How to make a TI-RTOS Closure that can be used by multiple projects

Todd Mullanix

Feb 28, 2016

Overview

Terminology

.cfg file: The .cfg file is the TI-RTOS configuration file. It allows a user to fine-tune TI-RTOS to meet their specific applications needs. It can be edited as a text file or graphically in CCS.

Platform: Everything you need to link. Supplied in TI-RTOS.

Target: Everything you need to compile a TI-RTOS library. Supplied in TI-RTOS.

TI-RTOS Closure: Collection of files derived from a TI-RTOS product and a .cfg file. At its essence, the closure contains TI-RTOS libraries, header files and a linker file. Projects will point to this closure to get these. The projects that consume a TI-RTOS closure do not need XDCtools or a .cfg file. It still needs the TI-RTOS product to get the driverlib libraries (and we are trying to fix this also).

Goals

- Learn how to make a TI-RTOS closure based on a specific .cfg file, platform and target.
- Learn how to point your applications to use the TI-RTOS closure.
- Show this for TI/GCC/IAR tool-chains
- Show this for both CCS and IAR IDEs.

Trade-offs

Advantages of using a TI-RTOS .cfg within your application project[1]

- Having the .cfg in the application allows TI-RTOS to be fine-tuned for that specific project.
- Easier development flow (the project controls everything).
- Easy to do “What if” scenarios. For example, enable debug features within TI-RTOS.[2]

Advantages of pointing to a TI-RTOS Closure within your application project

- Faster application builds.
- Application builds do not require XDCtools.
- Having a closure makes having a consistent RTOS configuration across multiple projects easier.

[1] Even though the .cfg is part of the project, it can be shared across multiple projects.

[2] For different “What if” scenarios, you can have multiple TI-RTOS closures (e.g. a full featured closure, a no debug minimal closure, etc.) that projects can point to and be rebuild against. Please refer to the “Configuration File Tricks” slide near the end for ways to make a flexible .cfg file

Software needed to make the TI-RTOS Closure

These presentation will be focused toward the CC13xx/CC26xx devices but is applicable for other devices.

You'll need the following software products to make a TI-RTOS closure

- **TI-RTOS product** (in CCS App Center or here). For the purpose of this document, "TI-RTOS for TI-RTOS CC13XX and CC26XX 2.16.00.08" will be used.
- **Compiler Tools** (in IDE or stand-alone)
- **XDCtools** (shipped in CCS and also in the TI-RTOS product for non-CCS users)
- **ClosureExample.zip**: Zip includes the two files needed to make an example TI-RTOS closure (files are described on the next slide).

Example: Files need to make a TI-RTOS Closure

This presentation will use “TI-RTOS CC13XX and CC26XX 2.16.00.08” release and the two files in ClosureExample.zip to make a TI-RTOS Closure.

- **makefile_cc13xx_cc26xx**: Makefile used to build the TI-RTOS closure. Contains paths to the TI-RTOS product, the .cfg, compiler location, and platform that will be used to build the closure.
- **empty_min.cfg**: This file is from the CC2650 LaunchPad’s “Empty Min” example in “TI-RTOS for CC13XX and CC26XX”.

Assuming all software is installed as specified on the next slide, these files do not need to be modified for the example. There will be slides commenting on what would need to be modified for a different example.

Build Overview

You will use XDCtools to run “gmake –f makefile_cc13xx_cc26xx” to build the TI-RTOS Closure. The result will be a portable directory that contains the TI-RTOS Closure based off the information in the makefile.

Build Closure Step 1: Installing Software Products

For this presentation, it's assumed that the following product are install

Compiler Tools

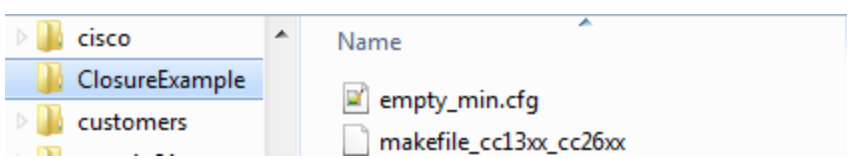
- CCS 6.1.2 is installed in c:\ti.

TI-RTOS

- TI-RTOS CC13XX and CC26XX 2.16.00.08 is installed in c:\ti.
- xdctools_3_32_00_06 is in c:\ti

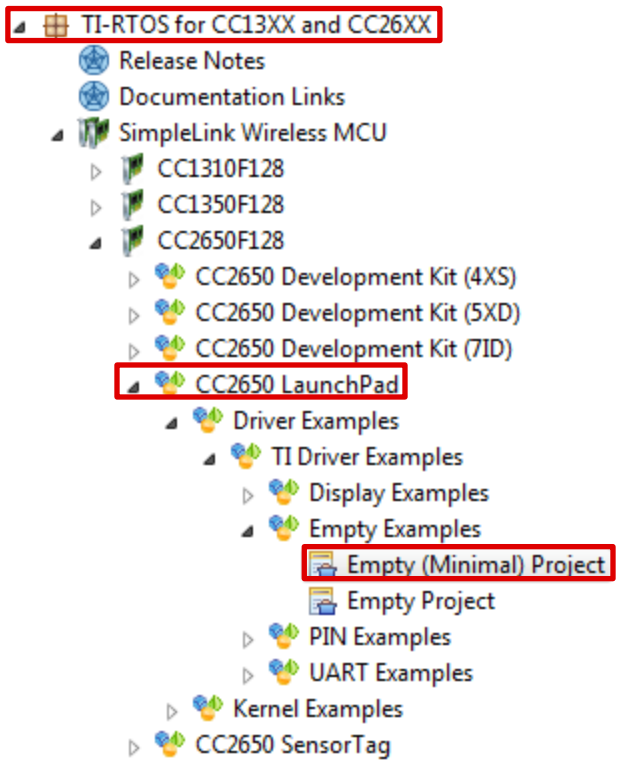
ClosureExample.zip

- Is extracted into c:\ClosureExample



Build Closure Step 2: .cfg file

As noted earlier, the empty_min.cfg is from the CC2650 LaunchPad “Empty Min” example in “TI-RTOS for CC13XX and CC26XX”. Simply use a different .cfg if desired. The name and location of the .cfg is defined at the top of the makefile (next slide).

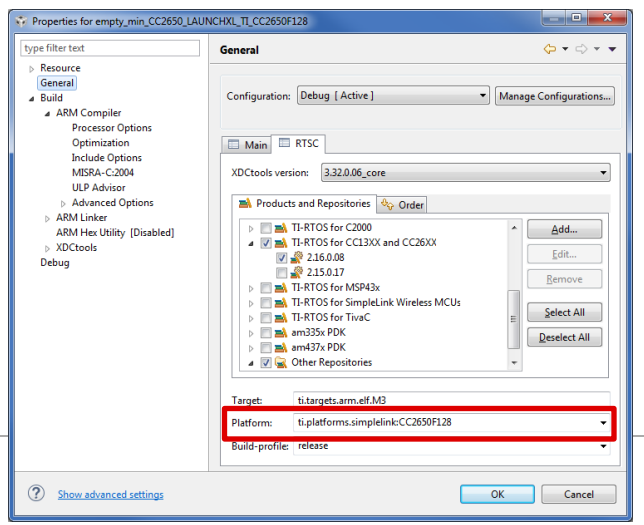


Build Closure Step 3: makefile

The makefile (in this case “makefile_cc13xx_cc26xx”) allows the user to set the following

- Closure location
- Name of the .cfg file
- Location of compiler tools
- Platform name*
- Location of TI-RTOS and XDCtools
- Compiler options (lower in the file)

* Same as a TI-RTOS CC2650 LP example



makefile_cc13xx_cc26xx

```
1  #
2  #  ======== makefile_cc13xx_cc26xx ========
3  #
4
5  #
6  # Destination directory of the TI-RTOS closure
7  #
8  CLOSURE_DIR            ?= c:/empty_min_closure
9
10 #
11 # TI-RTOS application configuration file used to make this closure
12 #
13 APPLICATION_CFG_FILE   ?= ./empty_min.cfg
14
15 #
16 # Location of compilers. Leave blank if not used.
17 #
18 TI_COMPILER_DIR        ?= C:/ti/ccs6_1_1/ccsv6/tools/compiler/ti-cgt-arm_5.2.5
19 IAR_COMPILER_DIR       ?= c:/iar/arm
20
21 #
22 # Platform to build
23 #
24 PLATFORM               ?= ti.platforms.simplelink:CC2650F128
25
26 #
27 # TI-RTOS and XDCTools locations
28 #
29 XDCTOOLS_INSTALL_DIR   ?= c:/ti/xdctools_3_32_00_06_core
30 TIRTOS_INSTALL_DIR     ?= c:/ti/tirtos_cc13xx_cc26xx_2_16_00_08
```

Build Closure Step 4: Build a TI-RTOS Closure

Now that everything is ready, building the TI-RTOS closure is easy!

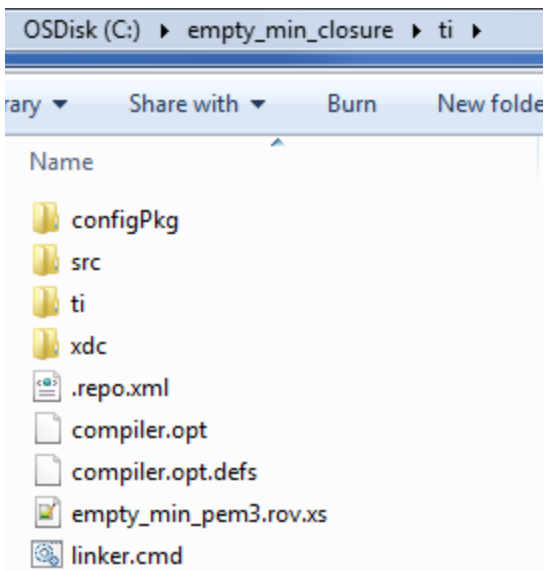
a. cd into the c:\ClosureExample directory

b. Run the following command “c:\ti\xdctools_3_32_00_06_core\gmake -f makefile_cc13xx_cc26xx ti”

```
C:\ClosureExample>c:\ti\xdctools_3_32_00_06_core\gmake -f makefile_cc13xx_cc26xx
 ti
c:/ti/xdctools_3_32_00_06_core/bin/rm.exe -rf configPkg
c:/ti/xdctools_3_32_00_06_core/bin/rm.exe -rf src
c:/ti/xdctools_3_32_00_06_core/bin/rm.exe -rf c:/empty_min_closure/ti
c:/ti/xdctools_3_32_00_06_core/bin/mkdir.exe -p c:/empty_min_closure
c:/ti/xdctools_3_32_00_06_core/xs --xdcpath="c:/ti/tirtos_cc13xx_cc26xx_2_16_00_
08/packages;c:/ti/tirtos_cc13xx_cc26xx_2_16_00_08/products/tidrivers_cc13xx_cc26
xx_2_16_00_08/packages;c:/ti/tirtos_cc13xx_cc26xx_2_16_00_08/products/bios_6_45_
01_29/packages;c:/ti/tirtos_cc13xx_cc26xx_2_16_00_08/products/uia_2_00_05_50/pac
```

After a couple minutes, you will have a TI-RTOS closure in c:\empty_min_closure\ti

To build the TI-RTOS closure with IAR, select the “iar” rule instead of the “ti” rule. This will make an iar directory similar to the ti one shown on the right.



Multiple Closures

You can make multiple closures by having multiple makefiles that generate different CLOSURE_DIR destinations or a different .cfg file. You can use different platforms or compiler options also.

The supplied makefile file is meant to be an relatively simple example that can be used as a template for additional ones as needed.

Now we have our TI-RTOS closure built, let’s point an application at it…

Consume Closure Step 1: CCS Project + TI-RTOS Closure

To consume the TI-RTOS closure, we are going to start with a simple “Hello World “example (note: that is not from TI-RTOS).

A. Select CCS->Projects->New CCS Project…

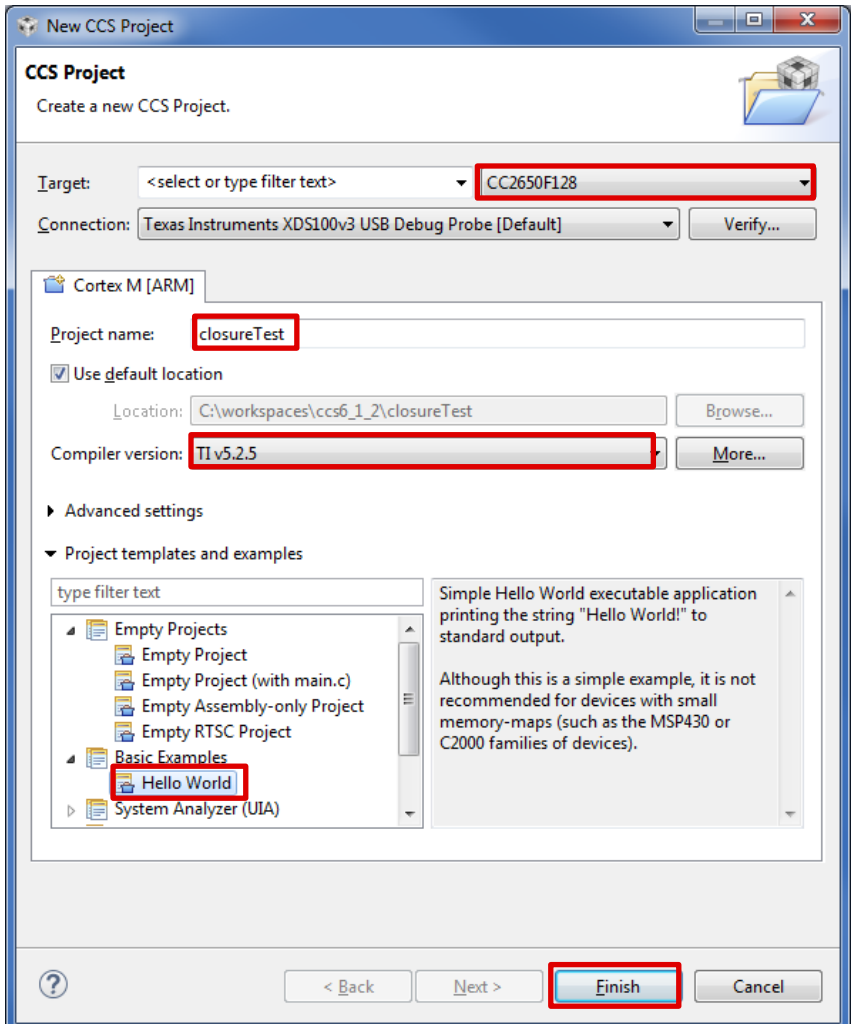
B. Select CC2605F128 Target

C. “closureTest” as the Project name

D. Make sure a TI compiler is selected

E. Select “Hello World”

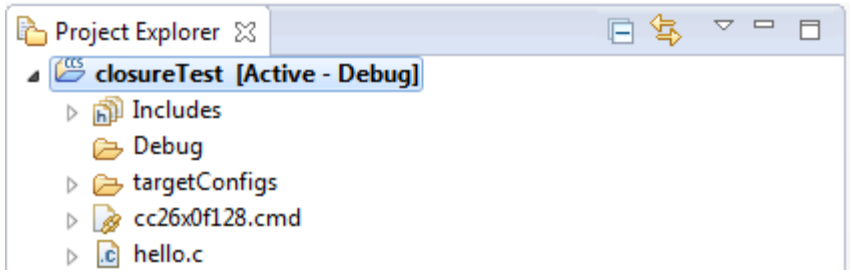
F. And “Finish”



Please note, currently there is no simple way to remove the .cfg from a CCS project that includes a TI-RTOS .cfg file. That’s why we start with a CCS project without a .cfg file.

Consume Closure Step 2: CCS Project + TI-RTOS Closure

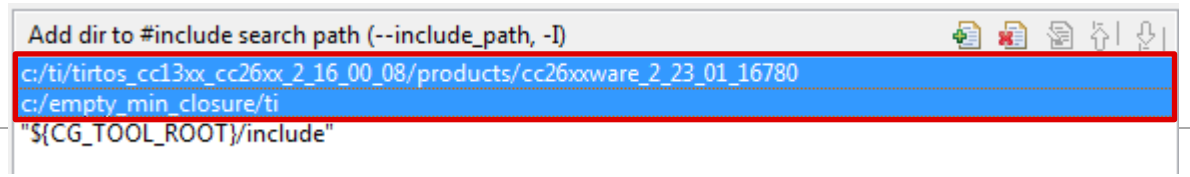
Here is the resulting project. Note the small CCS( ) on the project icon. Applications that have a .cfg file are denoted with a small RTSC ( ) instead.



Now the project needs to be modified to point to the closure in C:\empty_min_closure.

a. Add the following include search paths into the project (CCS->Project->Properties->Build->Arm Compiler->Include Options) before the codegen include. Make sure the order is as shown!!

```
c:/ti/tirtos_cc13xx_cc26xx_2_16_00_08/products/cc26xxware_2_23_01_16780
c:/empty_min_closure/ti
```



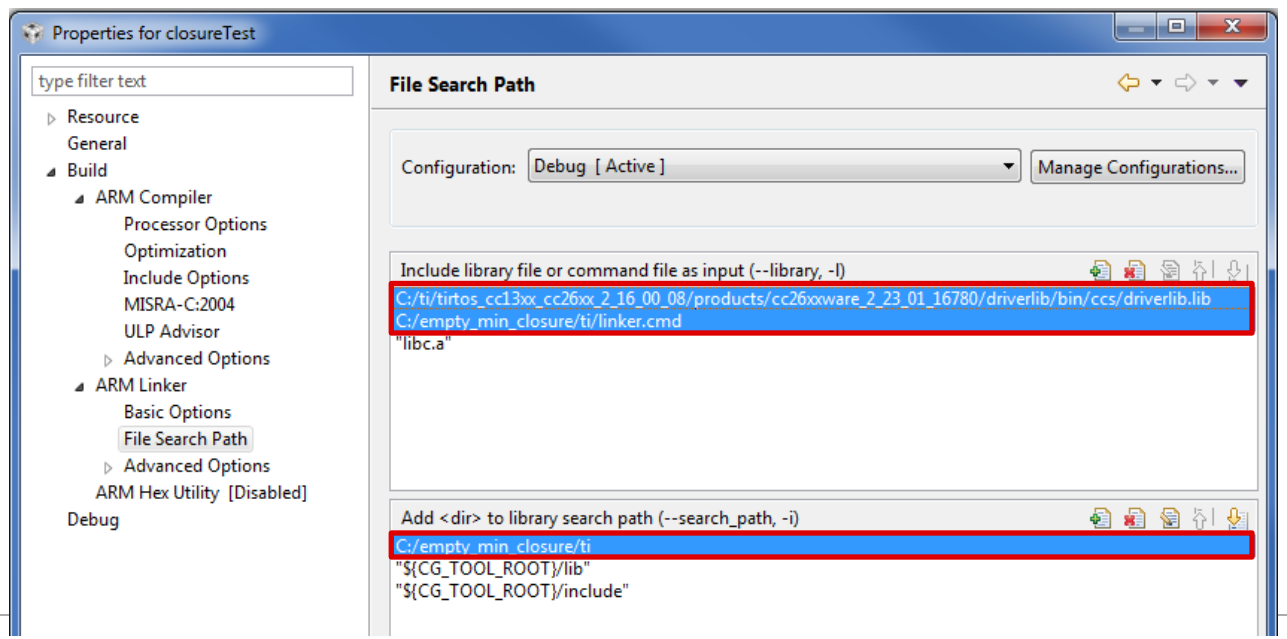
Consume Closure Step 3: CCS Project + TI-RTOS Closure

a. Add the following library and linker file into the project (CCS->Project->Properties->Build->Arm Linker>File Search Path) before “libc.a”. Make sure the order is as shown!!

```
C:/ti/tirtos_cc13xx_cc26xx_2_16_00_08/products/cc26xxware_2_23_01_16780/driverlib/bin/ccs/driverlib.lib
C:/empty_min_closure/ti/linker.cmd
```

b. Add the following library search path in the same window. Make sure the order is as shown!!

```
C:/empty_min_closure/ti
```



Consume Closure Step 4: CCS Project + TI-RTOS Closure

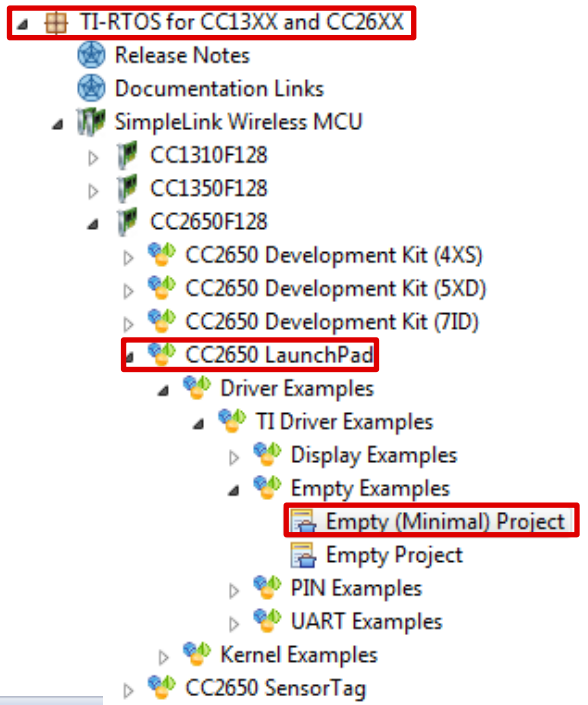
Let's change the example to be exactly like the empty_min in TI-RTOS.

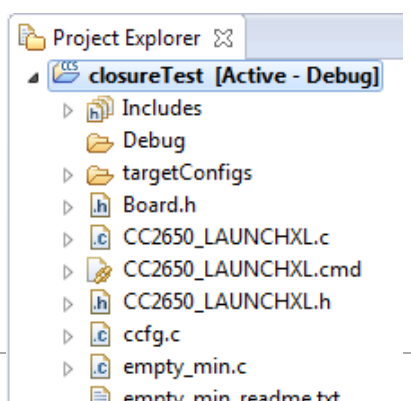
a. Import the LaunchPad's empty_min example from TI-RTOS.

b. Copy/paste the following files from this project into the closureTest (note: do not copy the empty_min.cfg)
   - Board.h
   - CC2650_LAUNCHXL.c
   - CC2650_LAUNCHXL.cmd
   - CC2650_LAUNCHXL.h
   - ccfg.c
   - empty_min.c
   - empty_min_readme.txt

c. Remove the following "old" files from the closureTest
   - cc26x0f128.cmd
   - main.c

d. Build and run the example. Example Finished!



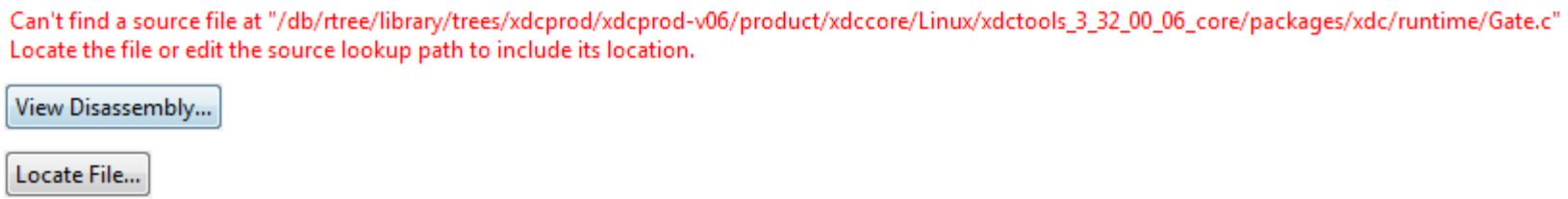


Debugging: TI-RTOS Source Level Debugging

TI-RTOS Source Level Debugging Option 1

When you build your project, you point the project to the TI-RTOS closure (as noted in the previous slides). However the source files (e.g. Task.c) are still in the TI-RTOS product. So if you want to step into the source when you are debugging, you need to point the IDE at the corresponding file in the TI-RTOS product.

For example, a breakpoint was set in xdc_runtime_Gate_enterSystem__E. When the breakpoint was hit, the following message was displayed.



To locate, look at the end of the string. It tells you where the file is. Hit the “Locate File…” button and navigate to file (in this case c:\ti\xdctools_3_32_00_06_core\packages\xdc\runtime). This occurs because the libraries where built on servers within Texas Instruments and the file information is embedded into the library.

The TI-RTOS generated .c file is still in the closure making directory. For this document’s example, that would be c:\ClosureExample\configPkg\package\cfg\empty_min_pem3.c. It is not in the generated closure. This file contains small generated helper functions and TI-RTOS’s malloc/free/calloc/etc. functions.

Debugging: TI-RTOS Source Level Debugging

TI-RTOS Source Level Debugging Option 2

If the following “--filter=0”is added in the closure’s makefile (makefile_cc13xx_cc26xx in this example), all the TI-RTOS source code is added into the TI-RTOS closure.

```
72  ti: local_clean ti_clean
73          $(MKDIR) $(CLOSURE_DIR)
74          $(XS) --xdcpath="$(XDCPATH)" xdc.tools.configuro -v -o configPkg      \
75              -t ti.targets.arm.elf.M3 -p $(PLATFORM) -c $(TI_COMPILER_DIR)     \
76              --compileOptions " --include_path=\"$(CC26XXWARE_INSTALL_DIR)\" -o3
77              $(APPLICATION_CFG_FILE)
78          $(XS) xdc.tools.closure -v --filter=0 -d $(CLOSURE_DIR)/ti ./configPkg
```

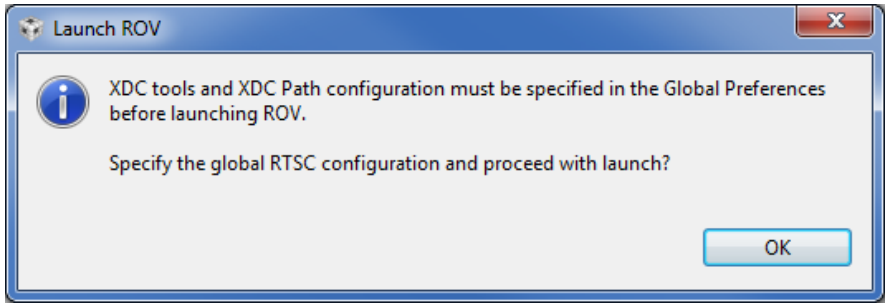
The IDE can be pointed at the TI-RTOS source files (including the generated .c file) within the TI-RTOS closure. The downside of this approach is the size of the non-filtered closure directory (e.g. ~110 MB vs ~30MB for the default).

From a consistency stand-point, pointing to the closure’s source files is preferred if the disk space is acceptable.

Debugging: ROV

RTOS Object Viewer (ROV)

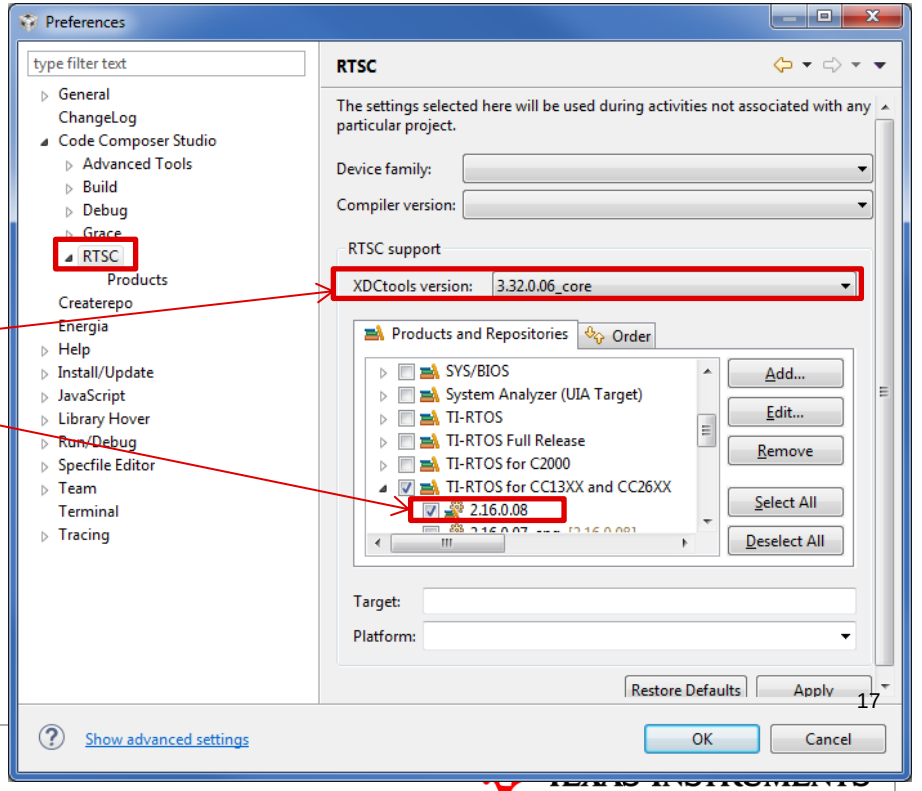
ROV in CCS will still work, but an extra step is required. When ROV is opened, the XDCtools and TI-RTOS product must be specified.



Option 1

This can be done in Windows->Preferences and select the appropriate versions.

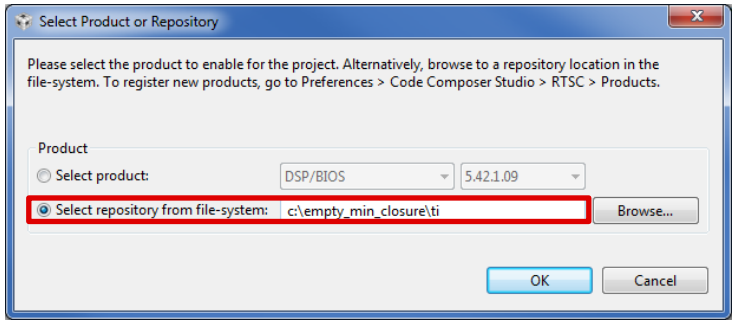
Please remember to change this if you use a new version of TI-RTOS and/or XDCtools.

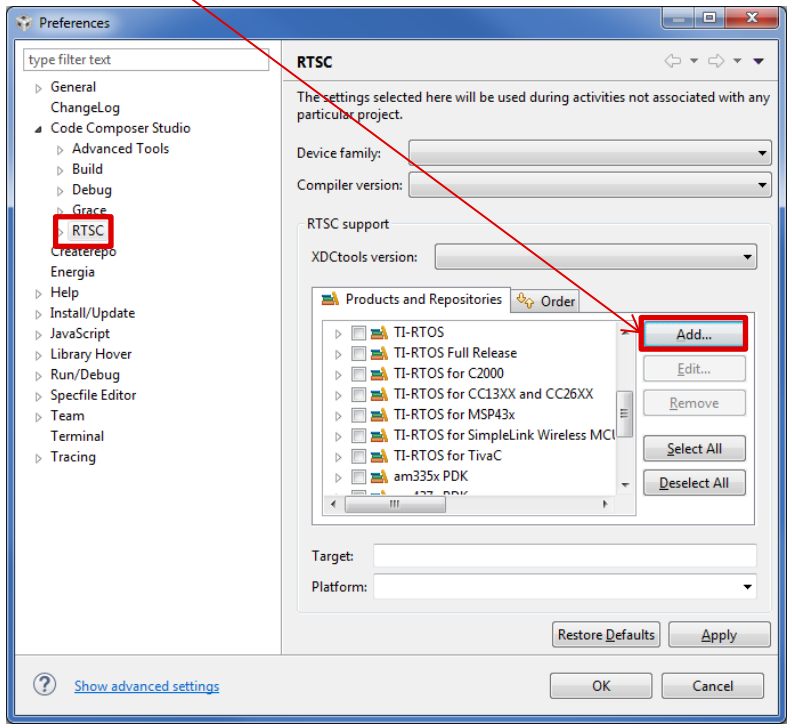


Debugging: ROV

Option 2

Like Option 2 in the TI-RTOS source level debugging on a previous slide, if “--filter=0” is used in the makefile, useful ROV files are not filtered out. Instead of pointing to XDCtools and TI-RTOS installations, a new Repository can be made that points to the TI-RTOS closure. Click “Add…”, select a repository from the file-system and enter “c:\empty_min_closure\ti”.





There is an open bug with this approach. “Constructed” objects (e.g. Task_construct) are not shown.

Please remember to change this if a different closure directory is used.

Configuration File Tricks

Making a flexible .cfg file

If a .cfg is used for multiple projects, it is sometimes nice to add logic in the .cfg to allow some flexibility. Here are a few ways to add logic into the .cfg file. Please note, the configuration file is JavaScript.

- **Device**: The Program.cpu.deviceName can be examined. For example:

```
if (Program.cpu.deviceName.match(/CC26/)) {
    ROM.romName = ROM.CC2650;
}
else if (Program.cpu.deviceName.match(/CC13/)) {
    ROM.romName = ROM.CC1350;
}
else {
    Program.$logError("Device not support: " + Program.cpu.deviceName, Program, "cpu.deviceName");
}
```

- **Toolchain**: The Program.build.target variable can be examined. For example:

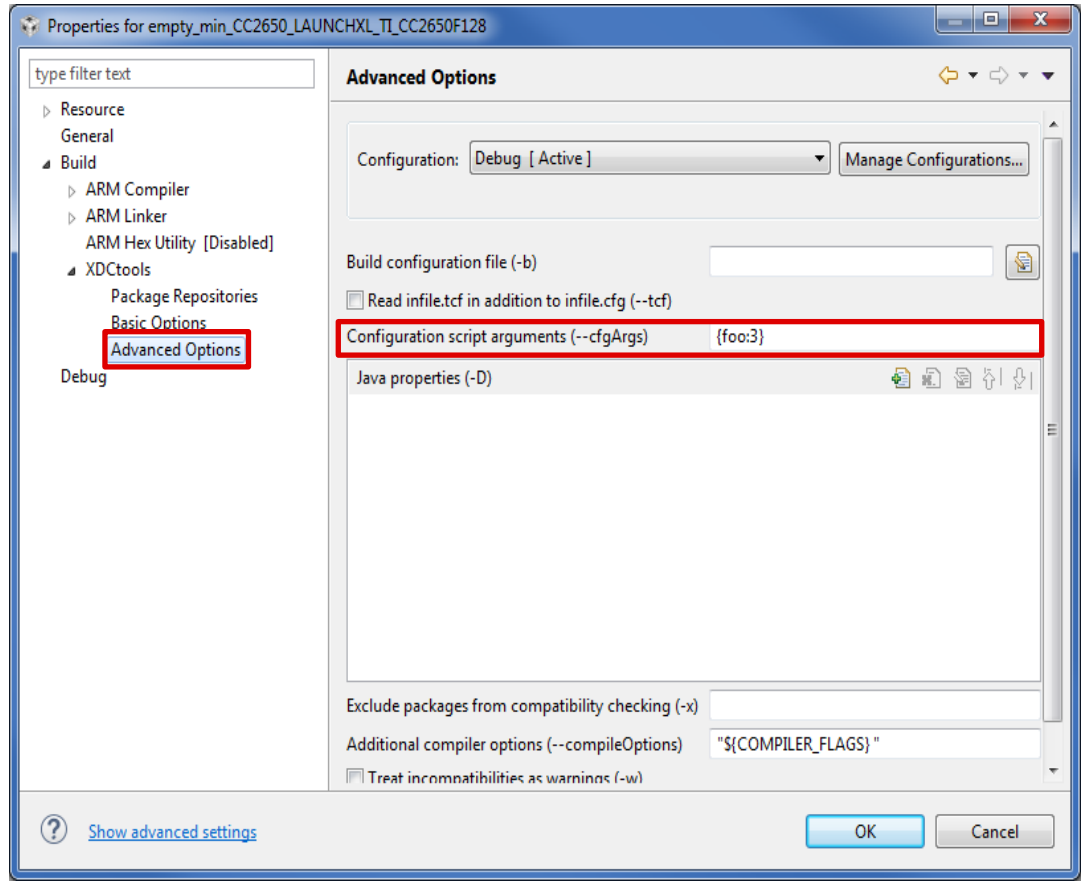
```
/* Enable Semihosting for GNU targets to print to CCS console */
if (Program.build.target.$name.match(/gnu/)) {
    var SemiHost = xdc.useModule('ti.sysbios.rts.gnu.SemiHostSupport');
}
```

Configuration File Tricks

– **cfgArgs**: In the XDCtools Advanced Options, you can specify a name/value pair that can be accessed in the .cfg file. Please note, this feature really not applicable for a TI-RTOS closure. It can be used when the .cfg file include in multiple projects.

  Here code in the .cfg that accesses the setting

```
if (Program.build.cfgArgs.foo == 3) {
      …
```



Outstanding Items to be added in to this ppt/zip

The following items will be added during late Q1/early Q2 2016. The items are in the expected order of completion. Please use the TI-RTOS forum (https://e2e.ti.com/support/embedded/tirtos/f/355) for additional requests or to check on the status of the additions.

IAR Support

Building a TI-RTOS closure with IAR is described, but the steps to consume are not yet. Also viewing source files and ROV in IAR needs to be detailed.

Other TI-RTOS products

Include steps/files for TI-RTOS TivaC, CC3200, MSP, and C2000 products.

Host Validation

Need to validate with Linux and MacOS. Slight changes in the makefile are expected!

Non-IDE projects

Describe how to build non-IDE projects (e.g. makefile based).

Issues

Pre-2.16.00 Version of TI-RTOS for CC13xx and CC26xx

Do we need to support pre-2.16.00 releases? If so, we need to add some steps to get the PDM libraries properly. There was a fix in 2.16.00 that addressed this.

Missing Driverlib libraries in TI-RTOS closure

There is an enhancement request to include the driverlib libraries in the TI-RTOS closure. This will allow a consumer of the TI-RTOS closure to not need the TI-RTOS product.

TIRTOS-434: “Missing Driverlib libraries in TI-RTOS closure”

ROV bug with Closure Repository

When using option 2 to view ROV, some of the constructed objects are not shown. There is a bug opened against TI-RTOS for this.

Version

1.0: Initial Version (2/28/2016)