# TMS320x2802x, 2803x Piccolo High Resolution Pulse Width Modulator (HRPWM)

# Reference Guide

Literature Number: SPRUGE8E
February 2009–Revised October 2011

# Contents

## List of Figures

## List of Tables

# Read This First

## About This Manual

This document describes the operation of the high-resolution extension to the pulse width modulator (HRPWM) on the TMS320x2802x, 2803x Piccolo™ devices. The HRPWM module described in this reference guide is a Type 1 HRPWM. See the *TMS320x28xx, 28xxx DSP Peripheral Reference Guide* (SPRU566) for a list of all devices with an HRPWM module of the same type, to determine the differences between types, and for a list of device-specific differences within a type.

## Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion.

## Related Documentation From Texas Instruments

The following documents describe the C2000™ devices and related support tools. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

The current documentation that describes the devices, related peripherals, and other technical collateral, is available in the C2000 product folder at: www.ti.com/c2000.

**Data Manuals—**

**SPRS523** — **TMS320F28020, TMS320F28021, TMS320F28022, TMS320F28023, TMS320F28026, TMS320F28027 Piccolo Microcontrollers Data Manual** contains the pinout, signal descriptions, as well as electrical and timing specifications for the 2802x devices.

**SPRZ292** — **TMS320F28020, TMS320F28021, TMS320F28022, TMS320F28023, TMS320F28026, TMS320F28027 Piccolo MCU Silicon Errata** describes known advisories on silicon and provides workarounds.

**SPRS584** — **TMS320F28032, TMS320F28033, TMS320F28034, TMS320F28035 Piccolo Microcontrollers Data Manual** contains the pinout, signal descriptions, as well as electrical and timing specifications for the 2803x devices.

**SPRZ295** — **TMS320F28032, TMS320F28033, TMS320F28034, TMS320F28035 Piccolo MCU Silicon Errata** describes known advisories on silicon and provides workarounds.

**CPU User's Guides—**

**SPRU430** — **TMS320C28x CPU and Instruction Set Reference Guide** describes the central processing unit (CPU) and the assembly language instructions of the TMS320C28x fixed-point digital signal processors (DSPs). It also describes emulation features available on these DSPs.

**Peripheral Guides—**

**SPRUFN3** — **TMS320x2802x Piccolo System Control and Interrupts Reference Guide** describes the various interrupts and system control features of the 2802x microcontrollers (MCUs).

**SPRUGL8** — **TMS320x2803x Piccolo System Control and Interrupts Reference Guide** describes the various interrupts and system control features of the 2803x microcontrollers (MCUs).

**SPRU566** — **TMS320x28xx, 28xxx DSP Peripheral Reference Guide** describes the peripheral reference guides of the 28x digital signal processors (DSPs).

**SPRUGO0** — **TMS320x2803x Piccolo Boot ROM Reference Guide** describes the purpose and features of the bootloader (factory-programmed boot-loading software) and provides examples of code. It also describes other contents of the device on-chip boot ROM and identifies where all of the information is located within that memory.

**SPRUFN6** — **TMS320x2802x Piccolo Boot ROM Reference Guide** describes the purpose and features of the bootloader (factory-programmed boot-loading software) and provides examples of code. It also describes other contents of the device on-chip boot ROM and identifies where all of the information is located within that memory.

**SPRUGE6** — **TMS320x2803x Piccolo Control Law Accelerator (CLA) Reference Guide** describes the operation of the Control Law Accelerator (CLA).

**SPRUGE2** — **TMS320x2803x Piccolo Local Interconnect Network (LIN) Module Reference Guide** describes the operation of the Local Interconnect Network (LIN) Module.

**SPRUFK8** — **TMS320x2803x Piccolo Enhanced Quadrature Encoder Pulse (eQEP) Reference Guide** describes the operation of the Enhanced Quadrature Encoder Pulse (eQEP) module, which is used for interfacing with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine in high performance motion and position control systems. It includes the module description on registers.

**SPRUGL7** — **TMS320x2803x Piccolo Enhanced Controller Area Network (eCAN) Reference Guide** describes the operation of the Enhanced Controller Area Network (eCAN) which uses established protocol to communicate serially with other controllers in electrically noisy environments.

**SPRUGE5** — **TMS320x2802x, 2803x Piccolo Analog-to-Digital Converter (ADC) and Comparator Reference Guide** describes how to configure and use the on-chip ADC module, which is a 12-bit pipelined ADC.

**SPRUGE9** — **TMS320x2802x, 2803x Piccolo Enhanced Pulse Width Modulator (ePWM) Module Reference Guide** describes the main areas of the enhanced pulse width modulator that include digital motor control, switch mode power supply control, UPS (uninterruptible power supplies), and other forms of power conversion.

**SPRUGE8** — **TMS320x2802x, 2803x Piccolo High-Resolution Pulse Width Modulator (HRPWM)** describes the operation of the high-resolution extension to the pulse width modulator (HRPWM).

**SPRUGH1** — **TMS320x2802x, 2803x Piccolo Serial Communications Interface (SCI) Reference Guide** describes how to use the SCI.

**SPRUFZ8** — **TMS320x2802x, 2803x Piccolo Enhanced Capture (eCAP) Module Reference Guide** describes the enhanced capture module. It includes the module description and registers.

**SPRUG71** — **TMS320x2802x, 2803x Piccolo Serial Peripheral Interface (SPI) Reference Guide** describes the SPI - a high-speed synchronous serial input/output (I/O) port - that allows a serial bit stream of programmed length (one to sixteen bits) to be shifted into and out of the device at a programmed bit-transfer rate.

**SPRUFZ9** — **TMS320x2802x, 2803x Piccolo Inter-Integrated Circuit (I2C) Reference Guide** describes the features and operation of the inter-integrated circuit (I2C) module.

**Tools Guides**—

**SPRU513** — **TMS320C28x Assembly Language Tools v5.0.0 User's Guide** describes the assembly language tools (assembler and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for the TMS320C28x device.

**SPRU514** — **TMS320C28x Optimizing C/C++ Compiler v5.0.0 User's Guide** describes the TMS320C28x™ C/C++ compiler. This compiler accepts ANSI standard C/C++ source code and produces TMS320 DSP assembly language source code for the TMS320C28x device.

**SPRU608** — **TMS320C28x Instruction Set Simulator Technical Overview** describes the simulator, available within the Code Composer Studio for TMS320C2000 IDE, that simulates the instruction set of the C28x™ core.

Reference Guide
SPRUGE8E–February 2009–Revised October 2011

# High-Resolution Pulse Width Modulator (HRPWM)

This document is used in conjunction with the device-specific *Enhanced Pulse Width Modulator (ePWM) Module Reference Guide.*

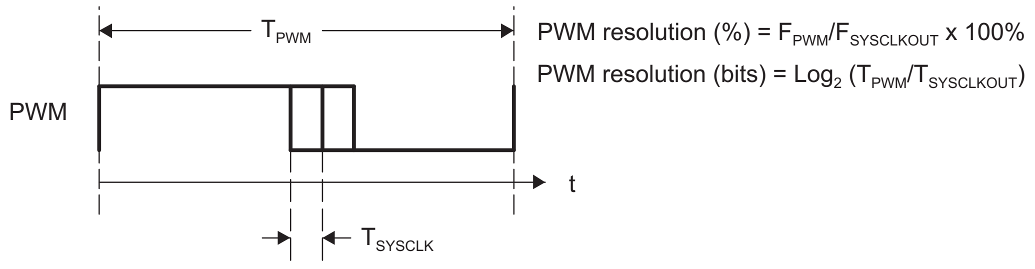
The HRPWM module extends the time resolution capabilities of the conventionally derived digital pulse width modulator (PWM). HRPWM is typically used when PWM resolution falls below ~ 9-10 bits. The key features of HRPWM are:

- Extended time resolution capability
- Used in both duty cycle and phase-shift control methods
- Finer time granularity control or edge positioning using extensions to the Compare A and Phase registers
- Implemented using the A signal path of PWM, i.e., on the EPWMxA output.
- Self-check diagnostics software mode to check if the micro edge positioner (MEP) logic is running optimally
- Enables high-resolution output on B signal path of PWM via PWM A and B channel path swapping
- Enables high-resolution output on B signal output via inversion of A signal output
- Enables high-resolution period control on the ePWMxA output on devices with a type 1 ePWM module. See the device-specific data manual to determine if your device has a type 1 ePWM module for high-resolution period support. The ePWMxB output will have +/- 1-2 cycle jitter in this mode.

## 1 Introduction

The ePWM peripheral is used to perform a function that is mathematically equivalent to a digital-to-analog converter (DAC). As shown in Figure 1, the effective resolution for conventionally generated PWM is a function of PWM frequency (or period) and system clock frequency.

**Figure 1. Resolution Calculations for Conventionally Generated PWM**



PWM resolution (%) = $F_{PWM}/F_{SYSCLKOUT}$ x 100%

PWM resolution (bits) = $Log_2 (T_{PWM}/T_{SYSCLKOUT})$

If the required PWM operating frequency does not offer sufficient resolution in PWM mode, you may want to consider HRPWM. As an example of improved performance offered by HRPWM, Table 1 shows resolution in bits for various PWM frequencies. These values assume a MEP step size of 180 ps. See the device-specific datasheet for typical and maximum performance specifications for the MEP.

**Table 1. Resolution for PWM and HRPWM**

| **PWM Freq** | **Regular Resolution (PWM)** | | | | **High Resolution (HRPWM)** | |
|---|---|---|---|---|---|---|
| | **60 MHz SYSCLKOUT** | | **50 MHz SYSCLKOUT** | | | |
| **(kHz)** | Bits | % | Bits | % | Bits | % |
| 20 | 11.6 | 0.0 | 11.3 | 0 | 18.1 | 0.000 |
| 50 | 10.2 | 0.1 | 10 | 0.1 | 16.8 | 0.001 |
| 100 | 9.2 | 0.2 | 9 | 0.2 | 15.8 | 0.002 |
| 150 | 8.6 | 0.3 | 8.4 | 0.3 | 15.2 | 0.003 |
| 200 | 8.2 | 0.3 | 8 | 0.4 | 14.8 | 0.004 |
| 250 | 7.9 | 0.4 | 7.6 | 0.5 | 14.4 | 0.005 |
| 500 | 6.9 | 0.8 | 6.6 | 1 | 13.4 | 0.009 |
| 1000 | 5.9 | 1.7 | 5.6 | 2 | 12.4 | 0.018 |
| 1500 | 5.3 | 2.5 | 5.1 | 3 | 11.9 | 0.027 |
| 2000 | 4.9 | 3.3 | 4.6 | 4 | 11.4 | 0.036 |

Although each application may differ, typical low frequency PWM operation (below 250 kHz) may not require HRPWM. HRPWM capability is most useful for high frequency PWM requirements of power conversion topologies such as:

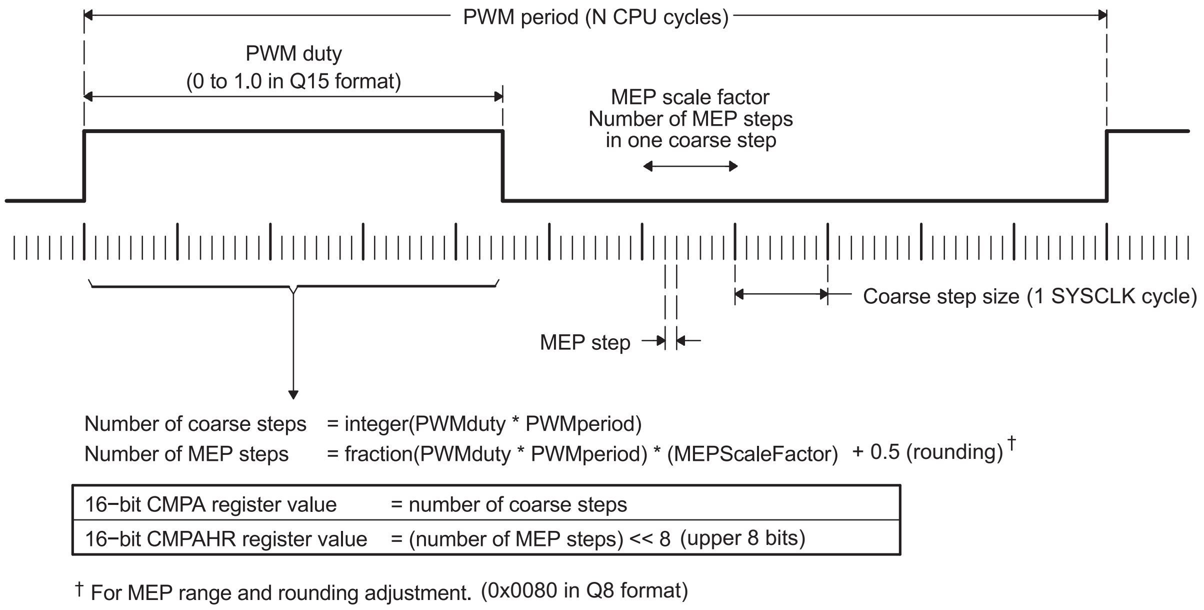
- Single-phase buck, boost, and flyback
- Multi-phase buck, boost, and flyback
- Phase-shifted full bridge
- Direct modulation of D-Class power amplifiers

## 2 Operational Description of HRPWM

The HRPWM is based on micro edge positioner (MEP) technology. MEP logic is capable of positioning an edge very finely by sub-dividing one coarse system clock of a conventional PWM generator. The time step accuracy is on the order of 150 ps. See the device-specific data sheet for the typical MEP step size on a particular device. The HRPWM also has a self-check software diagnostics mode to check if the MEP logic is running optimally, under all operating conditions. Details on software diagnostics and functions are in Section 2.4.

Figure 2 shows the relationship between one coarse system clock and edge position in terms of MEP steps, which are controlled via an 8-bit field in the Compare A extension register (CMPAHR).

**Figure 2. Operating Logic Using MEP**


PWM period (N CPU cycles)
PWM duty
(0 to 1.0 in Q15 format)
MEP scale factor
Number of MEP steps
in one coarse step
Coarse step size (1 SYSCLK cycle)
MEP step
Number of coarse steps = integer(PWMduty * PWMperiod)
Number of MEP steps = fraction(PWMduty * PWMperiod) * (MEPScaleFactor) + 0.5 (rounding) †
16−bit CMPA register value = number of coarse steps
16−bit CMPAHR register value = (number of MEP steps) << 8 (upper 8 bits)
† For MEP range and rounding adjustment. (0x0080 in Q8 format)

To generate an HRPWM waveform, configure the TBM, CCM, and AQM registers as you would to generate a conventional PWM of a given frequency and polarity. The HRPWM works together with the TBM, CCM, and AQM registers to extend edge resolution, and should be configured accordingly. Although many programming combinations are possible, only a few are needed and practical. These methods are described in Section 2.5.

Registers discussed but not found in this document can be seen in the device-specific *Enhanced Pulse Width Modulator (ePWM) Module Reference Guide*.

The HRPWM operation is controlled and monitored using the following registers:

**Table 2. HRPWM Registers**

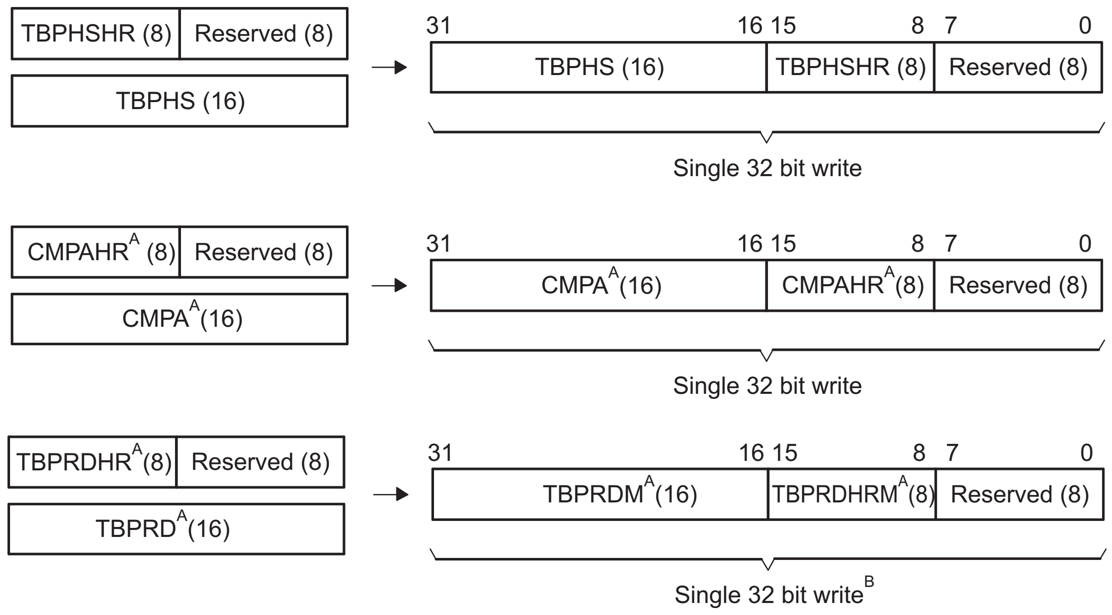
| mnemonic | Address Offset | Shadowed | Description |
|---|---|---|---|
| TBPHSHR | 0x0002 | No | Extension Register for HRPWM Phase (8 bits) |
| TBPRDHR | 0x0006 | Yes | Extension Register for HRPWM Period (8 bits) |
| CMPAHR | 0x0008 | Yes | Extension Register for HRPWM Duty (8 bits) |
| HRCNFG | 0x0020 | No | HRPWM Configuration Register |
| HRPWR | 0x0021 | No | HRPWM Power Register |
| HRMSTEP | 0x0026 | No | HRPWM MEP Step Register |
| TBPRDHRM | 0x002A | Yes | Extension Mirror Register for HRPWM Period (8 bits) |
| CMPAHRM | 0x002C | Yes | Extension Mirror Register for HRPWM Duty (8 bits) |

## 2.1 Controlling the HRPWM Capabilities

The MEP of the HRPWM is controlled by three extension registers, each 8-bits wide. These HRPWM registers are concatenated with the 16-bit TBPHS, TBPRD, and CMPA registers used to control PWM operation.

- TBPHSHR - Time Base Phase High Resolution Register
- CMPAHR - Counter Compare A High Resolution Register
- TBPRDHR - Time Base Period High Resolution Register. (available on some devices)

**Figure 3. HRPWM Extension Registers and Memory Configuration**



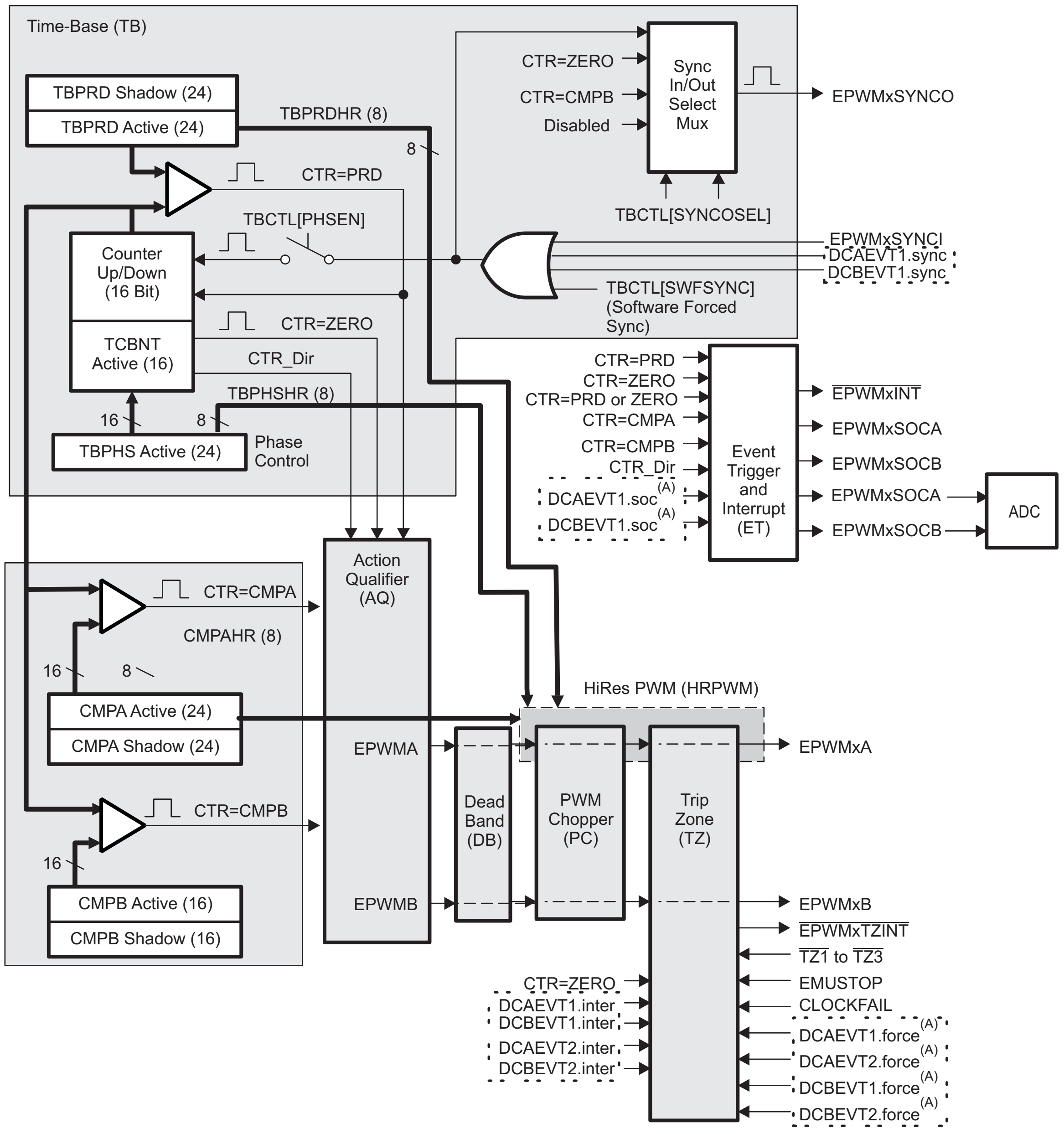
A These registers are mirrored and can be written to at two different memory locations (mirrored registers have an "M" suffix ( i.e. CMPA mirror = CMPAM). Reads of the high-resolution mirror registers will result in indeterminate values.

B TBPRDHR and TBPRD may be written to as a 32-bit value only at the mirrored address

Not all devices may have TBPRD and TBPRDHR registers. See device-specific data sheet for more information

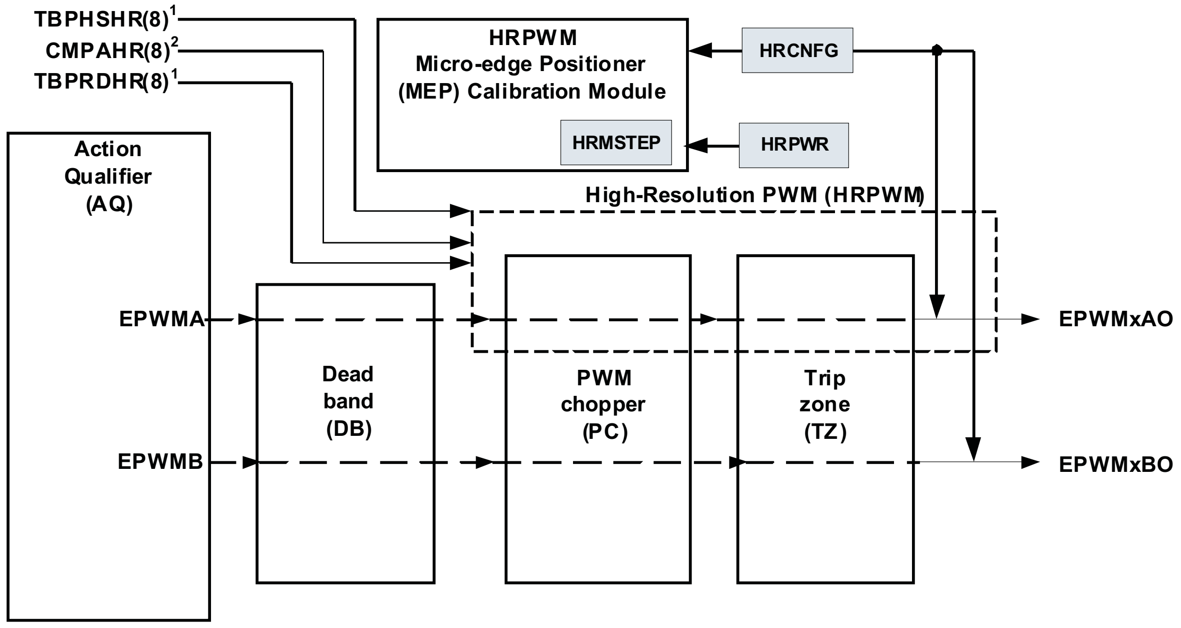
HRPWM capabilities are controlled using the Channel A PWM signal path. HRPWM support on the channel B signal path is available by properly configuring the HRCNFG register. Figure 4 shows how the HRPWM interfaces with the 8-bit extension registers.

**Figure 4. HRPWM System Interface**



A These events are generated by the type 1 ePWM digital compare (DC) submodule based on the levels of the COMPxOUT and $\overline{\text{TZ}}$ signals.

Figure 5. HRPWM Block Diagram



(1) From ePWM Time-base (TB) submodule

(2) From ePWM counter-compare (CC) submodule

## 2.2 Configuring the HRPWM

Once the ePWM has been configured to provide conventional PWM of a given frequency and polarity, the HRPWM is configured by programming the HRCNFG register located at offset address 20h. This register provides the following configuration options :

**Edge Mode** — The MEP can be programmed to provide precise position control on the rising edge (RE), falling edge (FE) or both edges (BE) at the same time. FE and RE are used for power topologies requiring duty cycle control(CMPA high-resolution control), while BE is used for topologies requiring phase shifting, e.g., phase shifted full bridge (TBPHS or TBPRD high-resolution control).

**Control Mode** — The MEP is programmed to be controlled either from the CMPAHR register (duty cycle control) or the TBPHSHR register (phase control). RE or FE control mode should be used with CMPAHR register. BE control mode should be used with TBPHSHR register. When the MEP is controlled from the TBPRDHR register (period control) the duty cycle and phase can also be controlled via their respective high-resolution registers.

**Shadow Mode** — This mode provides the same shadowing (double buffering) option as in regular PWM mode. This option is valid only when operating from the CMPAHR and TBPRDHR registers and should be chosen to be the same as the regular load option for the CMPA register. If TBPHSHR is used, then this option has no effect.

**High-Resolution B Signal Control** — The B signal path of an ePWM channel can generate a high-resolution output by either swapping the A and B outputs (the high- resolution signal will appear on ePWMxB instead of ePWMxA) or by outputting an inverted version of the high-resolution ePWMxA signal on the ePWMxB pin.

**Auto-conversion Mode** — This mode is used in conjunction with the scale factor optimization software only. For a type 1 HRPWM module, if auto-conversion is enabled, CMPAHR = fraction(PWMduty*PWMperiod)<<8. The scale factor optimization software will calculate the MEP scale factor in background code and automatically update the HRMSTEP register with the calculated number of MEP steps per coarse step. The MEP Calibration Module will then use the values in the HRMSTEP and CMPAHR register to automatically calculate the appropriate number of MEP steps represented by the fractional duty cycle and move the high-resolution ePWM signal edge accordingly. If auto-conversion is disabled, the CMPAHR register behaves like a type 0 HRPWM module and CMPAHR = (fraction(PWMduty * PWMperiod) * MEP Scale Factor + 0.5)<<8). All of these calculations will need to be performed by user code in this mode, and the HRMSTEP register is ignored. Auto-conversion for high-resolution period has the same behavior as auto-conversion for high-resolution duty cycle. Auto-conversion must always be enabled for high-resolution period mode.

## 2.3 Principle of Operation

The MEP logic is capable of placing an edge in one of 255 (8 bits) discrete time steps (see device-specific data sheet for typical MEP step size). The MEP works with the TBM and CCM registers to be certain that time steps are optimally applied and that edge placement accuracy is maintained over a wide range of PWM frequencies, system clock frequencies and other operating conditions. Table 3 shows the typical range of operating frequencies supported by the HRPWM.

**Table 3. Relationship Between MEP Steps, PWM Frequency and Resolution**

| System (MHz) | MEP Steps Per SYSCLKOUT [(1)] [(2)] [(3)] | PWM MIN (Hz) [(4)] | PWM MAX (MHz) | Res. @ MAX (Bits) [(5)] |
|---|---|---|---|---|
| 50.0 | 111 | 763 | 2.50 | 11.1 |
| 60.0 | 93 | 916 | 3.00 | 10.9 |
| 70.0 | 79 | 1068 | 3.50 | 10.6 |
| 80.0 | 69 | 1221 | 4.00 | 10.4 |
| 90.0 | 62 | 1373 | 4.50 | 10.3 |
| 100.0 | 56 | 1526 | 5.00 | 10.1 |

(1) System frequency = SYSCLKOUT, i.e., CPU clock. TBCLK =SYSCLKOUT.
(2) Table data based on a MEP time resolution of 180 ps (this is an example value, see the device-specific data sheet for MEP limits.
(3) MEP steps applied = $T_{SYSCLKOUT}$/180 ps in this example.
(4) PWM minimum frequency is based on a maximum period value, i.e., TBPRD = 65535. PWM mode is asymmetrical up-count.
(5) Resolution in bits is given for the maximum PWM frequency stated.

### 2.3.1 Edge Positioning

In a typical power control loop (e.g., switch modes, digital motor control [DMC], uninterruptible power supply [UPS]), a digital controller (PID, 2pole/2zero, lag/lead, etc.) issues a duty command, usually expressed in a per unit or percentage terms. Assume that for a particular operating point, the demanded duty cycle is 0.405 or 40.5% on time and the required converter PWM frequency is 1.25 MHz. In conventional PWM generation with a system clock of 60 MHz, the duty cycle choices are in the vicinity of 40.5%. In Figure 6, a compare value of 19 counts (i.e., duty = 39.6%) is the closest to 40.5% that you can attain. This is equivalent to an edge position of 316.7 ns instead of the desired 324 ns. This data is shown in Table 4.

By utilizing the MEP, you can achieve an edge position much closer to the desired point of 324 ns. Table 4 shows that in addition to the CMPA value, 44 steps of the MEP (CMPAHR register) will position the edge at 323.9 2 ns, resulting in almost zero error. In this example, it is assumed that the MEP has a step resolution of 180 ps.

**Figure 6. Required PWM Waveform for a Requested Duty = 40.5%**

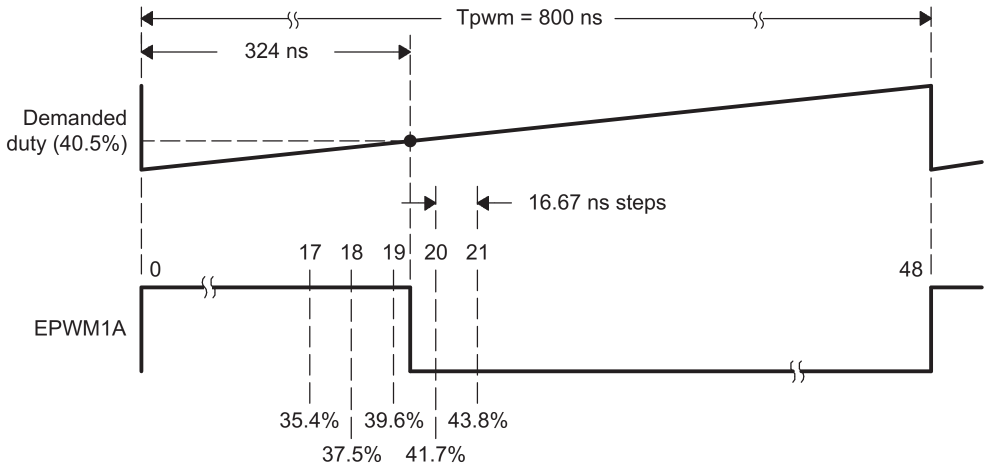


**Table 4. CMPA vs Duty (left), and [CMPA:CMPAHR] vs Duty (right)**

| CMPA (count) [(1)] [(2)] [(3)] | Duty % | High Time (ns) | CMPA (count) | CMPAHR (count) | Duty (%) | High Time (ns) |
|---|---|---|---|---|---|---|
| 15 | 31.25% | 250 | 19 | 40 | 40.40% | 323.2 |
| 16 | 33.33% | 267 | 19 | 41 | 40.42% | 323.38 |
| 17 | 35.42% | 283 | 19 | 42 | 40.45% | 323.56 |
| 18 | 37.50% | 300 | 19 | 43 | 40.47% | 323.74 |
| 19 | 39.58% | 316 | 19 | 44 | 40.49% | 323.92 |
| 20 | 41.67% | 333 | 19 | 45 | 40.51% | 324.1 |
| 21 | 43.75% | 350 | 19 | 46 | 40.54% | 324.28 |
| | | | 19 | 47 | 40.56% | 324.46 |
| Required | | | 19 | 48 | 40.58% | 324.64 |
| 19.4 | 40.50% | 324 | 19 | 49 | 40.60% | 324.82 |

(1) System clock, SYSCLKOUT and TBCLK = 60 MHz, 16.67 ns

(2) For a PWM Period register value of 48 counts, PWM Period = 48 x 16.67 ns = 800 ns , PWM frequency = 1/800 ns = 1.25 MHz

(3) Assumed MEP step size for the above example = 180 ps
See the device-specific data manual for typical and maximum MEP values.

### 2.3.2 Scaling Considerations

The mechanics of how to position an edge precisely in time has been demonstrated using the resources of the standard CMPA and MEP (CMPAHR) registers. In a practical application, however, it is necessary to seamlessly provide the CPU a mapping function from a per-unit (fractional) duty cycle to a final integer (non-fractional) representation that is written to the [CMPA:CMPAHR] register combination. This section describes the mapping from a per-unit duty cycle only. The method for mapping from a per-unit period is described in Section 2.3.4.

To do this, first examine the scaling or mapping steps involved. It is common in control software to express duty cycle in a per-unit or percentage basis. This has the advantage of performing all needed math calculations without concern for the final absolute duty cycle, expressed in clock counts or high time in ns. Furthermore, it makes the code more transportable across multiple converter types running different PWM frequencies.

To implement the mapping scheme, a two-step scaling procedure is required.

**Assumptions for this example:**

| | | |
|---|---|---|
| System clock , SYSCLKOUT | = | 16.67 ns (60 MHz) |
| PWM frequency | = | 1.25 MHz (1/800 ns) |
| Required PWM duty cycle, **PWMDuty** | = | 0.405 (40.5%) |
| PWM period in terms of coarse steps, **PWMperiod** (800 ns/ 16.67 ns) | = | 48 |
| Number of MEP steps per coarse step at 180 ps ( 16.67 ns/ 180 ps), **MEP_ScaleFactor** | = | 93 |
| Value to keep CMPAHR within the range of 1-255 and fractional rounding constant (default value). In the event that frac(**PWMDuty * PWMperiod**) * MEP_ScaleFactor results in a value with a decimal portion ≥ 0.5, this rounding constant will round the CMPAHR value up 1 MEP step. | = | 0.5 (0 080h in Q8 format) |

**Step 1: Percentage Integer Duty value conversion for CMPA register**

| | | |
|---|---|---|
| CMPA register value | = | int(**PWMDuty*****PWMperiod**); int means integer part |
| | = | int(0.405* 48) |
| CMPA register value | = | 19 (13h) |

**Step 2: Fractional value conversion for CMPAHR register**

| | | |
|---|---|---|
| CMPAHR register value | = | (frac(**PWMDuty*****PWMperiod**)***MEP_ScaleFactor**+0.5 ) << 8; frac means fractional part |
| | = | (frac( 19.4)* 93 + 0.5) <<8; Shift is to move the value as CMPAHR high byte |
| | = | (( 0.4 * 93 + 0.5) << 8) |
| | = | ( 37.2 + 0.5) <<8 |
| | = | 37.7*256 ; Shifting left by 8 is the same as multiplying by 256. |
| | = | 9651 |
| CMPAHR value | = | 25B3h; CMPAHR value = 25B3h, lower 8 bits will be ignored by hardware. |

**NOTE:** If the AUTOCONV bit (HRCNFG.6) is set and the MEP_ScaleFactor is in the HRMSTEP register, then CMPAHR register value = frac (PWMDuty*PWMperiod<<8). The rest of the conversion calculations are performed automatically in hardware, and the correct MEP-scaled signal edge appears on the ePWM channel output. If AUTOCONV is not set, the above calculations must be performed by software.

**NOTE:** The MEP scale factor (MEP_ScaleFactor) varies with the system clock and DSP operating conditions. TI provides an MEP scale factor optimizing (SFO) software C function, which uses the built in diagnostics in each HRPWM and returns the best scale factor for a given operating point.

The scale factor varies slowly over a limited range so the optimizing C function can be run very slowly in a background loop.

The CMPA and CMPAHR registers are configured in memory so that the 32-bit data capability of the 28x CPU can write this as a single concatenated value, i.e., [CMPA:CMPAHR]. The TBPRDM and TBPRDHRM (mirror) registers are similarly configured in memory.

The mapping scheme has been implemented in both C and assembly, as shown in Section 2.5. The actual implementation takes advantage of the 32-bit CPU architecture of the 28xx, and is somewhat different from the steps shown in Section 2.3.2.

For time critical control loops where every cycle counts, the assembly version is recommended. This is a cycle optimized function (11 SYSCLKOUT cycles ) that takes a Q15 duty value as input and writes a single [CMPA:CMPAHR] value.

### 2.3.3 Duty Cycle Range Limitation

In high resolution mode, the MEP is not active for 100% of the PWM period. It becomes operational:

- 3 SYSCLK cycles after the period starts when high-resolution period (TBPRDHR) control is not enabled.
- When high resolution period (TBPRDHR) control is enabled via the HRPCTL register:
  - In up-count mode: 3 SYSCLK cycles after the period starts until 3 SYSCLK cycles before the period ends.
  - In up-down count mode: when counting up, 3 cycles after CTR = 0 until 3 cycles before CTR = PRD, and when counting down, 3 cycles after CTR = PRD until 3 cycles before CTR = 0.

Duty cycle range limitations are illustrated in Figure 7 to Figure 10 . This limitation imposes a duty cycle limit on the MEP. For example, precision edge control is not available all the way down to 0% duty cycle. When high-resolution period control is disabled, although for the first 3 cycles, the HRPWM capabilities are not available, regular PWM duty control is still fully operational down to 0% duty. In most applications this should not be an issue as the controller regulation point is usually not designed to be close to 0% duty cycle. To better understand the useable duty cycle range, see Table 5. When high-resolution period control is enabled (HRPCTL[HRPE]=1), the duty cycle must not fall within the restricted range. Otherwise, there may be undefined behavior on the ePWMxA output.

**Figure 7. Low % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0)**

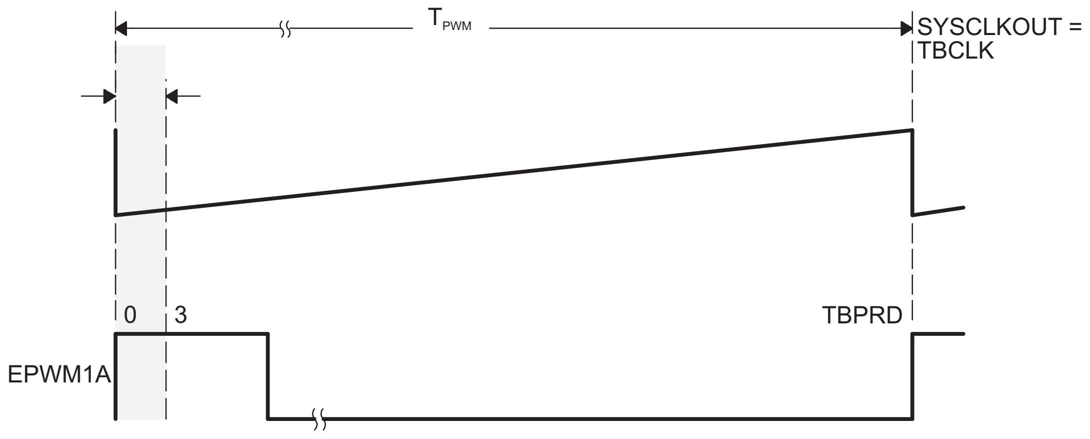


**Table 5. Duty Cycle Range Limitation for 3 SYSCLK/TBCLK Cycles**

| PWM Frequency [(1)] (kHz) | 3 Cycles Minimum Duty | 3 Cycles Maximum Duty [(2)] |
|---|---|---|
| 200 | 1.00% | 99.00% |
| 400 | 2.00% | 98.00% |
| 600 | 3.00% | 97.00% |
| 800 | 4.00% | 96.00% |
| 1000 | 5.00% | 95.00% |
| 1200 | 6.00% | 94.00% |
| 1400 | 7.00% | 93.00% |
| 1600 | 8.00% | 92.00% |
| 1800 | 9.00% | 91.00% |
| 2000 | 10.00% | 90.00% |

(1) System clock - $T_{SYSCLKOUT}$ = 16.67 ns System clock = TBCLK = 60 MHz
(2) This limitation applies only if high-resolution period (TBPRDHR) control is enabled.

If the application demands HRPWM operation in the low percent duty cycle region, then the HRPWM can be configured to operate in count-down mode with the rising edge position (REP) controlled by the MEP when high-resolution period is disabled (HRPCTL[HRPE] = 0). This is illustrated in Figure 8. In this case, low percent duty limitation is no longer an issue. However, there will be a maximum duty limitation with same percent numbers as given in Table 5.

**Figure 8. High % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0)**

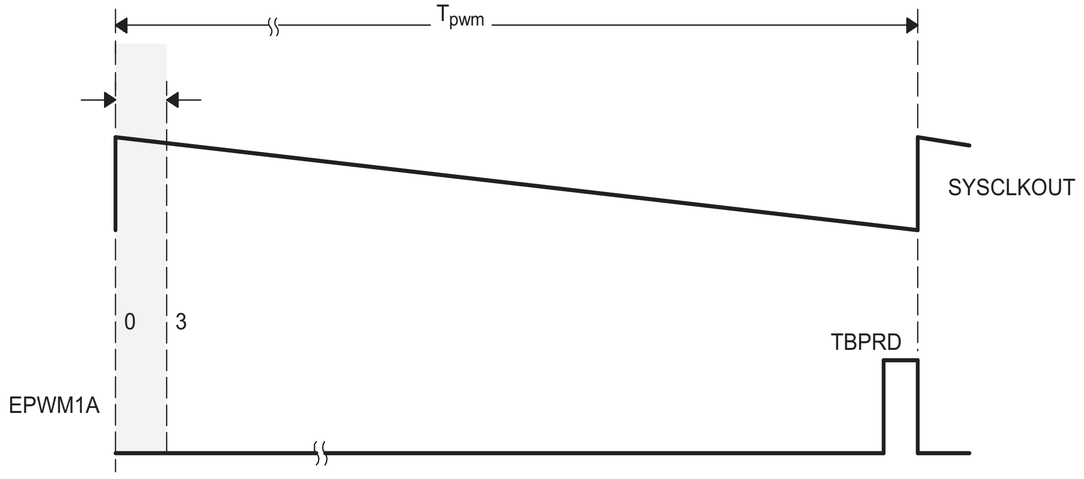


**Figure 9. Up-Count Duty Cycle Range Limitation Example (HRPCTL[HRPE]=1)**

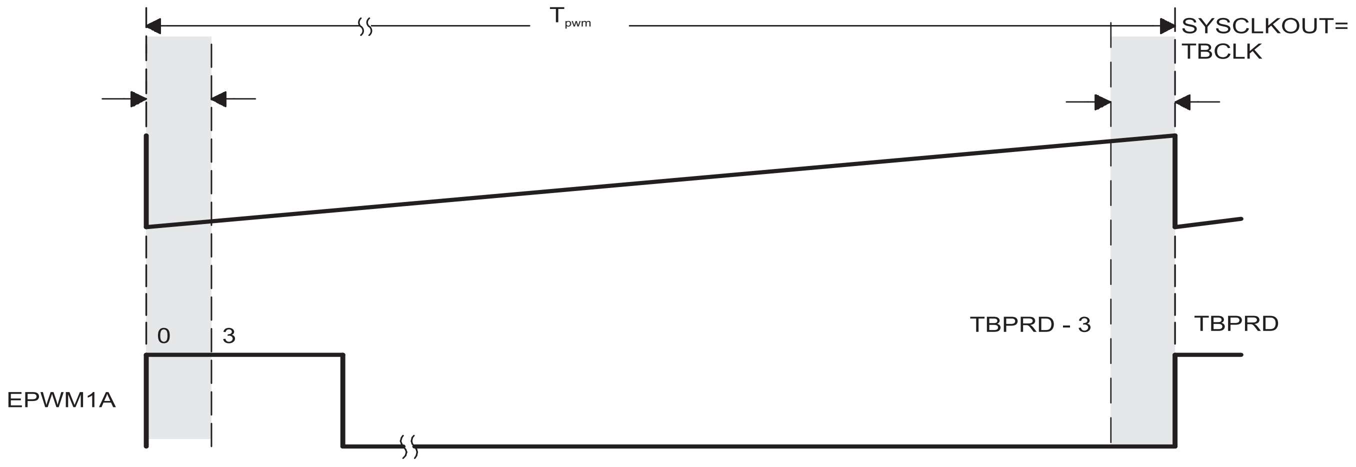
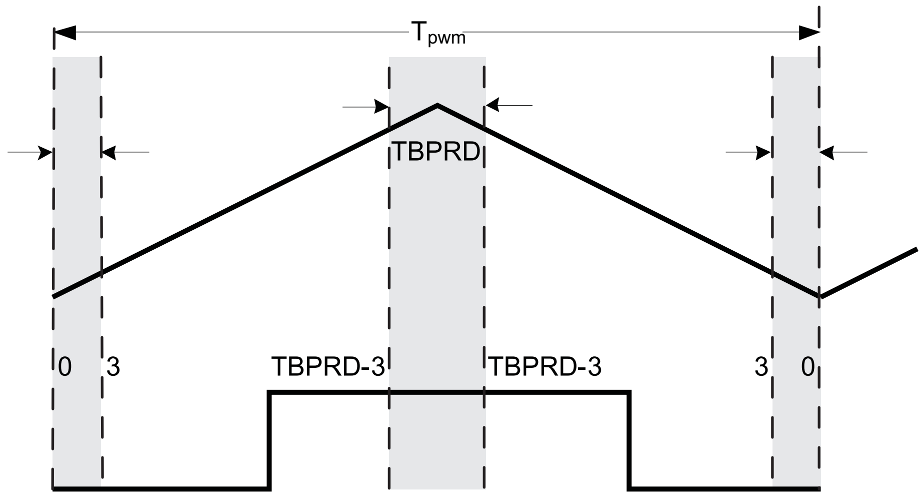


**Figure 10. Up-Down Count Duty Cycle Range Limitation Example (HRPCTL[HRPE]=1)**



**NOTE:** If the application has enabled high-resolution period control (HRPCTL[HRPE]=1), the duty cycle must not fall within the restricted range. Otherwise, there will be undefined behavior on the ePWM output.

### 2.3.4 High Resolution Period

High resolution period control using the MEP logic is supported on devices with a Type 1 ePWM module via the TBPRDHR(M) register.

**NOTE:** When high-resolution period control is enabled, the ePWMxB output will have +/- 1 TBCLK cycle jitter in up-count mode and +/- 2 TBCLK cycle jitter in up-down count mode.

The scaling procedure described for duty cycle in Section 2.3.2 applies for high-resolution period as well:

**Assumptions for this example:**

| | |
|---|---|
| System clock , SYSCLKOUT | = 16.67 ns (60 MHz) |
| Required PWM frequency | = 175 kHz (period of 342.857) |
| Number of MEP steps per coarse step at 180 ps (MEP_ScaleFactor) | = 93 (16.67 ns/180 ps) |
| Value to keep TBPRDHR within range of 1-255 and fractional rounding constant (default value) | = 0.5 (0080h in Q8 format) |

**Problem:**

In up-count mode:

If TBPRD = 342, then PWM frequency = 174.93 kHz (period = (342+1)* $T_{TBCLK}$).

TBPRD = 341, then PWM frequency = 175.44 kHz (period = (341+1)* $T_{TBCLK}$).

In up-down count mode:

If TBPRD = 172, then PWM frequency = 174.42 kHz (period = (172*2)* $T_{TBCLK}$).

If TBPRD = 171, then PWM frequency = 175.44 kHz (period = (171*2)* $T_{TBCLK}$).

**Solution:**

With 93 MEP steps per coarse step at 180 ps each:

**Step 1: Percentage Integer Period value conversion for TBPRD register**

| | |
|---|---|
| Integer period value | = 342 * $T_{TBCLK}$ |
| | = int (342.857) * $T_{TBCLK}$ |
| | = int (PWMperiod) * $T_{TBCLK}$ |
| In up-count mode: | |
| TBPRD register value | = 341 (TBPRD = period value - 1) |
| | = 0155h |
| In up-down count mode: | = 171 (TBPRD = period value / 2) |
| TBPRD register value | = 00ABh |

**Step 2: Fractional value conversion for TBPRDHR register**

| | |
|---|---|
| TBPRDHR register value | = (frac(PWMperiod) * MEP_ScaleFactor + 0.5) (shift is to move the value as TBPRDHR high byte) |
| If auto-conversion enabled and HRMSTEP = | |
| MEP_ScaleFactor value (93): | =frac (PWMperiod)<<8 |
| TBPRDHR register value | =frac (342.857)<<8 |
| | =0.857 × 256 |
| | =DB00h |

| | |
|---|---|
| The autoconversion will then automatically perform the calculation such that TBPRDHR MEP delay is scaled by hardware to: | =((TBPRDHR(15:0) >> 8) × HRMSTEP + 80h)>>8 |
| | = (00DBh × 93 + 80h) >> 8 |
| | =(500Fh) >> 8 |
| Period MEP delay | =0050h MEP Steps |

#### 2.3.4.1 High-Resolution Period Configuration

To use High Resolution Period, the ePWMx module must be initialized, following the steps in this exact order:

1. Enable ePWMx clock.
2. Disable TBCLKSYNC
3. Configure ePWMx registers - AQ, TBPRD, CC, etc.
   - ePWMx may only be configured for up-count or up-down count modes. High-resolution period is not compatible with down-count mode.
   - TBCLK must equal SYSCLKOUT
   - TBPRD and CC registers must be configured for shadow loads.
   - CMPCTL[LOADAMODE]
     – In up-count mode:CMPCTL[LOADAMODE] = 1 (load on CTR = PRD)
     – In up-down count mode: CMPCTL[LOADAMODE] = 2 (load on CTR=0 or CTR=PRD)
4. Configure HRPWM register such that:
   - HRCNFG[HRLOAD] = 2 (load on either CTR = 0 or CTR = PRD)
   - HRCNFG[AUTOCONV] = 1 (Enable auto-conversion)
   - HRCNFG[EDGMODE] = 3 (MEP control on both edges)
5. For TBPHS: TBPHSHR synchronization with high-resolution period, set both HRPCTL[TBPSHRLOADE] = 1 and TBCTL[PHSEN] = 1. In up-down count mode these bits must be set to 1 regardless of the contents of TBPHSHR.
6. Enable high-resolution period control (HRPCTL[HRPE] = 1)
7. Enable TBCLKSYNC
8. TBCTL[SWFSYNC] = 1
9. HRMSTEP must contain an accurate MEP scale factor (# of MEP steps per SYSCLKOUT coarse step) because auto-conversion is enabled. The MEP scale factor can be acquired via the SFO() function described in Appendix A.
10. To control high-resolution period, write to the TBPRDHR(M) registers.

**NOTE:** When high-resolution period mode is enabled, an EPWMxSYNC pulse will introduce +/- 1-2 cycle jitter to the PWM (+/- 1 cycle in up-count mode and +/- 2 cycle in up-down count mode). For this reason, TBCTL[SYNCOSEL] should not be set to 1 (CTR = 0 is EPWMxSYNCO source) or 2 (CTR = CMPB is EPWMxSYNCO source). Otherwise the jitter will occur on every PWM cycle with the synchronization pulse.

When TBCTL[SYNCOSEL] = 0 (EPWMxSYNCI is EPWMxSYNCO source), a software synchronization pulse whould be issued only once during high-resolution period initialization. If a software sync pulse is applied while the PWM is running, the jitter will appear on the PWM output at the time of the sync pulse.

## 2.4 Scale Factor Optimizing Software (SFO)

The micro edge positioner (MEP) logic is capable of placing an edge in one of 255 discrete time steps. As previously mentioned, the size of these steps is on the order of 150 ps (see device-specific data sheet for typical MEP step size on your device). The MEP step size varies based on worst-case process parameters, operating temperature, and voltage. MEP step size increases with decreasing voltage and increasing temperature and decreases with increasing voltage and decreasing temperature. Applications that use the HRPWM feature should use the TI-supplied MEP scale factor optimizer (SFO) software function. The SFO function helps to dynamically determine the number of MEP steps per SYSCLKOUT period while the HRPWM is in operation.

To utilize the MEP capabilities effectively during the Q15 duty (or period) to [CMPA:CMPAHR] or [TBPRD(M):TBPRDHR(M)] mapping function (see Section 2.3.2), the correct value for the MEP scaling factor (MEP_ScaleFactor) needs to be known by the software. To accomplish this, the HRPWM module has built in self-check and diagnostics capabilities that can be used to determine the optimum MEP_ScaleFactor value for any operating condition. TI provides a C-callable library containing one SFO function that utilizes this hardware and determines the optimum MEP_ScaleFactor. As such, MEP Control and Diagnostics registers are reserved for TI use.

A detailed description of the SFO library - SFO_TI_Build_V6.lib software can be found in Appendix A.

## 2.5 HRPWM Examples Using Optimized Assembly Code

The best way to understand how to use the HRPWM capabilities is through two real examples:

1. Simple buck converter using asymmetrical PWM (i.e. count-up) with active high polarity.
2. DAC function using simple R+C reconstruction filter.

The following examples all have Initialization/configuration code written in C. To make these easier to understand, the #defines shown below are used. Note, #defines introduced in the device-specific *Pulse Width Modulator (ePWM) Module Reference Guide* are also used.

Example 1 This example assumes MEP step size of 150 ps and does not use the SFO library.

### *Example 1. #Defines for HRPWM Header Files*

```
// HRPWM (High Resolution PWM) //
================================
// HRCNFG
#define HR_Disable 0x0
#define HR_REP 0x1          // Rising Edge position
#define HR_FEP 0x2          // Falling Edge position
#define HR_BEP 0x3          // Both Edge position #define HR_CMP 0x0 // CMPAHR controlled
#define HR_PHS 0x1          // TBPHSHR controlled #define HR_CTR_ZERO 0x0 // CTR = Zero event
#define HR_CTR_PRD 0x1      // CTR = Period event
#define HR_CTR_ZERO_PRD 0x2 // CTR = ZERO or Period event
#define HR_NORM_B  0x0      // Normal ePWMxB output
#define HR_INVERT_B 0x1     // ePWMxB is inverted ePWMxA output
```

### 2.5.1 Implementing a Simple Buck Converter

In this example, the PWM requirements for SYSCLKOUT = 60 MHz are:

- PWM frequency = 600 kHz (i.e., TBPRD = 100 )
- PWM mode = asymmetrical, up-count
- Resolution = 12.7 bits (with a MEP step size of 150 ps)

Figure 11 and Figure 12 show the required PWM waveform. As explained previously, configuration for the ePWM1 module is almost identical to the normal case except that the appropriate MEP options need to be enabled/selected.

**Figure 11. Simple Buck Controlled Converter Using a Single PWM**

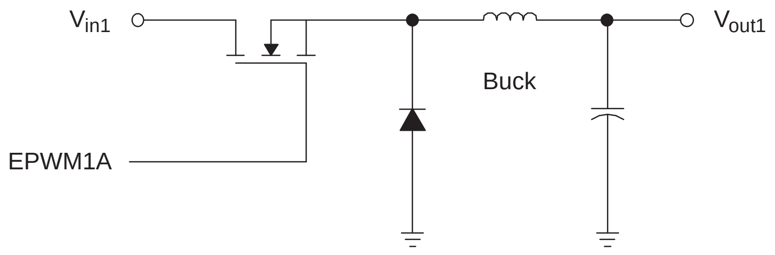
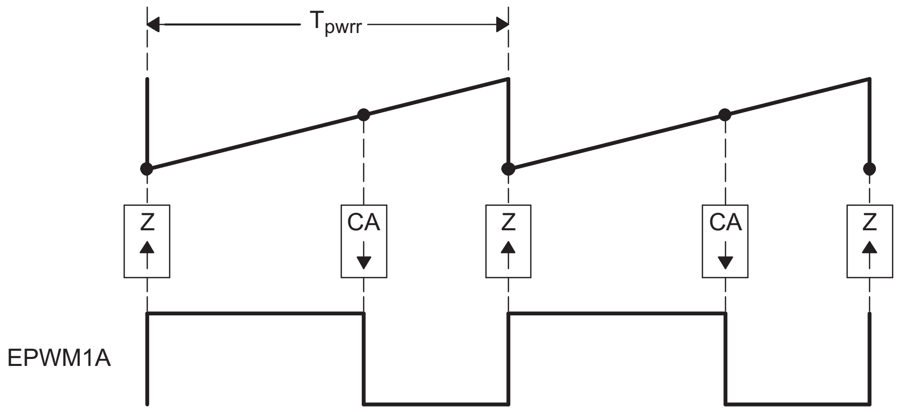


**Figure 12. PWM Waveform Generated for Simple Buck Controlled Converter**



The example code shown consists of two main parts:

- Initialization code (executed once)
- Run time code (typically executed within an ISR)

Example 2 shows the Initialization code. The first part is configured for conventional PWM. The second part sets up the HRPWM resources.

This example assumes MEP step size of 150 ps and does not use the SFO library.

### Example 2. HRPWM Buck Converter Initialization Code

```
void HrBuckDrvCnf(void)
 {
// Config for conventional PWM first
EPwm1Regs.TBCTL.bit.PRDLD = TB_IMMEDIATE;          // set Immediate load
EPwm1Regs.TBPRD = 100;                             // Period set for 600 kHz PWM
hrbuck_period = 200;                               // Used for Q15 to Q0 scaling
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;            // EPWM1 is the Master
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
// Note: ChB is initialized here only for comparison purposes, it is not required

EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;      // optional
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;        // optional

EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;
EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.ZRO = AQ_SET;                 // optional
EPwm1Regs.AQCTLB.bit.CBU = AQ_CLEAR;               // optional
// Now configure the HRPWM resources
EALLOW;                                            // Note these registers are protected
                                                   // and act only on ChA
EPwm1Regs.HRCNFG.all = 0x0;                        // clear all bits first
EPwm1Regs.HRCNFG.bit.EDGMODE = HR_FEP;             // Control Falling Edge Position
EPwm1Regs.HRCNFG.bit.CTLMODE = HR_CMP;             // CMPAHR controls the MEP
EPwm1Regs.HRCNFG.bit.HRLOAD = HR_CTR_ZERO;         // Shadow load on CTR=Zero
EDIS;
MEP_ScaleFactor = 111*256;                         // Start with typical Scale Factor
                                                   // value for 60 MHz
                                                   // Note: Use SFO functions to update
                                                   // MEP_ScaleFactor dynamically
}
```

Example 3 shows an assembly example of run-time code for the HRPWM buck converter.

### Example 3. HRPWM Buck Converter Run-Time Code

```
EPWM1_BASE .set 0x6800
CMPAHR1 .set EPWM1_BASE+0x8
;==============================================
HRBUCK_DRV; (can execute within an ISR or loop)
;==============================================
     MOVW DP, #_HRBUCK_In
     MOVL XAR2,@_HRBUCK_In      ; Pointer to Input Q15 Duty (XAR2)
     MOVL XAR3,#CMPAHR1         ; Pointer to HRPWM CMPA reg (XAR3)
; Output for EPWM1A (HRPWM)
     MOV T,*XAR2                ; T <= Duty
     MPYU ACC,T,@_hrbuck_period ; Q15 to Q0 scaling based on Period
```

***Example 3. HRPWM Buck Converter Run-Time Code (continued)***

```
      MOV T,@_MEP_ScaleFactor    ; MEP scale factor (from optimizer s/w)
      MPYU P,T,@AL               ; P <= T * AL, Optimizer scaling
      MOVH @AL,P                 ; AL <= P, move result back to ACC
      ADD ACC, #0x080            ; MEP range and rounding adjustment
      MOVL *XAR3,ACC             ; CMPA:CMPAHR(31:8) <= ACC
; Output for EPWM1B (Regular Res) Optional - for comparison purpose only
      MOV *+XAR3[2],AH           ; Store ACCH to regular CMPB
```

### 2.5.2 Implementing a DAC function Using an R+C Reconstruction Filter

In this example, the PWM requirements are:

- PWM frequency = 400 kHz (i.e., TBPRD = 150)
- PWM mode = Asymmetrical, Up-count
- Resolution = 14 bits ( MEP step size = 150 ps)

Figure 13 and Figure 14 show the DAC function and the required PWM waveform. As explained previously, configuration for the ePWM1 module is almost identical to the normal case except that the appropriate MEP options need to be enabled/selected.

**Figure 13. Simple Reconstruction Filter for a PWM Based DAC**

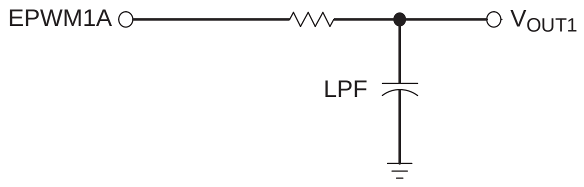
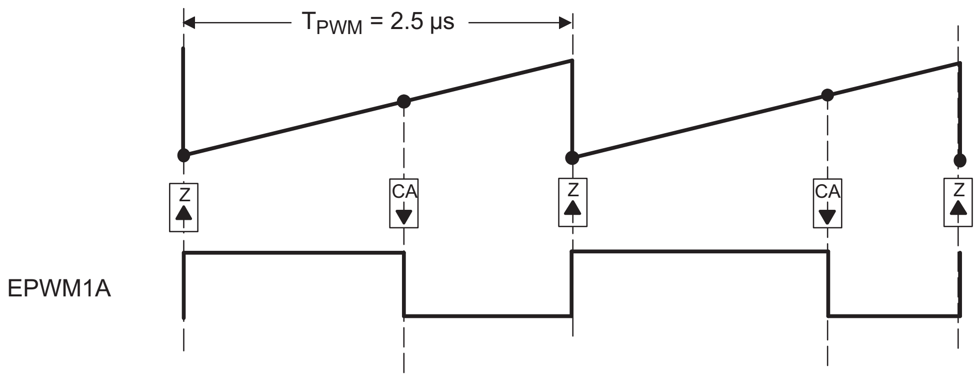


**Figure 14. PWM Waveform Generated for the PWM DAC Function**



The example code shown consists of two main parts:

- Initialization code (executed once)
- Run time code (typically executed within an ISR)

This example assumes a typical MEP_ScaleFactor and does not use the SFO library.

Example 4 shows the Initialization code. The first part is configured for conventional PWM. The second part sets up the HRPWM resources.

### *Example 4. PWM DAC Function Initialization Code*

```
void HrPwmDacDrvCnf(void)
{
// Config for conventional PWM first
EPwm1Regs.TBCTL.bit.PRDLD = TB_IMMEDIATE;      // Set Immediate load
EPwm1Regs.TBPRD = 150;                         // Period set for 400 kHz PWM
hrDAC_period = 150;                            // Used for Q15 to Q0 scaling
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;        // EPWM1 is the Master
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
// Note: ChB is initialized here only for comparison purposes, it is not required

EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;  // optional
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;    // optional

EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;
EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.ZRO = AQ_SET;             // optional
EPwm1Regs.AQCTLB.bit.CBU = AQ_CLEAR;           // optional
// Now configure the HRPWM resources
EALLOW;                                        // Note these registers are protected
                                               // and act only on ChA.
EPwm1Regs.HRCNFG.all = 0x0; // Clear all bits first
EPwm1Regs.HRCNFG.bit.EDGMODE = HR_FEP;         // Control falling edge position
EPwm1Regs.HRCNFG.bit.CTLMODE = HR_CMP;         // CMPAHR controls the MEP.
EPwm1Regs.HRCNFG.bit.HRLOAD = HR_CTR_ZERO;     // Shadow load on CTR=Zero.
EDIS;
MEP_ScaleFactor = 111*256;                     // Start with typical Scale Factor
                                               // value for 60 MHz.
                                               // Use SFO functions to update MEP_ScaleFactor
                                               // dynamically.
}
```

Example 5 shows an assembly example of run-time code that can execute in a high-speed ISR loop.

### *Example 5. PWM DAC Function Run-Time Code*

```
EPWM1_BASE .set 0x6800
CMPAHR1 .set EPWM1_BASE+0x8
;==================================================
HRPWM_DAC_DRV; (can execute within an ISR or loop)
;==================================================
      MOVW DP, #_HRDAC_In
      MOVL XAR2,@_HRDAC_In            ; Pointer to input Q15 duty (XAR2)
      MOVL XAR3,#CMPAHR1              ; Pointer to HRPWM CMPA reg (XAR3)

; Output for EPWM1A (HRPWM
      MOV T,*XAR2                      ; T <= duty
      MPY ACC,T,@_hrDAC_period         ; Q15 to Q0 scaling based on period
      ADD ACC,@_HrDAC_period<<15       ; Offset for bipolar operation
      MOV T,@_MEP_ScaleFactor                   ; MEP scale factor (from optimizer s/w)
      MPYU P,T,@AL                     ; P <= T * AL, optimizer scaling
      MOVH @AL,P                       ; AL <= P, move result back to ACC
      ADD ACC, #0x080                  ; MEP range and rounding adjustment
      MOVL *XAR3,ACC                   ; CMPA:CMPAHR(31:8) <= ACC

; Output for EPWM1B (Regular Res) Optional - for comparison purpose only
      MOV *+XAR3[2],AH                 ; Store ACCH to regular CMPB
```

## 3 HRPWM Register Descriptions

This section describes the applicable HRPWM registers

### 3.1 Register Summary

A summary of the registers required for the HRPWM is shown in the table below.

**Table 6. Register Descriptions**

| Name | Offset | Size(x16)/Shadow | Description |
|---|---|---|---|
| **Time Base Registers** | | | |
| TBCTL | 0x0000 | 1/0 | Time Base Control Register |
| TBSTS | 0x0001 | 1/0 | Time Base Status Register |
| **TBPHSHR** | 0x0002 | 1/0 | Time Base Phase High Resolution Register |
| TBPHS | 0x0003 | 1/0 | Time Base Phase Register |
| TBCNT | 0x0004 | 1/0 | Time Base Counter Register |
| TBPRD | 0x0005 | 1/1 | Time Base Period Register Set |
| **TBPRDHR** | 0x0006 | 1/1 | Time Base Period High Resolution Register Set |
| **Compare Registers** | | | |
| CMPCTL | 0x0007 | 1/0 | Counter Compare Control Register |
| **CMPAHR** | 0x0008 | 1/1 | Counter Compare A High Resolution Register Set |
| CMPA | 0x0009 | 1/1 | Counter Compare A Register Set |
| CMPB | 0x000A | 1/1 | Counter Compare B Register Set |
| **HRPWM Registers** | | | |
| **HRCNFG** | 0x0020 | 1/0 | HRPWM Configuration Register |
| **HRPWR** | 0x0021 | 1/0 | HRPWM Power Register |
| **HRMSTEP** | 0x0026 | 1/0 | HRPWM MEP Step Register |
| **High Resolution Period & Mirror Registers** | | | |
| HRPCTL | 0x0028 | 1/0 | High Resolution Period Control Register |
| **TBPRDHRM** | 0x002A | 1/1 | Time Base Period High Resolution Mirror Register Set |
| TBPRDM | 0x002B | 1/1 | Time Base Period Mirror Register Set |
| **CMPAHRM** | 0x002C | 1/1 | Counter Compare A High Resolution Mirror Register Set |
| CMPAM | 0x002D | 1/1 | Counter Compare A Mirror Register Set |

## 3.2 Registers and Field Descriptions

**Figure 15. HRPWM Configuration Register (HRCNFG)**

| 15–8 |
|---|
| Reserved |
| R-0 |

| 7 | 6 | 5 | 4–3 | 2 | 1–0 |
|---|---|---|---|---|---|
| SWAPAB | AUTOCONV | SELOUTB | HRLOAD | CTLMODE | EDGMODE |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 7. HRPWM Configuration Register (HRCNFG) Field Descriptions**

| Bit | Field | Value | Description [(1)] |
|---|---|---|---|
| 15-8 | Reserved | | Reserved |
| 7 | SWAPAB | | Swap ePWM A & B Output Signals |
| | | | This bit enables the swapping of the A & B signal outputs. The selection is as follows: |
| | | 0 | ePWMxA and ePWMxB outputs are unchanged. |
| | | 1 | ePWMxA signal appears on ePWMxB output and ePWMxB signal appears on ePWMxA output. |
| 6 | AUTOCONV | | Auto Convert Delay Line Value |
| | | | Selects whether the fractional duty cycle/period/phase in the CMPAHR/TBPRDHR/TBPHSHR register is automatically scaled by the MEP scale factor in the HRMSTEP register or manually scaled by calculations in application software. The SFO library function automatically updates the HRMSTEP register with the appropriate MEP scale factor. |
| | | 0 | Automatic HRMSTEP scaling is disabled. |
| | | 1 | Automatic HRMSTEP scaling is enabled. |
| | | | If application software is manually scaling the fractional duty cycle, or phase (i.e. software sets CMPAHR = (fraction(PWMduty * PWMperiod) * MEP Scale Factor)<<8 + 0x080 for duty cycle), then this mode must be disabled. |
| 5 | SELOUTB | | EPWMxB Output Select Bit |
| | | | This bit selects which signal is output on the ePWMxB channel output. |
| | | 0 | ePWMxB output is normal. |
| | | 1 | ePWMxB output is inverted version of ePWMxA signal. |
| 4-3 | HRLOAD | | Shadow Mode Bit |
| | | | Selects the time event that loads the CMPAHR shadow value into the active register. |
| | | 00 | Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) |
| | | 01 | Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) |
| | | 10 | Load on either CTR = Zero or CTR = PRD |
| | | 11 | Reserved |
| 2 | CTLMODE | | Control Mode Bits |
| | | | Selects the register (CMP/TBPRD or TBPHS) that controls the MEP: |
| | | 0 | CMPAHR(8) or TBPRDHR(8) Register controls the edge position (i.e., this is duty or period control mode). (Default on Reset) |
| | | 1 | TBPHSHR(8) Register controls the edge position (i.e., this is phase control mode). |
| 1-0 | EDGMODE | | Edge Mode Bits |
| | | | Selects the edge of the PWM that is controlled by the micro-edge position (MEP) logic: |
| | | 00 | HRPWM capability is disabled (default on reset) |
| | | 01 | MEP control of rising edge (CMPAHR) |
| | | 10 | MEP control of falling edge (CMPAHR) |
| | | 11 | MEP control of both edges (TBPHSHR or TBPRDHR) |

[(1)] This register is EALLOW protected.

**Figure 16. Counter Compare A High Resolution Register (CMPAHR)**

| | |
|---|---|
| 15 — 8 | 7 — 0 |
| CMPAHR | Reserved |
| R/W-0 | R-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 8. Counter Compare A High Resolution Register (CMPAHR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15-8 | CMPAHR | 00-FEh | Compare A High Resolution register bits for MEP step control. These 8-bits contain the high-resolution portion (least significant 8-bits) of the counter-compare A value. CMPA:CMPAHR can be accessed in a single 32-bit read/write. Shadowing is enabled and disabled by the CMPCTL[SHDWAMODE] bit. |
| 7-0 | Reserved | 00-FFh | Any writes to these bit(s) must always have a value of 0. |

**Figure 17. TB Phase High Resolution Register (TBPHSHR)**

| | |
|---|---|
| 15 — 8 | 7 — 0 |
| TBPHSH | Reserved |
| R/W-0 | R-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 9. TB Phase High Resolution Register (TBPHSHR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15-8 | TBPHSH | 00-FEh | Time base phase high resolution bits |
| 7-0 | Reserved | 00-FFh | Any writes to these bit(s) must always have a value of 0. |

**Figure 18. Time Base Period High Resolution Register**

| |
|---|
| 15 — 8 |
| TBPRDHR |
| R/W-0 |
| 7 — 0 |
| Reserved |
| R-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 10. Time Base Period High-Resolution Register (TBPRDHR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15-8 | PRDHR | 00-FFh | Period High Resolution Bits<br>These 8-bits contain the high-resolution portion of the period value.<br>The TBPRDHR register is not affected by the TBCTL[PRDLD] bit. Reads from this register always reflect the shadow register. Likewise writes are also to the shadow register. The TBPRDHR register is only used when the high resolution period feature is enabled.<br>This register is only available with ePWM modules which support high-resolution period control. |
| 7-0 | Reserved | | Reserved for TI Test |

**Figure 19. Compare A High Resolution Mirror Register**

| 15 … 8 |
|---|
| CMPAHR |
| R/W-0 |

| 7 … 0 |
|---|
| Reserved |
| R-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 11. Compare A High-Resolution Mirror Register (CMPAHRM) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15-8 | CMPAHR | 00-FFh | Compare A High Resolution Bits<br>Writes to both the CMPAHR and CMPAHRM locations access the high-resolution (least significant 8-bit) portion of the Counter Compare A value. The only difference is that unlike CMPAHR, reads from the mirror register, CMPAHRM, are indeterminate (reserved for TI Test).<br>By default writes to this register are shadowed. Shadowing is enabled and disabled by the CMPCTL[SHDWAMODE] bit as described for the CMPAM register. |
| 7-0 | Reserved | 00-FFh | Reserved for TI Test |

**Figure 20. Time-Base Period High Resolution Mirror Register**

| 15 … 8 |
|---|
| TBPRDHR |
| R/W-0 |

| 7 … 0 |
|---|
| Reserved |
| R-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 12. Time-Base Period High-Resolution Mirror Register (TBPRDHRM) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15-8 | TBPRDHR | 00-FFh | Period High Resolution Bits<br>These 8-bits contain the high-resolution portion of the period value.<br>TBPRD provides backwards compatibility with earlier ePWM modules. The mirror registers (TBPRDM and TBPRDHRM) allow for 32-bit writes to TBPRDHR in one access. Due to the odd-numbered memory address location of the TBPRD legacy register, a 32-bit write is not possible with TBPRD and TBPRDHR.<br>The TBPRDHRM register is not affected by the TBCTL[PRDLD] bit<br>Writes to both the TBPRDHR and TBPRDM locations access the high-resolution (least significant 8-bit) portion of the Time Base Period value. The only difference is that unlike TBPRDHR, reads from the mirror registerTBPRDHRM, are indeterminate (reserved for TI Test).<br>The TBPRDHRM register is available with ePWM modules which support high-respolution period control and is used only when the high resolution period feature is enabled. |
| 7-0 | Reserved | | Reserved |

**Figure 21. High Resolution Period Control Register (HRPCTL)**

| 15 … 8 |
|---|
| Reserved |
| R-0 |

| 7 … 3 | 2 | 1 | 0 |
|---|---|---|---|
| Reserved | TBPHSHR LOADE | PWMSYNCSEL | HRPE |
| R-0 | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

## Table 13. High Resolution Period Control Register (HRPCTL) Field Descriptions

| Bit | Field | Value | Description [(1)] [(2)] |
|---|---|---|---|
| 15-3 | Reserved | | Reserved |
| 2 | TBPHSHRLOADE | | TBPHSHR Load Enable |
| | | | This bit allows you to synchronize ePWM modules with a high-resolution phase on a SYNCIN, TBCTL[SWFSYNC] or digital compare event. This allows for multiple ePWM modules operating at the same frequency to be phase aligned with high-resolution. |
| | | 0 | Disables synchronization of high-resolution phase on a SYNCIN, TBCTL[SWFSYNC] or digital compare event: |
| | | 1 | Synchronize the high-resolution phase on a SYNCIN, TBCTL[SWFSYNC] or digital comparator synchronization event. The phase is synchronized using the contents of the high-resolution phase TBPHSHR register. |
| | | | The TBCTL[PHSEN] bit which enables the loading of the TBCTR register with TBPHS register value on a SYNCIN or TBCTL[SWFSYNC] event works independently. However, users need to enable this bit also if they want to control phase in conjunction with the high-resolution period feature. |
| | | | This bit and the TBCTL[PHSEN] bit must be set to 1 when high-resolution period is enabled for up-down count mode even if TBPHSHR = 0x0000. This bit does not need to be set when only high-resolution duty is enabled. |
| 1 | PWMSYNCSEL | | PWMSYNC Source Select Bit |
| | | | This bit selects the source for the PWMSYNC signal. |
| | | | The PWMSYNC signal is used by external modules (such as COMP+DAC) for synchronizing timing to the selected ePWM module. |
| | | | **Note:** This bit is not used for high-resolution period control. |
| | | 0 | PWMSYNC is generated by TBCTR = PRD pulse. |
| | | 1 | PWMSYNC is generated by TBCTR = 0 pulse. |
| 0 | HRPE | | High Resolution Period Enable Bit |
| | | 0 | High resolution period feature disabled. In this mode the ePWM behaves as a Type 0 ePWM. |
| | | 1 | High resolution period enabled. In this mode the HRPWM module can control high-resolution of both the duty and frequency. |
| | | | When high-resolution period is enabled, TBCTL[CTRMODE] = 0,1 (down-count mode) is not supported. |

(1) This register is EALLOW protected.
(2) This register is used with Type 1 ePWM modules (support high-resolution period) only.

## Figure 22. High Resolution Micro Step Register (HRMSTEP) (EALLOW protected):

| 15–8 | 7–0 |
|---|---|
| Reserved | HRMSTEP |
| R-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

## Table 14. High Resolution Micro Step Register (HRMSTEP) Field Descriptions

| Bit | Field | Value | Description |
|---|---|---|---|
| 15:8 | Reserved | | Reserved |
| 7:0 | HRMSTEP | 00-FFh | High Resolution MEP Step |
| | | | When auto-conversion is enabled (HRCNFG[AUTOCONV] = 1), This 8-bit field contains the MEP_ScaleFactor (number of MEP steps per coarse steps) used by the hardware to automatically convert the value in the CMPAHR, TBPHSHR, or TBPRDHR register to a scaled micro-edge delay on the high-resolution ePWM output. |
| | | | The value in this register is written by the SFO calibration software at the end of each calibration run. |

**Figure 23. High Resolution Power Register (HRPWR) (EALLOW protected)**

| 15–10 | 9–6 | 5–0 |
|---|---|---|
| Reserved | MEPOFF | Reserved |
| R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 15. High Resolution Power Register (HRPWR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15-10 | Reserved | | Reserved for TI Test |
| 9-6 | MEPOFF | 0-Fh | MEP Calibration OFF bits<br>When not using MEP calibration, setting these bits to all 1's disables MEP calibration logic in the HRPWM and reduces power consumption. |
| 5-0 | Reserved | | Reserved for TI Test |

## Appendix A SFO Library Software - SFO_TI_Build_V6.lib

The following table lists several features of the SFO_TI_Build_V6.lib library.

Table 16. SFO Library Features

| | SYSCLK Freq | SFO_TI_Build_V6.lib | Unit |
|---|---|---|---|
| Max. HRPWM channels supported | - | 8 | channels |
| Total static variable memory size | - | 11 | words |
| Completion-checking? | - | Yes | - |
| Typical time required for SFO() to update MEP_ScaleFactor if called repetitively without interrupts | 60 MHz | 2.23 | milliseconds |
| | 40 MHz | 3.36 | milliseconds |

A functional description of the SFO library routine, SFO(), is found below.

### A.1 *Scale Factor Optimizer Function - int SFO()*

This routine drives the micro-edge positioner (MEP) calibration module to run SFO diagnostics and determine the appropriate MEP scale factor (number of MEP steps per coarse SYSCLKOUT step) for a device at any given time.

If SYSCLKOUT = TBCLK = 60 MHz and assuming the MEP step size is 150 ps, the typical scale factor value at 60 MHz = 111 MEP steps per TBCLK unit (16.67 ns)

The function returns a MEP scale factor value:

MEP_ScaleFactor = Number of MEP steps/SYSCLKOUT.

**Constraints when using this function:**

- SFO() can be used with a minimum SYSCLKOUT = TBCLK = 50 MHz. MEP diagnostics logic uses SYSCLKOUT and not TBCLK, so the SYSCLKOUT restriction is an important constraint. Below 50 MHz, with device process variation, the MEP step size may decrease under cold temperature and high core voltage conditions to such a point, that 255 MEP steps will not span an entire SYSCLKOUT cycle.
- At any time, SFO() can be called to run SFO diagnostics on the MEP calibration module

**Usage:**

- SFO() can be called at any time in the background while the ePWM channels are running in HRPWM mode. The scale factor result obtained in MEP_ScaleFactor can be applied to all ePWM channels running in HRPWM mode because the function makes use of the diagnostics logic in the MEP calibration module (which runs independently of ePWM channels).
- This routine returns a 1 when calibration is finished, and a new scale factor has been calculated or a 0 if calibration is still running. The routine returns a 2 if there is an error, and the MEP_ScaleFactor is greater than the maximum 255 fine steps per coarse SYSCLKOUT cycle. In this case. the HRMSTEP register will maintain the last MEP scale factor value less than 256 for auto conversion.
- All ePWM modules operating in HRPWM incur only a 3-SYSCLKOUT cycle minimum duty cycle limitation when high-resolution period control is not used. If high-resolution period control is enabled, there is an additional duty cycle limitation 3-SYSCLKOUT cycles before the end of the PWM period (see Section 2.3.3).
- In SFO_TI_Build_V6b.lib, the SFO() function also updates the HRMSTEP register with the scale factor result. If the HRCNFG[AUTOCONV] bit is set, the application software is responsible only for setting CMPAHR = fraction(PWMduty*PWMperiod)<<8 or TBPRDHR = fraction (PWMperiod) while running SFO() in the background. The MEP Calibration Module will then use the values in the HRMSTEP and CMPAHR/TBPRDHR register to automatically calculate the appropriate number of MEP steps represented by the fractional duty cycle or period and move the high-resolution ePWM signal edge accordingly. In SFO_TI_Build_V6.lib, the SFO() function does not automatically update the HRMSTEP register. Therefore, after the SFO function completes, the application software must write MEP_ScaleFactor to the HRMSTEP register (EALLOW-protected).

- If the HRCNFG[AUTOCONV] bit is clear, the HRMSTEP register is ignored. The application software will need to perform the necessary calculations manually so that:
  - CMPAHR = (fraction(PWMduty * PWMperiod) * MEP Scale Factor)<<8 + 0x080.
  - Similar behavior applies for TBPHSHR. Auto-conversion must be enabled when using TBPRDHR.

The routine can be run as a background tasks in a slow loop requiring negligible CPU cycles. The repetition rate at which an SFO function needs to be executed depends on the application's operating environment. As with all digital CMOS devices temperature and supply voltage variations have an effect on MEP operation. However, in most applications these parameters vary slowly and therefore it is often sufficient to execute the SFO function once every 5 to 10 seconds or so. If more rapid variations are expected, then execution may have to be performed more frequently to match the application. Note, there is no high limit restriction on the SFO function repetition rate, hence it can execute as quickly as the background loop is capable.

While using the HRPWM feature, HRPWM logic will not be active for the first 3 SYSCLKOUT cycles of the PWM period (and the last 3 SYSCLKOUT cycles of the PWM period if TBPRDHR is used). While running the application in this configuration, if high-resolution period control is disabled (HRPCTL[HRPE=0]) and the CMPA register value is less than 3 cycles, then its CMPAHR register must be cleared to zero. If high-resolution period control is enabled (HRPCTL[HRPE=1]), the CMPA register value must not fall below 3 or above TBPRD-3.This would avoid any unexpected transitions on the PWM signal.

## A.2 Software Usage

The software library function SFO(), calculates the MEP scale factor for the HRPWM-supported ePWM modules. The scale factor is an integer value in the range 1-255, and represents the number of micro step edge positions available for a system clock period. The scale factor value is returned in an integer variable called MEP_ScaleFactor. For example, see Table 17.

**Table 17. Factor Values**

| Software Function call | Functional Description | Updated Variables |
|---|---|---|
| SFO() | Returns MEP scale factor in MEP_ScaleFactor<br>Returns MEP scale factor in the HRMSTEP register in SFO_TI_Build_V6bt.lib | MEP_ScaleFactor & HRMSTEP register. |

To use the HRPWM feature of the ePWMs it is recommended that the SFO function be used as described here.

**Step 1. Add "Include" Files**

The SFO_V6.h file needs to be included as follows. This include file is mandatory while using the SFO library function. For the TMS320F2802x devices, the *C2802x C/C++ Header Files and Peripheral Examples*. DSP2802x_Device.h and DSP2802x_Epwm_defines.h are necessary. For other device families, the device_specific equivalent files in the header files and peripheral examples software packages for those devices should be used. These include files are optional if customized header files are used in the end applications.

***Example 6. A Sample of How to Add "Include" Files***

```
#include "DSP2802x_Device.h"       // DSP2802x Headerfile
#include "DSP2802x_EPwm_defines.h" // init defines
#include "SFO_V6.h"                // SFO lib functions (needed for HRPWM)
```

**Step 2. Element Declaration**

Declare an integer variable for the scale factor value as shown below. The first &EPwm1Regs is a dummy placeholder only, and can be disregarded.

***Example 7. Declaring an Element***

```
int MEP_ScaleFactor = 0;   //scale factor value
volatile struct EPWM_REGS *ePWM[] = {0, &EPwm1Regs, &EPwm2Regs, &EPwm3Regs,
&EPwm4Regs};
```

**Step 3. MEP_ScaleFactor Initialization**

The SFO() function does not require a starting scale factor value in MEP_ScaleFactor. Prior to using the MEP_ScaleFactor variable in application code, SFO() should be called to drive the MEP calibration module to calculate an MEP_ScaleFactor value.

As part of the one-time initialization code prior to using MEP_ScaleFactor, include the following:

***Example 8. Initializing With a Scale Factor Value***

```
MEP_ScaleFactor initialized using function SFO ()
while (SFO() == 0) {} // MEP_ScaleFactor calculated by MEP Cal Module
```

**Step 4. Application Code**

While the application is running, fluctuations in both device temperature and supply voltage may be expected. To be sure that optimal Scale Factors are used for each ePWM module, the SFO function should be re-run periodically as part of a slower back-ground loop. Some examples of this are shown here.

**NOTE:** See the HRPWM_SFO example in the device-specific C/C++ header files and peripheral examples available from the TI website.

***Example 9. SFO Function Calls***

```
main ()
 {
 int status;
    // User code
    // ePWM1, 2, 3, 4 are running in HRPWM mode
    // The status variable returns 1 once a new MEP_ScaleFactor has been
    // calculated by the MEP Calibration Module running SFO
    // diagnostics.

status = SFO();
if(status==2) {ESTOP0;}   // The function returns a 2 if MEP_ScaleFactor is greater
                          // than the maximum 255 allowed (error condition)
 }
```

## *A.3 SFO Library Version Software Differences*

There are two different versions of the SFO library - SFO_TI_Build_V6.lib, and SFO_TI_Build_V6b.lib. SFO_TI_Build_V6.lib does not update the HRMSTEP register with the value in MEP_ScaleFactor, while SFO_TI_Build_V6b.lib updates the register. Therefore, if using SFO_TI_Build_V6.lib and auto-conversion is enabled, the application should write MEP_Scalefactor in the HRMSTEP register as shown below.

### Example 10. Manually Updating the HRMSTEP Register if using SFO_TI_Build_V6b.lib

```
main ()
 {
     int status;

     status = SFO_INCOMPLETE;

     while (status==SFO_INCOMPLETE) {
            status = SFO();
 }

     if(status!=SFO_ERROR) { // IF SFO() is complete with no errors
        EALLOW;
        EPwm1Regs.HRMSTEP=MEP_ScaleFactor;
        EDIS;

 }
```

## Appendix B Revision History

This document was revised and lists only revisions made in this most recent version. The scope of the revisions was limited to technical changes as shown in Table 18.

**Table 18. Technical Changes in the Current Revision**

| Location | Additions, Deletions, Modifications |
|---|---|
| **Global** | **Replaced Map with Mep** |
| Global | Replaced MEP_SF to MEP_ScaleFactor |
| Section 1 | Revised "Table values assume a MEP step size of 180 ps" to " These values assume a 100 MHz SYSCLK frequency and a MEP step size of 180 ps." |
| Figure 2 | Revised 16-bit CMPAHR register value = ... |
|  | Revised Note: For MEP range and rounding adjustment to include "(0x0180 in Q8 format)" |
| Figure 4 | Changed TBCTL[CNTLDE] to TBCTL[PHSEN] |
| Section 2.3.1 | Changed 180 ns to 180 ps |
| Section 2.3.2 | Revised the **Assumptions for this example** section |