

CLA Documentation

SE 423
Spring 2021
Joe Koszut

Table of Contents

PATHS	2
PATH VARIABLES	2
INCLUDE OPTIONS	3
PREDEFINED SYMBOLS.....	5
FILE SEARCH PATH.....	6
COMMAND FILE PREPROCESSING	7
MEMORY.....	8
LSxMSEL.....	9
LSxCLAPGM.....	9
SETTING UP .C FILE	10
SETTING UP .CLA FILE	12
NOTES	14

Paths

Path Variables

Resource --> Linked Resources --> Path Variables

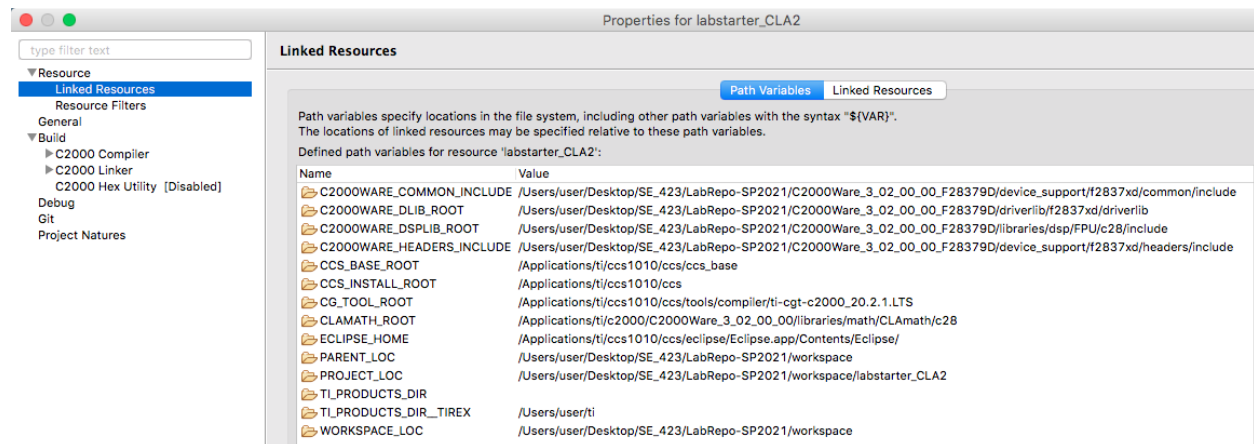


Figure 1. Paths used for Linked Resources.

Name	Value
C2000WARE_COMMON_INCLUDE	/Users/user/Desktop/SE_423/LabRepo-SP2021/C2000Ware_3_02_00_00_F28379D/device_support/f2837xd/common/include
C2000WARE_DLIB_ROOT	/Users/user/Desktop/SE_423/LabRepo-SP2021/C2000Ware_3_02_00_00_F28379D/driverlib/f2837xd/driverlib
C2000WARE_DSPLIB_ROOT	/Users/user/Desktop/SE_423/LabRepo-SP2021/C2000Ware_3_02_00_00_F28379D/libraries/dsp/FPU/c28/include
C2000WARE_HEADERS_INCLUDE	/Users/user/Desktop/SE_423/LabRepo-SP2021/C2000Ware_3_02_00_00_F28379D/device_support/f2837xd/headers/include
CCS_BASE_ROOT	/Applications/ti/ccs1010/ccs/ccs_base
CCS_INSTALL_ROOT	/Applications/ti/ccs1010/ccs
CG_TOOL_ROOT	/Applications/ti/ccs1010/ccs/tools/compiler/ti-cgt-c2000_20.2.1.LTS
CLAMATH_ROOT	/Applications/ti/c2000/C2000Ware_3_02_00_00/libraries/math/CLAMath/c28
ECLIPSE_HOME	/Applications/ti/ccs1010/ccs/eclipse/Eclipse.app/Contents/Eclipse/
PARENT_LOC	/Users/user/Desktop/SE_423/LabRepo-SP2021/workspace
PROJECT_LOC	/Users/user/Desktop/SE_423/LabRepo-SP2021/workspace/labstarter_CLA2
TI_PRODUCTS_DIR	
TI_PRODUCTS_DIR_TIREX	/Users/user/ti
WORKSPACE_LOC	/Users/user/Desktop/SE_423/LabRepo-SP2021/workspace

Figure 2. Zoomed in version of Figure 1.

Include Options

Build --> C2000 Compiler --> Include Options

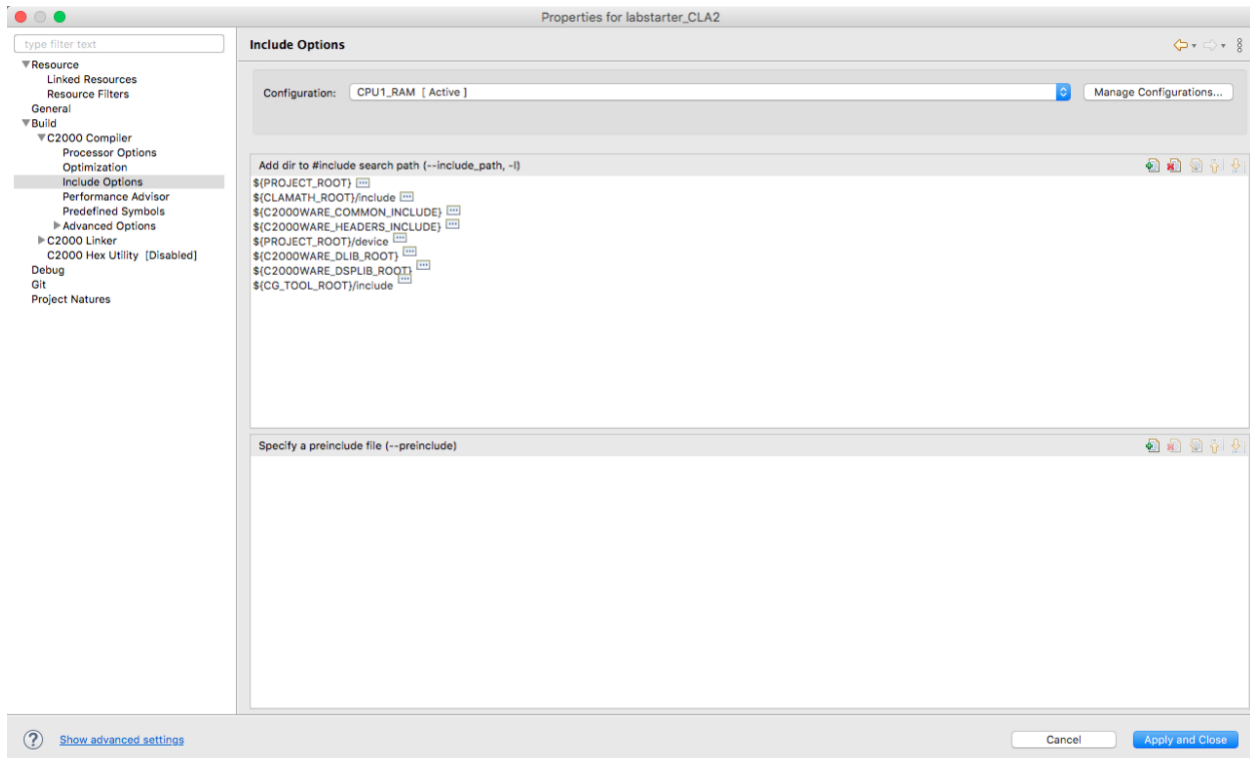


Figure 3. Paths used for include options. Details for each entry shown below.

PROJECT_ROOT:

/Users/user/Desktop/SE_423/LabRepo-SP2021/workspace/labstarter_CLA2

CLAMATH_ROOT/include:

/Applications/ti/c2000/C2000Ware_3_02_00_00/libraries/math/CLAmath/c28/include


C2000WARE_COMMON_INCLUDE:

/Users/user/Desktop/SE_423/LabRepo-SP2021/C2000Ware_3_02_00_00_F28379D/device_support/f2837xd/common/include


C2000WARE_HEADERS_INCLUDE:

/Users/user/Desktop/SE_423/LabRepo-SP2021/C2000Ware_3_02_00_00_F28379D/device_support/f2837xd/headers/include


PROJECT_ROOT/device:

 /Users/user/Desktop/SE_423/LabRepo-SP2021/workspace/labstarter_CLA2/device


C2000WARE_DLIB_ROOT:

 /Users/user/Desktop/SE_423/LabRepo-SP2021/C2000Ware_3_02_00_00_F28379D/driverlib/f2837xd/driverlib

C2000WARE_DSPLIB_ROOT:

 /Users/user/Desktop/SE_423/LabRepo-SP2021/C2000Ware_3_02_00_00_F28379D/libraries/dsp/FPU/c28/include

CG_TOOL_ROOT/include:

 /Applications/ti/ccs1010/ccs/tools/compiler/ti-cgt-c2000_20.2.1.LTS/include

Predefined Symbols

Build --> C2000 Compiler --> Predefined Symbols

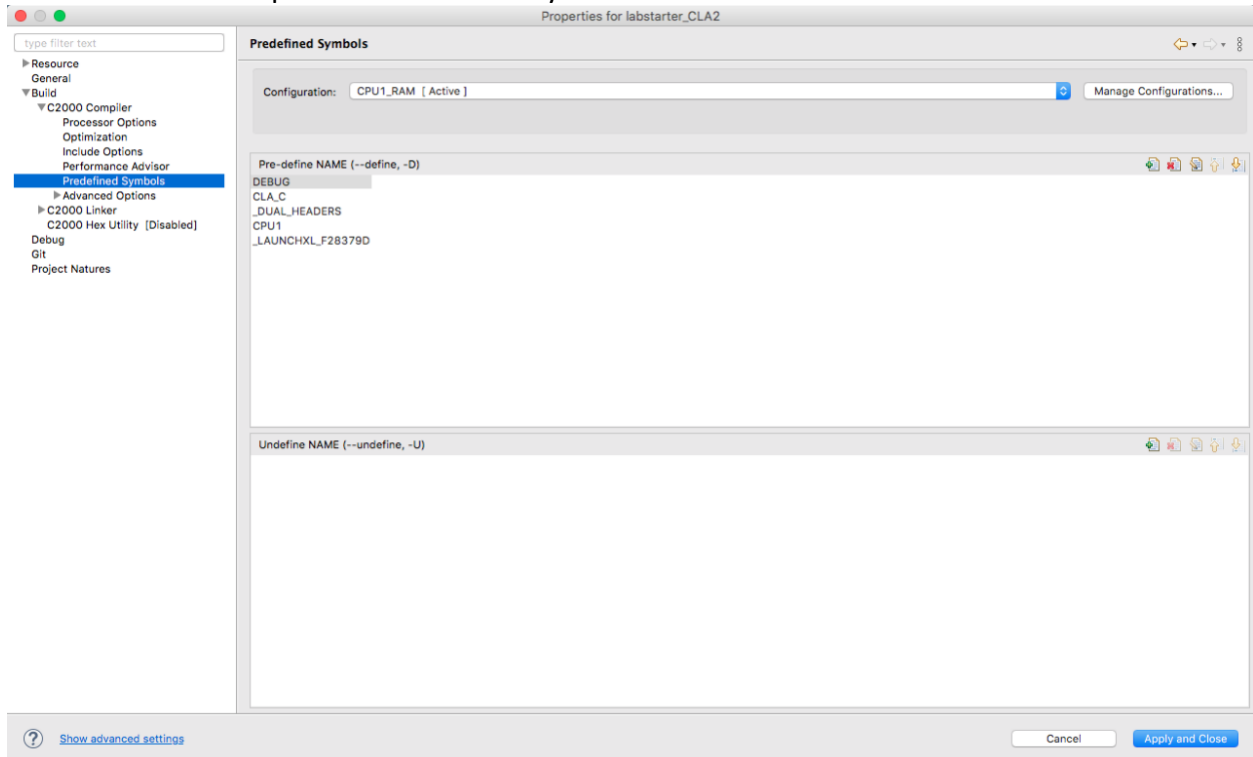


Figure 4. Entries for predefined symbols.

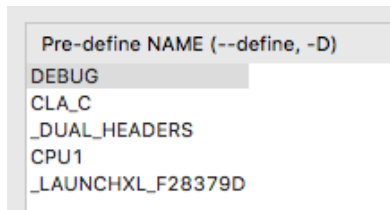


Figure 5. Zoomed in version of Figure 4.

File Search Path

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

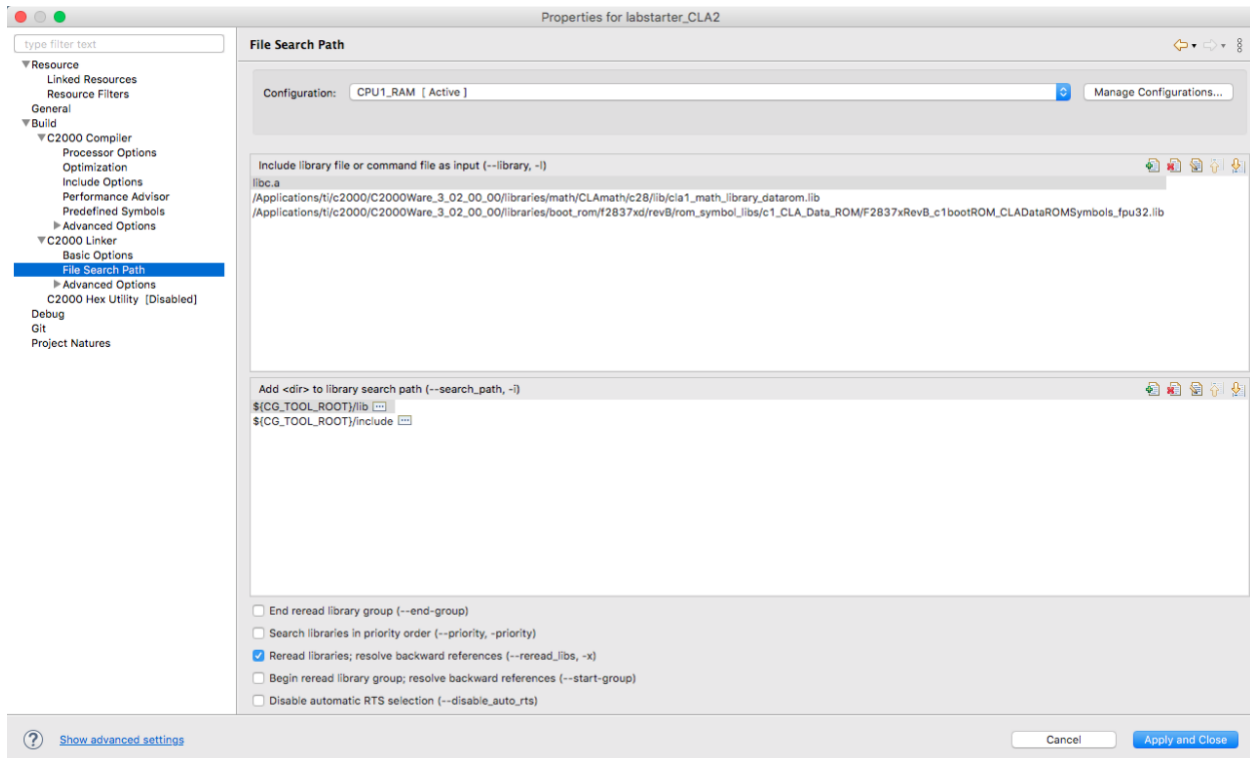


Figure 6. File Search Paths

Library files:

```
libc.a
/Applications/ti/c2000/C2000Ware_3_02_00_00/libraries/math/CLAmath/c28/lib/c1a1_math_library_datarom.lib
/Applications/ti/c2000/C2000Ware_3_02_00_00/libraries/boot_rom/f2837xd/revB/rom_symbol_libs/c1_CLA_Data_ROM/F2837xRevB_c1bootROM_CLADatROMSymbols_fpu32.lib
```

Library search paths:

```
Add <dir> to library search path (--search_path, -i)
$(CG_TOOL_ROOT)/lib
$(CG_TOOL_ROOT)/include
```

Command File Preprocessing

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

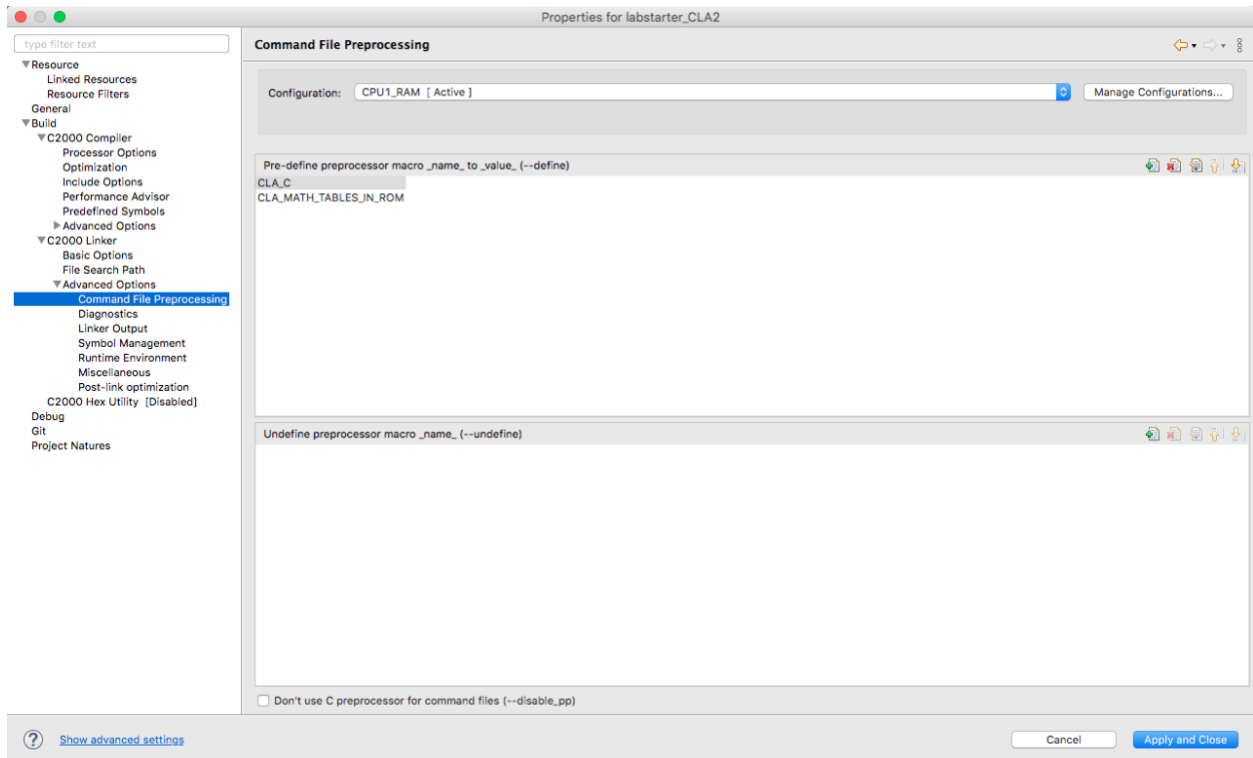


Figure 7. Pre-defined preprocessor macros.

Pre-defines:

```
Pre-define preprocessor macro _name_ to _value_ (--define)
CLA_C
CLA_MATH_TABLES_IN_ROM
```

Memory

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

6.3.1 C28x Memory Map

Both C28x CPUs on the device have the same memory map except where noted in Table 6-1. The GSx_RAM (Global Shared RAM) should be assigned to either CPU by the GSxMSEL register. Memories accessible by the CLA or DMA (direct memory access) are noted as well.

Table 6-1. C28x Memory Map

MEMORY	SIZE	START ADDRESS	END ADDRESS	CLA ACCESS	DMA ACCESS
M0 RAM	1K × 16	0x0000 0000	0x0000 03FF		
M1 RAM	1K × 16	0x0000 0400	0x0000 07FF		
PieVectTable	512 × 16	0x0000 0D00	0x0000 0EFF		
CPUx.CLA1 to CPUx MSGRAM	128 × 16	0x0000 1480	0x0000 14FF	Yes	
CPUx to CPUx.CLA1 MSGRAM	128 × 16	0x0000 1500	0x0000 157F	Yes	
UPP TX MSG RAM	512 × 16	0x0000 6C00	0x0000 6DFF	Yes (CPU1.CLA1 only)	
UPP RX MSG RAM	512 × 16	0x0000 6E00	0x0000 6FFF	Yes (CPU1.CLA1 only)	
LS0 RAM	2K × 16	0x0000 8000	0x0000 87FF	Yes	
LS1 RAM	2K × 16	0x0000 8800	0x0000 8FFF	Yes	
LS2 RAM	2K × 16	0x0000 9000	0x0000 97FF	Yes	
LS3 RAM	2K × 16	0x0000 9800	0x0000 9FFF	Yes	
LS4 RAM	2K × 16	0x0000 A000	0x0000 A7FF	Yes	
LS5 RAM	2K × 16	0x0000 A800	0x0000 AFFF	Yes	
D0 RAM	2K × 16	0x0000 B000	0x0000 B7FF		
D1 RAM	2K × 16	0x0000 B800	0x0000 BFFF		
GS0 RAM ⁽¹⁾	4K × 16	0x0000 C000	0x0000 CFFF		Yes
GS1 RAM ⁽¹⁾	4K × 16	0x0000 D000	0x0000 DFFF		Yes
GS2 RAM ⁽¹⁾	4K × 16	0x0000 E000	0x0000 EFFF		Yes
GS3 RAM ⁽¹⁾	4K × 16	0x0000 F000	0x0000 FFFF		Yes
GS4 RAM ⁽¹⁾	4K × 16	0x0001 0000	0x0001 0FFF		Yes
GS5 RAM ⁽¹⁾	4K × 16	0x0001 1000	0x0001 1FFF		Yes
GS6 RAM ⁽¹⁾	4K × 16	0x0001 2000	0x0001 2FFF		Yes
GS7 RAM ⁽¹⁾	4K × 16	0x0001 3000	0x0001 3FFF		Yes
GS8 RAM ⁽¹⁾	4K × 16	0x0001 4000	0x0001 4FFF		Yes
GS9 RAM ⁽¹⁾	4K × 16	0x0001 5000	0x0001 5FFF		Yes
GS10 RAM ⁽¹⁾	4K × 16	0x0001 6000	0x0001 6FFF		Yes
GS11 RAM ⁽¹⁾	4K × 16	0x0001 7000	0x0001 7FFF		Yes
GS12 RAM ⁽¹⁾⁽²⁾	4K × 16	0x0001 8000	0x0001 8FFF		Yes
GS13 RAM ⁽¹⁾⁽²⁾	4K × 16	0x0001 9000	0x0001 9FFF		Yes
GS14 RAM ⁽¹⁾⁽²⁾	4K × 16	0x0001 A000	0x0001 AFFF		Yes
GS15 RAM ⁽¹⁾⁽²⁾	4K × 16	0x0001 B000	0x0001 BFFF		Yes
CPU2 to CPU1 MSGRAM ⁽¹⁾	1K × 16	0x0003 F800	0x0003 FBFF		Yes
CPU1 to CPU2 MSGRAM ⁽¹⁾	1K × 16	0x0003 FC00	0x0003 FFFF		Yes
CAN A Message RAM ⁽¹⁾	2K × 16	0x0004 9000	0x0004 97FF		
CAN B Message RAM ⁽¹⁾	2K × 16	0x0004 B000	0x0004 B7FF		
Flash	256K × 16	0x0008 0000	0x000B FFFF		
Secure ROM	32K × 16	0x003F 0000	0x003F 7FFF		
Boot ROM	32K × 16	0x003F 8000	0x003F FBF		
Vectors	64 × 16	0x003F FFC0	0x003F FFFF		

(1) Shared between CPU subsystems.

(2) Available only on F28379D, F28378D, F28377D, and F28375D.

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, "CLADDataLS2")  
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.

CLA C Language Restrictions (2 of 2)

◆ No support for certain fundamental math operations

◆ integer division: $z = x/y$;

◆ modulus (remainder): $z = x\%y$;

◆ unsigned 32-bit integer compares

```
Uint32 i;  if(i < 10) {...} // not valid
int32 i;   if(i < 10) {...} // valid
Uint16 i;  if(i < 10) {...} // valid
int16 i;   if(i < 10) {...} // valid
float32 x; if(x < 10) {...} // valid
```

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.