Post-build script to populate the CCFG's CRC of CC2340R5 images built with TIClang toolchain

Introduction

TIClang toolchain does not populate the CCFG's CRC of the images built for CC2340R5. A post build script should be executed to ensure this.
This guide describes the steps to be followed in order to implement such script.

Set-up steps

1- Import and build the project

As usual – ensure the project builds properly before going to next steps.
The files generated by SysConfig during the build will be reused for the next steps.

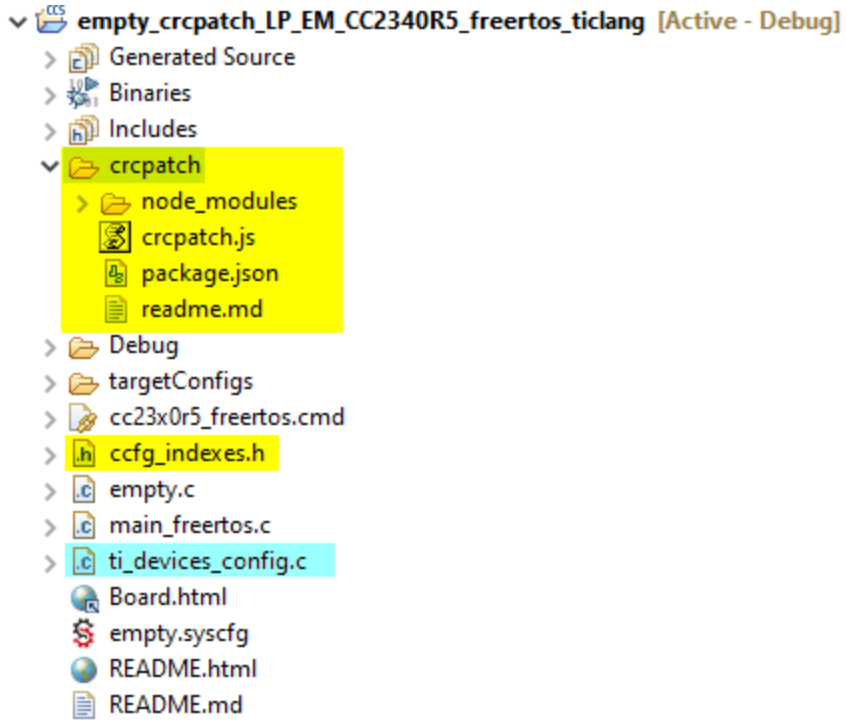
2- Add the required files to the project

1. Unzip the content of "post_build_script.zip"
2. Paste the crcpatch folder and the ccfg_indexes.h file at the root of the project (see the files highlighted in yellow on the screenshot at next step).

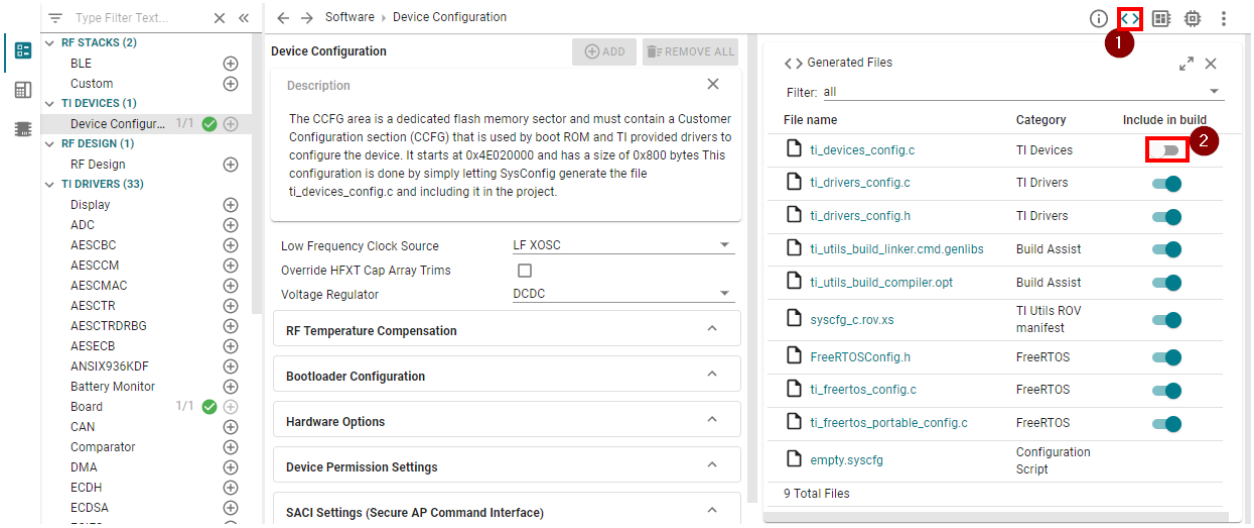
3- Modify the file ti_devices_config.c

The file ti_devices_config.c is generated by SysConfig. It can be found in the project under [Debug|Release] > syscfg (note: the folder where the files generated by SysConfig depend on the compilation target selected)

1. Copy-paste the file ti_devices_config.c at the project root (see the file highlighted in blue)



2. Disable the generation of the ti_devices_config.c in SysConfig



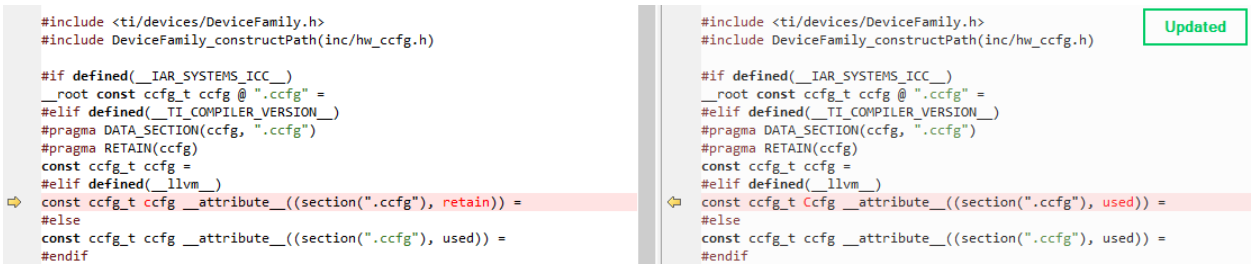
3. In the newly created ti_devices_config.c, replace the following line:

```
const ccfg_t ccfg __attribute__((section(".ccfg"), retain)) =
```

By:

```
const ccfg_t Ccfg __attribute__((section(".ccfg"), used)) =
```

Below is a screenshotshowing the same



4- Modify the main_freertos.c file

Add the following lines in the file:

```
#include <ti/devices/DeviceFamily.h>
#include DeviceFamily_constructPath(inc/hw_ccfg.h)
#include "ccfg_indexes.h"

extern const volatile ccfg_t Ccfg;
```

5- Modify the linker command file (cc23x0r5_freertos.cmd)

Add the following lines at the end of the file

```
/* Create global constant that points to top of stack */

/* CCS: Change stack size under Project Properties    */

__STACK_TOP = __stack + __STACK_SIZE;


#include "ccfg_indexes.h"


/* These calculations presume BYTES_PER_CCFG == sizeof(ccfg_t).  main.c has */

/* a compile time check for this condition                                  */

crc_ccfg_bootCfg_start    = Ccfg + (CRC_BOOTCFG_START * BYTES_PER_CCFG);

crc_ccfg_bootCfg_stop     = Ccfg + (CRC_BOOTCFG_STOP  * BYTES_PER_CCFG) + BYTES_PER_CCFG;

crc_ccfg_bootCfg_value    = Ccfg + (CRC_BOOTCFG_VALUE * BYTES_PER_CCFG);


crc_ccfg_genCfg_start     = Ccfg + (CRC_GENCFG_START * BYTES_PER_CCFG);

crc_ccfg_genCfg_stop      = Ccfg + (CRC_GENCFG_STOP  * BYTES_PER_CCFG) + BYTES_PER_CCFG;

crc_ccfg_genCfg_value     = Ccfg + (CRC_GENCFG_VALUE * BYTES_PER_CCFG);


crc_ccfg_userRecord_start = Ccfg + (CRC_USERRECORD_START * BYTES_PER_CCFG);

crc_ccfg_userRecord_stop  = Ccfg + (CRC_USERRECORD_STOP  * BYTES_PER_CCFG) +
BYTES_PER_CCFG;
```

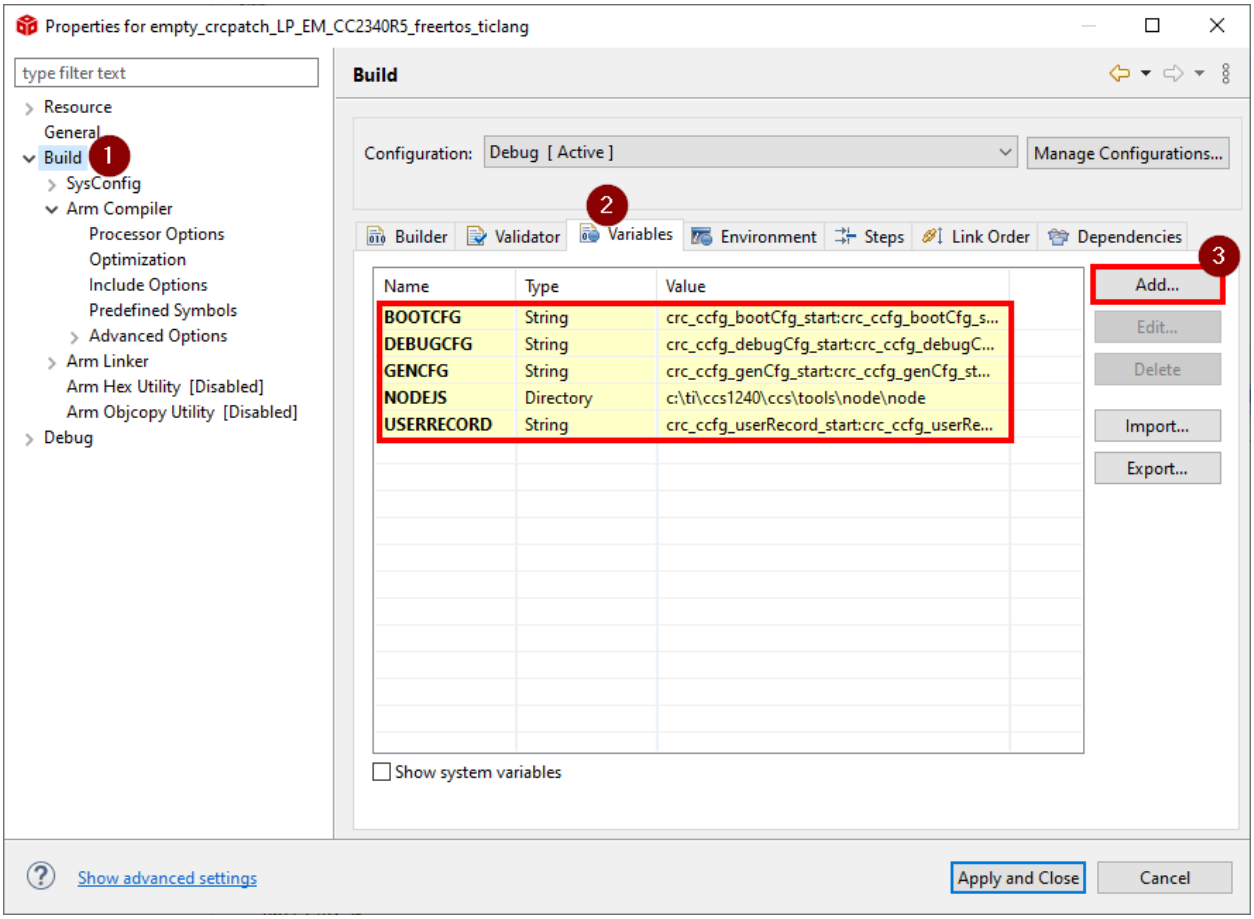
```
crc_ccfg_userRecord_value = Ccfg + (CRC_USERRECORD_VALUE * BYTES_PER_CCFG);


crc_ccfg_debugCfg_start   = Ccfg + (CRC_DEBUGCFG_START * BYTES_PER_CCFG);
crc_ccfg_debugCfg_stop    = Ccfg + (CRC_DEBUGCFG_STOP  * BYTES_PER_CCFG) + BYTES_PER_CCFG;
crc_ccfg_debugCfg_value   = Ccfg + (CRC_DEBUGCFG_VALUE * BYTES_PER_CCFG);
```

6- Add the execution of the crcpatch.js script as post-build step

1. Right click on the project > Properties > Variables
2. Add the following variables

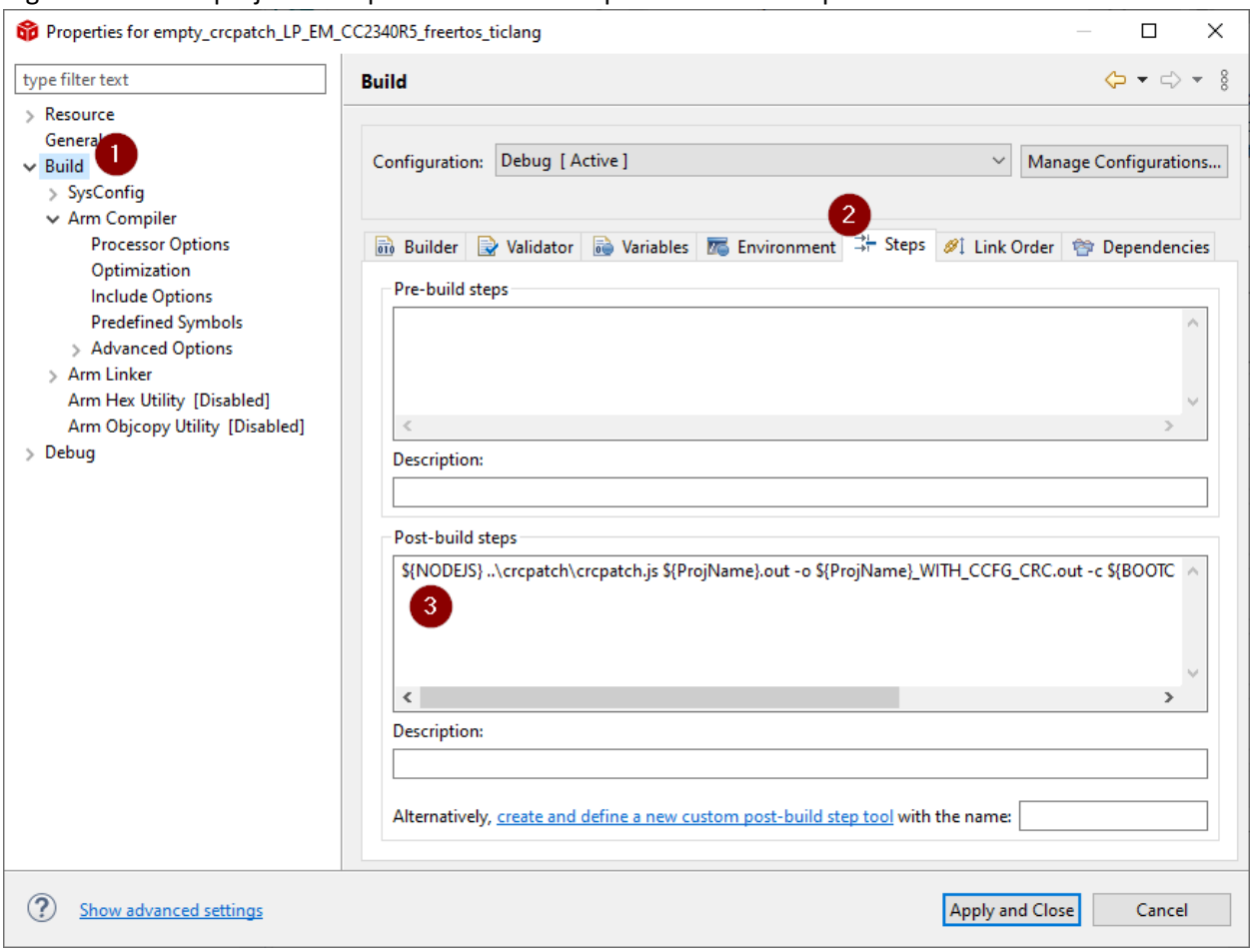
Name	Type	Value
BOOTCFG	string	crc_ccfg_bootCfg_start:crc_ccfg_bootCfg_stop:crc_ccfg_bootCfg_value
DEBUGCFG	string	crc_ccfg_debugCfg_start:crc_ccfg_debugCfg_stop:crc_ccfg_debugCfg_value
GENCFG	string	crc_ccfg_genCfg_start:crc_ccfg_genCfg_stop:crc_ccfg_genCfg_value
NODEJS	dir	c:\ti\ccs1240\ccs\tools\node\node
USERRECORD	string	crc_ccfg_userRecord_start:crc_ccfg_userRecord_stop:crc_ccfg_userRecord_value



Click on “Apply and Close”

7- Add the execution of the crcpatch.js script as post-build step

1. Right click on the project > Properties > Build > Steps > Post-build steps



2. Add the following line in the Post-build steps

```
${NODEJS} ..\crcpatch\crcpatch.js ${ProjName}.out -o ${ProjName}_WITH_CCFG_CRC.out -c
${BOOTCFG} -c ${GENCFG} -c ${USERRECORD} -c ${DEBUGCFG}
```

This will lead to the execution of the script and the creation of a second .out file suffixed "_WITH_CCFG_CRC".

Click on "Apply and Close"

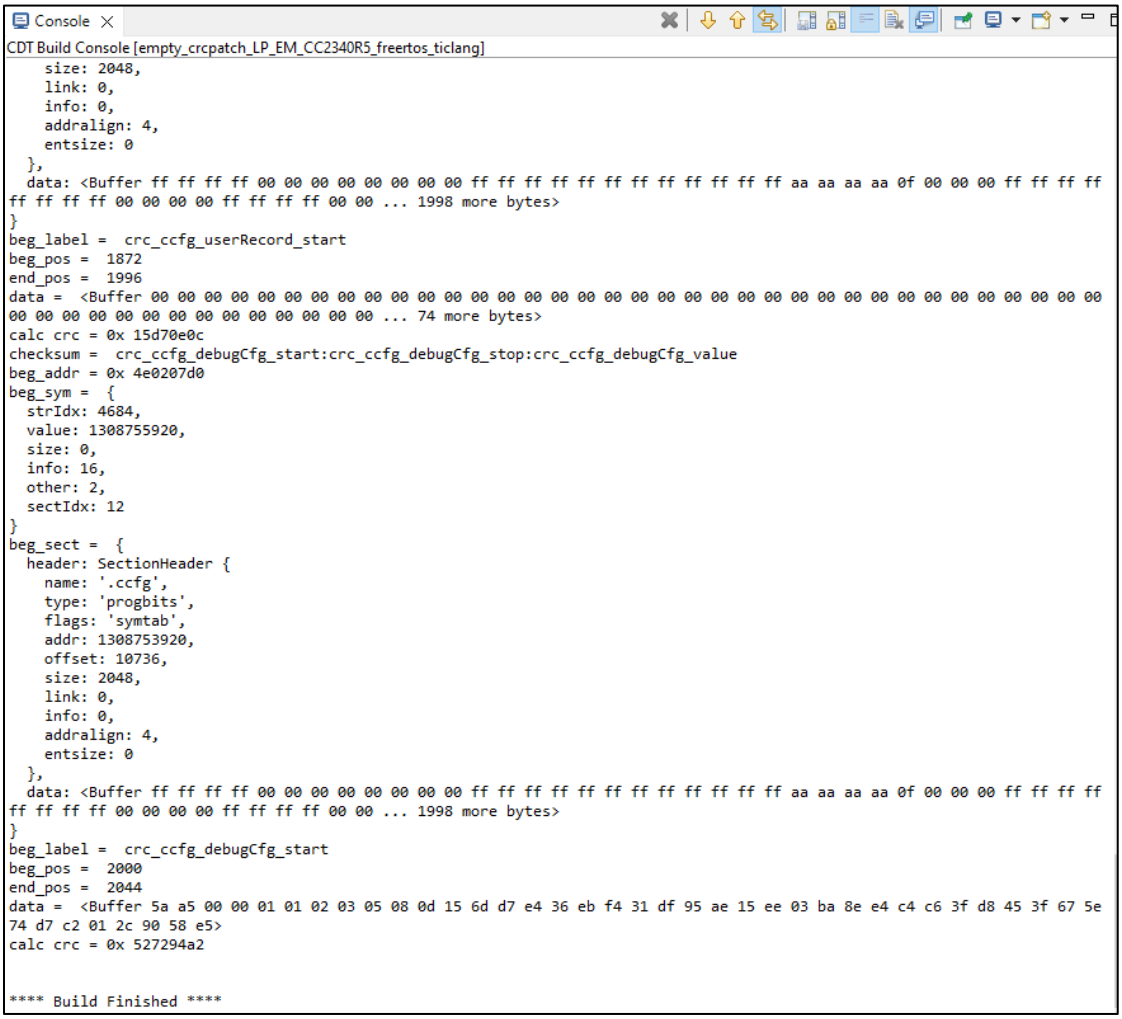
8- Re-Build the project

Right click on the project > Rebuild Project

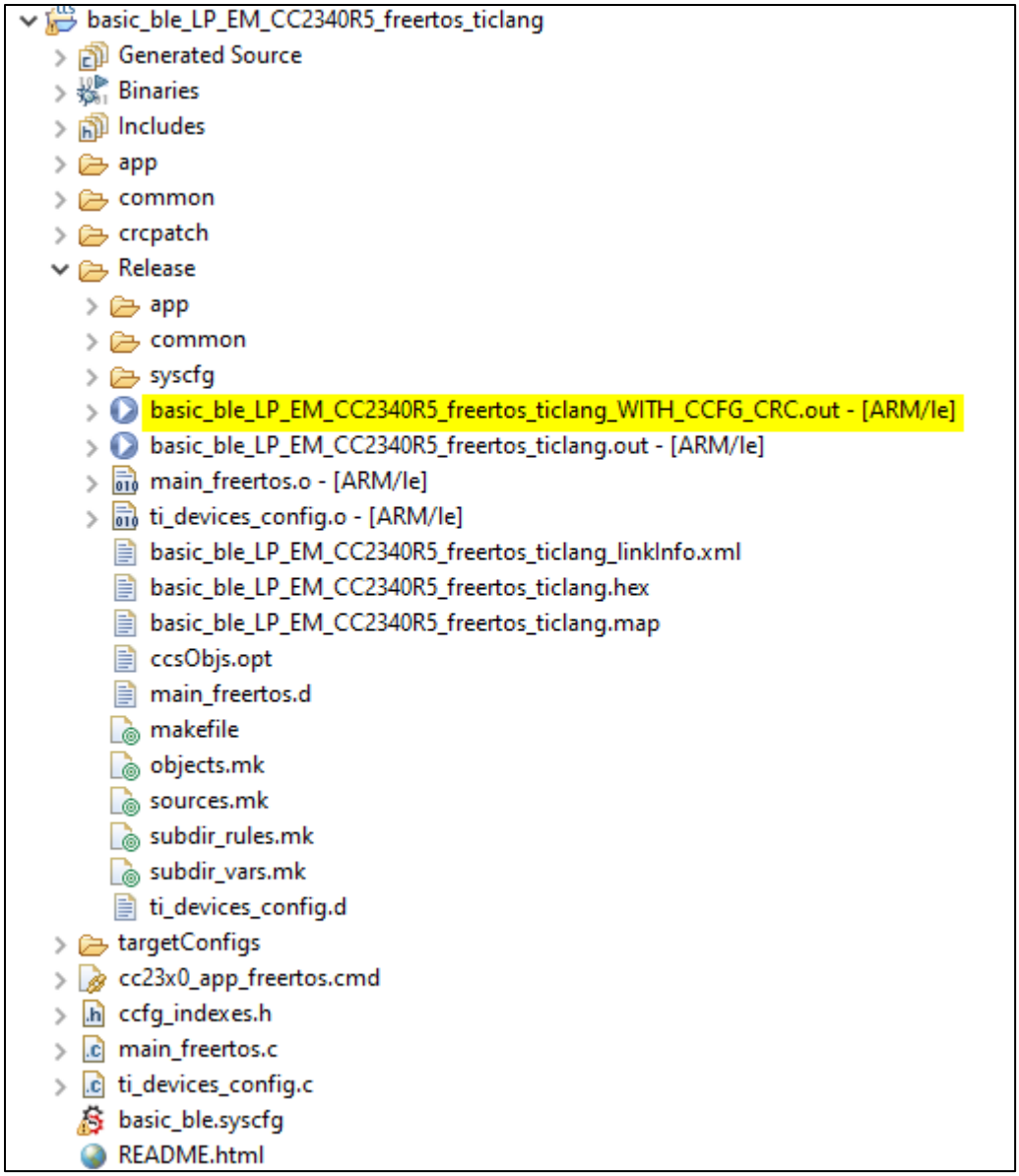
Expected results

- An error will be raised by the toolchain in case an issue occurs with the post-build steps

- The script is expected to output several lines of log showing the CRC calculated.
- An additional .out file should be produced. **Ensure this file is flashed on the device**.
- Log outputted (partial)



- An additional .out file is generated


basic_ble_LP_EM_CC2340R5_freertos_ticlang
Generated Source
Binaries
Includes
app
common
crcpatch
Release
app
common
syscfg
basic_ble_LP_EM_CC2340R5_freertos_ticlang_WITH_CCFG_CRC.out - [ARM/le]
basic_ble_LP_EM_CC2340R5_freertos_ticlang.out - [ARM/le]
main_freertos.o - [ARM/le]
ti_devices_config.o - [ARM/le]
basic_ble_LP_EM_CC2340R5_freertos_ticlang_linkInfo.xml
basic_ble_LP_EM_CC2340R5_freertos_ticlang.hex
basic_ble_LP_EM_CC2340R5_freertos_ticlang.map
ccsObjs.opt
main_freertos.d
makefile
objects.mk
sources.mk
subdir_rules.mk
subdir_vars.mk
ti_devices_config.d
targetConfigs
cc23x0_app_freertos.cmd
ccfg_indexes.h
main_freertos.c
ti_devices_config.c
basic_ble.syscfg
README.html