CLA Math Library

USER'S GUIDE

TEXAS INSTRUMENTS

CLA-MATH-LIB-UG-V4.00.01.00

Copyright

Texas Instruments
12203 Southwest Freeway
Houston, TX 77477
http://www.ti.com/c2000

TEXAS INSTRUMENTS

Revision Information

This is version V4.00.01.00 of this document, last updated on Mar 20, 2014.

Table of 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 several C callable assembly math functions. This revision of the library is meant to work with the CLA C compiler (codegen version v6.2.4 and above). All source code is provided so it can be modified to suit the user's requirements.

Chapter 2 provides a host of resources on the CLA in general, the C compiler as well as training material.

Chapter 3 describes the directory structure of the package.

Chapter 4 provides step-by-step instructions on how to integrate the library into a project and use any of the math routines.

Chapter 5 describes each function in the library.

Chapter 6 lists The performance of each of the library routines.

Chapter 7 provides a revision history of the library.

Examples are provided with this package to show the user how to integrate the library into their projects and use any of the routines. They can be found in the *examples* directory. For the current revision, all examples have been written for the *F2806x* and *F2837xD* devices and tested on their respective *controlCard* platforms. Each example has a script **"SetupDebugEnv.js"** that can be launched from the *Scripting Console* in CCS. These scripts will set-up the watch variables for the example. In some examples graphs (.graphProp) are provided; these can be imported into CCS during debug.

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://processors.wiki.ti.com/index.php/Control_Law_Accelerator_(C2000_CLA)_FAQ.

The following Wiki provides details on the C compiler for the CLA (available with codegen v6.2.4 and above): http://processors.wiki.ti.com/index.php/C2000_CLA_C_Compiler.

The same information may be found in the **F2806x Firmware Development Package Users Guide v136** and **F2837xD Firmware Development Package Users Guide v100**. Please note that although the examples provided in this package were developed for the F2806x and F2837xD devices, the library can be used on any device that has a CLA accelerator.

Also check out the TI Piccolo page: http://www.ti.com/piccolo

And don't forgete the TI community website: http://e2e.ti.com

Building the CLA library and examples requires **Codegen Tools v6.2.4 or later**

3 Library Structure

By default, the library and source code is installed into the following directory:

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

VERSION indicates the current revision of the CLAmath library. Figure. 3.1 shows the directory structure while the subsequent table 3.1 provides a description for each folder.

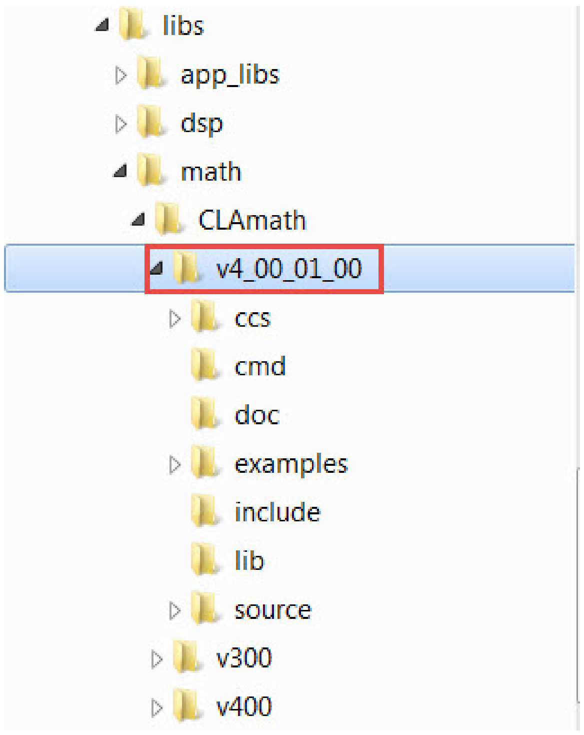


Figure 3.1: Directory Structure of the CLAMath Library

Folder	Description
<base>	Base install directory. By default this is C:/TI/controlSUITE/libs/math/CLAmath/VERSION For the rest of this document <base> will be omitted from the directory names
<base>/ccs	Project files for the library. Allows the user to reconfigure, modify and re-build the library to suit their particular needs.
<base>/cmd	Linker command files used in the examples.
<base>/doc	Documentation for the current revision of the library including revision history
<base>/examples	Examples that illustrate the library functions. At the time of writing these examples were built for the F2806x platform using CCS4 platform but they can be imported into CCS5
<base>/include	Header files for the CLAMath library
<base>/lib	Pre-built CLAMath libraries
<base>/source	Source files and project for the library. Allows the user to reconfigure, modify and re-build the library to suit their particular needs

Table 3.1: CLAMath Library Directory Structure Description

3.1 Build Options used to build the library

The cla0 math library was built with C28x Codegen Tools v6.2.4 with the following options:

```
-v28 -ml -mt --cla_support=cla0 -g --diag_warning=225
```

The cla0 math library was built with C28x Codegen Tools v6.2.4 with the following options:

```
-v28 -ml -mt --cla_support=cla1 -g --diag_warning=225
```

The fpu32 variants of the libraries required the **–fpu_support=fpu32** option enabled.

3.2 Header Files

A library header file is supplied in the <base>/include folder. This file contains coefficient, table declarations and function prototypes.

4 Using the CLA Math Library

The source code and project for the CLA math library is provided. If you import the library project into CCSv4 you will be able to view and modify the source code for all the math routines and lookup tables (see Fig. 4.1)

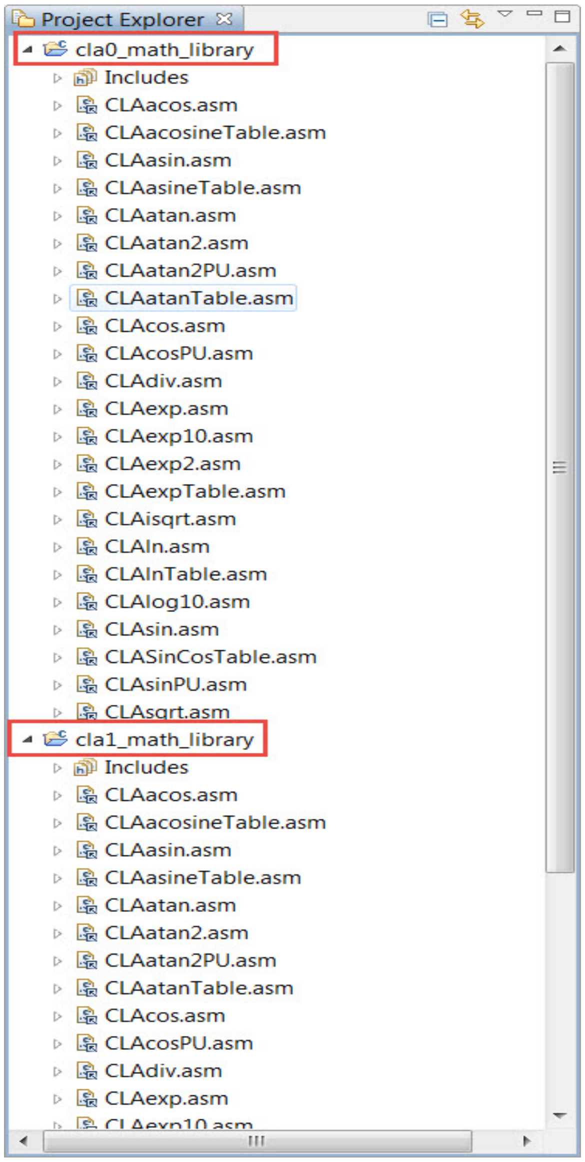


Figure 4.1: CLA Math Library Project View

4.1 Library Build Configurations

There are two libraries provided, one for the type 0 CLA and another for the type 1 CLA. Each library project has four build configurations (Fig. 4.2)

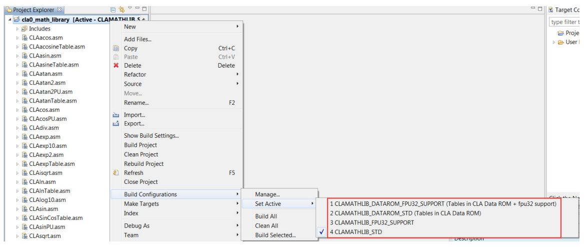
- **CLAMATHLIB_STD** - the standard build
- **CLAMATHLIB_FPU32_SUPPORT** - for devices with the hardware floating point unit turned on (projects that use the **–fpu_support=fpu32** option)
- **CLAMATHLIB_DATAROM_STD** - for devices with the lookup tables in CLA data ROM
- **CLAMATHLIB_DATAROM_FPU32_SUPPORT** - for devices with the lookup tables in CLA data ROM and the hardware floating point unit turned on (projects that use the **–fpu_support=fpu32** option)

Some devices, like the F2837x and F2805x, have all the lookup tables in a special data ROM (CLA Data ROM) which is accessible to the CLA. The user is encouraged to use the datarom variant of the library on these devices as it frees up RAM space that would otherwise have been used to store the tables.

Each build configuration, when compiled, generates the following libraries:

1. **cla0_math_library.lib** - the standard build (ISA C2800)
2. **cla0_math_library_fpu32.lib** - floating point unit supported (ISA C28xFPU32)
3. **cla0_math_library_datarom.lib** - tables in CLA data ROM (ISA C2800)
4. **cla0_math_library_datarom_fpu32.lib** - tables in CLA data ROM and floating point unit supported (ISA C28xFPU32)
5. **cla1_math_library.lib** - the standard build (ISA C2800)
6. **cla1_math_library_fpu32.lib** - floating point unit supported (ISA C28xFPU32)
7. **cla1_math_library_datarom.lib** - tables in CLA data ROM (ISA C2800)
8. **cla1_math_library_datarom_fpu32.lib** - tables in CLA data ROM and floating point unit supported (ISA C28xFPU32)

NOTE: IF YOU TRY TO LINK IN THE STANDARD BUILD LIBRARY INTO A PROJECT WHICH HAS FPU32 SUPPORT TURNED ON YOU WILL GET A COMPILER ERROR ABOUT MISMATCHING INSTRUCTION SET ARCHITECTURES, HENCE THE NEED FOR THE FPU32_SUPPORT BUILD CONFIGURATIONS



1 CLAMATHLIB_DATAROM_FPU32_SUPPORT (Tables in CLA Data ROM + fpu32 support)
2 CLAMATHLIB_DATAROM_STD (Tables in CLA Data ROM)
3 CLAMATHLIB_FPU32_SUPPORT
✓ 4 CLAMATHLIB_STD

Figure 4.2: Library Build Configurations

4.2 Examples Build Configurations

Each example has two build configurations, **FLASH** and **RAM** (Fig. 4.3)

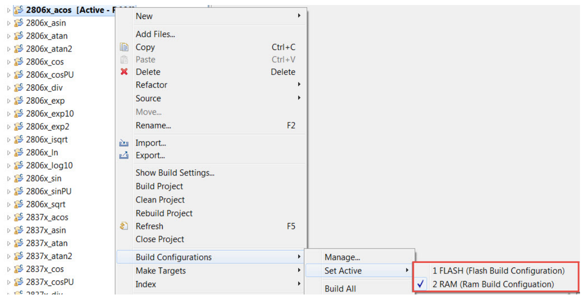


Figure 4.3: Examples Build Configurations

The **acos** example for the F2837xD has three build configurations (Fig. 4.4):

1. FLASH
2. RAM
3. FLASH_NOROM

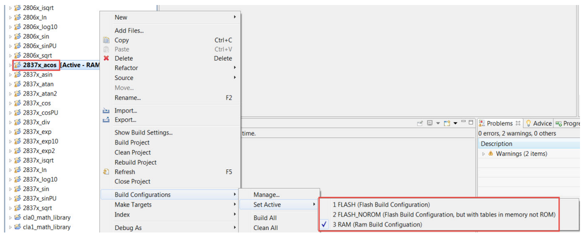

2837x_acos [Active - RAM
Build Configurations
Make Targets
Index
Debug As
Manage...
Set Active
Build All
Clean All
Problems
Advice
0 errors, 2 warnings, 0 others
Description
Warnings (2 items)
1 FLASH (Flash Build Configuration)
2 FLASH_NOROM (Flash Build Configuration, but with tables in memory not ROM)
✓ 3 RAM (Ram Build Configuation)

Figure 4.4: F2837xD acos Build Configurations

For the FLASH build, all initialized CLA sections, such as *Cla1Prog* and *CLA1mathTables*, are loaded into flash and copied over to RAM at runtime. For the F2837xD examples, both the FLASH and RAM builds use the datarom variant of the CLA math library i.e. they use the lookup tables present in the CLA data ROM.

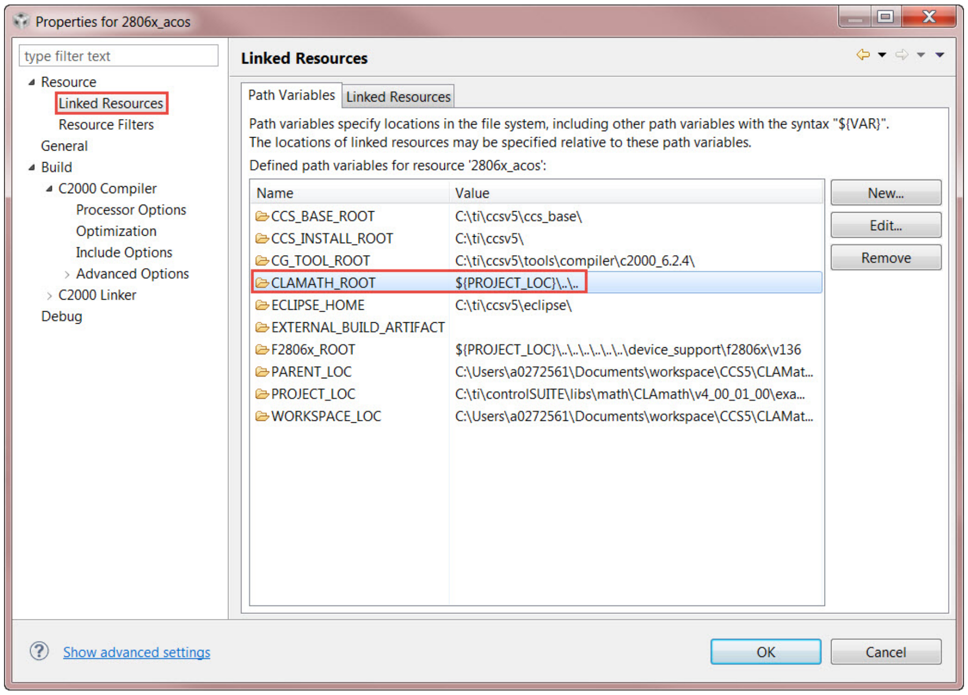
The F2837xD **acos** example has a third build configuration, FLASH_NOROM, that illustrates how to use the standard CLA math library instead of its datarom variant. For this build, the math tables are loaded into FLASH and copied to RAM at runtime.

NOTE: THE F2806X DOES NOT HAVE A CLA DATA ROM, THEREFORE, THE DATAROM VARIANT OF THE MATH LIBRARY CANNOT BE USED.

4.3 Integrating the Library into your Project

To begin integrating the library into your project you need to follow these easy steps

1. Go to **Project Properties->Linked Resources** and add a new variable (see Fig. 4.5), CLA-MATH_ROOT, and point it to the root directory of the CLA Math library in controlsuite, i.e. the version folder.

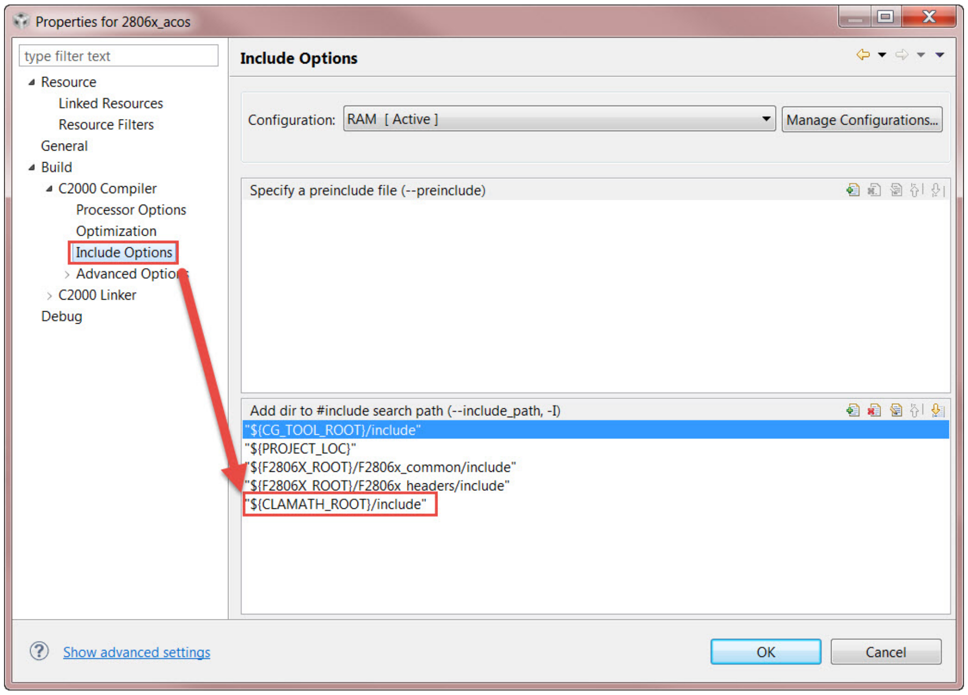


Path variables specify locations in the file system, including other path variables with the syntax "${VAR}".
The locations of linked resources may be specified relative to these path variables.
Defined path variables for resource '2806x_acos':

Name	Value
CCS_BASE_ROOT	C:\ti\ccsv5\ccs_base\
CCS_INSTALL_ROOT	C:\ti\ccsv5\
CG_TOOL_ROOT	C:\ti\ccsv5\tools\compiler\c2000_6.2.4\
CLAMATH_ROOT	${PROJECT_LOC}\..\..
ECLIPSE_HOME	C:\ti\ccsv5\eclipse\
EXTERNAL_BUILD_ARTIFACT	
F2806x_ROOT	${PROJECT_LOC}\..\..\..\..\..\device_support\f2806x\v136
PARENT_LOC	C:\Users\a0272561\Documents\workspace\CCS5\CLAMat...
PROJECT_LOC	C:\ti\controlSUITE\libs\math\CLAmath\v4_00_01_00\exa...
WORKSPACE_LOC	C:\Users\a0272561\Documents\workspace\CCS5\CLAMat...

Figure 4.5: Creating a new build variable

2. Add the new path, **CLAMATH_ROOT/include**, to the *Include Options* section of the project properties (Fig. 4.6). This option tells the compiler where to find the library header files.



Include Options

Configuration: RAM [Active] | Manage Configurations...

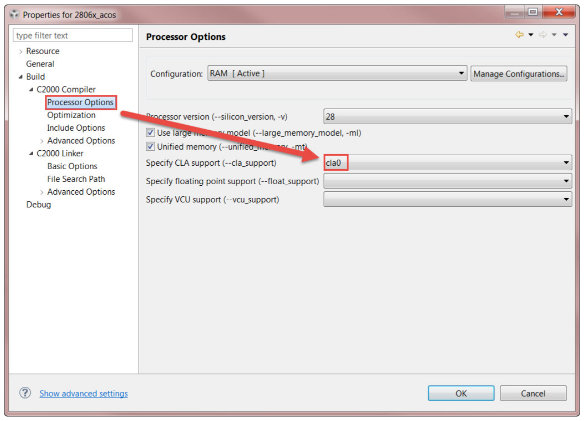
Specify a preinclude file (--preinclude)

Add dir to #include search path (--include_path, -I)

```
"${CG_TOOL_ROOT}/include"
"${PROJECT_LOC}"
"${F2806X_ROOT}/F2806x_common/include"
"${F2806X_ROOT}/F2806x_headers/include"
"${CLAMATH_ROOT}/include"
```

Figure 4.6: Adding the Library Header Path to the Include Options

3. For devices that have a Type 0 CLA, enable the **–cla_support** option in **Processor Options** to **cla0** as shown in Fig. 4.8



Processor Options

Configuration: RAM [Active] | Manage Configurations...

Processor version (--silicon_version, -v): 28

[✓] Use large memory model (--large_memory_model, -ml)

[✓] Unified memory (--unified_memory, -mt)

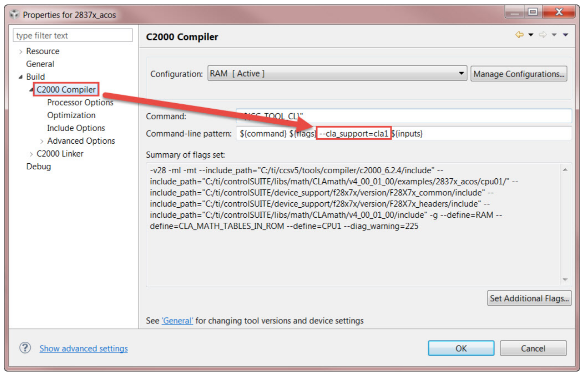
Specify CLA support (--cla_support): cla0

Specify floating point support (--float_support):

Specify VCU support (--vcu_support):

Figure 4.7: Turning on CLA support

4. For devices that have a Type 1 CLA, enable the **–cla_support** option at the compiler command-line pattern to **cla1** as shown in Fig. 4.8



Configuration: RAM [Active]

Command: "${CG_TOOL_CL}"

Command-line pattern: ${command} ${flags} --cla_support=cla1 ${inputs}

Summary of flags set:

```
-v28 -ml -mt --include_path="C:/ti/ccsv5/tools/compiler/c2000_6.2.4/include" --
include_path="C:/ti/controlSUITE/libs/math/CLAmath/v4_00_01_00/examples/2837x_acos/cpu01/" --
include_path="C:/ti/controlSUITE/device_support/f28x7x/version/F28X7x_common/include" --
include_path="C:/ti/controlSUITE/device_support/f28x7x/version/F28X7x_headers/include" --
include_path="C:/ti/controlSUITE/libs/math/CLAmath/v4_00_01_00/include" -g --define=RAM --
define=CLA_MATH_TABLES_IN_ROM --define=CPU1 --diag_warning=225
```

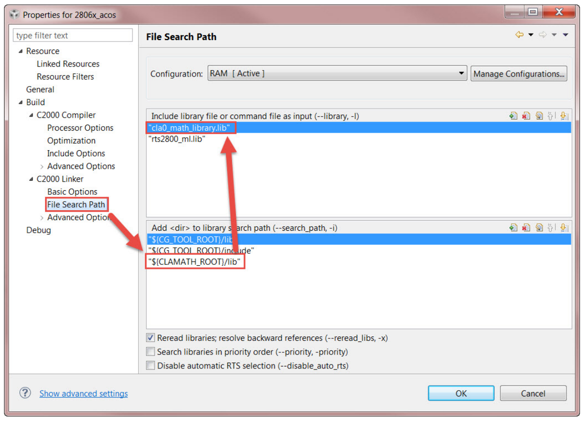
See 'General' for changing tool versions and device settings

Figure 4.8: Turning on CLA support

NOTE: AT THE TIME OF WRITING V6.2.4 DOES NOT HAVE THE CLA1 OPTION IN THE DROP DOWN MENU UNDER PROCESSOR OPTIONS AND MUST, INSTEAD, BE ADDED DIRECTLY TO THE COMMAND-LINE PATTERN.

5. Add the name of the library and its location to the **File Search Path** as shown in Fig. 4.9.

   NOTE: IF YOUR PROJECT HAS FPU32 SUPPORT TURNED ON YOU WILL NEED TO ADD THE **cla<N>_math_library_fpu32.lib** LIBRARY IN THE UPPER BOX



Configuration: RAM [Active]

Include library file or command file as input (--library, -l)

"cla0_math_library.lib"
"rts2800_ml.lib"

Add <dir> to library search path (--search_path, -i)

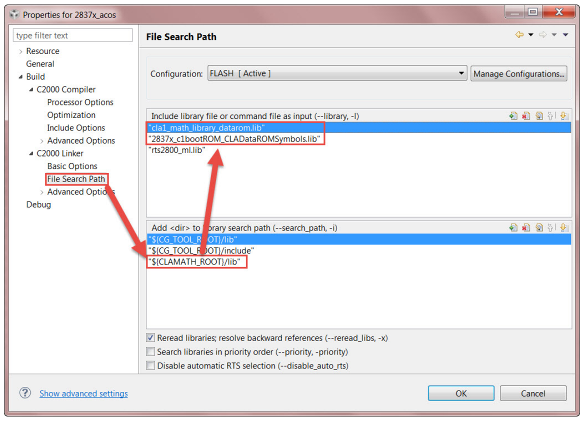
"${CG_TOOL_ROOT}/lib"
"${CG_TOOL_ROOT}/include"
"${CLAMATH_ROOT}/lib"

[✓] Reread libraries; resolve backward references (--reread_libs, -x)
[ ] Search libraries in priority order (--priority, -priority)
[ ] Disable automatic RTS selection (--disable_auto_rts)

Figure 4.9: Adding the library and location to the file search path

For devices, that have the math tables in CLA data rom, use the datarom variant of the library as shown in Fig. 4.10.

NOTE: THE USER MUST ALSO ADD THE SYMBOLS LIBRARY, e.g. **2837x_c1bootROM_CLADataROMSymbols.lib** OR **2837x_c1bootROM_CLADataROMSymbols_fpu32.lib** FOR THE 2837X DEVICE LINE, THE **2805x_CLADataROMSymbols_revA.lib** FOR THE 2805X DEVICE LINE (REVISION A) , AND ITS LOCATION TO THE FILE SEARCH PATH. THE EXAMPLE SHOWN IN THE FIGURE IS FOR CPU1 OF THE F2837XD. THE USER MUST ADD THE CORRECT SYMBOLS LIBRARY FOR THE DEVICE IN QUESTION



File Search Path

Configuration: FLASH [Active]

Include library file or command file as input (--library, -l)

"cla1_math_library_datarom.lib"
"2837x_c1bootROM_CLADataROMSymbols.lib"
"rts2800_ml.lib"

Add <dir> to library search path (--search_path, -i)

"${CG_TOOL_ROOT}/lib"
"${CG_TOOL_ROOT}/include"
"${CLAMATH_ROOT}/lib"

[✓] Reread libraries; resolve backward references (--reread_libs, -x)
[ ] Search libraries in priority order (--priority, -priority)
[ ] Disable automatic RTS selection (--disable_auto_rts)

Figure 4.10: Adding the symbols library and datarom variant of the math library to the file search path

5 Mathematical Functions

The following functions are included in this release of the CLAmath Library. The source code for these functions can be found in the *source/CLAMathLib* folder.

Trigonometric	
CLAcos	CLAsin
CLAcosPU	CLAsinePU
CLAacos	CLAasin
CLAacos_spc	CLAatan
CLAatan2	CLAatan2PU
Logarithmic	
CLAln	CLAlog10
Exponential	
CLAexp	CLAexp10
CLAexp2	
Miscellaneous	
CLAdiv	CLAisqrt
CLAsqrt	

Table 5.1: List of Functions

5.1 Arc-Cosine

Prototype:
float CLAacos(float fVal)

Parameters:
fVal Input Value ($-1 \leq fVal \leq 1$)

Returns:
Angle in radians ($0 \leq Angle \leq \pi$)

Description:
This function calculates the arc-cosine of an argument value i.e. $acos(fVal)$ or $\cos^{-1}(fVal)$, in the following manner

1. Calculate absolute of the input X
2. Use the upper 6-bits of input “X” value as an index into the table to obtain the coefficients for a second order equation
3. Calculate the angle using the following equation:

$$\begin{aligned}\cos^{-1}(Ratio) &= A0 + A1 * fVal + A2 * fVal * fVal \\ &= A0 + fVal(A1 + A2 * fVal)\end{aligned}$$

4. The final angle is determined as follows:

$$\begin{aligned}&if(X < 0) \\ &\quad Angle = Pi - Angle\end{aligned}$$

Note:
Do not use this function on a F2805x device, with the **DATAROM** variant of the CLA math library. Use the CLAacos_spc function instead.

Equation:
$\theta = cos^{-1}(fVal)$

5.2 Arc-Cosine (F2805x Specific)

Prototype:
float CLAacos_spc(float fVal)

Parameters:
fVal Input Value ($-1 \leq fVal \leq 1$)

Returns:
Angle in radians ($0 \leq Angle \leq \pi$)

Description:
This is a device specific variant of the arc-cosine function. It is meant to be used on the F2805x line of devices, if using the tables in the CLA data ROM i.e. using the **DATAROM** variants of the CLA Math library.

The arc-cosine table in the F2805x ROM has 65 triplets instead of the usual 64; this routine will skip over the first triplet and proceed with its calculations as though it were operating on a lookup table with 64 triplets. It calculates the arc-cosine of an argument value i.e. $acos(fVal)$ or $\cos^{-1}(fVal)$, in the following manner

1. Calculate absolute of the input X
2. Use the upper 6-bits of input “X” value as an index into the table to obtain the coefficients for a second order equation
3. Calculate the angle using the following equation:

$$\begin{aligned}\cos^{-1}(Ratio) &= A0 + A1 * fVal + A2 * fVal * fVal \\ &= A0 + fVal(A1 + A2 * fVal)\end{aligned}$$

4. The final angle is determined as follows:

$$\begin{aligned}&if(X < 0) \\ &\quad Angle = Pi - Angle\end{aligned}$$

Equation:
$\theta = cos^{-1}(fVal)$

5.3 Arc-Sine

Prototype:

float CLAasin(float fVal)

Parameters:

fVal Input Value ($-1 \leq fVal \leq 1$)

Returns:

Angle in radians ($-\frac{\pi}{2} \leq Angle \leq \frac{\pi}{2}$)

Description:

This function calculates the arc-sine of an argument i.e. $asin(fVal)$ or $\sin^{-1}(fVal)$ in the following manner

1. Calculate absolute of the input X
2. Use the upper 6-bits of input “X” value as an index into the table to obtain the coefficients for a second order equation
3. Calculate the angle using the following equation:

$$\begin{aligned} \sin^{-1}(Ratio) &= A0 + A1 * fVal + A2 * fVal * fVal \\ &= A0 + fVal(A1 + A2 * fVal) \end{aligned}$$

4. The final angle is determined as follows:

$$\begin{aligned} &if(X < 0) \\ &\quad Angle = -Angle \end{aligned}$$

Equation:

$\theta = \sin^{-1}(fVal)$

5.4 Arc-Tangent of a ratio

Prototype:

float CLAatan2(float fVal1, float fVal2)

Parameters:

fVal1 First Input Value (normal range of floating point values)

fVal2 Second Input Value (normal range of floating point values)

Returns:

Angle in radians ($-\pi \leq Angle \leq \pi$)

Description:

This function calculates the arc-tangent of the ratio of two input variables i.e. $atan(\frac{fVal1}{fVal2})$ or $\tan^{-1}(\frac{fVal1}{fVal2})$ in the following manner

1.

$$\begin{aligned} if(|fVal1| >= |fVal2|) \\ Numerator &= |fVal2| \\ Denominator &= |fVal1| \\ else \\ Numerator &= |fVal1| \\ Denominator &= |fVal2| \end{aligned}$$

2. Ratio = $\frac{Numerator}{Denominator}$
   NOTE: RATIO RANGE = 0.0 TO 1.0
3. Use the upper 6-bits of the "Ratio" value as an index into the table, **CLAatan2Table**, to obtain the coefficients for a second order equation
4. Calculate the angle using the following equation:

$$\begin{aligned} \tan^{-1}(Ratio) &= A0 + A1 * Ratio + A2 * Ratio * Ratio \\ &= A0 + Ratio(A1 + A2 * Ratio) \end{aligned}$$

5. The final angle is determined as follows:

$$\begin{aligned} if(fVal1 >= 0\ and\ fVal2 &>= 0\ and\ |fVal1| >= |fVal2|) \\ Angle &= arctan(\frac{|fVal2|}{|fVal1|}) \\ if(fVal1 >= 0\ and\ fVal2 &>= 0\ and\ |fVal1| < |fVal2|) \\ Angle &= PI/2 - arctan(\frac{|fVal2|}{|fVal1|}) \\ if(fVal1 < 0\ and\ fVal2 &>= 0\ and\ |fVal1| < |fVal2|) \\ Angle &= PI/2 + arctan(\frac{|fVal2|}{|fVal1|}) \\ if(fVal1 < 0\ and\ fVal2 &>= 0\ and\ |fVal1| >= |fVal2|) \\ Angle &= PI - arctan(\frac{|fVal2|}{|fVal1|}) \\ if(fVal2 < 0) \end{aligned}$$

$$Angle = -Angle$$

Equation:

$\theta = \tan^{-1}(\frac{fVal1}{fVal2})$

5.5 Arc-Tangent of a Ratio per Unit

Prototype:

float CLAatan2PU(float fVal1, float fVal2)

Parameters:

fVal1 First Input Value (normal range of floating point values)

fVal2 Second Input Value (normal range of floating point values)

Returns:

Angle per 2π radians $(-0.5 \leq Angle \leq 0.5)$

Description:

This function calculates the arc-tangent of a ratio per unit i.e. $\frac{atan(\frac{fVal1}{fVal2})}{2*\pi}$ or $\frac{\tan^{-1}(\frac{fVal1}{fVal2})}{2*\pi}$ in the following manner

1.

$$\begin{aligned} & if(|fVal1| >= |fVal2|) \\ Numerator &= |fVal2| \\ Denominator &= |fVal1| \\ & else \\ Numerator &= |fVal1| \\ Denominator &= |fVal2| \end{aligned}$$

2. $Ratio = \frac{Numerator}{Denominator}$
   NOTE: RATIO RANGE = 0.0 TO 1.0
3. Use the upper 6-bits of the "Ratio" value as an index into the table, **CLAatan2Table**, to obtain the coefficients for a second order equation
4. Calculate the angle using the following equation:

$$\begin{aligned} \tan^{-1}(Ratio) &= A0 + A1 * Ratio + A2 * Ratio * Ratio \\ &= A0 + Ratio(A1 + A2 * Ratio) \end{aligned}$$

5. The final angle is determined as follows:

$$\begin{aligned} if(fVal1 >= 0\ and\ fVal2 &>= 0\ and\ |fVal1| >= |fVal2|) \\ Angle &= arctan(\frac{|fVal2|}{|fVal1|}) \\ if(fVal1 >= 0\ and\ fVal2 &>= 0\ and\ |fVal1| < |fVal2|) \\ Angle &= PI/2 - arctan(\frac{|fVal2|}{|fVal1|}) \\ if(fVal1 < 0\ and\ fVal2 &>= 0\ and\ |fVal1| < |fVal2|) \\ Angle &= PI/2 + arctan(\frac{|fVal2|}{|fVal1|}) \\ if(fVal1 < 0\ and\ fVal2 &>= 0\ and\ |fVal1| >= |fVal2|) \\ Angle &= PI - arctan(\frac{|fVal2|}{|fVal1|}) \\ if(fVal2 < 0) \end{aligned}$$

$$\begin{aligned} Angle &= -Angle \\ AnglePU &= \frac{Angle}{2 \times \pi} \end{aligned}$$

Equation:

$$\theta_{PU} = \frac{\tan^{-1}(\frac{fVal1}{fVal2})}{2 * pi}$$

5.6 Arc-Tangent

Prototype:
float CLAatan(float fVal)

Parameters:
fVal Input Value (normal range of floating point values)

Returns:
Angle in radians ($-\frac{\pi}{2} \leq Angle \leq \frac{\pi}{2}$)

Description:
This function calculates the arc-tangent of the argument i.e. $atan(fVal)$ or $\tan^{-1}(fVal)$ in the following manner

1. $$\begin{aligned} if(1.0 \quad & >= \quad |fVal|) \\ Numerator = \; & |fVal| \\ Denominator = \; & 1.0 \\ else \quad & \\ Numerator = \; & 1.0 \\ Denominator = \; & |fVal| \end{aligned}$$

2. $Ratio = \frac{Numerator}{Denominator}$
   NOTE: RATIO RANGE = 0.0 TO 1.0
3. Use the upper 6-bits of the “Ratio” value as an index into the table, **CLAatan2Table** to obtain the coefficients for a second order equation
4. Calculate the angle using the following equation:

$$\begin{aligned} \tan^{-1}(Ratio) &= A0 + A1 * Ratio + A2 * Ratio * Ratio \\ &= A0 + Ratio(A1 + A2 * Ratio) \end{aligned}$$

5. The final angle is determined as follows:

$$\begin{aligned} if(fVal >= 0 \; and \; 1.0 \quad & >= \quad abs(fVal)) \\ Angle \quad & = \quad \tan^{-1}(\frac{abs(fVal)}{1.0}) \\ if(fVal >= 0 \; and \; 1.0 \quad & < \quad abs(fVal)) \\ Angle \quad & = \quad PI/2 - \tan^{-1}(\frac{1.0}{abs(fVal)}) \\ if(fVal < 0) \quad & \\ Angle \quad & = \quad -Angle \end{aligned}$$

Equation:
$\theta = \tan^{-1}(fVal)$

5.7 Cosine

Prototype:
float CLAcos(float fAngleRad)

Parameters:
fAngleRad Input angle in radians ($-2\pi \leq Angle \leq 2\pi$)

Returns:
cosine of the angle(float) ($-1 \leq Result \leq 1$)

Description:
This function calculates the cosine of an anlge i.e. $cos(rad)$, where rad is the input angle in radians and rad = K + X.

Using Taylor series expansion around the value K we get,

$$\begin{aligned}
cos(rad) = cos(K) &- sin(K) \times X \\
&- cos(K) \times \frac{X^2}{2!} \\
&+ sin(K) \times \frac{X^3}{3!} \\
&+ cos(K) \times \frac{X^4}{4!} \\
&- sin(K) \times \frac{X^5}{5!} \\
cos(rad) = cos(K) &+ X \times (-1.0 \times sin(K) \\
&+ X \times (-0.5 \times cos(K) \\
&+ X \times (0.166666 \times sin(K) \\
&+ X \times (0.04166666 \times cos(K) \\
&+ X \times (-0.00833333 \times sin(K)))))) \\
cos(rad) = cos(K) &+ X \times (-sin(K) \\
&+ X \times (Coef0 \times cos(K) \\
&+ X \times (Coef1\_pos \times sin(K) \\
&+ X \times (Coef2 \times cos(K) \\
&+ X \times (Coef3\_neg \times sin(K))))))
\end{aligned}$$

Equation:
$Y = cos(fAngleRad)$

5.8 Cosine Per-Unit

Prototype:
float CLAcosPU(float fAngleRadPU)

Parameters:
fAngleRadPU Input angle in radians(per 2π units) ($-1 \leq Angle \leq 1$)

Returns:
Cosine of the angle ($-1 \leq Result \leq 1$)

Description:
This function calculates the cosine of a per-unit angle i.e. $cos(radPU)$, where radPU is the angle in radians(per 2π units) and radPU= K + X

Therefore rad= $radPU * 2 * \pi$

Using Taylor series expansion around the value K we get,

$$\begin{aligned}
cos(rad) = cos(K) &- sin(K) \times X \\
&- cos(K) \times \frac{X^2}{2!} \\
&+ sin(K) \times \frac{X^3}{3!} \\
&+ cos(K) \times \frac{X^4}{4!} \\
&- sin(K) \times \frac{X^5}{5!} \\
cos(rad) = cos(K) &+ X \times (-1.0 \times sin(K) \\
&+ X \times (-0.5 \times cos(K) \\
&+ X \times (0.166666 \times sin(K) \\
&+ X \times (0.04166666 \times cos(K) \\
&+ X \times (-0.00833333 \times sin(K))))) \\
cos(rad) = cos(K) &+ X \times (-sin(K) \\
&+ X \times (Coef0 \times cos(K) \\
&+ X \times (Coef1\_pos \times sin(K) \\
&+ X \times (Coef2 \times cos(K) \\
&+ X \times (Coef3\_neg \times sin(K)))))
\end{aligned}$$

Equation:
$Y = cos(fAngleRadPU)$

5.9 Divide

Prototype:
float CLAdiv(float fNum, float fDen)

Parameters:
fNum Numerator (normal range of floating point values)

fDen Denominator (normal range of floating point values $\neq 0$)

Returns:
(float) $\frac{fNum}{fDen}$ (normal range of floating point values)

Description:
This function uses the Newton Raphson approximation to converge on the answer.

$$\begin{aligned}
Y^{'} &\approx \frac{1}{Den} \\
Y^{'} &= Y^{'} \times Den \\
Y^{''} &= Y^{'} - Y^{'} \times (2.0 - Y^{'} \times Den) \\
Y^{'''} &= Y^{''} \times Den \\
Y^{'''} &= Y^{''} - Y^{''} \times (2.0 - Y^{''} \times Den) \\
Y &= Y^{'''} \times Num
\end{aligned}$$

Equation:
$Y = \frac{fNum}{fDen}$

5.10 Exponential

Prototype:

float CLAexp(float fVal)

Parameters:

fVal Input argument (non-negative range of floating point values)

Returns:

Exponential raised to the input argument (positive range of floating point values)

Description:

This function calculates the exponential of the input argument i.e. e^x, where x is the input value. It is calculated as follows:

1. Calculate absolute of x
2. Identify the integer and mantissa of the input
3. Obtain the $e^{integer(x)}$ from the table **CLAExpTable**
4. Calculate the value of $e^{(mantissa)}$ by using the polynomial approx:

$$e^{X_m} = 1 + X_m \times (1 + X_m \times 0.5(1 + (\frac{X_m}{3}) \times (1 + \frac{X_m}{4} \times (1 + \frac{X_m}{5} \times (1 + \frac{X_m}{6} \times (1 + \frac{X_m}{7}))))))$$

5. The value of e^x is the product of results from (3) and (4)

Equation:

$Y = e^{fVal}$

5.11 Exponential of a Ratio

Prototype:
float CLAexp2(float fNum, float fDen)

Parameters:
fNum First argument (normal range of floating point values)

fDen Second argument (normal range of floating point values $\neq$ 0)

Returns:
Value of the exponential raised to the ratio of the two input arguments (positive range of floating point values)

Description:
This function calculates the exponential of the ratio of two numbers i.e. $e^{\frac{A}{B}}$, where A and B are the two input arguments. These are the steps in the calculation:

1. Calculate absolute of $x = \frac{A}{B}$
2. Identify the integer and mantissa of the input
3. Obtain the $e^{integer(x)}$ from the table **CLAExpTable**
4. Calculate the value of $e^{(mantissa)}$ by using the following polynomial approx:

$$e^{X_m} = 1 + X_m \times (1 + X_m \times 0.5(1 + (\frac{X_m}{3}) \times (1 + \frac{X_m}{4} \times (1 + \frac{X_m}{5} \times (1 + \frac{X_m}{6} \times (1 + \frac{X_m}{7}))))))$$

5. The value of e^x is the product of results from (3) and (4)

Equation:
$Y = e^{\frac{fNum}{fDen}}$

5.12 Exponential (Base 10)

Prototype:

float CLAexp10(float fVal)

Parameters:

fVal Input argument (non-negative range of floating point values)

Returns:

Base 10 exponential of the input argument (positive range of floating point values)

Description:

This function calculates the base 10 exponential function of the input argument i.e. 10^x, where x is the input value. It is calculated as follows:

1. x = $|\frac{x}{log_(10)(e)}|$
2. Identify the integer and mantissa of the input
3. Obtain the $e^{integer(x)}$ from the table **CLAExpTable**
4. Calculate the value of $e^{(mantissa)}$ by using the polynomial approx:

$$e^{X_m} \quad = \quad 1 + X_m \times (1 + X_m \times 0.5(1 + (\frac{X_m}{3}) \times (1 + \frac{X_m}{4} \times (1 + \frac{X_m}{5} \times (1 + \frac{X_m}{6} \times (1 + \frac{X_m}{7}))))))$$

5. The value of e^x is the product of results from (3) and (4).

   It can be proven that $10^x = e^{\frac{x}{log_{10}e}}$ and since we have divided x by $log_{10}(e)$ in step (1), the result we obtain will be the desired 10^x

Equation:

$Y = 10^{fVal}$

5.13 Inverse Square Root

Prototype:
float CLAisqrt(float fVal)

Parameters:
fVal Input number (positive range of floating point values)

Returns:
Inverse Square root of input argument (positive range of floating point values)

Description:
This function calculates the inverse square root of the input argument i.e. $\frac{1}{\sqrt{X}}$, where X is the input argument

This function uses the Newton Raphson approximation to converge on the answer.

$$\begin{aligned} Y^{'} &\approx \frac{1}{\sqrt{X}} \\ Y^{''} &= Y^{'} \times (1.5 - Y^{'} \times Y^{'} \times X \times 0.5) \\ Y^{'''} &= Y^{''} \times (1.5 - Y^{''} \times Y^{''} \times X \times 0.5) \\ Y &= Y^{'''} \end{aligned}$$

Equation:
$Y = \frac{1}{\sqrt{fVal}}$

5.14 Natural Logarithm

Prototype:

float CLAln(float fVal)

Parameters:

fVal Input argument (positive range of floating point values)

Returns:

Natural log of the input argument (non-negative range of floating point values)

Description:

This function calculates the natural log of the input argument i.e. $log_e(x)$, where x is the input value.

1. Calculate absolute of x
2. Identify the exponent of the input, store it float.
3. Identify the mantissa, X_m and use it to look up the polynomial coefficients in the table **CLALnTable**
4. Subtract the bias from the exponent and multiply it by Ln(2)
5. Calculate the value of $log_e(1 + mantissa)$ by using the polynomial approx: $log_e(1 + X_m) = a_0 + X_m \times (a_1 + X_m \times a_2)$
6. $Result = log_e(1 + X_m) + (Exponent - 127) \times (log_e(2))$

Equation:

$Y = log_e(fVal)$

5.15 Logarithm(Base 10)

Prototype:
float CLAlog10(float fVal)

Parameters:
fVal Input argument (positive range of floating point values)

Returns:
Base 10 log of the input argument (non-negative range of floating point values)

Description:
This function calculates the Log(base 10) of the input argument i.e. $log_{10}(x)$, where x is the input value

1. Calculate absolute of x
2. Identify the exponent of the input, store it float.
3. Identify the mantissa, X_m and use it to look up the polynomial coefficients in the table **CLALnTable**
4. Subtract the bias from the exponent and multiply it by Ln(2)
5. Calculate the value of $log_e(1 + mantissa)$ by using the polynomial approx: $log_e(1 + X_m) = a_0 + X_m \times (a_1 + X_m \times a_2)$
6. $Result = \frac{log_e(1+X_m)+(Exponent-127)\times(log_e(2))}{log_e(10)}$

Equation:
$Y = log_{10}(fVal)$

5.16 Sine

Prototype:
float CLAsin(float fAngleRad)

Parameters:
fAngleRad Input angle in radians ($-2\pi \leq Angle \leq 2\pi$)

Returns:
Sine of the input angle ($-1 \leq Result \leq 1$)

Description:
This function calculates the sine of an input angle i.e. $sin(rad)$, where rad is the input angle in radians and rad = K + X

Using Taylor series expansion around the value K we get,

$$\begin{aligned}
Sin(rad) = Sin(K) &+ Cos(K) \times X \\
&- Sin(K) \times \frac{X^2}{2!} \\
&- Cos(K) \times \frac{X^3}{3!} \\
&+ Sin(K) \times \frac{X^4}{4!} \\
&+ Cos(K) \times \frac{X^5}{5!} \\
Sin(rad) = Sin(K) &+ X \times (Cos(K) \\
&+ X \times (-0.5 \times Sin(K) \\
&+ X \times (-0.166666 \times Cos(K) \\
&+ X \times (0.04166666 \times Sin(K) \\
&+ X \times (0.00833333 \times Cos(K)))))) \\
Sin(rad) = Sin(K) &+ X \times (Cos(K) \\
&+ X \times (Coef0 \times Sin(K) \\
&+ X \times (Coef1 \times Cos(K) \\
&+ X \times (Coef2 \times Sin(K) \\
&+ X \times (Coef3 \times Cos(K))))))
\end{aligned}$$

Equation:
$Y = sin(fAngleRad)$

5.17 Sine Per-Unit

Prototype:

float CLAsinPU(float fAngleRadPU)

Parameters:

fAngleRadPU Input angle in radians(per 2π units) ($-1 \leq Angle \leq 1$)

Returns:

Sine of the angle ($-1 \leq Result \leq 1$)

Description:

This function calculates the sine of a per-unit angle i.e. $sin(radPU)$, where where radPU is the input angle in radians (per unit 2π) and radPU = K + X

Therefore rad= radPU*2*π

Using Taylor series expansion around the value K we get,

$$\begin{aligned}
Sin(rad) = Sin(K) &+ Cos(K) \times X \\
&- Sin(K) \times \frac{X^2}{2!} \\
&- Cos(K) \times \frac{X^3}{3!} \\
&+ Sin(K) \times \frac{X^4}{4!} \\
&+ Cos(K) \times \frac{X^5}{5!} \\
Sin(rad) = Sin(K) &+ X \times (Cos(K) \\
&+ X \times (-0.5 \times Sin(K) \\
&+ X \times (-0.166666 \times Cos(K) \\
&+ X \times (0.04166666 \times Sin(K) \\
&+ X \times (0.00833333 \times Cos(K))))) \\
Sin(rad) = Sin(K) &+ X \times (Cos(K) \\
&+ X \times (Coef0 \times Sin(K) \\
&+ X \times (Coef1 \times Cos(K) \\
&+ X \times (Coef2 \times Sin(K) \\
&+ X \times (Coef3 \times Cos(K)))))
\end{aligned}$$

Equation:

$Y = sin(fAngleRadPU)$

5.18 Square Root

Prototype:

float CLAsqrt(float fVal)

Parameters:

fVal Input number (positive range of floating point values)

Returns:

Square root of input argument (positive range of floating point values)

Description:

This function calculates the square root of the input argument i.e. $\sqrt{X}$, where X is the input value

This function uses the Newton Raphson approximation to converge on the answer.

$$\begin{aligned} Y^{'} &\approx \frac{1}{\sqrt{X}} \\ Y^{''} &= Y^{'} \times (1.5 - Y^{'} \times Y^{'} \times X \times 0.5) \\ Y^{'''} &= Y^{''} \times (1.5 - Y^{''} \times Y^{''} \times X \times 0.5) \\ Y &= Y^{'''} \times X \end{aligned}$$

Equation:

$Y = \sqrt{fVal}$

6 Benchmarks

All the CLA assembly instructions execute in a single cycle. The benchmark numbers were obtained by simply counting the number of instructions in each of the routines. The benchmarks include the return but not the function call. The call instruction could add between 1 to 4 cycles since the compiler, depending on the optimization level, often places some of the routine's instructions in the delay slot of the call instruction.

Type	**Function**	**Cycles**[1]
Trigonometric	CLAcos	31
	CLAsin	31
	CLAatan	44
	CLAatan2	47
	CLAatan2PU	49
	CLAcosPU	31
	CLAsinePU	31
	CLAacos	27
	CLAacos_spc	27
	CLAasin	25
Logarithmic	CLAln	31
	CLAlog10	32
Exponential	CLAexp	44
	CLAexp10	46
	CLAexp2	56
Miscellanous	CLAdiv	16
	CLAisqrt	17
	CLAsqrt	19

Table 6.1: Benchmark for the CLA Math Library Routines.

[1] numbers include the return but not the function call

7 Revision History

V4.00.01.00: Minor Update

- Created two library projects for CLA Type 0 and Type 1
- Updated all projects (library and examples) to work with CCSv5 and CGT v6.2.4
- Fixed issue with table lookup in the acos and asin routines
- Added F2805x specific acos routine (used with datarom variant of the CLA library)
- Added FLASH and RAM build configurations for all examples
- Deleted first triplet in the acos lookup table (this was an incorrect entry). The total number of triplets is now 64
- Changed declaration of tables in the header file from pointers to arrays. This allows the user to use the tables in custom C code.
- Added examples for the F2837x which use the Type 1 CLA
- Fixed bug in the CLAdiv() and CLAsqrt() where the ZF bit kept its state across multiple calls

V4.00: Major Update

- Source library re-built with CLA **C** compiler (codegen v6.1.0)
- Math macros from the previous release were retained and modified into C-callable assembly functions

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* added to the list of available macros

V1.00a: Minor Update

Source code has not been altered. Changes made to prepare the package for controlSUITE release and improved usability in CCSv4.

V1.00: Initial Release

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Broadband	www.ti.com/broadband
DSP	dsp.ti.com	Digital Control	www.ti.com/digitalcontrol
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Military	www.ti.com/military
Logic	logic.ti.com	Optical Networking	www.ti.com/opticalnetwork
Power Mgmt	power.ti.com	Security	www.ti.com/security
Microcontrollers	microcontroller.ti.com	Telephony	www.ti.com/telephony
RFID	www.ti-rfid.com	Video & Imaging	www.ti.com/video
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Wireless	www.ti.com/wireless