Application Note

AI Assisted Development and Migration Flow for AM13x MCUs

Nilabh Anand

Abstract

The growing complexity of embedded software development demands methodologies that accelerate new product development which involves evaluation of MCU platform for feasibility, building prototype, assessing scope of migration from previous platform and automated test and validation strategies, thereby reducing time-to-market.

This application note introduces TI's AI Assisted Migration tool which aids customers in above as well as describing other application areas in which this tool can be helpful for embedded developers.

Table of Contents

1 Introduction 2
2 Prerequisites 2
3 How to Setup and Use the Tool 3
4 Getting Started Fast: AI as a Bootstrap for New Projects 10
4.1 Identify the Target Device and Connected Hardware 10
4.2 Use SDK and Device-Specific Guidance 10
4.3 Start with Import a Matching Example Project 10
4.4 Configure Peripherals through SysConfig Using Natural Language 10
4.5 Write the Functionally Validated Application Code 10
4.6 Build and (If Starter Kit is Present) Flash and Run 10
5 Debugging Through the Probe: AI-Driven Control Over Real Hardware 11
6 Migrating Projects from Other MCU Ecosystems 12
7 Supported Devices and Peripherals 12
8 Summary 12
9 References 13

Trademarks

All trademarks are the property of their respective owners.

Copyright © 2026 Texas Instruments Incorporated

1 Introduction

The firmware development requires that the developer spends time with finding the right device, SDK, then tries to understand a clock tree, map peripherals to pins without creating conflicts, configure a toolchain, and obtain a evaluation board and debugger to final load and run the firmware. This approach particularly proved to be challenging when moving a production trial and testing code onto a new family of microcontrollers.

With the AM13x AI assisted migration and development flow, the above-described flow gets addressed in a much shorter time span. The IDE integrated AI assisted flow helps with analyzing the pinmux, source code and finally migrating to or developing from scratch on a new platform, matter of days than months. This document discusses how to setup the tool, how the tool helps with migration from the older microcontroller platform to TI AM13x MCU platform and how to leverage it,when developing a new platform or product.

2 Prerequisites

1. CCStudio Latest version
2. AM13E2x SDK
3. Application code or Reference Project for Migration
   a. STMG4 Github Repo (Reference)

Copyright © 2026 Texas Instruments Incorporated

3 How to Setup and Use the Tool

1. Download and install the latest version of CCS. Version 21.0.1 is the latest version.

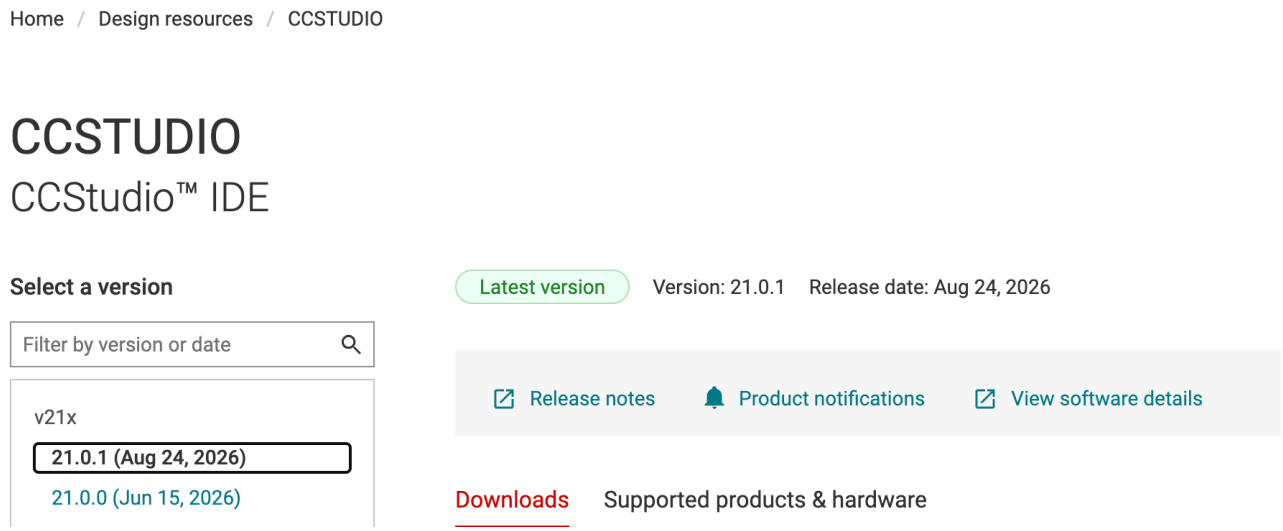


Figure 3-1. Download CCStudio

2. Open CCS -> check for updates, Install updates (CCStudio IDE AI MD Files)

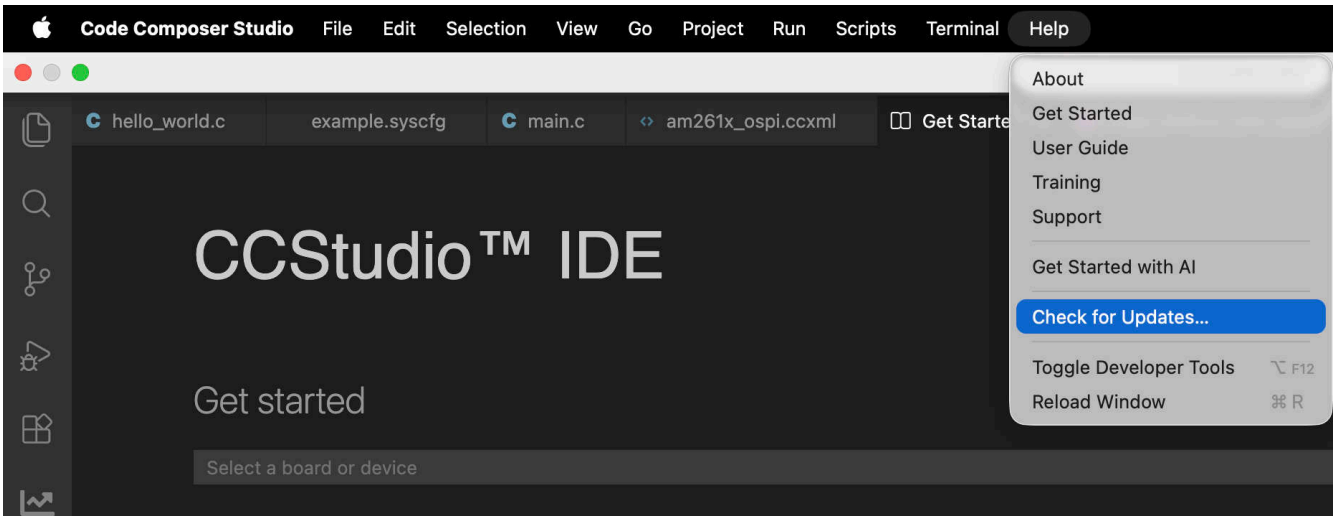


Figure 3-2. Install Updates



Figure 3-3. Update the AI MD Files

3. Once updates are installed, restart CCS.
4. Setting up AI provider:
   a. Go to help and click -> *Get Started with AI*

Copyright © 2026 Texas Instruments Incorporated



Figure 3-4. Setup AI Assiatant in CCS

Configure the AI provider in the toolbox

- Claude code subscription from anthropic: skip this and go next page where the user has an option to sign in using Anthropic account.
- Onprem AI provider setup : Use the console to enter URL, Auth Token and other parameters.
- Github: Follow this Appnote-ccs_ai_litellm_copilot
- Deepseek/Qwen: Follow this Appnote-ccs_ai_cline

Copyright © 2026 Texas Instruments Incorporated

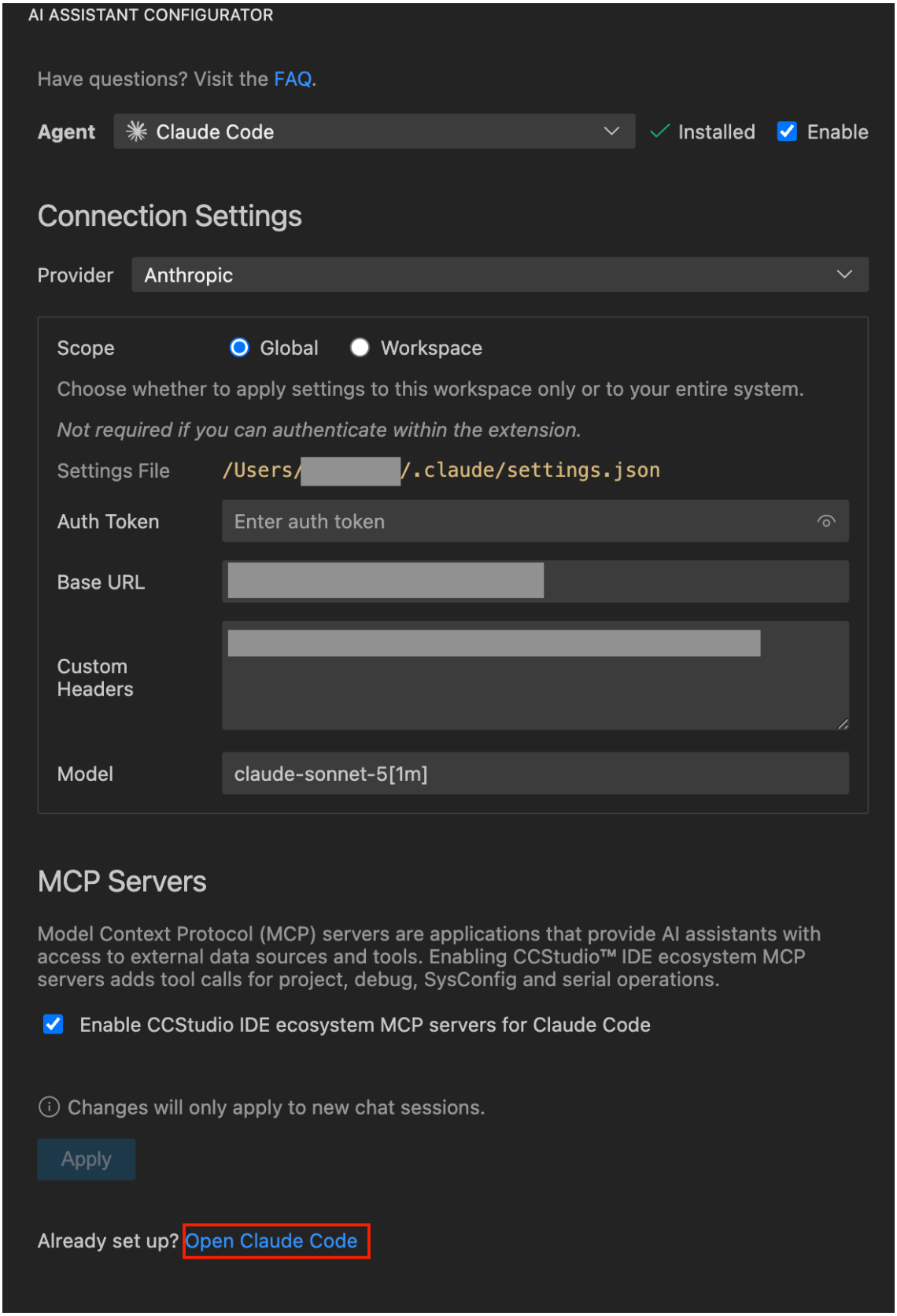


Figure 3-5. Select and Sign In To AI Provider Portal

5. A Claude code window opens.
   a. Set permission mode to Auto

Copyright © 2026 Texas Instruments Incorporated

b. Type */mcp* and check if all mcp servers are connected

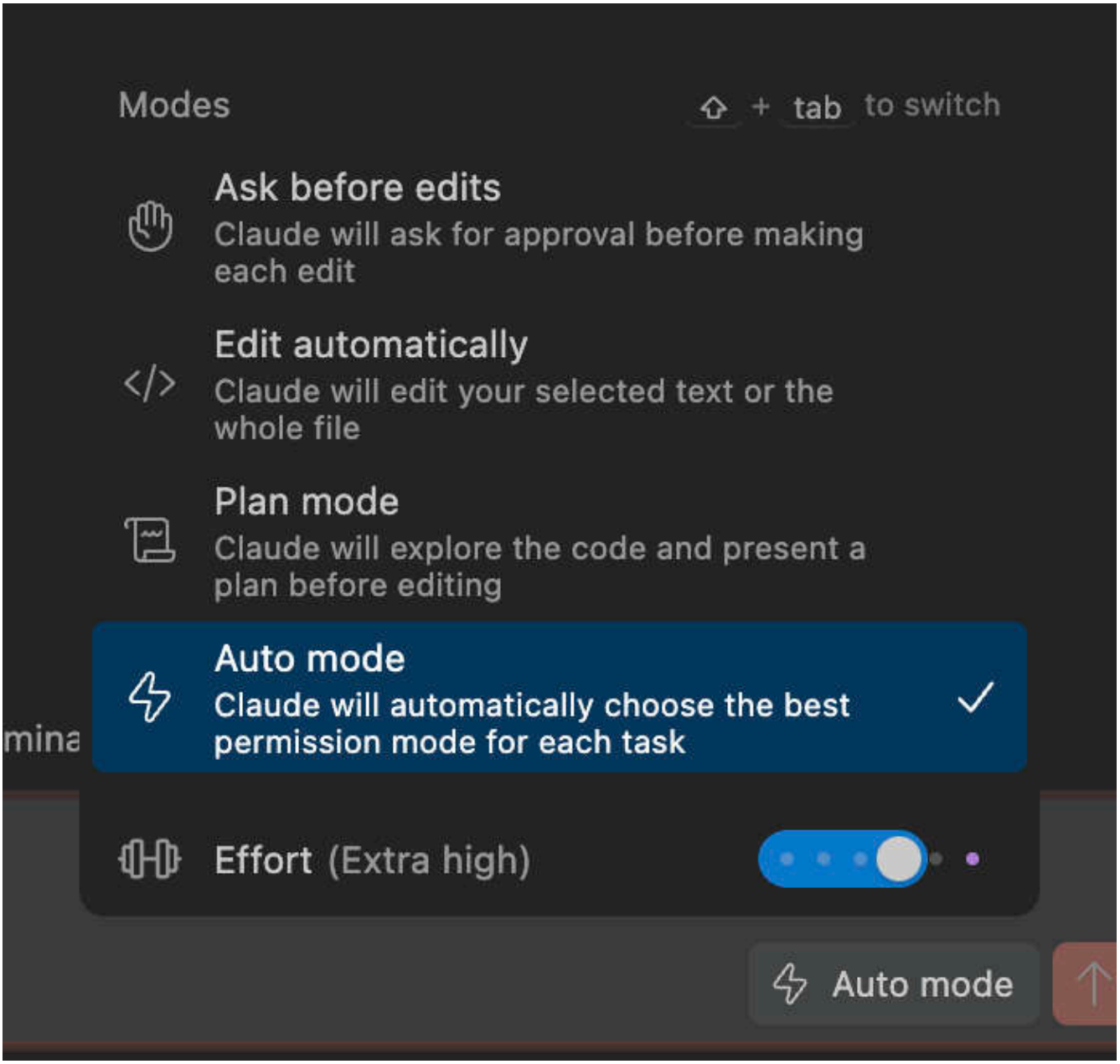


Figure 3-6. Auto Mode Selection to Avoid Multiple Approvals

6. Migrating a project
   a. Type */migrate-to-ti* and press *Enter* and skills are invoked, asking for project path selection

Copyright © 2026 Texas Instruments Incorporated

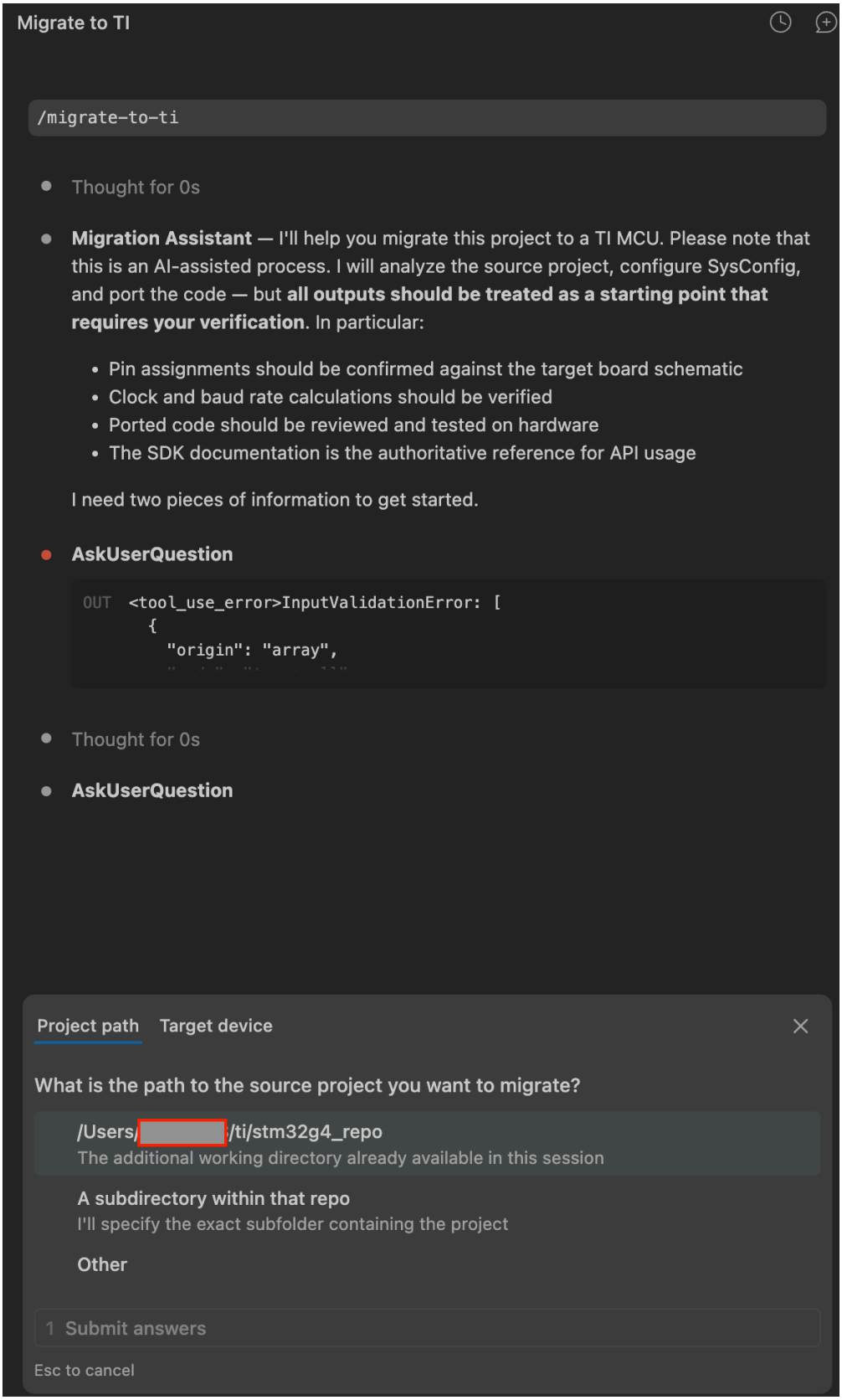


Figure 3-7. Starting the Migrate-to-ti Skill

Select the path where the project is located and target device as LP-AM13E230.

Copyright © 2026 Texas Instruments Incorporated

Figure 3-8 shows a basic prompt for migrating example from competition to AM13E320-LP

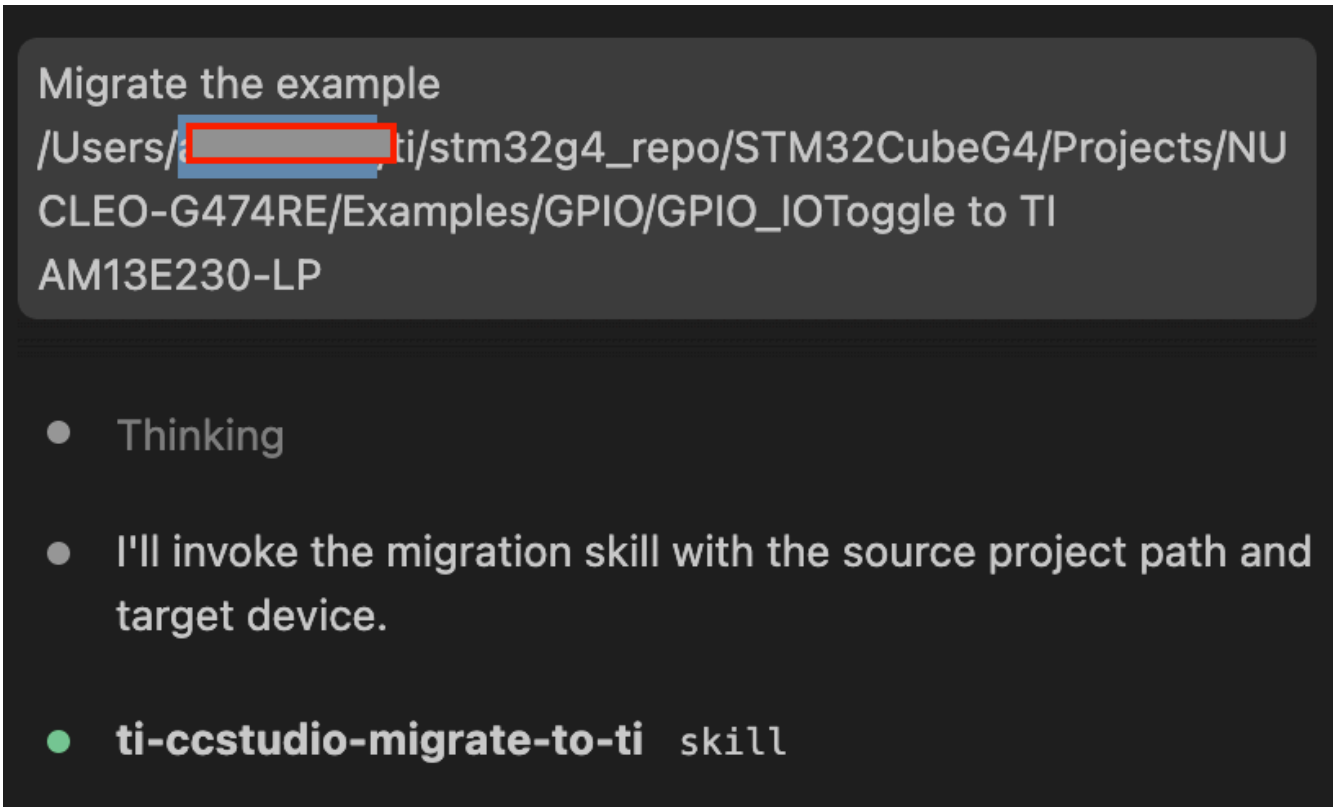


Figure 3-8. Sample Prompt to Migrate a Project

7. Once the migration starts, the skill generates and migrates a code, configures sysconfig and prompts the user for selection, once the migration is complete, the skill:
   - – Generate a CCS project
   - – Build test the projectr
   - – If the hardware/EVM is connected, it can flash the program and test it. For more details, see section 5.
   - – Develop Unit test and framework. For more details, see Debugging Through the Probe: AI-Driven Control Over Real Hardware.

1. If all steps are complete, then the process concludes by generating a migration report:

Copyright © 2026 Texas Instruments Incorporated

Migration Report COMPLETE

Source: STM32G474RETx (NUCLEO-G474RE) **Target:** AM13E23019 (package LQFP64_G)
LaunchPad: am13e230x_lp (exact LaunchPad SKU unconfirmed — candidates: LP-AM13E2304-48QFN, LP-AM13E2304-48QFP, LP-AM13E2307-100QFP)
Last Updated: 2026-10-16

AI-Assisted Migration — This report was generated by an AI assistant. It is a starting point, not a guaranteed specification. Verify all peripheral mappings, pin assignments, clock calculations, and API usage against the target device datasheet and SDK documentation before deploying to production.

PROJECT DETAILS

SOURCE PROJECT **GPIO_IOToggle_AM13E230**	SOURCE BUILD SYSTEM **ti_arm_clang (CCS native project)**
CCSTUDIO IDE PROJECT **`GPIO_IOToggle_AM13E230`**	SDK STARTING POINT **`examples/empty/am13e230x_lp/m33_nortos/ti_arm_clang`**

TOOLS

CCSTUDIO IDE **21.0.0**	SYSCONFIG **1.28.0**
TI ARM CLANG COMPILER **5.1.1.LTS**	SDK **AM13E2X-SDK 26.1.0.03**

MEMORY

Resource	Source	Target Available	Status
No flash/RAM memory analysis data provided for this migration (simple GPIO-toggle demo; not memory-constrained)			

PERIPHERAL SUMMARY

Peripheral	STM32 Usage	TI Equivalent	Status	Notes
GPIO	PA5 push-pull output (LED2) via	GPIO_GRP_0 digital output,	Ported	Pin PA5 maps to package pin 19 (LQFP64_G);

Figure 3-9. Sample Migration Report

Copyright © 2026 Texas Instruments Incorporated

4 Getting Started Fast: AI as a Bootstrap for New Projects

This section talks about how this tool can be used for an accelerated getting started experience. In a traditional setup, a developer who wants to prototype an idea: *blink an LED*, *stream ADC samples over UART*, *drive a motor with a PWM pattern* — has to context-switch between a device datasheet, a reference manual, SDK example code, and the IDE itself before starting actual prototyping.

With the AM13x AI Assisted Development and Migration tool built into the CCStudio IDE, the bootstrapping process collapses into a conversation. The goal can be described in plain language (the peripheral configuration and required setup) and the AI model:

4.1 Identify the Target Device and Connected Hardware

Based on user input, the tool provides information about the SoC package info and pins. The tool accesses the device data present inside IDE/Syscfg to access pinmux database, connections and components that are wired along with EVM-specific pinmux, onboard peripherals (LEDs, buttons, sensors).

4.2 Use SDK and Device-Specific Guidance

The AM13Ex family of MCU ships SDK with driver libs, example projects and reference designs. The structured knowledge base provides a device-specific guide covering initialization sequences, SoC peripherals (UART, I2C, and so on) and board components (LEDs) pin assignments, known issues, and required coding patterns, plus supplemental guides for individual peripherals and board-specific documentation for the exact LaunchPad or BoosterPack in use. The tool accesses this material before generating or modiyfing any code. This provides a deliberate guardrail from hallucination.

4.3 Start with Import a Matching Example Project

Instead of generating a new project structure, the tool uses the SDK project as a starting point. The advantages are SDK-validated examples, the build system, linker command file, and startup code that are already correct, making sure there are minimal build hiccups.

4.4 Configure Peripherals through SysConfig Using Natural Language

SysConfig is a configuration tool from TI that simplifies hardware and software configuration by providing an intuitive graphical user interface for configuring pins, peripherals, software stacks, RTOS, clock trees and other components, and automatically detects, exposes and resolves conflicts to speed software development. This allows users to capture this hardware and software configuration for TI platforms and generates the.c/.h files that are built as part of CCS project.

The AI assisted migration and development tools further simplifies the configuration. The tools allow for natural language based SysConfig's UI option selection. Ask AI to configure deadtime for MCPWM to 40ns; an AI agent performs this in sysconfig. The agent can intelligently use sysconfig to generate verified peripheral initialization and configuration code, can add a module instance, set its configurable, including conflicts and warnings and device/package migration within SysConfig are all exposed as hooks the AI migration tool can use.

4.5 Write the Functionally Validated Application Code

This is another major advantage that the tool brings. When any general use LLMs are used to write the firmware, the LLM often hallucinates and invents plausible looking but incorrect API names. In the tool, there are guard rails set to write the application code, using the SDK's actual generated macros and driver-layer function calls. As mentioned in the prerequisite, the AM13E2x SDK is needed for the tool to function properly. Before generating the code, the AI agent has already read the SDK's structured guidance and inspected real examples for the peripheral in question.

4.6 Build and (If Starter Kit is Present) Flash and Run

At the last step, the tool offers to iteratively compile the project and fix all the build errors. If hardware is present, flash and run — closing the loop from an idea to a running, observable result (an LED blinking, a UART stream appearing in a console, a sensor value changing). Section 5 discusses this more.

Copyright © 2026 Texas Instruments Incorporated

5 Debugging Through the Probe: AI-Driven Control Over Real Hardware

The ability to debug the project autonomously is perhaps the most consequential capability, because it touches physical hardware directly. CCStudio IDE's debug tooling gives the AM13 AI assistant tool the low-level control at the debugger console.

It allows hardware discovery by enumerating connected debug probes and the devices behind them — for example recognizing a specific LaunchPad by its board name and silicon ID over its XDS110 (or compatible) connection — before launching anything. It can launch a debug session within a project or project-less as well, set source-level breakpoints set by file and line. This further extends debug ability by providng full execution control: continue, pause, restart, and full target reset, expression and register view included.

Now the AI has a lot more context to inspect for the currently halted stack frame than just raw memory. This can identify the actual meaning behind the exceptions by looking at variable or peripheral register at the moment of interest and even modify values at runtime.

Logic Analyzer (Salea): The debug ability gets further extended by allowing the tool visibility into the physical pin behavior by installing a logic analyzer to understand the signal behavior. This feature allows the users to debug faulty MCPWM behavior, UART signals, and much more, by providing a signal view to AI agent [6].

During the debug session it will ask use to execute or observe specific physical setup like jumper wires or external signal sources. When source code changes during an active session, the AI follows the correct rebuild-reload-continue sequence: rebuild, explicitly load the new program image. The AI accelerates the mechanics of debugging; it does not take the human out of the decision to act on physical hardware.

Copyright © 2026 Texas Instruments Incorporated

6 Migrating Projects from Other MCU Ecosystems

The capability with another broadest practical impact is whole-project migration — taking an existing embedded application written for a different microcontroller family entirely and porting it to the AM13x MCU family, with the AI assisted migration and development tool.

This is a structured, multi-stage pipeline built on a layered knowledge base specifically designed for cross-platform migration. This takes into account the source ecosystem's conventions namely pin configuration, project structure, configuration file formats, HAL-style API patterns versus register level code), manifests that describe each supported target device family (the SDK, unsupported features relative to the source, and known pin-translation formulas), and path-specific guides for common source-to-target combinations that go further — down to peripheral-by-peripheral translation notes for timers, communication interfaces, analog comparators, and more.

A migration proceeds through the following stages:

Source analysis: The tool scans the source project in its entirety, its configuration files, HAL or driver-layer calls, middleware dependencies, and build system. It produces a structured summary of exactly what the project does, how peripherals are configured and used, what clock configuration it relies on, and what pin each signal is wired to and the exact memory requirement

Migration planning: That summary is then matched against the target device's capabilities and known unsupported features, producing a concrete plan: which peripherals map directly, which require a different approach, which source features (if any) are not supported on the target and need a workaround or an explicit scope decision, and a human-readable assessment presented.

Developer checkpoint: The developer must then review the assessment and explicitly command whether to proceed, request changes to the plan, or stop.

Configuration and porting, area by area: For each functional area of the project, the target device's configuration is built up through the same validated SysConfig tooling described previously, including, where a pin-translation table exists for the source-to-target pair, directly resolving pin assignments. The application code is rewritten against the target's actual API, with the same identifier-preservation discipline applied during ordinary development: the structure, naming, and intent of the original code are preserved; only what genuinely needs to change for the new platform does.

Migration report: This is an artifact capturing exactly what was changed, how each API was mapped, if it was fully ported or partially or cannot be ported. All warnings or caveats and what manual verification remains outstanding are put together for developer assessment.

Build verification and Hardware validation: This is described in Debugging Through the Probe: AI-Driven Control Over Real Hardware.

7 Supported Devices and Peripherals

See the FAQ for more information.

8 Summary

This document discusses a few use cases on how to use the ASM AI assistant migration tool for developing a new project from the beginning as well as migrating the example from older platforms. While there are other use cases such as prompting AI to learn about device architecture and to understand peripherals, these use cases are up to the the user to execute.

Copyright © 2026 Texas Instruments Incorporated

9 References

1. Texas Instruments, *AI Driven Application Development in CCStudio*, video.
2. Texas Instruments, *Using GITHUB Copilot with CCStudio AI Tool* , webpage.
3. Texas Instruments, *How to Use Qwen/DeepSeek and Other Model in CCStudio*, webpage.
4. Texas Instruments, FAQ, FAQ.
5. Texas Instruments, E2E, forum.
6. Texas Instruments, *Using Logic Analyzer with AM13x Migration and Development Tool* , video.
7. Texas Instruments, *Supercharge development with AI-powered CCStudio™ IDE*, video.
8. Texas Instruments, *AM13E230 STM32G4 Migration Guide*, user's guide.

Copyright © 2026 Texas Instruments Incorporated

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.

Copyright © 2026, Texas Instruments Incorporated

Last updated 10/2025