**CLA Documentation**

SE 423

Spring 2021

Joe Koszut

Table of Contents

[Paths 2](#_Toc72241515)

[Path Variables 2](#_Toc72241516)

[Include Options 3](#_Toc72241517)

[Predefined Symbols 5](#_Toc72241518)

[File Search Path 6](#_Toc72241519)

[Command File Preprocessing 7](#_Toc72241520)

[Memory 8](#_Toc72241521)

[LSxMSEL 9](#_Toc72241522)

[LSxCLAPGM 9](#_Toc72241523)

[Setting Up .c File 10](#_Toc72241524)

[Setting Up .cla File 11](#_Toc72241525)

[Notes 12](#_Toc72241526)

# Paths

Path Variables

Resource --> Linked Resources --> Path Variables

Graphical user interface, text, application, email

Description automatically generated

Figure . Paths used for Linked Resources.

Graphical user interface, text, application, email

Description automatically generated

Figure . Zoomed in version of Figure 1.

Include Options

Build --> C2000 Compiler --> Include Options

Graphical user interface, text, application, email

Description automatically generated

Figure . Paths used for include options. Details for each entry shown below.

PROJECT\_ROOT:

Graphical user interface, text, application, email

Description automatically generated

CLAMATH\_ROOT/include:

Graphical user interface, text, application, email

Description automatically generated

C2000WARE\_COMMON\_INCLUDE:

Graphical user interface, text, application, email

Description automatically generated

C2000WARE\_HEADERS\_INCLUDE:

Graphical user interface, text, application, email

Description automatically generated

PROJECT\_ROOT/device:

Graphical user interface, text, application, email

Description automatically generated

C2000WARE\_DLIB\_ROOT:

Graphical user interface, text, application, email

Description automatically generated

C2000WARE\_DSPLIB\_ROOT:

Graphical user interface, text, application, email

Description automatically generated

CG\_TOOL\_ROOT/include:

Graphical user interface, text, application, email

Description automatically generated

Predefined Symbols

Build --> C2000 Compiler --> Predefined Symbols

Graphical user interface, text, application, email

Description automatically generated

Figure . Entries for predefined symbols.

Graphical user interface, text, application, email

Description automatically generated

Figure 5. Zoomed in version of Figure 4.

File Search Path

Build --> C2000 Linker --> File Search Path

Graphical user interface, text, application, email

Description automatically generated

Figure 6. File Search Paths

Library files:

Graphical user interface, text, application, email

Description automatically generated

Library search paths:

Graphical user interface, text, application, email

Description automatically generated

Command File Preprocessing

Build --> C2000 Linker --> Advanced Options --> Command File Preprocessing

Graphical user interface, text, application

Description automatically generated

Figure 7. Pre-defined preprocessor macros.

Pre-defines:

Graphical user interface, text, application

Description automatically generated

# Memory

Note that the CLA cannot use GSx RAM. This can be seen from Table 6-1 (pg. 179) of the TMS320F2837xD datasheet.  
Table

Description automatically generated

The LSxMSEL and LSxCLAPGM registers are both used in setting up the CLA memory.

LSxMSEL

A memory block can be assigned to the CLA by writing a 1 to the memory block's MemCfgRegs.LSxMSEL[MSEL\_LSx] bit:

MemCfgRegs.LSxMSEL.bit.MSEL\_LS2 = 1;

MemCfgRegs.LSxMSEL.bit.MSEL\_LS3 = 1;

LSxCLAPGM

A memory block can be configured as a code block or data block by writing to the memory block's MemCfgRegs.LSxCLAPGM[CLAPGM\_LSx] bit.

Writing a 0 configures the memory block as a data block:

MemCfgRegs.LSxCLAPGM.bit.CLAPGM\_LS2 = 0;

Writing a 1 configures the memory block as a code block:

MemCfgRegs.LSxCLAPGM.bit.CLAPGM\_LS3 = 1;

Note that the value of the LSxCLAPGM bit is 0 at reset.

# Setting Up .c File

Include the CLA header file:

**#include** "cla\_shared.h"

Set up data to go from CPU to CLA:

**#pragma** DATA\_SECTION(CLAin\_Value,"CpuToCla1MsgRAM")

**float** CLAin\_Value = 0;

Set up data to go from CLA to CPU:

**#pragma** DATA\_SECTION(numtimes1,"Cla1ToCpuMsgRAM")

int32\_t numtimes1 = 0;

Set up data to be used by only the CLA:

**#pragma** DATA\_SECTION(CLA\_enc1,"CLADataLS2")

**float** CLA\_enc1 = 0;

Configure the CLA memory\*:

**void** **CLA\_configClaMemory**(**void**);

Initialize the CLA\*:

**void** **CLA\_initCpu1Cla1**(**void**);

\* Configure the CLA memory and initialize the CLA in the *main* function:

CLA\_configClaMemory();

CLA\_initCpu1Cla1();

Enable the CLA interrupt in main:

PieVectTable.CLA1\_1\_INT = &cla1Isr1;

Initialize an interrupt to occur at a fixed rate, e.g. ADC interrupt triggered by EPWM5 at 1 msec. Inside this interrupt, force the CLA to start. This will effectively run the CLA at whatever rate the interrupt is set.

Cla1ForceTask1();

Once the CLA is forced to start, it will run and then generate the CLA interrupt inside the *.c* file once the CLA finishes. Inside this interrupt, things like printing the states and control signal can be done. Simply make sure that any CLA data used in the CLA ISR function is initialized correctly, i.e., using "Cla1ToCpuMsgRAM".

# Setting Up .cla File

Include the CLA math header file:

**#include** <CLAMath.h>

Include the CLA header file:

**#include** "cla\_shared.h"

Include the CLA type definitions header file:

**#include** "F2837xD\_Cla\_typedefs.h" // CLA type definitions

Include any desired F2837xD header files:

**#include** "F2837xD\_device.h" // F2837xD peripheral address definitions

Create the CLA1 task function:

**\_\_interrupt** **void** **Cla1Task1** ( **void** ) {...}

Define all local CLA1 variables:

**float** x1,x2,x1dot,x2dot;

**float** K1 = 5.5782;

A PWM output can be sent from within the CLA as long as it is correctly defined. In the CLA1 task function, setting the PWM can be done the same was as it is done from within the *.c* file:

setEPWM6Acla(CLA\_u);

with the PWM function being set up in the CLA file as

**void** **setEPWM6Acla**(**float** u)

{

**float** pwmCountMax = 2500.0;

**float** pwmVal = 0;

**if** (u > 10) u = 10;

**if** (u < -10) u = -10;

**if** (u>0) {

GpioDataRegs.GPACLEAR.bit.GPIO29 = 1;

} **else** {

GpioDataRegs.GPASET.bit.GPIO29 = 1;

}

pwmVal = fabs(u) \* (pwmCountMax / 10.0);

// set compareA compare value

EPwm6Regs.CMPA.bit.CMPA = (**int**)pwmVal;

}

# Notes

1. The *buffer.h* file had *BUF\_SIZE* changed from 4008 to 256. This was done because a size of 4008 was too big, causing the CLA to not work when using flash memory.
2. Breakpoints within the *.cla* file do not work.
3. A small number of math operations are not supported for use with CLA. These are shown below, taken from page 9-12 of TI's TMS320F2837xD Microcontroller Workshop manual.  
   Text

   Description automatically generated with low confidence
4. More detailed information on how the initialization of the CLA memory works can be found on page 704 of TI's TMS320F2837xD Technical Reference Manual.
5. More detailed information on building a CLA program can be found on page of TI's TMS320F2837xD Technical Reference Manual.