-π, π] radians.

Header File

```
C28x C code:   #include  "CLAmathLib_type0.h"
CLA Code:      .cdecls  C,LIST,"CLAmathLib_type0.h"
```

Macro Description

```
CLAatan    .macro  y,x      ; y = atan(x)

Input:    x in 32-bit floating-point
Output: y = atan(x) in 32-bit floating-point
```

Lookup Tables This function requires the CLAatan2Table located in the file called CLAatanTable_type0.asm. By default, this table is in the assembly section named CLAmathTables. Make sure this table is loaded into the CLA data space.

CLAatan2	*Single-Precision Floating-Point ATAN2 (radians)*

Description Returns the 4-quadrant arctangent of floating-point arguments y/x. The return value is an angle in the range [-π, π] radians

Header File

```
C28x C code:   #include  "CLAmathLib_type0.h"
CLA Code:      .cdecls  C,LIST,"CLAmathLib_type0.h"
```

Macro Description

```
CLAatan2    .macro  z,y,x      ; z = atan(y/x)

Inputs:  x in 32-bit floating-point
         y in 32-bit floating-point
Output: z = atan2(y,x) in 32-bit floating-point
```

Lookup Tables This function requires the CLAatan2Table located in the file called CLAatanTable_type0.asm. By default, this table is in the assembly section named CLAmathTables. Make sure this table is loaded into the CLA data space.

CLAatan2PU	*Single-Precision Floating-Point Atan2PU (Number1,Number2)*

Description	Returns the ArctanPU of "ratio" of two 32-bit floating-point arguments x and y. The return value is an angle in the per unit range [-0.5, 0.5].
Header Files	C28x C code: #include "CLAmathLib_type0.h" CLA Code : .cdecls C,LIST,"CLAmathLib_type0.h"
Macro Description	CLAatan2 .macro z, x, y ; z = Atan2PU(x/y) Input : x & y = 32-bit floating-point number Output : z = Atan2PU(x/y) in 32-bit floating-point
Lookup Tables	This function requires the CLAatan2Table located in the file called CLAatanTable_type_assign1.asm. By default, this table is in the assembly section named CLAmathTables. Make sure this table is loaded into the CLA data space.

CLAcosPU	*Single-Precision Floating-Point Cos (radians per unit)*

Description	Returns the cosine of a 32-bit floating-point argument "radPU" (in radians per unit) using table look-up and Taylor series expansion between the look-up table entries.
Header Files	C28x C code: #include "CLAmathLib_type0.h" CLA Code : .cdecls C,LIST,"CLAmathLib_type0.h"
Macro Description	CLAcosPU .macro y, radPU ; y = cosPU(radPU) Input : radPU = 32-bit floating-point number Output : y = cosPU(radPU) in 32-bit floating-point
Lookup Tables	This function requires the CLAsincosTable located in the file called CLAsincosTable_type0.asm. By default, this table is in the assembly section named CLAmathTables. Make sure this table is loaded into the CLA data space.

CLAsinPU	*Single-Precision Floating-Point Sin (radians per unit)*

Description Returns the sine of a 32-bit floating-point argument “radPU” (in radians per unit) using table look-up and Taylor series expansion between the look-up table entries.

Header Files C28x C code: #include "CLAmathLib_type0.h"
CLA Code : .cdecls C,LIST,"CLAmathLib_type0.h"

Macro Description CLAsinPU .macro y, radPU ; y = sinPU(radPU)

Input : radPU = 32-bit floating-point number
Output : y = sinPU(radPU) in 32-bit floating-point

Lookup Tables This function requires the CLAsincosTable located in the file called CLAsincosTable_type0.asm. By default, this table is in the assembly section named CLAmathTables. Make sure this table is loaded into the CLA data space.

CLAacos	*Single-Precision Floating-Point Arccosine (Number)*

Description Returns the Acos of a 32-bit floating-point argument x in range [-1,1], using table look-up and second degree polynomial approximation between the look-up table entries. The return value is an angle in the range [0, π] radians.

Header Files C28x C code: #include "CLAmathLib_type0.h"
CLA Code : .cdecls C,LIST,"CLAmathLib_type0.h"

Macro Description CLAacos .macro y, x ; y = Arccosine(x)

Input : x = 32-bit floating-point number
Output : y = Arccosine(x) in 32-bit floating-point

Lookup Tables This function requires the CLAacosinTable located in the file called CLAacosinTable_assign1.asm. By default, this table is in the assembly section named CLAmathTables. Make sure this table is loaded into the CLA data space.

CLAasin	*Single-Precision Floating-Point Arcsine (Number)*

Description Returns the Arcsine of a 32-bit floating-point argument x in range [-1,1], using table look-up and second degree polynomial approximation between the look-up table entries. The return value is an angle in the range [-π/2, π/2] radians.

Header Files

```
C28x C code: #include "CLAmathLib_type0.h"
CLA Code   : .cdecls C,LIST,"CLAmathLib_type0.h"
```

Macro Description

```
CLAasin   .macro y, x ;    y = Arcsine(x)

Input    : x = 32-bit floating-point number
Output : y = Arcsin(x) in 32-bit floating-point
```

Lookup Tables This function requires the CLAasinTable located in the file called CLAasineTable_assign1.asm. By default, this table is in the assembly section named CLAmathTables. Make sure this table is loaded into the CLA data space.

CLAln	*Single-Precision Floating-Point Ln (Number)*

Description Returns the Natural Log of a floating-point argument x.

Header Files

```
C28x C code: #include "CLAmathLib_type0.h"
CLA Code   : .cdecls C,LIST,"CLAmathLib_type0.h"
```

Macro Description

```
CLAln   .macro y, x   ;    y = Ln(x)

Input    : x = 32-bit floating-point number
Output : y = Ln(x) in 32-bit floating-point
```

Lookup Tables This function requires the CLALnTable located in the file called nat_log.asm. By default, this table is in the assembly section named CLAmathTables. Make sure this table is loaded into the CLA data space.

CLAlog10 — *Single-Precision Floating-Point Log10 (Number)*

Description

Returns the Log to the base 10 of a floating-point argument x.

Header Files

C28x C code: #include "CLAmathLib_type0.h"
CLA Code : .cdecls C,LIST,"CLAmathLib_type0.h"

Macro Description

CLAlog10 .macro y, x ; y = Log10(x)

Input : x = 32-bit floating-point number
Output : y = Log10(x) in 32-bit floating-point

Lookup Tables

This function requires the CLALnTable located in the file called nat_log.asm. By default, this table is in the assembly section named CLAmathTables. Make sure this table is loaded into the CLA data space.

CLAlog — *Single-Precision Floating-Point Log (Number, Base)*

Description

Returns the Log of a floating-point argument A to the base of a floating point number B.

Header Files

C28x C code: #include "CLAmathLib_type0.h"
CLA Code : .cdecls C,LIST,"CLAmathLib_type0.h"

Macro Description

CLAlog .macro C, yt1, yt2, A, B ; C = Log(A, B)

Input : A & B = 32-bit floating-point number
Output : C = Log(A, B) in 32-bit floating-point

Lookup Tables

This function requires the CLALnTable located in the file called nat_log.asm. By default, this table is in the assembly section named CLAmathTables. Make sure this table is loaded into the CLA data space.

Note

The macro CLALog is a nested macro, which requires two locations {yt1 & yt2} for storing the intermediate results for the computation of Log (Number, Base).

CLAexp

Single-Precision Floating-Point Exp (Number)

Description

Returns the Exponential of a floating-point argument x in the range of [-88, 88], using table look-up and Taylor series expansion between the look-up table entries.

Header Files

C28x C code: #include "CLAmathLib_type0.h"
CLA Code   : .cdecls C,LIST,"CLAmathLib_type0.h"

Macro Description

```
CLAexp   .macro y, x   ;    y = Exp(x)

Input    : x = 32-bit floating-point number
Output : y = Exp(x) in 32-bit floating-point
```

Lookup Tables

This function requires the CLAExpTable located in the file called CLAExp_Table.asm. By default, this table is in the assembly section named CLAmathTables. Make sure this table is loaded into the CLA data space.

CLAexp10

Single-Precision Floating-Point Exp10 (Number)

Description

Returns the Exp10 of a floating-point argument x in the range of [-38,38], using table look-up and Taylor series expansion between the look-up table entries.

Header Files

C28x C code: #include "CLAmathLib_type0.h"
CLA Code   : .cdecls C,LIST,"CLAmathLib_type0.h"

Macro Description

```
CLAexp10   .macro y, x   ;    y = Exp10(x)

Input    : x = 32-bit floating-point number
Output : y = Exp10(x) in 32-bit floating-point
```

Lookup Tables

This function requires the CLAExpTable located in the file called CLAExp_Table.asm. By default, this table is in the assembly section named CLAmathTables. Make sure this table is loaded into the CLA data space.

CLAexp2 — *Single-Precision Floating-Point Exp2 (Number1,Number2)*

Description Returns the Exponential of ratio of two floating-point arguments x & y.

Header Files C28x C code: #include "CLAmathLib_type0.h"
CLA Code : .cdecls C,LIST,"CLAmathLib_type0.h"

Macro Description CLAexp2 .macro z, x, y ; z = Exp(x/y)

Input : x & y = 32-bit floating-point number
Output : z = Exp(x/y) in 32-bit floating-point

Lookup Tables This function requires the CLAExpTable located in the file called CLAExp_Table.asm. By default, this table is in the assembly section named CLAmathTables. Make sure this table is loaded into the CLA data space.

Note The Exp2 Macro is valid for the ratio of inputs {x/y} between [-88,88]

CLAexpN — *Single-Precision Floating-Point Power (Number1, Number2)*

Description Returns the Power of a floating-point argument Num, raised to the power of a floating point argument x.

Header Files C28x C code: #include "CLAmathLib_type0.h"
CLA Code : .cdecls C,LIST,"CLAmathLib_type0.h"

Macro Description CLAexpN .macro y, Num, x ; y = Num ^ x

Input : Num & x = 32-bit floating-point number
Output : y = Num ^ x in 32-bit floating-point

Lookup Tables This function requires the CLAExpTable and CLALnTable located in the file called CLAExp_Table.asm and nat_log.asm respectively. By default, these tables are in the assembly section named CLAmathTables. Make sure these tables are loaded into the CLA data space.

CLAdiv	*Single-Precision Floating-Point Division*

Description Performs a 32-bit/32-bit = 32-bit single-precision floating point division. This function uses a Newton-Raphson algorithm.

Header File
C28x C code: #include "CLAmathLib_type0.h"
CLA Code : .cdecls C,LIST,"CLAmathLib_type0.h"

Macro Description CLAdiv .macro Dest, Num, Den

Input: Num = Numerator 32-bit floating-point
Den = Denominator 32-bit floating-point
Output: Dest= Num/Den in 32-bit floating-point

Lookup Tables None

CLAisqrt	*Single-Precision Floating-Point 1.0/Square Root*

Description Returns 1.0 /square root of a floating-point argument using a Newton-Raphson algorithm.

Header File
C28x C code: #include "CLAmathLib_type0.h"
CLA Code : .cdecls C,LIST,"CLAmathLib_type0.h"

Macro Description CLAisqrt .macro y, x

Input: x = 32-bit floating-point input
Output: y = 1/(sqrt(x)) in 32-bit floating-point

Lookup Tables None

Special Cases
If x = FLT_MAX or FLT_MIN, CLAisqrt will set the LUF flag.
If x = -FLT_MIN, CLAisqrt will set both the LUF and LVF flags.
If x = 0.0, CLAisqrt sets the LVF flag.
If x is negative, CLAisqrt will set LVF and return 0.0.

CLAsqrt	*Single-Precision Floating-Point Square Root*

Description	Returns the square root of a floating-point argument X using a Newton-Raphson algorithm.
Header File	C28x C code: #include "CLAmathLib_type0.h" CLA Code : .cdecls C,LIST,"CLAmathLib_type0.h"
Macro Description	CLAsqrt .macro y, x Input: x = 32-bit floating-point input Output: y = sqrt(x) in 32-bit floating-point
Lookup Tables	None
Special Cases	If X = FLT_MAX or FLT_MIN CLAsqrt will set the LUF flag. If X = -FLT_MIN CLAsqrt will set both the LUF and LVF flags. If X = 0.0, CLAsqrt sets the LVF flag. If X is negative, CLAsqrt will set LVF and return 0.0.

6. CLAmathBasic Macro Summary

The following functions are included in this release of the CLAmathBasic Library. All of the macros are provided in the CLAmathBasicLib_type0.inc file and can be modified as required for a particular application.

Table 2. CLAmathBasicLib_type0.inc functions

BASIC	
CLAcopy	CLAadd
CLAsub	CLAmul
CLAinv	CLAsatmax
CLAsatmin	CLAsatmaxmin
CLAabs	CLAsuboffsetsatzero
CONDITIONAL EXECUTION	
CLAcondexec	
TYPE CONVERSION	
CLAIQ24toF	CLAFtoIQ24

CLAcopy	*Single Precision Floating point Copy*

Description Copies a 32-bit floating-point variable in CLA readable memory to another variable located in CLA writeable memory. This is useful when trying to copy variables between the CLA shared memories

Header File CLA Code: .cdecls C,LIST,"CLAmathBasicLib_type0.h"

Macro Description CLAcopy .macro my, mx

Input: mx = 32-bit floating-point variable
Output: my = 32-bit floating-point variable

Symbolic my = mx

CLAadd	*Single Precision Floating point Addition*

Description	Performs a 32-bit single-precision floating point addition of two variables in CLA readable memory and stores the result in another 32 bit floating point variable in CLA writeable memory.
Header File	CLA Code: .cdecls C,LIST,"CLAmathBasicLib_type0.h"
Macro Description	CLAadd .macro my,ma,mb Input: ma = First 32-bit floating-point operand mb = Second 32-bit floating-point operand Output: my = 32-bit floating-point result
Symbolic	my = ma + mb
Note	Overflows and underflows are not handled by the macro.

CLAsub	*Single-Precision Floating-Point Subtraction*

Description	Performs a 32-bit single-precision floating point subtraction of two variables in CLA readable memory and stores the result in another 32 bit floating point variable in CLA writeable memory.
Header File	CLA Code: .cdecls C,LIST,"CLAmathBasicLib_type0.h"
Macro Description	CLAsub .macro my,ma,mb Input: ma = First 32-bit floating-point operand mb = Second 32-bit floating-point operand Output: my = 32-bit floating-point result
Symbolic	my = ma - mb
Note	Overflows and underflows are not handled by the macro.

CLAmul	*Single-Precision Floating-Point Multiplication*

Description	Performs a 32-bit single-precision floating point multiplication of two variables in CLA readable memory and stores the result in another 32 bit floating point variable in CLA writeable memory.
Header File	CLA Code: .cdecls C,LIST,"CLAmathBasicLib_type0.h"
Macro Description	CLAsub .macro my,ma,mb Input: ma = First 32-bit floating-point operand mb = Second 32-bit floating-point operand Output: my = 32-bit floating-point result
Symbolic	my = ma * mb
Note	Overflows and underflows are not handled by the macro.

CLAinv	*Single-Precision Floating-Point Inverse*

Description	Returns the inverse of a 32-bit floating-point argument using an iterative Newton-Raphson's method to get a more accurate result
Header File	CLA Code: .cdecls C,LIST,"CLAmathBasicLib_type0.h"
Macro Description	CLAinv .macro my,mx Input: mx = 32-bit floating-point operand Output: my = 32-bit floating-point inverse

Symbolic	my_e = estimate(1 / mx)
	my_ee = my_e *(2.0 – my_e * mx)
	my = my_ee*(2.0 – my_ee* mx)

CLAsatmax	*Single Precision Floating point Maximum Saturation*

Description Limits a 32 bit floating point number in CLA writeable memory to its maximum value. If the number does not exceed its limit it remains unchanged

Header File CLA Code: .cdecls C,LIST,"CLAmathBasicLib_type0.h"

Macro Description CLAsatmax .macro mx,mxmax

```
Input:  mx    = 32-bit floating-point operand
        mxmax = 32-bit floating-point max limit
Output: mx    = 32-bit floating-point result
```

Symbolic mx = min(mx,mxmax)

CLAsatmin	*Single Precision Floating point Minimum Saturation*

Description Limits a 32 bit floating point number in CLA writeable memory to its minimum value. If the number is more negative than its minimum limit it is set to the limiting value. If it is greater than the limit it remains unchanged

Header File CLA Code: .cdecls C,LIST,"CLAmathBasicLib_type0.h"

Macro Description CLAsatmax .macro mx,mxmin

```
Input:  mx    = 32-bit floating-point operand
        mxmin = 32-bit floating-point min limit
Output: mx    = 32-bit floating-point result
```

Symbolic mx = max(mx,mxmin)

CLAsatmaxmin	*Single Precision Floating point Bounding Function*

Description

Limits a 32 bit floating point number in CLA writeable memory to a range specified by a minimum and maximum limit respectively. If the number exceeds the given range it will be saturated to the closest limit.

Header File

CLA Code: .cdecls C,LIST,"CLAmathBasicLib_type0.h"

Macro Description

```
CLAsatmaxmin .macro  mx,mxmax,mxmin

Input:   mx      = 32-bit floating-point operand
         mxmax = 32-bit floating-point max limit
         mxmin = 32-bit floating-point min limit
Output: mx      = 32-bit floating-point result
```

Symbolic

mx = max(min(mx,mxmax),mxmin

CLAsuboffsetsatzero	*Single Precision Floating point Offset subtraction with zero saturate*

Description

Subtract a 32 bit floating point offset in CLA readable memory from a 32 bit floating point variable in CLA writeable memory. If the difference is negative then set the result to zero.

Header File

CLA Code: .cdecls C,LIST,"CLAmathBasicLib_type0.h"

Macro Description

```
CLAsuboffsetsatzero .macro  mx,mxoffset

Input:   mx         = 32-bit floating-point operand
         mxoffset = 32-bit floating-point offset
Output: mx         = 32-bit floating-point result
```

Symbolic

mx = max(0,(mx-mxoffset))

CLAabs — *Single Precision Floating point Absolute Value*

Description Returns the absolute value of a 32 bit floating point number in CLA readable memory to another 32 bit floating point variable in CLA writeable memory

Header File CLA Code: .cdecls C,LIST,"CLAmathBasicLib_type0.h"

Macro Description CLAabs.macro my,mx

```
Input:  mx     = 32-bit floating-point operand
Output: my     = 32-bit floating-point result
```

Symbolic my = |mx|

CLAcondexec — *If-Else Conditional Branch*

Description Tests two variables (CLA readable memory) using the condition (refer to Table 3 for valid test conditions) specified. If the result of the evaluation is true the execution will branch to the first label else it branches to the location specified by the second label.

Header File CLA Code: .cdecls C,LIST,"CLAmathBasicLib_type0.h"

Macro Description CLAcondexec .macro my,mx,cnd,L1,L2

```
Input:  my = 32 bit floating-point variable
        mx =  32 bit floating-point variable
        cnd = Test condition (refer Table 3)
        L1 = Label in the CLA assembly file
        L2 = Label in the CLA assembly file
Output: None
```

Symbolic if((my *cnd* mx) == TRUE) branch L1
else branch L2

Table 3. Condition List used in CLAcondexec

Condition	Description
NEQ	Not Equal
EQ	Equal
GT	Greater Than
GEQ	Greater Than or Equal
LT	Lesser Than
LEQ	Lesser Than or Equal

CLAIQ24toF	*IQ24 to Single Precision Floating Point Conversion*

Description	Converts a 32 bit IQ24 number in CLA readable memory to a single precision 32 bit floating point number which is written to a variable in CLA writable memory.
Header File	CLA Code: .cdecls C,LIST,"CLAmathBasicLib_type0.h"
Macro Description	CLAIQ24toF .macro my,mx Input: mx = 32-bit IQ24 operand Output: my = 32-bit floating-point result
Symbolic	my = float(mx) / (2^24), where mx is IQ24

CLAFtoIQ24	*Single Precision Floating Point to IQ24 Conversion*

Description	Converts a single precision 32 bit floating point number in CLA readable memory to a 32 bit IQ24 number which is written to a variable in CLA writable memory.
Header File	CLA Code: .cdecls C,LIST,"CLAmathBasicLib_type0.h"
Macro Description	CLAFtoIQ24 .macro my,mx Input: mx = 32-bit floating-point operand Output: my = 32-bit IQ24 result
Symbolic	my = int32(mx * (2^24)), where mx is a floating point number

7. Revision History

V3.00: Major update

- Twelve optimized floating point macros performing trigonometric, exponential and logarithmic operations were added to the CLAmath library.
- Added a new macro library, CLAmathBasic, that implements 13 simple operations like basic arithmetic, type conversion and conditional statements

V2.00: Moderate update

Two more functions, atan and atan2 have been added to the list of available CLA math macros.

V1.00a: Minor update

This update is a minor update that does not change any of the source code. The changes were to prepare the package to be included in controlSUITE and improve usability in Code Composer Studio V4.

V1.00: Initial release