Application Note

Appropriate CCS Configurations for the UCD3138 Family of Digital Power Controllers

Bliss Zhou

ABSTRACT

This document was translated from a simplified Chinese source. (ZHCAFM3)

The UCD3138, a family of TI's high-performance digital power controllers, requires appropriate CCS configurations for operation. When the user renames a UCD3138 project, upgrades CCS, or uses another computer, unexpected issues may occur even with a previously working Firmware project. These issues are generally caused by inappropriate CCS configurations. This application note lists these issues and provides solutions.

Table of Contents

1 List of Issues Caused by Inappropriate CCS Configurations

- Build failure after project renaming
- Build failure after upgrading CCS or porting the project to another computer
- Device lockup, preventing a new image from being re-flashed to the UCD
- No generation of the .x0 target file after build completion
- No generation of the .pp file after build completion

2 Avoiding Issues with Appropriate Configurations

2.1 Avoiding Build Failures After Project Renaming

Typically, users do not use the TI EVM code project name directly as their CCS project name. They generally require a custom project name. However, they may encounter some build errors when renaming CCS projects. Figure 2-1 shows common error messages.

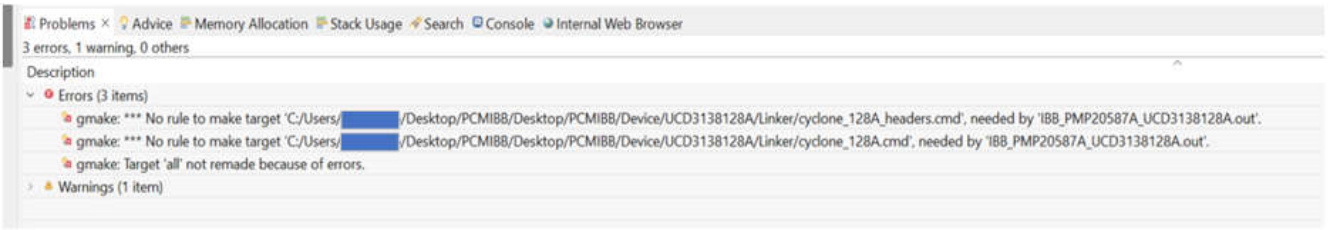

Figure 2-1. "No rule to make target" Error

Renaming the project causes the *Linked Resources* to automatically change the default path to "*copy_PARENT*". However, this is a path that does not exist on the user's local computer. An *Invalid Location* error appears under *Project Properties -> Resource -> Linked Resources -> Linked Resources,* as shown in Figure 2-2.

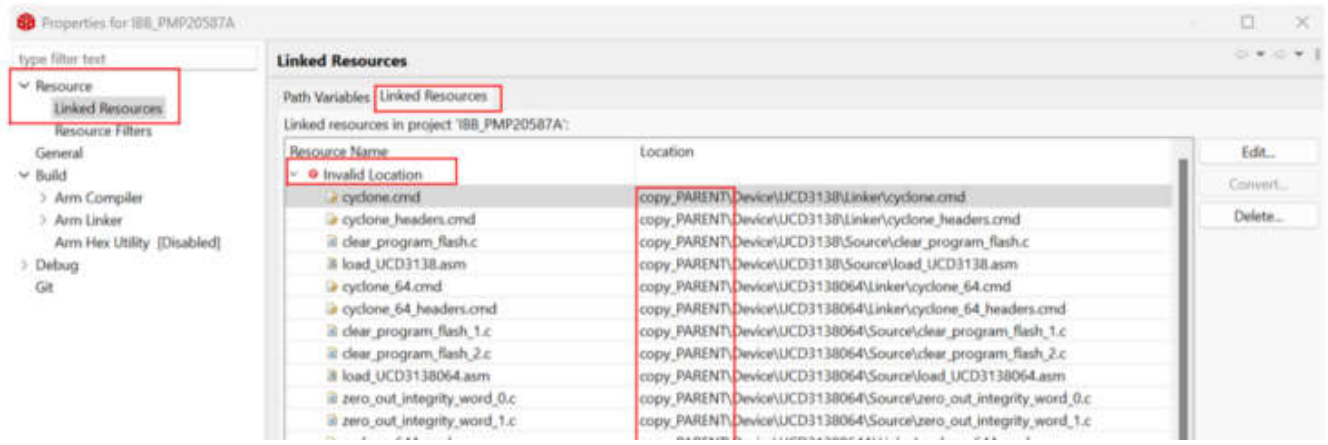

Figure 2-2. Invalid Location for Linked Resources

To resolve this error, users need to replace all "*copy_PARENT*\" with "*PROJECT_LOC*\..\..\". A correct path setting is shown in Figure 2-3.

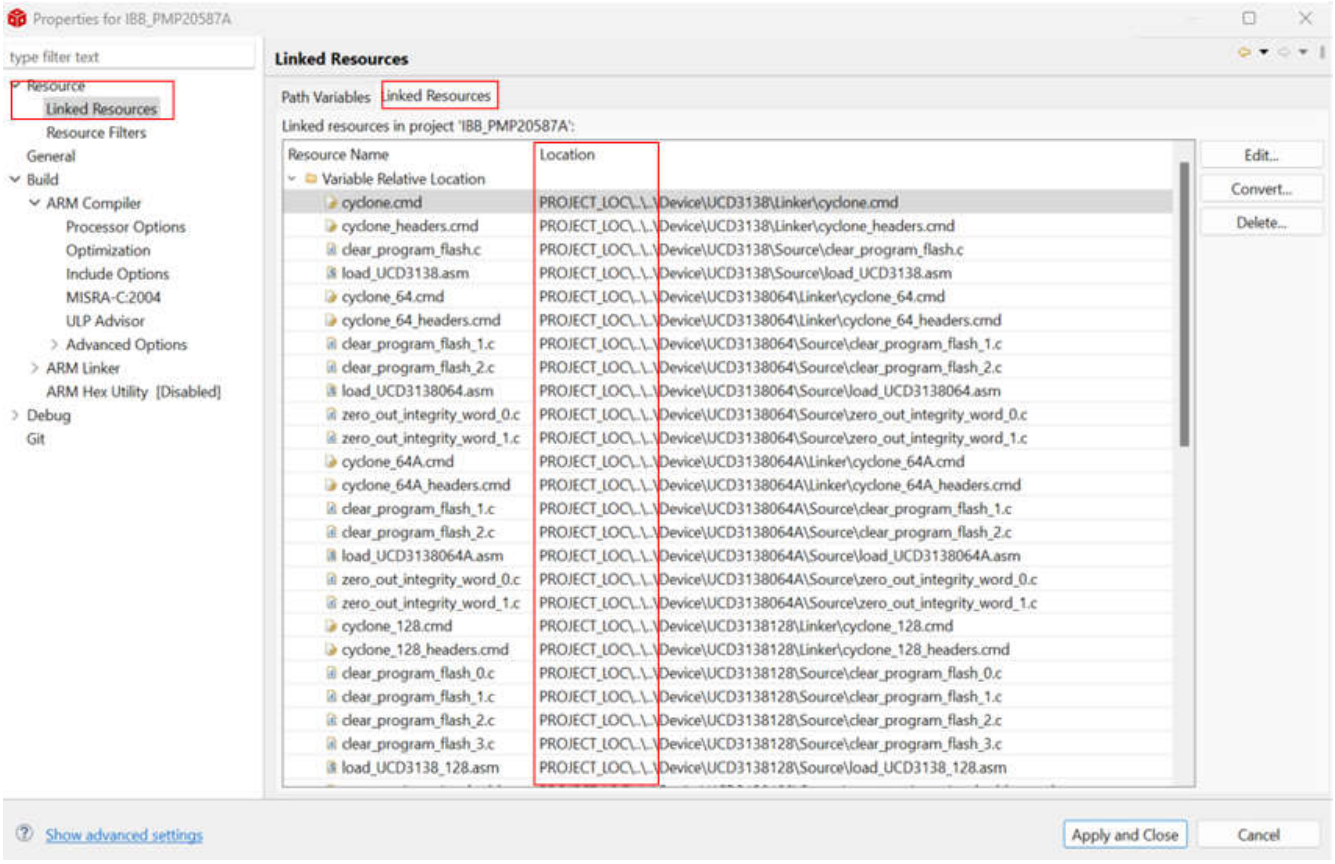

Figure 2-3. Valid Location for Linked Resources

2.2 Avoiding Build Failures Caused by Upgrading CCS or Porting a Project to Another Computer

A project that was previously built successfully may experience build failures caused by upgrading CCS or setting up a new computer. Common error prompts include "*#1559: Interrupt handlers must be compiled in ARM mode*", which refers to files *"standard_interrupt.c", "software_interrupt.c"*, and *"interrupts.c"*, as shown in Figure 2-4. The root cause of this error is that interrupt service functions must be built in ARM mode instead of Thumb mode. During CCS upgrading, the original configuration for the ARM mode may not be preserved and is automatically changed to the configuration for the Thumb mode, resulting in this issue (Note that .C files are usually built in Thumb mode to reduce code size).

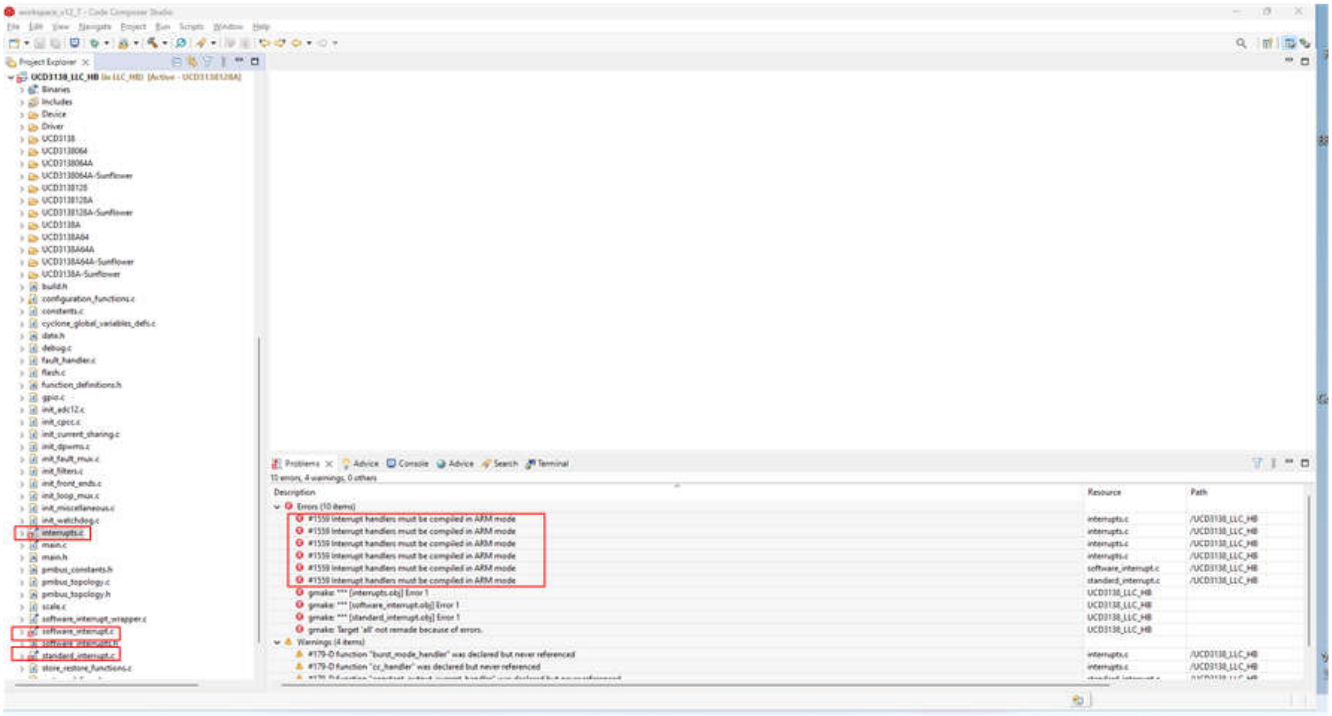

Figure 2-4. Interrupt Handlers Must Be Compiled in ARM Mode

Two solutions are typically available:

1. Configure files *"standard_interrupt.c", "software_interrupt.c", and "interrupts.c"* to be built in ARM mode. For example, to configure the entire *"software_interrupt.c"* file to be built in ARM mode, as shown in Figure 2-5, follow the steps below to complete the configuration. Open the *Project Explorer pane, right-click "standard_interrupt.c", go to Properties -> Build -> ARM Compiler -> Processor Options*, and select "32".

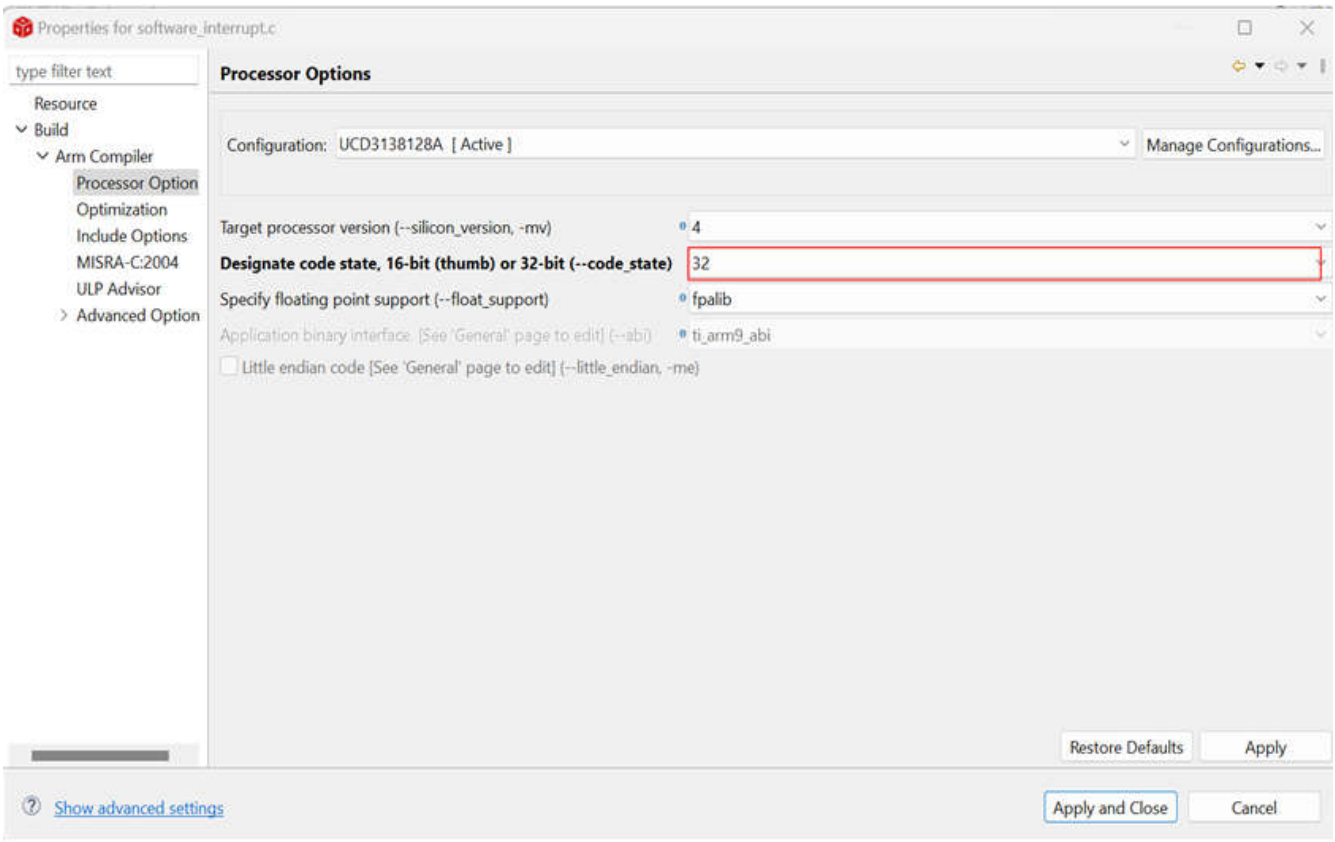

Figure 2-5. Setting the Designate Code State to ARM 32-Bit Mode for Interrupt Handler Compilation

2. Place *"#pragma CODE_STATE (function name,32)"* before the interrupt function to build only this function in ARM mode. For example, in the file *"standard_interrupt.c"*, "*void standard_interrupt(void)*" must be built in ARM mode. Use "*#pragma CODE_STATE (standard_interrupt, 32)*" to build "*standard_interrupt(void)*" in ARM mode, as shown in Snippet 1.

```
#pragma INTERRUPT (standard_interrupt, IRQ) #pragma
          CODE_STATE (standard_interrupt, 32) void standard_interrupt(void) {
          handle_standard_interrupt_global_tasks(); switch (supply_state) { case STATE_IDLE:
          handle_idle_state(); break; case STATE_CHECK_RAMP_UP: handle_check_ramp_up(); break;
          case STATE_HIGH_CURRENT_DURING_RAMP_UP: handle_high_current_during_ramp_up();
          break;
```

Snippet 1: Enforce "standard_interrupt(void)" to Be Built in ARM Mode

2.3 Avoiding Device Lockup

Similarly, users may encounter device lockup after upgrading CCS, which prevents the image from being re-flashed. An error prompt, "*Device does not appear to be in ROM mode after ENABLE_ROM_command*", is shown in Figure 2-6 .

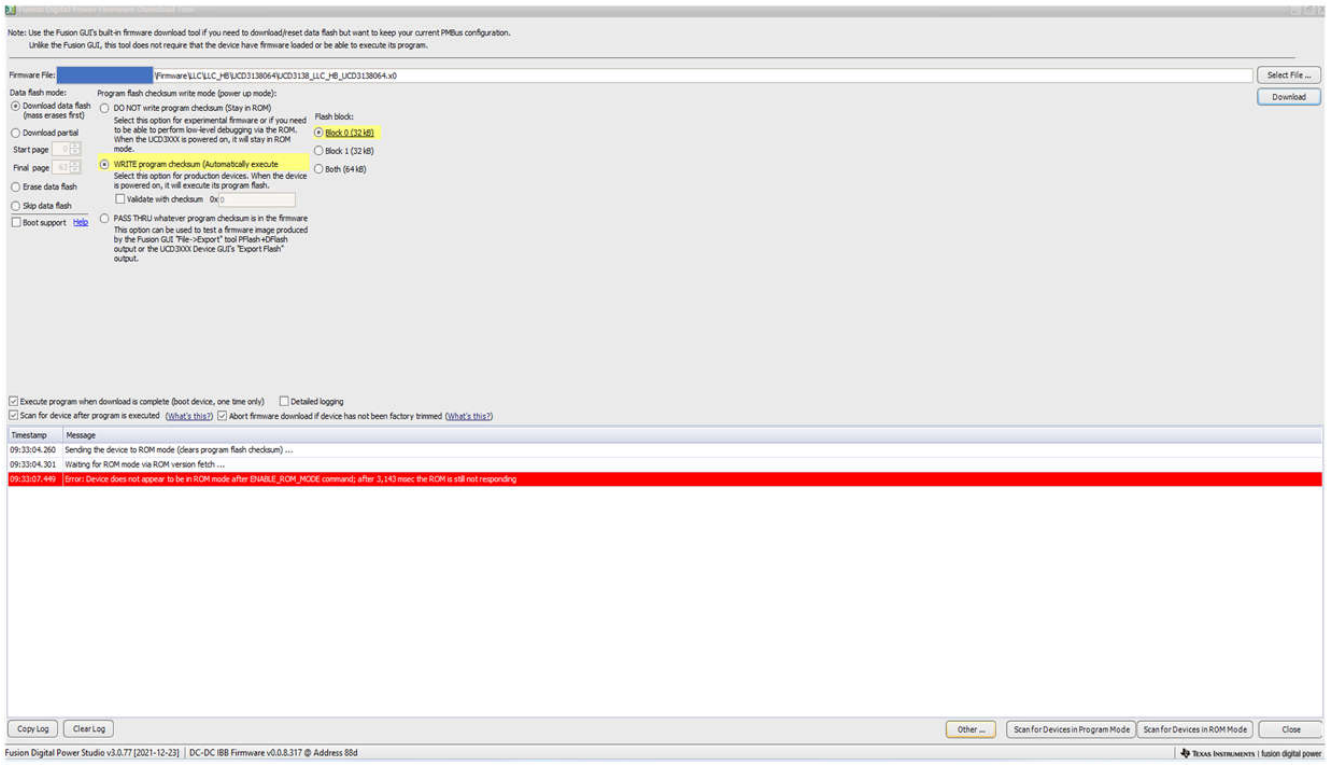

Figure 2-6. Device Lockup

The application note, Lockup Avoidance for UCD3138 Devices, provides a detailed analysis of the causes of and solutions for device lockup. In addition to the causes analyzed in this application note, two additional potential causes associated with the compiler version are introduced here.

- The first one is that the user is using an incompatible version of the compiler. The EVM codes for UCD are all developed using compiler versions TI V5.2.2-V5.2.5. Some EVM codes do not work well with other compiler versions, such as V5.2.9. If the user encounters a chip lockup, first check whether a compatible compiler version is being used. If not, a compatible compiler must to be installed. The installation steps are as follows:

  Step1: Go to *Help -> Install Code Generation Compiler Tools -> TI Compiler Updates*.

  Step2: Deselect *"Show only the latest versions of available software", select "ARM Compiler Tools 5.2.5", and click Next*, as shown in Figure 2-7.

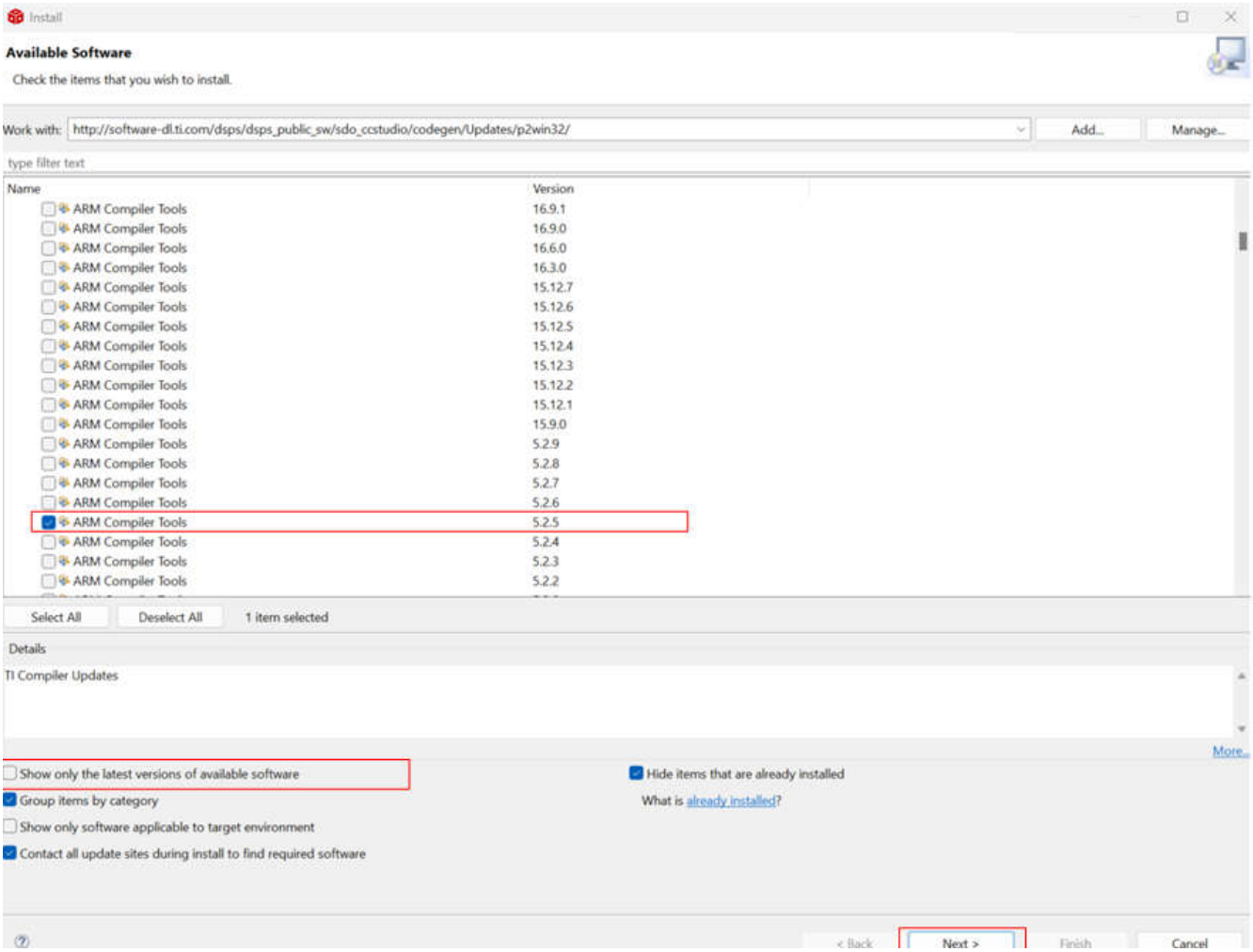

Figure 2-7. Install a Compatible Compiler Version

Step3: Restart the computer after the installation is complete.

Step4: Switch the compiler to v5.2.5, as shown in Figure 2-8.

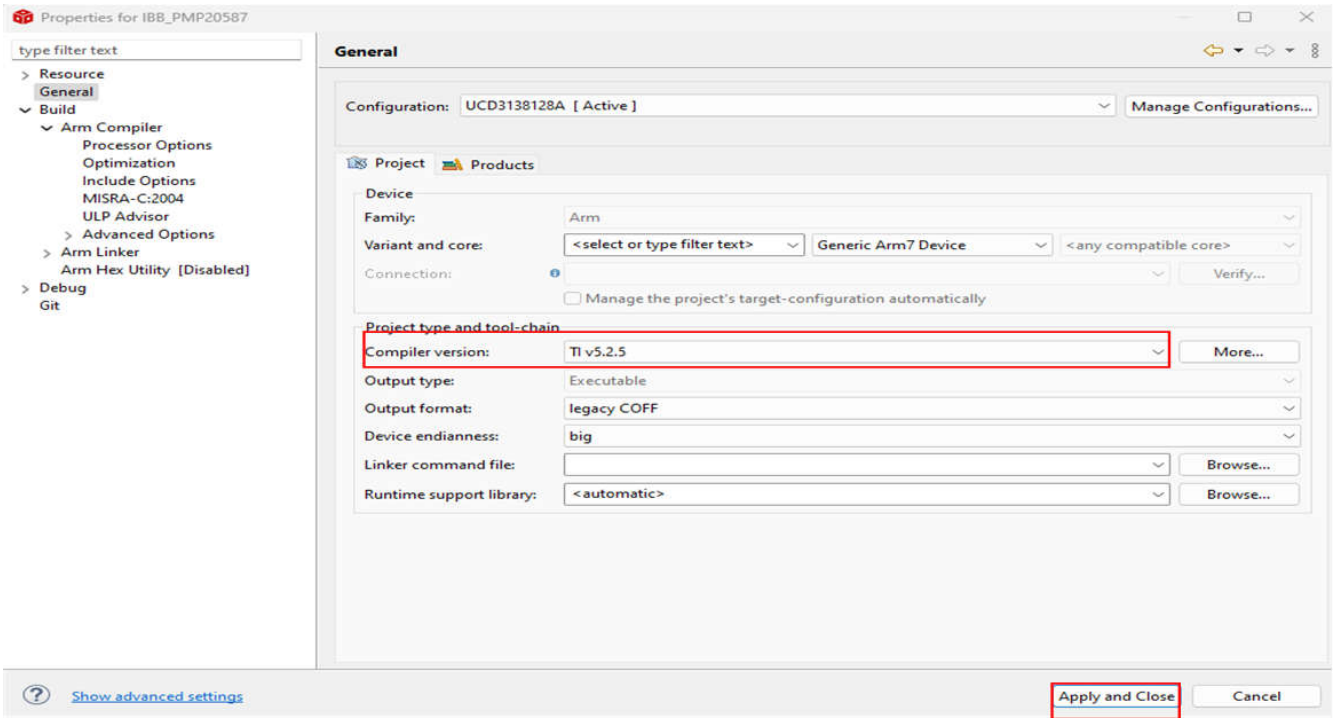

Figure 2-8. Switch to the Compatible Compiler Version

- Another potential cause is that the file "*software_interrupt.c*" is configured to a *code optimization level* that is too aggressive (such as "*2-Global Optimizations*"), causing the *checksum* to not be erased successfully. At the "*2-Global Optimizations*" level, it is true that some UCD3138 devices can successfully erase the checksum. However, some other UCD3138 devices fail to erase their *checksum*. To resolve this issue, the file "*software_interrupt.c*" must be configured to a lower *optimization level*, such as "*1-Local Optimizations*". The configuration steps are shown in Figure 2-9. Open the *Project Explorer pane*, right-click *"software_interrupt.c", go to Properties -> Build -> ARM Compiler -> Optimization -> 1-Local Optimizations*.

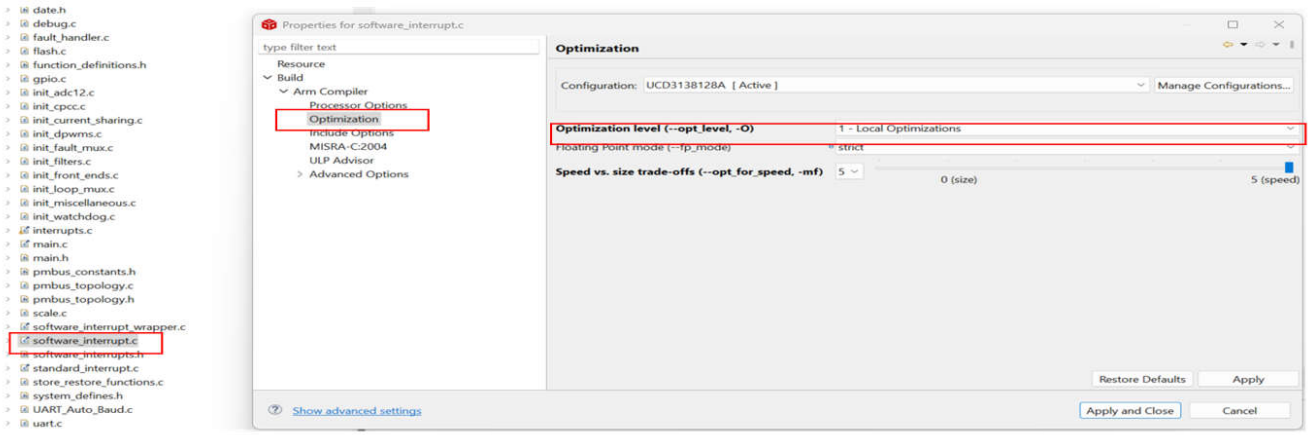

Figure 2-9. Select 1-Local Optimizations

2.4 Fixing the Issue Where No .x0 File Is Generated

The image downloaded to a UCD device is typically in .x0 format. However, sometimes users cannot find the .x0 file after a successful build. Instead, a .out file is typically generated directly after the build is complete. To generate a .x0 file, a post-build command line, *"${CG_TOOL_HEX}.exe" -x "${BuildArtifactFileName}" -o "${BuildArtifactFileBaseName}.x0" -memwidth 8*, should be executed. As shown in Figure 2-10, check whether this command line is configured correctly.

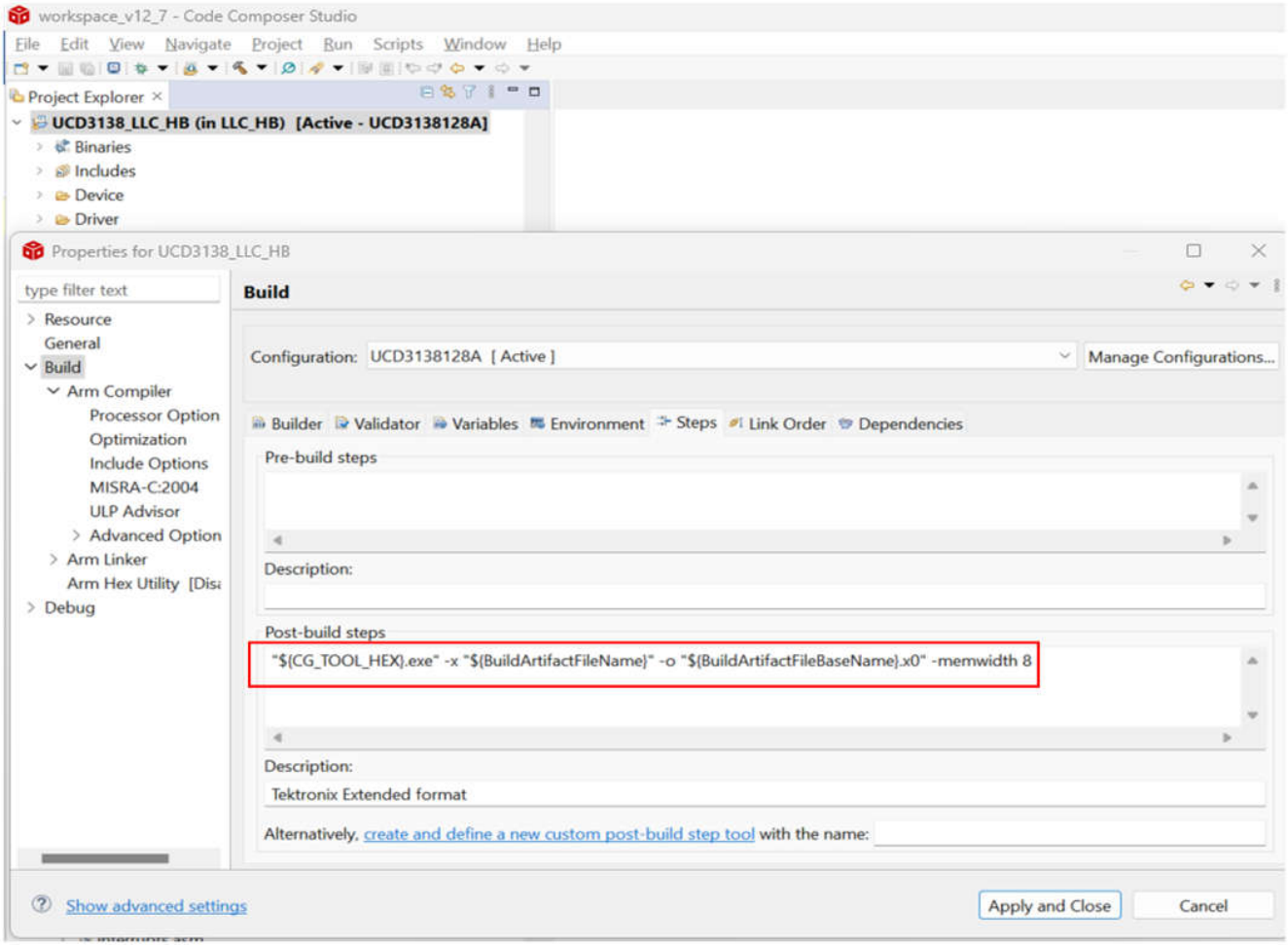

Figure 2-10. Post-build Step for Converting the .out File to a .x0 File

2.5 Fixing the Issue Where No .pp File Is Generated After Successful Build

If no .pp file is found in the project files, follow the steps below to complete the configuration.

- Configure the correct *preprocessor* file path. Open the *Project Explorer pane,* right-click *the project name, and go to Properties -> Build -> Arm Compiler -> Advanced Options -> Directory Specifier* to set the correct project path "*${PROJECT_ROOT}/UCD3138128A*" in the *Preprocessor file directory*. Figure 2-11 shows an example for UCD3138128A.

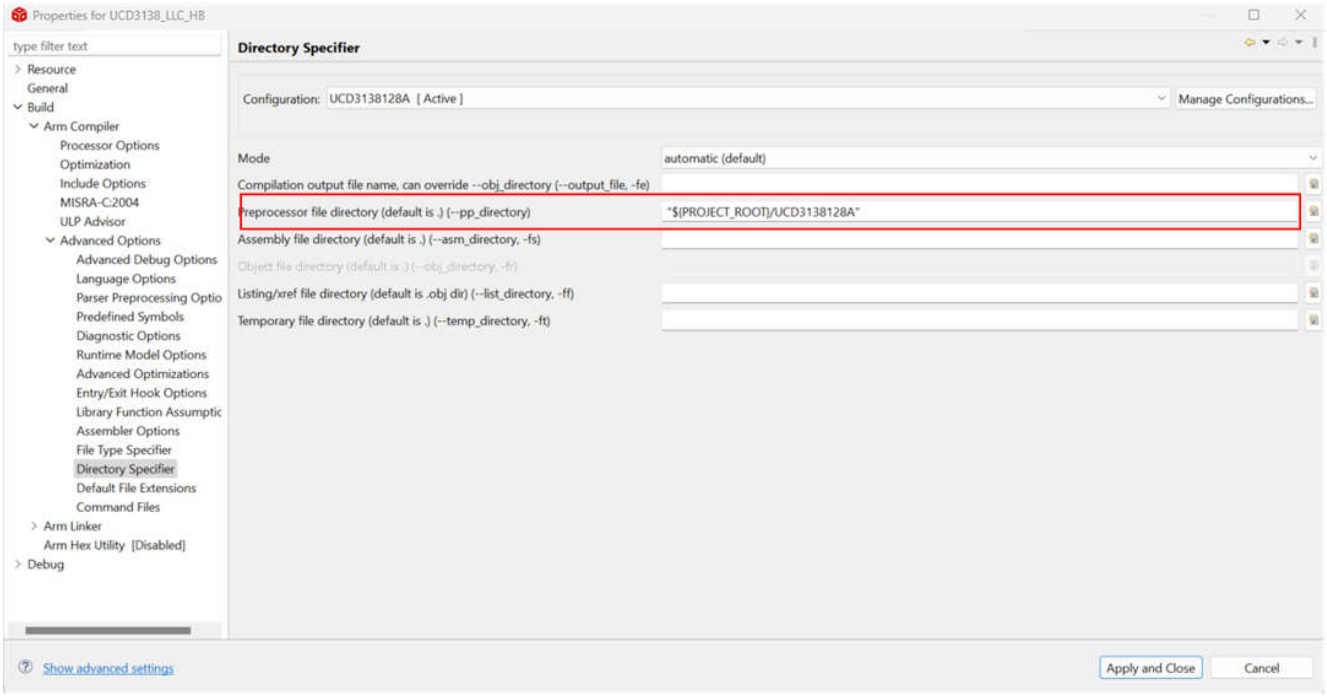

Figure 2-11. Correct Project Path for Preprocessor File Directory

- Configure the correct pre-defined symbols. Open the *Project Explorer pane,* right-click *the project name, go to Properties -> Build -> ARM Linker -> Advanced Options -> Command File Preprocessing -> Pre-define...,* and set UCD3138128A=1 as shown in Figure 2-12.

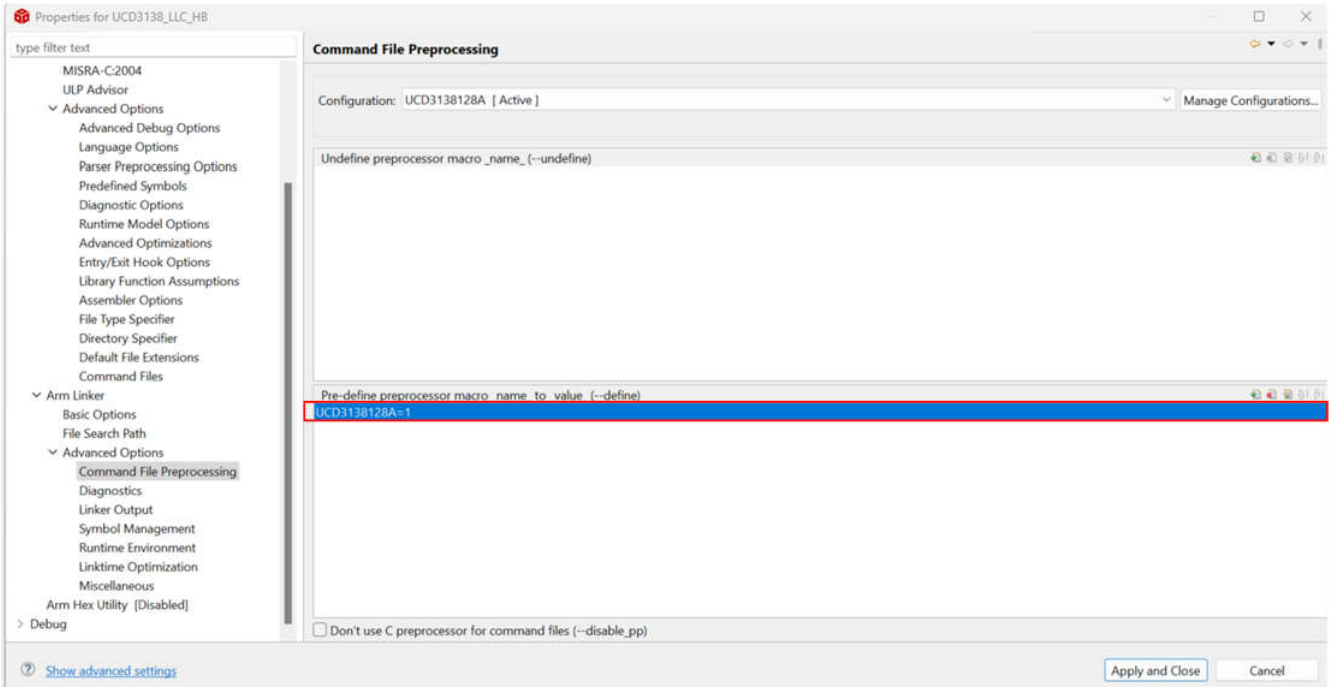

Figure 2-12. Correct Pre-define Preprocessor for Arm Linker

- *Click the Project Explorer pane, right-click the project name, go to Properties -> Build -> Advanced Options -> Predefined Symbols -> Pre-define NAME,* and set UCD3138128A=1 as shown in Figure 2-13.

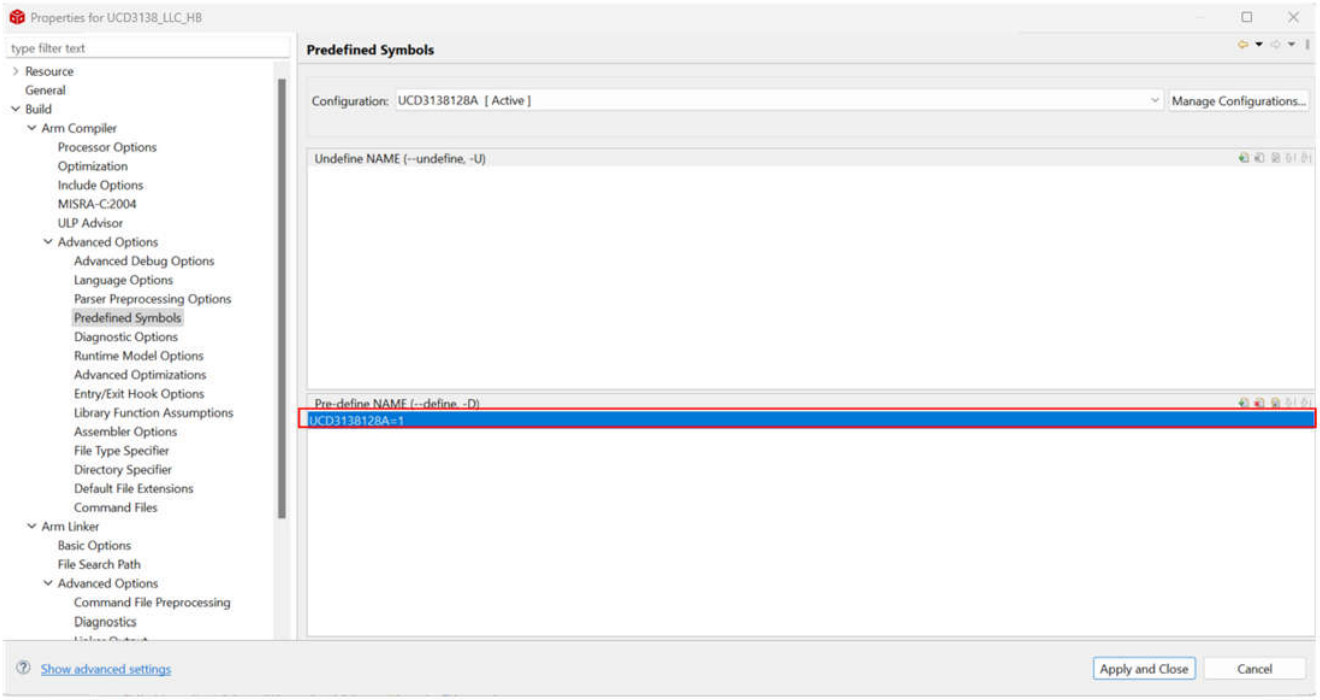

Figure 2-13. Correct Pre-define Preprocessor for Build

3 References

- UCD31xx Technical Reference Manual (Rev. E)
- Lockup Avoidance for UCD3138 Devices

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale, TI’s General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Last updated 10/2025