Efficient Microcontroller Peripheral Modeling with PLECS®

Mr. Munadir Ahmed
Plexim Inc.
5 Upland Road, Suite 4
Cambridge, MA 02140

1 Introduction

When modeling power controls at the system level with a circuit simulator such as PLECS, the focus is typically on modeling the algorithms, while models for Micro Controller Unit (MCU) peripherals are often idealized to improve overall simulation efficiency and speed. In fact, frequently, the Analog-to-Digital (ADC) peripheral modules are modeled as simple sample-and-hold blocks, and basic pulse generators are used for generating Pulse Width Modulation (PWM) waveforms. These simplified models have inherent limitations in comparison to the functionality provided by the real peripheral modules. As a result, the fidelity of the system model is substantially reduced and effects that are critical to the power controls may be lost or inaccurately simulated. Furthermore, the limited functionality provided by basic peripheral models may be insufficient to model advanced modulation and sampling techniques.

For example, typical PWM modules provide the flexibility to trigger start of conversion (SOC) of the ADC module at different events. For systems with high current or voltage ripple, this provides engineers the ability to sample the ADC inputs at a desired instance of the PWM waveform. Both the PWM and ADC modules can be used to trigger the control interrupt as would be done in the real system. Additionally, with high fidelity peripheral models (HFPMs) engineers can verify the effect of their PWM and ADC configuration on the overall system. It is therefore desirable to utilize detailed peripheral models to more accurately reflect the complex functionality offered by an MCU to facilitate the implementation of sophisticated control strategies. However, it is critical that such peripheral models are implemented in the most efficient fashion to ensure that their impact on simulation time is minimal.

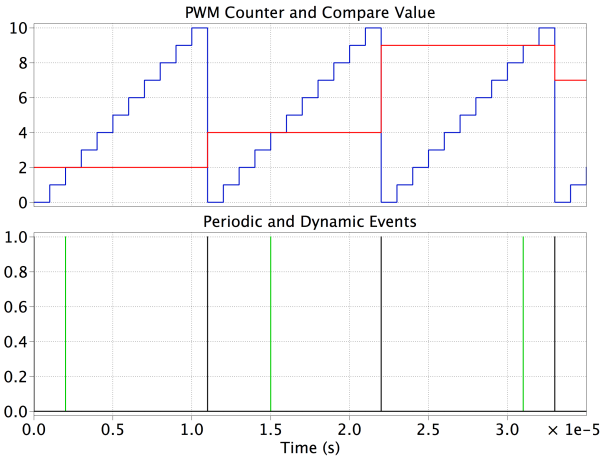

Fig. 1: Typical counter behavior of a PWM module.

2 Efficient Peripheral Modeling Using PLECS

Two major types of solvers are available to simulate power electronic systems. A fixed-step solver discretizes the modeled system to a user-specified step size. The solver does not have the ability to change the step size during the simulation to meet the accuracy requirements of the system. In the context of modeling a PWM module, the step size must be chosen to have enough resolution to capture the duty cycle, period, and dead time effects accurately. This would result in a step size defined by the counter period (e.g. 10 ns for a peripheral clocked at 100 MHz) and therefore result in a very inefficient simulation. To achieve higher simulation efficiency, the step size must be increased to multiples of the counter period at the expense of the available PWM resolution. The second class of solvers is the variable-step solver that has the ability to change the step-size during simulation. This dynamic nature allows a more efficient modeling of the PWM module. The solver takes steps ranging from small multiples of the counter period when capturing dead time effects to large multiples of the counter period to capture the duty cycle and period of the generated PWM signal. Compared to a fixed simulation step, this

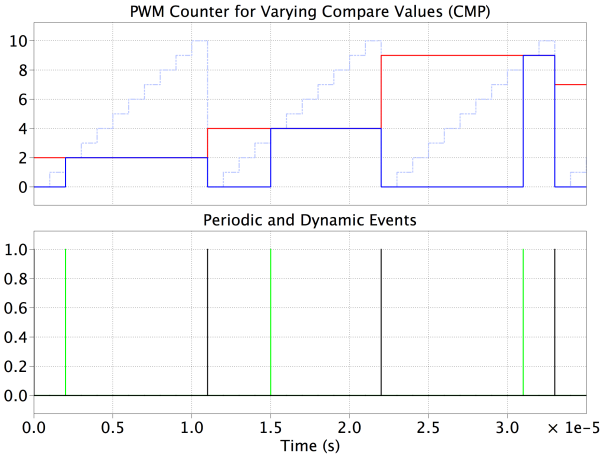

Fig. 2: Efficient implementation of a PWM behavior.

allows the user to model the PWM module with enhanced functionality very efficiently.

Fig. 1 shows a typical behavior of an actual PWM counter (blue trace) running with a fixed period. The compare value (red trace) is changed at every period of the PWM. In a typical modulator, the PWM outputs are changed at certain events that could either be periodic or dependent on an external configuration. In this particular case, the dynamic events (green trace) are determined by the compare value, while the periodic events (black trace) are determined by the PWM period. For a high fidelity PWM model, with a full duty cycle resolution, we either need a fixed-step solver with a step size defined by the counter period or a variable-step solver and the ability to invoke a solver step at every periodic and dynamic event. The features required for the efficient modeling of HFPMs are available in the C-Script block in the PLECS component library. This block allows users to develop custom controllers and components for use in their simulation. It provides the advanced capabilities of the C programming language combined with the flexibility of using variable and/or fixed sample time settings. A fixed sample time for a C-Script means that the block is evaluated with a specified period. A variable sample time gives the user the ability to manually specify the next evaluation of the C-Script block. This makes the C-Script a versatile tool that is well suited for the efficient development of high fidelity peripheral models.

For the efficient modeling of a PWM module, the C-Script block is defined to use a fixed sample time, which determines the periodic events (black trace) at the PWM period. At those events, the time for the next dynamic event (green trace) is calculated and the solver evaluates the block at that instant using the variable sample time of the C-Script block. This allows the PWM model to be evaluated at only the relevant points in time and therefore is the most efficient approach for implementing a high fidelity PWM. As seen in Fig. 2, the modeled counter value (blue trace) is only updated at those instants, but coincides with the actual counter value (dotted trace). This approach obtains the full PWM resolution without requiring a very inefficient sampling of the model based on the counter period.

3 Modeling of a TI ePWM (Type 0) Module in PLECS

A Type 0 ePWM module [1] from TI's C2000 series was modeled in PLECS. This module is capable of generating two independent PWM output signals. It also includes functionality provided by an Event Trigger submodule that can be used to dynamically trigger ADC conversions and/or control interrupts. Furthermore, it contains a Deadband submodule that can be configured to invert the PWM outputs or to implement a dead time between the two outputs.

Fig. 3 shows a PLECS model of the ePWM module and its parameter mask. The block's configuration is split into static and non-static parameters. Users can specify general static parameters such as the basic counter period (System Clock), the PWM period (TBPRD) and the behavior of the counter (TBCTL) in the mask parameters. Furthermore, the ETx parameters define the behavior of the Event Trigger submodule and the DBx parameters can be used to configure the Deadband submodule. These parameters directly correspond to the registers used in the hardware and can be entered as integer, binary or hexadecimal values. This gives the user the ability to test a hardware configuration in a simulation environment. The two counter compare inputs (CMPA and CMPB) define the dynamic events of the ePWM model and the Action Qualifier Control Registers (AQCTLx) inputs are used to configure the actions at those events. For example, the output EPWMA can be configured to be set high when the counter equals CMPB and reset at the PWM period. The registers are implemented as inputs to the ePWM block, and can be modified while the simulation is running. More detailed information on the Type 0 ePWM module and its configurations is found in the TI technical reference guides, available on the TI website.

4 Simulation Results

To further illustrate the advantages of HFPMs, a current-controlled buck converter was developed with the above discussed ePWM module as well as

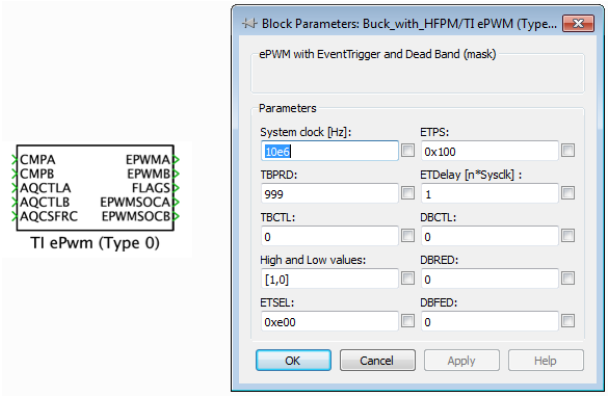

Fig. 3: Model of a TI Type 0 ePWM module implemented in PLECS

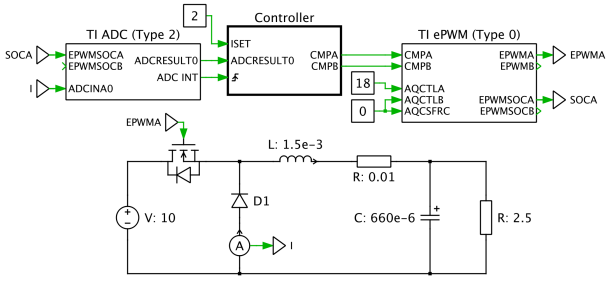

Fig. 4: Current-controlled buck converter with peripheral models.

a HFPM for a TI Type 2 ADC module[2]. As seen in Fig. 4, the current measurement is realized using a simple shunt resistor in series with the diode. Such a configuration requires the current to be sampled while the diode is conducting, ideally at the center of the conducting phase.

The ePWM is configured to operate in up-counting mode with a frequency of 10 kHz. Furthermore, the EPWMA signal is configured to be set high when the counter equals zero and set low when a CTR = CMPA event (green trace) occurs as defined by the AQCTLA register. Fig. 5 shows the resulting characteristics of the ePWM module. As already mentioned, this block is only evaluated at the relevant points in time and therefore the counter value is only updated at those instants.

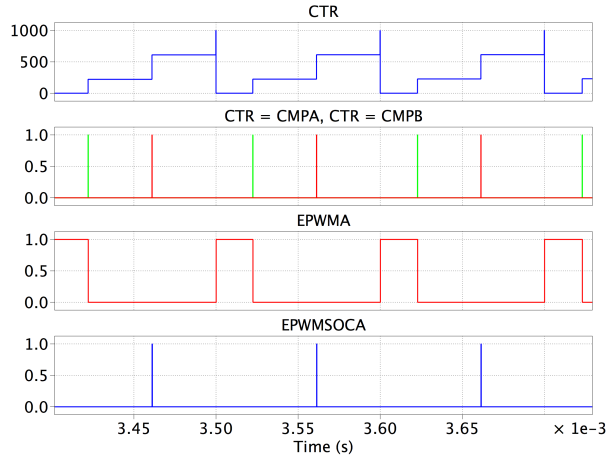

Fig. 5: PWM modulation and conversion trigger based on the ePWM model.

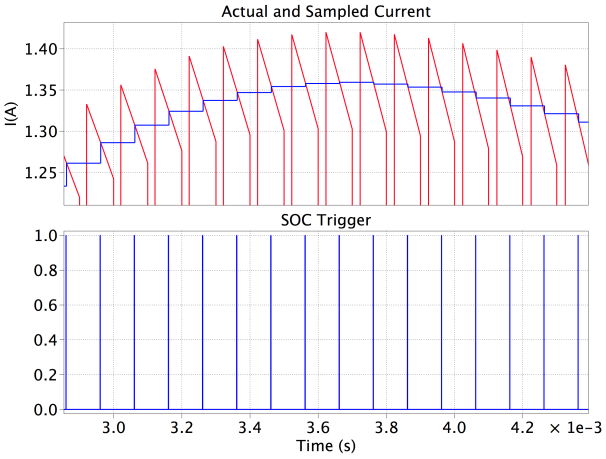

Fig. 6: Current measurement invoked by the ePWM module.

Additionally, the internal Event Trigger module is configured to invoke an EPWMSOCA trigger for every CTR = CMPB event (red trace). This is used to trigger an ADC SOC for current measurement, as seen in Fig. 4. Once the measurement is finished, an ADC interrupt is generated to trigger the controller, which then updates the CMPA and CMPB registers for the ePWM module.

Fig. 6 shows the actual diode current (red trace) and the sampled current (blue trace) during the startup transient. As shown, the ADC is always triggered to measure the current at the midpoint of the interval during which the diode is conducting current. The graph also shows the EPWMSOCA signal used as the SOC trigger for the ADC module.

5 Conclusion

The HFPMs allow users to develop models that are a closer representation of the real system. The proposed implementation enables an efficient integration of these models into a system level simulation without limiting the PWM resolution or the supported functionalities.

References

[1] "Tms320x2833x, 2823x enhanced pulse width tms320x2833x, 2823x enhanced pulse width modulator (epwm) module," July 2009.

[2] "Tms320x2833x analog-to-digital converter (adc) module," *Literature Number: SPRU812A*, October 2007.