Introduction

Something has gone wrong while using a TI compiler. You need to submit a test case so TI can reproduce the problem. This article shows you how to do that.

If you experience a problem while linking, this article does *not* apply to you. A different (not yet written) article addresses that situation.

In Video Form

For those who prefer, this video (https://www.youtube.com/watch?v=d-hB8i534C0) shows the typical way CCS users submit a compiler test case.

What Must be Submitted

1. Preprocessed source file
2. Compiler Version
3. Compiler Options Used

This page is mostly about creating and handling the preprocessed source file. Please do not forget the compiler version and options used. That information must be supplied, or the problem cannot be addressed.

Preprocess the Source File

The problem source file probably includes many other header files. Preprocessing means you don't have to manually track down those header files (which often include yet other header files) and somehow submit them too. After preprocessing, there is only one file to submit.

The exact details for creating a preprocessed file depend on how you do your build. This article discusses four different methods.

1. CCS and TI proprietary compiler
2. A simple makefile and TI proprietary compiler
3. CCS and clang compiler
4. A simple makefile and clang compiler

Use these descriptions only as a general guide, and not a set of precise instructions.

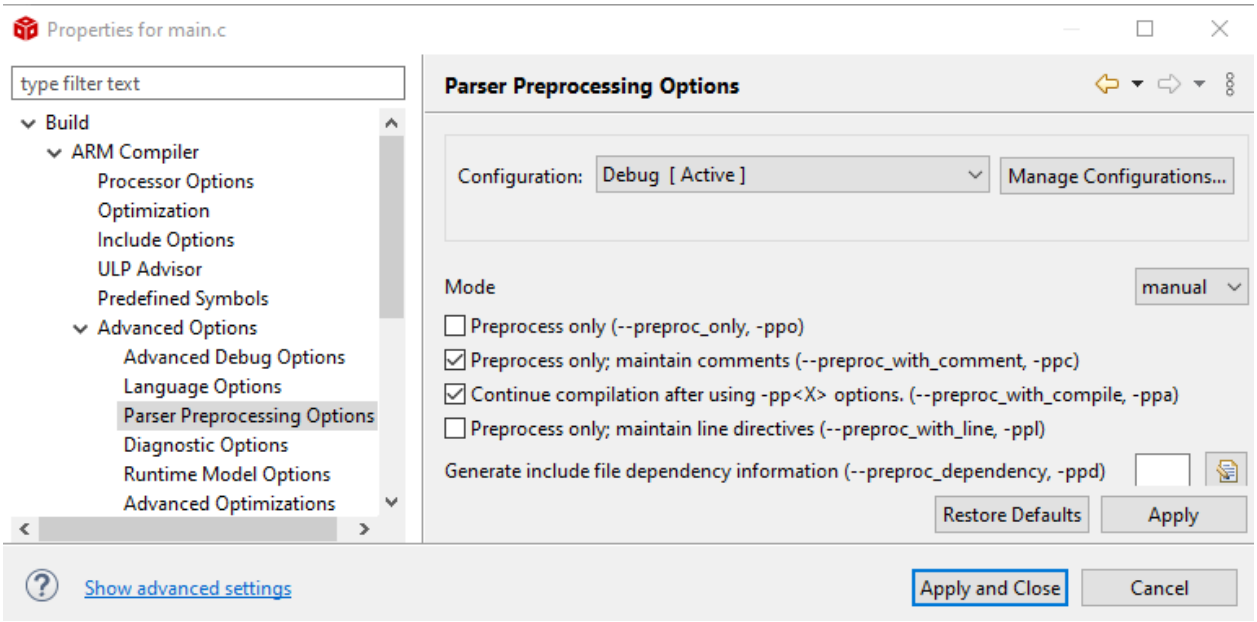
CCS Method and TI Proprietary compiler

These directions presume:

- The problem source file is part of a project you build within CCS
- You use a proprietary TI compiler such as `armcl`, `cl2000`, or `cl6x`

These directions are mostly accurate for CCS versions 6.x and higher. These particular directions were tested with CCS version 10.1.

- From the Project Explorer view, right-click on the file name and select **Show Build Settings ...**
- Navigate the directory like path on the left to something like **Build/ARM Compiler/Advanced Options**
- Highlight **Parser Preprocessing Options**
- Change the **Mode** drop-down box from **automatic (default)** to **manual**
- Check the box for **Preprocess only; maintain comments**
- Check the box for **Continue compilation after using -pp<X> options**
- Click **Apply and Close**



Now you are ready to build the file.

- Right-click on the file name and select **Build Selected File(s)**

A preprocessed file is created with the same base name as the source file, and the extension changed to `.pp`. For example, `file.pp`. This file is found in the same directory as the original source file. This is the file to send to TI. It is an ordinary text file. Don't be alarmed if it starts with many blank lines. That is not unusual. Note you can use the usual Windows operations to copy and paste this file from CCS.

You might see a `.pp` file in another directory, like `Debug` or `Release`. Do **not** send in that file. It is an automatically generated file used by CCS. Be sure you send in the `.pp` file found *in the same directory* as the source file.

Feel free to leave the above build option changes in place, or undo them.

Makefile Method and TI Proprietary Compiler

These directions presume:

- The problem source file is part of a system you build with a makefile
- You use a proprietary TI compiler such as `armcl`, `cl2000`, or `cl6x`

Start with this simple makefile. Presume iup.c is the problem file.

```
#---------------------------------------------------------------------------
# Name the object files
#---------------------------------------------------------------------------
OBJS := icdct.obj isbt.obj iup.obj iwinm.obj mhead.obj towave.obj wavep.obj

#---------------------------------------------------------------------------
# Develop C_OPTS: The compiler build options
#---------------------------------------------------------------------------
# Optimization level 2
C_OPTS := $(C_OPTS) --opt_level=2
# Build for C6600 processors
C_OPTS := $(C_OPTS) -mv6600
# Extra SW pipeline info
C_OPTS := $(C_OPTS) --debug_software_pipeline
# See C source like comments in assembly output
C_OPTS := $(C_OPTS) --src_interlist

#---------------------------------------------------------------------------
# Link build rule
#---------------------------------------------------------------------------
mp3.out : $(OBJS)
    cl6x -z $(OBJS) -o=$@ lnk.cmd

#---------------------------------------------------------------------------
# Compile build rule
#---------------------------------------------------------------------------
%.obj : %.c
    cl6x $(C_OPTS) $<
```

The first step is to add two options for preprocessing. Add the following lines to the `C_OPTS` part of the makefile.

```
# Preprocessing which preserves comments
C_OPTS := $(C_OPTS) --preproc_with_comment
# Compilation continues normally after preprocessing
C_OPTS := $(C_OPTS) --preproc_with_compile
```

Next, rebuild only `iup.c`. Here is one way to do that.

```
% rm iup.obj
% gmake iup.obj
cl6x --opt_level=2 -mv6600 --debug_software_pipeline --src_interlist --preproc_with_com
ment --preproc_with_compile iup.c
```

The file `iup.pp` appears in the current directory. It is an ordinary text file. Don't be alarmed if your `.pp` file starts with many blank lines. That is not unusual. Note `iup.pp` is the only file you need to send to TI.

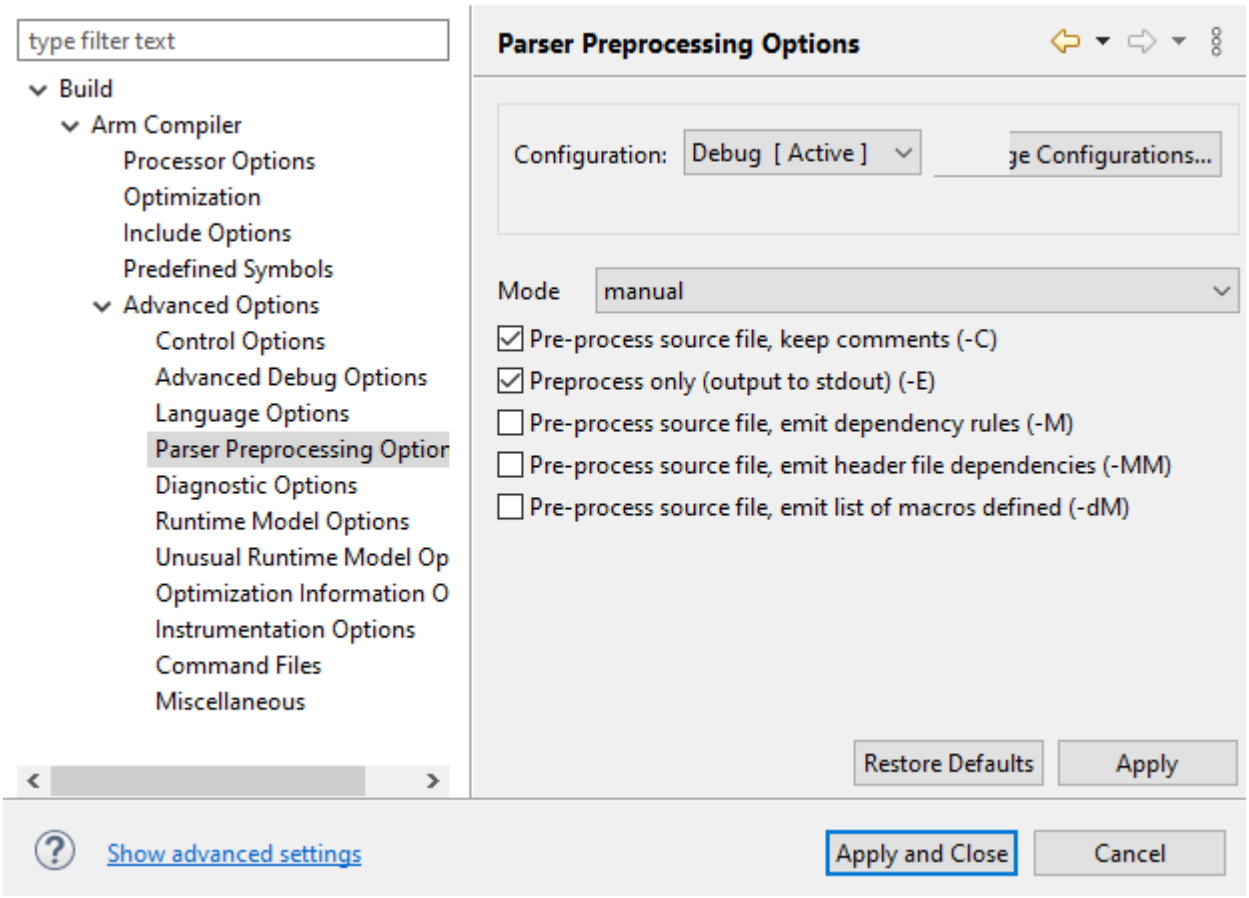
CCS Method and Clang Compiler

These directions presume:

- The problem source file is part of a project you build within CCS
- You use a compiler from TI that is based on the open source Clang/LLVM projects, such as `tiarmclang`

These particular directions were tested with CCS version 11.1. If you use a different version of CCS or a different compiler, these directions are mostly accurate, but probably not an exact match.

- From the Project Explorer view, right-click on the file name and select **Show Build Settings ...**
- Navigate the directory like path on the left to something like **Build/ARM Compiler/Advanced Options**
- Highlight **Parser Preprocessing Options**
- Change the **Mode** drop-down box from **automatic (default)** to **manual**
- Check the box for **Pre-process source file, keep comments (-C)**
- Check the box for **Pre-process only (output to stdout) (-E)**
- Click **Apply and Close**



Now you are ready to build the file.

- Right-click on the file name and select **Build Selected File(s)**

In a typical build, for source code in `file.c`, the compiler generates an object file named `file.o`. The option changes just made mean `file.o` is not an object file, but a text file that contains the preprocessed source. It is located in the directory named after the current build configuration, often `Debug` or `Release`. This is the file to send to TI. Don't be alarmed if it starts with many blank lines. That is not unusual. Note you can use the usual Windows operations to copy and paste this file from CCS.

The changes to the build options must be removed, or the next full build of the project will fail.

- Right-click on the file name and select **Resource Configurations → Reset to Default**
- Check the box next to the name of the build configuration being used, often `Debug` or `Release`
- Click **OK**

For more explanation of these steps, please see the File Specific Options (https://software-dl.ti.com/ccs/esd/documents/users_guide/ccs_project-management.html#file-specific-options) part of the CCS online manual (https://software-dl.ti.com/ccs/esd/documents/users_guide/index.html).

Makefile Method and Clang Compiler

These directions presume:

- The problem source file is part of a system you build with a makefile
- You use a compiler from TI that is based on the open source Clang/LLVM projects, such as `tiarmclang`

Start with this simple makefile. Presume iup.c is the problem file.

```
#-------------------------------------------------------------------------------
# Name the object files
#-------------------------------------------------------------------------------
OBJS := icdct.o isbt.o iup.o iwinm.o mhead.o towave.o wavep.o

#-------------------------------------------------------------------------------
# Develop C_OPTS: The compiler build options
#-------------------------------------------------------------------------------
# Cortex-M4 CPU core
C_OPTS := $(C_OPTS) -mcpu=cortex-m4
# Floating point instructions supported
C_OPTS := $(C_OPTS) -mfloat-abi=hard
# Variant of FPU instructions that are supported
C_OPTS := $(C_OPTS) -mfpu=fpv4-sp-d16

#-------------------------------------------------------------------------------
# Link build rule
#-------------------------------------------------------------------------------
mp3.out : $(OBJS)
    tiarmclang $(OBJS) -o$@ -Wl,-mmp3.map -Wl,lnk.cmd

#-------------------------------------------------------------------------------
# Compile build rule
#-------------------------------------------------------------------------------
%.o : %.c
    tiarmclang -c -o$@ $(C_OPTS) $<
```

To arrange for preprocessing, change the compile build rule. Change `-c` (compile only) to `-E -C` (preprocess and preserve comments).

```
    tiarmclang -E -C -o$@ $(C_OPTS) $<
```

Next, rebuild only `iup.c`. Here is one way to do that.

```
% rm iup.o
% gmake iup.o
tiarmclang -E -C -oiup.o  -mcpu=cortex-m4 -mfloat-abi=hard -mfpu=fpv4-sp-d16 iup.c
```

The preprocessed code for `iup.c` is in the file `iup.o`. It is an ordinary text file. Don't be alarmed if it starts with many blank lines. That is not unusual. Note `iup.o` is the only file you need to send to TI.

Attach Preprocessed File to a Forum Post

The most common way to notify TI about a compiler problem is by posting to the E2E forum (http://e2e.ti.com). The typical pattern is to post an initial description of the problem. TI responds by requesting a test case, with a pointer to this page. You prepare the test case, and now it is time for the next step.

The forum only accepts a very small set of file types as attachments. Files with the extension `.pp` or `.o` are **not** accepted. So, add the extension `.txt` to the file. This forms a file name like `file.pp.txt` or `file.o.txt`.

Reply to the last post on the forum thread about your problem. To attach the file, please follow the directions in this FAQ (https://e2e.ti.com/support/site-support-group/site-support/f/site-support-forum/761619/faq-how-do-i-attach-a-file-or-link-to-my-post).

Protecting Intellectual Property

This section discusses ways to improve (not guarantee) protection of your source code.

The methods used above keep the comments in the source code. To leave out the comments, make these changes to the preprocessing options:

- For a TI proprietary compiler, change `--preproc_with_comments` to `--preproc_only`.
- For a Clang compiler, change `-E -C` to `-E`. That is, do not use `-C`.

Rather than attaching the code to a forum post, you can send it privately to the TI employee who requested the test case. While on the forum, hover your mouse over the screen name or avatar of the TI employee. A box will pop up. Click on **Send a private message**. Attaching the file to a private message is very similar to attaching a file to a forum post, which is described in the previous section.

Compiler Version

When you send the preprocessed file to TI, be sure to also indicate the version of the compiler used. Note this is **not** the same as the version of CCS.

This short video (https://www.youtube.com/watch?v=vnetm9247y0) shows one way to see the compiler version.

Another way is to inspect the linker map file. The version is shown in the first few lines. In the folder named after the current build configuration, often `Debug` or `Release`, you will find a file with a name similar to `name_of_project.map`. Double click to open it and see the version number.

Compiler Options Used

In addition to the preprocessed file and the compiler version, also show the build options used, exactly as they are seen by the compiler. Always copy-and-paste the text of the options. Do not use a screen shot.

If you build with CCS, copy-and-paste the build command for the problem source file from the **Console** view. If requested, or if it is easier, supply the full contents of the **Console** view. Use the **Copy Build Log** icon to save everything to a text file. Be sure to use the file extension `.txt` . Then attach that file to your next post the same as you attach the preprocessed file.

Resources

TI Code Generation Tools (https://www.ti.com/tool/TI-CGT)

Related Technical Documents (https://software-dl.ti.com/ccs/esd/documents/ccs_documentation-overview.html)

TI E2E Technical Forums (https://e2e.ti.com)

Last updated: June 3 2022 10:11 Central Time