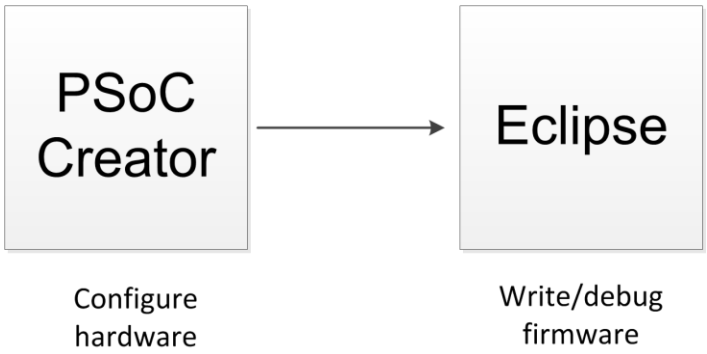
Using PSoC 4/PSoC 5LP Designs in Eclipse

PSoC Creator 4.x

Overview

Users can export designs from PSoC Creator and continue firmware development work in Eclipse. After the initial export, users can re-export additional hardware configuration changes from PSoC Creator to Eclipse. Eclipse will detect such changes and update the design accordingly. The reverse is not true. Do not attempt to import firmware changes made in Eclipse back into PSoC Creator.

Note The Export to IDE wizard does not apply to PSoC 6 devices. For PSoC 6 devices, use the Target IDEs section of the Build Settings dialog to choose third party IDEs for which to generate files.



Within Eclipse, users can archive their project in order to snapshot a design and supply it to another firmware developer using Eclipse.

The process to export a design from PSoC Creator for use in the Eclipse IDE involves the following high-level steps:

- Export from PSoC Creator
- Eclipse Installation Configuration
- Import into Eclipse
- Flashing and Debugging in Eclipse
- Eclipse Bootloader Support

Export a Design to Eclipse IDE

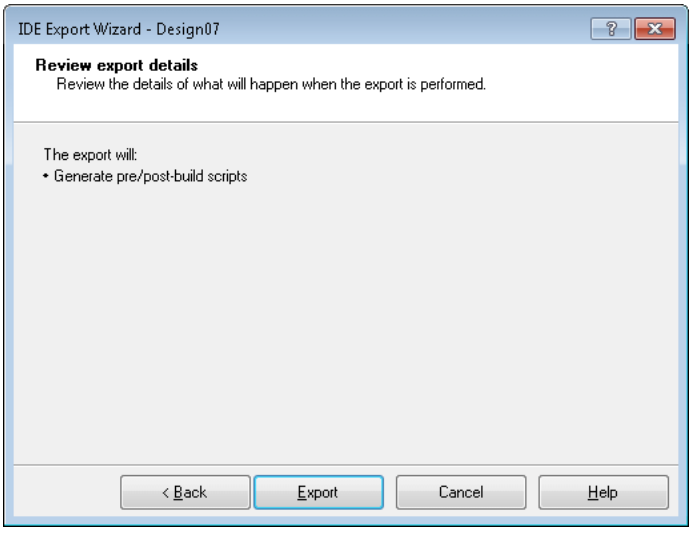
To export your PSoC Creator design for use in the Eclipse IDE, select the "Eclipse" option on the IDE Export Wizard dialog.

Note For PSoC 6 devices, use the Target IDEs section of the Build Settings dialog, and enable the **CMSIS Pack** option.

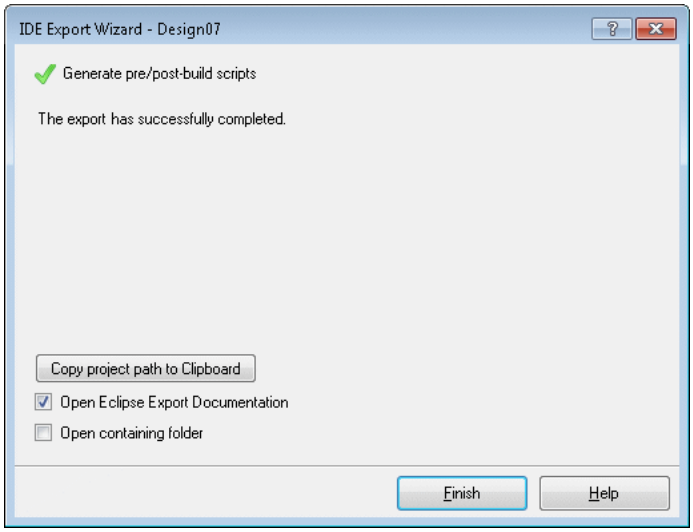


Note The Eclipse IDE option is not available for PSoC 3 projects.

The process is simple. The next step is to confirm that you want to export the design. Click **Export**.



Once the export completes, the final step displays.



- Use the Copy project path to Clipboard button to copy the project folder path so you can paste it in the Eclipse New Project dialog as the location path.
- Select the Open Eclipse Export Documentation option to display this document.
- Select the Open containing folder option to open a folder showing all the files that were exported.

Select the appropriate options and click **Finish** to complete the export process.

Notes:

- The project folder created in Eclipse shares file system folders with the original PSoC Creator project. This is required so that changes to merge regions in any Component files made within Eclipse by a firmware developer can be seen within PSoC Creator during any subsequent updates, and vice versa.
- When deleting a project in Eclipse, do not check the Delete project contents on disk check box. This will remove the file system folder contents, making them unavailable in PSoC Creator as well.

Next Steps:

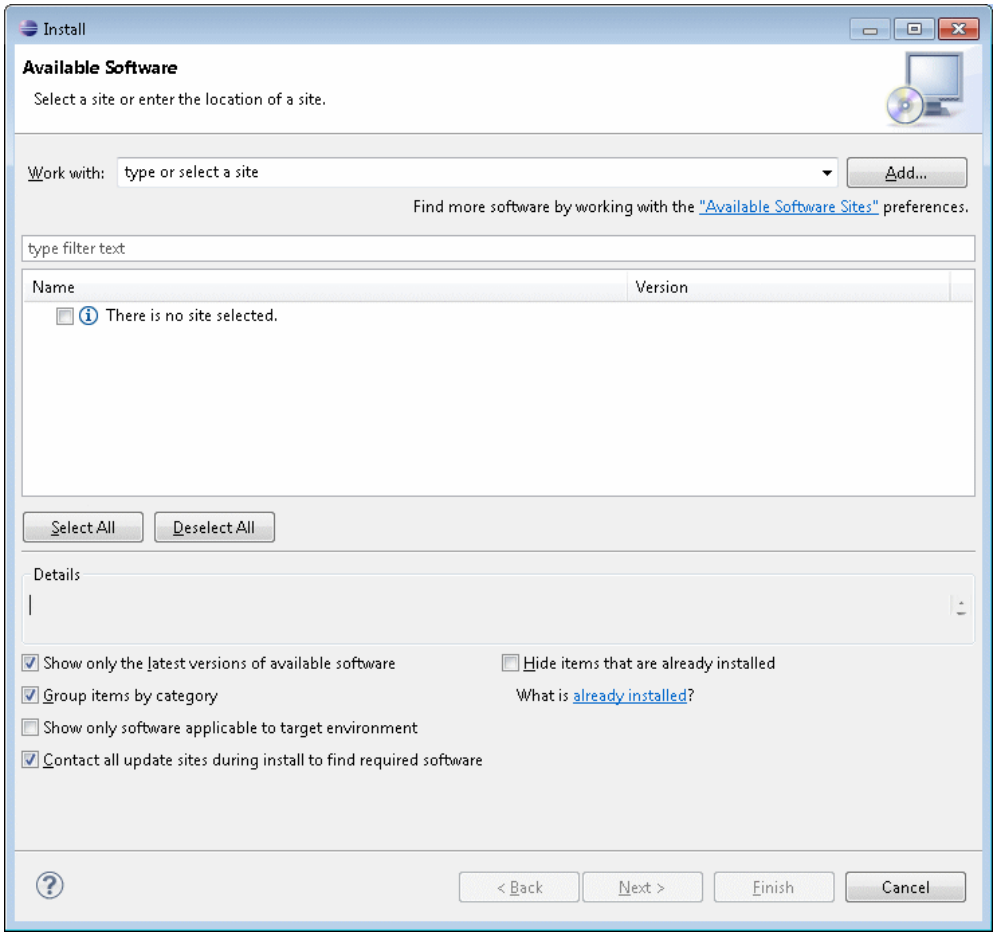
- Eclipse Installation Configuration
- Import into Eclipse
- Flashing and Debugging in Eclipse
- Eclipse Bootloader Support

Eclipse Installation Configuration

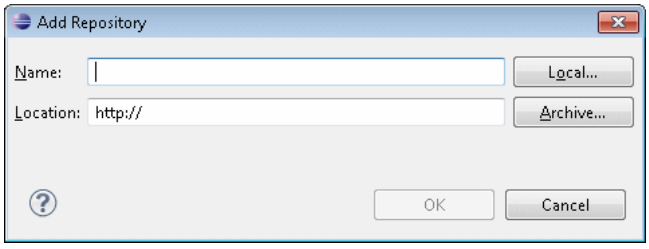
- When using the Eclipse IDE, you do not need a full PSoC Creator installation during firmware development.
- You must have Java JRE or JDK installed (Java version 7 or higher).
- You must have a GCC Arm compiler installed. The GNU Tools Arm Embedded toolset is available at https://launchpad.net/gcc-arm-embedded.
- You must install Eclipse CDT from http://www.eclipse.org/cdt. Look for the "Eclipse C/C++ IDE" download package. The Eclipse Luna and later releases support the Cypress-provided Eclipse feature.
  - If you are running a 32-bit operating system, download the 32-bit version of Eclipse.
  - If you are running a 64-bit operating system, you should download the version of Eclipse that matches the Java runtime version that you have installed; that is, 32-bit Eclipse for 32-bit Java or 64-bit Eclipse for 64-bit Java.
- You must also have the new Cypress-provided Eclipse feature (com.cypress.psoccreatorimport) installed, which provides the following functionality:
  - Speeds the creation of an Eclipse project associated with a PSoC Creator design with a new PSoC Creator project type and pre-populating tool chain build options.
  - Provides project resource synchronization between PSoC Creator and Eclipse. Changes made in the set of files that constitute a project in PSoC Creator are reflected in subsequent builds in Eclipse.

Download this Cypress-provided Eclipse feature from www.cypress.com/go/creator/eclipseimportdownload, and install the downloaded zip file as follows:

- In Eclipse, select **Help > Install New Software…** to open the Install dialog.



- Check the Contact all update sites during install to find required software check box in the Details section of this dialog.
- Select the **Add…** button at the top of the dialog to open the Add Repository dialog.



- Click **Archive…** and browse to the location where you saved the Cypress-provided Eclipse plugin zip file that you downloaded above. Select the file and click **Open**; then click **OK**.

- Back on the Install dialog, select the check box next to "PSoC Creator Import Feature" and then click **Next >**.
- The next wizard page summarizes the features you are about to install. Click **Next >** and read and accept the license for this feature.
- Click **Finish** to install the plugin. There might be a warning about the software not being signed; click **OK**.
- Restart Eclipse when prompted.

- In order to use the Segger J-Link debug probe, you must download the Segger J-Link toolset from http://www.segger.com/jlink-software.html. You will need release 5.12 or later, which is required for PSoC 4000S flash support.

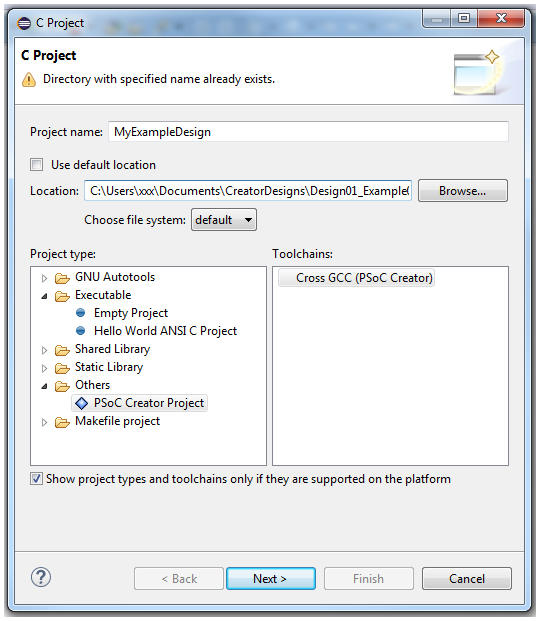
Import into Eclipse

Create New Eclipse CDT Firmware Application Project

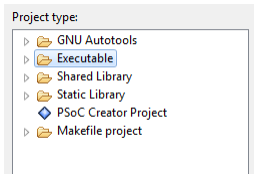
Create an Eclipse CDT executable project for your firmware. This must be created as a project rooted not in the Eclipse workspace, but instead sharing the source code folder of your PSoC Creator design.

Note When you start Eclipse, you will create a new workspace or select an existing one. You must choose one that does not include the folder where your PSoC Creator design is located. This is an Eclipse limitation.

1. Create a project via the **File > New > C Project** menu entry.
2. On the wizard's C Project page:
   - Type a **Project name**.
   - Unselect the **Use default location** check box, browse to and select your PSoC Creator project's top-level folder (the one ending in .cydsn). This allows PSoC Creator and Eclipse to share the same source files.
   - Under **Project Type > Others**, select "PSoC Creator Project."
   - Under **Toolchains**, select "Cross GCC (for PSoC Creator)."



Note In Eclipse releases earlier than the Luna release, the **Others** category does not exist. The "PSoC Creator Project" entry can be found in the Project Type list as shown.



2. Click **Next** to move to the Select Configurations wizard page. No changes are needed here. Click **Next** again.
3. On the **Cross GCC Command** wizard page:
   - Prefix will have a default value of: arm-none-eabi-
   - Set the path by navigating to the bin folder of your compiler's install directory, where you can find *arm-none-eabi-gcc.exe*. This is typically found using the following path:

   <Arm_tools_install_path>/bin

   (Look for *arm-none-eabi-gcc.exe* and not *gcc.exe*.)

   Note You can use the Arm GCC installation included in PSoC Creator, if you have PSoC Creator installed.
   - In PSoC Creator 3.1 and earlier, this is found at:

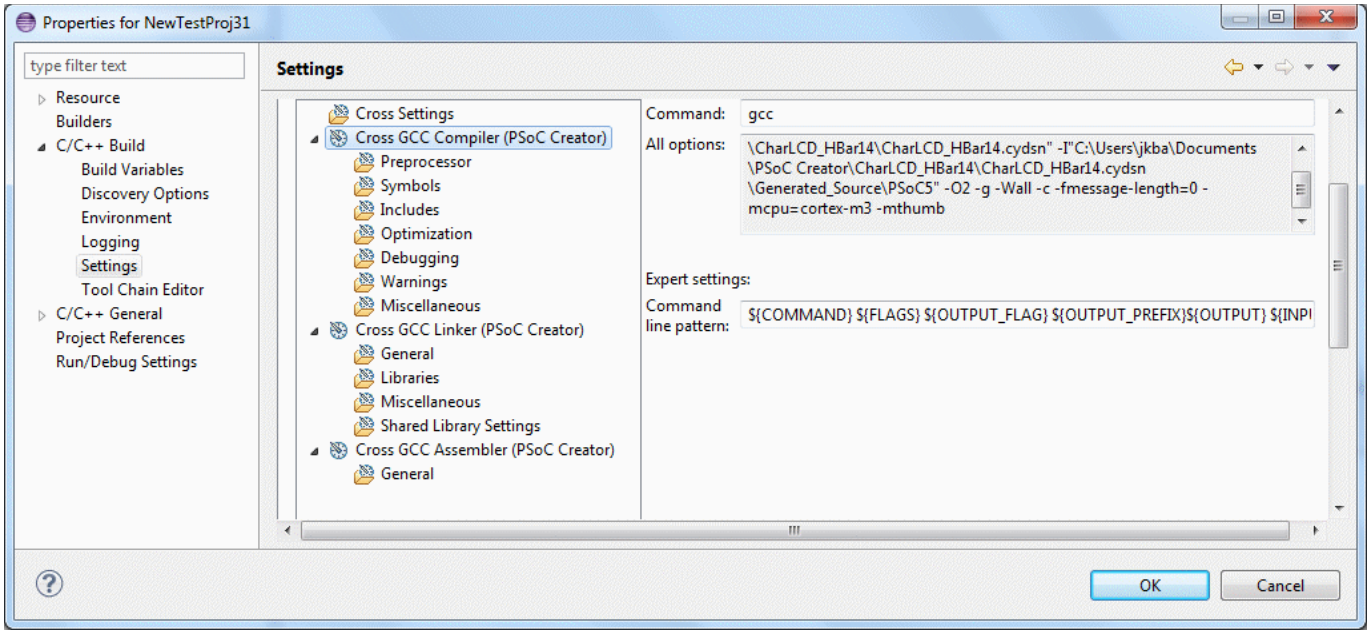
   <PSoC_Creator_install_path>/import/gnu_cs/arm/<version>/bin
   - In PSoC Creator 3.2 and later, this is found at:

   <PSoC_Creator_install_path>/import/gnu/arm/<version>/bin

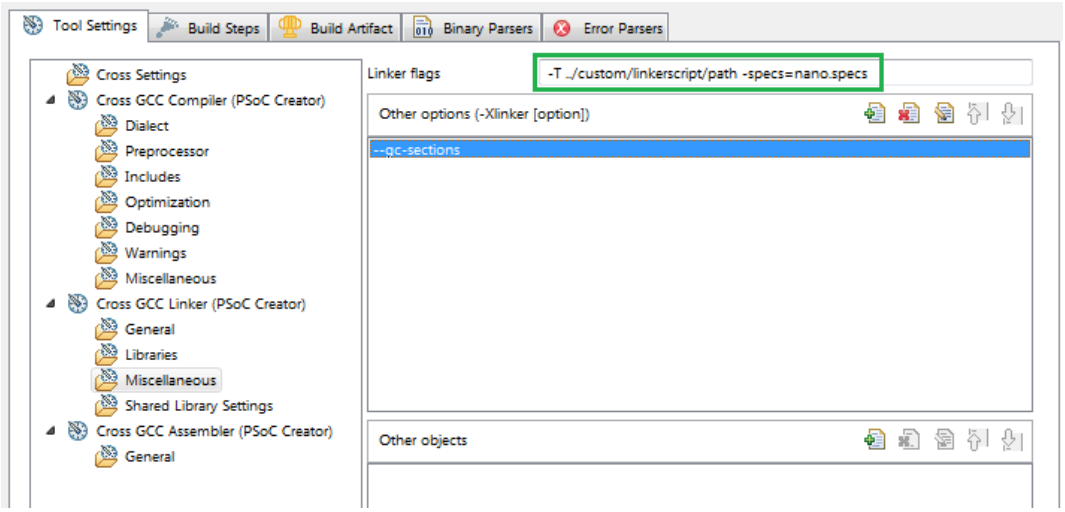
The newly created PSoC Creator project in Eclipse includes toolchain settings, accessible in the Project Explorer pane.

1. Right-click the project name and select **Properties > C/C++ Build > Settings**.
2. Then select the **Tool Settings** tab.

These settings are driven by the project's PSoC device. Certain options will be different depending on whether the project is for a PSoC 4 or PSoC 5LP device. (The pre-populated option settings are listed in the Project Build Settings section at the end of this document in case you decide to change them and later want to restore these values. These values are set by default; you typically do not need to modify them for your project.)



Note If you are using custom toolchain options, add those options manually to the tool setting in Eclipse. For example, if you are using custom linker scripts in the project, modify the **Linker Flags** option as shown in the following image:



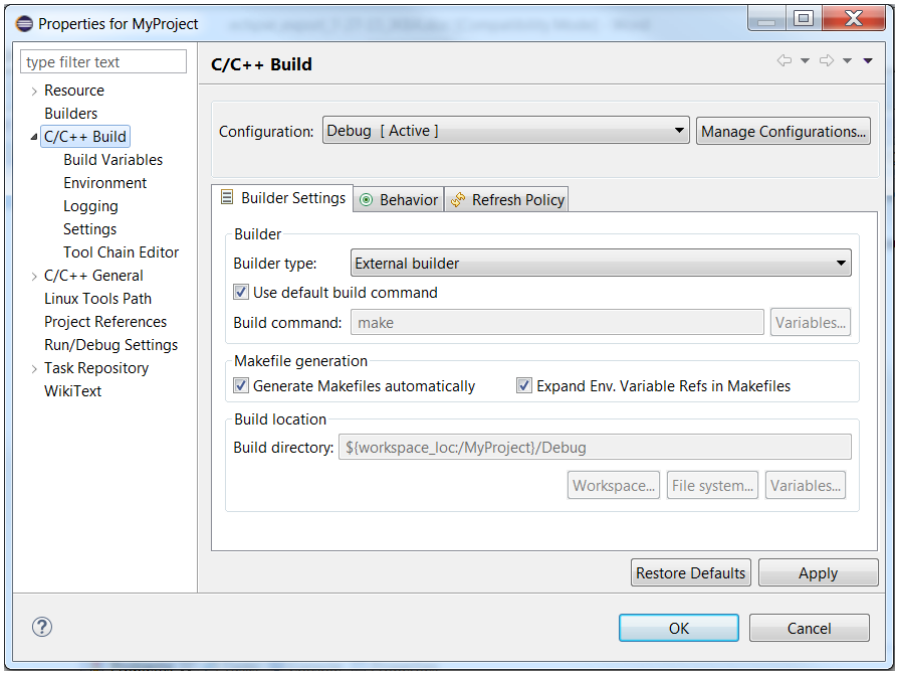
Building with Cygwin/make Installed

As of this feature version v2.2, designs are imported with the project builder set to the Eclipse Internal Builder. This means that you do not need the 'make' utility installed to build your design in Eclipse.

If you wish to use the 'make' utility, you will need to use the Eclipse External Builder. In this case, the Eclipse CDT Builder assumes that the 'make' utility is in your PATH, typically obtained through a cygwin installation on your machine. If using the Eclipse external builder, you'll need to have something like "C:\cygwin\bin" or C:\cygwin64\bin" in your Windows PATH environment variable or in the Eclipse environment settings.

Without 'make', you need to adjust your project build settings to use the Eclipse internal builder instead.

1. Right-click the project name and select **Properties > C/C++ Build**.
2. Select the **Builder Settings** tab.
3. Change the **Builder type** value to External builder. Click the **OK** button.



Note For more complex designs, Make files may be generated with absolute paths for files outside of the current project. This can expose a bug in GNU Make versions later than 3.8.0 and earlier than 4.0. You may see an error like: "Multiple targets. Stop.". The solution is to use the Eclipse Internal Builder, or to change your GNU Make version to one that does not contain this bug.

Design Iteration between PSoC Creator and Eclipse

Users are directed to create their Eclipse project using their PSoC Creator design folder. Users simply need to rebuild their design in PSoC Creator and run the IDE Export wizard again. The Cypress-provided plugin detects the build update to the PSoC Creator project and makes the corresponding change in the visible Eclipse project files. A corresponding check and update is performed when Eclipse is started as well.

Note that this synchronization support is one-way: changes in PSoC Creator can be seen in Eclipse, but file additions and deletions made in Eclipse will not be seen by PSoC Creator. This allows changes in the hardware configuration made upstream in PSoC Creator to be seen downstream in Eclipse for continued firmware development.

Change in Project Build Settings

Project build settings are not exported from PSoC Creator to Eclipse; any custom build settings that you add or change in PSoC Creator must be similarly added/changed in Eclipse, following the export step. As a starting point, the default build settings in PSoC Creator are also set as the default build settings on initial export of a project to Eclipse. You can then change these build settings as needed inside Eclipse. These default build settings are listed below for reference:

Under Cross GCC Compiler:

- Under Includes, these project include paths are provided as default values:
  - ".." (i.e., the project design folder)
  - "../Generated_Source/${GEN_ARCH}"
- Under Optimization, this flag must be set:
  - -ffunction-sections
- Under Miscellaneous, Other flags, these Arm-related flags default values are provided via the ${PROC_FLAGS} variable setting:
  - For PSoC 5LP: -mcpu=cortex-m3 -mthumb
  - For PSoC 4: -mcpu=cortex-m0 -mthumb
  - For PSoC 4000S/PSoC 4100S/PSoC 4100S Plus/PSoC Analog Coprocessor: -mcpu=cortex-m0plus -mthumb
- Under Miscellaneous, Other flags, these flags must be set:
  - -c -fmessage-length=0

Under Cross GCC Assembler:

- Under Includes, these project include paths are provided as default values:
  - ".." (i.e., the project design folder)
  - "../Generated_Source/${GEN_ARCH}"
- Under General, Other flags, these Arm-related flags default values are provided via the ${PROC_FLAGS} variable setting:
  - For PSoC 5LP: -mcpu=cortex-m3 -mthumb
  - For PSoC 4: -mcpu=cortex-m0 -mthumb
  - For PSoC 4000S/PSoC 4100S/PSoC 4100S Plus/PSoC Analog Coprocessor: -mcpu=cortex-m0plus -mthumb

Under Cross GCC Linker:

- Under Libraries, Library Search Path, this library path is provided:
  - -L "../Generated_Source/${GEN_ARCH}"
- Under Miscellaneous, Linker flags, these flag values are provided:
  - -T ../Generated_Source/${GEN_ARCH}/${LINKER_FILE} -specs=nano.specs
- Under Miscellaneous, Linker flags, these Arm-related flags default values are provided via the ${PROC_LINK_FLAGS} variable setting:
  - For PSoC 5LP: -mcpu=cortex-m3 -mthumb -mfix-cortex-m3-ldrd
  - For PSoC 4: -mcpu=cortex-m0 -mthumb
  - For PSoC 4000S/PSoC 4100S/PSoC 4100S Plus/PSoC Analog Coprocessor: -mcpu=cortex-m0plus -mthumb
- Under Miscellaneous, Linker flags, this default value is provided: -Xlinker --gc-sections
- Under Other Objects, the following object path must appear:
  - "${EXPORT_FOLDER_PATH}/${TOOL_NAME}/CyComponentLibrary.a"
- The link step command line is structured so that "-Wl,--start-group" and "-Wl,--end-group" options surround all input files and libraries, providing multiple linker passes to resolve cyclic references between libraries, if needed.

On the Build Steps tab:

The following default settings are provided:

- Pre-build step should read:
  "../${EXPORT_FOLDER_PATH}/prebuild.bat"
- Post-build step should read:
  "../${EXPORT_FOLDER_PATH}/postbuild.bat"
"${POSTBUILD_HEXFILE_PATH}/${ConfigName}/${ProjName}"

Build Customization

Project Customization

- Compiler optimization settings – Other optimizations that might be useful in reducing your generated flash image size include:

Under Cross GCC Compiler:

  - Under **Optimizations**, set Optimization level to Optimize for size (-Os)

- Linker script changes – A default GCC linker script is located at:
  - For PSoC 5LP: Generated_Source/PSoC5/cm3gcc.ld
  - For PSoC 4: Generated_Source/PSoC4/cm0gcc.ld
  - For PSoC 4000S/PSoC 4100S/PSoC 4100S Plus/PSoC Analog Coprocessor: Generated_Source/PSoC4/cm0plusgcc.ld

If you need to customize the link step for your executable image, do the following:

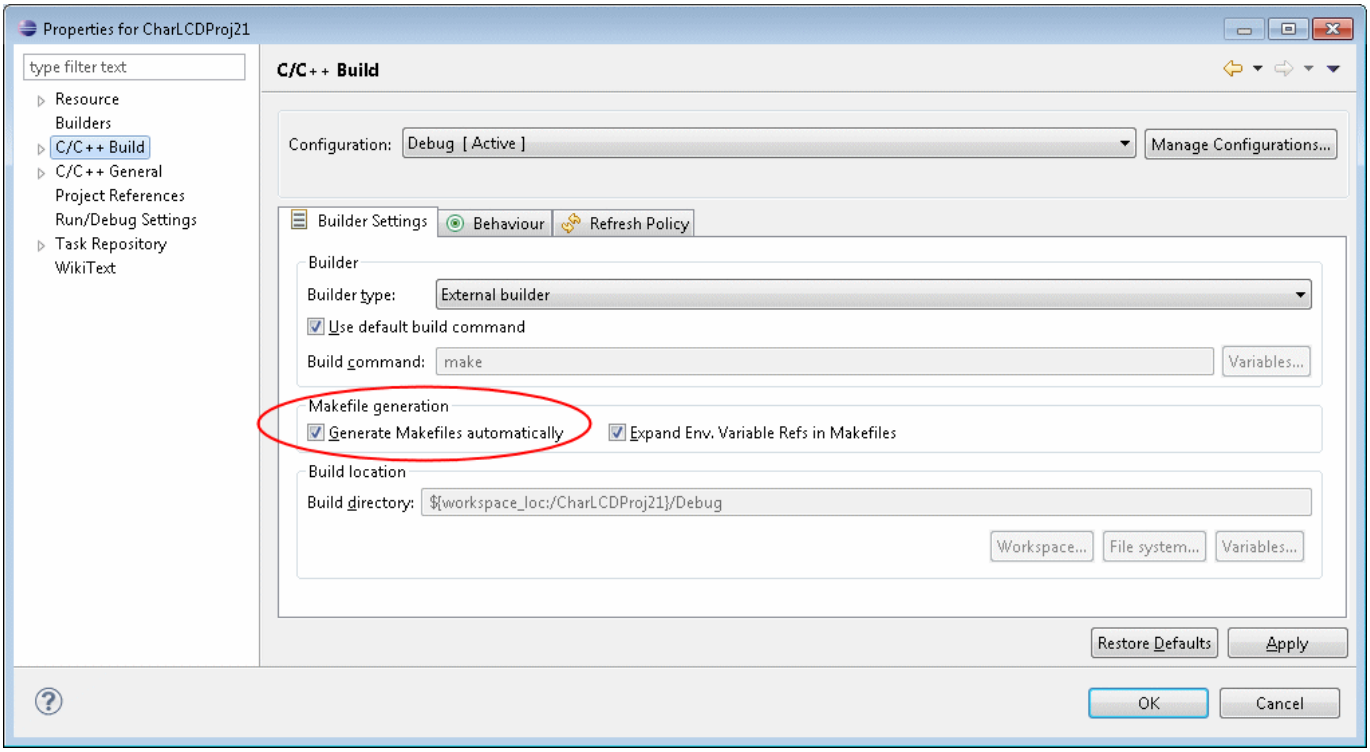
1. Copy the file from the above location to the top level folder of your Eclipse project. This will prevent any subsequent builds in PSoC Creator from overwriting your changes to this file.
2. Edit your new copy of the file as needed.
3. Change the Build Settings linker flags (under **Cross GCC Linker, Miscellaneous, Linker flags**) to be:
   - For PSoC 5LP: -T ../cm3gcc.ld
   - For PSoC 4: -T ../cm0gcc.ld
   - For PSoC 4000S/PSoC 4100S/PSoC 4100S Plus/PSoC Analog Coprocessor: -T ../cm0plusgcc.ld

Build Customization

If your design requires specialized steps in order to build to completion (beyond the standard pre-build/assemble/ compile/link/post-build stages of the standard Eclipse CDT Managed Build System [MBS]), you may want to consider managing your own build for the project. Perhaps the easiest way to do this is to turn off the MBS automatic generation of Makefiles with every project build. You can then customize the already generated Makefiles to complete your project build as needed.

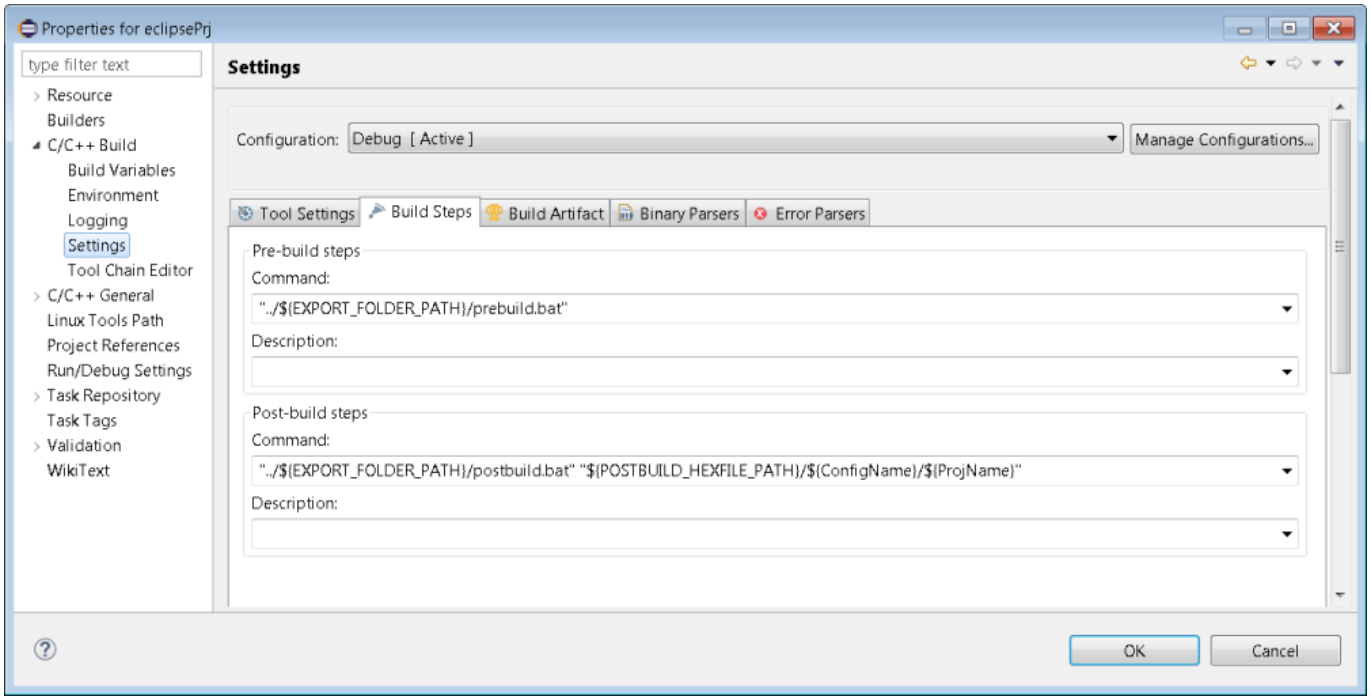
To turn off the automatic generation of Makefiles for your project, do the following:

- In the Eclipse Project Explorer, right-click on your project name and select Properties…
- Select C/C++ Build in the left hand pane
- Uncheck the Generate Makefiles automatically check box



User Commands

PSoC Creator provides the ability to insert pre-build and post-build user commands during project builds. However, these steps are not included in the files generated for integration into third-party IDEs. For Eclipse, you can edit your project's settings (under the project's **Properties > C/C++ Build > Settings**) on the Build Steps tab. PSoC Creator provides a call to our generated pre-build and post-build scripts where needed; you can add your own commands, separating all commands with semicolons.



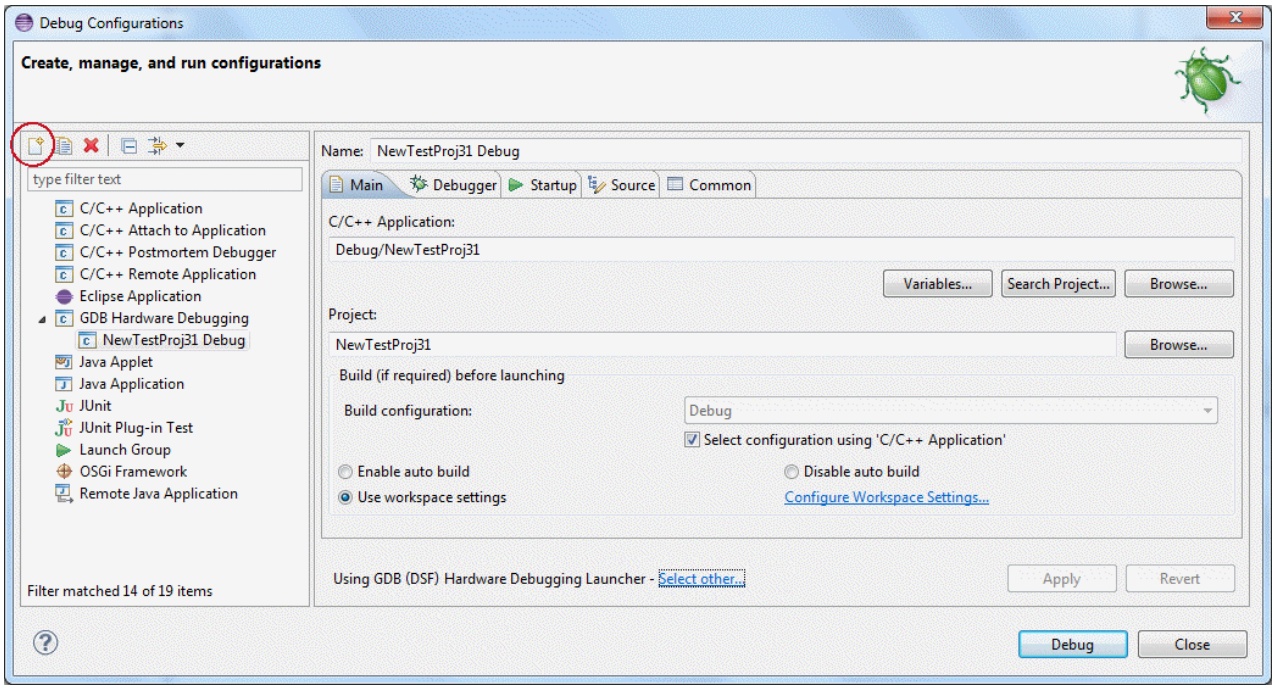
Flashing and Debugging in Eclipse

In order to use a J-Link or J-Trace debug probe, you will first need to download and install Segger's software and documentation pack from their web site (https://www.segger.com/jlink-software.html). This package provides USB drivers for their probes, several utilities, and a GDB server, which connects the Eclipse debugger to your Cypress hardware.

Debug Flow within Eclipse

Follow these steps to flash and debug an imported PSoC Creator project in Eclipse/CDT using a Segger J-Link probe.

1. Provide power to your PSoC hardware, then connect a J-Link debug probe to your PSoC hardware and host machine. Run J-Link Commander.exe from the All Programs->SEGGER->J-Link Arm menu. This initializes the J-Link probe and only needs to be done once following each J-Link probe connection. (Without this step, the J-Link GDB Server cannot successfully connect to the target.) Verify the following before exiting the application:
    - Target device is set to your PSoC device.
    - Target interface is set to JTAG (PSoC 5LP only), or SWD (PSoC 4 or PSoC 5LP)
    - Type 'exit' at the tool's command prompt to exit the application.
2. In Eclipse, create a new Debug Launch configuration for your design. Use the menu selection **Run > Debug Configurations…**
3. Create a new debug launch configuration by selecting the "GDB Hardware Debugging" category and clicking the **New** button (circled in the following figure).



4. Enter the debug launch configuration settings as follows:

   Main tab:

   - Provide path to the design's executable file produced by the project build. Navigate to the project's Release or Debug folder to find it, or simply click the **Search Project…** button and select the executable.
   - The debugger launcher should be set to "GDB (DSF) Hardware Debugging Launcher."

   Note However, for the Eclipse Kepler release, it is suggested that users select the "Legacy GDB Hardware Debugging Launcher." (This is found at the bottom of the **Main** tab, change using the "Select other..." link.)

   Debugger tab:

   - Set the path to the GDB command by browsing to the GDB executable to use. Typically, this will be named arm-none-eabi-gdb.exe, located in the same folder as the "Cross compiler path" setting provided when importing into Eclipse.
   - "Use remote target" is checked
   - JTag device is set to "Generic TCP/IP"
   - Host name is "localhost" and port is "2331" (the default J-Link GDB server port).

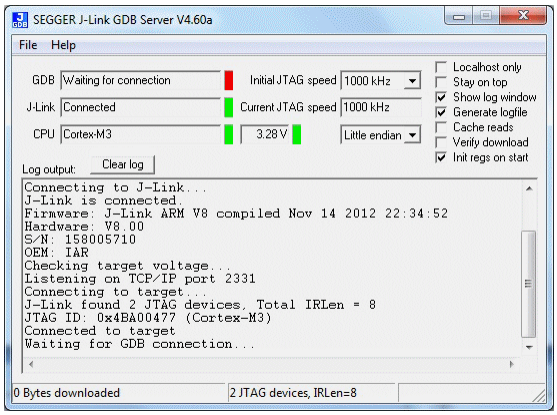
   Startup tab:

   - Reset and Delay set to 3 seconds
   - Halt is checked
   - Enter "monitor reset" in the Init command field:
   - Load Image and Load Symbols sections must have "Use project binary" selected.
   - Runtime options: "Resume" box is checked
   - Press Apply and Close to save this debug launch configuration for later use.

5. Launch the Segger J-Link GDB server by running "J-Link GDB Server" via All Programs->SEGGER->J-Link Arm menu. Verify the following are set:

   - Connection to J-Link is set to USB
   - Target interface is set to JTAG or SWD as needed.
   - Target device is set to the Cypress PSoC device you are using. Press the "…" button to browse the supported devices. Select Cypress in the Manufacturers pull-down to list only Cypress devices. Select your device family and press OK.
   - OK the main dialog window.

The server should connect to the target at this point and indicate it is waiting for a GDB connection, as shown in figure below.



6. In Eclipse, set any needed breakpoints in your code. Run the Debug launch configuration you created in step 2 above by using the **Run > Debug Configuration…** menu selection to locate your debug launch configuration from the ones under the GDB Hardware Debugging heading. Click the **Debug** button to start the debug session. The GDB Server will display additional output when the Eclipse GDB session begins.
7. When your breakpoint is encountered, Eclipse will change to its debug perspective and halt. Normal Eclipse debug functionality (breakpoint manipulation, examining/changing variables and memory, etc) is available at this point.

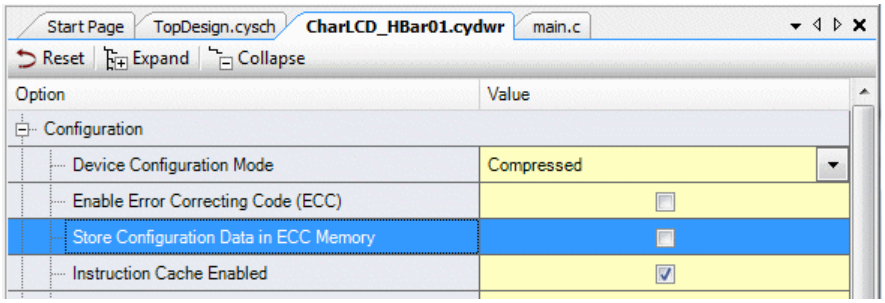
Note If you experience problems with the PC not tracking as you would expect while stepping through code, you may be able to correct this by turning off compiler optimization in your code while debugging, as follows:

1. Right-click the project name and select Properties > C/C++ Build.
2. Select Settings.
3. Select Cross GCC Compiler (PSoC Creator).
4. Select Optimization.
5. Change the optimization level to "None".
6. Rebuild your design and start debugging again.

Flashing PSoC 5LP Designs using the J-Link Probe

If you are exporting your design to Eclipse for a PSoC 5LP design, it is important that you understand the following limitations. These limitations will be removed in a future release of the Segger J-Link flashloader for PSoC 5LP devices.

- Before building and exporting a PSoC 5LP based design to Eclipse, go to your project's Design Wide Resources System page and uncheck the "Store Configuration Data in ECC" check box. Storing configuration data in ECC memory is the default setting for PSoC Creator projects, but must be changed for designs exported to Eclipse in order for them to execute properly once flashed.



- Your design cannot use EEPROM memory with any initial values set from the PSoC Creator EEPROM Editor. The Segger-provided PSoC 5LP flashloader does not yet program this memory space.
- Your design cannot use ECC memory, as the Segger-provided PSoC 5LP flashloader does not yet program this memory space. Your project's DWR System settings should have the "Enable Error Correcting Code (ECC)" box unchecked, as shown above.
- The Segger-provided PSoC 5LP flashloader does not yet program the NVL memory space. This means the following settings, if changed in your design in PSoC Creator, must be programmed by building and downloading your design first in PsoC Creator with a Cypress MiniProg3 probe. You can then export your design to Eclipse and program via a J-Link probe:
  - DWR System setting for Enable Fast IMO During Startup
  - DWR System setting for Enable Device Protection
  - DWR System setting for Debug Select
  - DWR System setting for Use Optional XRES
  - Initial state settings on pins

Eclipse Bootloader Support

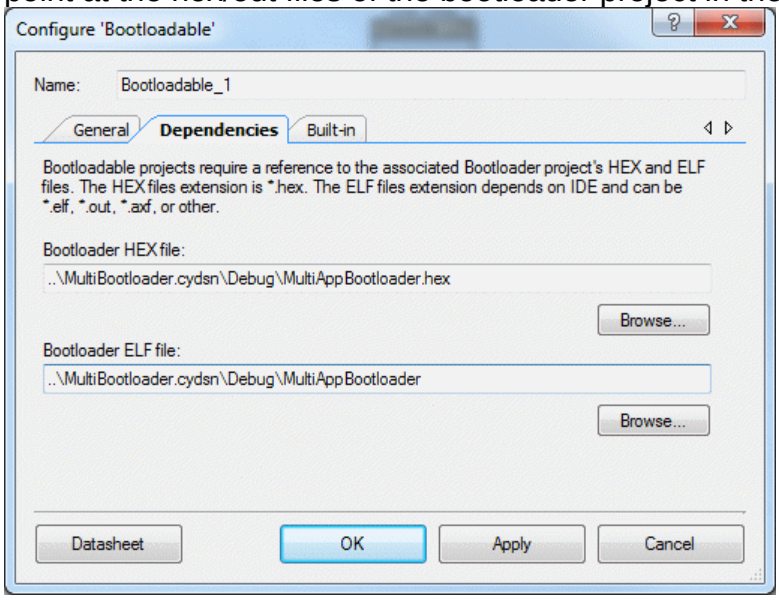
Note These steps apply to all Bootloader/Bootloadable type projects, including Launcher and Combination projects.

Note For PSoC 6 devices, refer to the Bootloader SDK guide and related collateral.

Bootloader/Bootloadable Project Export

This section describes the steps necessary to export a single application Bootloader/Bootloadable project for Eclipse IDE.

1. Export the bootloader project as usual. See Exporting a Design to Eclipse IDE.
2. Import and build bootloader project in Eclipse.
3. In PSoC Creator, configure the Bootloadable Component for the bootloadable project to point at the hex/out files of the bootloader project in the Eclipse output directory.

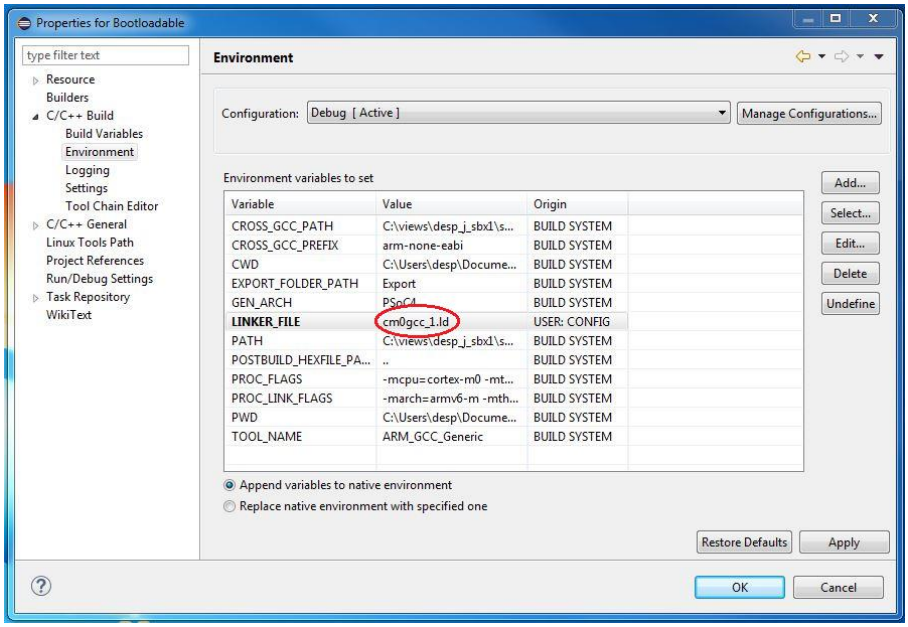


4. Export bootloadable project.

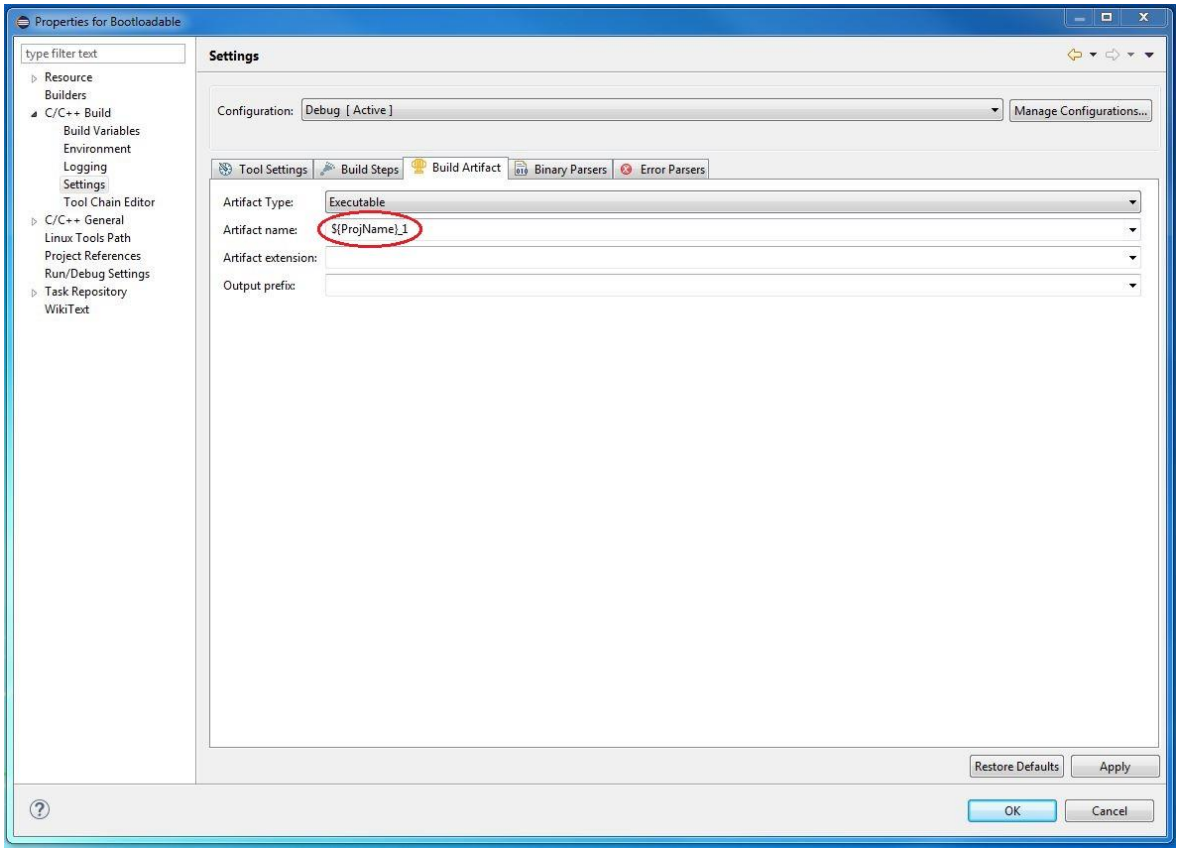
Multi-Application Bootloader/Bootloadable Support

This section describes how to import a PSoC Creator multi-application bootloader project for use in the Eclipse IDE.

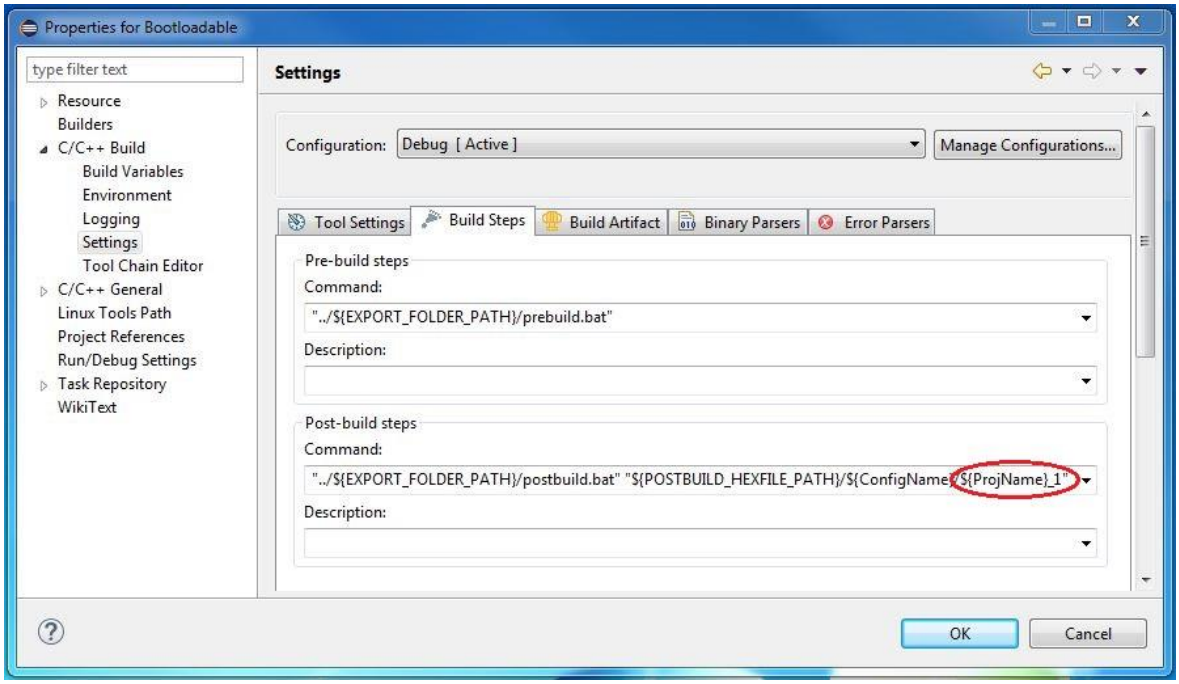
1. Export Bootloader and Bootloadable projects as usual. See Bootloader/Bootloadable Project Export. Import them into Eclipse as separate projects.
2. In Eclipse. modify the Eclipse Bootloadable project's settings as follows:
    a. Update the project's LINKER_FILE setting by appending "_1" before the ".ld" extension.



b. Update the Project's Artifact name by appending "_1".



c. Update Project Post-build steps by appending "_1".

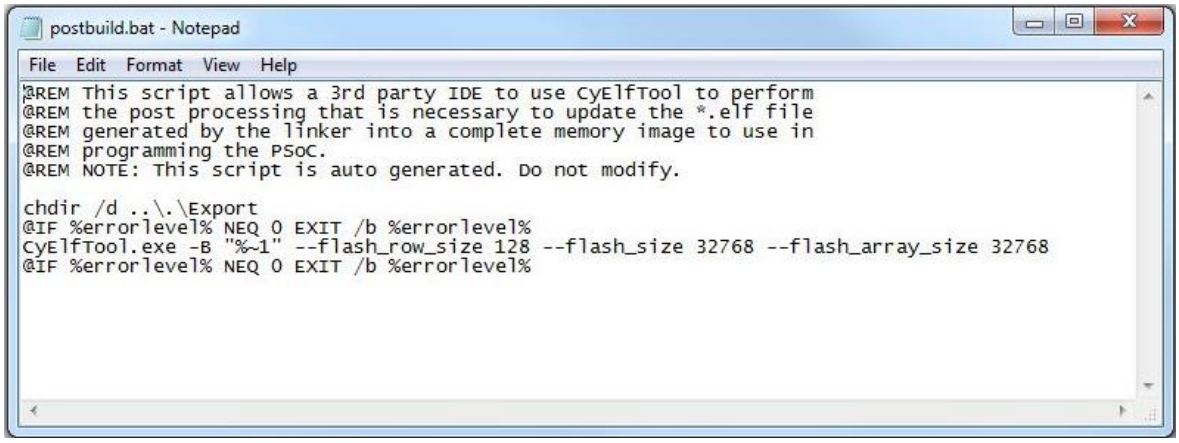


d. Build Project

e. Repeat step 2.a to step 2.d, but replace "_1" with "_2".

3. Then, use *CyElfTool.exe* to create a combination hex file that has the bootloader and both bootloadable images in it.

   a. Open the *postbuild.bat* file found in the \Export directory.



   b. Note the `-flash_row_size`, `flash_array_size`, `flash_size` command line arguments. Note the `-ee_array` and `-ee_row_size` arguments as well, if they exist. You will use these values in the command line shown in step 3.d. below.

   c. In a Windows command line, cd into the .cydsn directory of your bootloadable project.

   d. Run the *CyElfTool.exe* tool using a command similar to the following:

```
Export\CyElfTool.exe -M Debug\{ProjectName}_1 Debug\{ProjectName}_2
Debug\{ProjectName}.hex --flash_row_size 128 --flash_size 32768 --
flash_array_size 32768
```

Note Flash and EE arguments are device-dependent and can be copied from the bootloadable's *postbuild.bat* file.

Note If you use the release configuration, make sure to change all instances of Debug in the above command line to Release. The above *CyElfTool.exe* command will generate a combination hex file that contains the bootloader image and both bootloadable images. It will have name *{ProjectName}.hex*.

4. You should now have two bootloadable .cyacd files, a bootloader elf file, and a combination hex file in the three Eclipse project Debug directories listed above. You can now program and debug your bootloader, either bootloadable or combination files as you wish.

© Cypress Semiconductor Corporation, 2017-2019. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical Components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical Component is any Component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.

