LAB 3: Pulse Width Modulation Control with the TI F28335

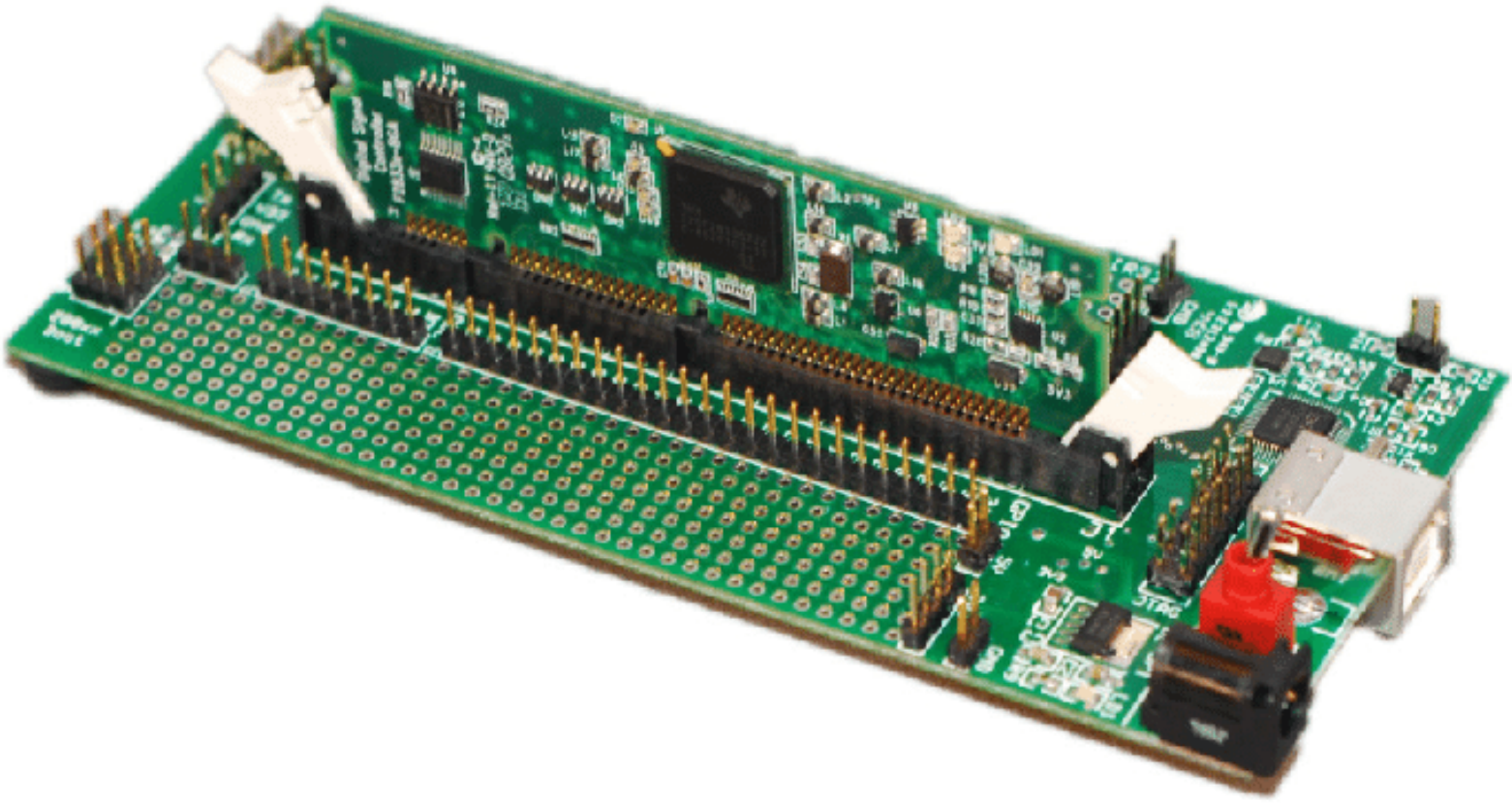
Using MATLAB and Simulink for Pulse Width Modulation

Work carried out by :
CHERIF Enzo
Students at Seatech, Class of 2025
2A, track SYSMER

As part of the course :

Programming on Electronic Card

August 2024



Contents

1 Introduction

1.1 Presentation of the Practical Work and the TI F28335 Card

In this lab module, we will focus on the TI F28335 board from Texas Instruments, designed for digital control and signal processing applications. This board incorporates a TMS320F28335 processor from the C2000 family, capable of operating up to 150 MHz, ideal for applications such as motor control and energy conversion.

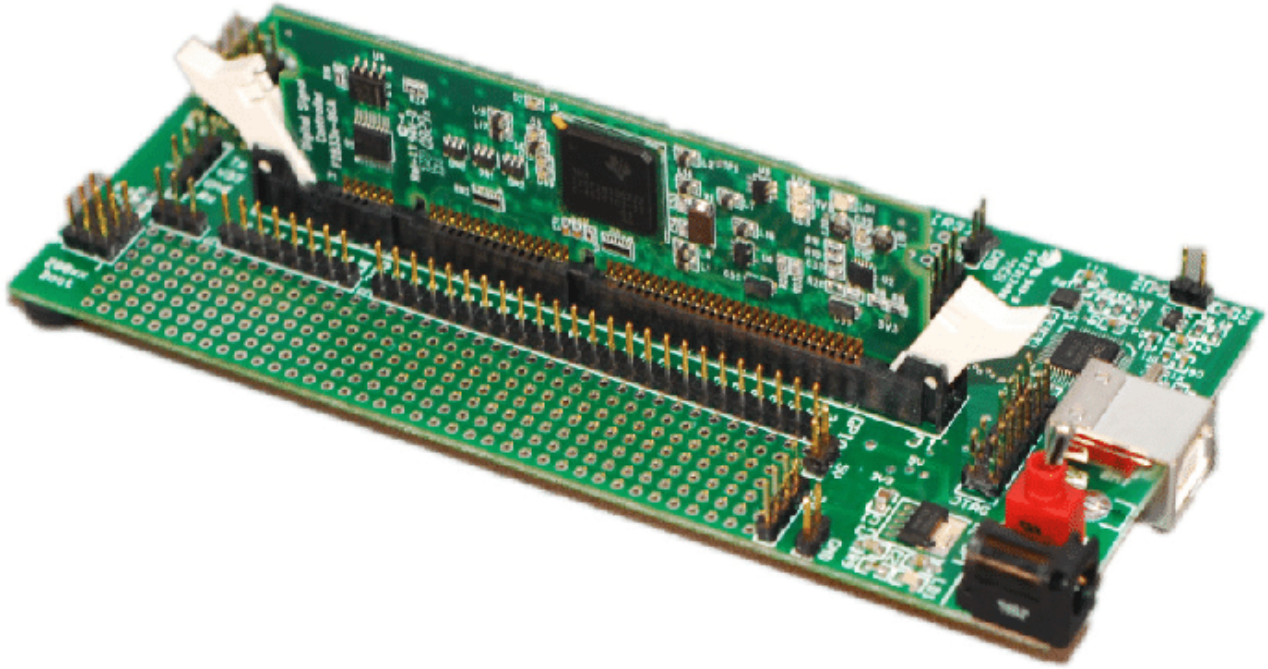


Figure 1: TI F28335 Board

The TI F28335 board offers various connectivity options, including CAN ports and analog and digital inputs/outputs. It is also compatible with expansion modules, thereby increasing its flexibility for various projects. During this lab, you will learn to use and program this board, exploiting its capabilities through practical exercises that cover the fundamentals of digital control and automation.

1.2 Objectives of the Practical Work

The main objective of this lab with the TI F28335 board is to introduce you to digital control and automation. By the end of this module, you should be able to:

1. **Install and Configure the Necessary Software**:
   - Master the installation of MATLAB, Simulink, and the necessary add-ons for communicating with the board.
2. **Program and Experiment with the Board**:
   - Program the board for basic applications such as sensor reading and actuator control.
   - Apply digital control concepts to design simple systems.
3. **Develop Problem-Solving Skills**:
   - Solve technical problems encountered during the lab.
   - Use available resources to find solutions to challenges.
4. **Collaborate and Communicate**:
   - Work in groups to complete exercises.
   - Present results and share learnings with peers.

These objectives aim to give you a practical understanding of the TI F28335 board, while developing both your technical and collaborative skills.

2 Preparation of the Development Environment

Setting Up the Environment for Communication with the TI F28335 Board

After completing all necessary installations, it is crucial to prepare and configure your development environment to work effectively with the TI F28335 board. Here are the steps to get started with a first practical example, the ADC PWM, which is essential for understanding the operation of the board.

1. **Restarting MATLAB**:
   - Close MATLAB once all installations are completed. Reopen it to ensure that all configurations and installations are properly supported. This restart allows MATLAB to update its settings and recognize the newly installed packages.
2. **Verifying the Installation**:
   - If a problem occurs during startup, MATLAB will generally indicate the steps to follow. Installation issues can often be resolved by opening 'Manage Add-Ons' and clicking on the configuration icon, which looks like a small gear, to adjust or reinstall the necessary packages.
3. **Opening the ADC PWM Example**:
   - In the MATLAB search bar, type "adc pwm" and press Enter. Select the first result and click 'Open' to open the model in Simulink.

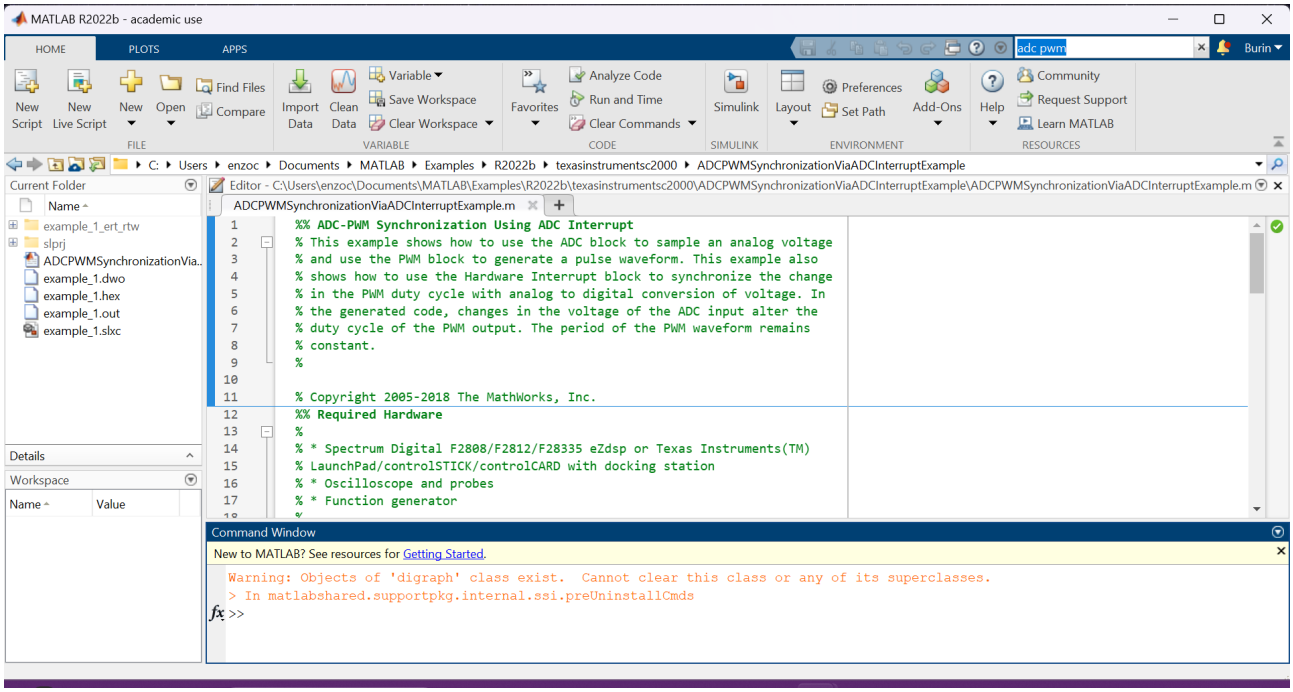


Figure 2: Search for the ADC PWM model in MATLAB.

   - This specific model is designed to introduce you to handling analog input and pulse width modulation (PWM).
4. **Configuring the Simulink Model**:
   - Once the ADC PWM model is open in Simulink, you will find that it is already pre-configured. We will adjust the settings in more detail later.

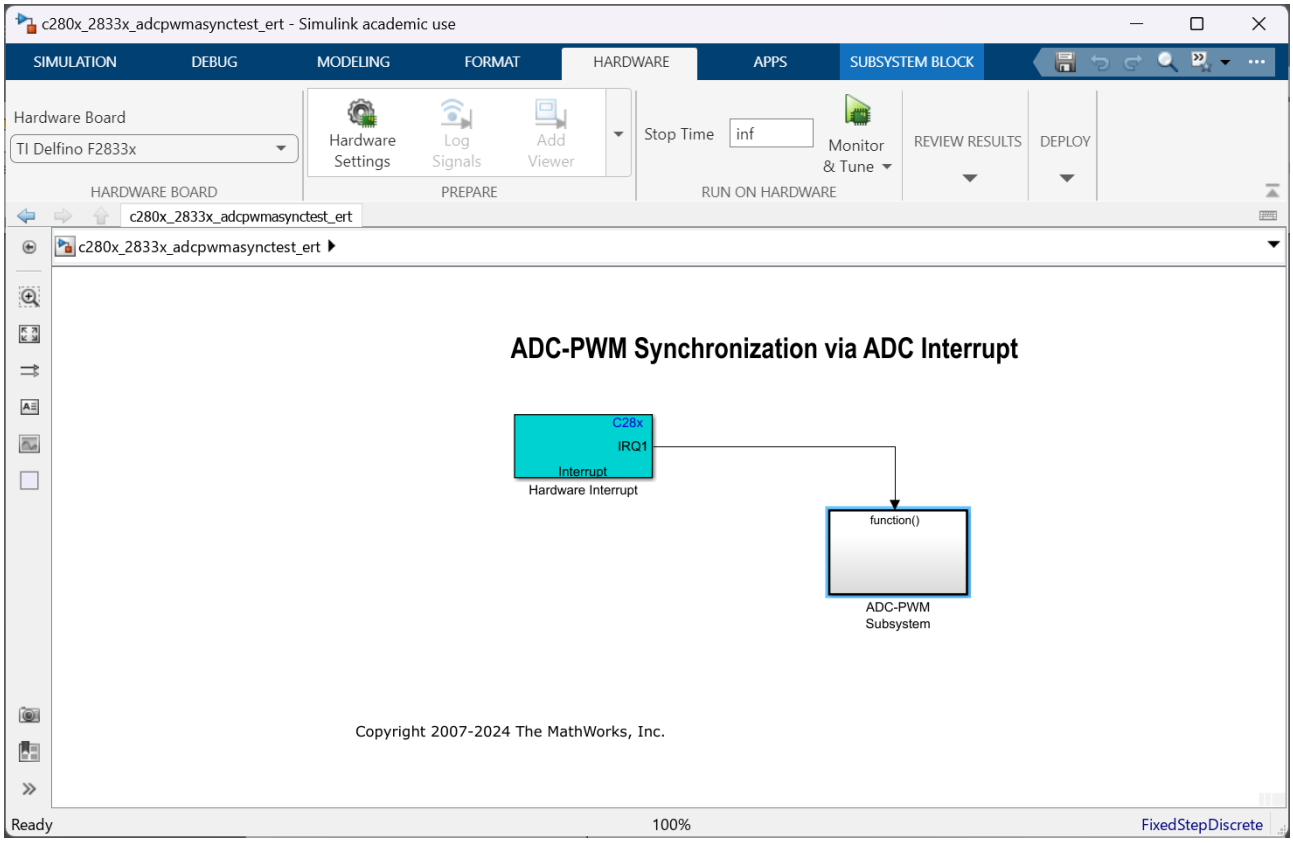


Figure 3: Simulink model with the 'ADC-PWM' block preconfigured.

- Double-click on the 'ADC-PWM' block to open the configuration and verify that you have the following diagram.

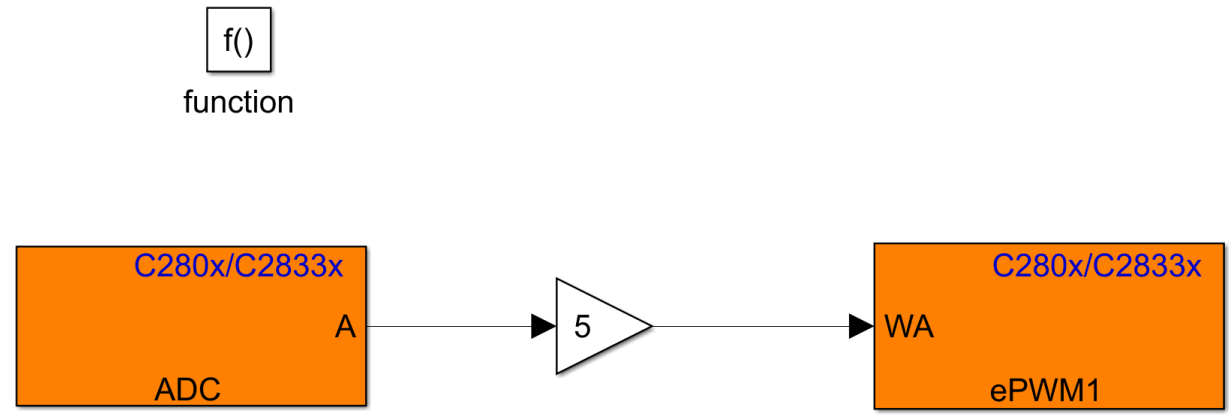


Figure 4: Diagram of the ADC PWM model in Simulink.

5. **Testing the Electronic Board**:
   - Before proceeding to test, save your work under an appropriate name, such as 'example_1'.
   - Connect the TI F28335 electronic board to your computer via a USB cable. Ensure that the connection is secure and that the board is properly powered.
   - If possible, connect an oscilloscope to observe the PWM outputs and the analog signals processed by the board.
6. **Beginning the Tests**:

- Start the simulation in Simulink to see how the board reacts. Check the outputs on the oscilloscope and ensure that the signals match what is expected from the Simulink model (we will see this step in more detail later in the lab).

7. **Troubleshooting**:

- If the results are not as expected, review the settings, check the physical connections, and ensure that all necessary drivers and software are properly installed and configured.

By following these steps, you will be well-prepared to start experimenting with the TI F28335 board and to develop your skills in programming embedded systems. This methodical approach will help you understand the operation of the board and maximize the efficiency of your development projects.

3 Quick Start Guide

Getting Started with the TI F28335 Board

To get started with the TI F28335 board, follow these step-by-step instructions to connect the board to your PC, program a first simple example, and test the outputs using an oscilloscope.

1. **Connecting the Board to the PC**:

- Connect the TI F28335 board to your computer using a USB cable. Ensure that the connection is well established and that the board is detected by your system.

2. **Saving and Compiling the Project**:

- Before compiling the project, save your work in MATLAB/Simulink. To start the compilation of the Simulink model, press `Ctrl` + `B` simultaneously. This command initiates the build of the model for the board.
- The first compilation may take a bit longer, especially if it is the first time or depending on your computer's specifications. Wait until the process is completed without errors.

3. **Verifying the Transfer to the Board**:

- After compilation, it is crucial to ensure that the program has been successfully transferred to the TI F28335 board. Indicators on the board can help confirm that the download was successful.

4. **Settings in Simulink**:

- Go back to Simulink and open the settings of the ePWM (Electronic Pulse Width Modulation) block in your model. Check that the "Timer period" is set to 20480. This value determines the period of the PWM signal generated by the board.

5. **Theory and Practice**:

- Before moving on to observing the results on the oscilloscope, it is useful to theoretically understand how the PWM signal is generated and controlled by the board. This theoretical understanding will help correctly interpret the signals observed.

By following these steps, you will have successfully taken the first steps with the TI F28335 board, including programming, compiling, and initially testing basic functionalities. This guide will help you start quickly and effectively in developing your embedded projects.

4 Development of Exercises

Getting Started with PWM Implementation and Testing on the TI F28335 Board

This exercise focuses on implementing and testing Pulse Width Modulation (PWM) using the TI F28335 board. You will explore setting switching frequencies and calculating the period and duty cycle.

Detailed Instructions for Each Exercise

1. **PWM Setup**:

- **Switching Frequency (f_s)**: Set the switching frequency to 10 kHz or 20 kHz. The PWM period is determined by the following formula:

$$\text{Time Period} = \frac{\text{Main Clock}}{f_s} \tag{1}$$

Since the main clock frequency is specified in the datasheet as 150 MHz, with $f_s = 10$ kHz, you have:

$$\text{Time Period} = \frac{150 \text{ MHz}}{10 \text{ kHz}} = 15000 \text{ clock cycles} \tag{2}$$

- **Duty Cycle (D)**: The duty cycle is the ratio of the active time to the total period. For a duty cycle of 20%, T_{on} is calculated as follows:

$$T_{on} = D \times T = 0.2 \times 15000 = 3000 \text{ clock cycles} \tag{3}$$

- Consulting the electronic board's datasheet, you find the recommended switching frequencies (10 kHz or 25 kHz) and the value of the main clock (150 MHz). This allows us to determine T_{Board}:

$$T_{Board} = \frac{\text{Main Clock}}{f_s} = \frac{150 \text{ MHz}}{10 \text{ kHz}} = 14999 \text{ clock cycles} \tag{4}$$

Use this T_{Board} value to precisely configure the PWM in your Simulink model.

2. **PWM Diagram**:

- Here is a concrete representation of a Pulse Width Modulation (PWM) signal. This type of signal varies its active duration (T_{on}) relative to the total period (T), defining its duty cycle. The duty cycle is the percentage of the total period during which the signal is in the high state (logical "1"). In this example, the duty cycle is 50%, meaning T_{on} and T_{off} are equal and each occupy half of the total period. This is crucial for controlling fewer switch in electronic devices, as it determines the average amount of power delivered to the load.

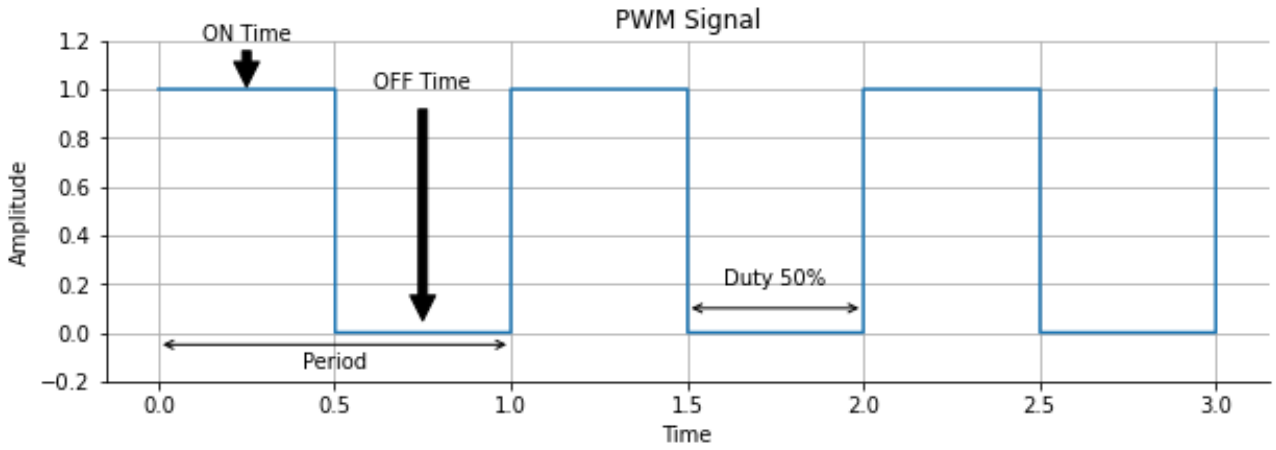


Figure 5: Illustration of a PWM signal with a 50% duty cycle, clearly showing T_{on} and T_{off} for a period of $T = 1$ clock cycles.

3. **Configuration in Simulink**:

- **If You Know the Clock Cycle :**
  - Open the Simulink model and go to the ADC-PWM block. The blocks are already pre-configured and will serve as a starting point for adjusting the settings.
  - Double-click on the ePWM block to open the settings window and ensure that you have the same settings as shown in the following screenshots. We will adjust the specific settings later, according to the requirements of your project and the capabilities of the TI F28335 board.

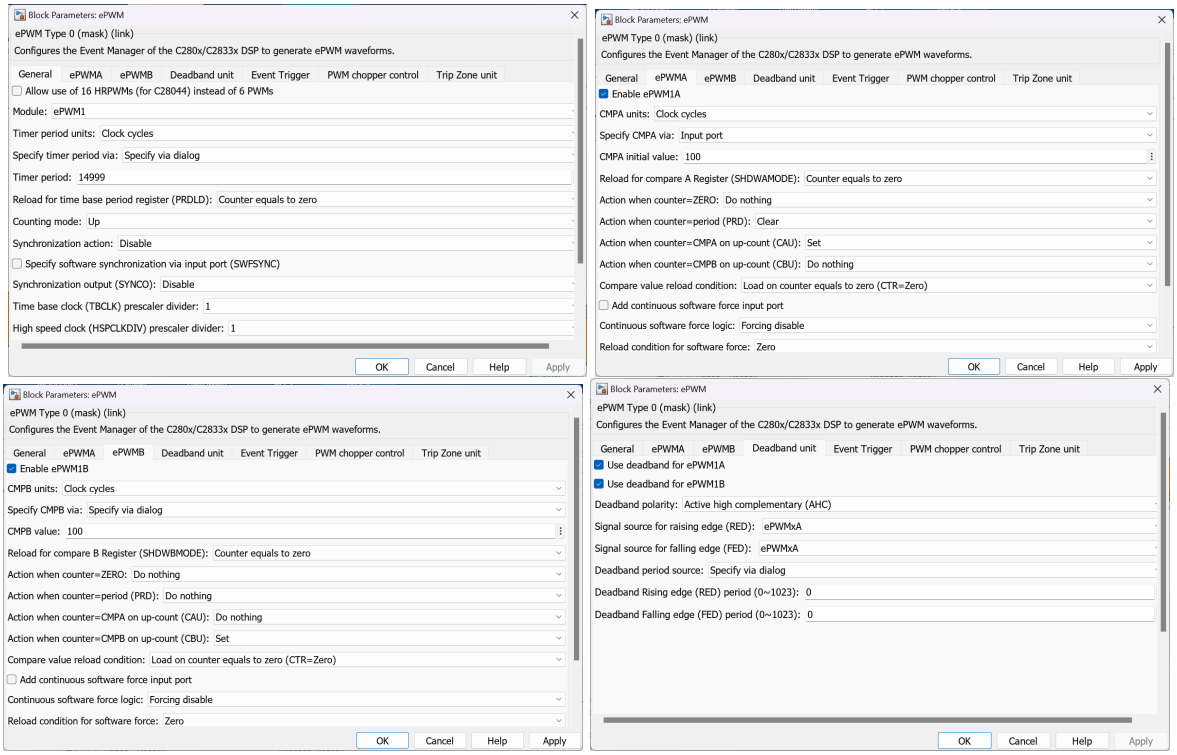


Figure 6: Settings of the ePWM block in Simulink for generating PWM signals.

- **If You Only Have the Period in Seconds :**
  - Open the Simulink model and navigate to the ADC-PWM block configuration. Start with the general settings to match the period that you have.
  - Adjust the frequency settings in the ePWM block to correspond to the provided period by calculating the necessary parameters from the period. Detailed calculations and settings adjustments can be seen in the provided figures.

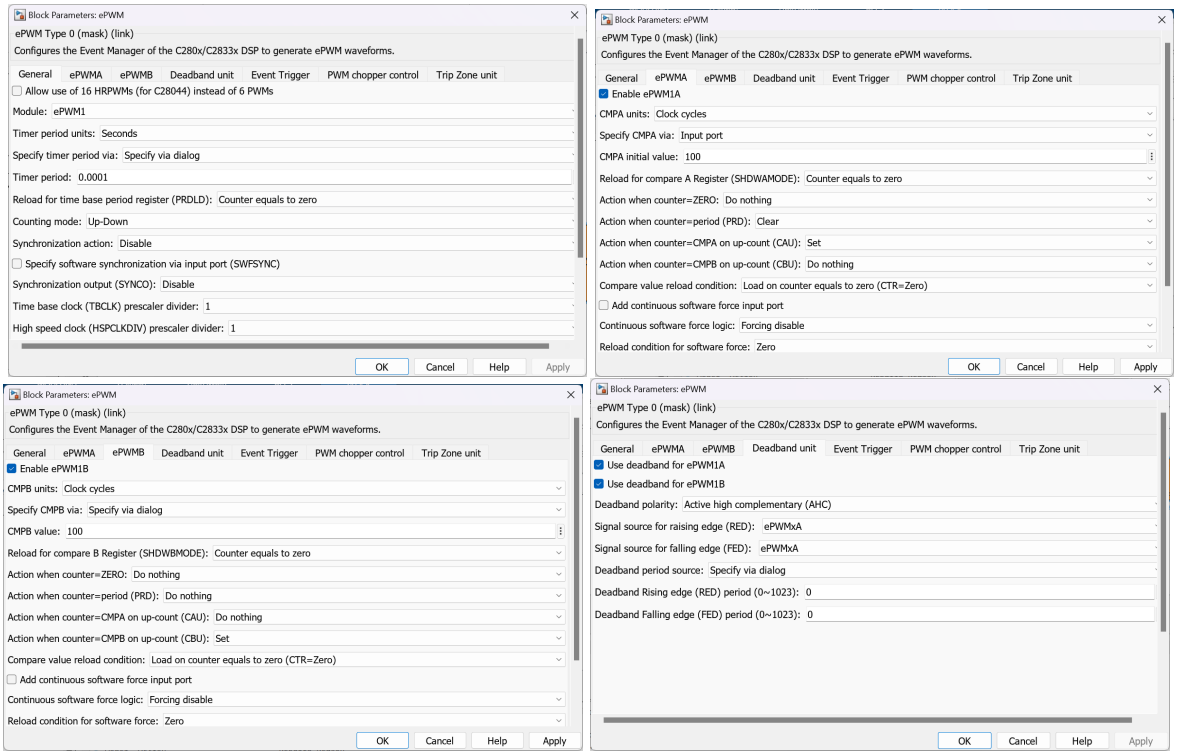


Figure 7: Adjusting the ePWM settings based on the period in seconds.

4. **Preparation for Code Testing**:

- To proceed with the code testing, it is imperative to have the proper diagram in place. You can find the necessary "Constant" block to build your model by opening the "Library Browser" in Simulink.
- Navigate to ‘Simulink > Commonly Used Blocks‘ and locate the "Constant" block. Then drag this block into your model to integrate it into your diagram.
- Here is the screenshot of the required diagram for the test. Ensure your setup matches this example.

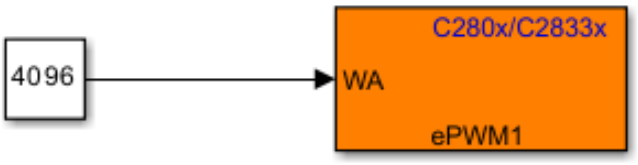


Figure 8: Configuration diagram of the "Constant" block connected to the input of the ePWM1 block.

5. **Code Testing**:
   - After setting up the model, compile it by pressing ‘Ctrl+B‘. Verify that the code functions correctly by observing the outputs on the board.
   - To observe the PWM signal, connect an oscilloscope to the PWM pins indicated in the technical documentation of the board. Ensure that the cable is correctly plugged into the GPI0-00 (PWM1A) or the GPI0-01 (PWM1B) pin and the ground (GND). The best option is to use pin B; it will be better for observation. The following figure shows the example of the proper connection for the oscilloscope.

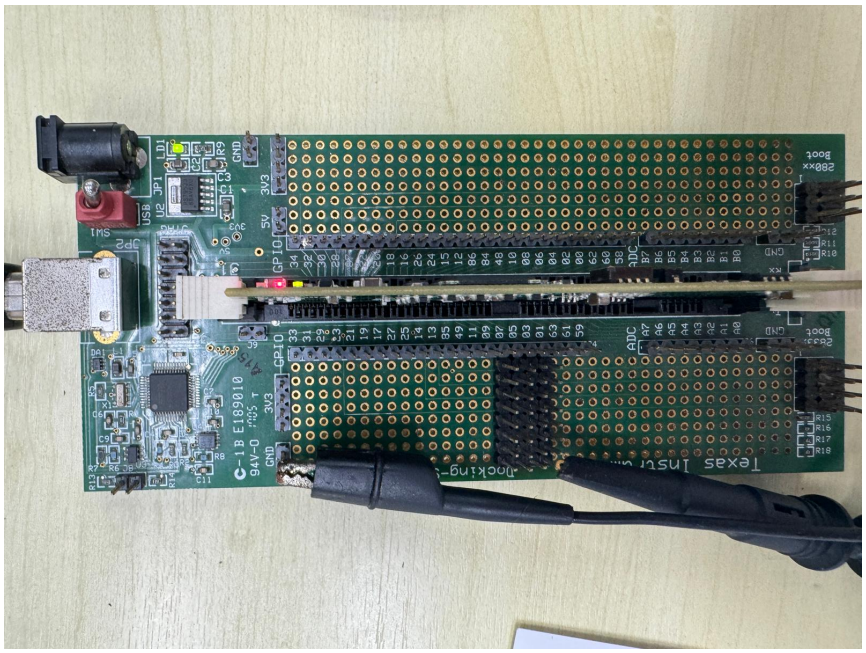


Figure 9: Connection of pins for testing the PWM signal with an oscilloscope.

6. **Oscilloscope Adjustment**:
   - Connect an oscilloscope cable to the desired channel (Channel 1) and check the attenuation scale indicated on the cable. Adjust the oscilloscope settings accordingly to match the attenuation, often marked by a factor of 10.
   - To access the oscilloscope’s attenuation settings, use the menu accessible where indicated by the finger in the following image:

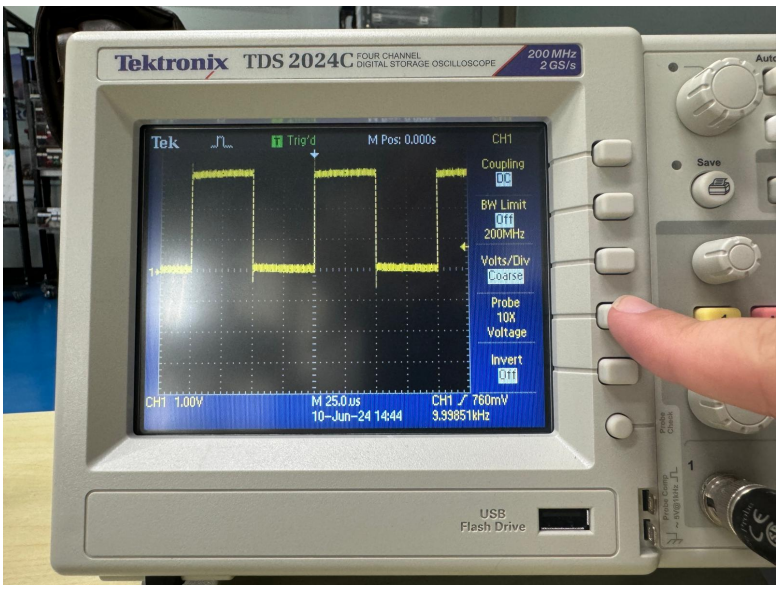


Figure 10: The menu to change attenuation is where the finger is pointed.

It is important to correctly adjust the attenuation so that the measurements accurately reflect the tested signal. Here is an example of the oscilloscope screen showing the signal with the correct settings:

Figure 11: The oscilloscope screen displaying the signal with the attenuation correctly set.

- To visualize the PWM signal, set the horizontal scale to 25 microseconds to capture the desired period. This setting is crucial for a clear and precise observation of the signal.
- Also adjust the vertical scale so that the voltage corresponds to 1V on Channel 1. This helps to correctly calibrate the amplitude of the observed signal.

Figure 12: Oscilloscope menu.

- It can be challenging to observe a stable signal without using the 'level' button in the trigger settings. Move the adjustment arrow until it is within the y-axis range of the signal. This will stop the signal at a fixed point on the screen, allowing you to clearly observe the high and low pulses of the PWM signal.

7. **Explanation of Deadband Polarity Options :**
   The Deadband Polarity function of the TI 28335 board is crucial for understanding how to effectively control Pulse Width Modulation (PWM) outputs. This feature allows for defining the behavior of PWM signals based on the Deadband settings. Here is a detailed explanation of the four available Deadband Polarity options, clearly illustrating their impact on PWM signals:

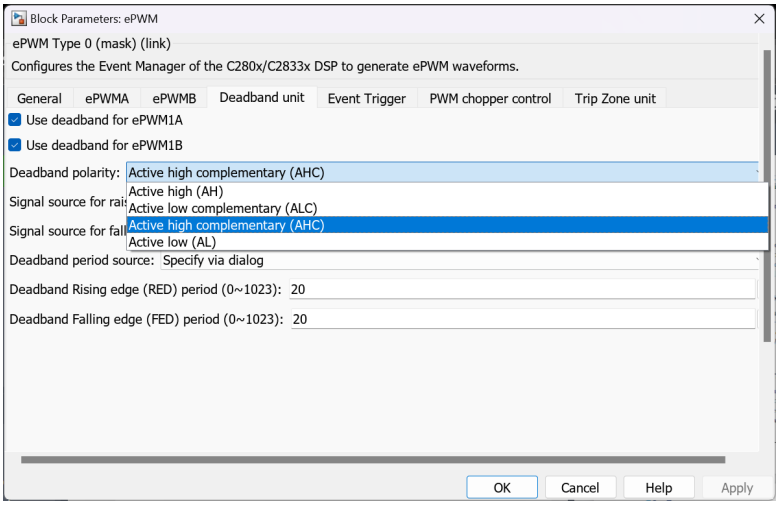


Figure 13: ePWM Deadband Polarity settings in the block parameters window.

Active High (AH)

- **Description:** By selecting the "Active High" option, the PWM1A signal is active high regardless of the pins to which the oscilloscope probes are connected, whether pins 00 or 01.
- **Observation:** When you connect to the output pins, the PWM1B signal will be visible on the opposite pin as a complement, reflecting a non-inverted behavior of PWM1A.

Active Low (AL)

- **Description:** The "Active Low" option is the exact opposite of "Active High". Here, only the PWM1B signal is actively low.
- **Observation:** In Active Low mode, you will primarily observe the PWM1B signal on the outputs, and the PWM1A signal will be less apparent or inactive.

Active Low Complementary (ALC)

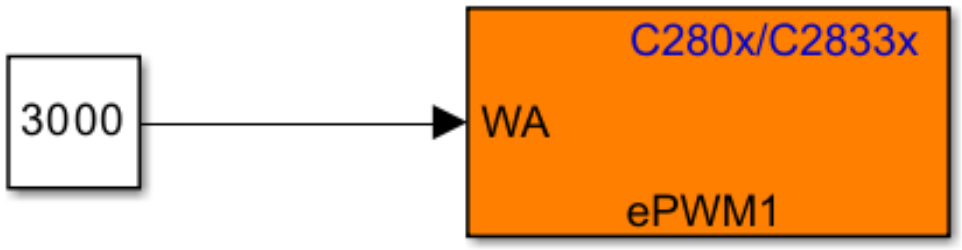
- **Description:** This configuration allows for observing both PWM signals, PWM1A and PWM1B, in a complementary manner.
- **Observation:** If you set a duty cycle, for example, 20% (0.2), this percentage will be visible on pin 01 corresponding to the PWM1B signal. The signal on pin 00 will be the opposite, i.e., primarily at a high level.

Active High Complementary (AHC)

- **Description:** "Active High Complementary" is the inverse of "Active Low Complementary". In this mode, the signals are also complementary but with reversed roles.
- **Observation:** The specified duty cycle will predominantly be observed on pin 00 corresponding to PWM1A, with pin 01 displaying the inverse of this duty cycle.

8. **Results Verification**:
   - To verify the results, maintain the duty cycle at 20% and recalculate T_{on} with the T_{Board} value obtained from the datasheet, i.e., $T_{on} = 14999 \times 0.2 = 3000$. Check that these values are correctly configured in your Simulink model.
   - In Simulink, use the 'function' block to apply the equation $u[1] \times 14999$ and observe changes in the outputs. Here is what the configured function block looks like in Simulink:

Basic Schema



Desired Schema

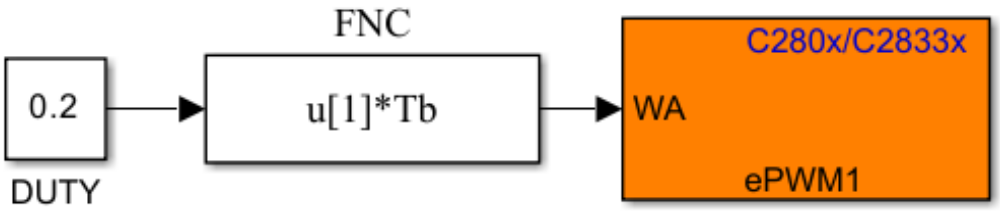


Table 1: Transition from the basic schema to the desired schema.

By following these detailed instructions, you should be able to set up, test, and validate the operation of PWM on the TI F28335 board, relying on MATLAB and Simulink development tools and test equipment like the oscilloscope.

5 Explanation of the Offset Between PWM1A and PWM1B Signals on the Oscilloscope

When using pulse width modulation (PWM) to control devices such as motors or switches, it is crucial to understand the behavior of the PWM1A and PWM1B signals, especially their synchronization. An important aspect of this synchronization is managing the temporal offset between these two signals to prevent short-circuit risks. Here is a structured explanation of the observed phenomenon and the associated settings in the ePWM block.

Operation of Deadband and Visualization of the Gap

1. **Purpose of the Deadband**

   - The Deadband is used in PWM configurations to prevent the short-circuit that could occur from the simultaneous activation of control transistors in an H-bridge (typically used to control motors). This mechanism is essential for protecting the circuit against damage caused by excessive currents.

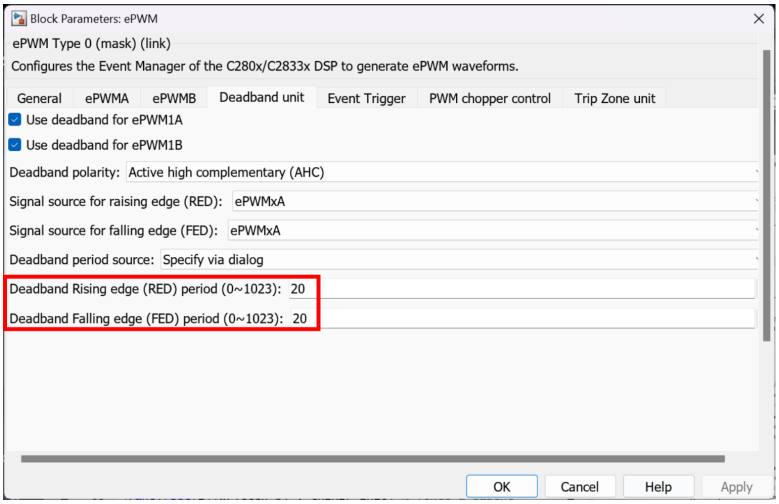


Figure 14: Screenshot of the Deadband settings in the ePWM configuration panel, highlighting the need to set both the Rising and Falling edge periods to 20 units.

2. **Configuration of the Deadband**

   - In the development software or PWM controller interface, you can set the Deadband Rising Edge (RED) and Deadband Falling Edge (FED) periods. A common value for these parameters is 20 clock cycle periods, which creates a small offset between the closing of one switch and the opening of the other.

3. **Visualization on the Oscilloscope**

   - To observe this offset, connect channel 1 of the oscilloscope to pin 00 (PWM1A) and channel 2 to pin 01 (PWM1B). Configure the oscilloscope to simultaneously display both signals. You will notice a slight temporal offset where, when one signal is high (ON), the other begins to go high (ON) after a short interval.

Logic of Switching Phases

To further clarify, consider the following sequence in a complete cycle of PWM with Deadband:

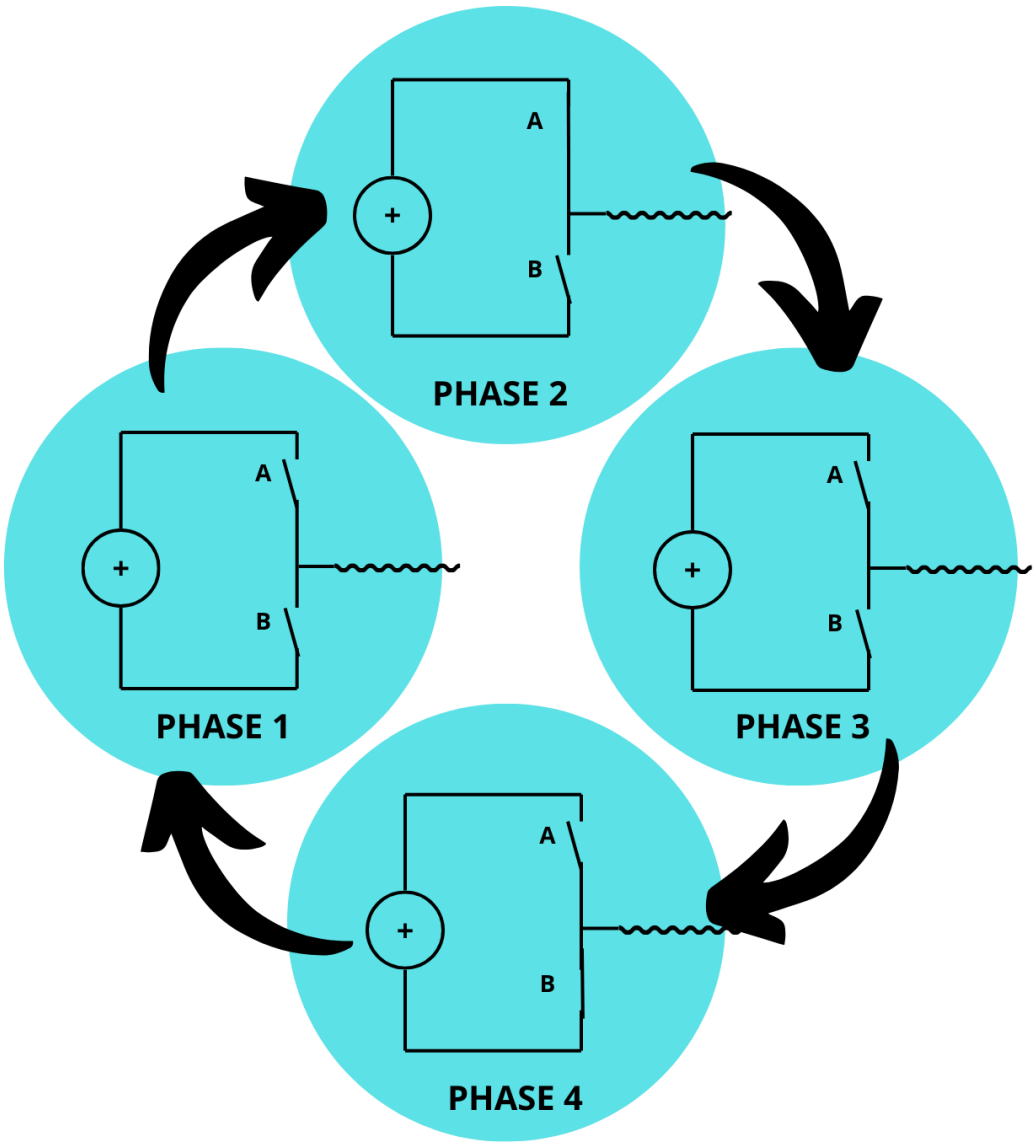


Figure 15: Illustration of the switching logic phases in PWM control.

- **Phase 1**: None of the switches is closed. Both signals are low (OFF).
- **Phase 2**: Only switch A is closed. PWM1A is high (ON), and PWM1B is still low (OFF), waiting for the Deadband delay.
- **Phase 3**: None of the switches is closed again. A transition where both signals return to low, preparing for the switch to B.
- **Phase 4**: Only switch B is closed. PWM1B is high (ON), while PWM1A goes low (OFF) after the Deadband delay.

These phases illustrate how the Deadband separates the drive periods of the two switches to ensure that they are never ON at the same time, which could cause a short-circuit. By following this logic, PWM control systems ensure safe and efficient operation of motorized applications or any other application requiring precise power control.

6 Exploration and Adjustment of PWM Signals

Manipulation et Observation des Paramètres

During the manipulation of embedded systems such as the TI F28335 board, it is essential to understand how parameter adjustments affect the observed results. Here are some tips for actively adjusting parameters and directly observing their impacts on the oscilloscope:

1. **Adjusting PWM Parameters**: Modify the switching frequency and duty cycle in the ePWM block of your Simulink model. For example, change the frequency from 10 kHz to 20 kHz or adjust the duty cycle from 20% to 50%. Observe the changes in the PWM signal shape on the oscilloscope.
2. **Observing Changes**: After each adjustment, compile and download the code to the board. Use the oscilloscope to see how the signal changes, paying attention to the amplitude, signal period, as well as the clarity and regularity of signal transitions.
3. **Using Multiple Channels of the Oscilloscope**: Connect channel 1 of the oscilloscope to pin GPI0-00 to observe the PWM signal (Signal A). Connect channel 2 to pin GPI0-01 to visualize another output signal (Signal B). This allows you to see how the two signals interact or differ in real-time.
4. **Analyzing Signals A and B**: Examine both signals to understand their interactions or differences. This can be particularly useful for diagnosing timing or synchronization issues between control signals and received responses.

Optimizing Oscilloscope Observation

- **Checking Scale and Settings**: Ensure that the oscilloscope is properly configured to measure the signals. Adjust the time and voltage scale for each channel to optimize signal visualization.
- **Using Measurement Functions**: Take advantage of the oscilloscope's built-in measurement functions to obtain precise data on frequency, duty cycle, and maximum and minimum voltage levels.
- **Experimenting with Oscilloscope Settings**: Experiment with trigger settings, hold-off, and capture modes to better understand signal behavior under different conditions.

By following these steps, you can not only more effectively identify problems but also deepen your understanding of how the TI F28335 board operates in relation to your configuration and code.

Integration of Parameter Coding in MATLAB

An effective method for managing and adjusting parameters in your Simulink model is to use a MATLAB script to define and modify these parameters. This allows for centralized and simplified modification of values used in Simulink, especially when working with recurring parameters such as the timer period or duty cycle.

Parameter Coding in a MATLAB Script

1. **Creating the Script**: Create a MATLAB script in the same folder as your Simulink project file. This allows the script to run in the same workspace as your model.
2. **Defining Parameters**: In the script, define variables for all key parameters of your model, such as the switching frequency (`fs`), duty cycle (`D`), and timer period (`T_board`). For example:

```
fs = 10e3;  % 10 kHz switching frequency
D = 0.2;    % 20% duty cycle
main_clock = 150e6;  % 150 MHz main clock
T_board = main_clock / fs;  % Calculating the period
```

3. **Advantages of Coding**:
   - **Flexibility**: Modifying a parameter in the script automatically updates the Simulink model upon the next execution.
   - **Reproducibility**: Facilitates the reproduction of configurations for different tests or simulations.
   - **Automation**: Enables automation of tests with different configurations by simply changing values in the script.

By integrating these practices into your workflow, you can significantly improve the efficiency and accuracy of your developments on the TI F28335 board, while simplifying parameter management through the use of MATLAB scripts.

Appendix

A TI F28335 controlCARD Pin-Out Table

F28335 controlCARD [R1.0] DIMM100 pin-out

V33D-ISO	1	51	V33D-ISO
ISO-RX-RS232	2	52	ISO-TX-RS232
	3	53	
	4	54	
	5	55	
GND_ISO	6	56	GND_ISO
ADCIN-B0	7	57	ADCIN-A0
GND	8	58	GND
ADCIN-B1	9	59	ADCIN-A1
GND	10	60	GND
ADCIN-B2	11	61	ADCIN-A2
GND	12	62	GND
ADCIN-B3	13	63	ADCIN-A3
GND	14	64	GND
ADCIN-B4	15	65	ADCIN-A4
	16	66	
ADCIN-B5	17	67	ADCIN-A5
GPIO-58 / MCLKR-A / XD21 (EMIF)	18	68	GPIO-59 / MFSR-A / XD20 (EMIF)
ADCIN-B6	19	69	ADCIN-A6
GPIO-60 / MCLKR-B / XD19 (EMIF)	20	70	GPIO-61 / MFSR-B / XD18 (EMIF)
ADCIN-B7	21	71	ADCIN-A7
GPIO-62 / SCIRX-C / XD17 (EMIF)	22	72	GPIO-63 / SCITX-C / XD16 (EMIF)
GPIO-00 / EPWM-1A	23	73	GPIO-01 / EPWM-1B / MFSR-B
GPIO-02 / EPWM-2A	24	74	GPIO-03 / EPWM-2B / MCLKR-B
GPIO-04 / EPWM-3A	25	75	GPIO-05 / EPWM-3B / MFSR-A / ECAP-1
GPIO-06 / EPWM-4A / SYNCI / SYNCO	26	76	GPIO-07 / EPWM-4B / MCLKR-A / ECAP-2
GND	27	77	+5V in
GPIO-08 / EPWM-5A / CANTX-B / ADCSOC-A	28	78	GPIO-09 / EPWM-5B / SCITX-B / ECAP-3
GPIO-10 / EPWM-6A / CANRX-B / ADCSOC-B	29	79	GPIO-11 / EPWM-6B / SCIRX-B / ECAP-4
GPIO-48 / ECAP5 / XD31 (EMIF)	30	80	GPIO-49 / ECAP6 / XD30 (EMIF)
GPIO-84	31	81	GPIO-85
GPIO-86	32	82	+5V in
GPIO-12 / TZ-1 / CANTX-B / MDX-B	33	83	GPIO-13 / TZ-2 / CANRX-B / MDR-B
GPIO-15 / TZ-4 / SCIRX-B / MFSX-B	34	84	GPIO-14 / TZ-3 / SCITX-B / MCLKX-B
GPIO-24 / ECAP-1 / EQEPA-2 / MDX-B	35	85	GPIO-25 / ECAP-2 / EQEPB-2 / MDR-B
GPIO-26 / ECAP-3 / EQEPI-2 / MCLKX-B	36	86	GPIO-27 / ECAP-4 / EQEPS-2 / MFSX-B
GND	37	87	+5V in
GPIO-16 / SPISIMO-A / CANTX-B / TZ-5	38	88	GPIO-17 / SPISOMI-A / CANRX-B / TZ-6
GPIO-18 / SPICLK-A / SCITX-B	39	89	GPIO-19 / SPISTE-A / SCIRX-B
GPIO-20 / EQEPA-1 / MDX-A / CANTX-B	40	90	GPIO-21 / EQEPB-1 / MDR-A / CANRX-B
GPIO-22 / EQEPS-1 / MCLKX-A / SCITX-B	41	91	GPIO-23 / EQEPI-1 / MFSX-A / SCIRX-B
GPIO-87	42	92	+5V in
GPIO-28 / SCIRX-A / Resv / TZ5	43	93	GPIO-29 / SCITX-A / Resv / TZ6
GPIO-30 / CANRX-A	44	94	GPIO-31 / CANTX-A
GPIO-32 / I2CSDA / SYNCI / ADCSOCA	45	95	GPIO-33 / I2CSCL / SYNCO / ADCSOCB
GPIO-34 / ECAP1 / XREADY (EMIF)	46	96	+5V in
GND	47	97	TDI
TCK	48	98	TDO
TMS	49	99	TRSTn
EMU1	50	100	EMU0

Texas Instruments – C2000

Figure 16: TI F28335 controlCARD Pin-Out Table