Application Note

PMP41140 - 280W Digital Asymmetric Half Bridge DC-DC Converter Software Guide

Shashank Madineni, Xingqi Chen

ABSTRACT

This document provides the software documentation for a digitally controlled Asymmetrical Half Bridge (AHB) converter, a Flyback-derived resonant topology designed for high-efficiency and wide-output power conversion. The firmware implements zero-voltage switching (ZVS) control using a fast PWM on-time calculation and adaptive zero-voltage detection (ZVD) technique to maintain boundary ZVS across varying input and load conditions. It also includes programmable ZCD count control, adaptive soft-start, and inductance compensation algorithms to ensure stable performance under production variations.

The software has been validated on a 280 W AHB hardware prototype, achieving a peak efficiency of 97.8% at 28 V / 9A. This documentation details the software structure, control algorithms, and implementation methods that enable robust digital control of AHB converters, targeting high-power applications (>100 W) such as USB PD 3.1 adapters, industrial chargers, and power tools.

Table of Contents

List of Figures

List of Tables

Trademarks

All trademarks are the property of their respective owners.

1 Introduction

The Asymmetrical Half Bridge (AHB) converter is a resonant isolated DC-DC topology that merges the simplicity of Flyback converters with the performance of resonant switching designs. As shown in Figure 1-1, the AHB consists of a high-side (HS) and low-side (LS) MOSFET's connected in series between the DC bus voltage V_{BUS} and ground, forming the half-bridge node V_{HB} . A resonant inductor (L_r) and resonant capacitor (C_r) shape the voltage waveform at the switching node, enabling smooth transitions and achieving zero-voltage switching (ZVS). The transformer provides isolation and energy transfer between the primary and secondary sides, while its leakage inductance (L_r) and magnetizing inductance (L_m) define the resonant characteristics. On the secondary side, a synchronous rectifier (SR) and output capacitor provide a regulated output voltage V_O and current I_O.

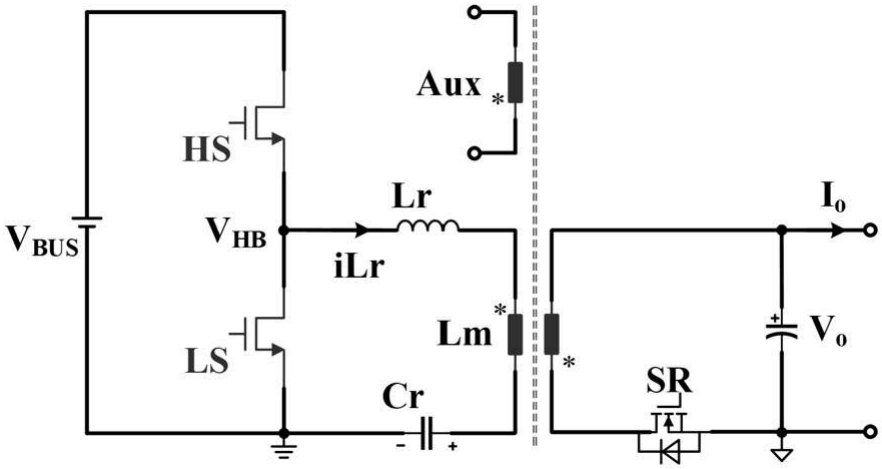


Figure 1-1. Asymmetrical Half Bridge (AHB) DC-DC Converter

The inductors L_r and L_m represent the leakage and magnetizing inductances of the transformer equivalent model, while the capacitor C_r denotes an external resonant capacitor added to shape the resonant current and voltage. During each switching cycle, energy is stored in the transformer's magnetizing inductance during the ON period of the low-side MOSFET and released to the output through the secondary winding during the OFF period. The resonant interaction between L_r and C_r allows the high-side MOSFET to switch on when its drain voltage approaches zero, thus achieving ZVS and minimizing switching losses. The AHB topology is ideal for medium- to high-power (>100 W) isolated applications requiring compact size, high power density, and high efficiency. It supports a wide output voltage range and allows duty cycle operation beyond 50%, unlike traditional Flyback or LLC converters. These advantages make it an attractive choice for USB PD 3.1 adapters, industrial power tools, all-in-one PCs, and smart chargers.

Maintaining ZVS across a wide operating range remains a key challenge. The ZVS condition of the high-side switch (HS) is met when the half-bridge node V_{HB} naturally resonates up to V_{BUS} before the next turn-on event. Achieving this condition depends on the load current, transformer parameters, and switching frequency. To ensure consistent ZVS, the controller dynamically tunes the PWM on-time and switching frequency based on system feedback. While conventional analog control approaches offer limited flexibility and parameter visibility, a digital implementation provides precise timing control, programmability, and the ability to adapt in real time. The digital controller described in this documentation uses auxiliary winding feedback for ZVS detection, implements adaptive soft-start and safe-stop routines, and allows programmable parameters for ZCD count and delay adjustment. These features enhance converter performance, enable faster design iteration, and improve manufacturing robustness. This document provides a comprehensive software overview of the digital AHB converter, covering the firmware architecture, control algorithms, state machine flow, configuration settings, and diagnostic features. It aims to serve as a technical reference for developers implementing or customizing digital AHB control solutions for high-efficiency DC-DC power systems.

2 Power Stage Overview

The Asymmetrical Half-Bridge (AHB) hardware prototype, derived from the PMP41140 reference design, has been developed using the C2000™ Digital Power SDK. This platform serves as a validation environment for the proposed fast ZVS calculation loop and adaptive soft-start control. The complete electrical specifications of this prototype, including input/output ratings and key parameters, are summarized in Table 2-1 below.

Table 2-1. Power Stage Specifications

Parameter	Specifications
Input Voltage	120-420V
Output voltage	9V, 15V, 20V, 28V
Output Current	15A, 12A, 9A
Output Power	150-250W
Switching Frequency	20KHz - 200KHz

As shown in Figure 2-1, the AHB converter employs LMG2650 GaN drivers to control the high-side and low-side switches efficiently, ensuring low-loss switching through the resonant tank formed by L_r , C_r , and the transformer magnetizing inductance (L_m). The auxiliary transformer winding is utilized exclusively for ZVS detection, enabling accurate zero-voltage transition identification under varying load and input conditions.

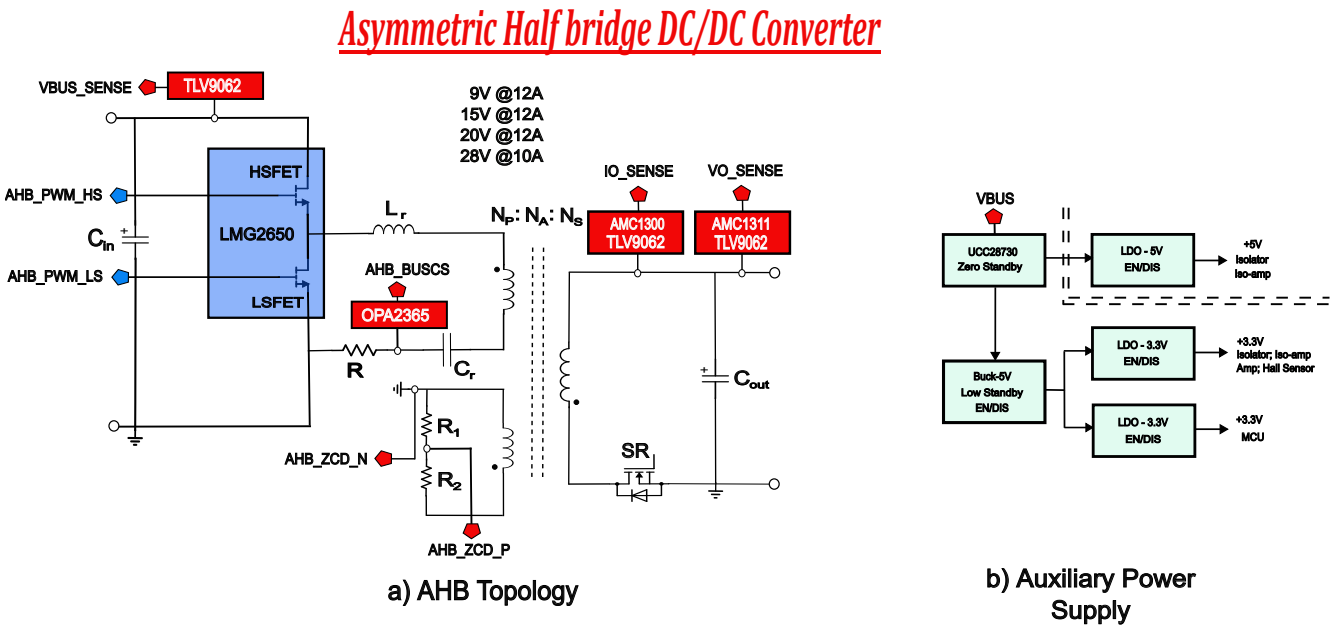


Figure 2-1. PMP41140 High Level Schematic View

Signal Conditioning and Feedback Network

Feedback and sensing circuits use precision amplifiers to ensure signal integrity, isolation and accurate real time measurements:

- AMC1300 / AMC1311 – Isolated amplifiers for primary-side current and voltage sensing, enabling high common mode rejection
- TLV9062 – Operational amplifier for voltage feedback scaling and buffering
- OPA2365 – High-bandwidth amplifier for resonant signal amplification and phase detection

These analog front-end components provide the C2000™ ADC subsystem with accurate, noise-free inputs for control computation and diagnostics.

Auxiliary Power Supply

The auxiliary power supply section generates regulated +12 V, +5 V, and +3.3 V rails using UCC28730 and low-power buck converters. These rails provide stable biasing for analog, digital, and gate-driver sections, ensuring proper startup sequencing and fault immunity.

Digital control implementation

The digital control platform is based on the C2000™F28P55x microcontroller, which serves as the central control unit for the AHB converter. Figure 2-2, shows the high-level signal interaction between the AHB hardware and the control logic.

Core peripherals utilized include:

- ADC – for real-time sampling of voltage and current feedbacks (VBUS, VO, and resonant current)
- CMPSS – for fast protection response and ZVS event detection
- CLB – for implementing custom digital logic and timing-critical functions
- PWM – for precise generation of high-resolution gate drive signals

These peripherals operate in a tightly synchronized manner to execute adaptive on-time computation, ZVS boundary detection, and smooth soft-start sequencing. The flexible real-time control capability of the C2000™ MCU enables dynamic adjustment of the switching behavior based on operating conditions, ensuring optimized efficiency and stability.

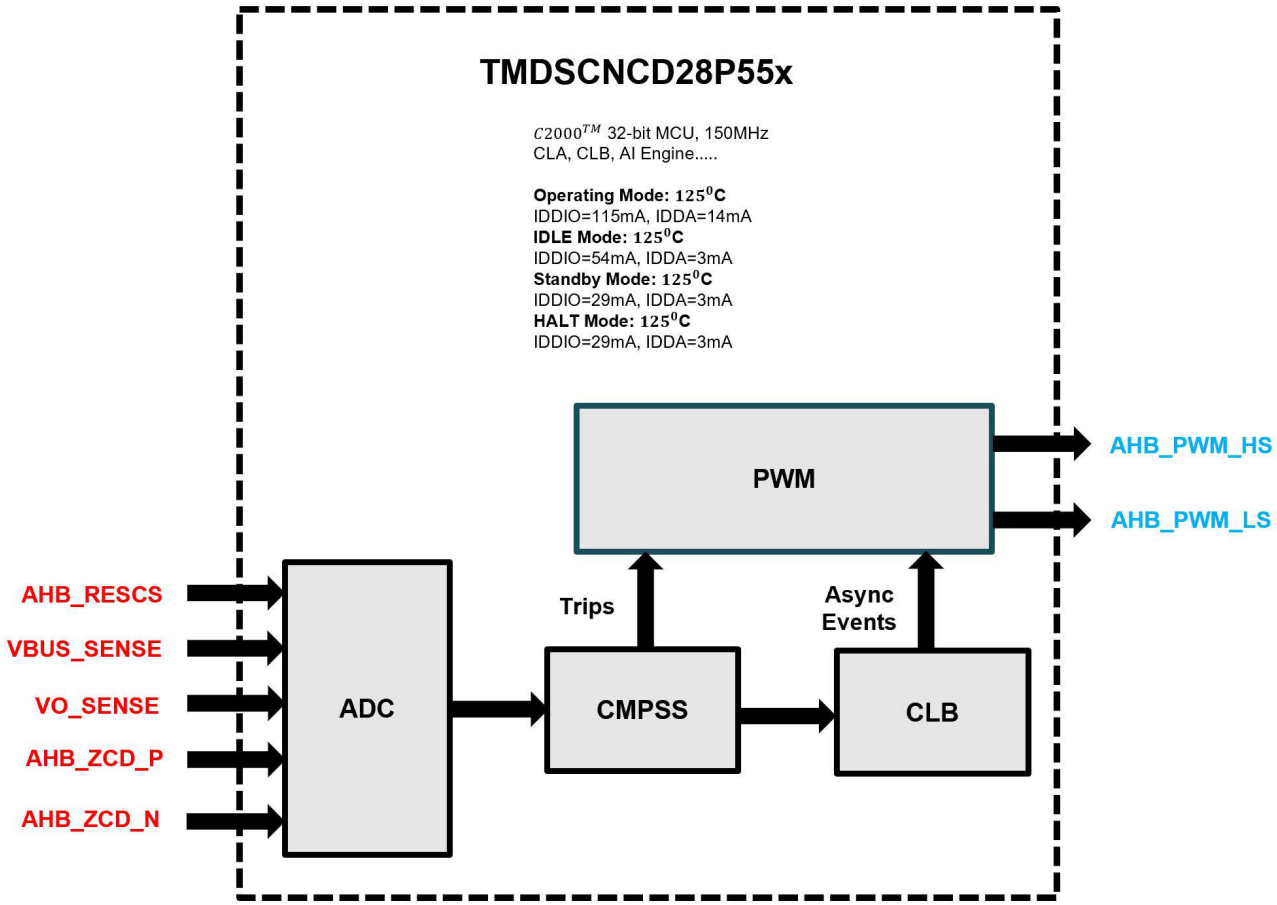


Figure 2-2. Control implementation with C2000™ MCU

A high-level overview of the digital control architecture is presented in figure, illustrating the signal flow from the AHB hardware to the control logic. Each peripheral's specific functionality and timing coordination will be discussed in later sections of this documentation.

3 Software Overview

Find related software information at the following link: C2000WARE-DIGITAL-POWER-SDK

Opening the CCS project:

The software of this design is available inside C2000Ware_DigitalPower_SDK and is supported with peripheral SYSCFG framework for initialization. To open the project:

1. Install CCS (version 12.5 or above)
2. Install C2000Ware DigitalPower SDK from the tool page
3. Open CCS, and create a new workspace
4. Inside CCS, go to View → Resource Explorer. Under Resource Explorer, go to Software → C2000Ware DigitalPower SDK - <version> → solutions and select this solution; that is, PMP41140, and click import project. The code is available for F28P55x device.

Note

CCS can recommend installing a particular version of the compiler relevant to the imported project. If requested, find the compiler on ti.com to download and install. Configure the compiler version in the project properties menu after installing the compiler. Make sure the CCS project tool discovery path includes the path of your compiler installation.

3.1 Software Architecture

Once the PMP41140 project is successfully imported into Code Composer Studio (CCS), the Project Explorer displays the complete directory structure as shown in Figure 3-1.

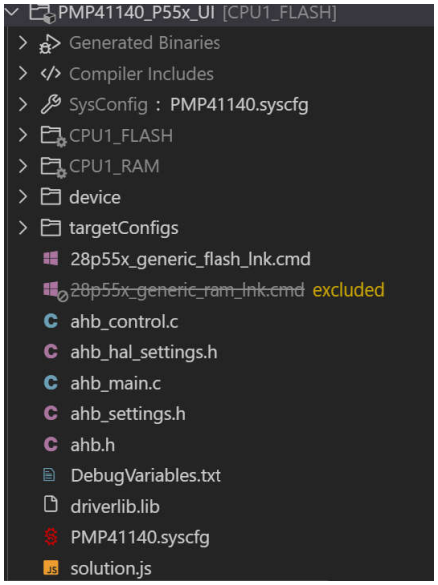


Figure 3-1. Project structure Overview

The project leverages SysConfig for device and peripheral initialization on the C2000™F28P55x family of microcontrollers. For this design, the specific device used is TMS320F28P550SGE-64PM, which provides the computational performance and peripheral set required for high-frequency digital control of an Asymmetric Half-Bridge (AHB) DC-DC Converter.

The following sections in the Table 3-1 describe the structure and functionality of the key files included in the project.

Table 3-1. Supporting Source and Header File

File	Description
ahb_main.c	Entry point of the firmware. This file manages system initialization, startup sequence, and execution of the main control loop. It also monitors lab-specific fault conditions and triggers appropriate fault diagnostics and recovery mechanisms.
PMP41140.syscfg	Configuration file generated by TI SysConfig tool. Defines clock setup, GPIO assignments, peripheral mapping, interrupt sources, and other low-level initialization parameters. Any change in system configuration should be performed through this file to maintain consistency and prevent register conflicts.
ahb_control.c	Core control file containing interrupt service routines (ISRs) and real-time digital control algorithms. This includes voltage and current feedback processing, PWM duty cycle updates, and control loop execution for AHB converter regulation.
ahb.h	Central header file that includes all device-specific and project-level declarations. Defines global variables, macros, and data structures used across control and diagnostic modules. It also declares prototypes for all ISRs.
ahb_settings.h	Configuration header for user-adjustable parameters. Includes definitions for control constants, protection thresholds, and scaling factors. This file enables quick tuning of system parameters without altering the core algorithm.
ahb_hal_settings.h	Hardware abstraction layer header containing mapping of physical ADC channels, to logical control variables. It ensures hardware portability and simplifies firmware migration across similar devices or boards.

The PMP41140 firmware is organized in a modular manner to simplify development, debugging, and reusability. Each source module handles a specific subsystem:

- **Initialization Layer:** Managed through SysConfig and AHB_main.c, ensuring deterministic boot and peripheral setup.
- **Control Layer:** Implemented in AHB_control.c, responsible for executing the main closed-loop control algorithm in real time.
- **Hardware Abstraction Layer (HAL):** Implemented via AHB_hal_settings.h to maintain a clean separation between hardware configuration and algorithm logic.
- **User Configuration Layer:** Exposed via AHB_user_settings.h for flexible system tuning and lab evaluation.

This structured approach provides a clear separation between hardware configuration, real-time control, and application-specific tuning, enabling faster debugging, enhanced scalability, and easier porting to other C2000™ F28x devices.

3.1.1 Devcie Initialization

In this project, the TI SysConfig tool is utilized for the initialization and configuration of all device peripherals. The *board_init()* function, invoked within the "AHB_main.c" file, links the application software with the configuration parameters defined in the "c2000.syscfg" file.

The SysConfig file is device-specific to the F28P55x family and serves as the central repository for all peripheral and pin assignments. Through this configuration, both analog and control peripherals are initialized in a consistent and maintainable manner.

Specifically, the following modules are configured using SysConfig:

- **Analog peripherals**: ADC, ASYSCTL, CMPSS
- **Control peripherals**: ePWM, CLB

- **System peripherals**: GPIO, X-BAR, Interrupt, and CPU Timer

This approach ensures deterministic initialization at startup, reduces manual register configuration errors, and provides a graphical interface for easy modification of peripheral mappings. Detailed descriptions of each configured peripheral and its role in the control architecture are provided in the subsequent sections.

3.1.1.1 Analog Peripherals Initialization

The analog peripherals in this project are primarily responsible for interfacing the microcontroller with analog feedback signals from the application hardware. These peripherals capture real-time voltage and current information, enabling the digital control algorithm to make precise and deterministic decisions.

ASYSCTL Initialization:

The ASYSCTL (Analog System Control) peripheral is used to configure the analog reference (VREF) that serves as the voltage reference for the ADC conversions. Proper configuration of VREF ensures measurement consistency and accuracy across temperature and operating conditions.

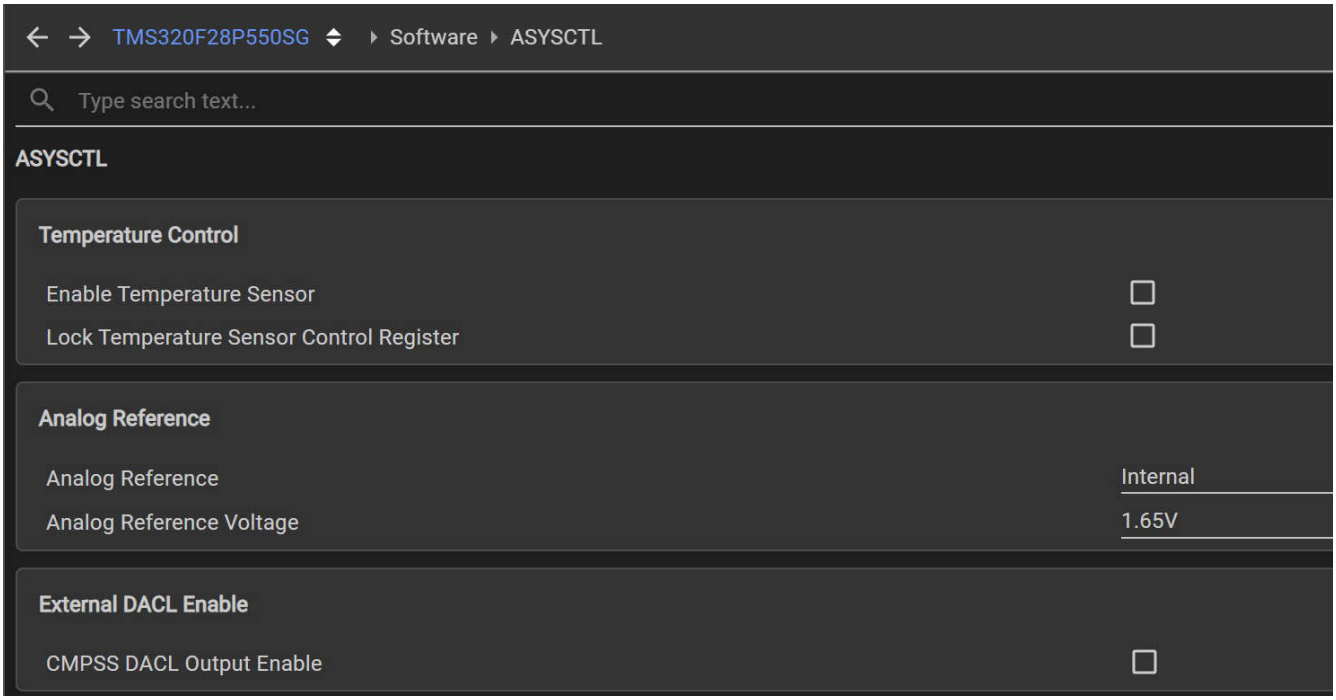


Figure 3-2. ASYSCTL Peripheral Initialization

As shown in Figure, the analog reference in this project is configured as internal, with an analog reference voltage of 1.65 V. The internal reference is further scaled by an internal gain factor of ×2, resulting in an effective ADC input range of 0 V to 3.3 V. This configuration allows the ADC to accurately sample full-scale analog signals corresponding to the converter's feedback range without requiring external reference circuitry.

ADC Initialization:

The Analog-to-Digital Converter (ADC) module is used to sample various analog feedback signals such as output voltage, input voltage, and inductor current. SysConfig allows selection of the analog input channels, configuration of the sampling window, conversion speed, trigger source, and ADC clock frequency. These

parameters determine the sampling accuracy and synchronization of the feedback signals with the PWM switching cycle.

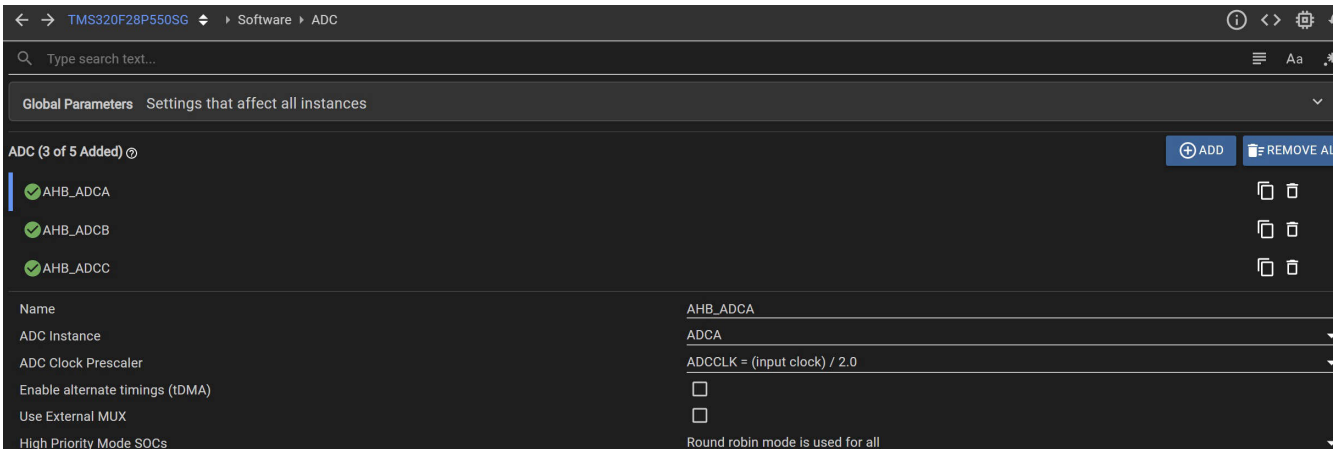


Figure 3-3. ADC Clock Initialization

In this project, three ADC modules—AHB_ADCA, AHB_ADCB, and AHB_ADCC—are utilized to acquire analog feedback signals from the power stage. All ADCs are configured to operate with an ADC clock frequency of 75 MHz, achieved by setting the appropriate ADC clock prescaler in SysConfig of each ADC, as shown in figure.

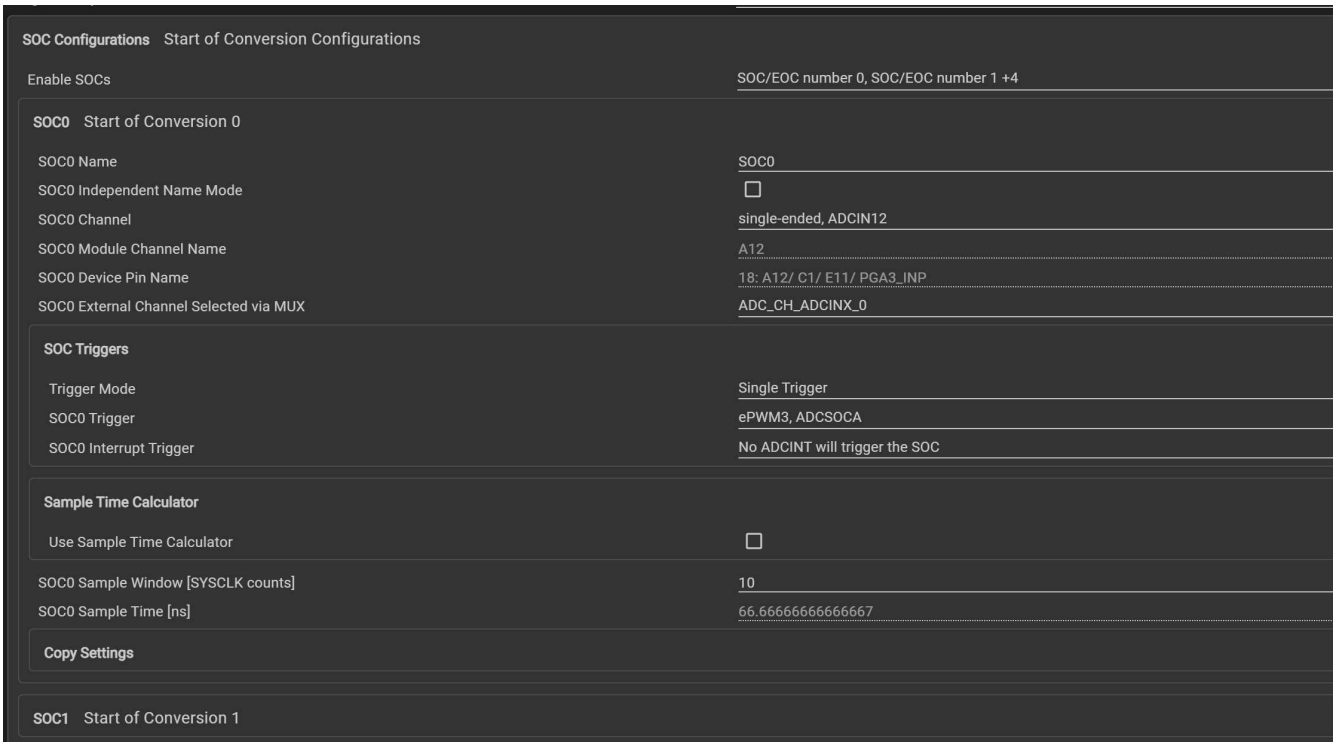


Figure 3-4. ADC SOC Initialization

Each ADC module contains 16 Start-of-Conversion (SOC) units, labeled SOC0 through SOC15. The firmware uses the default round-robin priority scheme, where SOC0 has the highest conversion priority and SOC15 the lowest. This scheduling ensures deterministic sampling across multiple feedback channels while maintaining balanced conversion timing.

Each SOC can be individually configured for:

- Input Channel Selection: Specifies which analog input pin (A0–A15, B0–B15, etc.) is connected to the ADC input multiplexer.

- Trigger Source: Determines the event that initiates the conversion (for example, an ePWM trigger).
- Sample-and-Hold Window: Defines the acquisition period for stable signal sampling before conversion begins.

As shown in figure, one example configuration is ADCA-SOC0, which uses analog channel A12 as the input, with ePWM3.ADCSOCA as the trigger source and a sample window of 10 cycles. This setup synchronizes ADC sampling with the PWM switching event to ensure consistent timing between control and measurement.

Priority	ADC-A	ADC-B	ADC-C
High Priority Frequency with PWM (20KHz - 200KHz)	AHB_ZCD_N (A12) SOC0 -> ADC_TRIGGER_EPWM3_SOCA SOC1 -> ADC_TRIGGER_EPWM3_SOCA		
	AHB_RESCS (A1) SOC3 -> ADC_TRIGGER_EPWM3_SOCA		
	VO_SENSE (A3) SOC2 -> ADC_TRIGGER_EPWM3_SOCA SOC4 -> ADC_TRIGGER_EPWM3_SOCA SOC5 -> ADC_TRIGGER_EPWM3_SOCA		
low Priority Frequency with Timer (50KHz)		VBUS_SENSE (B15) SOC2 -> ADC_TRIGGER_CPU1_TINT1 SOC3 -> ADC_TRIGGER_CPU1_TINT1 SOC4 -> ADC_TRIGGER_CPU1_TINT1	AHB_TEMP (C14) SOC5 -> ADC_TRIGGER_CPU1_TINT1 SOC6 -> ADC_TRIGGER_CPU1_TINT1

Figure 3-5. Analog Signals Mapping with ADC

The mapping of analog pins to ADC channels, the associated SOC configurations, and their respective trigger sources are illustrated in figure. This mapping defines how the system captures key feedback variables such as output voltage, input voltage, and inductor current for real-time control functions.

CMPSS Initialization:

The CMPSS (Comparator Subsystem) peripheral is utilized for hardware-based fault monitoring and protection. Each CMPSS module can generate a trip signal when the sensed analog input exceeds or falls below a programmed threshold. These trip events are commonly used to initiate immediate protective actions—such as disabling PWM outputs or triggering fault interrupts—to safeguard the system during overcurrent, overvoltage, or other abnormal operating conditions.

Module	Comparator	Analog Pins	Default DAC Value	Description
CMPSS1	High	AHB_RESCS (A1)	3000	Inductor Current Peak Trip
	Low	AHB_RESCS (A1)	3120	Over Current Trip
CMPSS2	High	AHB_ZCD_N (A12)	AHB_ZCD_P (A10)	Zero Crossing Detection
	Low	AHB_ZCD_P (A10)	1490	ZVS Diagnostics
CMPSS3	High	VO_SENSE (A3)	4040	Over Voltage Trip

Figure 3-6. Analog Signal Mapping with CMPSS

In this project, three CMPSS (Comparator Subsystem) modules are utilized for protection, sensing, and control diagnostics as shown in figure. Each CMPSS module provides both high and low comparators with independent DAC references and flexible input configurations.

1. CMPSS1:
   - Both high and low comparators receive the same analog input from AHB_RESCS.

- The high comparator is configured for peak current detection. Its DAC reference value is dynamically updated in every ISR2 based on the controller output, enabling adaptive current control.
- The low comparator serves as an overcurrent protection mechanism. Its DAC reference is pre-calculated based on the defined current limit, providing a hardware-level fault response.

2. CMPSS2:

- The high comparator is configured for zero-crossing detection (ZCD) and operates in differential mode with inputs AHB_ZCD_P (A10) and AHB_ZCD_N (A12). This allows precise detection of zero-current or zero-voltage points in resonant or soft-switching operation.
- The low comparator is configured for ZVS (Zero Voltage Switching) diagnostics, using a single-ended input from AHB_ZCD_P (A10). The DAC shadow value is updated every PWM cycle to monitor switching conditions dynamically.

3. CMPSS3:

- The high comparator is used for overvoltage protection (OVP). The analog input VO_Sense (A3) is compared against a DAC threshold that corresponds to the configured voltage limit.

Both the overcurrent and overvoltage protection trips are routed through the EPWM X-BAR, generating one-shot trip signals that immediately force all power switches into a safe OFF state.

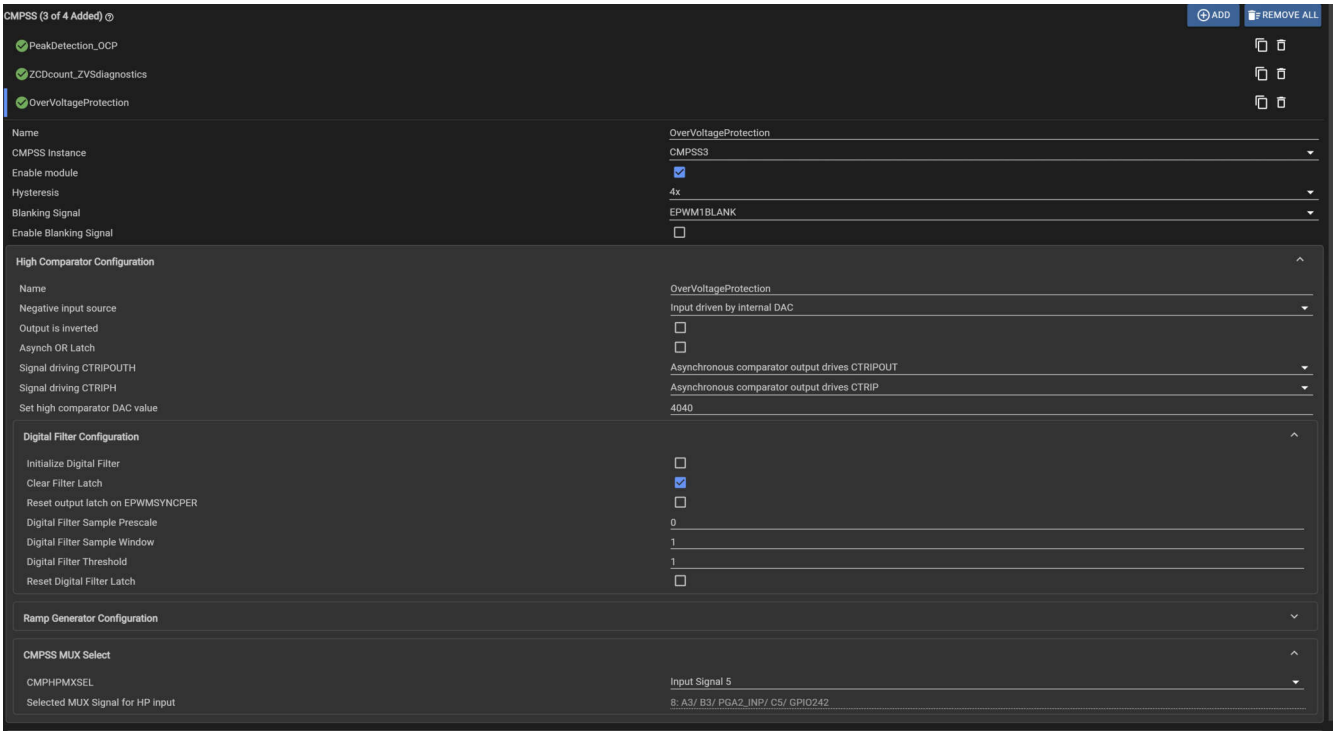


Figure 3-7. CMPSS Configuration For Single Ended

A sample configuration of the CMPSS3 High Comparator is shown in the figure below.

- To begin, the CMPSS3 instance must be enabled by selecting the “Enable Module” option in SysConfig. The comparator hysteresis is configured to ×4, providing improved noise immunity at the input.
- For the High Comparator, the negative input source is selected as “Input driven by internal DAC”, which configures the comparator in single-ended mode. (For differential operation, the negative input source would be set to “Input driven by external pin.”) The comparator output, CTRIPOUTH/CTRIPH, is configured as asynchronous, ensuring immediate fault response without waiting for the system clock. The DAC reference value for overvoltage comparison is configured as 4040, corresponding to the desired overvoltage trip threshold derived from system voltage scaling.
- Finally, the comparator input pin is selected using the CMPHMXSEL register. In this case, Input Signal 5 is chosen, which maps to analog input A3. (The mapping of CMPSS input signals to analog pins can be found in the device TRM.)

In summary, the analog peripherals collectively provide the essential sensing and protection framework that bridges the power stage and the digital control core, ensuring stable, safe, and efficient converter operation.

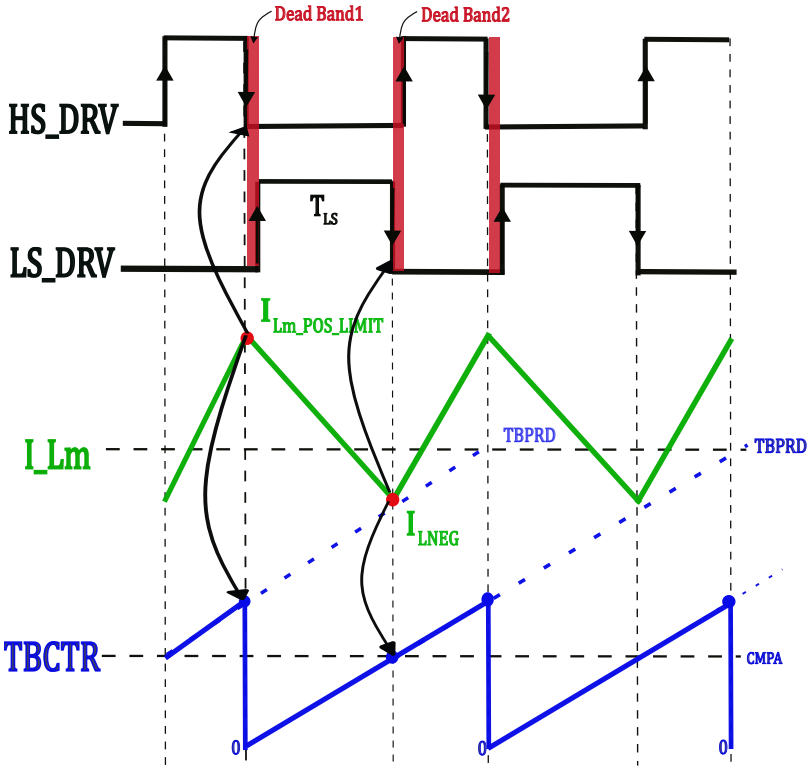
3.1.1.2 Control Peripherals Intialization

Load specific PWM Implementation:

In every PWM switching cycle, the load condition is continuously monitored, and based on predefined load threshold levels, the PWM configuration for the subsequent switching cycle is dynamically updated. This real-time adaptation enables the modulation strategy to respond effectively to varying load conditions, maintaining high efficiency, reliable ZVS (Zero Voltage Switching) operation, and stable converter performance across both light and heavy load scenarios.

Heavy-Load PWM Waveforms

As shown in figure, under heavy-load conditions, the PWM signals are configured as complementary waveforms with appropriately inserted dead-bands. The time-base period (TBPRD) is programmed to a large value, but due to a synchronization event triggered by the peak-trip signal, the time-base counter (TBCTR) is forced to restart before reaching TBPRD, effectively controlling the PWM switching period dynamically. During each switching cycle, the compare register (CMPA) is updated based on the T_{LS} calculation performed in the control loop.



$$T_{LS} = [L_m * (I_{Lm\_POS\_LIMIT} - I_{LNEG})] / (N * V_o)$$

Figure 3-8. Heavy Load PWM Waveforms

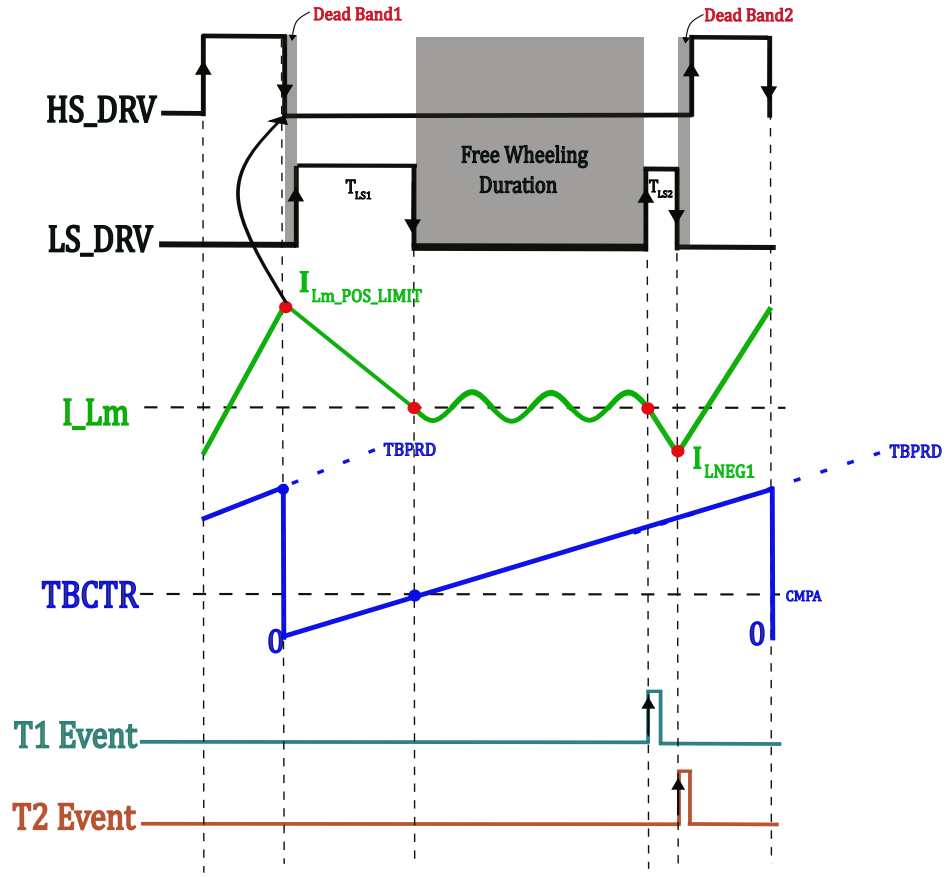
Table 3-2. Heavy Load PWM Events and Actions

PWM Events	HS_DRV	LS_DRV
TBCTR = 0	Falling Edge	Rising Edge
TBCTR = CMPA	Rising Edge	Falling Edge

As observed in table, PWM of HS_DRV and LS_DRV swicthes action is performed based on time base counter (TBCTR) of PWM module.

Light-Load PWM Waveforms

As shown in figure, under light-load conditions, the PWM signals are non-complementary. Similar to heavy-load operation, the TBPRD is configured with a large value, but the sync-in event caused by the peak-trip signal restarts the TBCTR counter before it reaches the programmed period, allowing variable on-time operation. In each switching cycle, CMPA is updated based on T_{LS1} calculations, and a freewheeling interval is introduced to minimize switching losses. During this phase, the system monitors the zero-crossing (ZCD) event, counting transitions from positive to negative current. Once the target zero-crossing count is achieved, a T1 event is generated, followed by a T2 event, which is a delayed version of T1 determined by the T_{LS2} timing parameter.



$$T_{LS\_1} = [L_m * (I_{Lm\_POS\_LIMIT} - 0)] / (N * V_o)$$

$$T_{LS\_2} = [L_m * (0 - I_{LNEG}1)] / (N * V_o)$$

Figure 3-9. Light Load PWM Waveforms

Table 3-3. Light Load PWM Events and Actions

PWM Events	HS_DRV	LS_DRV
TBCTR = 0	Falling Edge	Rising Edge
TBCTR = CMPA	No Change	Falling Edge
T1 Event	No Change	Rising Edge
T2 Event	Rising Edge	Falling Edge

These load-specific PWM waveforms are implemented through a coordinated use of the EPWM and CLB (Configurable Logic Block) peripherals. The EPWM provides precise time-base control and synchronization, while the CLB enables custom logic to generate adaptive timing events such as T1, T2, and zero-crossing

detection. The detailed configuration and functional interaction of these peripherals will be discussed in the following sections.

3.1.1.2.1 EPWM Initialization

The AHB_EPWM module serves as the primary control block for driving the power-stage switches — the High-Side FET (HSFET) and Low-Side FET (LSFET). In this project, EPWM3 is designated as the AHB_EPWM module, with GPIO4 assigned to EPWM3A (controlling the HSFET) and GPIO5 assigned to EPWM3B (controlling the LSFET).

Time-Base Submodule -

- The PWM clock prescalers are configured such that the time-base clock (TBCLK) equals the device system clock (i.e., TBCLK = SYSCLK/1). Both the time-base clock divider and high-speed clock divider are set to 1.
- The counter mode is configured as up-count, with an initial time-base period (TBPRD) of 4000, which will be dynamically updated inside the ISR during runtime based on the converter's operating conditions.
- PWM synchronization is achieved through the EPWM synchronization pulse triggered by a peak-trip event via Input XBAR5. Phase loading is enabled with a phase shift value of zero, ensuring alignment of the PWM counter during synchronization.

The counter-compare registers (CMPA/CMPB) are configured post-initialization and are shadow-loaded on the TBCTR = 0 event for deterministic updates.

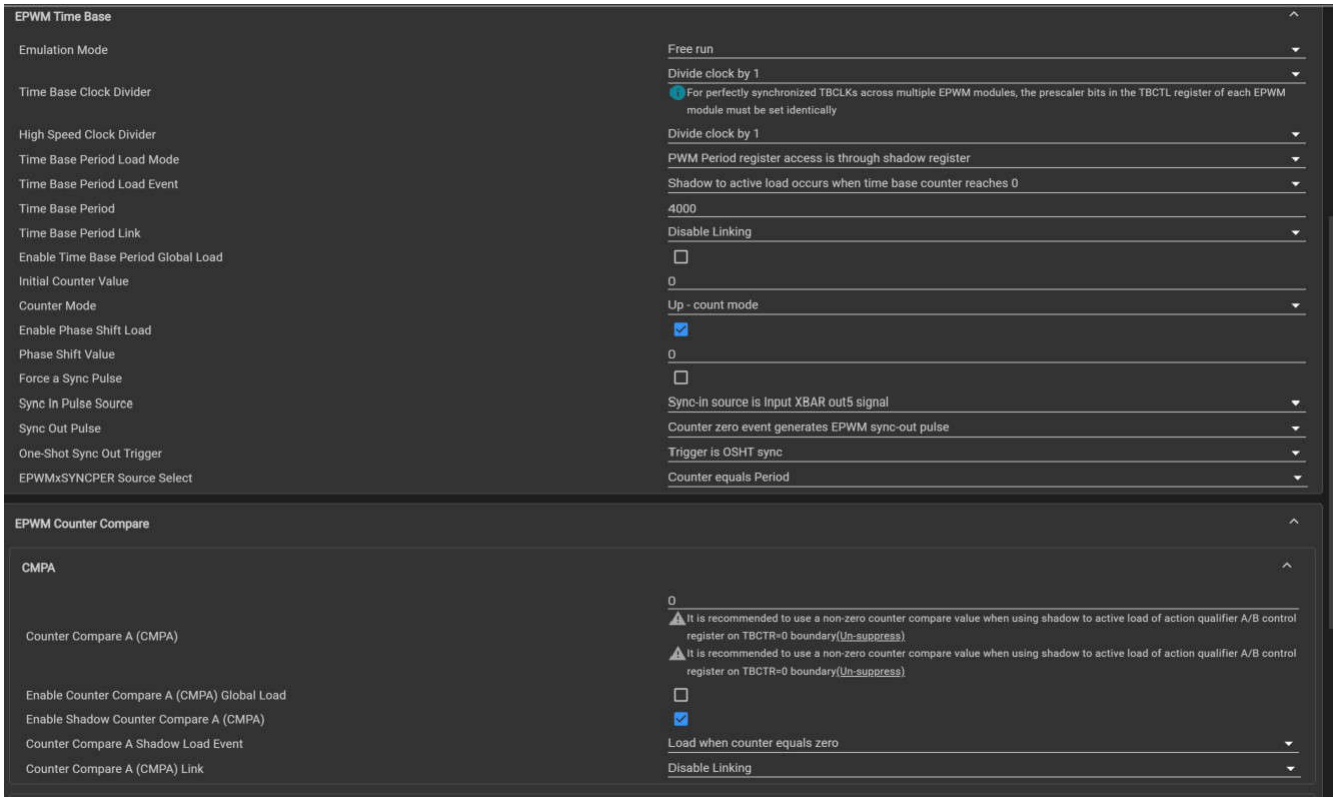


Figure 3-10. EPWM TimeBase and CounterCompare Submodule

Action Qualifier Submodule -

- In this configuration, Trigger 1 (T1) and Trigger 2 (T2) sources are mapped to DCAEVT2 and DCBEVT2, respectively.
- The EPWM A/B outputs are shadow-loaded either on synchronization or when the counter equals zero or the period, as shown in the reference figure.

- Initially, the action qualifier (AQ) events are configured such that both PWM outputs remain in the LOW state. During runtime, inside the ISR, the AQ actions are dynamically updated based on voltage threshold levels and load conditions of the AHB converter.

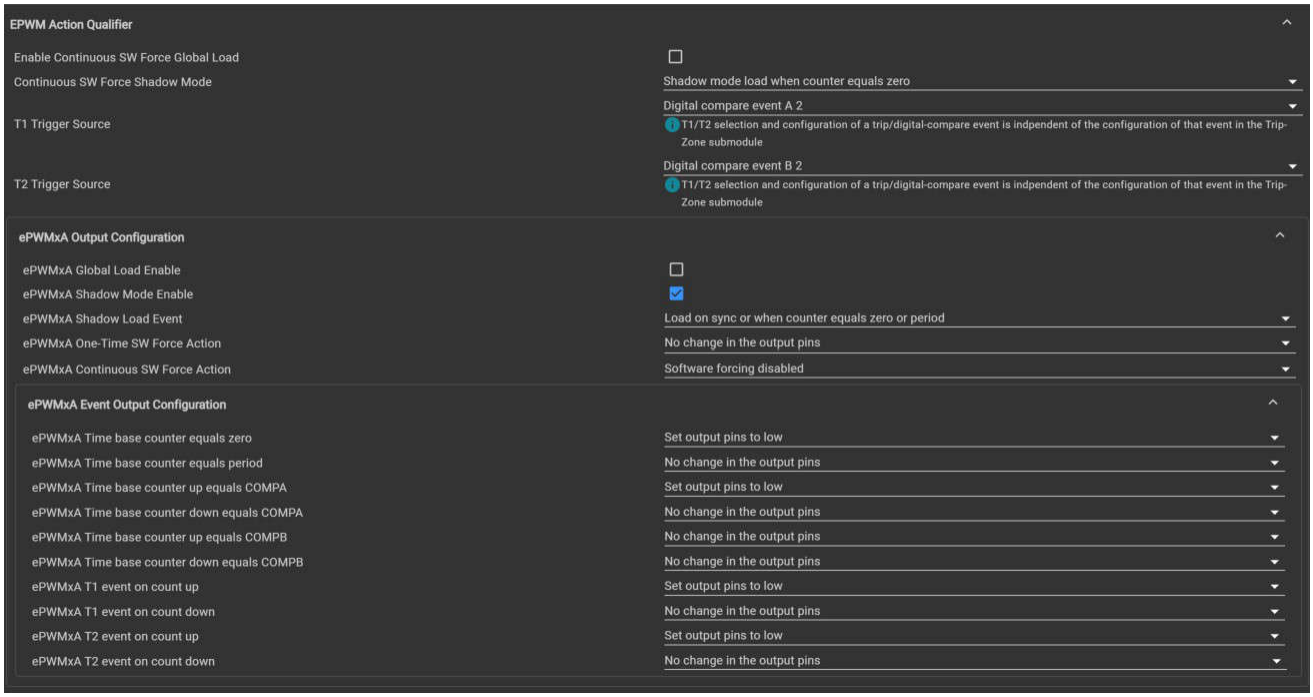


Figure 3-11. Action Qualifier Submodule

Dead-Band Submodule -

- The Dead-Band Generator is configured as shown in the figure, with rising-edge delay (RED) and falling-edge delay (FED) inputs driven by EPWM A and EPWM B, respectively. Both RED and FED polarities are non-inverted during initialization.
- However, following board initialization in main.c, the FED polarity is inverted to ensure correct complementary drive behavior. This inversion requires corresponding adjustments in the AQ submodule, where events intended to force a LOW output must be programmed as HIGH due to the inversion.

- Both RED and FED are configured in shadow mode, with load events occurring at TBCTR = 0, and their delay values are set to 66 and 22, respectively.

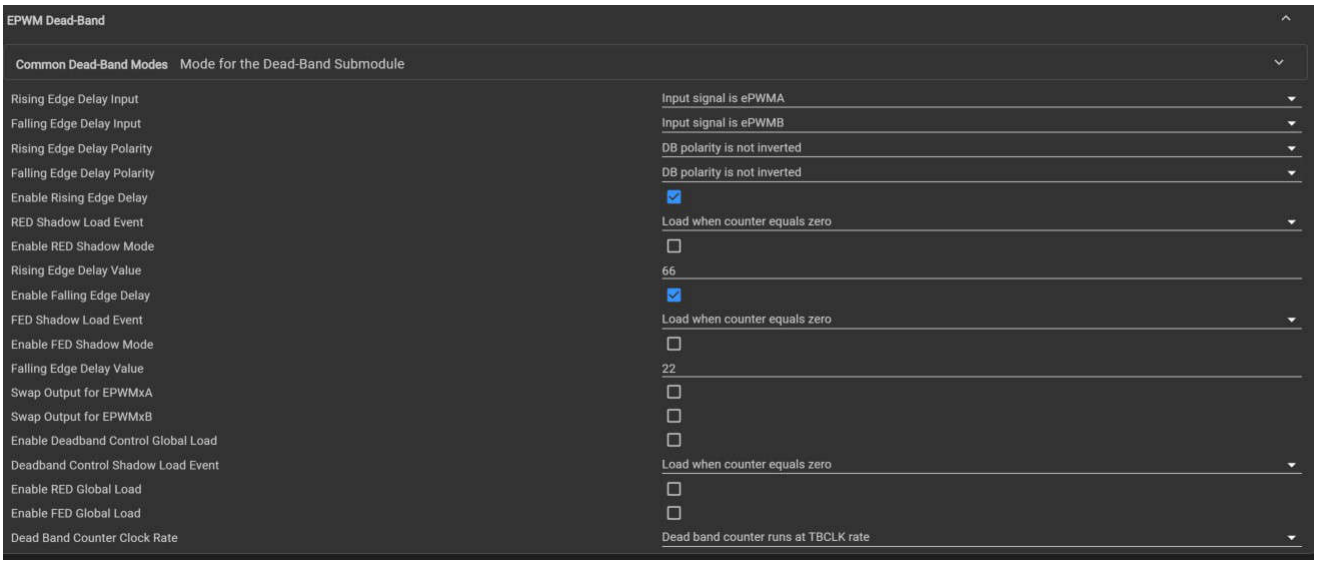


Figure 3-12. DeadBand Submodule

Trip-Zone Submodule -

- The Trip-Zone (TZ) submodule is configured to provide one-shot protection in the event of an overcurrent or overvoltage fault.
- The one-shot source is selected as DCAEVT1 or DCBEVT1, with direct digital compare actions disabled. Upon triggering, both TZA and TZB events drive the respective PWM outputs to a LOW state, immediately turning off the switches to protect the hardware.

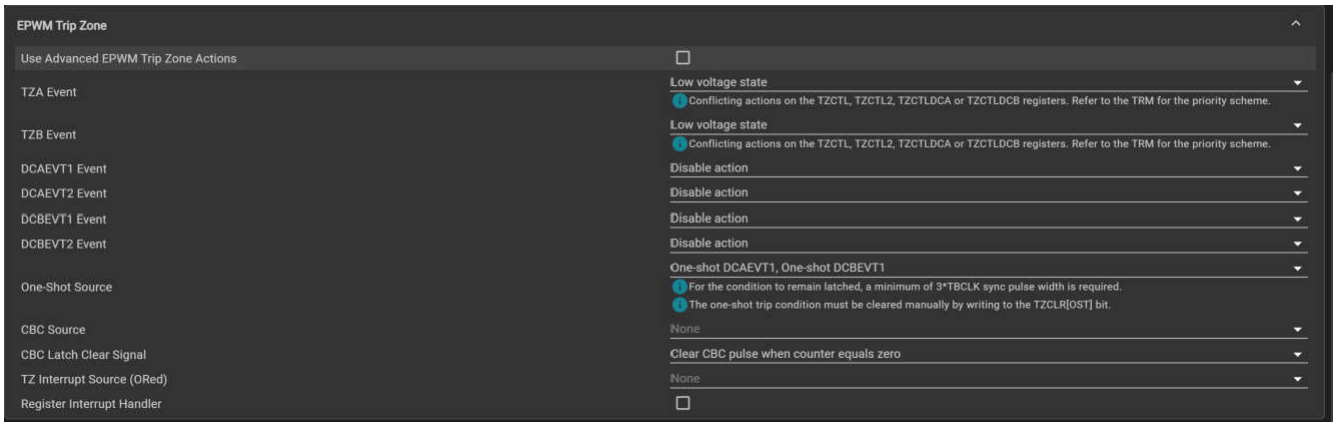


Figure 3-13. TripZone Submodule

Digital Compare Submodule -

- The Digital Compare (DC) submodule implements multiple event sources for trip and diagnostic conditions:
  - DCAEVT1 is generated by a logical combination of Trip8 and Trip9 from the EPWM XBAR.
  - DCAEVT2 is sourced from Trip10.
  - DCBEVT1 is derived from the combination of Trip8 and Trip9, similar to DCAEVT1.
  - DCBEVT2 is configured from Trip11 of the EPWM XBAR.

- Both DCA/BEVT1 and DCA/BEVT2 events are active when their source signals are logic high (DCxH/L = 1), ensuring immediate trip response during fault conditions.

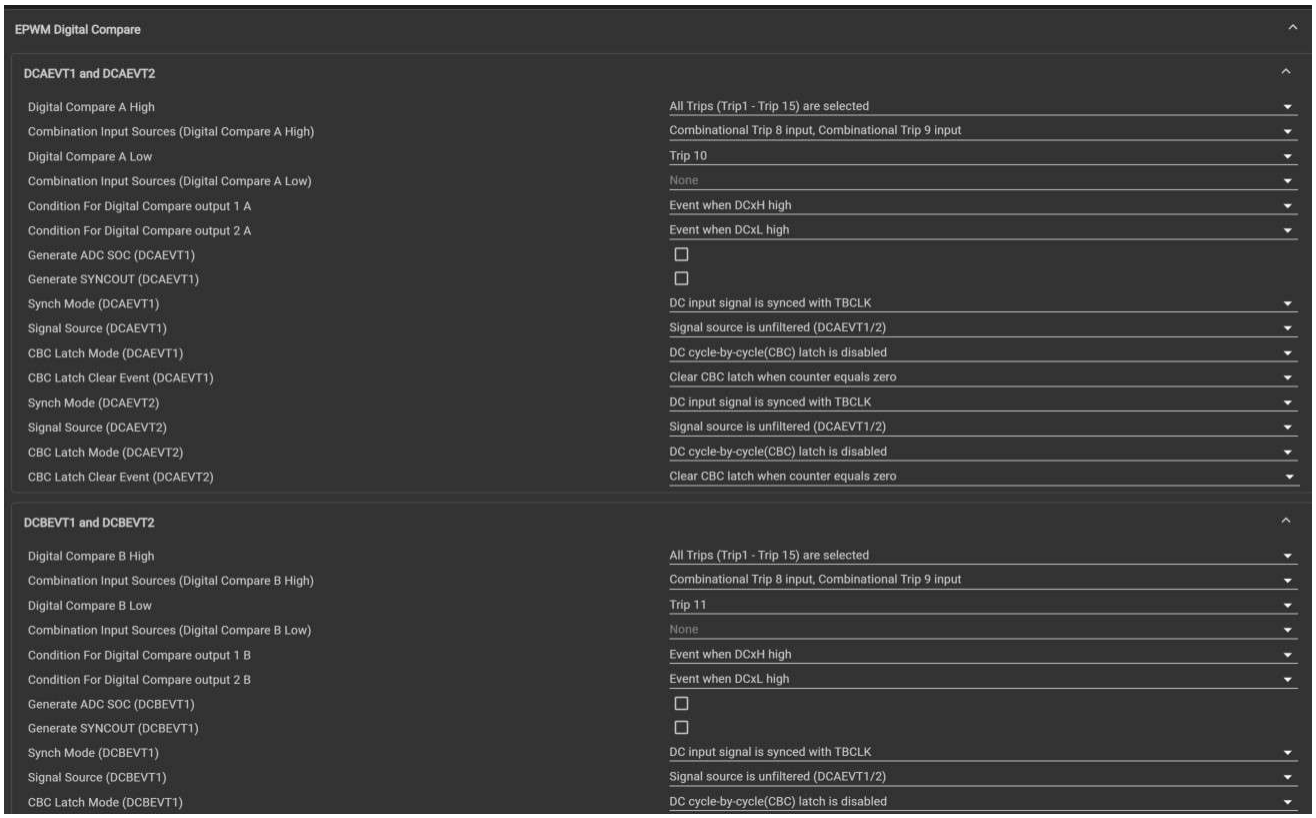


Figure 3-14. DigitalCompare Submodule

Event Trigger Submodule -

- The Event Trigger (ET) submodule is responsible for generating interrupts and ADC start-of-conversion (SOC) signals.
- An EPWM interrupt is enabled and configured to occur when the time-base counter equals zero (TBCTR = 0). This event also triggers the ADC SOC, synchronizing ADC sampling with the PWM switching cycle.
- The interrupt service routine is linked to the INT_AHB_EPWM_ISR (ISR1) handler for executing PWM update tasks.

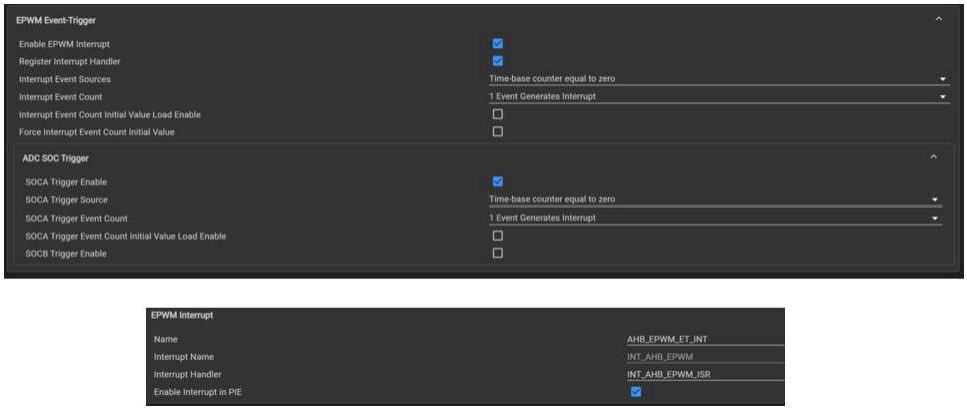


Figure 3-15. EventTrigger and Interrupt Submodule

In addition to the main AHB_EPWM module, two auxiliary EPWM modules — ZCD_EPWM and CMPSS_BLANK_EPWM — are utilized in this project to support signal conditioning and diagnostic

functionalities. Both of these modules share identical Time-Base submodule configurations, ensuring synchronization and uniform timing behavior across the system.

The ZCD_EPWM module is configured to assert its PWMA output high whenever DCAEVT1 becomes active. The DCAEVT1 event source is derived from EPWM XBAR Trip7, which typically indicates a zero-crossing detection (ZCD) event. This PWM pulse acts as a precise digital timing reference for the control algorithm, helping the firmware accurately detect and process zero-crossing transitions of the switching node or current waveform.

The CMPSS_BLANK_EPWM module is configured to generate a blanking window for the CMPSS comparators. This blanking signal masks unwanted transient signals or switching noise that may appear immediately after switching transitions, preventing false trips or spurious comparator triggering. By controlling the timing of the blanking pulse through the EPWM module, the comparator response can be finely tuned for both speed and reliability.

While these auxiliary EPWM modules are not mandatory for the converter to operate, they serve as valuable additions that improve system stability, enhance measurement reliability, and simplify the implementation of advanced diagnostics and protection mechanisms.

3.1.1.2.2 CLB Initialization

In this application, the Configurable Logic Block (CLB) modules are utilized to generate asynchronous events required for precise PWM control and diagnostics. Two CLB instances are employed AHB_CLB and ZVS_CLB each serving a distinct functional purpose within the system.

AHB_CLB

The AHB_CLB module is primarily responsible for generating the T1 and T2 events, which are crucial for light-load PWM waveform generation. These events define the timing boundaries within each PWM cycle, enabling controlled freewheeling intervals and adaptive switching behavior. This module uses the CLB1 instance configured on TILE0, as illustrated in Figure X. It utilizes seven input signals and implements three counter modules within the CLB fabric to synthesize precise timing events for T1 and T2 generation. The CLB outputs

are routed to both the EPWM X-BAR and other CLB modules, where they serve as triggers for corresponding actions such as PWM edge control or timing synchronization.

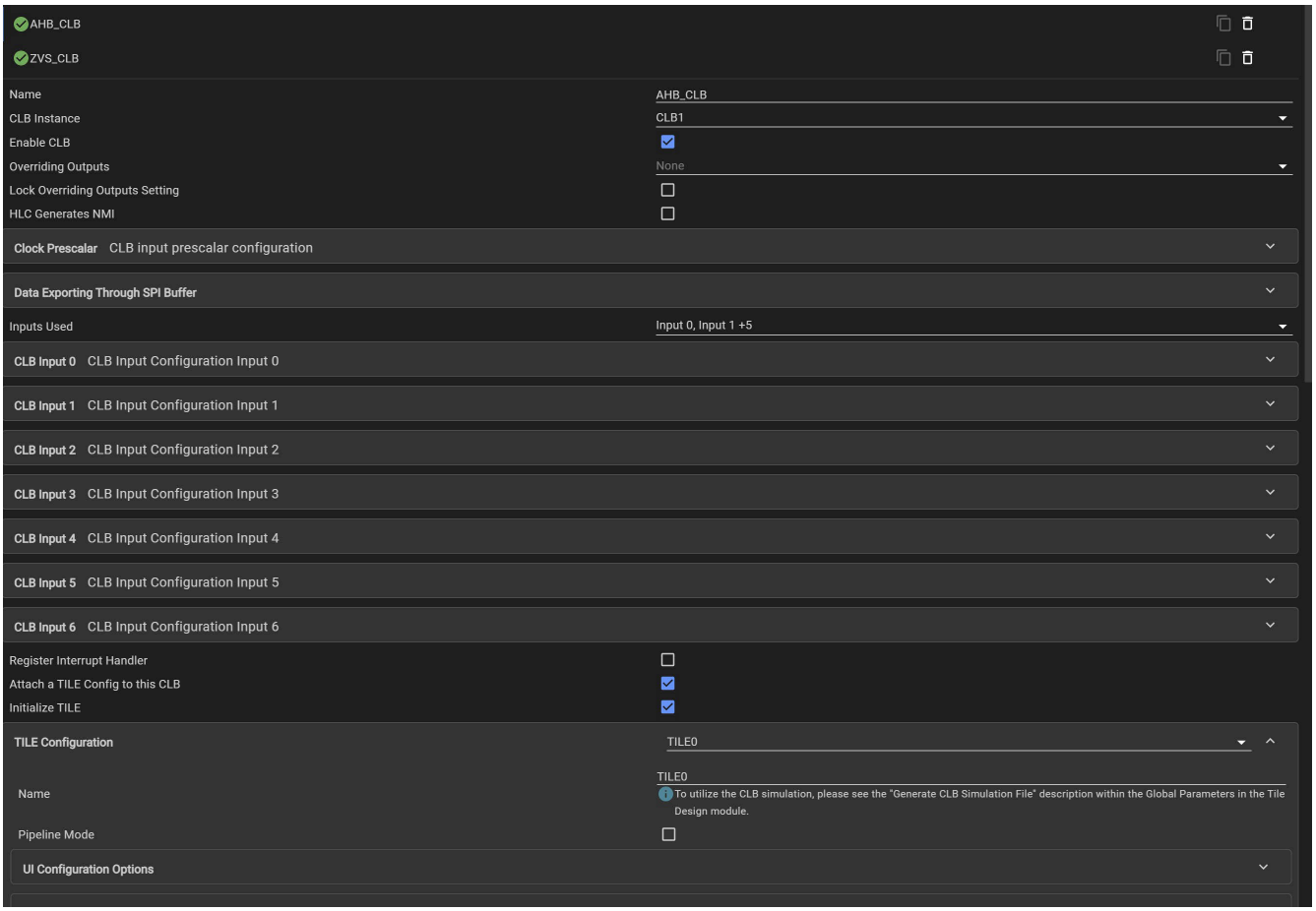


Figure 3-16. AHB_CLB Configuration

ZVS_CLB

The ZVS_CLB module is used to monitor ZVS operation and assist with diagnostic event generation. It ensures that PWM transitions occur under Zero Voltage Switching conditions and provides feedback to the control loop for corrective actions when deviations occur. This logic is implemented using the CLB2 instance, configured on TILE1, as shown in Figure Y. It utilizes six input signals and employs two look-up tables (LUTs) that act as programmable combinational logic blocks. These LUT outputs are further processed through counter modules,

and the final CLB outputs are routed via the CLBOUTPUT X-BAR and other CLB instances to create timing events and trigger diagnostics as needed.

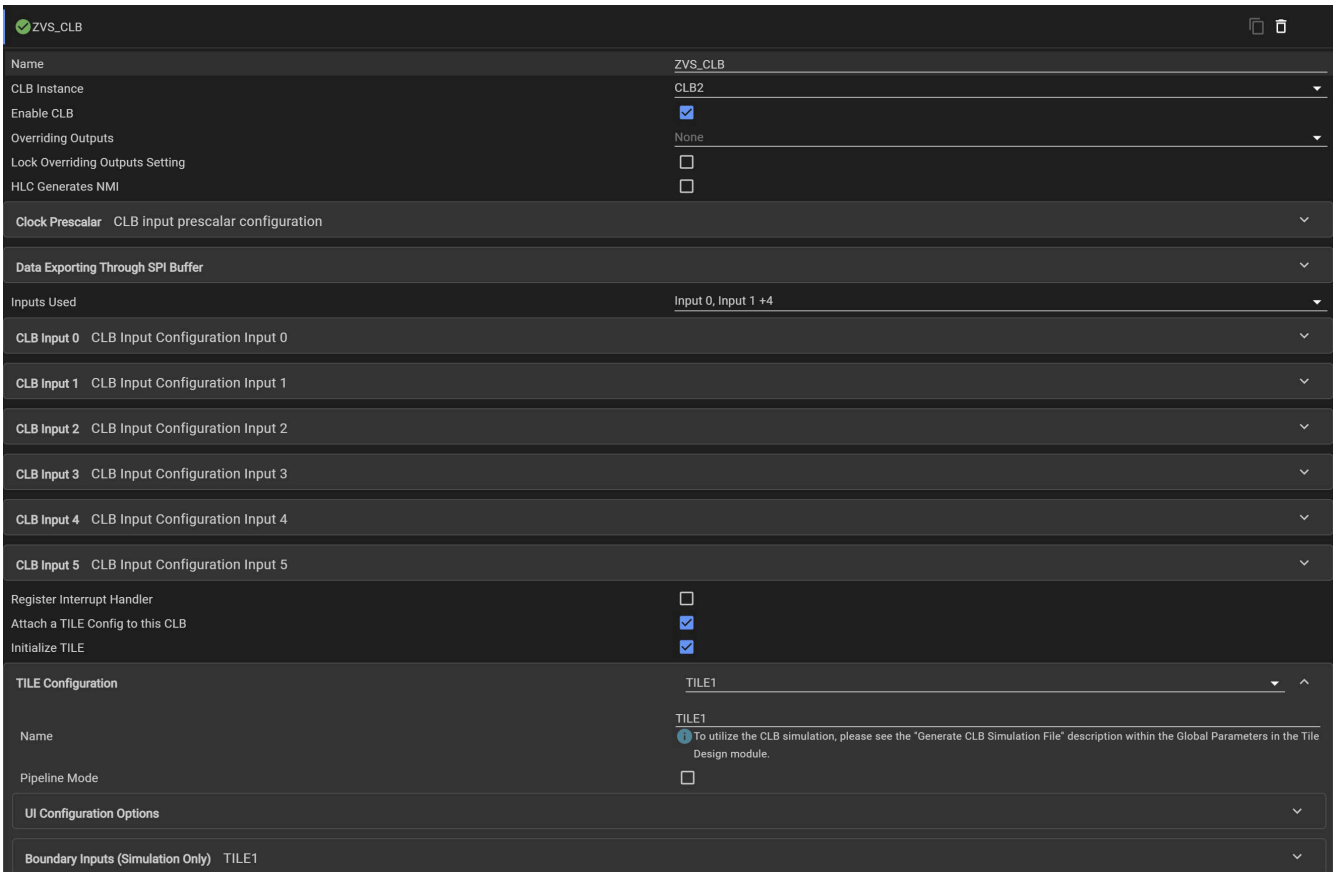


Figure 3-17. ZVS_CLB Configuration

The combined use of AHB_CLB and ZVS_CLB allows the firmware to flexibly generate and manage asynchronous events in hardware, significantly offloading the CPU. By integrating with the EPWM and X-BAR subsystems, these CLBs enable deterministic timing, low-latency event generation, and enhanced real-time responsiveness—critical for achieving high-efficiency, adaptive PWM control.

3.1.1.3 System Peripherals Initialization

System peripherals such as GPIOs, CPU Timers, Interrupts, and Crossbars (X-BARs) play a crucial role in system-level integration and observability. These peripherals act as the backbone for interconnecting modules, enabling firmware diagnostics, and managing real-time execution control.

In this application, they are used for various key purposes — to route internal events to external pins for debugging and monitoring, to connect one peripheral's output to another's input for functional coordination, to generate and link interrupts with their corresponding ISR routines in firmware, and to profile code execution time or measure performance of specific code sections. Together, these system peripherals provide the essential glue logic that binds the analog, control, and digital subsystems into a unified and responsive embedded control platform.

CPU Timer Initialization:

In this application, two CPU timers are utilized for distinct purposes: Timer1 for periodic interrupt generation and ProfilingTimer for execution-time measurement.

Timer1 is configured using the CPUTIMER1 instance with a timer period value of 3030. The corresponding interrupt is enabled and linked to the INT_Timer1_ISR service routine. Based on the configured parameters, the

timer overflow occurs at a frequency of 50 kHz, triggering the ISR at this rate to perform scheduled control or monitoring tasks.

The ProfilingTimer, on the other hand, is configured using the CPUTIMER2 instance. It operates with the SYSCLK as its clock source and a prescaler value of 1, providing the finest timing resolution. The timer period is set to its maximum value (65535), and the interrupt is disabled. This timer is exclusively used for profiling and benchmarking within the firmware, allowing precise measurement of code execution time for various user-defined sections.

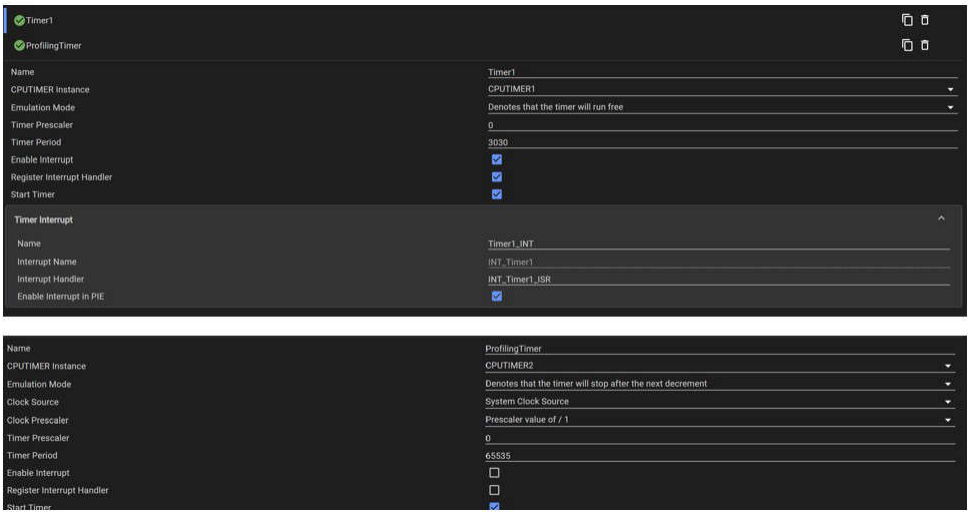


Figure 3-18. CPU Timer Configuration

Interrupt Initialization:

These interrupts are not configured separately under the interrupt configuration section. Instead, they are automatically configured when the corresponding interrupt is enabled within the Timer or PWM modules. This automation ensures proper interrupt mapping and linkage to their respective Interrupt Service Routines (ISRs) without requiring additional manual setup.

Ensure that the linked ISRs are correctly displayed and verified in the Interrupt Configuration view, as illustrated in the figure.

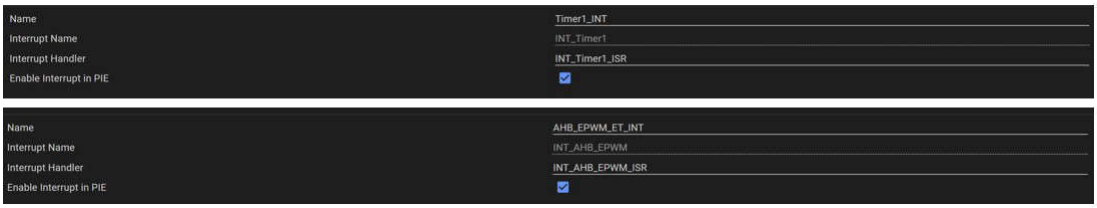


Figure 3-19. Interrupt Configuration

XBAR Initialization:

The Crossbar modules (X-BARs) serve as internal interconnects that link various on-chip peripherals and also facilitate signal routing to the external world.

- EPWMXBAR and CLBXBAR provide input connections to the EPWM and CLB peripherals respectively, enabling flexible event routing within the device.
- OUTPUTXBAR and CLBOUTPUTXBAR are used to route internal signals to external GPIO pins, allowing visibility for debugging and signal monitoring.

- INPUTXBAR enables routing of external GPIO inputs to internal peripherals for event triggering or control logic.

XBAR	Name	Source	Output
EPWMXBAR	ZCD Event	CMPSS2 TRIPH	Trip7
	PeakDetect	CMPSS1 TRIPH	Trip4
	T1 Event	CLB1 OUT4	Trip10
	T2 Event	CLB1 OUT5	Trip11
	OC_Protection	CMPSS1 TRIPL	Trip8
	OV_Protection	CMPSS3 TRIPH	Trip9
OUTPUTXBAR	PeakDetection_Outputxbar	CMPSS1 CTRIPOUTH	GPIO2
	ZeroCrossingDetect	CMPSS2 CTRIPOUTH	GPIO3
	ZVSDiagnosticsOutput	CLB2 OUT5	GPIO29
INPUTXBAR	Peakdetect_Inputxbar5	GPIO2	XBAR_INPUT5
CLBXBAR	PeakTrip_CLBXBAR	CMPSS1 CTRIPH	AUXSIG0
	ZCDTrip_CLBXBAR	CMPSS2 CTRIPH	AUXSIG1
	ZVS_Threshold	CMPSS2 CTRIPL	AUXSIG2
CLBOUTPUT XBAR	T1_Event	CLB1 OUT4	GPIO22
	T2_EVENT	CLB1 OUT5	GPIO23
	ZCD_Count_MatchEvent	CLB1 OUT3	GPIO10

Figure 3-20. XBAR Configuration

In this application, the XBARs are configured as illustrated in the figure, enabling signal-level coordination between EPWM, CLB, and external interfaces.

GPIO Initilaization:

As shown in the figure, one of the GPIOs is configured as an output signal from the application, which can be used as an input to the Synchronous Rectifier (SR) controller. This signal facilitates coordination between the main converter and the SR stage for efficient and synchronized operation.

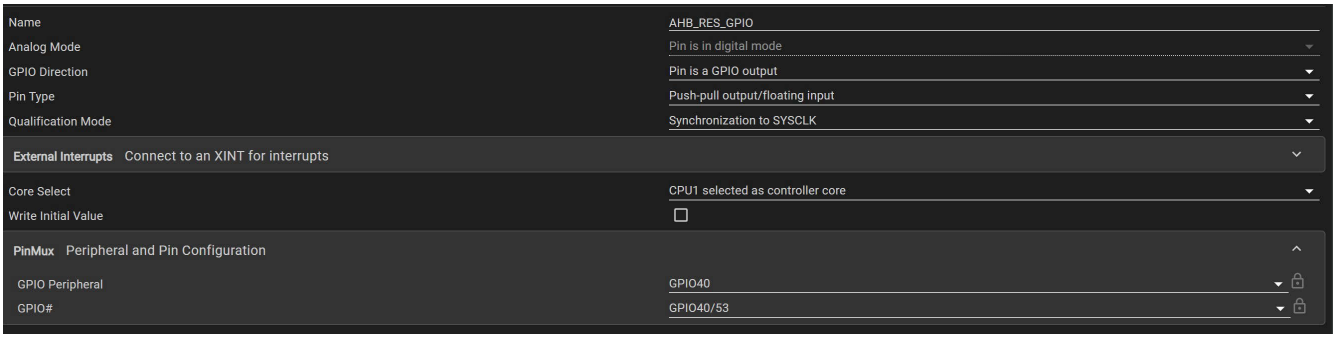


Figure 3-21. GPIO Configuration

3.1.2 Interrupt Structure

Interrupt Structure

This project operates using two primary Interrupt Service Routines (ISRs) that partition real-time control, PWM updates, and system maintenance tasks.

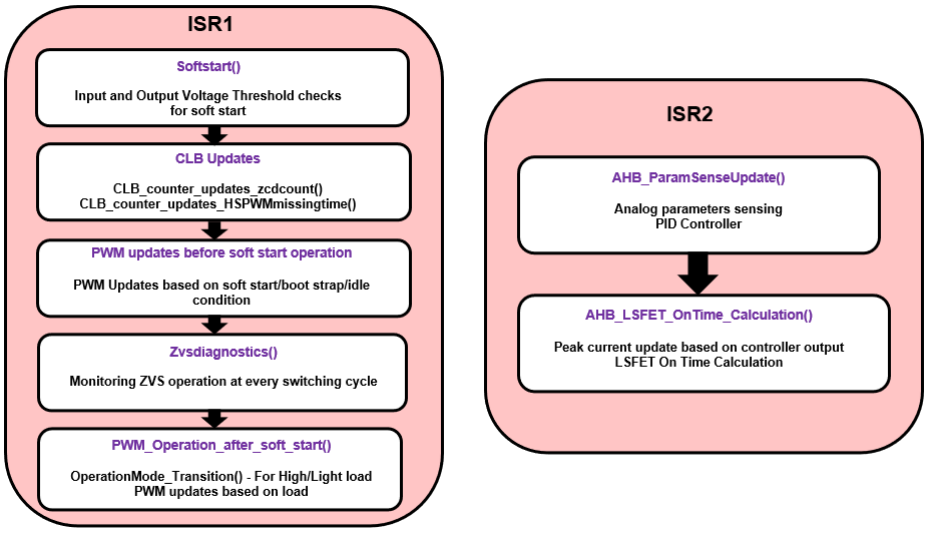


Figure 3-22. Interrupt Flow Diagram

As shown in Figure, this dual-interrupt structure enables deterministic PWM operation and stable closed-loop control at high switching frequencies.

1. ISR1 - Variable Frequency ISR
   - Function Name: INT_AHB_EPWM_ISR()
   - Trigger Source: AHB EPWM module when the Time-Base Counter (TBCTR) equals zero.
   - Operating Frequency: Variable, synchronized with the converter's switching frequency in the range of 20 kHz to 200 kHz.
   - Function:
     - Handles all PWM update activities and synchronization of complementary gate signals.
     - Ensures precise timing for high-side and low-side transitions to achieve Zero-Voltage Switching (ZVS) operation.
     - Updates modulation parameters derived from the latest control loop computation.
     - Executes with minimal latency to maintain consistent switching behavior across variable frequencies.
     - This ISR is time-critical and tightly optimized to guarantee deterministic PWM generation even at high operating frequencies.
2. ISR2 – Fixed-Frequency Control and Housekeeping ISR
   - Function Name: INT_Timer1_ISR()
   - Trigger Source: CPU Timer interrupt, generated by a timer overflow event.
   - Operating Frequency: Fixed at 50 kHz, independent of the converter's switching frequency.
   - Function:
     - Executes the main digital control loop, performing voltage and current feedback processing.
     - Applies digital filtering and running averages to measured signals to suppress switching noise.
     - Calculates switch on-times and duty cycles based on commanded voltage references.
     - Executes slew-rate limiting for smooth reference transitions and soft-start handling.
     - By maintaining a fixed execution rate, ISR2 provides a consistent computational window for control algorithms, while ISR1 adapts dynamically to the converter's switching frequency.

This interrupt hierarchy provides an efficient and scalable approach—where ISR1 handles fast, hardware-synchronized events, and ISR2 manages control computations and background tasks—ensuring precise timing, stable operation, and reduced CPU loading.

3.2 PowerSuite Usage

The PMP41140 application firmware includes a PMP41140.syscfg file that can be opened using the SysConfig editor in Code Composer Studio™ (CCS). Within the SysConfig interface, a dedicated PowerSUITE category is provided to simplify firmware configuration and customization.

The PowerSUITE interface allows users to modify key application parameters through a graphical user interface rather than manually changing firmware source files. This enables faster evaluation, easier customization for derivative hardware designs, and simplified tuning of control parameters.

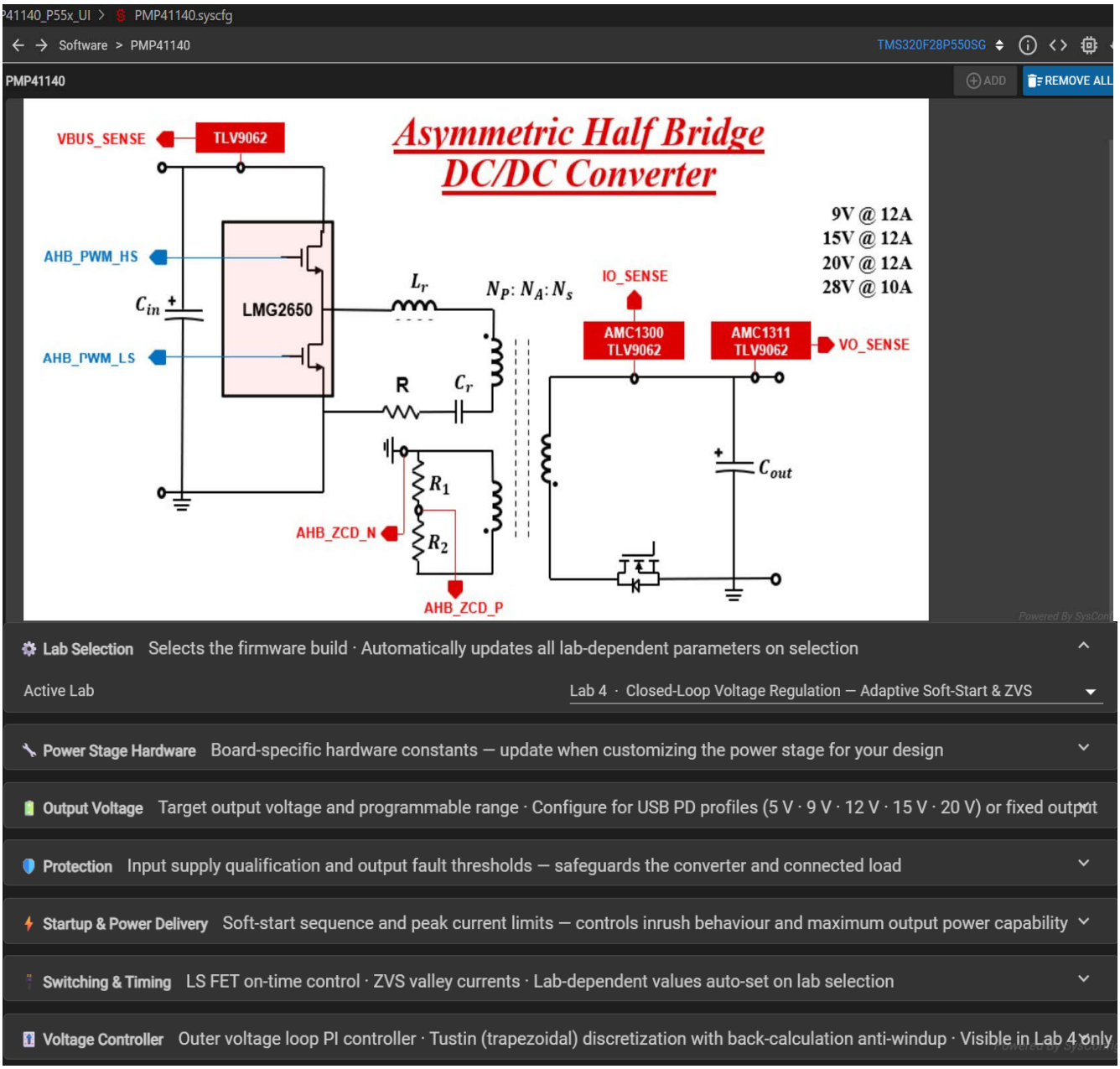


Figure 3-23. PMP41140 PowerSuite (Application UI)

The available configuration categories are described below:

Lab Selection

The firmware supports four laboratory configurations, with Lab 4 selected as the default option. The Lab Selection category allows users to switch between the available labs based on the desired operating mode or evaluation objective. Detailed descriptions of each lab are provided in subsequent sections of this document.

Power Stage Hardware

The Power Stage Hardware category contains parameters related to the physical power converter design.

- Transformer Configuration allows the transformer turns ratio to be updated when adapting the firmware to a custom hardware design.
- Electrical Parameters allow customization of key power-stage components, including magnetizing inductance, resonant inductance, and resonant capacitance.
- Sensing Calibration provides scaling configuration used to convert ADC measurements into real-world electrical quantities. Users can define the maximum measurable voltage and current ranges based on the sensing circuitry implemented on the power board.

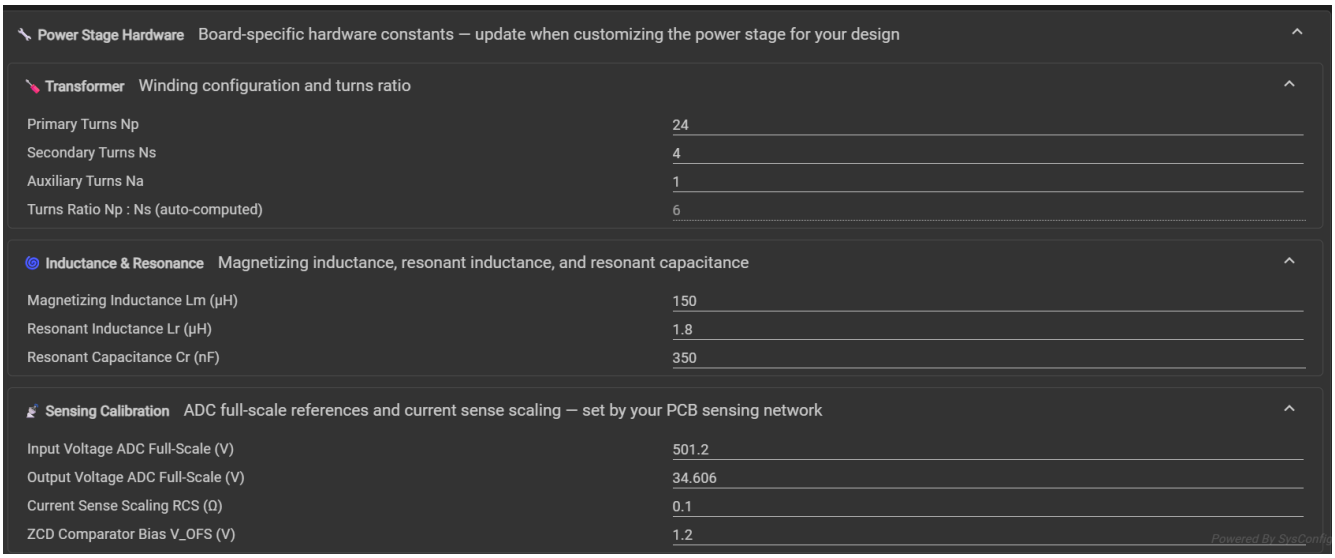


Figure 3-24. Power Stage Hardware

Output Voltage and Protection

This category provides configuration options for output voltage regulation and fault protection.

- The Target Output Voltage and allowable output voltage range can be configured. For the PMP41140 reference design, the supported output voltage range is 9 V to 28 V, with 28 V configured as the default target voltage.
- Protection settings include:
  - Input under-voltage thresholds for both open-loop and closed-loop operation.

– Output over-voltage protection thresholds used to protect the converter and load during abnormal operating conditions.

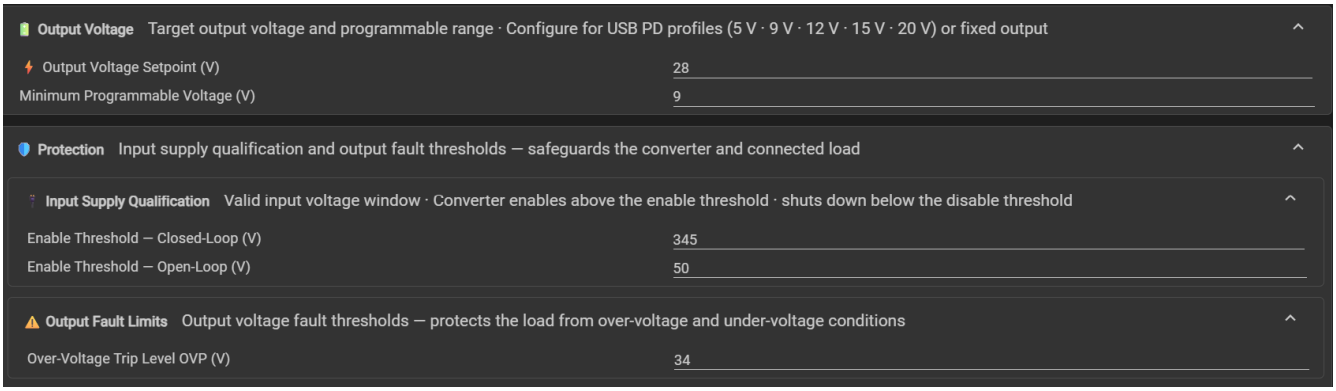


Figure 3-25. Output Volateg and Protection

Startup, Switching and Timing

This category contains parameters that influence startup behavior and switching operation.

- Adaptive Startup can be enabled or disabled depending on application requirements.
- LSFET On-Time Control parameters are provided to configure low-side switch timing behavior.
- ZVS Valley Current Target settings allow adjustment of the target current used to achieve Zero Voltage Switching (ZVS).
- Additional settings are available for ZVS correction inductance limits, diagnostic ratios, and adaptive correction boundaries used by the ZVS control algorithm.

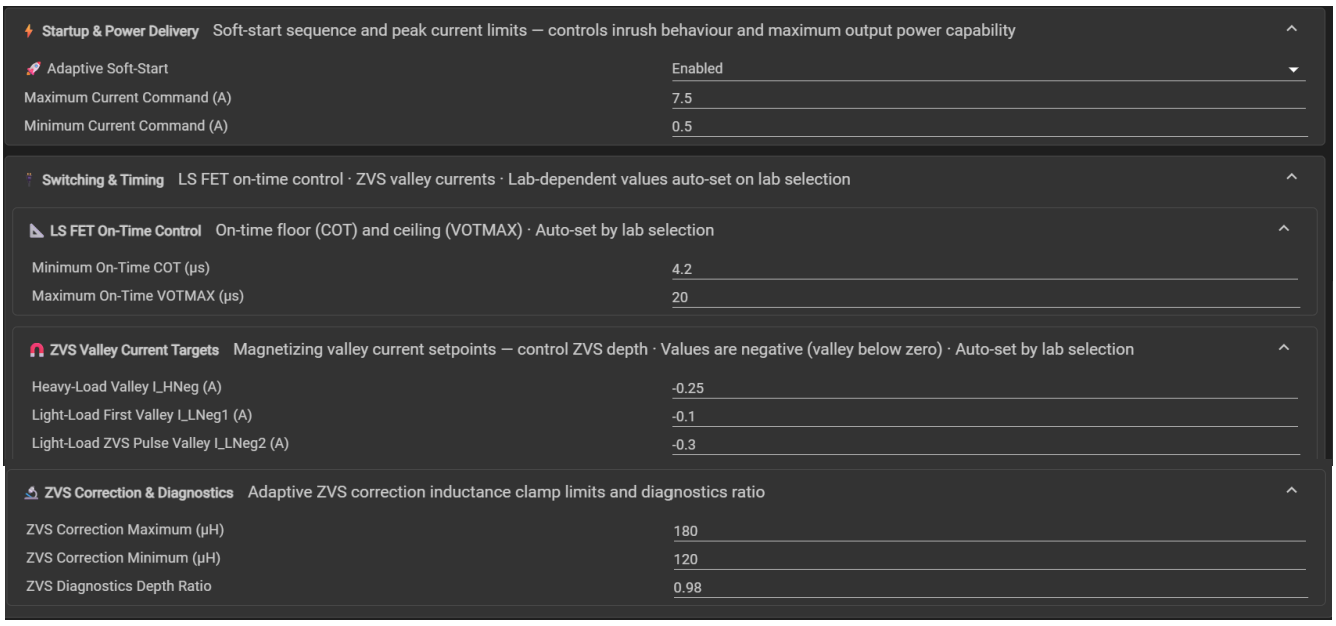


Figure 3-26. Startup, Switching and Timing

Voltage Controller

The Voltage Controller category contains the digital voltage-loop compensation parameters.

- PI controller coefficients and anti-windup parameters can be configured to achieve the desired control-loop response.

- The sampling period can also be adjusted. This value corresponds to the timer interrupt frequency at which the voltage control loop executes and directly impacts control-loop bandwidth and dynamic response.

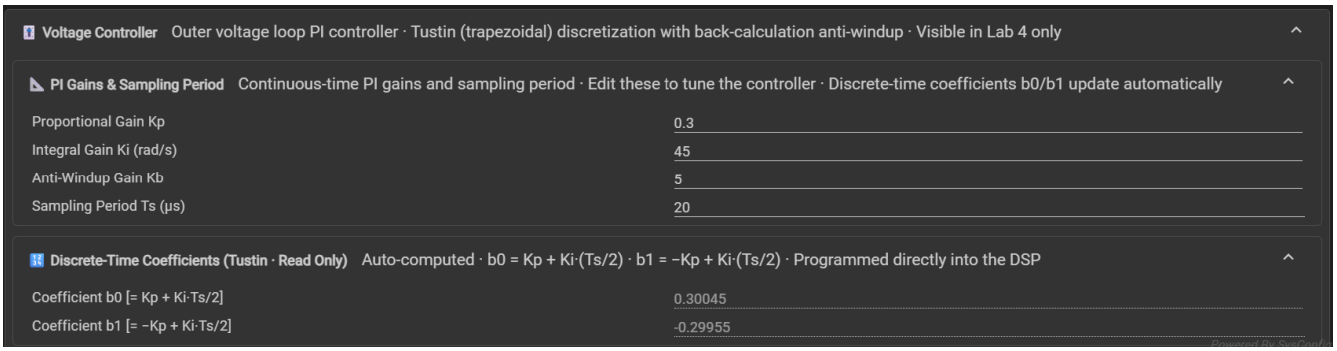


Figure 3-27. PI Controller with anti-windup

The PowerSUITE interface provides a centralized configuration environment that allows users to adapt the PMP41140 firmware to different hardware implementations while minimizing direct source-code modifications.

4 Lab Structure

To build the project, right-click on the project name and click Rebuild Project. The project builds successfully.

To load the project, first make sure in the Project Explorer the correct target configuration file is set as Active under targetConfigs (*.ccxml file). Then, click Run → Debug to launch a debugging session. The project then loads on the device and the CCS debug view becomes active. The code halts at the start of the main routine.

To debug the system, monitor the variables in the watch/expressions window. To populate this window with the correct variables, click View → Scripting Console to open the scripting console dialog box. On the upper right corner of this console, click on Open and then browse to the setupdebugenv_lab.js script file located inside the project folder. This populates the watch window with the appropriate variables needed to debug the system. Enable Continuous Refresh button on the watch window to enable continuous update of values from the controller.

All the labs in this section can be executed using the TMS320F28P550 control card.

To modify the application parameters or switch between different operating labs, the user must directly update the AHB_user_settings.h file. Specifically, the AHB_LAB macro determines which lab is active — by default, it is set to 4, corresponding to Lab4.

The application is organized into four structured labs as described below:

- Lab1: Validation on control card for heavy load PWM operation, analog signal sensing, and fault protection.
- Lab2: Open-loop heavy load operation.
- Lab3: Open-loop light load operation.
- Lab4: Closed-loop operation with load transition, ZVS diagnostics, and adaptive soft-start.

4.1 Hardware Setup

Prior to firmware validation and closed-loop testing, the hardware platform must be assembled and connected according to the recommended setup described in this section.

The purpose of this procedure is to:

1. Establish communication between the control card and power board.
2. Verify correct power-stage operation.
3. Enable safe monitoring of critical converter signals.
4. Minimize measurement noise during laboratory testing.

Hardware Connections

Before applying input power:

1. Connect the TMDSCNCD28P55x controller card to the PMP41140 interface connector
2. Verify proper orientation and connector seating.
3. Connect the input power source to the converter input terminals.
4. Connect an electronic load or resistive load to the output terminals.
5. Connect the debugger interface for firmware loading and real-time monitoring.

The recommended setup is shown in below figure

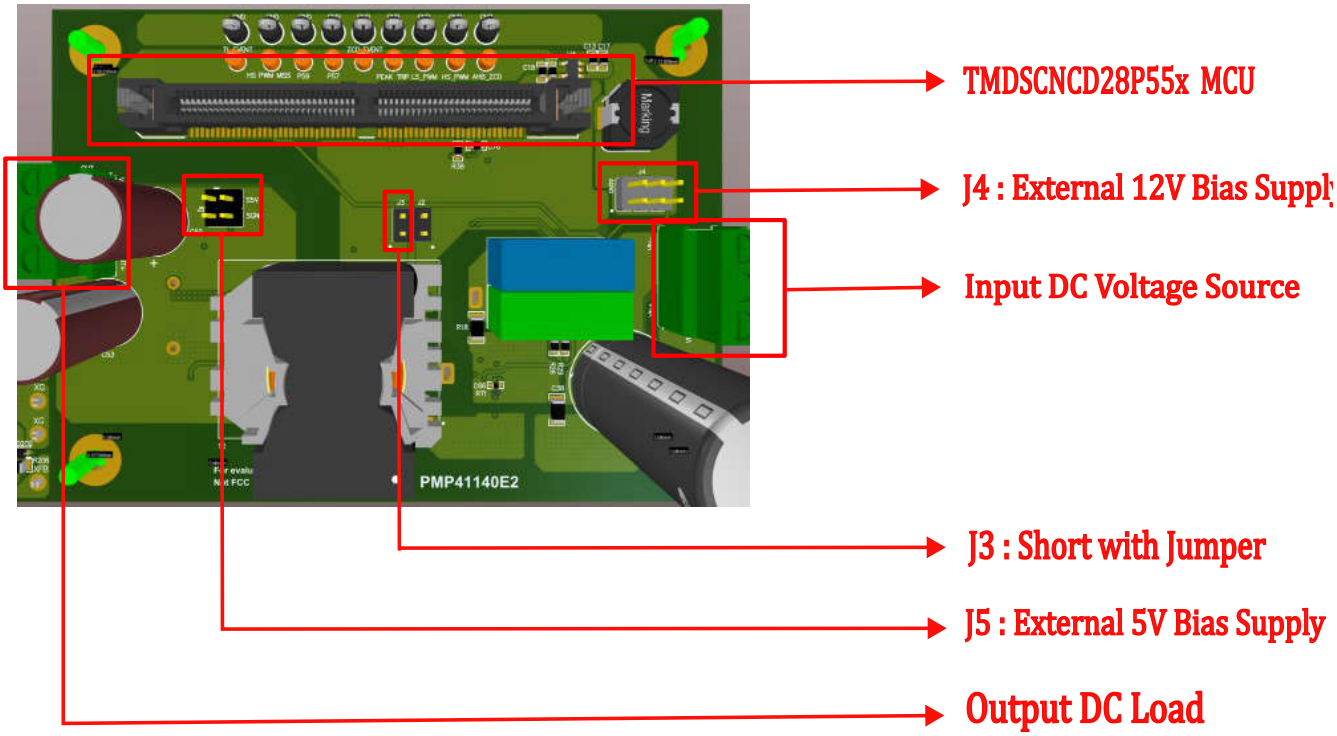


Figure 4-1. AHB Hardware Board Setup

Required Equipment:

1. TMDSCNCD28P55x Control Card
2. DC Source: Input voltage source capable to operate in range till 420V DC
3. Electronic Load: Load which can act as resistive load or constand current load
4. Oscilloscope for monitoring signals

Safety Warning : *Do not touch the board or electrical circuits while the board is energized because of high voltages capable of causing an electric shock hazard. Make sure the high voltage is fully discharged before handling the board.*

Test Setup

After hardware connections are complete:

1. Launch Code Xomposer Studio (CCS)
2. Import the firmware project and in PMP41140.syscfg powersuite choose reuqired Lab
3. Build the project -> Connect to teh target MCU -> Program the device Flash or RAM (Choose build configuration from project properties)
4. Verify successful execution through the watch window.

Key Monitoring Variables:

During the hardwrae validaion, the following variables should be monitored as in below table

Table 4-1. Run Time Variables

Variants	*Description*
AHB_VprimSensed_volts	Input voltage real-time sensing
AHB_aux_ZCD_N_volts	Offset voltage real-time sensing, should be around 1.2V

Table 4-1. Run Time Variables (continued)

Variants	*Description*
Vout_Sensed_Volts	Output voltage real-time sensing
AHB_VsecRef_volts	Target Output voltage command (9~28V)
Kp	Control loop coefficient
Ki	Control loop coefficient
Kd	Control loop coefficient
Kb	Control loop coefficient
AHB_MAG_INDUCTANCE_SET_uH	Initial magnetizing inductance configuration value
ZVS_correction_factor	Adaptive corrected magnetizing inductance value
AHB_ZCDP_Compare	Adaptive ZVS detection threshold
ILm_PeakCurLimit_Amps	Real-time updating primary peak current
Adaptive_ILNEG_AMPS	Adaptive target negative current in heavy load condition
Adaptive_ILNEG1_AMPS	Adaptive target negative current 1 in light load condition
Adaptive_ILNEG2_AMPS	Adaptive target negative current 2 in light load condition
LS_ONTIME_HL_us	Real-time calculating LS on-time in heavy load condition
LS_ONTIME1_LL_us	Real-time calculating LS on-time 1 in light load condition
LS_ONTIME2_LL_us	Real-time calculating LS on-time 2 in light load condition
light_load_enable	Light load mode enable flag; 1=enable, 0=disable
zcd_count_update_new	Adaptive zero-crossing count number in light load condition

User can change "AHB_VsecRef_volts" to the expected output voltage in the range between 9 V to 28V.

Hardware board has several test points including High side and Low side PWM signals. You can probe HS PWM, LS PWM, Switching node voltage of primary LMG2650 and primary resonant current to see the AHB board operation.

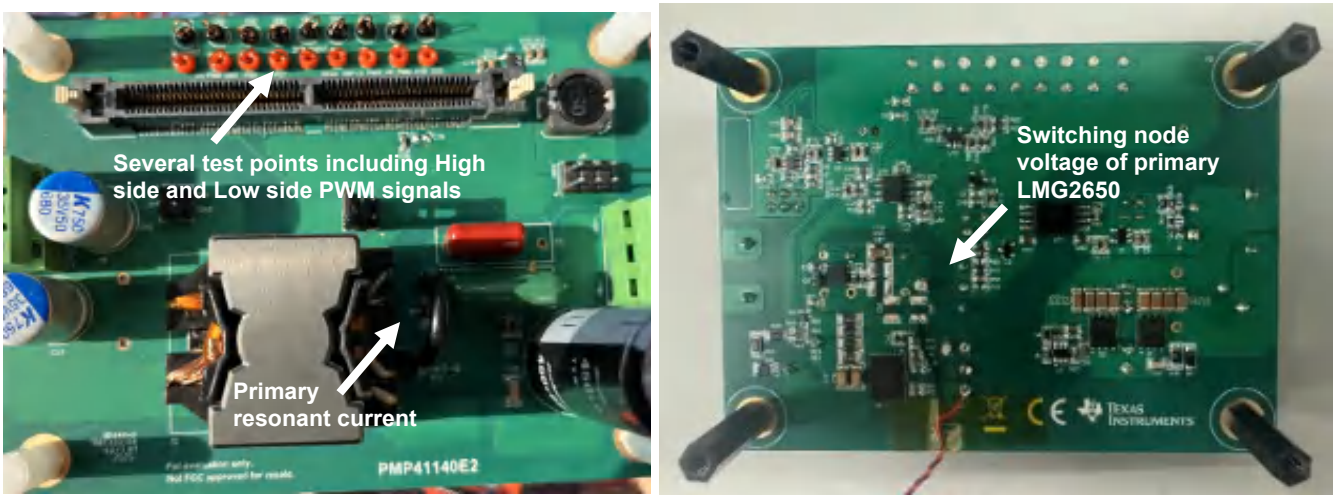


Figure 4-2. Hardware Board Test Points

Recommendation : *The minimum external test gorund loop as shown in above figure to avoid external noise.*

4.2 Lab1

This lab is intended to be executed on the control card along with the docking station.

Set the project to Lab 1 by modifying the Lab Number in the file AHB_user_settings.h. All other configuration parameters can remain at their default values for this stage.

Note

Most users can proceed directly to Lab 2 to begin the functional test flow.

The primary objective of this lab is to validate PWM operation and analog signal sensing, particularly for input and output voltage channels, which are critical for control loop accuracy. Users should observe the lab status and fault status variables to ensure proper system behavior. Key checks include verifying whether overcurrent or overvoltage events on the corresponding fault pins are correctly reflected in the fault status and PWM waveforms.

Building and Loading the Project and Setting up Debug Environment

1. Right-click on the project name and click Rebuild Project.
2. The project will build successfully.
3. In the Project Explorer, make sure the correct target configuration file is set as Active under targetConfigs
4. Then, click Run → Debug to launch a debugging session.
5. The project will then load on the device and the CCS debug view will become active. The code will halt at the start of the main routine.
6. To add the variables in the watch/expressions window, click View → Scripting Console to open the scripting console dialog box. In expressions window, right click -> choose import option and guide to import project folder in the workspace and open "AHB_Lab1Expressions.txt". This will populate the watch window with the appropriate variables needed to debug the system.
7. Click the Continuous Refresh button on the watch window to enable continuous update of values from the controller. The watch window appears as shown in Figure

Running the Code

1. Run the project by clicking run button in CCS debug window.
2. In the watch view, verify PWMISR_count and TimerISR_count are increment continuously.
3. Confirm that *LabStatus.type* displays “OffBoard_Lab_Sensing”.
4. Connect AHB_PWM_HS and AHB_PWM_LS (HSEC 50 and 52) to observe the expected PWM waveforms as shown in the waveform on the oscilloscope.
5. Check that *LS_ONTIME_HL_us* matches the LS-FET ON time measured in the PWM waveform.
6. Validate input voltage sensing by probing VBUS_SENSE (HSEC 9 → AHB_VprimSensed_volts) and validate output voltage sensing by probing VO_SENSE (HSEC 21 → AHB_VsecSensed_volts). Observe the default sensed voltages when pins are floating, as shown in the figure.
7. Apply 3.3 V to AHB_RESCS (HSEC 13) to simulate an over-current fault.
8. Apply 3.3 V to VO_SENSE (HSEC 21) to simulate an over-voltage fault.
9. Depending on the applied condition, verify that *faultStatus.type* updates to:
   a. “OVERCURRENT_FAULT”,
   b. “OVERVOLTAGE_FAULT”, or
   c. “OV_OVC_COMBINED_FAULT”
10. Set faultClear = 1 to clear PWM fault action and re-enable PWM outputs.

11. Set faultStatusClear = 1 to clear CMPSS fault status flags.

Variables | Expressions | Registers | Breakpoints

Expression	Type	Value
PWMISR_count	unsigned long	1976001
TimerISR_cont	unsigned long	11788416
labStatus.type	enum <unnam...	OffBoard_Lab_Sensing
AHB_VprimSensed_volts	float	26.9295235
AHB_VsecSensed_volts	float	1.64455044
AHB_aux_ZCD_N_volts	float	0.146630108
LS_ONTIME_HL_us	float	20.0
Vout_Sensed_Volts	float	1.66862416
faultStatus.type	enum <unnam...	NO_FAULT
faultClear	unsigned char	0 '\x00'
faultStatusClear	unsigned char	0 '\x00'
EPwm3Regs.TBPRD	Register	18081 (Decimal)
EPwm3Regs.CMPA.CMPA	pointer : 16	3000 (Decimal)
Cmpss1Regs.COMPSTS.COMPLSTS	<16-bit unsign...	0x0
Cmpss1Regs.COMPSTS.COMPLLATCH	<16-bit unsign...	0x0
Cmpss3Regs.COMPSTS.COMPHSTS	<16-bit unsign...	0x0
Cmpss3Regs.COMPSTS.COMPHLATCH	<16-bit unsign...	0x0
Add new expression		

Figure 4-3. Lab1 Expressions Window

Figure 4-4. Lab1 Waveform with Ch1 as HS PWM, ch2 as LS PWM

4.3 Lab2

This Lab is intended to execute on the board, for vaildating open loop operation for Heavy Load.

Set the project to Lab 2 by modifying the Lab Number in the file AHB_user_settings.h. All other configuration parameters can remain at their default values for this stage.

The primary objective of this lab is to validate PWM waveforms based on heavy load and also observing peak current of AHB_RESCS by changing the corresponding variable.

Hardware Connection:

1. Plug external bias supply of 12V to "J4" and 5V to "J5" connector on the board. Short J3 with jumper.
2. Connect DC Source of Input voltage 60V by limiting input current to 0.2A.
3. Conect CR Load of 150Ω.

Building and Loading the Project and Setting up Debug Environment

1. Right-click on the project name and click Rebuild Project.
2. The project will build successfully.
3. In the Project Explorer, make sure the correct target configuration file is set as Active under targetConfigs
4. Then, click Run → Debug to launch a debugging session.
5. The project will then load on the device and the CCS debug view will become active. The code will halt at the start of the main routine.
6. To add the variables in the watch/expressions window, click View → Scripting Console to open the scripting console dialog box. In expressions window, right click -> choose import option and guide to import project folder in the workspace and open "AHB_Lab2/3Expressions.txt". This will populate the watch window with the appropriate variables needed to debug the system.

7. Click the Continuous Refresh button on the watch window to enable continuous update of values from the controller. The watch window appears as shown in Figure

Running the Code

1. Run the project by clicking run button in CCS debug window.
2. In the Watch view, confirm PWMISR_count and TimerISR_count are incrementing continuously.
3. Verify LabStatus.type shows “OpenLoop_HighLoad”.
4. Connect AHB_PWM_HS and AHB_PWM_LS (HSEC 50 and 52) to observe the expected PWM waveforms as shown in the waveform on the oscilloscope. Ch3 and Ch4 in the waveform is AHB_ZCD_P and AHB_RESCS respectively.
5. Check that *LS_ONTIME_HL_us* matches the LS-FET ON time measured in the PWM waveform.
6. Validate input voltage sensing by probing VBUS_SENSE (HSEC 9 → AHB_VprimSensed_volts) and validate output voltage sensing by probing VO_SENSE (HSEC 21 → AHB_VsecSensed_volts).
7. With default ILm_PeakCurTemp = 1.0 A:
   a. Expect primary current peak (CH4 / AHB_RESCS) ≈ +1.0 A and negative peak ≈ −0.5 A (matches Adaptive_IHNEG_AMPS).
   b. Expect Output Voltage ≈ 3.25 V.
   c. Expect LS_ONTIME_HL_us ≈ 11.74 µs — confirm matches measured LS FET on time.
8. When ILm_PeakCurTemp is changed to 1.1A:
   a. Expect primary current peak (CH4 / AHB_RESCS) ≈ +1.1 A and negative peak ≈ −0.5 A (matches Adaptive_IHNEG_AMPS).
   b. Expect Output Voltage ≈ 3.89 V.
   c. Expect LS_ONTIME_HL_us ≈ 10.65 µs — confirm matches measured LS FET on time.

9. Observe the Expression window and verify all variables update in real time, and compare them with the oscilloscope waveforms as shown in the figure.

Expression	Value
ahb_opStatus	{type=AHB_SteadyState,desc...
AHB_aux_ZCD_N_volts	1.23226321
AHB_VprimSensed_volts	60.1660271
AHB_VsecRef_volts	9.0
Vout_Sensed_Volts	3.25785089
AHB_MAG_INDUCTANCE_SET_uH	150.0
ZVS_correction_factor	150.0
light_load_enable	0 '\x00'
zcd_count_update_new	5
ILm_PeakCurLimit_Amps	1.0
ILm_PeakCurTemp	1.0
LS_ONTIME_HL_us	11.7458363
LS_ONTIME1_LL_us	8.47186089
LS_ONTIME2_LL_us	3.0
Adaptive_IHNEG_AMPS	-0.518034279
Adaptive_ILNEG2_AMPS	-0.718032241
Kp	0.300000012
Ki	45.0
Kd	0.0
Kb	5.0

Expression	Value
ahb_opStatus	{type=AHB_SteadyState,desc...
AHB_aux_ZCD_N_volts	1.23347163
AHB_VprimSensed_volts	60.1256485
AHB_VsecRef_volts	9.0
Vout_Sensed_Volts	3.89781642
AHB_MAG_INDUCTANCE_SET_uH	150.0
ZVS_correction_factor	150.0
light_load_enable	0 '\x00'
zcd_count_update_new	5
ILm_PeakCurLimit_Amps	1.10000002
ILm_PeakCurTemp	1.10000002
LS_ONTIME_HL_us	10.6534576
LS_ONTIME1_LL_us	7.88961792
LS_ONTIME2_LL_us	3.0
Adaptive_IHNEG_AMPS	-0.518018901
Adaptive_ILNEG2_AMPS	-0.718022287
Kp	0.300000012
Ki	45.0
Kd	0.0
Kb	5.0

Figure 4-5. Lab2 Expressions

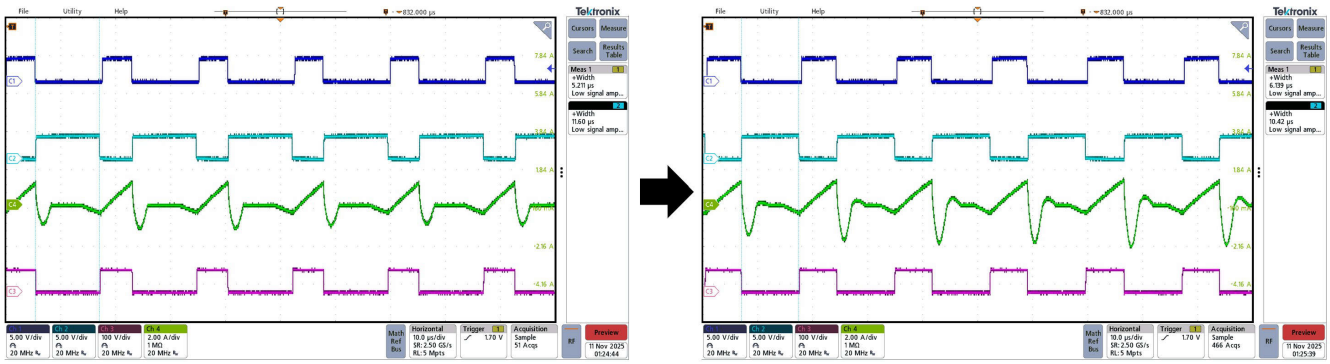


Figure 4-6. Lab2 Waveforms with Ch1 as HS PWM, Ch2 as LS PWM, Ch3 as Auxilary node, Ch4 as Primary Current

4.4 Lab3

This Lab is intended to execute on the board, for vaildating open loop operation for Light Load.

Set the project to Lab 3 by modifying the Lab Number in the file AHB_user_settings.h. All other configuration parameters can remain at their default values for this stage.

The primary objective of this lab is to validate PWM waveforms based on light load and also observing peak current of AHB_RESCS by changing the corresponding variable.

Hardware Connection:

1. Plug external bias supply of 12V to "J4" and 5V to "J5" connector on the board. Short J3 with jumper.
2. Connect DC Source of Input voltage 60V by limiting input current to 0.2A.
3. Conect CR Load of 150Ω.

Building and Loading the Project and Setting up Debug Environment

1. Right-click on the project name and click Rebuild Project.
2. The project will build successfully.
3. In the Project Explorer, make sure the correct target configuration file is set as Active under targetConfigs
4. Then, click Run → Debug to launch a debugging session.
5. The project will then load on the device and the CCS debug view will become active. The code will halt at the start of the main routine.
6. To add the variables in the watch/expressions window, click View → Scripting Console to open the scripting console dialog box. In expressions window, right click -> choose import option and guide to import project folder in the workspace and open "AHB_Lab2/3Expressions.txt". This will populate the watch window with the appropriate variables needed to debug the system.
7. Click the Continuous Refresh button on the watch window to enable continuous update of values from the controller. The watch window appears as shown in Figure

Running the Code

1. Run the project by clicking run button in CCS debug window.
2. In the Watch view, confirm PWMISR_count and TimerISR_count are incrementing continuously.
3. Verify LabStatus.type shows “OpenLoop_LightLoad”.
4. Connect AHB_PWM_HS and AHB_PWM_LS (HSEC 50 and 52) to observe the expected PWM waveforms as shown in the waveform on the oscilloscope. Ch3 and Ch4 in the waveform is AHB_ZCD_P and AHB_RESCS respectively.
5. In initial time (~10S), only LS PWM with 10us on time can be observed after that HS PWM will appear.
6. Check that *LS_ONTIME_HL_us* matches the LS-FET ON time measured in the PWM waveform.
7. Validate input voltage sensing by probing VBUS_SENSE (HSEC 9 → AHB_VprimSensed_volts) and validate output voltage sensing by probing VO_SENSE (HSEC 21 → AHB_VsecSensed_volts).
8. With default ILm_PeakCurTemp = 1.0 A:
    a. Expect primary current peak (CH4 / AHB_RESCS) ≈ +1.0 A and negative peak ≈ −0.71 A (matches Adaptive_ILNEG2_AMPS)
    b. Expect Output Voltage ≈ 3.1 V.
    c. Expect LS_ONTIME_1LL_us ≈ 9.09 µs and LS_ONTIME2_LL_us ≈ 3 µs— confirm matches measured LS FET on time.
    d. We can observe zcd_count_update_new variable is 3, then there will be 2 times zero crossing events in Auxilary node (SW Voltage)

9. Observe the Expression window and verify that all variables update in real time. Compare these values with the oscilloscope waveforms, as shown in the figure that includes the waveforms and the Expression window snapshots (before and after the HS PWM becomes active).

Expression	Value
ahb_opStatus	{type=AHB_SteadyState,desc...
AHB_aux_ZCD_N_volts	1.23508298
AHB_VprimSensed_volts	60.246788
AHB_VsecRef_volts	9.0
Vout_Sensed_Volts	0.0591411106
AHB_MAG_INDUCTANCE_SET_uH	150.0
ZVS_correction_factor	150.0
light_load_enable	1 '\x01'
zcd_count_update_new	3
ILm_PeakCurLimit_Amps	1.0
ILm_PeakCurTemp	1.0
LS_ONTIME_HL_us	20.0
LS_ONTIME1_LL_us	10.0
LS_ONTIME2_LL_us	3.0
Adaptive_IHNEG_AMPS	-0.518047631
Adaptive_ILNEG2_AMPS	-0.718050659
Kp	0.300000012
Ki	45.0
Kd	0.0
Kb	5.0

Expression	Value
ahb_opStatus	{type=AHB_SteadyState,desc...
AHB_aux_ZCD_N_volts	1.23145747
AHB_VprimSensed_volts	60.2871704
AHB_VsecRef_volts	9.0
Vout_Sensed_Volts	3.10022545
AHB_MAG_INDUCTANCE_SET_uH	150.0
ZVS_correction_factor	150.0
light_load_enable	1 '\x01'
zcd_count_update_new	3
ILm_PeakCurLimit_Amps	1.0
ILm_PeakCurTemp	1.0
LS_ONTIME_HL_us	12.5800095
LS_ONTIME1_LL_us	9.09319401
LS_ONTIME2_LL_us	3.0
Adaptive_IHNEG_AMPS	-0.518040836
Adaptive_ILNEG2_AMPS	-0.718044102
Kp	0.300000012
Ki	45.0
Kd	0.0
Kb	5.0

Figure 4-7. Lab3 Expressions

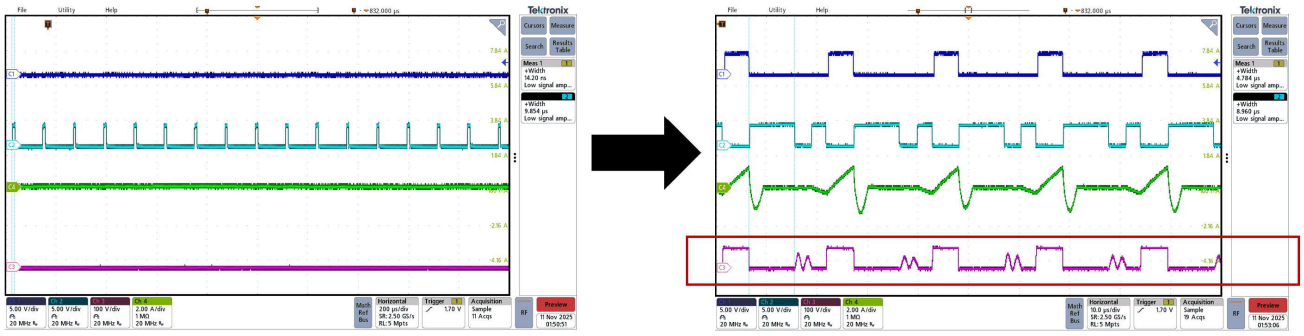


Figure 4-8. Lab3 Waveforms with Ch1 as HS PWM, Ch2 as LS PWM, Ch3 as Auxilary node, Ch4 as Primary Current

4.5 Lab4

This is the default lab and is intended to run on the hardware for validating the complete end-to-end solution. This includes closed-loop regulation across a wide input and output operating range, adaptive soft start, fast load transition response, and ZVS diagnostics to ensure reliable ZVS operation. To enable this lab, set the project to Lab 4 by updating the Lab Number in AHB_user_settings.h. All other configuration parameters can remain at their default values for this stage.

The primary purpose of this lab is to validate the PWM behavior under different load conditions and to observe the closed-loop voltage regulation by adjusting the output reference variable. This lab allows users to verify real-time response, stability, and protection behavior of the converter while operating under practical load and voltage scenarios.

Hardware Connection:

1. Plug external bias supply of 12V to "J4" and 5V to "J5" connector on the board.Short J3 with jumper

2. Connect DC Source of Input voltage should be in the range of 300-400V
3. Connect CC Load (preferbly electornic load) with current limit of 12A and output voltage expected would be in the range of 9 to 28V

Building and Loading the Project and Setting up Debug Environment

1. Right-click on the project name and click Rebuild Project.
2. The project will build successfully.
3. In the Project Explorer, make sure the correct target configuration file is set as Active under targetConfigs
4. Then, click Run → Debug to launch a debugging session.
5. The project will then load on the device and the CCS debug view will become active. The code will halt at the start of the main routine.
6. To add the variables in the watch/expressions window, click View → Scripting Console to open the scripting console dialog box. In expressions window, right click -> choose import option and guide to import project folder in the workspace and open "AHB_Lab4Expressions.txt". This will populate the watch window with the appropriate variables needed to debug the system.
7. Click the Continuous Refresh button on the watch window to enable continuous update of values from the controller. The watch window appears as shown in Figure

Running the Code

1. Run the project using the Run button in the CCS debug window
2. In the Watch view, verify that PWMISR_count and TimerISR_count increment continuously.
3. Confirm that LabStatus.type displays “Closedloop”.
4. Connect the DC input source and power up the board with 320 V input. The default reference output voltage is 9 V.
5. Modify *AHB_VsecRef_volts* variable within the 9 V to 28 V range to observe fast and stable closed-loop voltage regulation.
6. View the waveform to confirm dynamic adjustment of output voltage corresponding to changes in the reference variable. (Ch1: Output Voltage, Ch3: Auxilary node, Ch4: Primary Current)
7. When operating under light load, the firmware automatically adjusts the target ZCD count based on output voltage, improving efficiency; detailed efficiency data is available in the test report.

8. The figure also illustrates the adaptive soft start behavior and undervoltage lockout (UVLO) response.

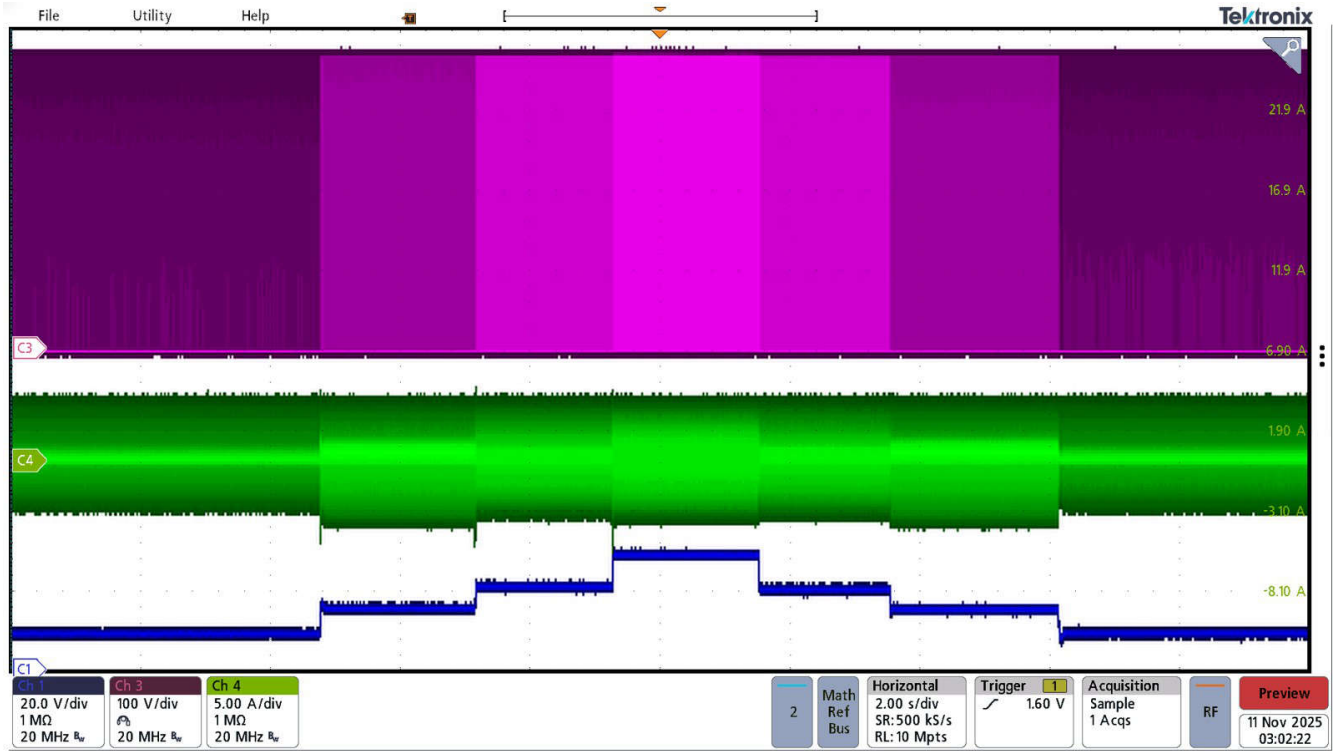
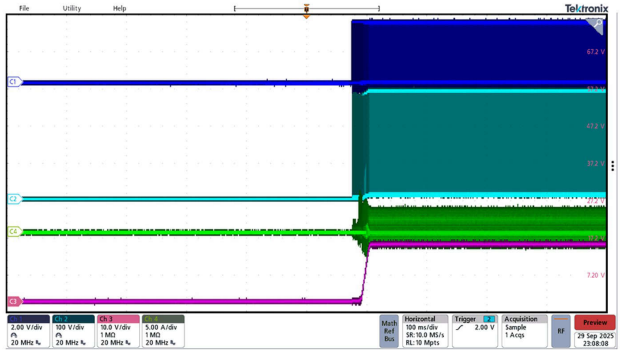
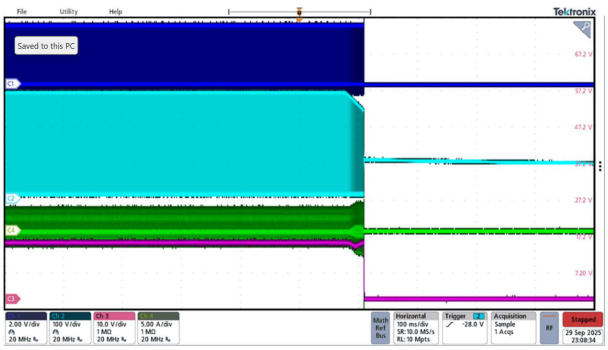


Figure 4-9. Output voltage adjusting 9V->15V->20V->28V->20V->15V->9V with 370V input 9A cc load



a) Soft Start Operation



b) Under-Voltage Protection

Figure 4-10. Soft start and under voltage protection operation with 320V Input, 15V@9A CC load condition

5 Summary

This documentation provides a complete overview of the AHB converter reference design implemented on the TMS320F28P550 MCU. It explains the overall software structure, system initialization flow using TI SysConfig, ISR execution model, crossbar routing, and how the control and sensing framework is organized for reliable converter operation. The architecture uses two ISRs—one operating at variable frequency for PWM updates and another fixed-frequency ISR for the control loop and housekeeping tasks. The document also describes the use of EPWM and CLB modules to generate load-dependent PWM patterns, manage protection events, and support timing requirements such as light-load T1/T2 events and ZVS diagnostics.

The system is divided into four labs that guide the user through testing and validation: sensing and basic PWM checks, open-loop heavy load, open-loop light load, and full closed-loop operation with load transitions and adaptive soft start. Each lab outlines the setup steps, expected signals, and observations to verify proper system behavior on hardware. Additional experimental waveforms—such as efficiency results, ZVS behavior, soft-start characteristics, load transient responses, and fault reaction timings—are provided in the test report, offering deeper validation and correlation with the documented implementation.

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