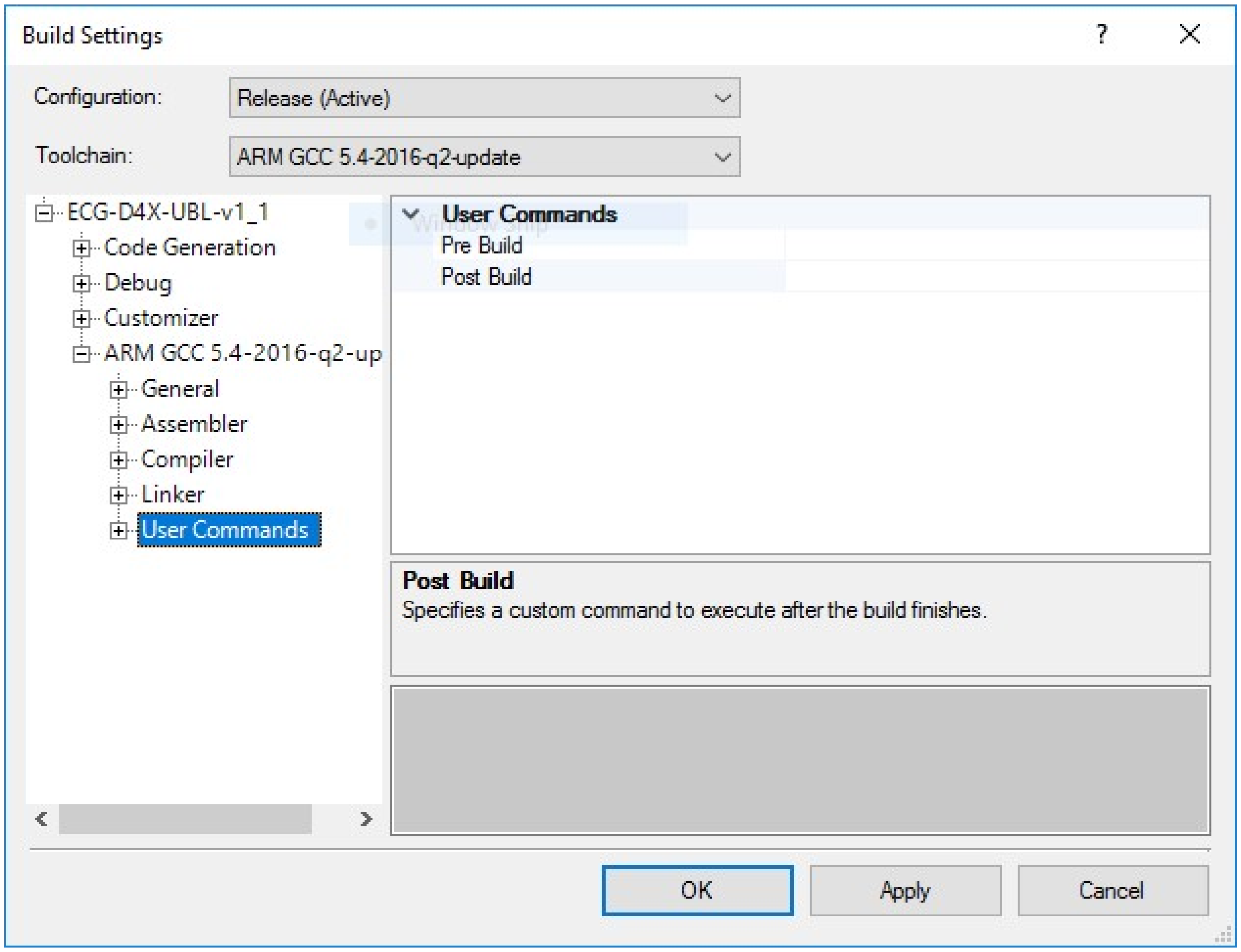
Some notes on pre/post build in Creator 4.1

I've not worked with pre-build, but assume all would apply. These notes are about my testing of post-build on Windows 10 using PSoC Creator 4.1. YMMV.

Background: we've implemented a post-build capability since Creator v2.2 by writing a "cyelftool" interceptor that invokes our custom perl scripts to do some post processing on generated hex, elf, and cyacd files. One major drawback to this has been the support headache to implement and test this into each generation of Creator. We've waited anxiously for v4.1 for the pre/post build support.

This morning I installed v4.1 and immediately went into the User Commands section as described in the Release Notes.


Build Settings

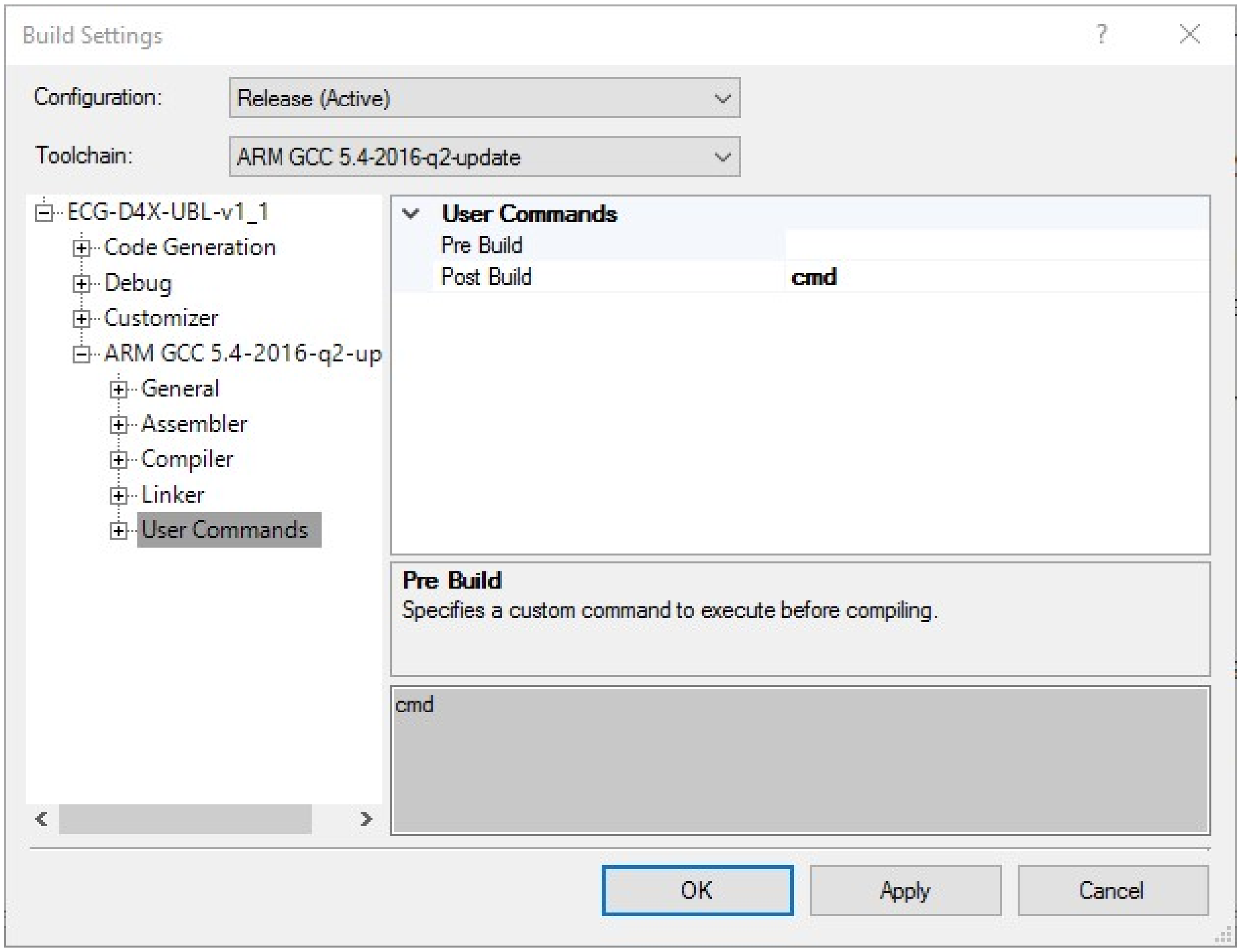
Configuration: Release (Active)

Toolchain: ARM GCC 5.4-2016-q2-update

Post Build
Specifies a custom command to execute after the build finishes.

I went into help (?). Looked around for more info, read the release notes again. Found no instructions or examples. Don't know "where I'll be", "environment", etc. when the post build is executed.

Thought I could maybe see that by interactively opening a command shell.



Configuration: Release (Active)

Toolchain: ARM GCC 5.4-2016-q2-update

User Commands

Pre Build	
Post Build	cmd

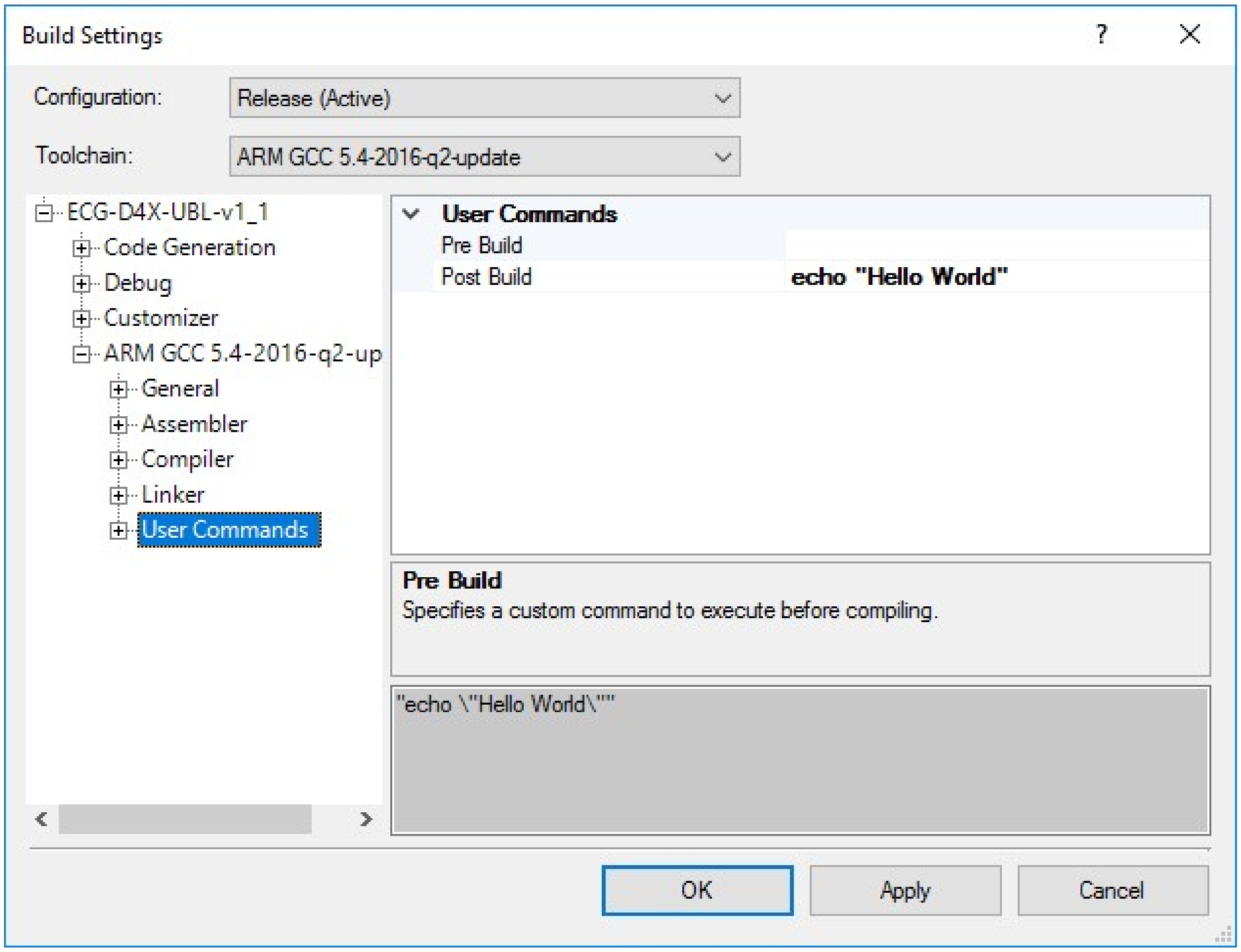
Pre Build
Specifies a custom command to execute before compiling.

```
cmd
```

Then hit Build.

DON'T try this. Locked up Creator tight as a drum. Only solution was task kill or system reset. I did a reset and approved canceling the 3 tasks.

So I played with 'echo' command (even though I thought it was inside cmd) and it worked.



Configuration: Release (Active)

Toolchain: ARM GCC 5.4-2016-q2-update

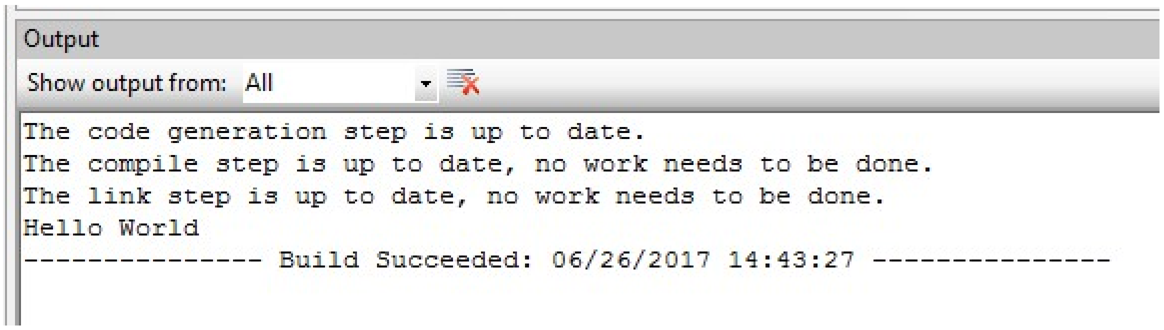
User Commands

Pre Build	
Post Build	**echo "Hello World"**

Pre Build
Specifies a custom command to execute before compiling.

```
"echo \"Hello World\""
```

Did a build and produced this output:



```
The code generation step is up to date.
The compile step is up to date, no work needs to be done.
The link step is up to date, no work needs to be done.
Hello World
--------------- Build Succeeded: 06/26/2017 14:43:27 ---------------
```

So post-build works, to a fashion. Now to make it do something.

I wanted to start with the simple expedient of copying the generated hex to a common area where I keep all my latest hex files for burning. I knew I didn't want to hard code the full filespec but wanted the build program to do it for me. I went looking for macros or text substitutions that might work.

By browsing through the other Build Settings I found:

${ProjectDir}

${ProcessorType}

${Platform}

${Config}

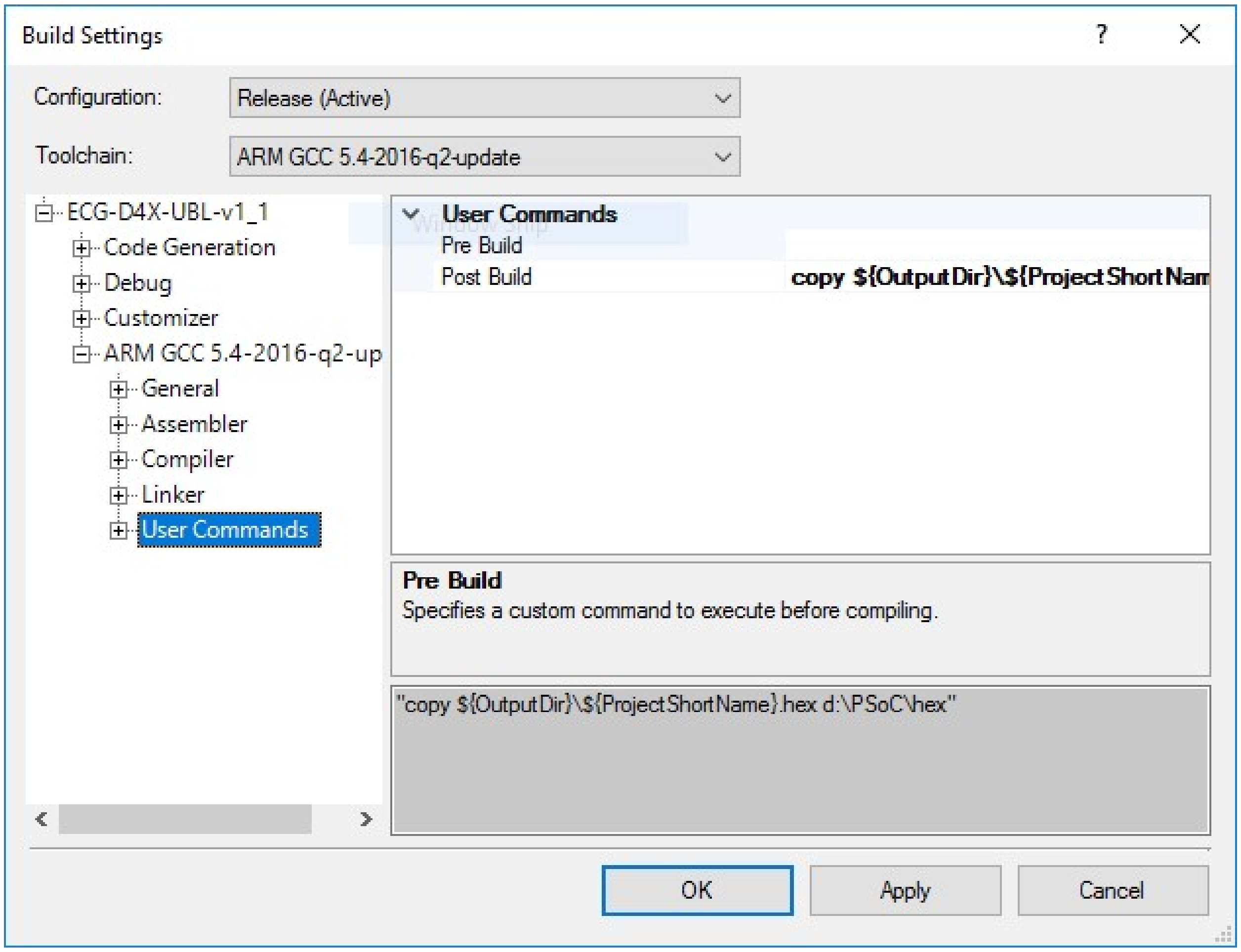
${ProjectShortName}

and finally

${OutputDir}

Most of these are self-explanatory. By playing with echo I found what they did and how (best) to use them. ${OutputDir} is the concatenation of the first four in order with backslashes inserted. The fact that ${ProjectDir} returns just a '.' Implies that we will be "living" in the cydsn directory when we start and everything will be addressed relatively to there.

So now I knew the macros I could use and wanted to do a simple copy.



Configuration: Release (Active)

Toolchain: ARM GCC 5.4-2016-q2-update

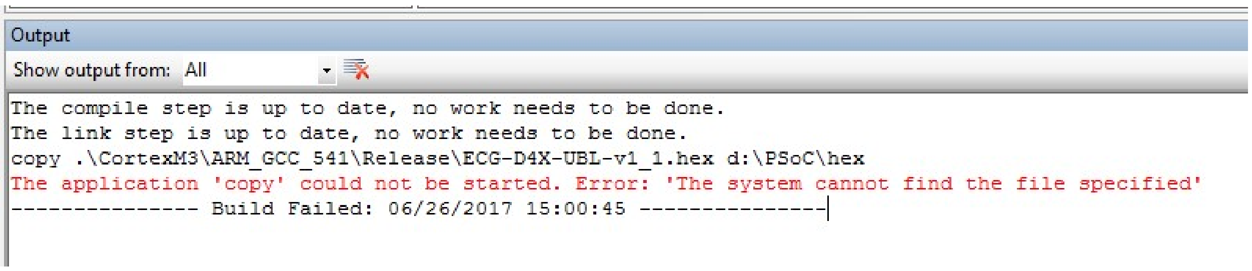
User Commands

Pre Build	
Post Build	copy ${OutputDir}\${ProjectShortNam

Pre Build
Specifies a custom command to execute before compiling.

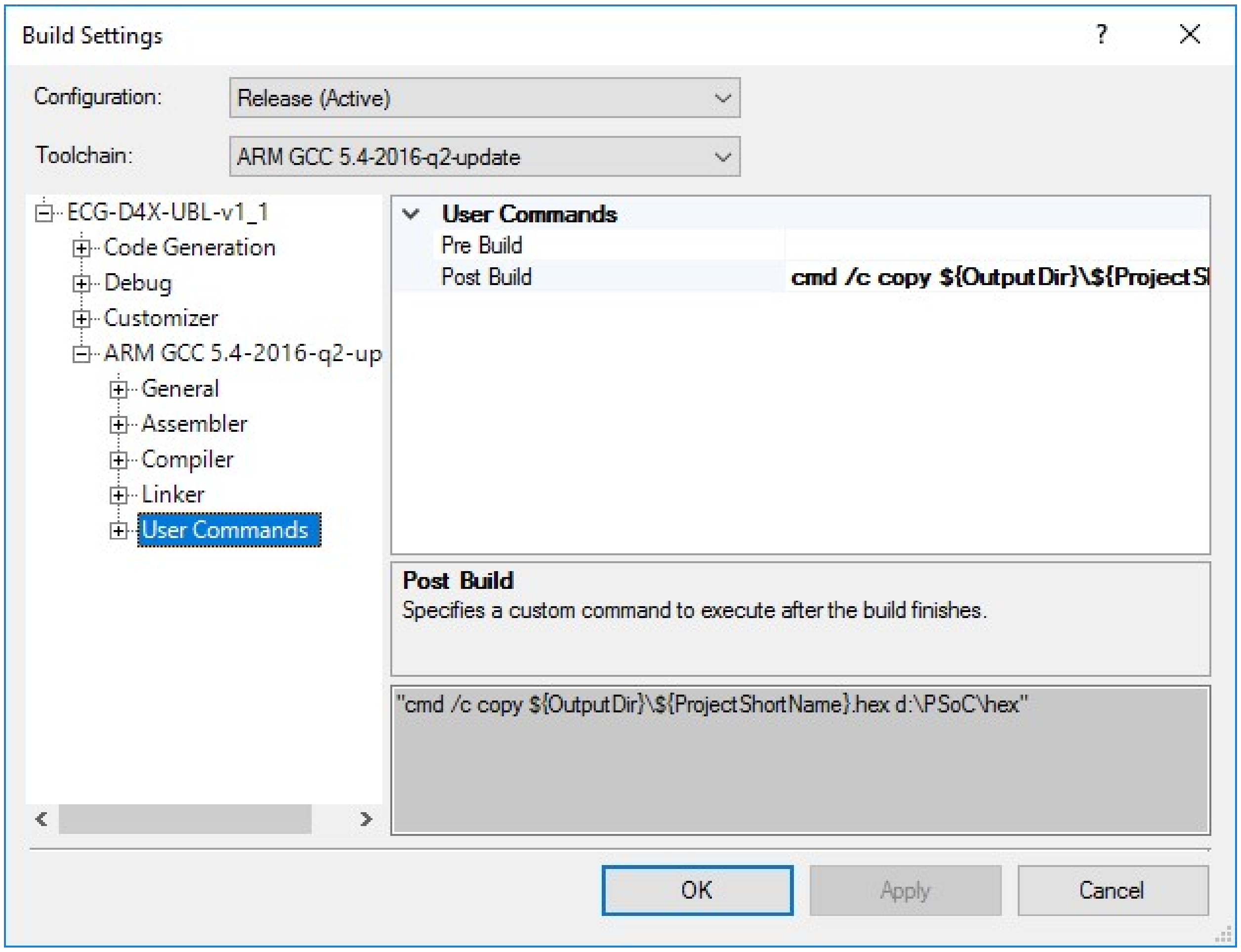
"copy ${OutputDir}\${ProjectShortName}.hex d:\PSoC\hex"

Did the build and got errors:



```
The compile step is up to date, no work needs to be done.
The link step is up to date, no work needs to be done.
copy .\CortexM3\ARM_GCC_541\Release\ECG-D4X-UBL-v1_1.hex d:\PSoC\hex
The application 'copy' could not be started. Error: 'The system cannot find the file specified'
--------------- Build Failed: 06/26/2017 15:00:45 ---------------
```

Dusting off my command processor knowledge I tried a hybrid of the "cmd" that locked-up Creator and the above.



Configuration: Release (Active)

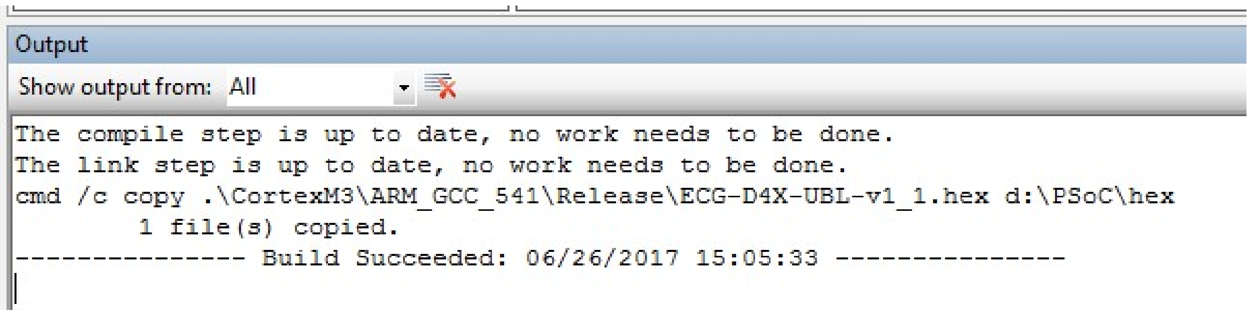
Toolchain: ARM GCC 5.4-2016-q2-update

Post Build: cmd /c copy ${OutputDir}\${ProjectS

Post Build
Specifies a custom command to execute after the build finishes.

"cmd /c copy ${OutputDir}\${ProjectShortName}.hex d:\PSoC\hex"

Success!!



```
The compile step is up to date, no work needs to be done.
The link step is up to date, no work needs to be done.
cmd /c copy .\CortexM3\ARM_GCC_541\Release\ECG-D4X-UBL-v1_1.hex d:\PSoC\hex
        1 file(s) copied.
--------------- Build Succeeded: 06/26/2017 15:05:33 ---------------
```

For a lot of people and projects this might be enough. But for our bootloader projects we like to copy the .hex and .elf together so bootloadable projects can refer to a central locatation. Also on our bootloadables we like to copy the .cyacd and then concatenate another hex file that contains a the hex representation of a simple file system that is stored in an external spi flash chip. Our bootloader knows how to burn the program to the PSoC and the other data to the spi flash chip. Our projects then use those to serve up web pages etc. Then even some complex projects do a little mangling of the hex or cyacd file with perl scripts.

For the dual copies we could do it with two source files in the copy. But two copy commands would look cleaner. A way to do the concatenations could be found but it might be easier to see/use with two commands. Either way we could not figure out how to issue TWO post-build commands. Perhaps a separator or something but we didn't want to spend all day playing with undocumented syntax (ahem Cypress).

So we decided that a set of generalized perl scripts with passed command line arguments kept in a centralized depository would be ok.

So we started to play with a perl script as a post-build command

We added .pl to the valid extensions and tried to execute a perl script directly. No joy.

So we executed "perl -v" as the user command and got expected results.

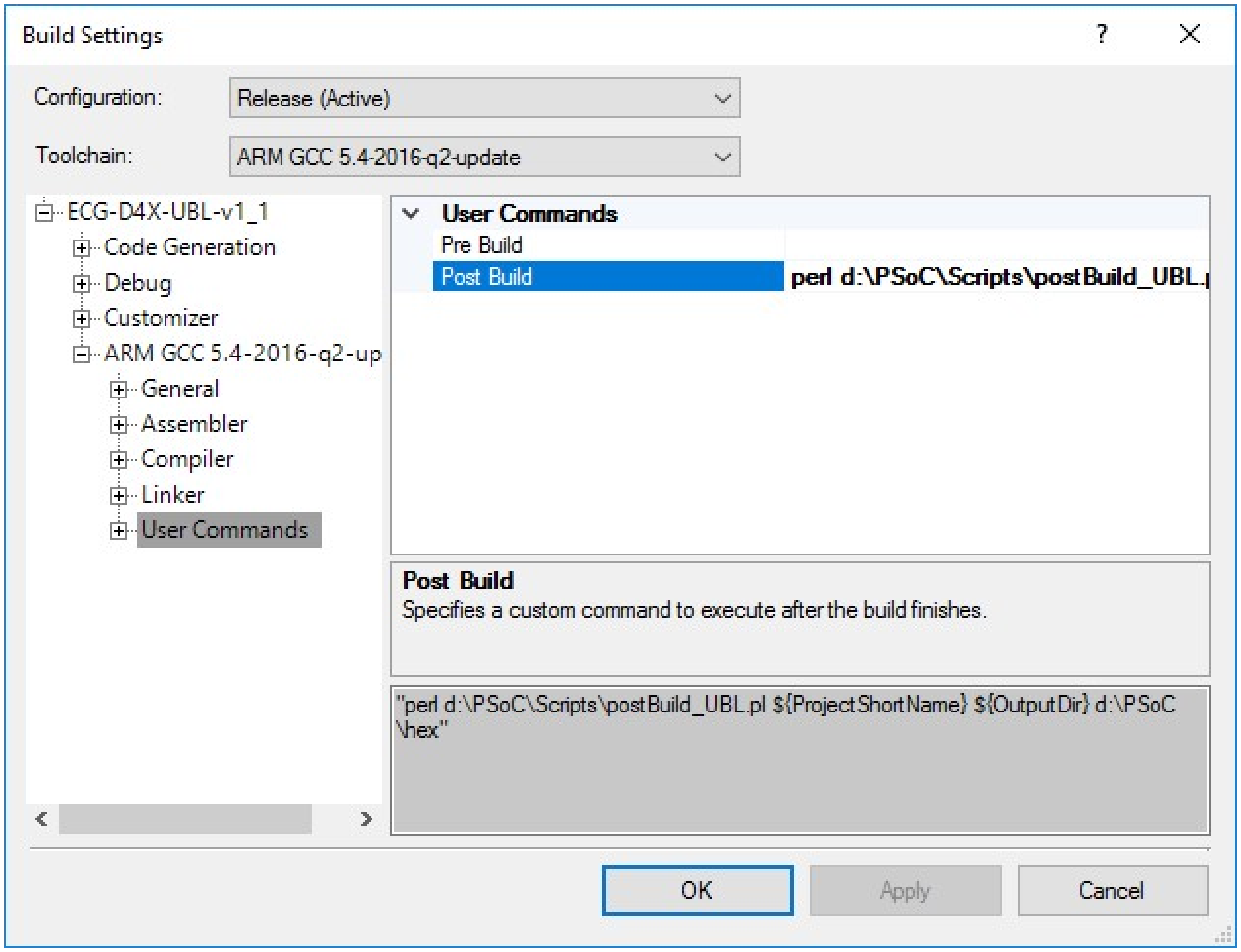
Then we wrote a test script that would dump the environment variables. Maybe Cypress had played nice and added all of the macros to the environment like some other IDEs had done. No joy. Possible change for later Cypress? That would also have been self-documenting on all of the available macros.

So we took our test script. Left in the environment dumper (commented out) and added argument parameters to copy .hex and .elf from the output directory to a common directory.

Here is the perl script:

```
#!/usr/bin/perl

#-----------------------------------------------------------------------
#
#     postBuild_UBL.pl - a perl script to complete at end of build
#
#     three args: projName projPath commonPath
#
#-----------------------------------------------------------------------

use strict;
use warnings;

use File::Copy;

#foreach my $key (sort keys(%ENV)) {
#  print "$key = $ENV{$key}\n";
#}
#print "\n";

my $projName    = shift(@ARGV);
my $projPath    = shift(@ARGV);
my $commonPath  = shift(@ARGV);

my $hexSpec     = $projName . ".hex";
my $elfSpec     = $projName . ".elf";

print "Copying:\n  $hexSpec\n  $elfSpec\n  to $commonPath ...\n";

copy($hexSpec , $commonPath);
copy($elfSpec , $commonPath);

print "Done!!\n";
```

Finally the following post build user command and the above script work together:



Configuration: Release (Active)

Toolchain: ARM GCC 5.4-2016-q2-update

User Commands

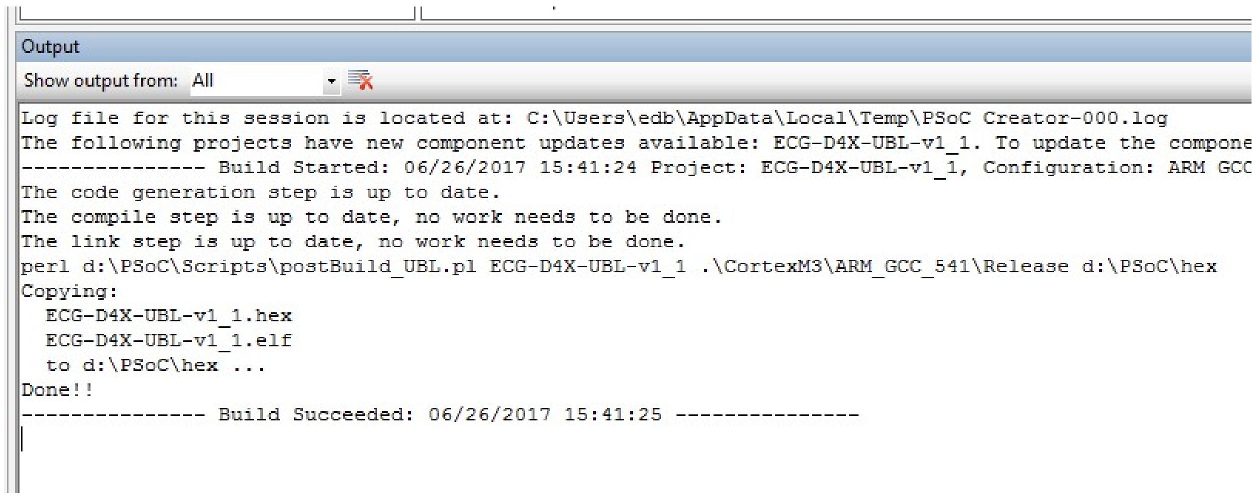
Pre Build

Post Build: perl d:\PSoC\Scripts\postBuild_UBL.

Post Build

Specifies a custom command to execute after the build finishes.

```
"perl d:\PSoC\Scripts\postBuild_UBL.pl ${ProjectShortName} ${OutputDir} d:\PSoC\hex"
```



```
Log file for this session is located at: C:\Users\edb\AppData\Local\Temp\PSoC Creator-000.log
The following projects have new component updates available: ECG-D4X-UBL-v1_1. To update the compone
--------------- Build Started: 06/26/2017 15:41:24 Project: ECG-D4X-UBL-v1_1, Configuration: ARM GCC
The code generation step is up to date.
The compile step is up to date, no work needs to be done.
The link step is up to date, no work needs to be done.
perl d:\PSoC\Scripts\postBuild_UBL.pl ECG-D4X-UBL-v1_1 .\CortexM3\ARM_GCC_541\Release d:\PSoC\hex
Copying:
  ECG-D4X-UBL-v1_1.hex
  ECG-D4X-UBL-v1_1.elf
  to d:\PSoC\hex ...
Done!!
--------------- Build Succeeded: 06/26/2017 15:41:25 ---------------
```

Just what I wanted. And I can use several stock scripts and parameterize them in the user command area.

Hope this helps others get started with this and hope Cypress can let us know ALL of the macros available and consider multiple commands for simple two step jobs along with maybe a better pop up editor for multiple lines or point to the documentation that I am not aware of.

- PSoC-Ed