Application Note

AM13x Model-Based Design Integrated with TI SysConfig

Arun Munuswamy, Nilabh Anand, Val Singh

ABSTRACT

The growing complexity of embedded control applications demands development methodologies that accelerate algorithm design, enable early-stage testing, and reduce time-to-market. Model-Based Design (MBD) with MATLAB/Simulink has emerged as a powerful paradigm that addresses this challenge, allowing engineers to design, simulate, and automatically generate production-ready embedded C code, catching design errors early in the development cycle.

Simultaneously, Texas Instruments' System Configuration Tool (SysConfig) has established itself as a unified graphical interface for peripheral and clock configuration across the entire TI microcontroller (? EP) portfolio. By providing a single, consistent configuration experience from pin multiplexing to driver setup, SysConfig eliminates hand-coded initialization boilerplate and ensures that hardware configuration intent is clearly captured, shareable, and reusable across projects and teams. This application note presents the integration of these two complementary technologies within the Embedded Coder Support Package for Texas Instruments AM13x.

Table of Contents

Trademarks

1 Introduction

A central enabler of this integration is the shared syscfg configuration file between the MATLAB/Simulink model and Code Composer Studio (CCS) IDE project. This common configuration verifies that peripheral settings, pin assignments, and clock definitions remain *consistent and synchronized* across both workflows, eliminating redundant setup and reducing the risk of configuration mismatches between simulation and hardware.

This interoperability creates a natural bridge for teams transitioning from traditional hand-coded development to Model-Based Design — or conversely, for model-based developers who need to hand-off or collaborate with embedded software engineers working in CCS. Algorithmic IP developed and validated in Simulink can be seamlessly carried forward into a CCS-based integration environment, and vice versa, without redefining the underlying hardware configuration.

The result is a cohesive and flexible development ecosystem that significantly reduces integration friction, supports parallel development across disciplines, and ultimately accelerates the journey from concept to a verified, deployable embedded application on AM13x devices.

2 Prerequisites and Getting Started

- Installation steps and required tools are well documented in GitHub - TexasInstruments/am13x-hsp
- Videos tutorials and FAQs are shown in [FAQ] Embedded Coder Support Package for Texas Instruments AM13x

3 SysConfig Integration with MATLAB and Simulink

Traditionally model settings also configure peripherals and other resources, since SysConfig provides all these features now user needs to just supply a *.syscfg* file, either sourced from an existing SDK example or created interactively within Code Composer Studio. Embedded Coder build process launches SysConfig tool to generate all necessary peripheral initialization code. This generated code from SysConfig linked with the algorithm and scheduler code produced by Simulink, results in a complete application binary.

4 Model Anatomy and SysConfig Entry Point

This led example is one of the simplest example explaining the integration with SysConfig

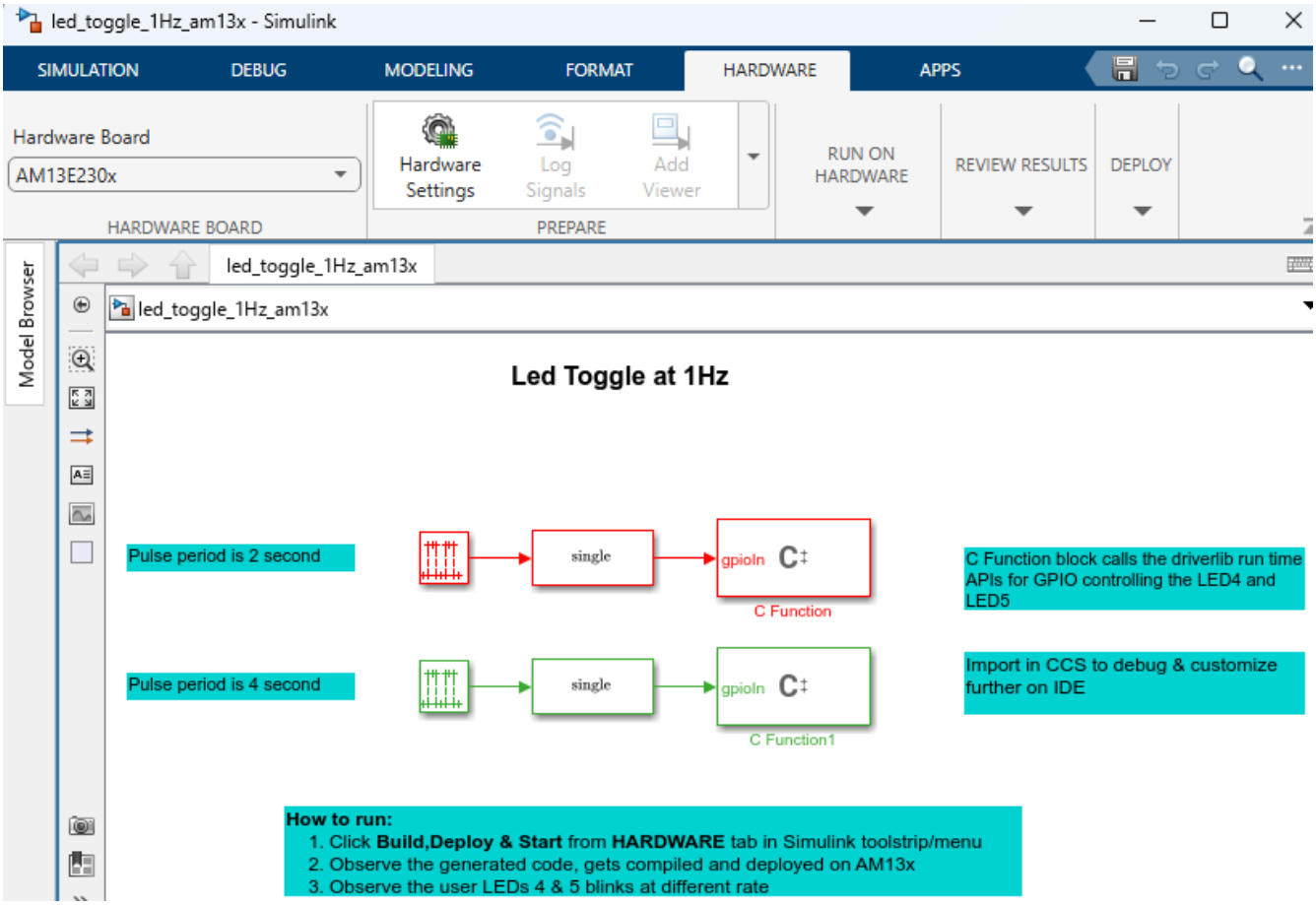


Figure 4-1. Simulink Block for Led Toggle Program

Hardware Settings/Model Settings showing the SysConfig entry point:

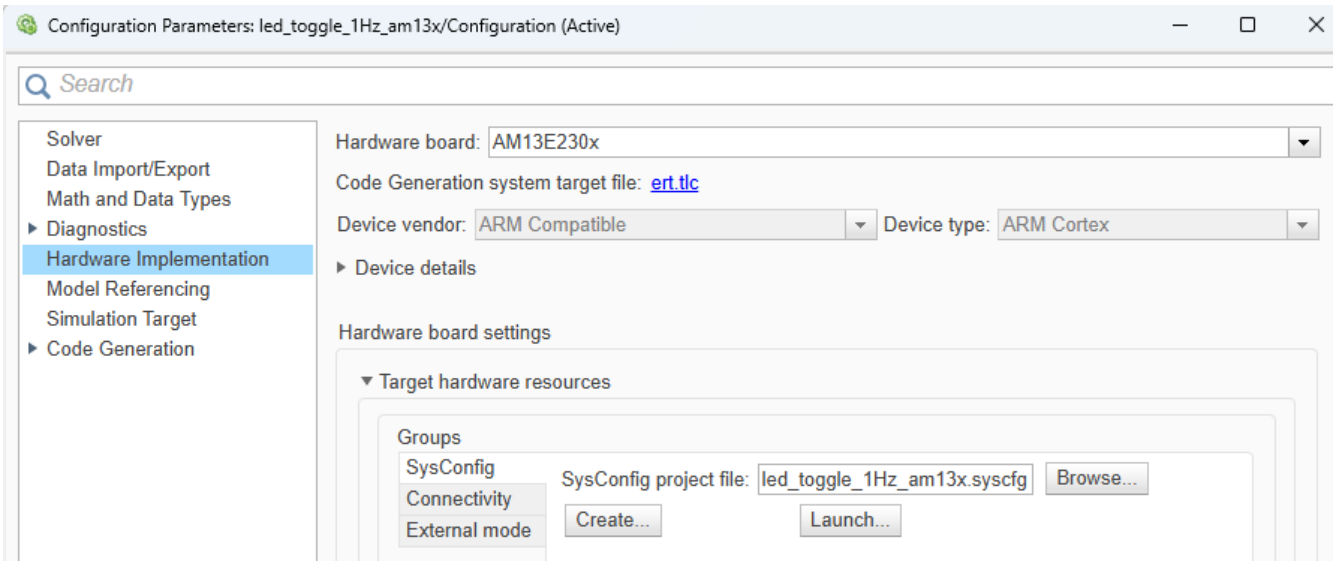


Figure 4-2. Hardware Target Selection

SysConfig launched from Simulink showing two LED configuration

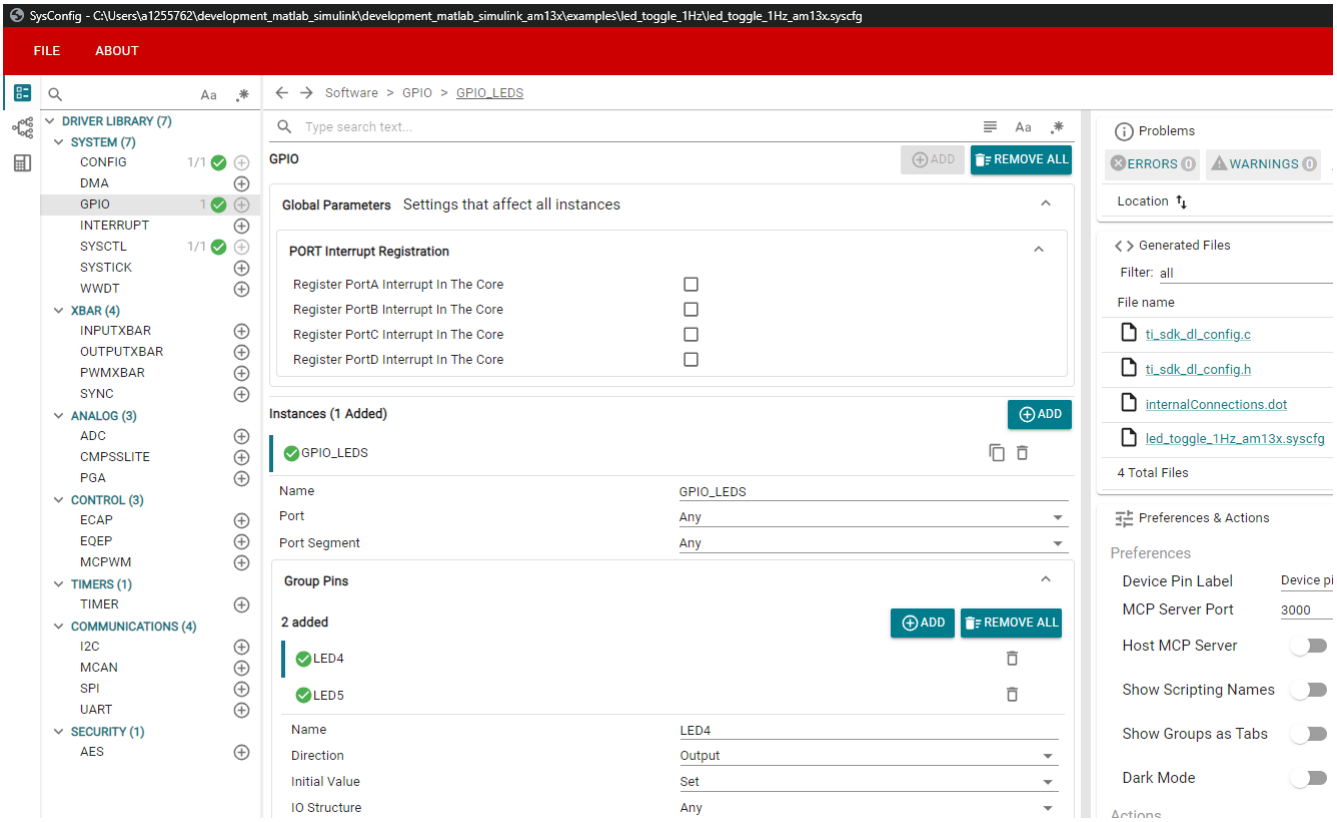
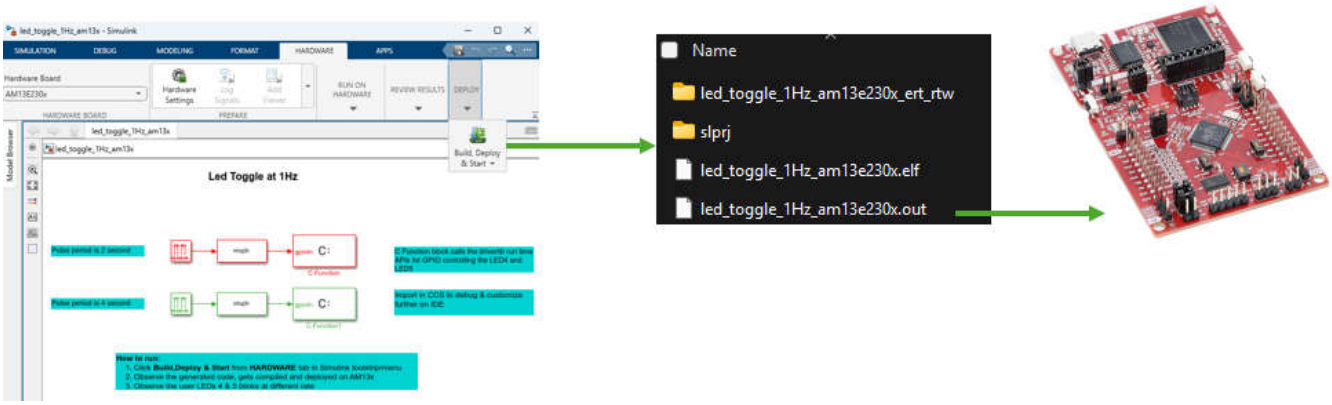


Figure 4-3. GPIO Sysconfig

5 Build and Deployment Overview

When the user clicks *Build, Deploy & Start* in Simulink, code generation, compilation, and deployment to the connected AM13x Launchpad execute automatically in a single operation. The generated output folder contains all artifacts required for the complete application build, organized as follows:



5.1 Codegen Structure

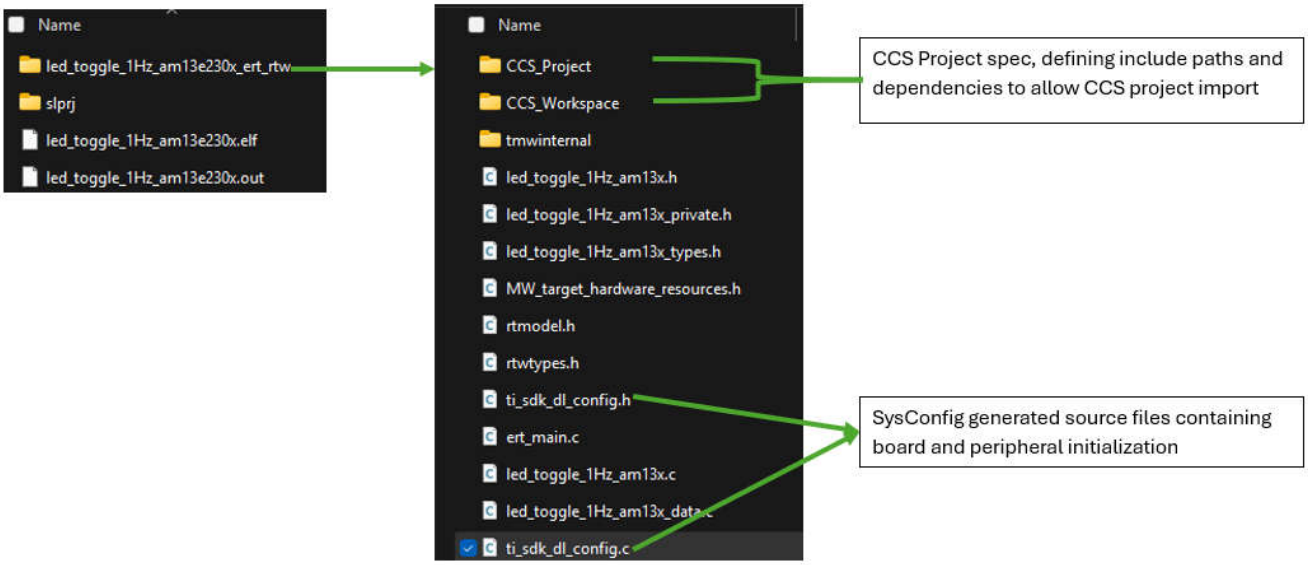


Figure 5-1. CCS Project Structure

Here is simplified explanation of application entry point main function defined in ert_main.c

```c
int main(void)
{
    /* Step 1: SysConfig-generated peripheral initialization */
    SYSCFG_DL_init();

    /* Step 2: Simulink-generated model initialization */
    led_toggle_initialize();

    /* Step 3: HSP scheduler initialization */
    am13x_tick_init(BASE_RATE_TICKS);

    /* Step 4: Enable interrupts — execution driven by SysTick_Handler */
    __enable_irq();

    while (1)
    {
        /* Idle loop */
    }
}

/* Simulink-generated scheduler ISR (am13x_tick.c) */
void SysTick_Handler(void)
{
    rt_OneStep();   /* Invokes led_toggle_step() at base sample time */
}
```

Figure 5-2. Program flow

5.2 Debug and Customize Simulink Generated Code in CCS

This section is explained in *Verification and Validation*.

6 SysConfig for a Reference Motor Control Setup

Similar to the led example, Motor control example contains models and SysConfig files for more complex FoC based motor control application where more peripherals such as ADC, PWM, Interrupts, GPIOs are used.

6.1 AM13x Simulink Model

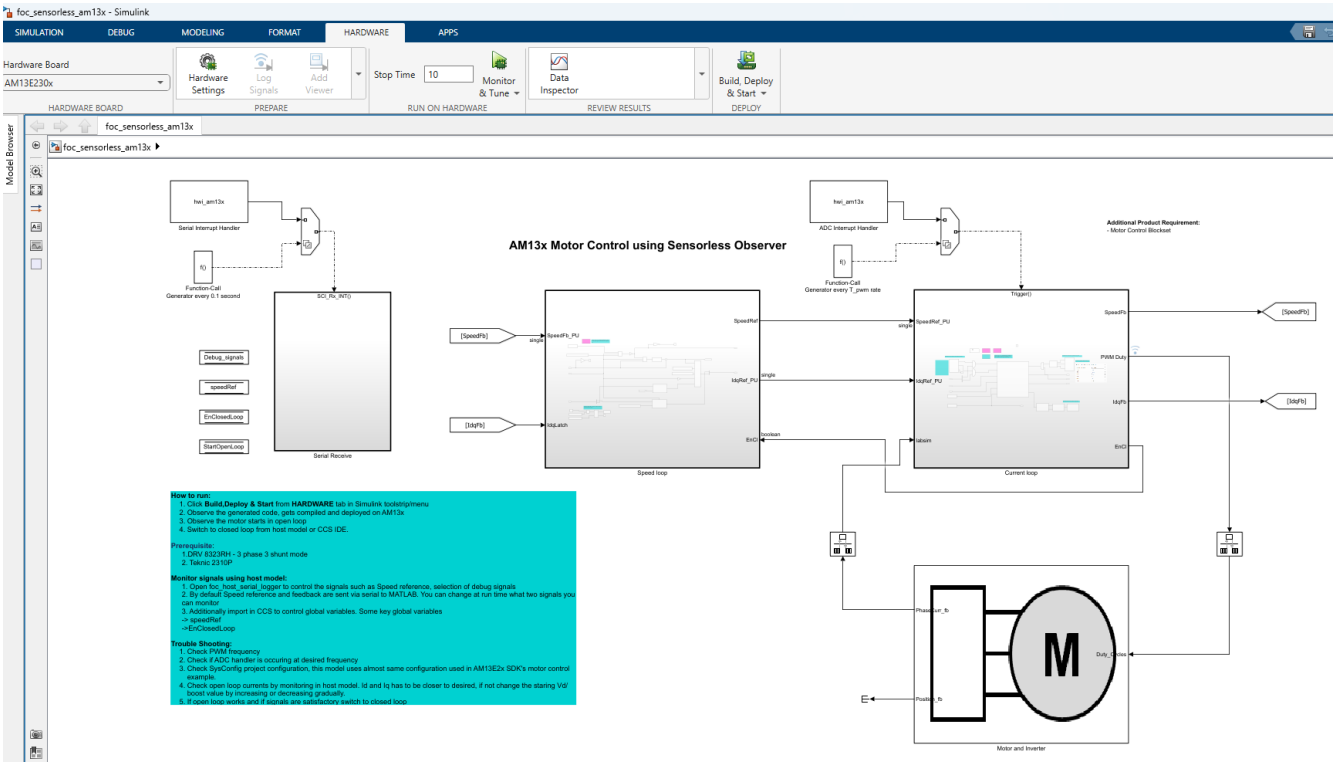


Figure 6-1. Simulink Block for Motor Control FoC

6.2 PWM Configuration in SysConfig

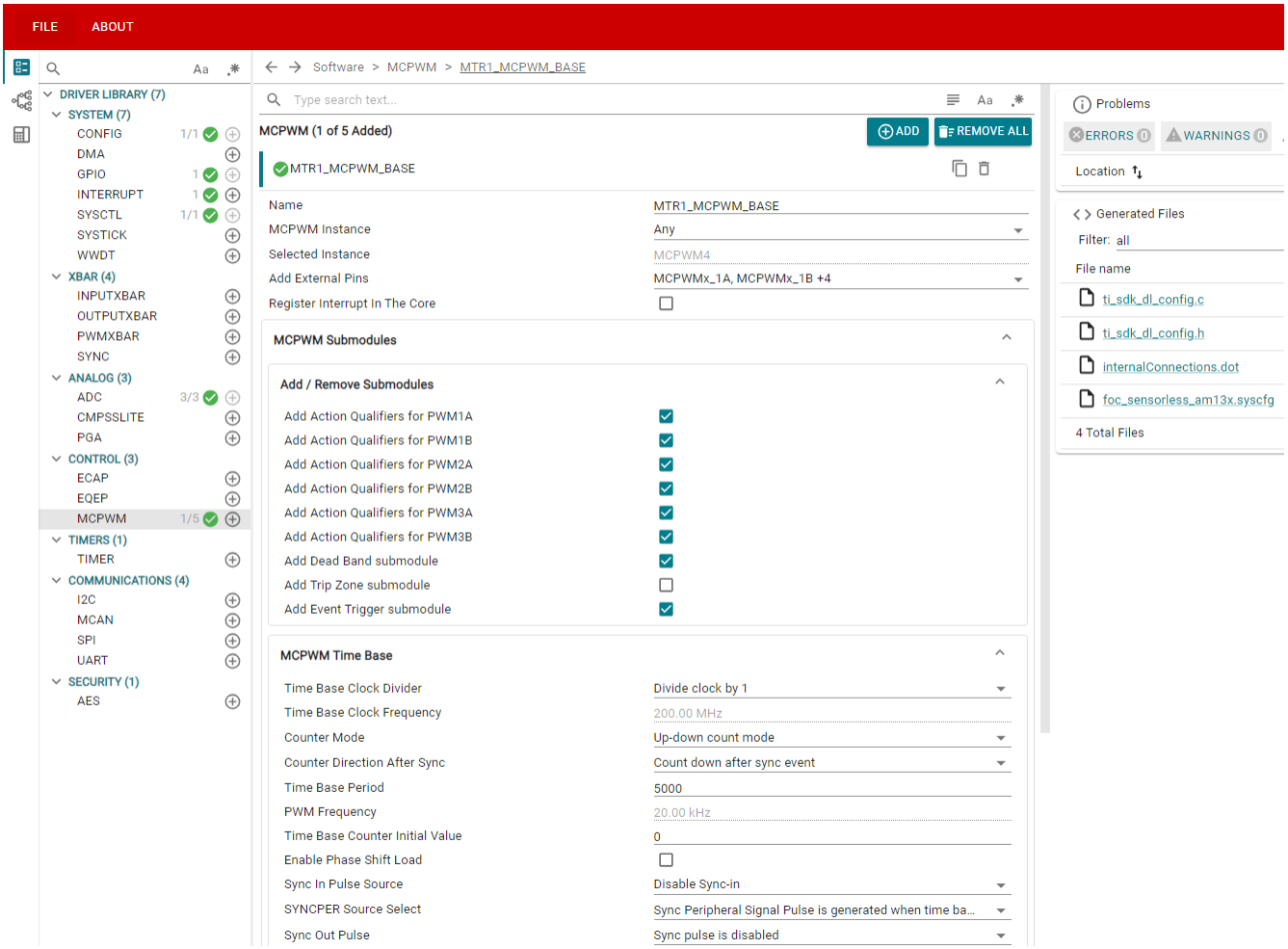


Figure 6-2.

6.3 ADC Configuration in SysConfig

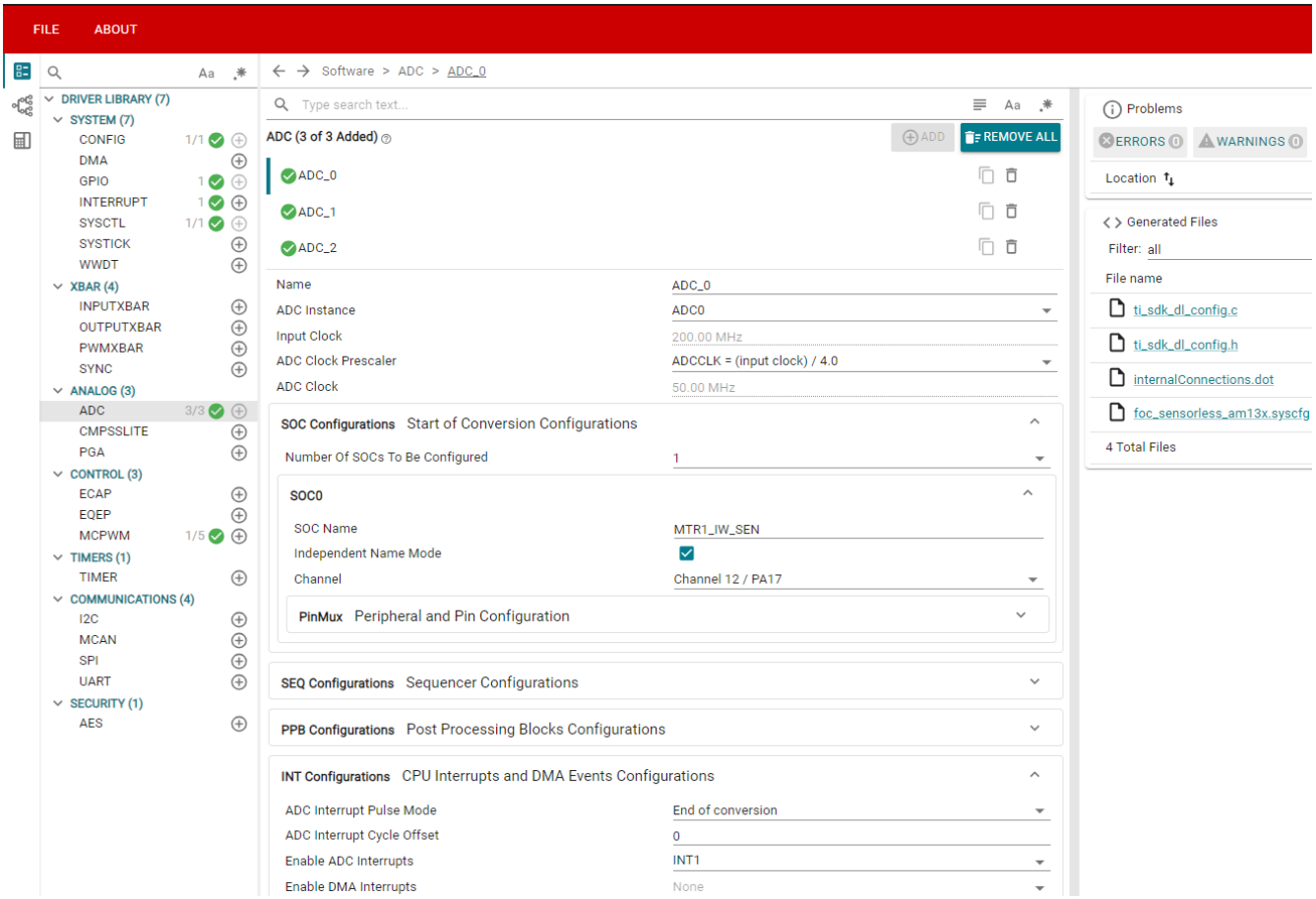


Figure 6-3.

7 Summary

This application note has presented a structured methodology for integrating TI SysConfig with the MATLAB/ Simulink Model-Based Design workflow targeting the AM13x microcontroller family. SysConfig serves as the single definition of all peripheral/MCU configuration verifying consistency between hand-coded and model-generated embedded software.

8 References

Texas Instruments, *Matlab BlockSet Training*, training.

Texas Instruments, *[FAQ] Embedded Coder Support Package for Texas Instruments AM13x*, FAQ.

Texas Instruments, *AM13E2X-SDK*, product page.

Texas Instruments, AM13E2X-HSP, hardware support package.

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