User's Guide

10-kW, Three-Phase, Three-Level (T-Type) Inverter Using AM263

Sri Vidya Gunturi, Salil Chellappan

ABSTRACT

This user's guide focuses on how AM263x microcontrollers can be used for controlling the TIDA-01606 bidirectional three-level, three-phase, SiC-based inverter and PFC power stage reference design. The TIDA-01606 reference design is useful for customers to implement unidirectional or bidirectional AC/DC and DC/AC power stages in solar inverters, EV chargers, battery energy storage systems, UPS, and EV onboard chargers (OBC). The AM263x effectively provides fast and optimized control-loop performance to gather data, process the gathered data, and updates the system within a defined time frame with tightly integrated high-resolution PWMs for actuation to generate precise duty cycles.

Table of Contents

List of Figures

List of Tables

Trademarks

1 Introduction

This document explains how the AM263x MCU can be used for controlling the TIDA-01606 bidirectional three-level, three-phase, SiC-based inverter and PFC power stage reference design. This hardware is compatible with MCUs in an HSEC controlCARD™ format. While originally designed for the C2000™ MCU product family, this reference design can accept the AM263x controlCARD (TMDSCNCD263) also with minimal modifications. This reference design is useful for implementing unidirectional or bidirectional AC/DC and DC/AC power stages in solar inverters, EV chargers, battery energy storage systems, and UPS and EV onboard chargers (OBC). The T-type three-level power stage used in this design is an excellent choice for improving the power density and efficiency of these systems.

The TIDA-01606 reference design is comprised of four separate boards that work together to form this three-phase inverter reference design:

- A power board, comprised of the switching devices, LCL filter, sensing electronics and board power
- A TMDSCNCD263 general-purpose HSEC controlCARD to connect to the AM2634 Arm® based MCU
- Three gate driver cards, each with two ISO5852S and two UCC5320 gate drivers
- A DC bus voltage measuring board (TIDA-01606 ISOHVCARD)

Additional hardware details are available in the *TIDA-01606: 10-kW, Bidirectional Three-Phase Three-Level (T-type) Inverter and PFC Reference Design* design guide.

2 Hardware Requirements

2.1 Hardware and Test Instruments Required

The reference design is set up with the following elements:

- One TIDA-01606 power board

The following list shows the key resources used for controlling the power stage on the MCU:

- Three TIDA-01606 gate driver cards
- One TIDA-01606 ISOHVCARD
- One TMDSCNCD263 controlCARD
- Mini USB cable
- Computer for loading software to the AM263x MCU

The following list of test equipment is required to power and evaluate the design:

- 15-V, 4-A bench-style supply for primary board power

For PFC mode:

- 400-V, L-L capable three-phase AC source
  - 10-kW resistive load to be connected at the DC output

For Inverter Mode:

- 10-kW star connected resistive load network
- 800-V, 12-A power supply for DC link input
- Four-channel power analyzer
- Oscilloscope with current and high-voltage probes

2.2 Microcontroller Resources Used in the Design

Table 2-1 lists the key resources used for controlling the power stage on the MCU:

Table 2-1. Key Controller Peripherals Used for Control of Power Stage on Board

Pin Number	Description	Software Name
15, 31, 28	Grid Voltage Sense Phase A, B, C	TINV_VGRID_A, B, C
21, 33, 30	Inverter Side Voltage Phase A, B, C	TINV_VINV_A, B, C
25, 37, 34	Inverter Side Current Phase A	TINV_IINV_A, B, C
42	Bus Voltage Sensing	TINV_VBUS

Table 2-1. Key Controller Peripherals Used for Control of Power Stage on Board (continued)

Pin Number	Description	Software Name
40	Bus Voltage Mid-Point Sensing	TINV_VBUS_MID
12, 14, 18, 20	Temperature A, B, C and Ambient	TINV_TEMP_A, B, C, AMB
49, 50, 58	Q1 PWM Phase A, B, C	TINV_Q1_A, B, C
51, 52, 60	Q3 PWM Phase A, B, C	TINV_Q3_A, B, C
53, 54, 62	Q2 PWM Phase A, B, C	TINV_Q2_A, B, C
99, 103, 107	SDFM Data IG A, B, C	TINV_IGRID_A, B, C
101, 105, 109	SDFM Clock IG A, B, C	
57, 75	SDFM Clock Source	
89, 87, 85	SiC Fault Signal A, B, C (active Low)	TINV_FAULT_A, B, C
86, 88, 90, 92	Relay on A, B, C, N	TINV_RELAY_A, B, C, N
61, 63	Gate driver supply PWM	TINV_GATE_DRIVE
59	Control GPIO for FAN	TINV_FAN
108, 110	This is used to see ISR nesting and so forth, on the docking station while starting firmware debug	TINV_PROFILING1,2
95	Gate driver enable	TNV_PWM_EN
81	Gate driver Reset	TINV_R

2.3 Hardware Changes to TIDA-01606, REV-6

Because the TIDA-01606 power board was originally made for controlCARDs using the C2000™ MCU product family, a few minor changes are needed on the power board to adapt to the AM263x controlCARD:

1. Short HSEC connector pins 88 and 94 with jumper wire to provide output for Relay B.
2. R196 is removed from the TIDA-01606 board, because R196 is connected to pin 32 of the HSEC-board which routes to the GND of AM263.

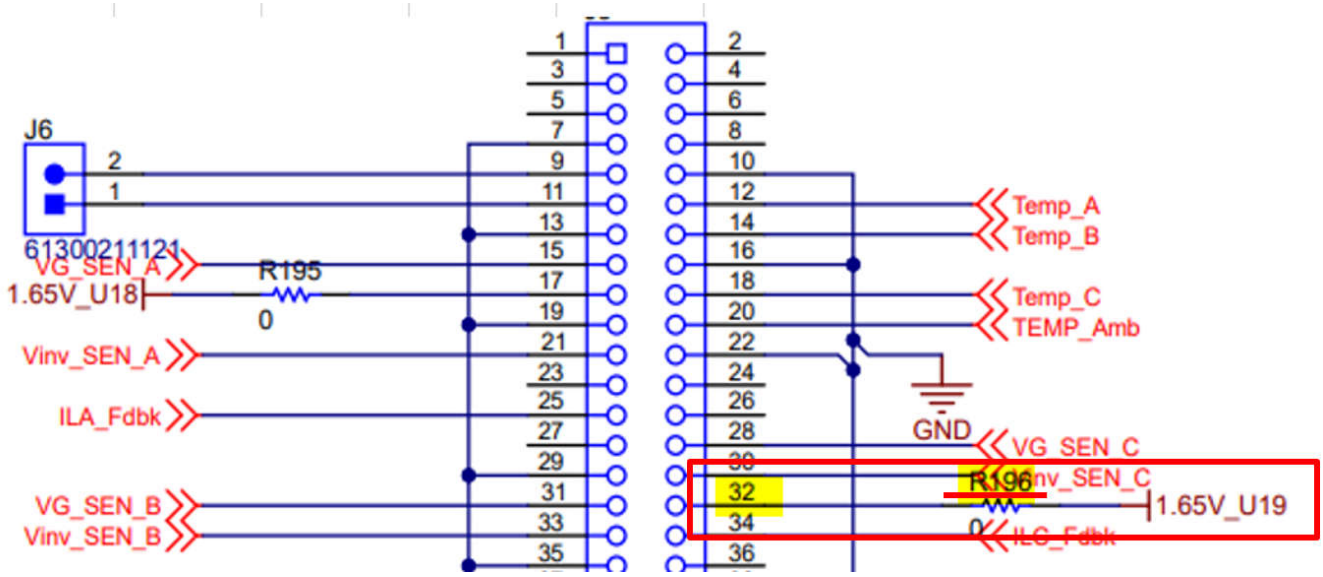


Figure 2-1. R196 in the TIDA-01606

2.4 TMDSCNCD263 controlCARD™ Changes

Minor changes are required on the TMDSCNCD263 controlCARD to control the power hardware. The first change required, allows for accessing pin 92 (GPIO 134) of the HSEC board which is needed to connect to the relay N pin of inverter board.

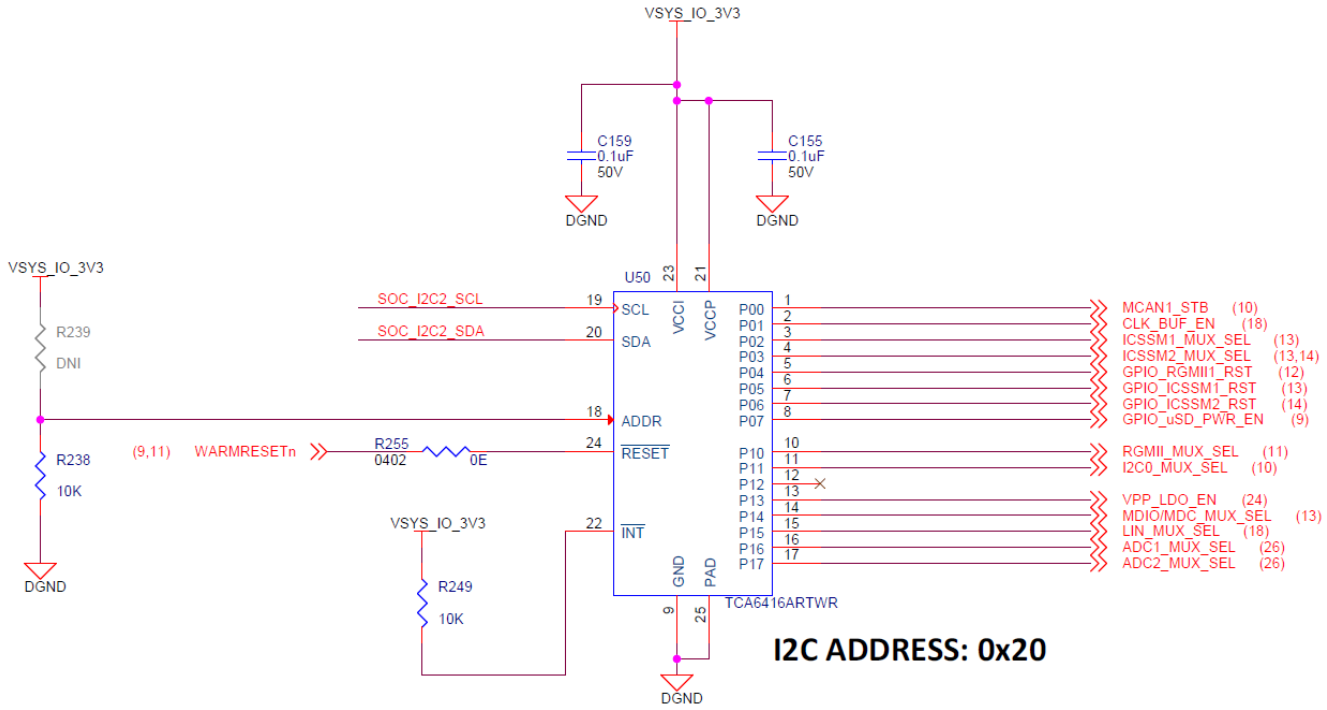
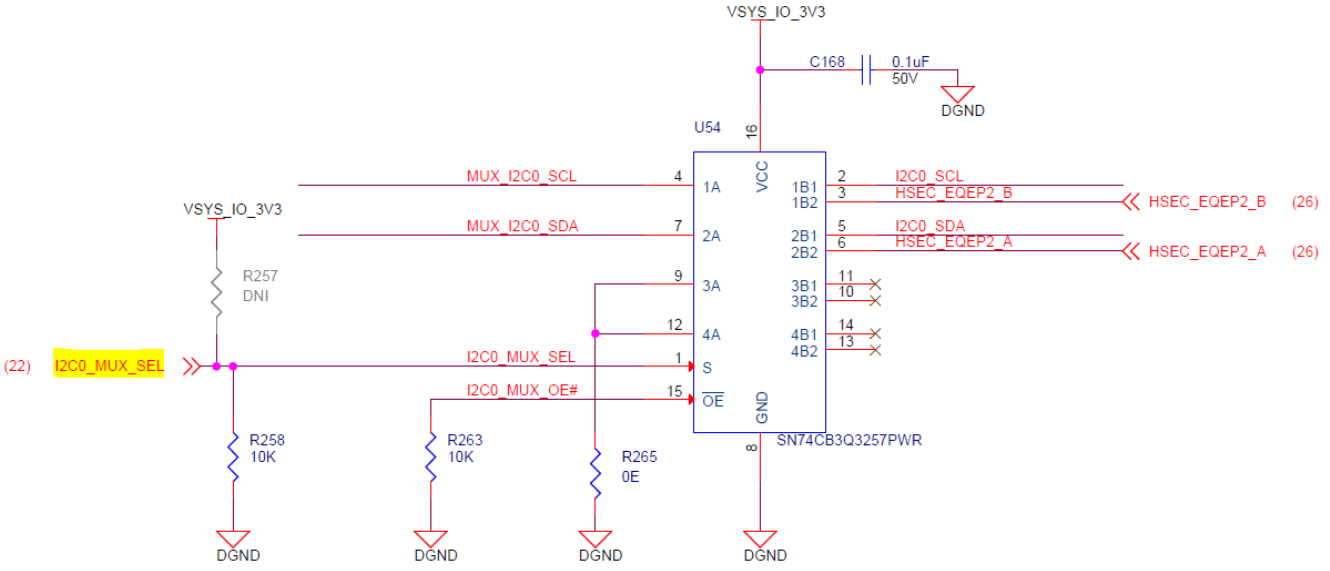


Figure 2-2. I2C2 IO Expander for HSEC Pin 92 Configuration



I2C0 - 1:2 MUX

SEL	CONDITION	FUNCTION
LOW	I2C0 SELECTED	A-->B1 port
HIGH	HSEC EQEP selected	A-->B2 port

Figure 2-3. I2C MUX and DEMUX for HSEC Pin 92 Configuration

Remove the R258 resistor and populate R257 with the same value resistor (10 kΩ). Alternatively, use the I2C2 instance to set the I2C0_MUX_SEL pin to high to access the HSEC_EQEP2_A pin. The SysConfig settings are provided with Figure 2-5 and the example C code for the same is located right after that figure.

GPIO 134 is configured as an I2C FS Open Drain voltage buffer and therefore requires an external pullup to achieve an active high signal.

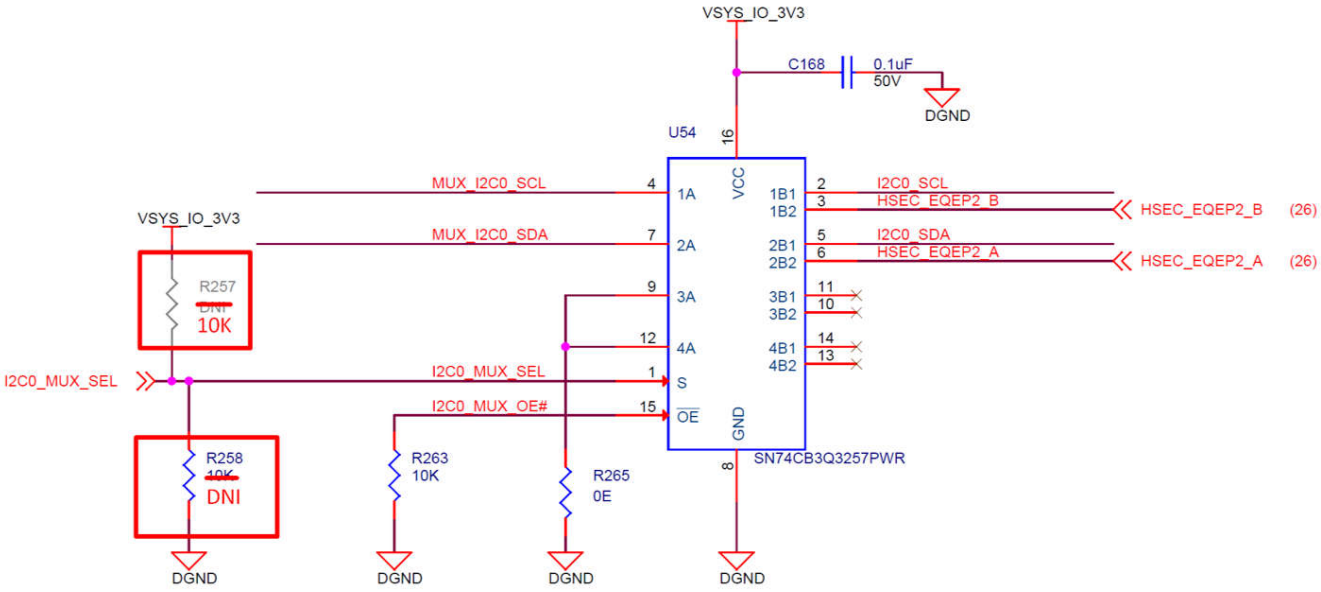


Figure 2-4. I2C0 MUX SEL in AM263x controlCARD

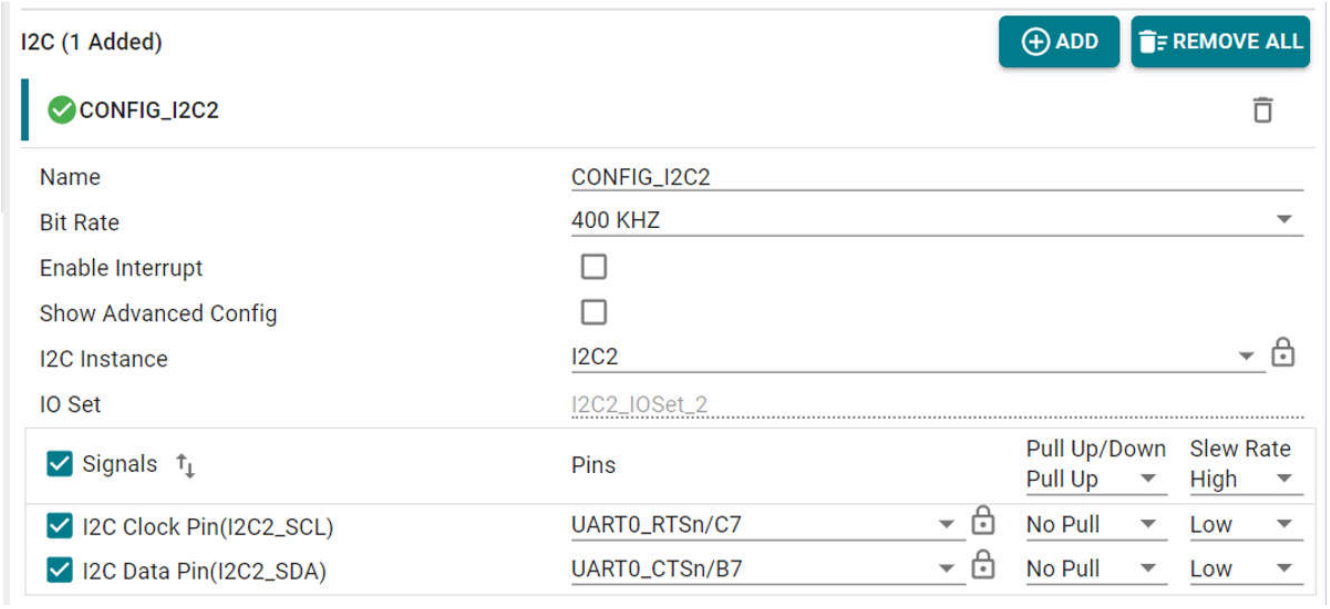


Figure 2-5. I2C SysConfig Configuration for the IO Expander

The following code block shows how to set the I2C0 MUX SEL to high.

```
    I2C_Transaction i2cTransaction;
    uint8_t buffer[2U];
    int32_t status = SystemP_SUCCESS;

    I2C_Transaction_init(&i2cTransaction);
    i2cTransaction.writeBuf     = buffer;
    i2cTransaction.writeCount   = 2U;
    i2cTransaction.slaveAddress = 0x20;

    buffer[0] = 0x03U;
    buffer[1] = (0x01 << 1);
    status += I2C_transfer(I2C_getHandle(CONFIG_I2C2), &i2cTransaction);

    buffer[0] = 0x07U;
    buffer[1] = ~(0x01 << 1);
    status += I2C_transfer(I2C_getHandle(CONFIG_I2C2), &i2cTransaction);
```

The second change needed is for accessing pin 90 (GPIO 86) of the HSEC board for the Relay C connection. Remove the R139 resistor and populate R146 with the same value resistor (0E).

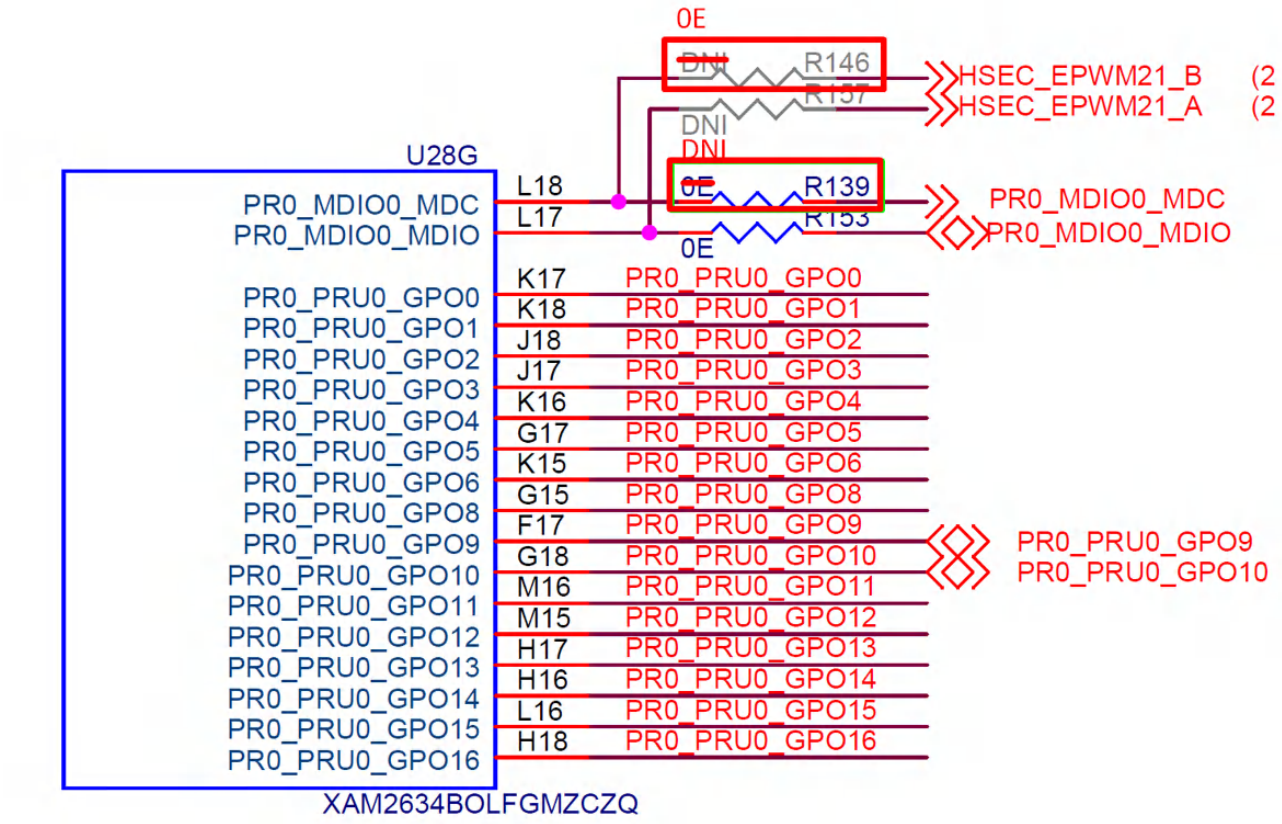


Figure 2-6. L18 Connected to EPW21_B

3 Software

3.1 Getting Started With Firmware

3.1.1 Opening the Code Composer Studio™ Project

The software of this design is available inside the MCU+ Academy. To open the project:

1. If necessary, install Code Composer Studio™ (CCS) version 12 or above by following the instructions on the AM263x MCU+ SDK Getting Started Guide.
2. The minimum supported SysConfig version is 1.14.0 or higher.
3. The minimum supported MCU_PLUS_SDK version for this demo is mcu_plus_sdk_am263x_08_05_00_24.
   - Report in the E2E™ forums if the latest MCU SDK is not supported for the demo.
4. If necessary, the installation steps for each package is provided as part of the SDK and Tools Setup.
5. Download the String Inverter Demo for AM263x from the AM26x Design Gallery in Resource Explorer.
6. Open CCS, and create a new workspace.
7. Inside CCS, go to *Project → Import CCS Project*. Navigate to the location of the firmware, and click import project.
8. To launch the project using CCS, set the EVM in NOBOOT mode. Follow the steps mentioned in the EVM Setup section of the Getting Started Guide to setup the EVM.

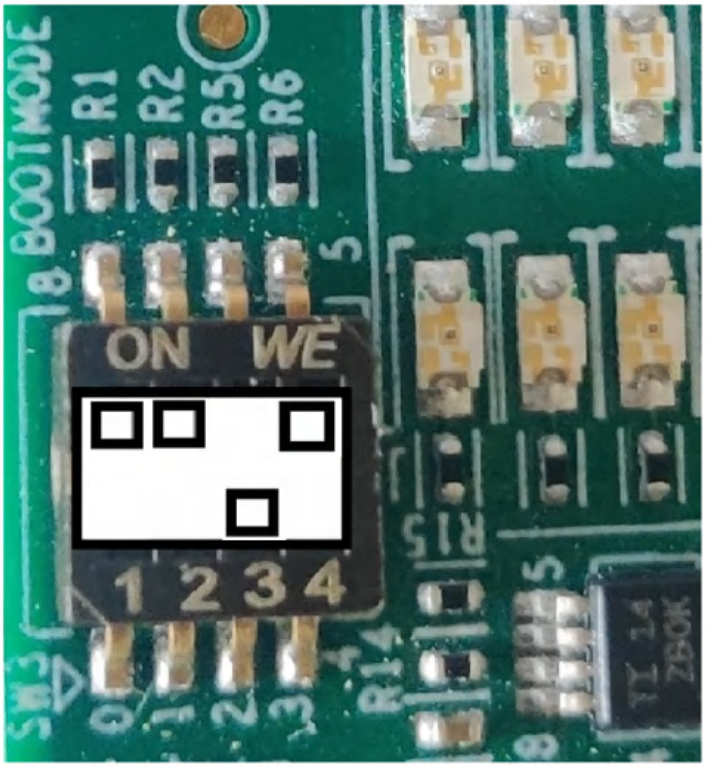


Figure 3-1. No Boot Mode

9. To load and Run the Inverter Demo code into the EVM, follow the steps mentioned in the *Load and Run Example* section of the *Getting Started Guide*.

3.1.2 Software Architecture

The project flow contains initialization of all the peripheral clocks and submodules of the system on a chip (SOC) using the AM263x SysConfig Tool and MCU_PLUS_SDK_AM263x. Initializing the Peripheral IPs used in this design like EPWM, ADC, CMPSS, GPIO, SDFM, and so forth, are discussed in Section 3.2.

The Software contains 2 ISRs and 1 background task. The code flow is described in Figure 3-2. ISR1 is scheduled to run every 20 µs and ISR2 is scheduled to run every 1 ms. ISR1 contains the main control loop explained in Section 3.3.1. ISR2 contains housekeeping functions like calculation RMS values of phase currents and voltages for background monitoring, this is explained in Section 3.3.3.

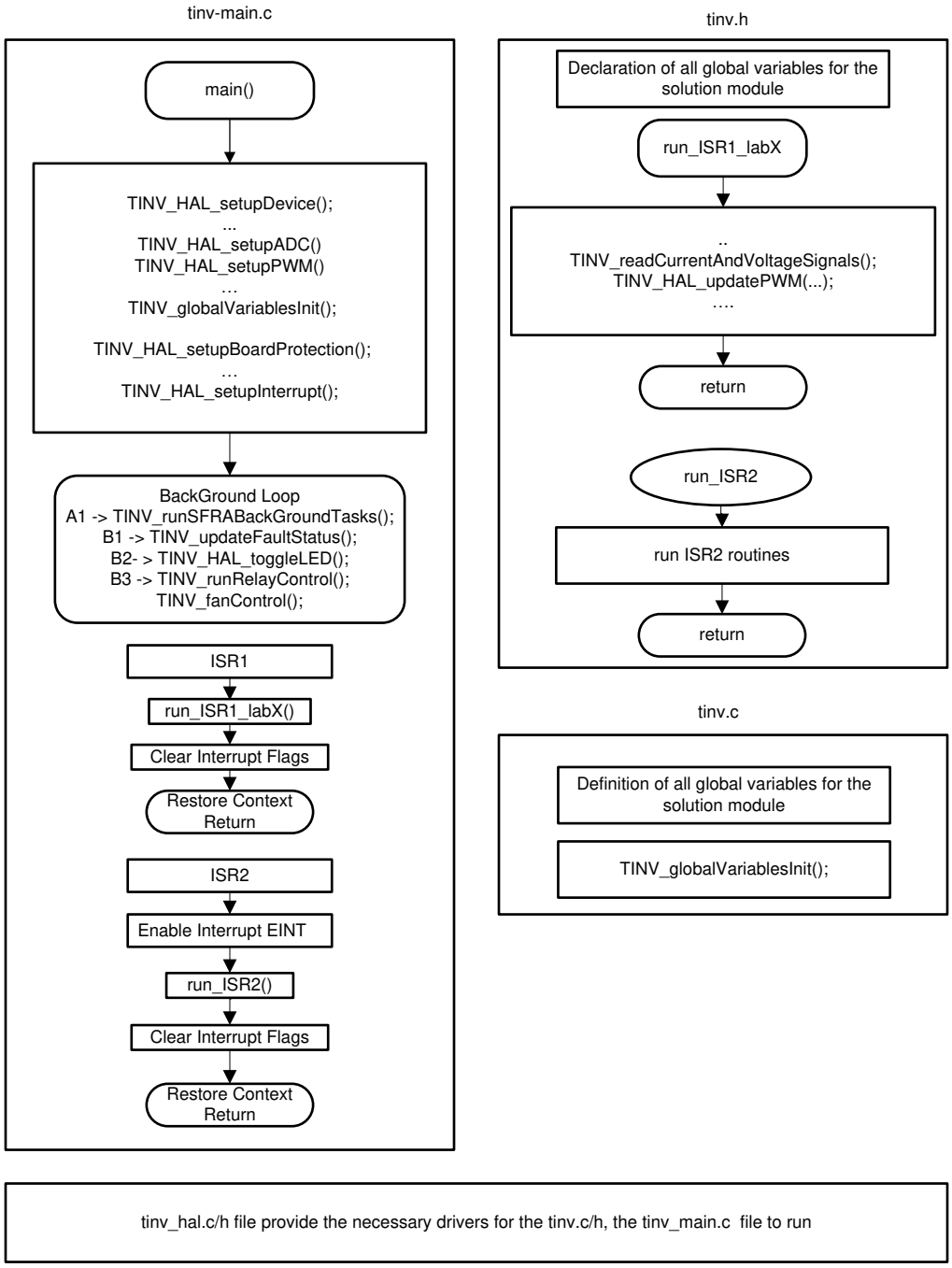


Figure 3-2. Software Block Diagram

The project has two versions for ADC-sensed data depending on the E1 and E2 versions of the controlCARD. Enable the macro AM263x_CC_E1 if the board being used is a E1 version of the controlCARD. Alternatively, enable the macro AM263x_CC_E2 if the board being used is a E2 version of the controlCARD. These macro settings are defined in tinv_user_settings.h file in the *Source/Includes* folder. Choose the .syscfg file according to the board revision in use and exclude the other file from build: `tinv_AM263x_E1.syscfg` for the E1 controlCARD or `tinv_AM263x_E2.syscfg` for the E2 controlCARD.

Background tasks are divided into four sections – A1, B1, B2, B3. A1 is executed every time the Timer connected is overflowed, that is, with a delay of 1 ms controlled by the RTI timer. A1 is used earlier in the C2000 architecture to run the SFRA loop, for now A1 is left open because SFRA is not supported.

B1, B2, and B3 are executed one after the other with a 1-ms delay controller with the RTI timer. This configuration is explained in Section 3.2.3. The B2 task calculates the CPU loading time and average CPU

loading time of ISR1 and ISR2 using ECAP by capturing the GPIO high and low waveforms at the ISRs. The B3 task monitors Fan and Relay Controls connected to the TIDA-01606. The B1 task updates the fault status of the System. This setting is enabled by setting the macro TINV_PROTECTION to TINV_PROTECTION_ENABLED. This step is disabled for Lab 1 by default and enabled for other labs. More information about this is described in Section 3.4.

The following libraries are taken from C2000 Digital Power SDK toolkit and are integrated with the Inverter Demo Project.

- **Transforms Library:** This library is used for performing Clarks and Parks transform functions. The three-phase AC waveform has three different quantities, making it difficult to control the output waveform and making it synchronous across all the three waves. Hence, the waveform must be converted into a DC domain to apply the control gain-related calculations.
  - ABC to DQ → Clarks + Parks transform. This converts the three-phase quantities into DC quantities in a rotating reference frame for positive and negative sequences.
  - DQ to ABC → Inverse Parks + Inverse Clarks transform. This converts the DC quantities in the rotating reference frame into three-phase quantities in the time domain. Both the transforms are implemented purely through calculations in software.
- **Rampgen Library:** This library provides a set of functions that generates a ramp signal of a desired frequency. In the Inverter mode of operation, a reference ramp signal is needed to calculate the required PLL angle for the 50-Hz sine wave output.

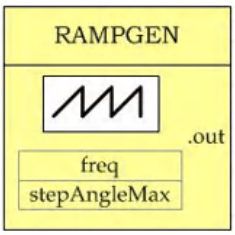


Figure 3-3. Rampgen Model

- **SPLL – Software Phase Locked Loop (PLL):** This software-based PLL code is based on a synchronous reference frame for grid connection to a three-phase grid. To start, three phase voltages A, B, and C are sensed and recorded. These voltages are then converted to D and Q, the DC components. To make sure the three-phase system is synchronous; the Q value must be almost zero. The SPLL uses that setpoint to generate the next theta to provide a valid output. This information is covered in depth in the *Software Phase Locked Loop Design Using C2000™ Microcontrollers for Three Phase Grid Connected Applications* application note.

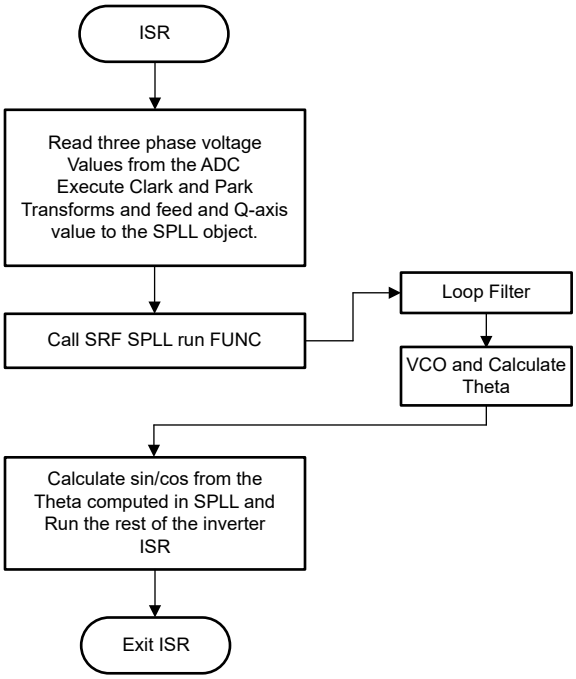
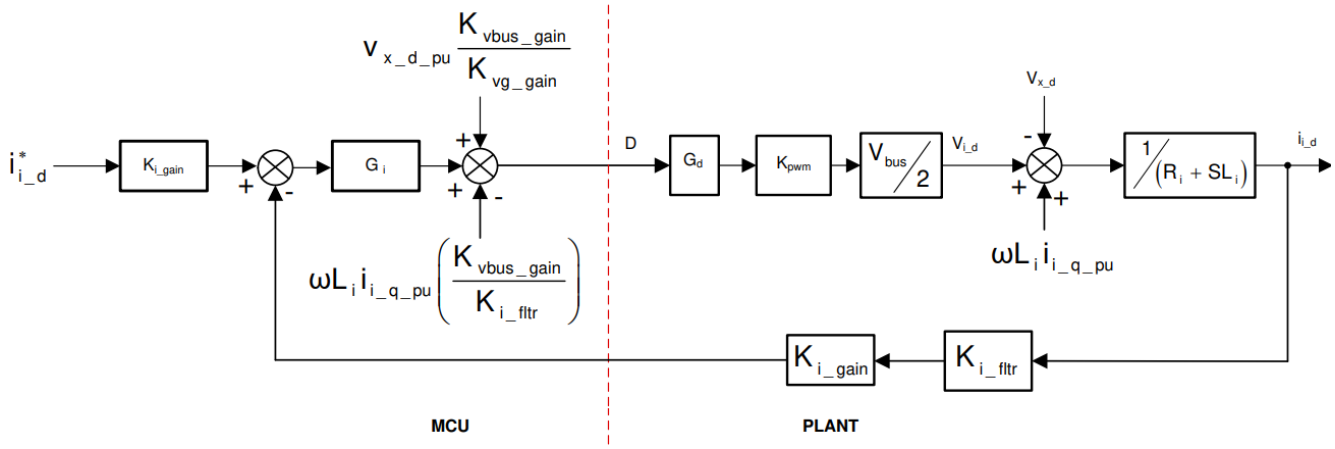


Figure 3-4. SPLL Usage Flow in Control ISR

- **PIC Controller Library:** This demo uses a Proportional Integral Control loop (PI) to maintain the output currents. The user chooses a reference current value for the system. The PI controller must maintain this current in the system no matter the input. Id and Iq values are calculated from Ia, Ib, and Ic. Now the generated Id is compared with the reference Id and the error is calculated. Next, the pulse width of the PWMs is determined, which regulates the output current. Figure 3-5 shows the block diagram for the Current Control Loop. The calculations for the PIC Controller are explained in the control design section of the *10-kW, Bidirectional Three-Phase Three-Level (T-type) Inverter and PFC Reference Design* design guide.



where:

- $i^*_{i\_d}$ = current reference
- K_{i_gain} = current sense scalar which is one over maximum current sense
- K_{i_fltr} is the filter that is connected on the current sense path. Current sense scalar which is one over maximum current sense
- K_{vbus_gain} = voltage sense scalar for the bus, which is one over maximum voltage sensed
- K_{vg_gain} = voltage sense scalar for the grid voltage, which is one over maximum voltage sensed

Figure 3-5. I_d Current Loop Model

3.1.3 Project Folder Structure

Figure 3-6 shows the general structure of the project. Once the project is imported, the *Project Explorer* appears inside CCS.

Design-specific and device-independent files that consist of the core algorithmic code are in the `Source` folder. Library and Control Algorithms related to Digital Power Applications are present in the `libraries` folder. Board specific and Driver specific changes are configured in SysConfig and this code is automatically generated by SysConfig in the `Release/SysCfg` folder. The `main.c` file consists of the main framework of the project. This file consists of calls to the board and file that helps create the system framework, along with the interrupt service routines (ISRs) and slow background tasks. Figure 3-6 explains a detailed folder structure.

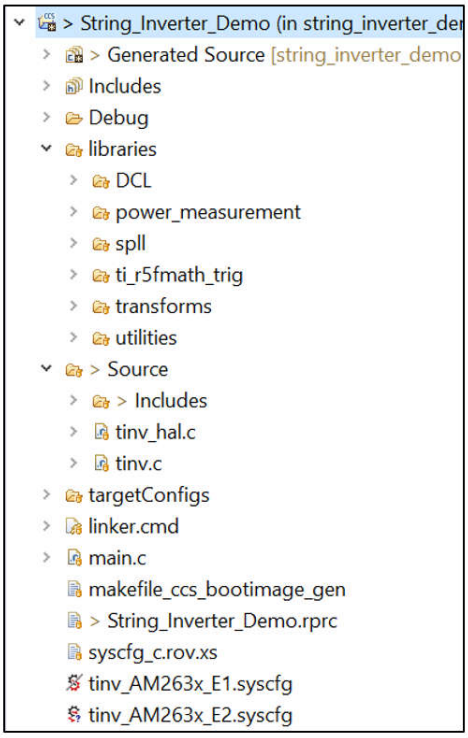


Figure 3-6. Folder Structure of the Demonstration

All variables and function calls are prepended by the *TINV* name (for example, *TINV*_vSecSensed_pu). This naming convention lets the user combine different designs while avoiding naming conflicts.

3.2 SysConfig Setup

The AM263x controlCARD has two released revisions – E1 and E2. The revisions differ in ADC – CMPSS Mappings. Choose the .syscfg file according to the board revision in use and exclude the other file from the build.

The `tinv_AM263x_E1.syscfg` or `tinv_Am263x_E2.syscfg` is used to configure all the modules present in the String Inverter Demo. To run the control loop, the modules needed are the ADC, SDFM, and EPWM modules. For board protection-related applications, CMPSS configurations are required.

Finally, to link these modules from one component to another, a few XBARs are configured. Each of these modules are independently explained in the next section.

3.2.1 EPWM Configuration

PWMs are one of the core parts of the software configuration for the inverter. The PWM module of AM263x inherited features from TI classic C28 controllers. A total of 12 PWM channels are needed to control the inverter output.

Six EPWM instances are created for controlling the three arms of the inverter: *TINV_Q1_Q3_A_PWM_BASE*, *TINV_Q2_Q4_A_PWM_BASE*, *TINV_Q1_Q3_B_PWM_BASE*, *TINV_Q2_Q4_B_PWM_BASE*, *TINV_Q1_Q3_C_PWM_BASE*, *TINV_Q2_Q4_C_PWM_BASE*. The EPWM peripheral clock is running at 200 MHz. Each EPWM is configured for 50-kHz frequency at Up-Down Count mode.

$$Time\ Base\ Period\ for\ Up-Down\ Count\ Mode = \frac{\left(\frac{EPWMCLK}{HSPCLK \times CLKDIV}\right)}{2 \times ReqFreq} = \frac{200\ MHz}{100\ kHz} = 2000 \quad (1)$$

From Equation 1, each EPWM Time base Period is 2000, when the High-Speed and Time Base Clock divider are 1. In this section, the initialization of only *TINV_Q1_Q3_A_PWM_BASE* is explained. However, the same initialization is repeated for the other 5 PWM instances. Figure 3-7 illustrates the complete Time Base configuration of *TINV_Q1_Q3_A_PWM_BASE*.

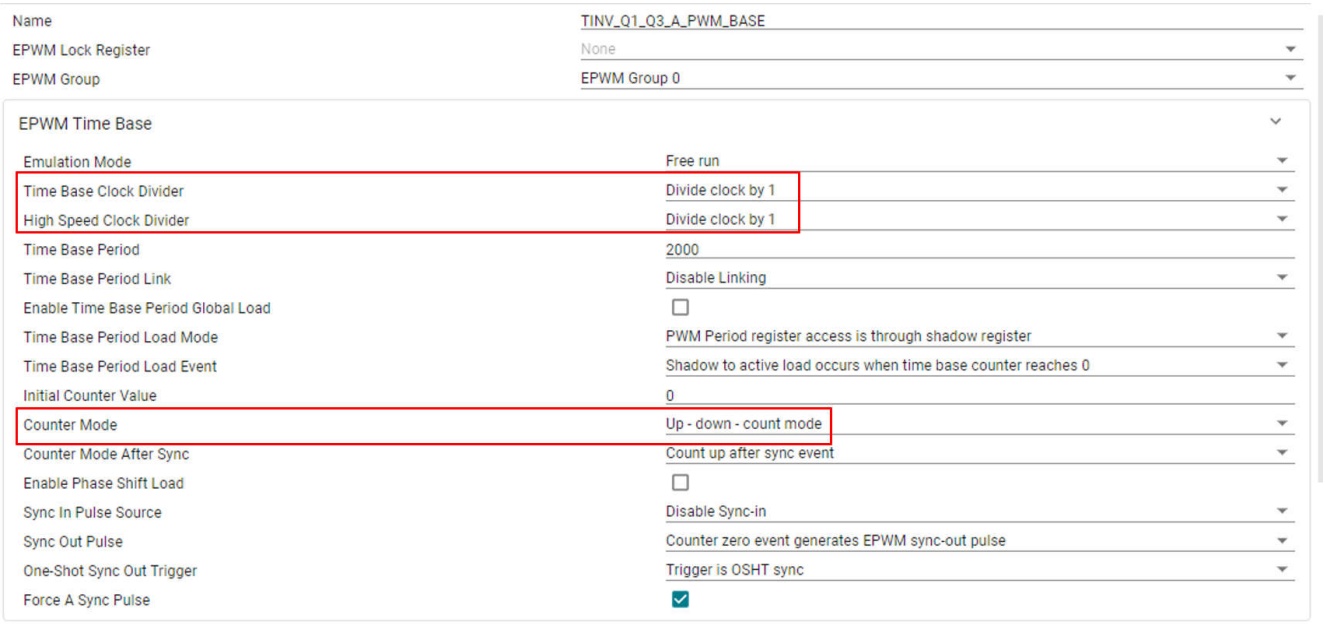


Figure 3-7. EPWM Time Base Submodule Configuration

The Pulse Width of every EPWM is changed every 20 µs in the ISR by modifying the Counter Compare A. The value being placed in the CMPA register is calculated from a PLL loop, more about this is discussed in future sections. For now, the CMPA is initialized to 0.

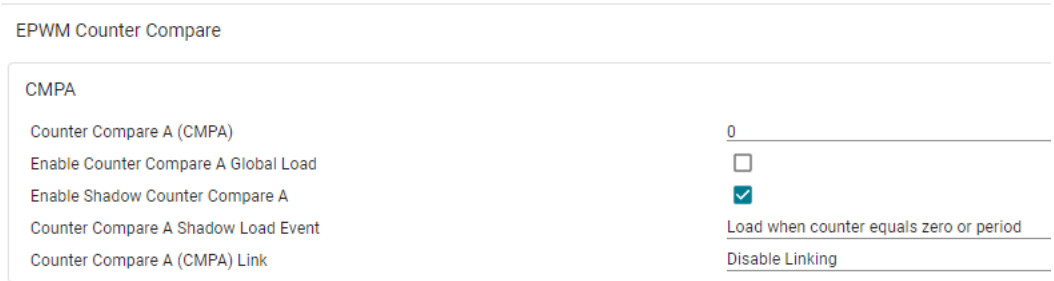


Figure 3-8. EPWM Counter Compare Submodule Configuration

The EPWM pulse is configured to go high and low depending on the event when the counter reaches the CMPA value. When the counter reaches CMPA while counting up, the EPWM Pulse goes Low. Similarly, when the counter reaches the CMPA value while counting down, the EPWM sets high. This gives higher duty cycles for high CMPA values and lower duty cycles for low CMPA values.

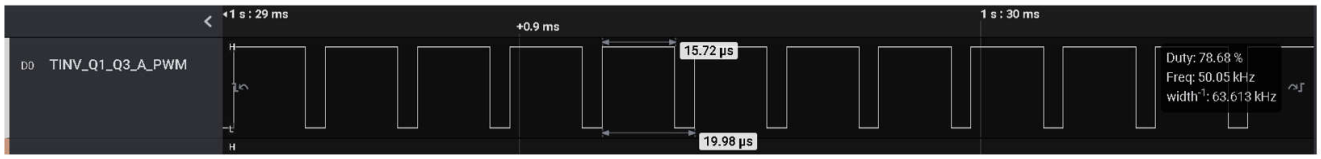


Figure 3-9. Higher CMPA Value Creates a High Duty Cycle

Figure 3-10. CMPA Value Alters the Duty Cycle of the EPWM0 Waveform

EPWM Action Qualifier

Continuous SW Force Global Load	☑
Continuous SW Force Shadow Mode	Shadow mode load when counter equals zero or period
EPWMXA OutputOutput Configuration	
EPWMXA Global Load Enable	☑
EPWMXA Shadow Mode Enable	☑
EPWMXA Shadow Load Event	Load when counter equals zero
EPWMXA T1 Trigger Source	Digital compare event A 1
EPWMXA T2 Trigger Source	Digital compare event A 1
EPWMXA One-Time SW Force Action	No change in the output pins
EPWMXA Continuous SW Force Action	Set output pins to low
Events Configured For EPWMXA Output	None
EPWMXA OutputEvent Output Configuration	
EPWMXA TBCTR Equals Zero	No change in the output pins
EPWMXA TBCTR Equals Period	No change in the output pins
EPWMXA TBCTR Up Equals COMPA	Set output pins to low
EPWMXA TBCTR Down Equals COMPA	Set output pins to High
EPWMXA TBCTR Up Equals COMPB	No change in the output pins
EPWMXA TBCTR Down Equals COMPB	No change in the output pins
EPWMXA T1 Event On Count Up	No change in the output pins
EPWMXA T1 Event On Count Down	No change in the output pins
EPWMXA T2 Event On Count Up	No change in the output pins
EPWMXA T2 Event On Count Down	No change in the output pins

Figure 3-11. EPWM Action Qualifier Submodule Configuration

These actions are loaded into all the EPWMs at their next CTR = 0 event. This is the shadow to the active load option when CTR = 0 in the Action Qualifier Settings. The load mode is chosen as Global to make sure the load happens at the same time to all PWMs. For more information on the shadow to active load, see the *AQCTLA and AQCTLB Shadow Mode Operations* section of the AM263x Sitara™ Microcontroller Technical Reference Manual.

EPWMB is configured through a dead band with a rising and falling edge delay of 20 (20 × 5 = 100-ns delay).

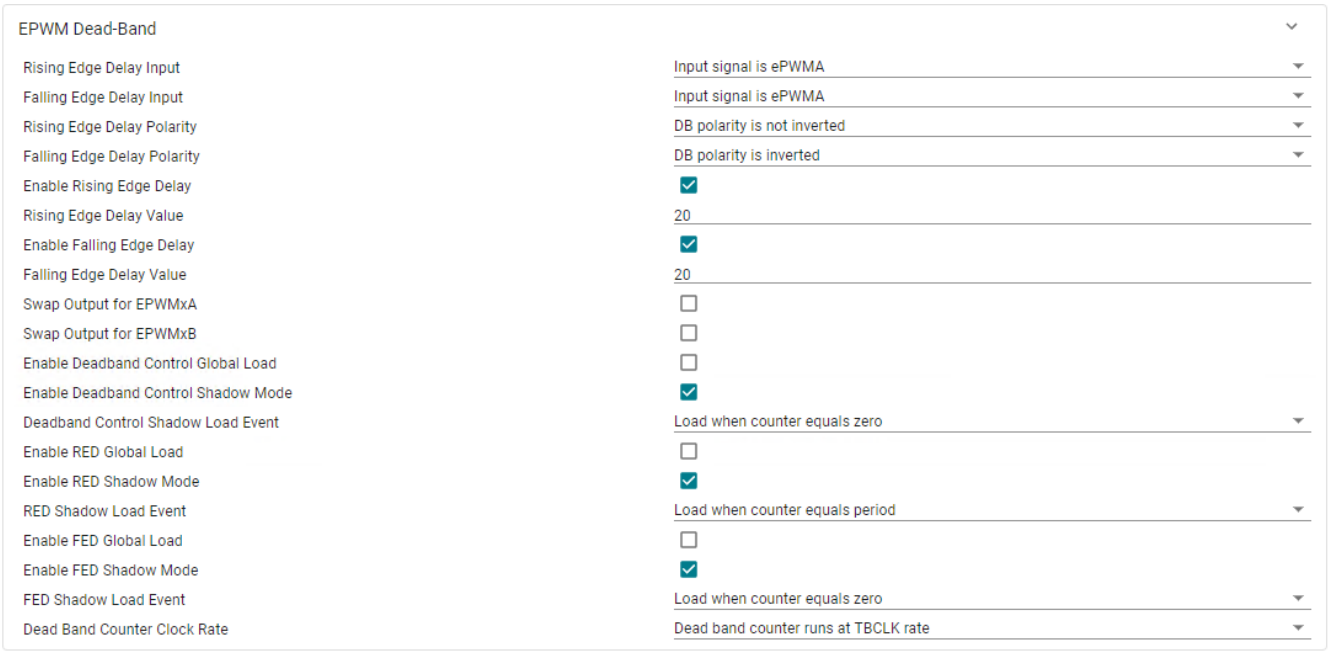


Figure 3-12. EPWM Dead Band Submodule Configuration

Digital Compare and Trip Zone EPWM Modules for overcurrent and overvoltage protection are configured in the CMPSS section of the document.

3.2.2 EPWM Event Trigger Interrupt

ISR1 is triggered by the counter = CMPC event of *TINV_Q1_Q3_A_PWM* while counting up. This is configured in the event trigger sub-module of EPWM as Figure 3-13 shows for a CMPC = 100 event. Hence, the ISR1 is triggered every 20 µs.

Figure 3-13. EPWM Event Trigger Submodule Configuration

These settings can be adjusted according to the design requirements. There are total 32 different interrupt XBARs in AM263. This EPWM Event is routed into the Interrupt XBAR0. The interrupt is later constructed and ISR routine is added to the same XBAR0.

Figure 3-14. INT XBAR Configuration for EPWM1 Interrupt

3.2.3 Timer Configuration

The ISR2 Timer is used to implement the following functions:

- Calculate the RMS values of sensed currents and voltages, then computes the RMS value of power.

- Calculate active and reactive powers from the computed I_d, I_q, V_d, and V_q values
- Sense and compute temperatures of all the channels using ADC.

This ISR2 is triggered every 1 ms, using an RTI timer interrupt. Figure 3-15 shows the SysConfig options necessary to configure the timer interrupt with a timer callback routine named ISR2.

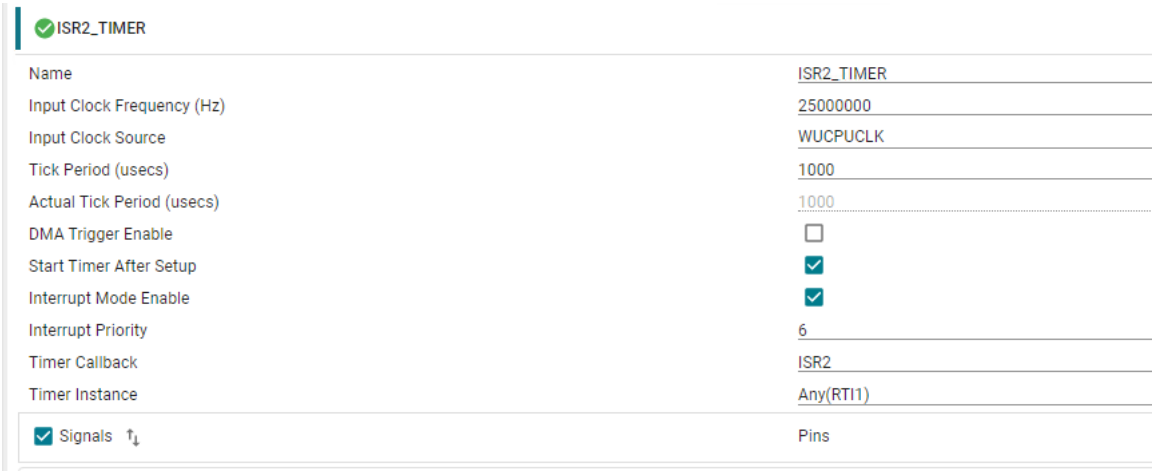


Figure 3-15. Timer Configuration for ISR2

Similarly, *TASK_A_TIMER* and *TASK_B_TIMER* are configured with a Tick Period of 1 ms for TASK_A and TASK_B operations. These tasks monitor the overcurrent events for board protection and calculate the loading averages of ISR1 and ISR2.

3.2.4 SDFM Configuration

Configure three SDFM Channels for current sensing in all three phases A, B, and C as shown in Figure 3-16.

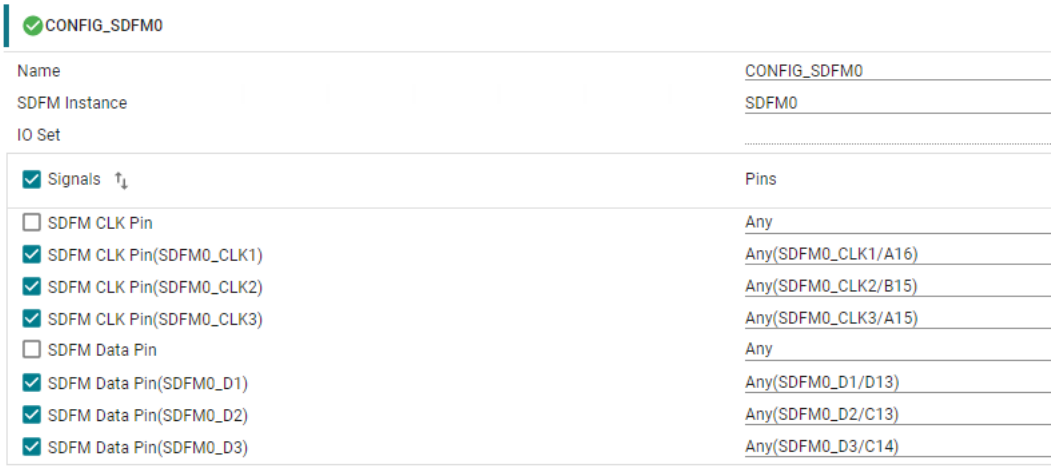


Figure 3-16. SDFM Configuration

3.2.5 ADC Configuration

ADCs are used in this demonstration to sense all voltages, currents, and temperatures on the grid side, inverter side, bus side, and so forth. A total of 11 ADCs are configured along with three optional ADCs for current sensing if ADC is used instead of SDFM. Table 3-1 through give a summary of the ADC SysConfig. The signals sensed on the inverter side of the circuit, are defined using the term *Inv* and the signals sensed on the grid side of the circuit are defined using the term *Grid*. The current values are the same on both Inverter and Grid sides.

The ADC HSEC Board Pin out is different for E1 and E2 versions of the AM263x controlCARD, which is mentioned in Table 3-2.

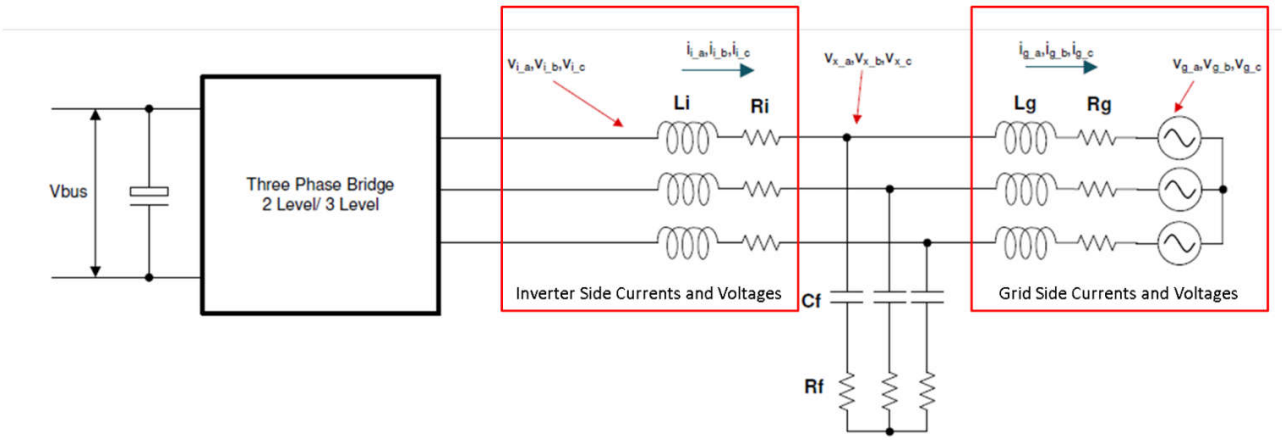


Figure 3-17. Grid Side and Inverter Side of TIDA-01606

Table 3-1. ADC to PWM SOC Mapping for E1 controlCARD

ADC Type	ADC Base Address	ADC SOC Number	ADC Pin	Trigger
TINV_TEMP_A_ADC_BASE	CONFIG_ADC0_BASE_ADDR	ADC_SOC_NUMBER0	ADC0_AIN0	ePWM0, ADCSOCB
TINV_TEMP_B_ADC_BASE	CONFIG_ADC0_BASE_ADDR	ADC_SOC_NUMBER1	ADC0_AIN1	ePWM0, ADCSOCB
TINV_VGRID_A_ADC_BASE	CONFIG_ADC0_BASE_ADDR	ADC_SOC_NUMBER2	ADC0_AIN2	ePWM0, ADCSOCA
TINV_TEMP_C_ADC_BASE	CONFIG_ADC1_BASE_ADDR	ADC_SOC_NUMBER0	ADC1_AIN0	ePWM0, ADCSOCB
TINV_TEMP_AMB_ADC_BASE	CONFIG_ADC1_BASE_ADDR	ADC_SOC_NUMBER1	ADC1_AIN1	ePWM0, ADCSOCB
TINV_VINV_A_ADC_BASE	CONFIG_ADC1_BASE_ADDR	ADC_SOC_NUMBER2	ADC1_AIN2	ePWM0, ADCSOCA
TINV_IINV_A_ADC_BASE (or SDFM)	CONFIG_ADC1_BASE_ADDR	ADC_SOC_NUMBER3	ADC1_AIN3	ePWM0, ADCSOCA
TINV_VGRID_C_ADC_BASE	CONFIG_ADC2_BASE_ADDR	ADC_SOC_NUMBER0	ADC2_AIN2	ePWM0, ADCSOCA
TINV_VINV_C_ADC_BASE	CONFIG_ADC2_BASE_ADDR	ADC_SOC_NUMBER1	ADC2_AIN3	ePWM0, ADCSOCA
TINV_VGRID_B_ADC_BASE	CONFIG_ADC3_BASE_ADDR	ADC_SOC_NUMBER0	ADC3_AIN0	ePWM0, ADCSOCA
TINV_VINV_B_ADC_BASE	CONFIG_ADC3_BASE_ADDR	ADC_SOC_NUMBER1	ADC3_AIN1	ePWM0, ADCSOCA
TINV_IINV_C_ADC_BASE (or SDFM)	CONFIG_ADC3_BASE_ADDR	ADC_SOC_NUMBER2	ADC3_AIN2	ePWM0, ADCSOCA
TINV_IINV_B_ADC_BASE (or SDFM)	CONFIG_ADC4_BASE_ADDR	ADC_SOC_NUMBER0	ADC4_AIN0	ePWM0, ADCSOCA
TINV_VBUS_ADC_BASE	CONFIG_ADC4_BASE_ADDR	ADC_SOC_NUMBER1	ADC4_AIN3	ePWM0, ADCSOCA

Table 3-2. ADC Mapping for E1 and E2 controlCARD With HSEC Board

HSEC Board	E1	E2
12	ADC0_AIN0	ADC1_AIN0
14	ADC0_AIN1	ADC1_AIN1
15	ADC0_AIN2	ADC0_AIN2
18	ADC1_AIN0	ADC1_AIN2
20	ADC1_AIN1	ADC1_AIN3
21	ADC1_AIN2	ADC0_AIN4
23	ADC1_AIN3	ADC0_AIN5
28	ADC2_AIN2	ADC3_AIN0
30	ADC2_AIN3	ADC3_AIN1
31	ADC3_AIN0	ADC2_AIN0
33	ADC3_AIN1	ADC2_AIN1
34	ADC3_AIN2	ADC3_AIN2
37	ADC4_AIN0	ADC2_AIN2
39	ADC4_AIN3	ADC2_AIN3

Table 3-3. ADC to PWM SOC Mapping for E2 controlCARD

ADC Type	ADC Base Address	ADC SOC Number	ADC Pin	Trigger
TINV_VGRID_A_ADC_BASE	CONFIG_ADC0_BASE_ADDR	ADC_SOC_NUMBER0	ADC0_AIN2	ePWM0, ADCSOCA
TINV_VINV_A_ADC_BASE	CONFIG_ADC0_BASE_ADDR	ADC_SOC_NUMBER1	ADC0_AIN4	ePWM0, ADCSOCA
TINV_IINV_A_ADC_BASE (or SDFM)	CONFIG_ADC0_BASE_ADDR	ADC_SOC_NUMBER2	ADC0_AIN5	ePWM0, ADCSOCA
TINV_TEMP_A_ADC_BASE	CONFIG_ADC1_BASE_ADDR	ADC_SOC_NUMBER0	ADC1_AIN0	ePWM0, ADCSOCB
TINV_TEMP_B_ADC_BASE	CONFIG_ADC1_BASE_ADDR	ADC_SOC_NUMBER1	ADC1_AIN1	ePWM0, ADCSOCB
TINV_TEMP_C_ADC_BASE	CONFIG_ADC1_BASE_ADDR	ADC_SOC_NUMBER2	ADC1_AIN2	ePWM0, ADCSOCB
TINV_TEMP_AMB_ADC_BASE	CONFIG_ADC1_BASE_ADDR	ADC_SOC_NUMBER3	ADC1_AIN3	ePWM0, ADCSOCB
TINV_VGRID_B_ADC_BASE	CONFIG_ADC2_BASE_ADDR	ADC_SOC_NUMBER0	ADC2_AIN0	ePWM0, ADCSOCA
TINV_VINV_B_ADC_BASE	CONFIG_ADC2_BASE_ADDR	ADC_SOC_NUMBER1	ADC2_AIN1	ePWM0, ADCSOCA
TINV_IINV_B_ADC_BASE (or SDFM)	CONFIG_ADC2_BASE_ADDR	ADC_SOC_NUMBER2	ADC2_AIN2	ePWM0, ADCSOCA
TINV_VBUS_ADC_BASE	CONFIG_ADC2_BASE_ADDR	ADC_SOC_NUMBER3	ADC2_AIN3	ePWM0, ADCSOCA
TINV_VGRID_C_ADC_BASE	CONFIG_ADC3_BASE_ADDR	ADC_SOC_NUMBER0	ADC3_AIN0	ePWM0, ADCSOCA
TINV_VINV_C_ADC_BASE	CONFIG_ADC3_BASE_ADDR	ADC_SOC_NUMBER1	ADC3_AIN1	ePWM0, ADCSOCA
TINV_IINV_C_ADC_BASE (or SDFM)	CONFIG_ADC3_BASE_ADDR	ADC_SOC_NUMBER2	ADC3_AIN2	ePWM0, ADCSOCA

Figure 3-18 shows a fully configured ADC0 instance.

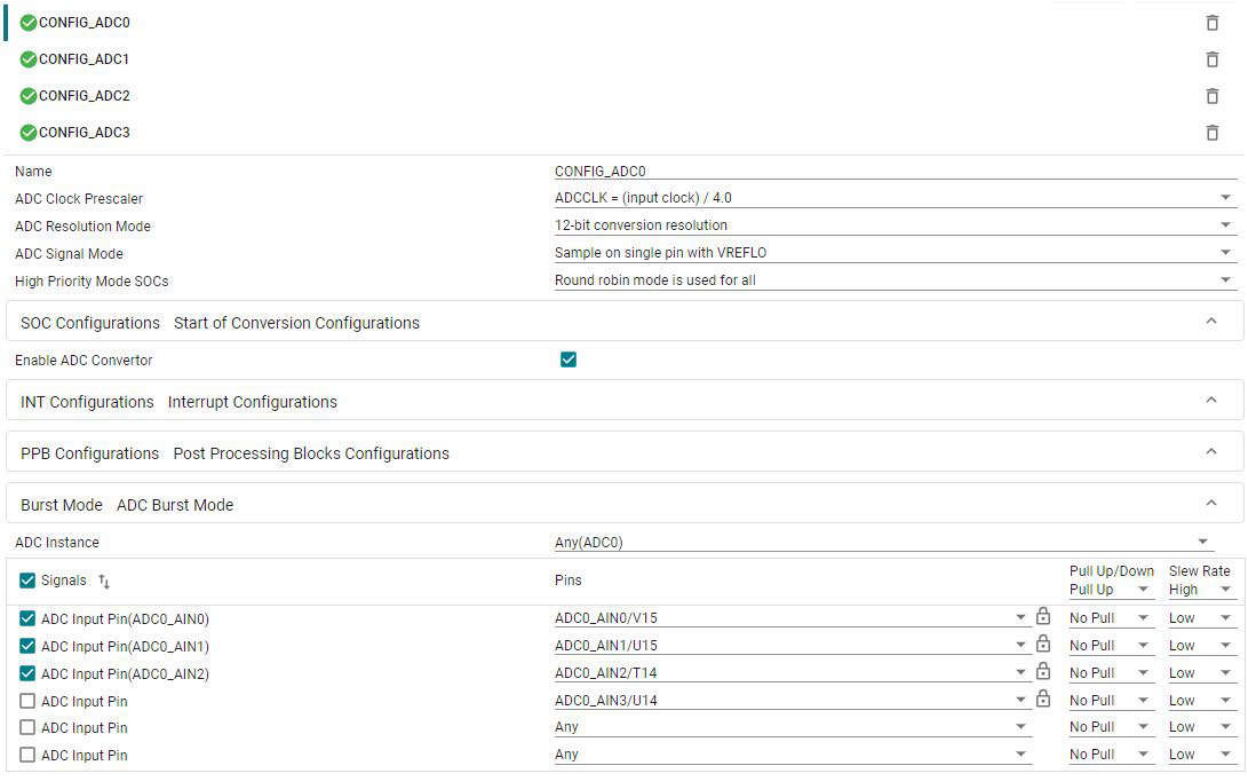


Figure 3-18. ADC Configuration

Set the ADC DAC reference voltage switches according to application requirements by checking in the respective board schematics document in the section - ADC and DAC Interfaces.

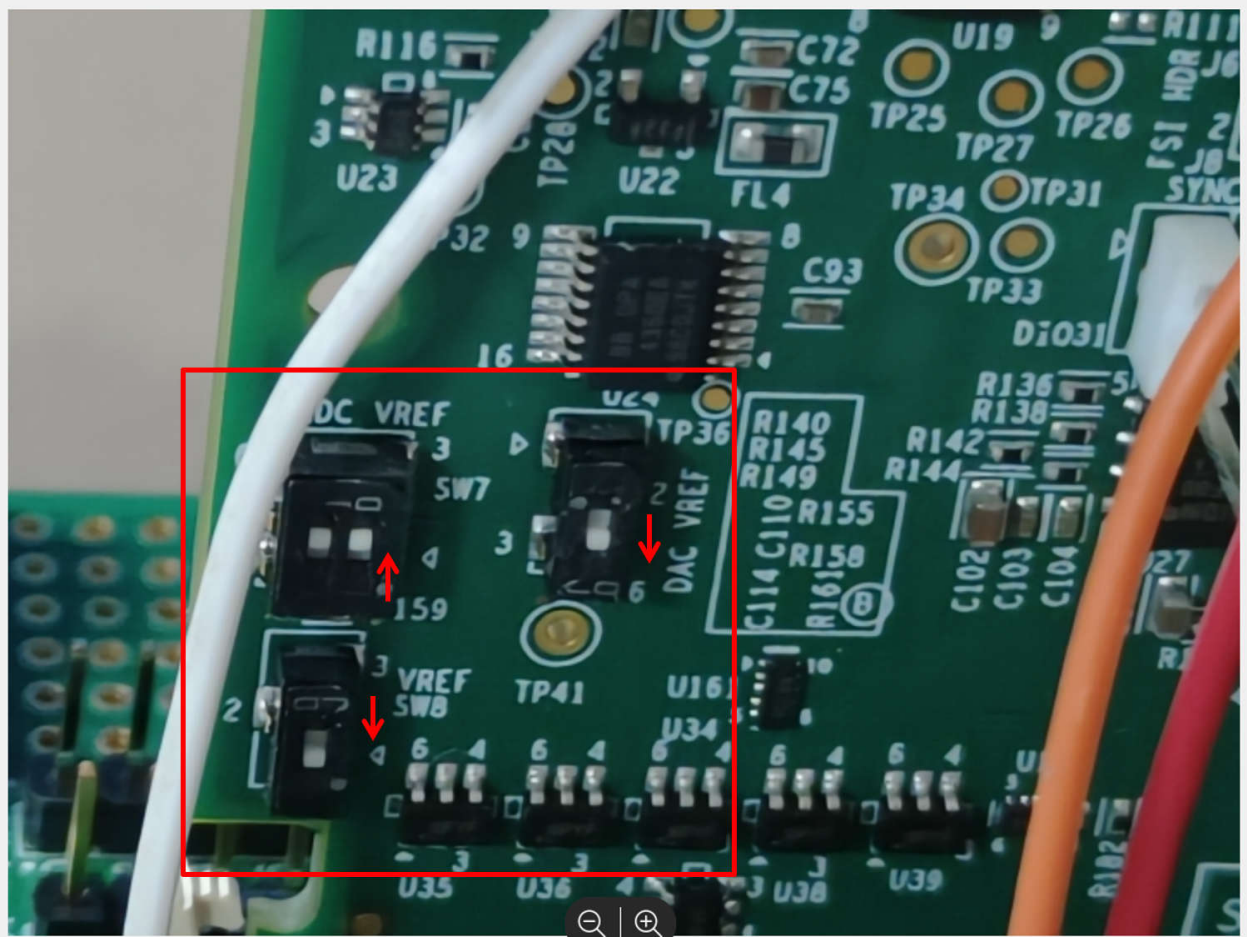


Figure 3-19. ADC and DAC Reference Switches in the AM263x controlCARD

3.2.6 CMPSS Configuration

The Comparator Subsystem (CMPSS) consists of analog comparators and supporting circuits that are useful for voltage and current trip monitoring. The current demo monitors 6 voltage units (grid side and inverter side combined) using ADC. Therefore, 6 CMPSS comparators are configured and connected to these respective ADCs. Each CMPSS has a reference DAC connected to the negative terminal. SysConfig provides a data field to modify this DAC value to set the threshold voltage for the trip. Table 3-4 explains the 6 CMPSS modules used. Since the ADC pinout is different between E1 and E2, the CMPSS and ADC mapping is also different from E1 and E2 versions of the controlCARD to the HSEC Board. These differences in the pinout are listed in Table 3-5.

Table 3-4. ADC to CMPSS Mapping for E1 controlCARD

ADC	CMPSS	DAC Reference	Trip Event
TINV_VGRID_A_ADC_BASE (ADC0_AIN2)	CMPSSA1:inH	3600	Input to +ve terminal goes above threshold in DCH
TINV_VINV_A_ADC_BASE (ADC1_AIN2)	CMPSSA3:inH	3600	Input to +ve terminal goes above threshold in DCH
TINV_VGRID_C_ADC_BASE (ADC2_AIN2)	CMPSSA5:inH	3600	Input to +ve terminal goes above threshold in DCH
TINV_VINV_C_ADC_BASE (ADC2_AIN3)	CMPSSA5:inL	3600	One shot Trigger not possible
TINV_VGRID_B_ADC_BASE (ADC3_AIN0)	CMPSSA6:inH	3600	Input to +ve terminal goes above threshold in DCH
TINV_VINV_B_ADC_BASE (ADC3_AIN1)	CMPSSA6:inL	3600	One shot Trigger not possible

Table 3-5. ADC to CMPSS Mapping for E2 controlCARD

ADC Type Name	ADC Pin	CMPSS	DAC Reference	Trip Event
TINV_VGRID_A_ADC_BASE	ADC0_AIN2	CMPSSA1:inH	3600	Input to +ve terminal goes above threshold in DCH
TINV_VINV_A_ADC_BASE	ADC0_AIN4	CMPSSB0:inH	3600	Input to +ve terminal goes above threshold in DCH
TINV_VGRID_B_ADC_BASE	ADC2_AIN0	CMPSSA4:inH	3600	Input to +ve terminal goes above threshold in DCH
TINV_VINV_B_ADC_BASE	ADC2_AIN1	CMPSSA4:inL	3600	One shot Trigger not possible

Table 3-5. ADC to CMPSS Mapping for E2 controlCARD (continued)

ADC Type Name	ADC Pin	CMPSS	DAC Reference	Trip Event
TINV_VGRID_C_ADC_BASE	ADC3_AIN0	CMPSSA6:inH	3600	Input to +ve terminal goes above threshold in DCH
TINV_VINV_C_ADC_BASE	ADC3_AIN1	CMPSSA6:inL	3600	One shot Trigger not possible

Figure 3-20. Choosing the CMPSS Instance and Enabling the Module

Figure 3-21 illustrates the high comparator configuration for ADC1_AIN2 connected to the grid side voltage of phase A (TINV_VGRID_A_ADC_BASE).

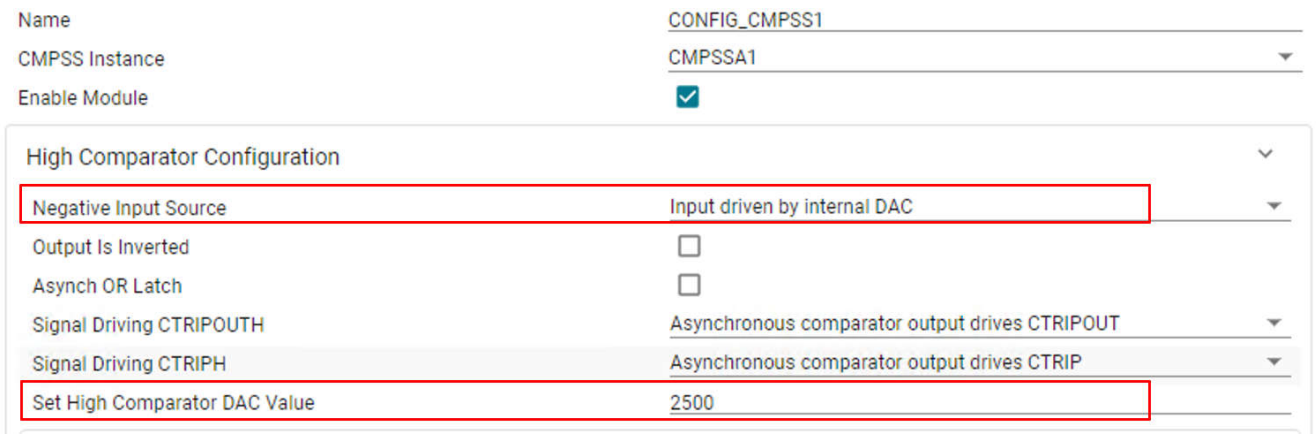
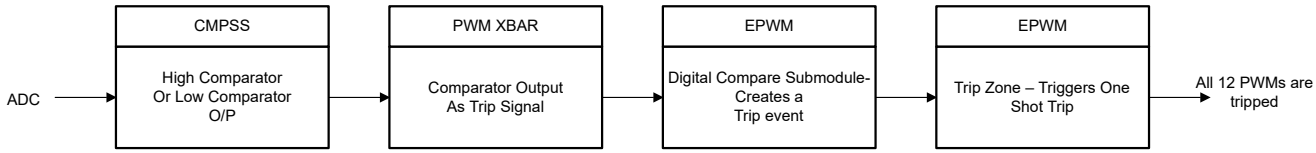


Figure 3-21. CMPSS High Comparator Configuration and Threshold Setting



The Comparator gives a CTRIPH, CTRIPL signal (high) when the threshold is crossed. This signal needs to be passed to the Trip Zone module of the EPWM with the help of the PWM XBAR system. Configure 6 PWM XBARS for every CTRIP of high comparator or low comparator.

3.2.7 EPWM XBAR Configuration

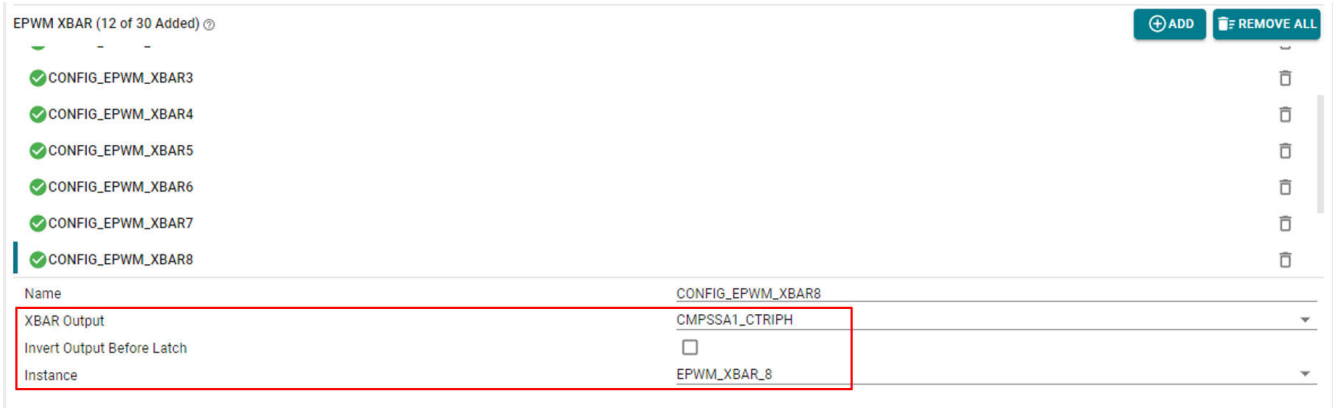


Figure 3-22. EPWM X-BAR Configuration

For the high comparators events, DCAEVT1 and DCBEVT1 events are generated when their respective trips are set high. Here EPWM_XBARx carry the trip signals. The mapping for the same is shown in Figure 3-23.

For low Comparators, DCAEVT2 and DCBEVT2 are generated. But this cannot generate a one-shot trip (EVT2 is only for CBC Trip).

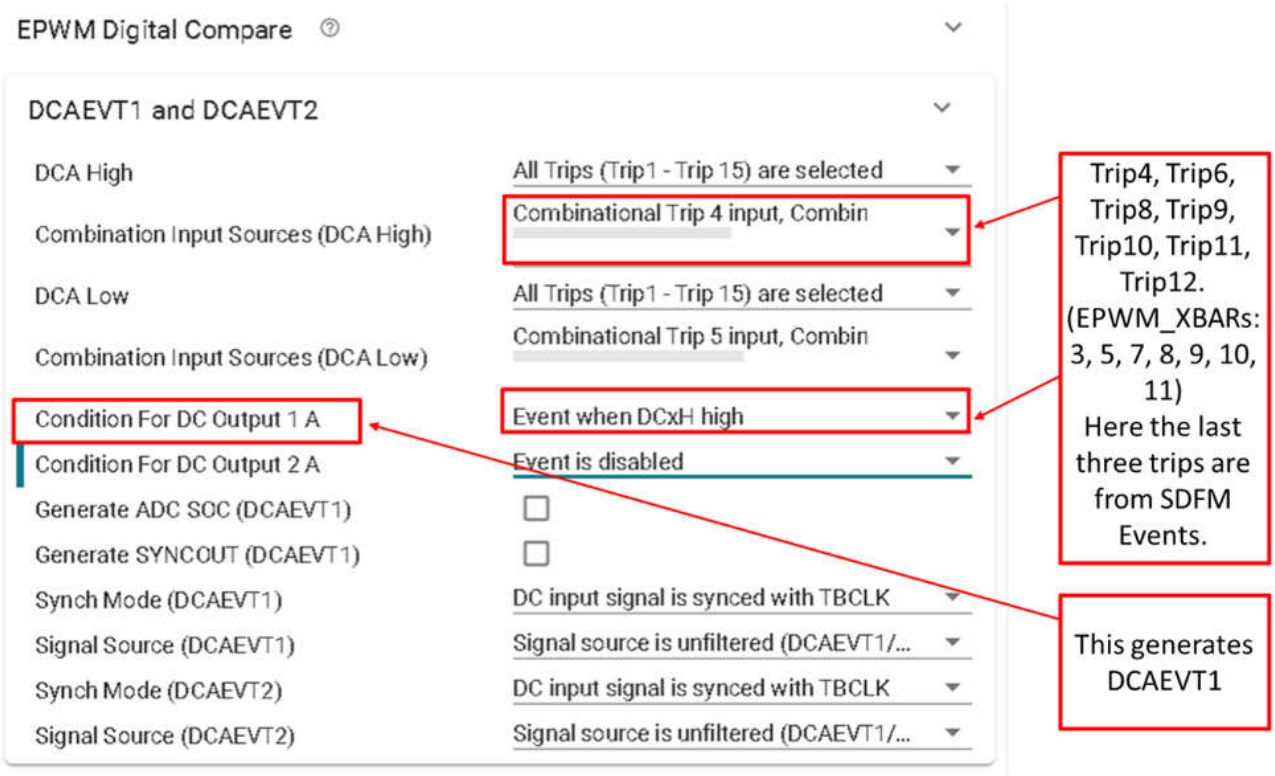


Figure 3-23. EPWM Digital Compare Submodule Configuration

Now that the events are generated, link the events to the One-Shot Trip Configuration. This Digital Compare and Trip Zone configuration is repeated for all the 6 PWM instances. Once configured, any phase voltage or current that crosses the specified threshold will trip all the PWMs in the system.

Figure 3-24. EPWM Trip Zone Submodule Configuration

3.2.8 ECAP Configuration

The IC AMC3306 is used as a sigma-delta modulator for the sensed current. The output of this IC is given as input to the SDFM module. This IC needs an input clock signal to create a synchronous modulated output. According to the IC data sheet, the maximum input clock frequency can be 21 MHz. Therefore, to generate maximum samples of data, an APWM waveform is created with a 50-ns period.

$$\text{APWM Period} = \left(\frac{\text{ClkFreq}}{\text{PWMFreq}}\right) - 1 = \left(\frac{200\ MHz}{20\ MHz}\right) - 1 = 9 \quad (2)$$

The APWM goes high at CTR = 0 event. ACMP (APWM Compare) is the value of CTR, when the PWM goes low. ACMP decides the duty cycle of APWM.

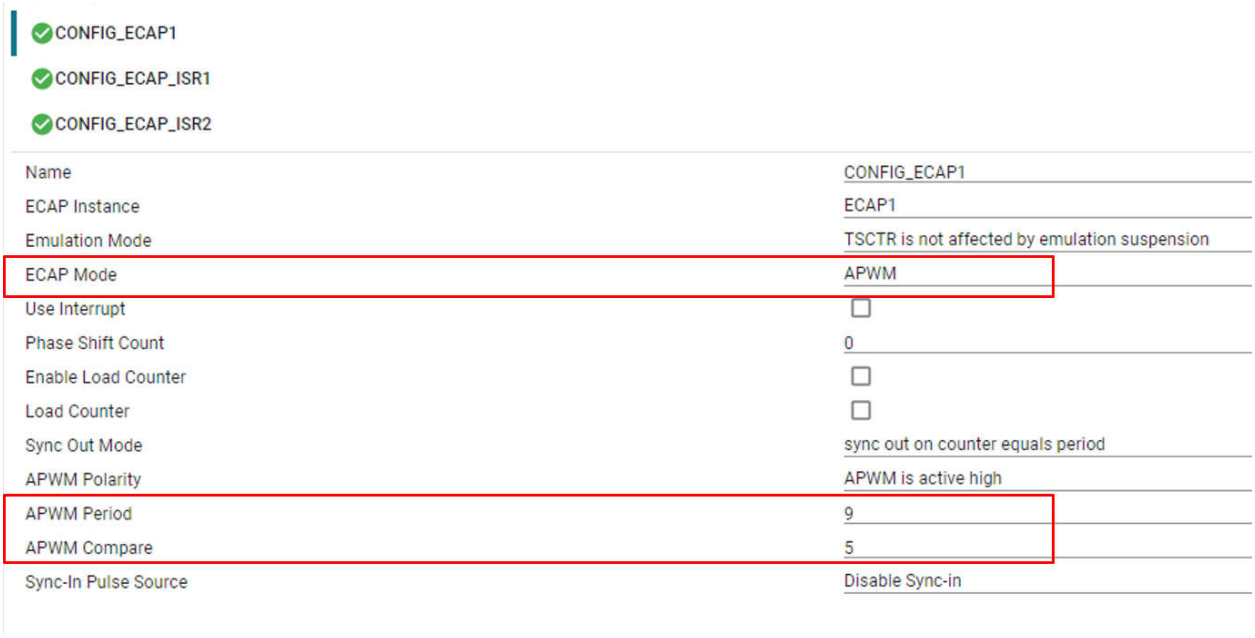


Figure 3-25. ECAP Configuration

3.2.9 Output XBAR Configuration

ECAP is configured to provide clock for the sigma-delta modulator and is routed to the GPIO pin using the output XBAR system. There are a total of 16 output XBAR pins in AM263. According to the TIDA-01606 schematics, clock out must be passed from HSEC-DOC pin 75, which maps to OUTPUTXBAR3 (B10 of AM263).

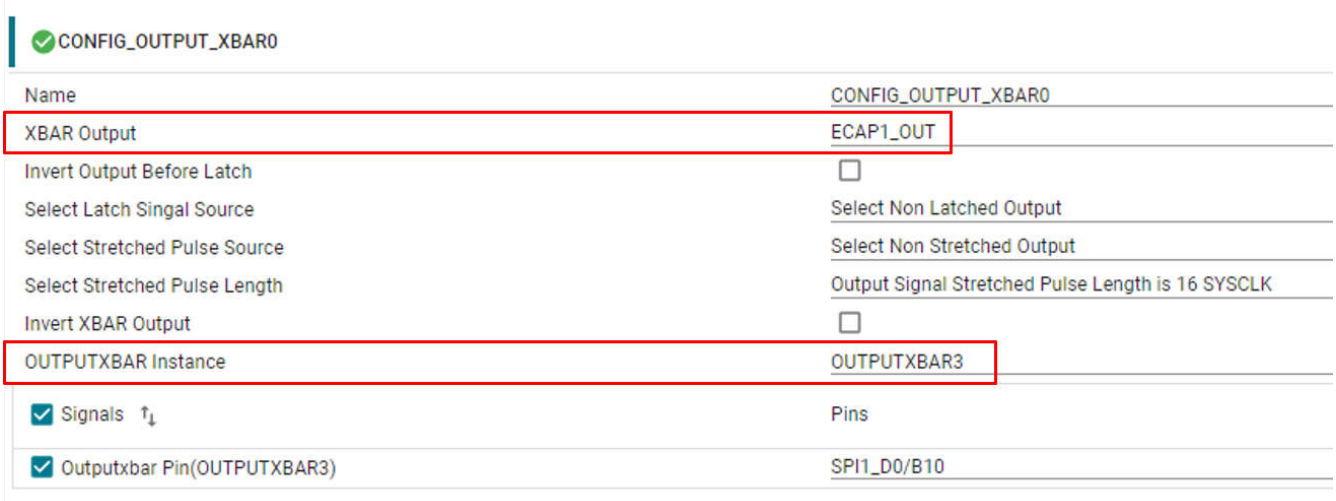


Figure 3-26. Output X-Bar Configuration

3.2.10 Input XBAR Configuration

Fault input from any of the three gate drivers is configured as input GPIOs *FLT_A*, *FLT_B*, *FLT_C*. When going high, these signals represent a fault condition in the gate driver hardware. This input is routed as a trip input to the one-shot Trip Zone of EPWM through input XBAR → EPWM XBAR.

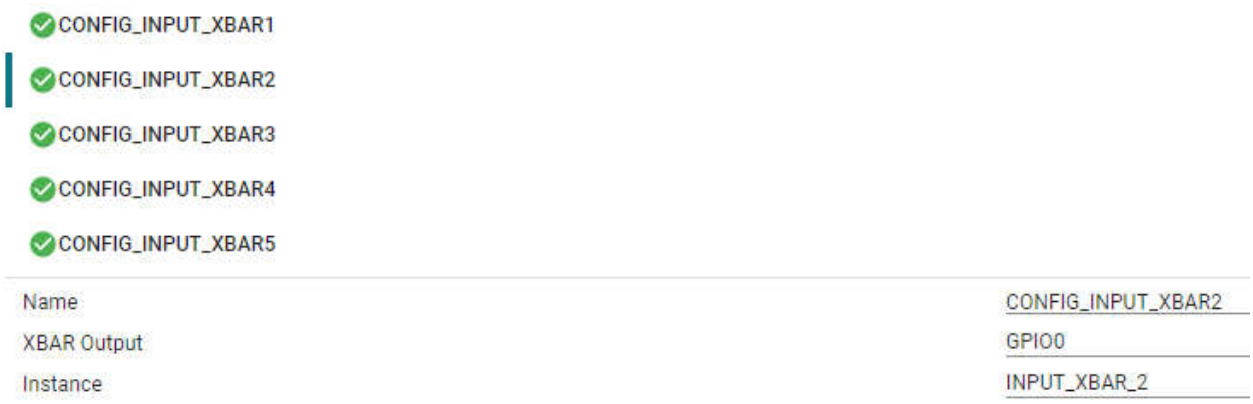


Figure 3-27. Input X-Bar Configuration

From the EPWM XBAR, trip input is given to the digital compare submodule to create DCAEVT1 and DCBEVT1. These events are again configured as an input to the one-shot trip in the Trip Zone submodule. This procedure is explained in Section 3.2.6.

3.3 Interrupts and Lab Structure

Testing high-voltage reference designs is usually performed in different stages beginning with open loop tests in low voltage where board protection is disabled. In this stage the basic functionality – for example, interrupt and task switching, ADC sensing, PWM outputs, and so on – are tested without going deep into the control loop functions. Once the basic functionality is established, board protection logics are introduced to validate the high-voltage functionality. As the voltage increases, the noise in the system increases, hence the software needs to retains the functionality even in high-noise conditions. Simultaneously the software needs to trip the PWMs in case of any undesired overvoltage or overcurrents in the system that degrade or destroy the design. The next stage involves closing the loop to test the full functionality of the design.

The software is designed to switch between these stages for ease of testing and validating the firmware. System initializations and background tasks are common across all the stages. The source code in the ISR1 has the stages differentiated using the macro *LAB_NUMBER*. This design has seven stages of testing via Labs, where Lab 1 and Lab 2 are open loop tests of the inverter mode of operation. Lab 3 has the closed current loop implemented for the inverter mode using resistive load. Lab 4 has the closed current loop implemented for the inverter mode with Grid-tied operation. Lab 5 and Lab 6 are for open loop and closed loop execution for PFC mode of operation, respectively. Lab 7 has the closed current loop and closed voltage loop implemented for the PFC mode with Grid-tied operation. The description of these labs is shown in Section 3.3.1.

The project consists of two ISRs (ISR1 and ISR2) with ISR1 being the fastest and non-nestable ISR. The content of the ISRs and their trigger of execution is discussed in Section 3.3.1 and Section 3.3.3.

3.3.1 ISR1

ISR1 is reserved for the control loop and the PWM update. ISR1 is triggered by the *LEG1_PWM_BASE → EPWM_INT_TBCTR_U_CMPC* event. ISR1 has a total of seven lab components. Only one lab can be chosen to run at one time. Each lab has APIs enabled depending on the testing requirements as described in Table 3-6.

Since the control loop needs to be pre-tested for the parameters being used, Lab 1 and Lab 2 are tested to make sure the PWM update and ADC sensing are stable in an open loop before enabling the closed loop control. Lab 3 contains a closed current loop implementation and is discussed in Section 3.3.2.

The software of this reference design is organized in seven labs. Table 3-6 lists the labs and how the labs were tested.

Table 3-6. All ISR1 Labs

Lab Number	Description	Comments	Test Environment
1	INV: PWM and ADC check	Test the PWM driver, ISR structure and execution rate, can be run on a controlCARD. Unit test protection mechanisms. Test ADC mapping and reading of conversion data.	controlCARD
2	INV: Open loop check	PWM Check, ADC check, protection check, inverter mode DC bus connected and resistive star network as load	controlCARD + Power Stage Hardware
3	INV: Closed current loop, resistive load connected at AC		controlCARD + Power Stage Hardware
4	INV: Closed current loop, grid connected test inverter mode		controlCARD + Emulated power stage under Hardware In-the Loop
5	PFC: Three-phase AC source, resistive load at DC, open loop check	1. Check if the vGridRms, iGridRms, vBus measurements are correct • Check if PLL is locked	controlCARD + Power Stage Hardware
6	PFC: Closed current loop, resistive load connected at DC, three phase AC		controlCARD + Power Stage Hardware
7	PFC: Closed voltage loop + current loop , resistive load connected at DC, three phase AC		controlCARD + Power Stage Hardware

Figure 3-28 and Figure 3-29 cover the major differences between Lab 1 (open loop test) and Lab 3 (closed loop test).

```
void TINV_runISR1_lab1(void)
{
    TINV_readCurrentAndVoltageSignals();

    TINV_generateInternalReference();

    TINV_vdInv_pu = TINV_vdInvRef_pu;
    TINV_vqInv_pu = TINV_vqInvRef_pu;
    TINV_vzInv_pu = 0.0f;

    DQ0_ABC_run(&TINV_vInv_dq0,
                TINV_vdInv_pu, TINV_vqInv_pu, TINV_vzInv_pu,
                TINV_sine, TINV_cosine);

    TINV_duty_A_pu = TINV_vInv_dq0.a;
    TINV_duty_B_pu = TINV_vInv_dq0.b;
    TINV_duty_C_pu = TINV_vInv_dq0.c;

    TINV_HAL_updatePWMDuty(TINV_duty_A_pu, TINV_duty_B_pu, TINV_duty_C_pu);

    TINV_runSPLL(TINV_internalRef_dq0_pos.d,
                 TINV_internalRef_dq0_neg.d,
                 TINV_internalRef_dq0_pos.q,
                 TINV_internalRef_dq0_neg.q);
}
```

The ADC is used to sense Signals, but the sensed values are not used further in the ISR

The Vd_Inv and Vq_Inv values are constants and are not changed based on the sensed currents.

Figure 3-28. Lab 1 in ISR1

```
void TINV_runISR1_lab3(void)
{
    TINV_readCurrentAndVoltageSignals();

    TINV_runTransformOnSensedSignals();

    TINV_generateInternalReference();

    TINV_runCurrentLoop();

    DQ0_ABC_run(&TINV_vInv_dq0,
                TINV_vdInv_pu, TINV_vqInv_pu, TINV_vzInv_pu,
                TINV_sine, TINV_cosine);

    TINV_duty_A_pu = TINV_vInv_dq0.a;
    TINV_duty_B_pu = TINV_vInv_dq0.b;
    TINV_duty_C_pu = TINV_vInv_dq0.c;

TINV_HAL_updatePWMDuty(TINV_duty_A_pu, TINV_duty_B_pu, TINV_duty_C_pu);

    TINV_runSPLL(TINV_internalRef_dq0_pos.d,
                 TINV_internalRef_dq0_neg.d,
                 TINV_internalRef_dq0_pos.q,
                 TINV_internalRef_dq0_neg.q);
}
```

The SDFM and ADC sense required Signals. These Inverter Currents and Grid Voltages are converted to Id , Iq, vd_grid, vq_grid using ABC to DQ Transforms

The current loop Processed these Id , Iq, vd_grid and vq_grid values based on the Id Reference value to give required Vd_Inv and Vq_Inv

The Final computed vd_Inv and vq_Inv are converter into A,B,C to be given as duty cycles to the PWMs

Figure 3-29. Lab 3 in ISR1

3.3.2 ISR1 - Lab 3

The initial task of Lab 3 – ISR1 is to sense the ADC and SDFM signals. The signals sensed on the inverter side of the circuit are defined using the term *Inv* and the signals sensed on the grid side of the circuit are defined using the term *Grid*. The three phase currents I_a, I_b, and I_c are sensed and the values are recorded using the SDFM module. The current values are same on the grid side and inverter side. Therefore, only the inverter side values are sensed through SDFM and are equated with the grid side values:

1. *TINV_iInv_A_sensed_pu = TINV_iGrid_A_sensed_pu;*
2. *TINV_iInv_B_sensed_pu = TINV_iGrid_B_sensed_pu;*
3. *TINV_iInv_C_sensed_pu = TINV_iGrid_C_sensed_pu;*

These values are converted into I_d and I_q using the ABC_DQ software blocks (Clarke's and Park's Transformation). These formatted input grid currents I_d and I_q, along with the reference I_d values are used to calculate the required output grid voltages `vd_Inv_out` and `vq_Inv_out`, as shown in Figure 3-30.

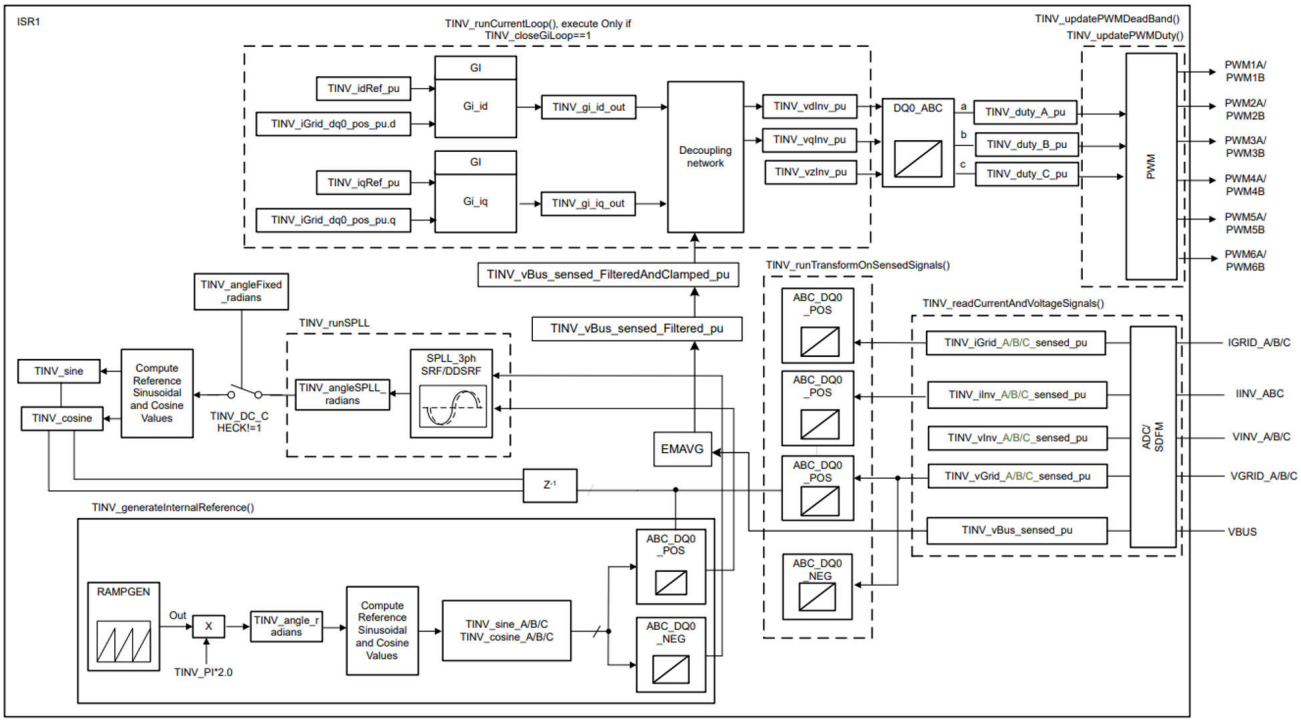


Figure 3-30. ISR1 Software Flow Block Diagram

The current I_d reference value is 0.005 by default, which corresponds to an output current of 500 mA. To increase the overall current output from the control loop, increase the I_d reference value. An I_d reference of 0.01 corresponds approximately to 1 A of overall output current.

Convert `vd_out` and `vq_out` into three-phase values by applying the DQ_ABC software blocks (Inverse Park's and Inverse Clark's Transforms). The obtained three phase values are formatted into duty cycles of PWM waveforms. These duty cycle values are given as input to the CMPA of their respective EPWMs, as shown in the following list.

1. Duty_ticks_A → CMPA of TINV_Q1_Q3_A_PWM and TINV_Q1_Q4_A_PWM
2. Duty_ticks_B → CMPA of TINV_Q1_Q3_B_PWM and TINV_Q1_Q4_B_PWM
3. Duty_ticks_C → CMPA of TINV_Q1_Q3_C_PWM and TINV_Q1_Q4_C_PWM

If the A value obtained from the DQ_ABC transform is positive (greater than 0), this represents the positive half cycle of the sine wave. Therefore, the PWM instance *TINV_Q1_Q3_A_PWM* is enabled and instance *TINV_Q1_Q4_A_PWM* is disabled to carry out switching for the positive half cycle.

If the A value obtained from the DQ_ABC transform is negative (less than 0), this represents the negative half cycle of the sine wave. Hence, the PWM instance *TINV_Q1_Q3_A_PWM* is disabled and instance *TINV_Q1_Q4_A_PWM* is enabled to carryout switching for the negative half cycle.

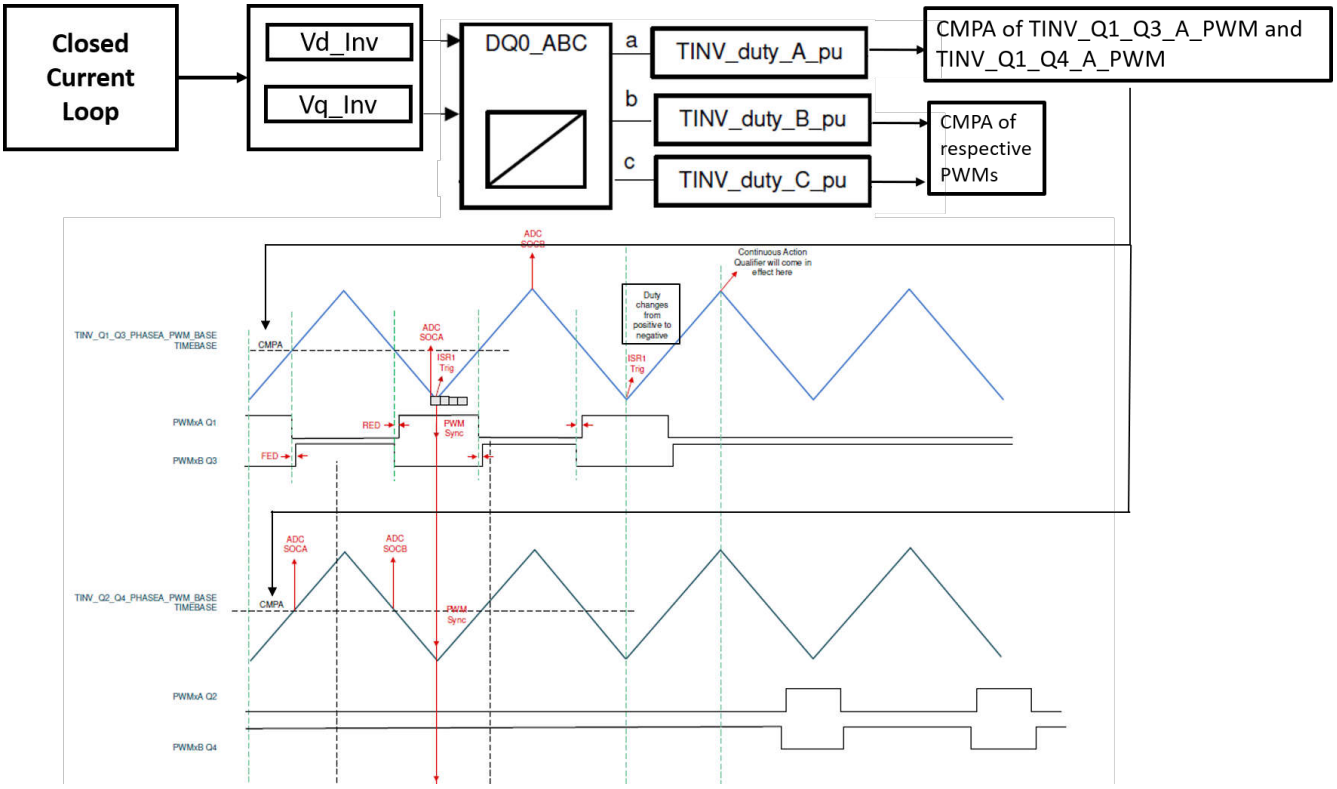


Figure 3-31. PWM Update through ISR1

In the demonstration, SRF Software PLL is used to lock the reference angle from the ramp generator. The sine and cosine of this PLL angle are used for computing ABC_DQ and DQ_ABC transforms. Figure 3-30 shows the block diagram.

3.3.3 ISR2

ISR2 is triggered by CPU timer INT which is initiated by an overflow on the CPU timer. ISR2 is used to run housekeeping functions such as doing a running average on the currents and voltage signals to remove noise and running the slew rate function for commanded references. Figure 3-32 describes the ISR2 Block diagram.

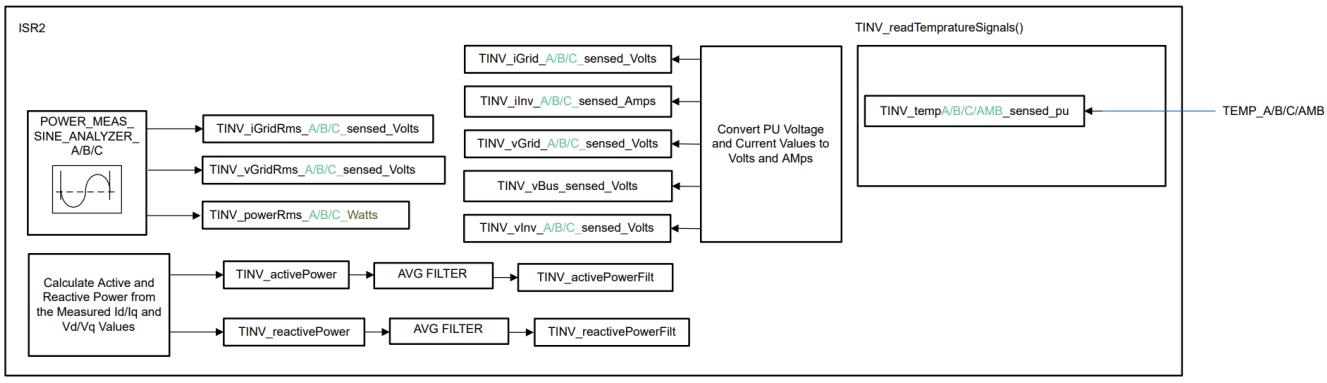


Figure 3-32. ISR2 Software Flow Block Diagram

3.4 Protection Scheme

There are few sensitive components in the hardware circuit especially the gate drivers, MOSFETS, and so forth, which need protection from high voltages and currents. To drive MOSFETS with equal power, TI-designed gate drivers are used. Three gate driver cards, each with two ISO5852S and two UCC5320 gate drivers are present in the hardware. These gate drivers provide soft turnoff (STO) during short circuit and a fault alarm upon desaturation detection of MOSFETs and is signaled on FLT. More information on the gate drivers is provided in the *TIDA-01606: 10-kW, Bidirectional Three-Phase Three-Level (T-Type) Inverter and PFC Reference Design* design guide.

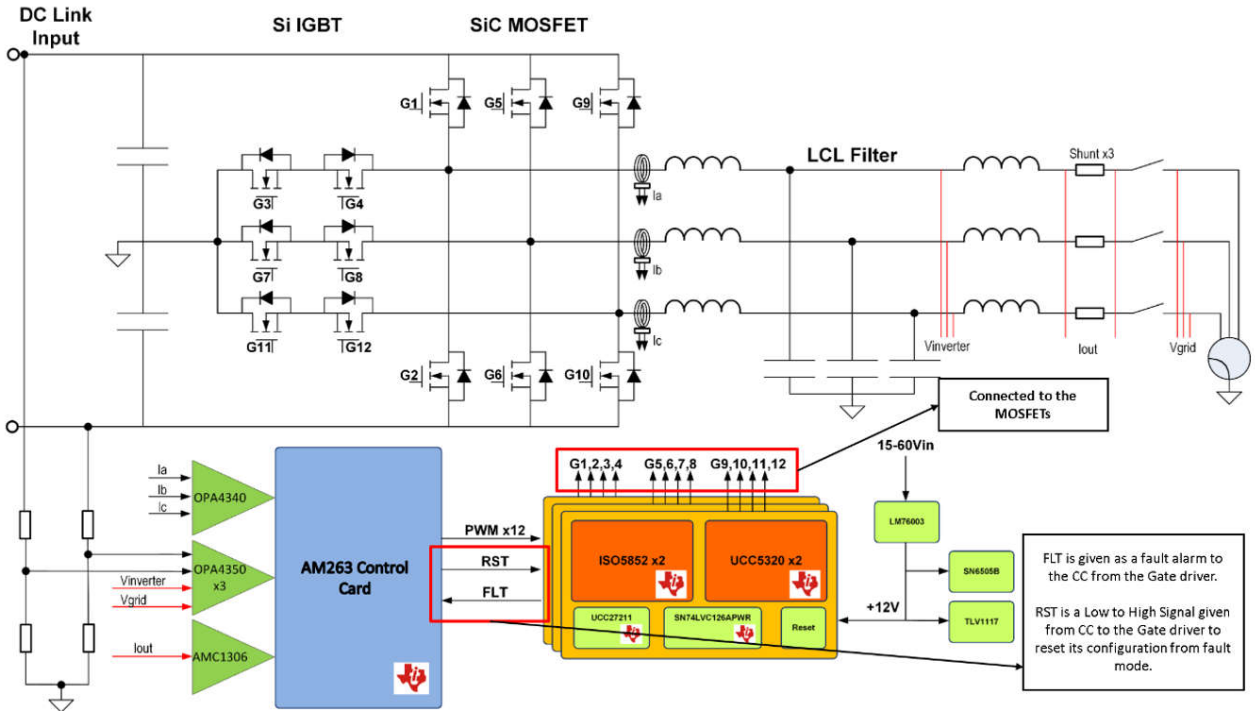


Figure 3-33. TIDA-01606 Block Diagram With Gate Driver Fault Detection

Table 3-7 lists the trip configurations made for overvoltage protection, overcurrent protection, and gate driver fault indication to protect the MOSFETs from any overvoltage, overcurrent, or surge events. Along with this, the MOSFETs are connected to heat sinks to protect from high temperatures in the circuit when running at high voltage.

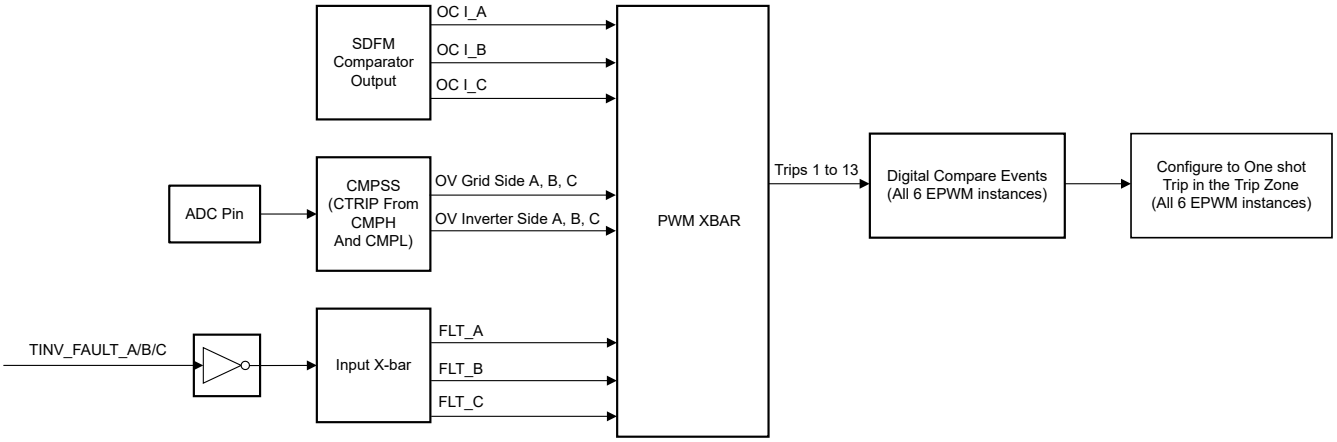


Figure 3-34. Board Protection Block Diagram

Table 3-7. PWM XBAR to Trip Mapping

Cause for Trip	PWM_XBAR (PWM Crossbar)	Trip Number (Digital Compare – EPWM)
FLT_A (GateDriver_A)	PWM XBAR0	Trip1
FLT_B (GateDriver_B)	PWM XBAR1	Trip2
FLT_C (GateDriver_C)	PWM XBAR2	Trip3
InvSide_OverVoltage_A	PWM XBAR3	Trip4
InvSide_OverVoltage_C	PWM XBAR4	Trip5
GridSide_OverVoltage_C	PWM XBAR5	Trip6
InvSide_OverVoltage_B	PWM XBAR6	Trip7
GridSide_OverVoltage_B	PWM XBAR7	Trip8
GridSide_OverVoltage_A	PWM XBAR8	Trip9
OverCurrent_A	PWM XBAR10	Trip11
OverCurrent_B	PWM XBAR11	Trip12
OverCurrent_C	PWM XBAR12	Trip13

By default, Trip 4 to Trip 13 are connected to the one-shot Trip Zone for immediate trip. Trip 1 to Trip 3 (fault detection from Gate Drivers) are disabled from the digital compare event. To enable, check the combinational trip inputs in the digital compare submodule of EPWM in SysConfig.

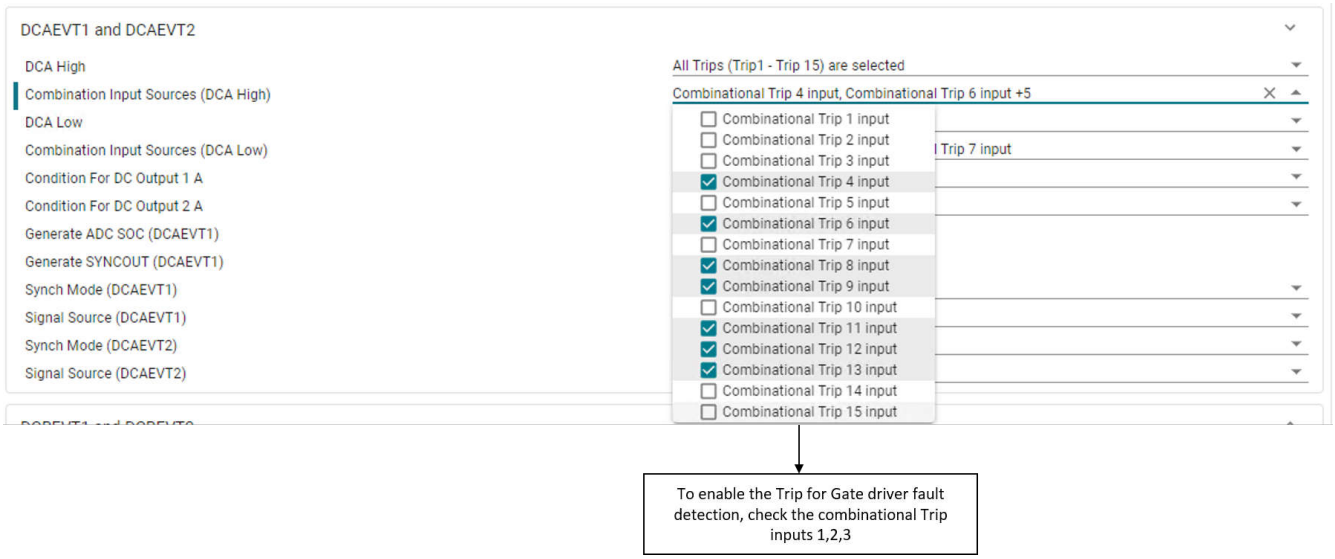


Figure 3-35. Trip Enable for the Fault Detection from Gate Drivers

`TINV_updateFaultStaus()` function is called periodically in a slow background task to update trip flags and reset the latch, if needed. If a trip event has occurred, the PWM needs trip flags to be cleared separately. This part is typically handled in the ISR by calling `TINV_clearPWMTrips()`.

B1 task updates the fault status of system. This setting is enabled by setting the macro *TINV_PROTECTION* to `TINV_PROTECTION_ENABLED`. This is disabled for Lab 1 by default and enabled for other Labs. More information about this is described in Section 3.4. The fault update continuously monitors PWM XBARs connected to the comparator trip outs, input XBAR trips, and SDFM comparator trip outs. When any trips are recorded in these sections, the *TINV_boardFaultFlags* structure is updated accordingly. This feature is helpful for debugging if the designer does not see any PWM waveforms at the output. The functions in Table 3-8 indicate overvoltage, overcurrent, and gate driver fault status.

Table 3-8. Fault APIs in the Software

Faults	APIs
Overcurrent Fault – Inverter and Grid Side	TINV_HAL_get_iInvA_overCurrentFlag TINV_HAL_get_iInvB_overCurrentFlag TINV_HAL_get_iInvC_overCurrentFlag
Overvoltage Fault – Inverter Side	TINV_HAL_get_VInvA_overVoltageFlag TINV_HAL_get_VInvB_overVoltageFlag TINV_HAL_get_VInvC_overVoltageFlag
Overcurrent Fault – Grid Side	TINV_HAL_get_VGridA_overVoltageFlag TINV_HAL_get_VGridB_overVoltageFlag TINV_HAL_get_VGridC_overVoltageFlag
Gate Driver Fault	TINV_HAL_get_faultAFlag TINV_HAL_get_faultBFlag TINV_HAL_get_faultCFlag

Another way to debug the trip status is to observe the register 0x52000C18 PWMXBAR status in the memory browser. A particular trip is identified by identifying the bits of this register that are set at the positions displayed in Table 3-9.

Table 3-9. PWM XBAR Bit Positions for Different System Faults

PWM_XBAR	PWM XBAR Output Status Register Bit Mapping
PWMXBAR_FLT_A	0
PWMXBAR_FLT_B	1
PWMXBAR_FLT_C	2
PWMXBAR_Inv_OverVoltage_A	3
PWMXBAR_Inv_OverVoltage_B	6
PWMXBAR_Inv_OverVoltage_C	4
PWMXBAR_Grid_OverVoltage_A	8
PWMXBAR_Grid_OverVoltage_B	7
PWMXBAR_Grid_OverVoltage_C	5
PWMXBAR_OverCurrent_A	10
PWMXBAR_OverCurrent_B	11
PWMXBAR_OverCurrent_C	12

3.5 CPU Loading

The main control ISR with Lab 3 takes approximately 2.5 µs to execute when the core R50_0 is running at 400 MHz. This Lab 3 executes once every 20 µs. This consumes approximately 12.5% of CPU time. CPU loading for ISR1 in all labs is mentioned in Section 5.

3.6 Building, Loading, and Debugging the Firmware

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 that the correct target configuration file is set as Active under `targetConfigs` (*.ccxml file). Then, click *Run* → *Debug* to launch a debugging session. In the case of dual-CPU devices, a window can appear for the user to select the CPU on which the debug is to be performed. In this case, select R50_0. Then the project loads on the device and the CCS debug view becomes active. The code halts at the start of the main routine. In different labs, sometimes the currents and voltages measured or the control variables need to be verified by viewing the data in the graph window. To import the graph into the CCS view select *Tools* → *Graph*.

4 Optimizations Implemented

The following techniques are used to optimize the execution time of ISR in the inverter demonstration:

- Build Settings are changed to Release Mode with Optimizations.
  - Release Mode has Optimizations that can be configured according to the requirement.
  - Options available: O[0 | 1 | 2 | 3 | fast | g | s | z]
  - For more information refer to the *Optimization Options* section of the *Compiler Tools User Manual*.
- Code added to TCM for faster execution. TCM is low latency and instructions accessed through TCM consume a single CPU cycle.
  - ISR is made to execute in TCM for faster code. All the variables being executed inside the ISR are made global and added to TCM.
  - custom sections are added in the TCM part in the linker.cmd file. Variables are functions added into separate TCM sections to avoid conflicts.
    - .controlfnc : {} palign(8) > R5F_TCMA
    - .controldata : {} palign(8) > R5F_TCMB
  - __section__("SECTION_NAME") attribute is used in the function prototype to add it to the TCM.
  - always_inline attribute is used to force the APIs into inline functions.
  - Figure 4-1 lists the TCM memory consumption for Lab3

```
MEMORY CONFIGURATION

         name             origin    length      used     unused   attr    fill
----------------------  --------  ---------  --------  --------  ----  --------
  R5F_VECS              00000000  00000040  00000040  00000000  RWIX
  R5F_TCMA              00000040  00007fc0  00007d10  000002b0  RWIX
  R5F_TCMB              00080000  00008000  000014e8  00006b18  RWIX
  FLASH                 60100000  00080000  00000000  00080000  RWIX
  OCRAM                 70040000  00040000  0000e3d0  00031c30  RWIX
  USER_SHM_MEM          701d0000  00004000  00000000  00004000  RWIX
  LOG_SHM_MEM           701d4000  00004000  00000000  00004000  RWIX
  RTOS_NORTOS_IPC_SHM_M 72000000  00003e80  00000000  00003e80  RWIX
```

Figure 4-1. TCMA and TCMB Memory Consumption When Lab 3 is Enabled

- Trigonometric calculations are replaced with r5_math_Trig library functions.
  - MCU_PLUS_SDK has r5_math_Trig library with optimized trigonometric operations compared to the standard mathlib.
  - sinf(TINV_angleSPLL_radians) → ti_r5fmath_sin(TINV_angleSPLL_radians, ti_r5fmath_PIconst, ti_r5fmath_sinCoef)
  - One sinf computation consumes approximately 0.45 µs of execution time. This can be replaced with the advanced trigonometric library APIs that consume 0.12 µs for one sine operation.
- Two 16-bit ADC reads if adjacent are converted to a 32-bit read and the values are split into 16 bits.
  - Best practice is to configure ADC sensing all in one group. For example, CSL_ADC_RESULT_ADCRESULT0 and CSL_ADC_RESULT_ADCRESULT1 can be configured and read together.
  - Read operations on a register are costly and this method reduces the read operations by one.

```
#define CSL_ADC_RESULT_ADCRESULT0                                    (0x00000000U)
#define CSL_ADC_RESULT_ADCRESULT1                                    (0x00000002U)
#define CSL_ADC_RESULT_ADCRESULT2                                    (0x00000004U)
#define CSL_ADC_RESULT_ADCRESULT3                                    (0x00000006U)
```

- Modifying MPU configuration for the register space for ADC, and PWM to enable posted writes. This configures the MPU Region Access Control Registers bit assignments.
  - All the register write operations are strongly ordered by default, that is, the data is first written to the register and the CPU waits for an *Ack* for successful write before the CPU can move onto the next write.
  - Removing the strongly ordered configuration enables continuous write operations. The configuration can be made cacheable or non-cacheable depending on the requirement. Usually, non-cacheable is preferred for faster execution because CPU cycles are not consumed to cache the data.

  - This can be combined with DMA to send more than one register data on time. (This ability is not implemented at the time of publication of this document).
  - For more information, see the MPU registers section of the Cortex-R4F Technical Reference Manual.
- More Compiler Optimizations:
  - Arm® Cortex® – R5 offloads arithmetic and logical calculations involving float to the FPU, enhancing the speed and operation. Choosing the floating-point *abi* as hard configures to use the FPU hardware to use for carrying the operations instead of software computations.

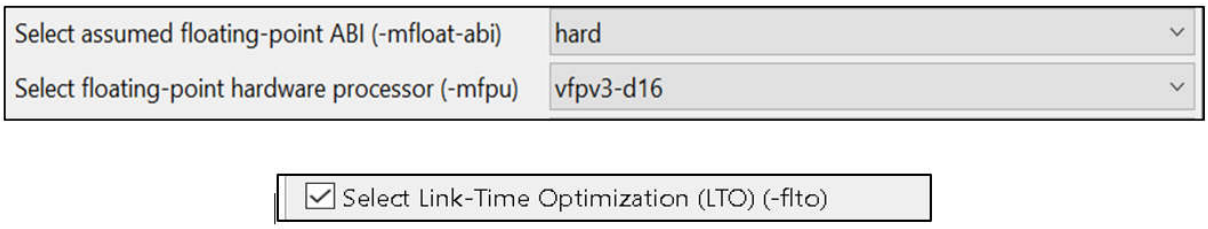


Figure 4-2. Compiler Optimization Settings

- Link Time Optimization:
  - Configured compiler optimization to -flto (link-time optimization - LTO) supported in TIARM Clang 2.1 LTS.
  - Compiler needs to configured to the supported version in your CCS Project to access -flto feature.
  - Go to Project Properties → Arm Compiler → Optimizations → Select Link-Time Optimization Option (LTO) as shown in Figure 4-2.
  - Once the settings are enabled and your project is rebuilt, check the map file for the ltoN option that signifies where the LTO option is being used.
  - The map file also shows that the .rodata output section is significantly smaller in the LTO-enabled -Oz build.
  - For more detailed information see the *Link Time Optimization* section of the *Compiler Tools User Manual*.

5 Testing and Results

5.1 Lab 1

Lab 1 can be run on the AM263x controlCARD and docking station. To choose Lab 1, change the macro as shown in the following code. All the other options can be left at default, for now.

```
#define TINV_LAB 1
```

Connect the EPWM0, EPWM1, EPWM2, EPWM3, EPWM6, EPWM7 pins of the controlCARD docking station to a logic analyzer to view the PWM Waveforms.

- EPWM0 – Pins 49, 51
- EPWM1 – Pins 53, 55
- EPWM2 – Pins 50, 52
- EPWM3 – Pins 54, 56
- EPWM6 – Pins 58, 60
- EPWM7 – Pins 62, 64

Figure 5-1 covers PWM waveforms for all six EPWM_A.

Figure 5-1. Lab 1 EPWM Output Waveforms

To test the ADC and SDFM sensing, send the acquired ADC result register or SDFM filter register output values to AM263x DAC while the controlCARD is placed inside of the TIDA-01606 setup. In Figure 5-2, the green line represents output channel A and the pink line represents the SDFM filtered output of sensed current for channel A.

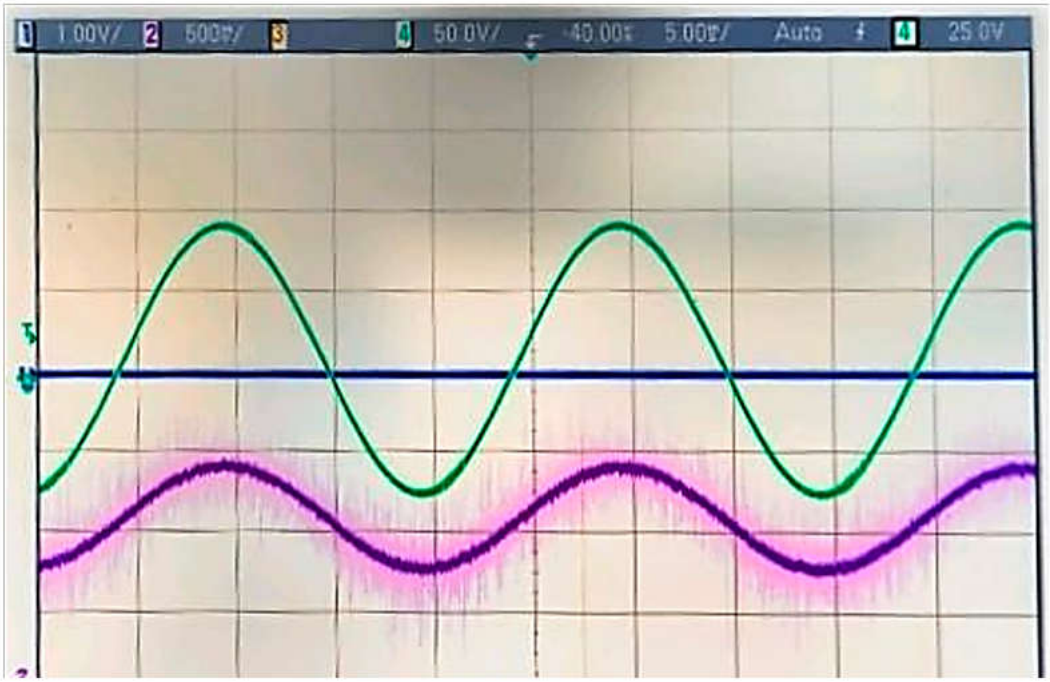


Figure 5-2. SDFM Channel A Sensed Signal Plotted on Oscilloscope Using DAC

5.2 Testing Inverter Operation

5.2.1 Lab 2 and Lab 3

Lab 2 is the inverter mode of operation in open loop, Lab 3 is the inverter mode of operation with closed current loop. The high voltage (800 VDC) is applied across terminals J1 and J2. 15-V auxiliary power supply is connected to terminal J33. Three-phase star connected resistive load is connected across terminals J3, J4, and J31. J32 is the neutral terminal which is left unconnected to the load.

Take the following steps into considerations while running Lab 3. For more information see the *TIDA-01606: 10-kW, Bidirectional Three-Phase Three-Level (T-type) Inverter and PFC Reference Design* design guide.

- To choose lab 3, change the macro as shown below. All the other options can be left at default for now.

```
#define TINV_LAB 3
```

- SFRA is not integrated in AM263; therefore, the current loop coefficients need to be manually adjusted in real time. The current compensator coefficients used for running the control loop are shown in the following code. Modify these coefficients to meet the necessary loop bandwidth and phase margin. The ideal coefficients with resistive load are slightly different than the one used for grid connection because the grid impedance is very low.

```
//
// PI Controller Settings from Compensation Designer
//
#define TINV_GI_PI_KP ((float32_t)0.14) //Inverter mode Kp for current loop
#define TINV_GI_PI_KI ((float32_t)0.1120860479) //Inverter mode Ki for current loop

//By default, Id reference:
#define TINV_IREF_DEFAULT    ((float32_t)0.005f)
TINV_idRef_pu = TINV_IREF_DEFAULT
```

- Set up an appropriate resistive load – around 200 Ω to start with – although the inverter mode can be started at no load as well. Slowly ramp the DC bus voltage Vbus to 800 V. Set the *TINV_clearPWMTrip* = 1, to clear the PWM trip signal. Now the switching action begins and sinusoidal voltages start appearing at the output. At this point the auxiliary power supply draws close to 500 mA.

- The *TINV_closeGiLoop* variable is enabled and closed current loop action begins. *TINV_idRef_pu* is the current command reference and by default, this reference is populated to a value of 0.005 pu at start-up. Slowly vary this to increase the output AC voltage and observe that the measured current tracks the commanded value.
- Verify *TINV_idRef_pu* and *TINV_iInv_dq0_pos.d* data in the watch window at low before proceeding to close the current loop in Lab 3. Slowly increase id_ref to 0.36 pu at 800-V input voltage to improve output power.
- Figure 5-3 shows this test setup.

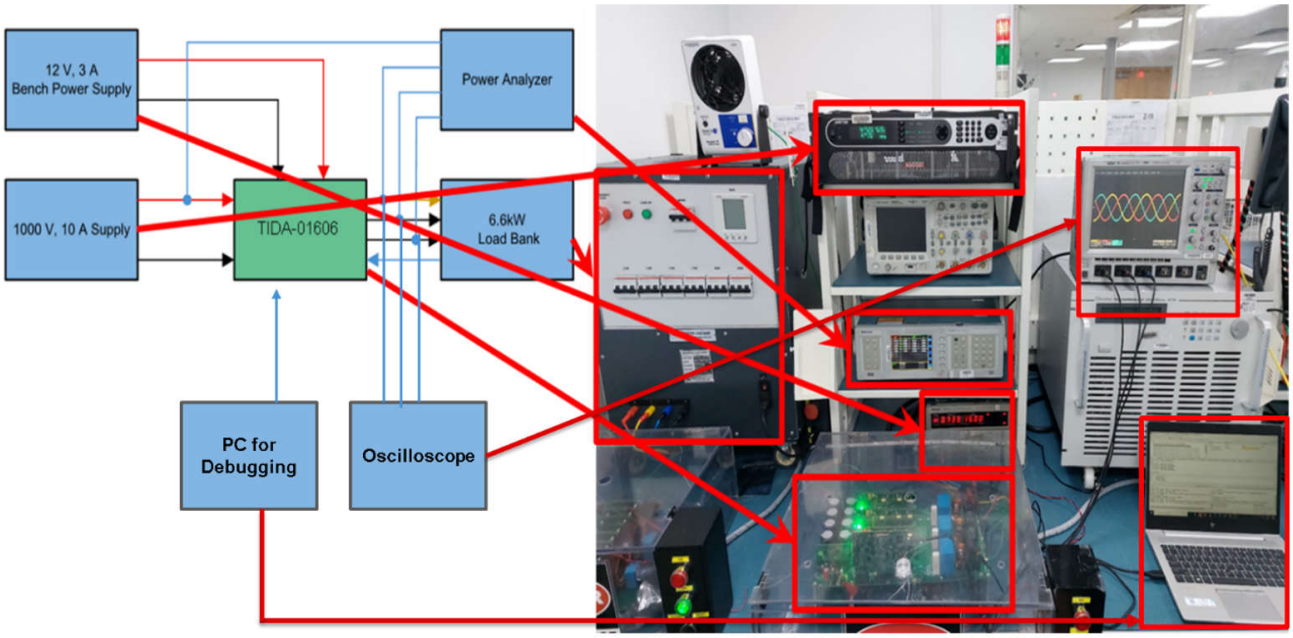


Figure 5-3. Testing Setup in the Lab for TIDA-01606

Figure 5-4 shows the output of the oscilloscope in three phases.

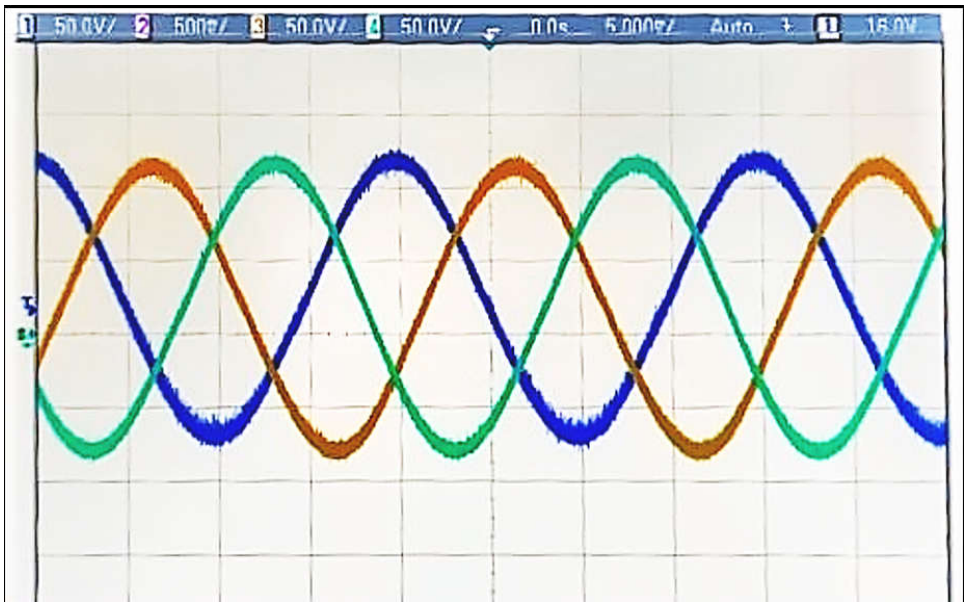


Figure 5-4. Output Waveforms A, B, C of Lab 3 From the Grid Side of TIDA-01606

The frequency is decreased to verify the stability of the current loop. The input voltage is now at 400 V and the voltage is quickly reduced to 100 V. The output current suddenly decreased, but the current loop adjusted the output waveform to get the output current back to the reference value as shown in Figure 5-5. Here the current reference value, *TINV_idRef_pu*, is 0.01 which corresponds to 1-A current.

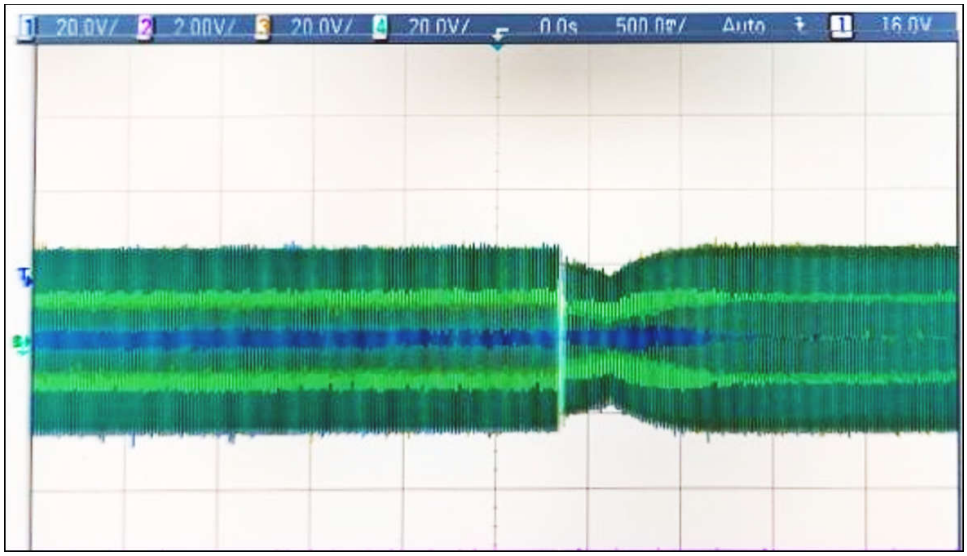


Figure 5-5. Control Loop Testing for Lab 3

5.2.2 Lab 3 Performance

Efficiency and Total Harmonic Distortion (THD) values of the string inverter for Lab 3 are captured in Table 5-1.

Table 5-1. THD Values in Interrupt Operation

Output Power	Efficiency (%)	Current - THD	Current - THD	Current - THD
(WATTS)		(PHASE-A)	(PHASE-B)	(PHASE-C)
982	96.6	1.78	1.63	1.58
1548	97.3	1.35	0.91	0.83
2576	97.6	1.02	0.82	0.75
3822	97.9	0.61	0.74	0.62

5.2.3 Inverter Interrupt Benchmarks

The execution time of ISR1 in Lab 1, Lab 2, and Lab 3 are tested using the CPU `CycleP_Counter` feature in the SDK. This approach gives accurate results compared to using GPIO toggle for calculating ISR execution time because the GPIO toggle has latency.

Table 5-2. ISR1 Maximum and Minimum CPU Execution Time Calculations

Lab Number	CPU Cycles - Maximum	CPU Time Consumed - Maximum	CPU Cycles – Minimum	CPU Time Consumed - Minimum
Lab 1	860	2.15 µs	829	2.072 µs
Lab 2	735	1.84 µs	708	1.77 µs
Lab 3	936	2.34 µs	890	2.225 µs

5.2.4 Lab 4

In Lab 4, the inverter is connected to the grid in real time. This lab is not tested by connecting to the grid.

5.3 Testing PFC Operation

Lab 5 to Lab 7 involve PFC operation for open loop and closed loop. Lab 5 is the PFC mode of operation in open loop, Lab 6 is the PFC mode of operation with closed current loop. Lab 7 is the PFC mode of operation in closed voltage and current loop.

The feed-forward and decoupling function is implemented inside ISR1 and added for all PFC Labs that use a current loop. Therefore, for the PFC mode, this is done in Lab 6 and Lab 7. For this feed-forward and decoupling function, filtered DC bus voltage is compared against a user-defined minimum bus voltage to calculate a

clamped filtered DC bus voltage. This is also done inside ISR1. This clamped filtered DC bus voltage and the current controller output are finally used to implement the feed-forward and decoupling function. The overcurrent and overvoltage protection is added.

Figure 5-6 shows the hardware setup, the DC terminals J1 and J2 are connected to a resistive load. A 15-V auxiliary power supply is connected to terminal J33. A three-phase AC source is connected across terminals J3, J4, and J31 (A, B, and C). J32 is the neutral.

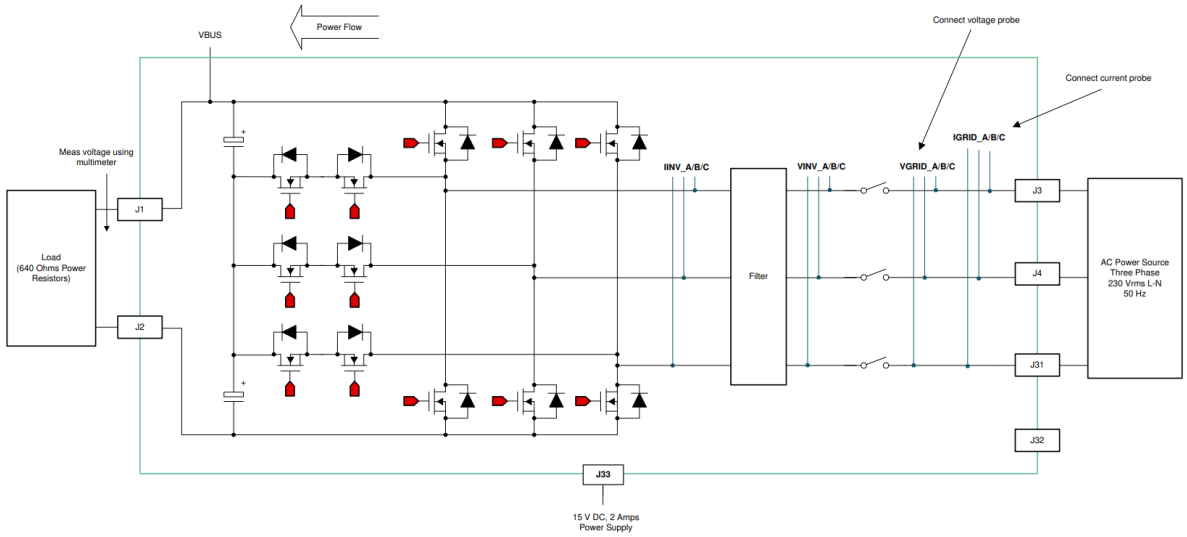


Figure 5-6. PFC Mode Test Setup

5.3.1 Lab 5

This lab checks sensing and no switching action occurs until *clearPWMTrip* is set to 1. Set the project to Lab 5 by changing the *LAB_NUMBER* in the user settings file. Under this condition, the converter operates as a rectifier and rectified current can be observed being drawn without any power factor correction. SPLL locking can also be safely verified in this build. Make sure the grid frequency is specified correctly, the grid frequency can be changed by using the *TINV_AC_FREQ_HZ* macro. The relays are closed during initialization. Figure 5-7 illustrates the working flow of Lab 5 ISR1. ISR2 is similar to the previous labs.

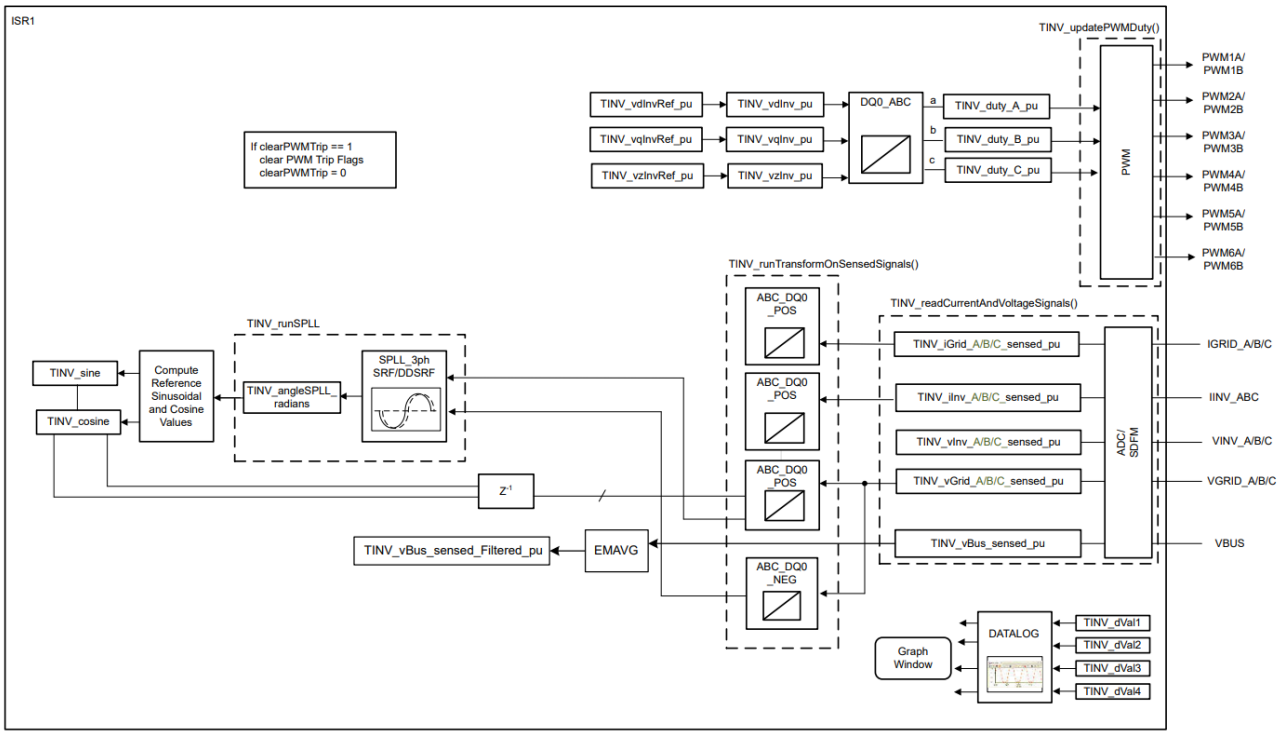


Figure 5-7. Lab 5 Flow Chart

To verify boost action in Lab 5, follow the steps according to the sequence provided:

- Turn on the auxiliary power supply, set the supply to 15 V, and then debug and run the code.
- Connect the load to the J1 and J2 terminals. Make sure to use a high load resistance (around 2 kΩ). A low resistance load can lead to high inrush currents and triggering of the overcurrent protection.
- Apply 30 V_{RMS} AC voltage to the three-phase terminals.
- Add the required variables to the watch window to change the values in runtime. The values added in the watch window can only be changed by pausing the R5 core.
- Set *TINV_startPowerStage* = 1 and *TINV_closeGiLoop* = 1.
- The overcurrent detection logic trips the PWMs due to inrush currents and the trips are set.
- Clear the PWM trip by setting *TINV_clearPwmTrip* to 1 to see a slight boost in DC voltage.
- Before PFC action begins, a rectified current is drawn due to the load on the Vbus. As soon as *clearPWMTrip* is set to 1, a slight boost in DC voltage occurs.

Start the experiment with a very low AC input. Once the *clearPWMTrip* successfully clears the inrush overcurrent trip, slowly increase the AC input step by step to the desired value. Avoid sudden increases in the AC input which can trip the PWMs.

5.3.1.1 Memory Browser - Continuous Refresh

For continuous monitoring of memory regions in runtime without pausing the code, use the Memory Browser. Implement the following steps for continuous monitoring:

1. Launch the .ccxml file of AM263x controlCARD as explained in the MCU API guide - CCS_Target_Launch
2. Find the AM263X target configuration in the *Debug* window. Right click on the .ccxml and choose *Show all cores*.
3. Connect to R50_0 core and load the .out firmware image for the project. Before resuming the execution, follow the remaining steps first.

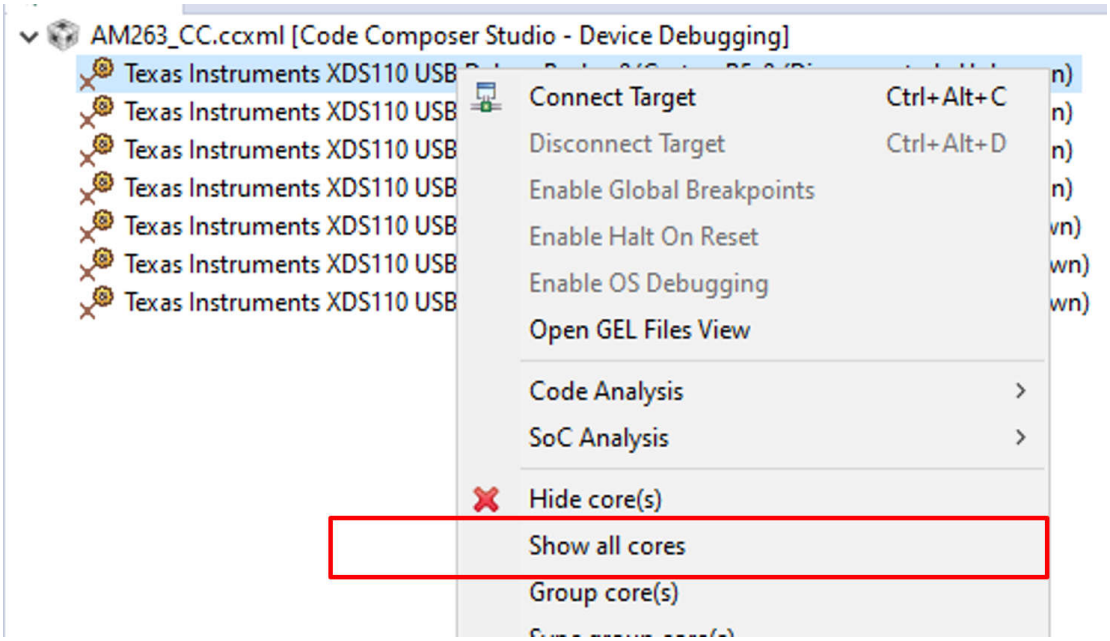


Figure 5-8. Show All Cores in AM263x

4. Go to the *View* menu item and open *Memory Browser*. Choose the DAP and pin the selected DAP to the Memory Browser window. Click on the *System_View* and choose *Continuous Refresh Option* (see Figure 5-10) for continuous monitoring and then run the code. This allows for the debug of memory regions in runtime without a need to pause the R5 core.

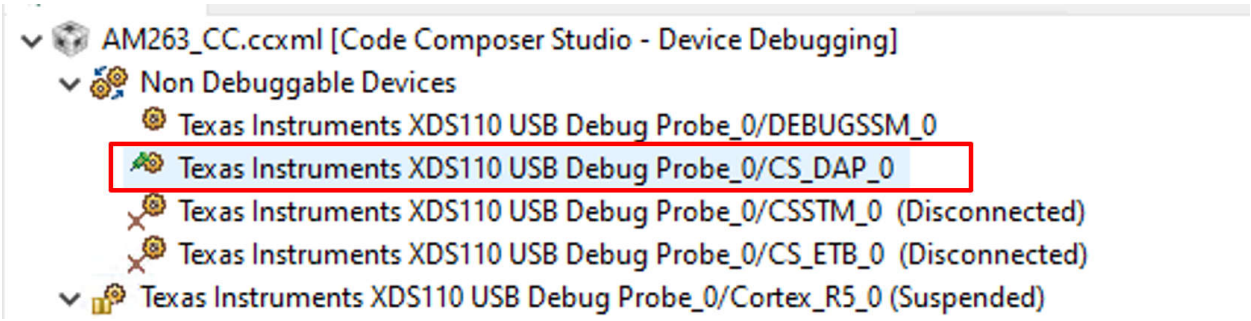


Figure 5-9. Pin CS_DAP_0 to the Memory Browser

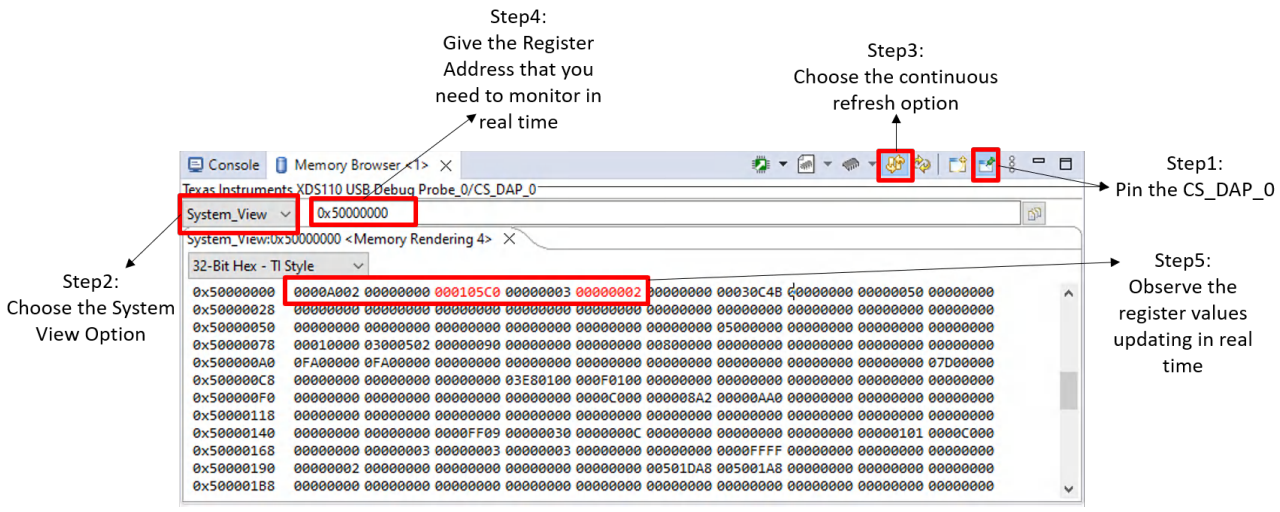


Figure 5-10. Memory Browser With CS_DAP_0 Pinned for Real-Time Debug

5. If a variable is present in TCM, then use the `attribute` keyword to place the variable in OCRAM for it to be visible in the Memory Browser. For example, to view the *TINV_startPowerStage* status variable in the Memory Browser, define the variable as below:

```
volatile int32_t TINV_startPowerStage __attribute__((__section__(".DebugData")));
```

   These variables can be monitored and modified in the Memory Browser as shown in Figure 5-11 and Figure 5-12.

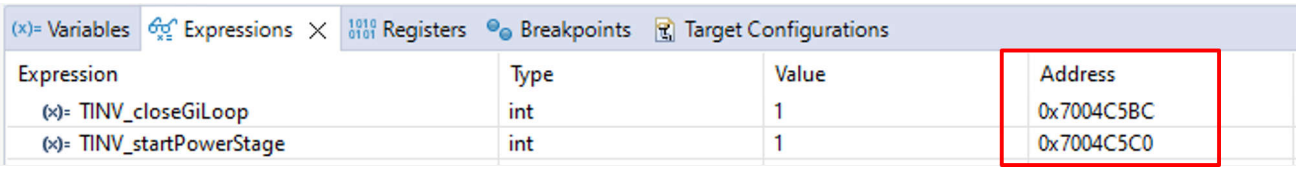


Figure 5-11. Gi Loop and Power Stage Status in Expressions Window

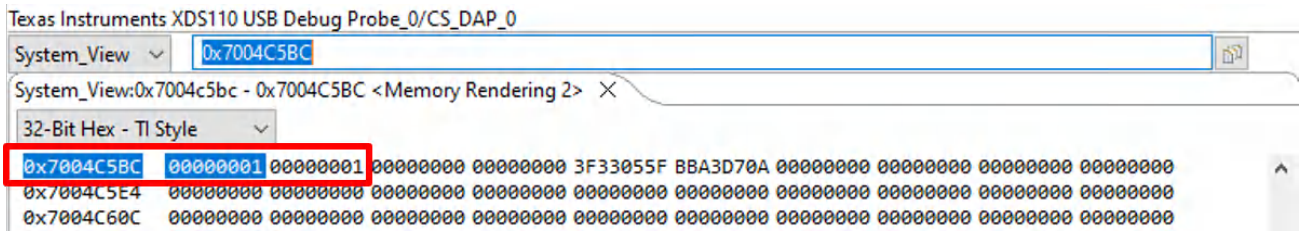


Figure 5-12. Gi Loop and Power Stage Status in Memory Browser

6. If the values are not being refreshed in real-time, pause the core and try disabling the data cache. To do this, go to *Tools → ARM Advanced Features* and uncheck the *Data Cache Enabled* option. Then resume the core execution and the real-time memory updates are viewable.

5.3.2 Lab 6

Set the project to Lab 6 by changing the *LAB_NUMBER* in the user settings file. Under this condition, the converter operates as a rectifier. In this lab, the ISR1 contains PFC operation with closed current loop and open voltage loop. Also, the *TINV_idRef_pu* variable is defined with a negative sign for PFC mode of operation and with a positive sign for inverter mode of operation. The locking of SPLL loop can be verified in this lab. Figure 5-13 explains the working flow for the lab.

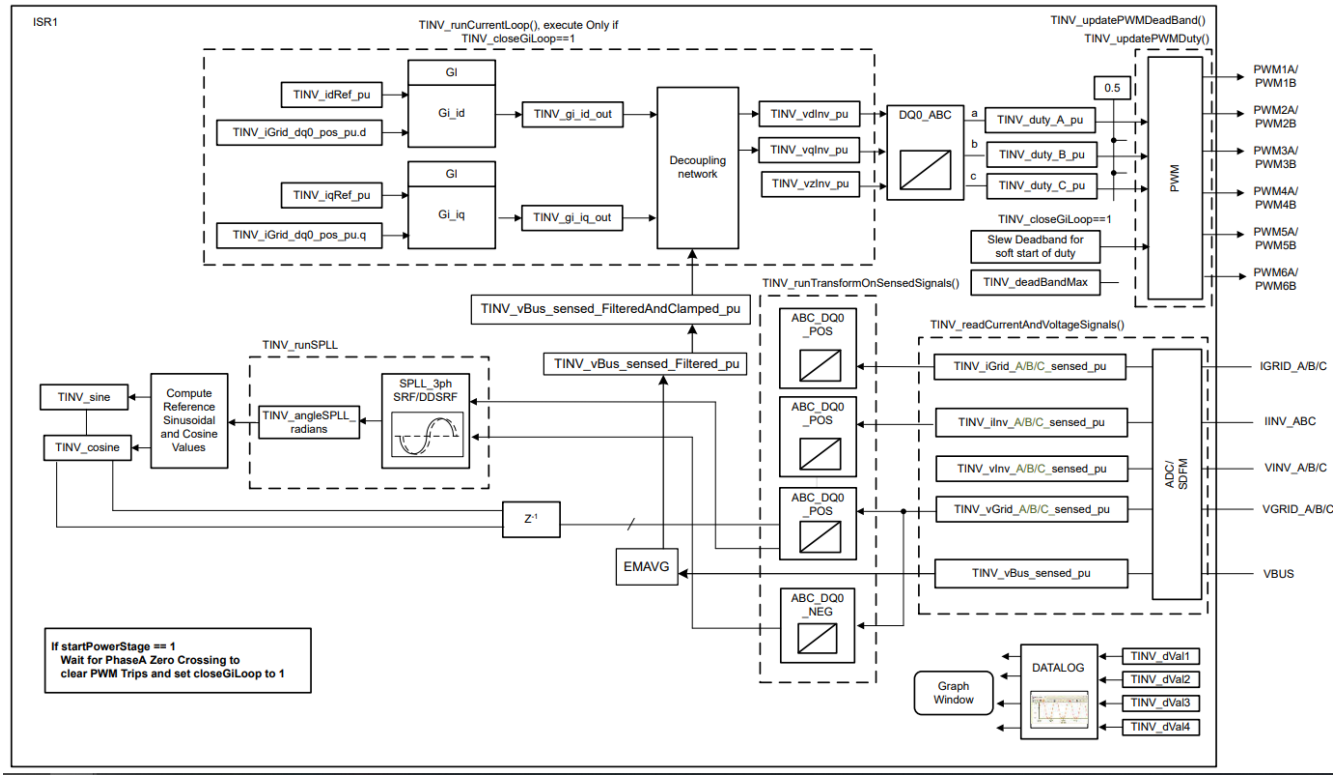


Figure 5-13. Lab 6 Flow Chart

In this lab the best K_p and K_i coefficients are defined as below. The user can modify these coefficients to meet the necessary loop bandwidth and phase margin.

```
#define TINV_GI_PI_KP ((float32_t)0.3))
#define TINV_GI_PI_KI ((float32_t)0.0120860479))
```

To verify Lab 6, carry out the following steps:

- Turn on the auxiliary power supply, set the to 15 V and load the code. When the code is executed, the relays are turned ON by default with the initialization.
- Set the load resistance to a high value such as 3.18 kΩ. Set the AC input voltage to 30 V_{RMS} with appropriate current limit.
- The following is the sequence of setting variables in the Memory Browser or Watch window. Set up the Memory Browser window to update the values without the need to pause the code as explained in Section 5.3.1.1.
  - Set *TINV_idRef_pu* to –0.013 pu.
  - To start the PFC mode, set *TINV_startPowerStage* = 1. The current is now drawn from the grid as a sinusoidal signal (with some harmonics because the current is at low power) and boost action seen on the vBus. If overcurrent trips are set, clear the PWM trip by setting *TINV_clearPwmTrip* to 1.
- The current becomes sinusoidal as the load is increased. This verifies the start-up of PFC at 30 V_{RMS}. Once the clearPWMTrip successfully clears the inrush overcurrent trip, slowly increase the AC input step by step to the desired value. Avoid sudden increase in the AC input which can trip the PWMs.
- Running this lab at higher input voltage without a supervisory voltage loop results in a big overvoltage across the DC terminals. Always start this lab at low voltage and low power as previously described for safety reasons and then slowly ramp to the desired voltage for closed current loop tuning.

The dead band is adjusted to a width of (30 × 5) ns for soft start and can be decreased to the desired level step by step. The PWM pulse width reduces as the width of dead band increases. Figure 5-14 illustrates the PWM configuration for this setup where the dead band is set to a large value and slowly reduced to the nominal value to limit the current spikes. Without soft start, a huge current spike appears which causes overcurrent trip and the DC bus voltage also collapses.

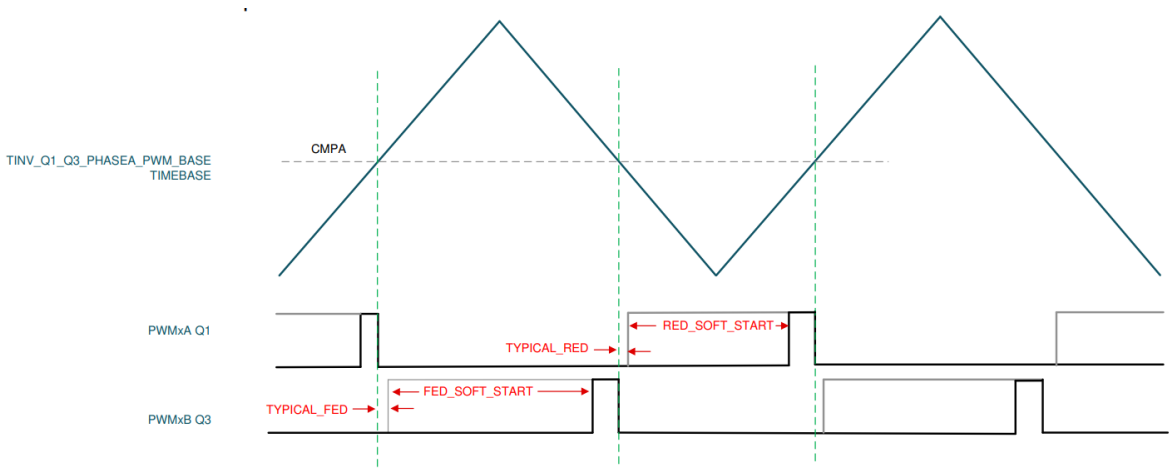


Figure 5-14. PWM Dead-Band Soft Start

5.3.3 Lab 7

Lab 7 is similar to Lab 6 except that Lab 7 has the DC bus voltage loop closed. PFC operation in this lab can be tested with closed current loop and closed voltage loop. Figure 5-15 explains the code flow for ISR1 in the lab.

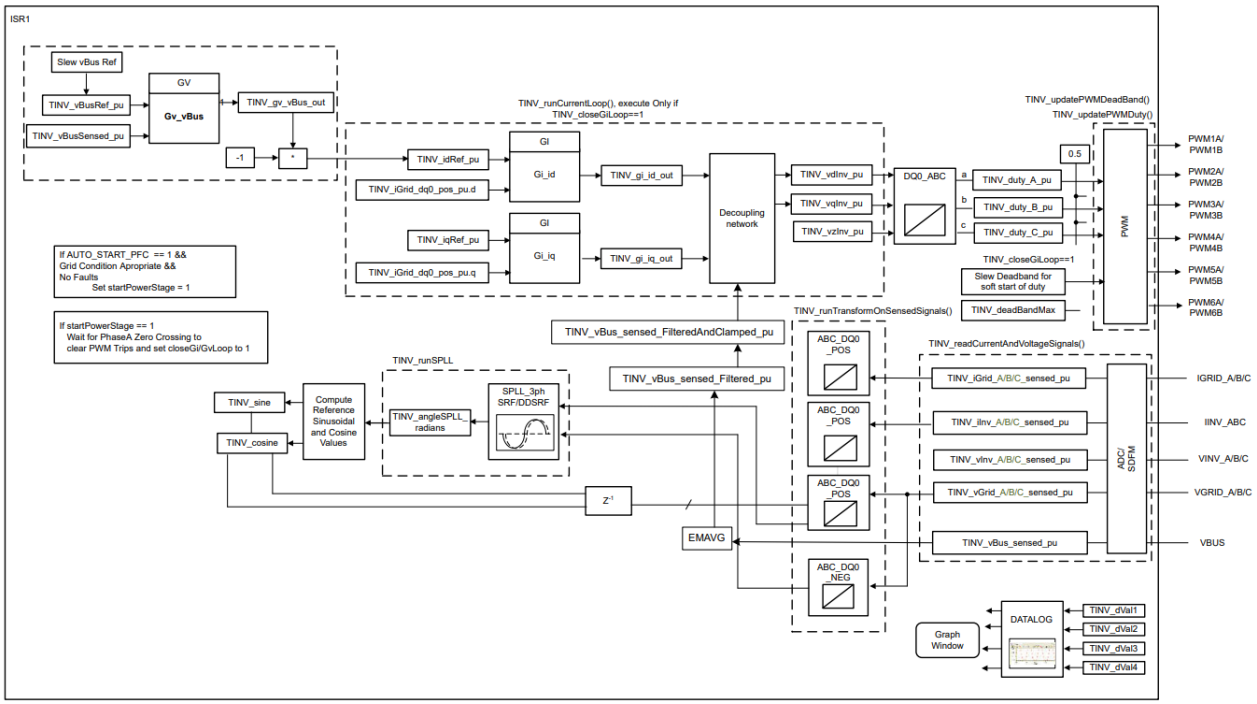


Figure 5-15. Lab 7 Flow Chart

Go through the following steps to verify Lab 7:

- Set the project to Lab 7 by changing the *LAB_NUMBER* in the user settings file. Under this condition, the converter operates as a rectifier. In this lab, the ISR1 contains PFC operation with closed current loop and closed voltage loop.
- The loop coefficients K_p and K_i for this lab are similar to the previous labs.
- This lab can be tested connected to grid because the voltage loop is closed and the *TINV_idRef_pu* value is adjusted for the input voltage.
- Build and load the code into the AM263x controlCARD.
- Set the AC input to a lower value of 30 V_{RMS} at the beginning of the test and increase the input step by step to 230 V_{RMS} once the EPWM trips are cleared.

- As the code is loaded, the relays are open by default. Turn on the AC supply. The PWMs go into a trip state due to the inrush currents.
- The following is the sequence of setting variables in the *Memory Browser* or *Watch* window. Set up the *Memory Browser* window to update the values without the need to pause the code as explained in Section 5.3.1.1.
  - To start the PFC mode, enter "1" for the *TINV_startPowerStage* variable. The current is now drawn from the grid as a sinusoidal signal (with some harmonics as it is at low power) and boost action seen on the vBus. If overcurrent trips are set, clear the PWM trip by setting *TINV_clearPwmTrip* to 1.
  - The current becomes sinusoidal as the load is increased. This verifies start-up of PFC at 30 V_{RMS}. Once the *clearPWMTrip* successfully clears the inrush overcurrent trip, slowly increase the AC input step by step to the desired value.
  - Increase the *TINV_vBusRef_pu* to 0.684 which corresponds to the bus voltage of 800 V.
  - After confirming that the system is stable, try and set the *TINV_vBusRef_pu* to 0.684 pu at the start-up.

5.3.4 Test Results for PFC Operation in Lab 7

The results for PFC operation obtained for Lab 7 are tabulated in this section for various bus reference values and input AC voltages. The AC sine wave input is given to the three phases of the PFC TIDA-1606 setup. The ADCs are expected to sense these voltages and produce PWMs which, when filtered, give a sinusoidal output that follows the input. This proves that the SPLL locking of the angle is working in sync with the input AC voltage. Figure 5-16 shows the output of the current waveforms on the grid side in three-phase when an input of 200-V AC is given to the PFC hardware.

Figure 5-16. I_a, I_b, I_c Waveforms on the Grid-side of TIDA Hardware in PFC Mode for 200-V AC Input

This three-phase input voltage is applied to TIDA-01606 input terminals to produce a regulated DC bus voltage while maintaining sinusoidal input currents. This proves the PFC operation in the TIDA-1606 setup with AM263x. At a given bus reference value, when input AC voltage is varied, the output DC voltage measured by the power analyzer tool remained constant and stable proving the working of voltage loop. This verifies the closed voltage loop operation. This is captured in Figure 5-17 for an input AC source of 200 V. This DC voltage is captured from the bus side of the PFC hardware using a bi-directional DC power supply & analyzer tool.

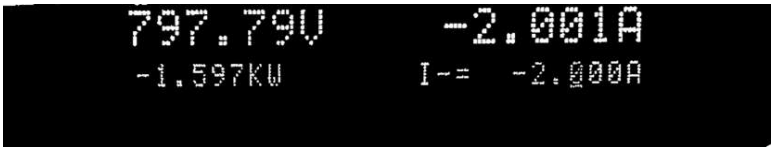


Figure 5-17. Output Bus Voltage of PFC Hardware for 200-V AC Input

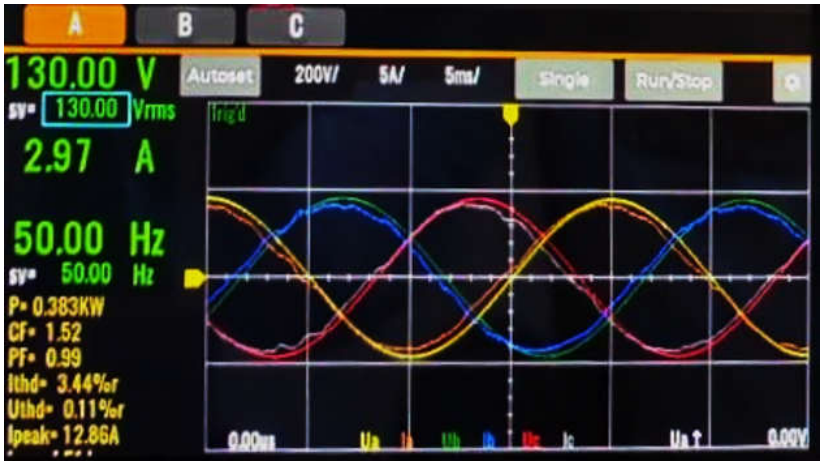


Figure 5-18. I_a, I_b, I_c Waveforms on the Grid-side of TIDA Hardware in PFC Mode for 130-V AC Input

Figure 5-19. Output Bus Voltage of PFC Hardware for 130-V AC Input

The output DC voltages against the varying bus reference value and sinusoidal AC input for the PFC Operation are recorded in the Table 5-3. The following recordings are taken for a Lab 7 test and the AC input is captured for phase A of the three-phase input. The bus voltage reference value is 0.7 with an input AC voltage of 200 V, and after the PFC stage that corresponds to an output DC voltage of 797.83 V. Under this condition the measured current THD (Total Harmonic Distortion) is 2.47%. The setup has yet to be tested for the nominal AC voltage of 230 V_{RMS}.

Table 5-3. Performance of Lab 7 in PFC Operation With AM263x controlCARD™

vBus Ref	Input Voltage AC A	Input Current A	Freq A	Input Power A	CF A	PF A	ITHD A	VTHD A	I_{peak} A	Measured Bus Voltage	Power at Bus Side	Constant Current Setting Measured Bus Side Current
	(V_{RMS})	(A)	(Hz)	(KW)			(%)	(%)		(V)	(KW)	(A)
0.7	200	2.79	50	0.547	1.51	0.98	2.47	0.13	9.73	797.83	1.596	2
0.6	170	2.75	50	0.46	1.54	0.98	2.64	0.13	12.86	672.83	1.345	2
0.5	130	2.97	50	0.383	1.52	0.99	3.44	0.11	12.86	560.24	1.12	2
0.4	80	3.89	50	0.311	1.49	1	3.73	0.18	12.86	447.84	0.896	2
0.3	70	3.32	50	0.232	1.48	1	5.38	0.22	12.86	335.22	0.67	2
0.3	50	3.96	50	0.198	1.45	1	5.32	0.31	12.86	280.83	0.562	2

The PF and ITHD values are observed to become better at higher AC input values for a given bus voltage reference value. In Figure 5-20 and Figure 5-21, the input voltage and other characteristics of all the three phases is captured from the power analyzer tool connected to the hardware.

Figure 5-20. ITHD and PF Values at 200-V AC Input With a Constant Bus Voltage Reference Value

Figure 5-21. ITHD and PF Values at 130-V AC Input With a Constant Bus Voltage Reference Value

5.3.5 PFC Interrupt Benchmarks

This section contains the interrupt execution time for ISR1 in PFC labs and walks through the procedure followed to run the profiling tests.

Add the profiling code for the required ISR before building and loading the code. Capture the cycle counter value at the beginning of the ISR using the API `CycleCounterP_getCount32()`. Capture the cycle counter value at the end of the ISR using the same API. Now the difference in the end and beginning counter values gives the exact number of CPU clock cycles consumed to execute the ISR1 code. This profiling code is commented in the application as it is used only for benchmarking the ISR1 execution time. To use the code, define the *PROFILING_ISR1_MAIN* macro in the project.

Build the code changes and load the code in AM263x controlCARD through CCS. Add the variables *TINV_startPowerStage*, *TINV_closeGiLoop*, *Max_CPU_Cycles* and *Min_CPU_Cycles* in the Watch window. Run the application in CCS, and set the values of *TINV_startPowerStage* = 1 and *TINV_closeGiLoop* = 1 to start the current and voltage loop conversions. Pause the code and look for the values of *Max_CPU_Cycles* and *Min_CPU_Cycles* to see the recorded values. To reset the CPU cycle values, set the *Max_CPU_Cycles* to 0 and *Min_CPU_Cycles* to any maximum value.

The ISR interrupt latency or the time taken by the ISR from hardware interrupt trigger to the start of the ISR1 in the software application is not captured in this benchmark.

The CPU cycle values in Table 5-4 are high in the 'Max' column due to the code not being cached and setting a few flags for first time execution. The execution of the Control ISR1 after the first time execution is captured in the 'Min' column of the table. The values in the Min column capture the best case ISR1 execution time.

Table 5-4. Interrupt Benchmarks for PFC Operation

	MIN		MAX	
Lab Number	**CPU Cycles**	**Execution Time (µs)**	**CPU Cycles**	**Execution Time (µs)**
Lab 5	599	1.4975	627	1.5675
Lab 6	842	2.105	895	2.2375
Lab 7	856	2.14	924	2.31

6 References

- Texas Instruments, *TIDA-01606: 10-kW, Bidirectional Three-Phase Three-Level (T-type) Inverter and PFC Reference Design Guide*
- Texas Instruments, *Software Phase Locked Loop Design Using C2000™ Microcontrollers for Three Phase Grid Connected Applications Application Note*

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), 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 will 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 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.

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

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265