TMS320x280x High-Resolution Pulse Width Modulator (HRPWM)

Reference Guide

Literature Number: SPRU924A
April 2005–Revised January 2006

Texas Instruments

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).

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 C2000 devices, related peripherals, and other technical collateral, is available in the C2000 DSP product folder at: www.ti.com/c2000.

Data Sheet —

SPRS230: — TMS320F2801, TMS320F2806, TMS320F2808, UCD9501 Digital Signal Processors Data Manual contains the pinout, signal descriptions, as well as electrical and timing specifications for the F280x devices.

User's Guides/Reference Guides —

SPRU051: — TMS320x281x, 280x Serial Communication Interface (SCI) Reference Guide

Describes the SCI, which is a two-wire asynchronous serial port, commonly known as a UART. The SCI modules support digital communications between the CPU and other asynchronous peripherals that use the standard non-return-to-zero (NRZ) format.

SPRU059: — TMS320x281x, 280x 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. The SPI is used for communications between the DSP controller and external peripherals or another controller.

SPRU074: — TMS320x281x, 280x Enhanced Controller Area Network (eCAN) Reference Guide

Describes the eCAN that uses established protocol to communicate serially with other controllers in electrically noisy environments. With 32 fully configurable mailboxes and time-stamping feature, the eCAN module provides a versatile and robust serial communication interface. The eCAN module implemented in the 281x DSP is compatible with the CAN 2.0B standard (active).

SPRU430: — TMS320C28x DSP 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.

SPRU513: — TMS320C28x Assembly Language Tools 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 Compiler 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.

SPRU566: — TMS320x281x, 280x Peripheral Reference Guide

Describes the peripheral reference guides of the 28x digital signal processors (DSPs).

SPRU608: — The 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.

SPRU625: — TMS320C28x DSP/BIOS Application Programming Interface (API) Reference Guide

Describes development using DSP/BIOS.

SPRU712: — TMS320x280x System Control and Interrupts Reference Guide

Describes the various interrupts and system control features of the 280x digital signal processors (DSPs).

SPRU716: — TMS320x280x Analog-to-Digital Converter (ADC) Reference Guide

Describes the ADC module. The module is a 12-bit pipelined ADC. The analog circuits of this converter, referred to as the core in this document, include the front-end analog multiplexers (MUXs), sample-and-hold (S/H) circuits, the conversion core, voltage regulators, and other analog supporting circuits. Digital circuits, referred to as the wrapper in this document, include programmable conversion sequencer, result registers, interface to analog circuits, interface to device peripheral bus, and interface to other on-chip modules.

SPRU721: — TMS320x280x Inter-Integrated Circuit (I2C) Reference Guide

Describes the features and operation of the inter-integrated circuit (I^2C) module that is available on the TMS320x280x digital signal processor (DSP). The I2C module provides an interface between one of these DSPs and devices compliant with Philips Semiconductors Inter-IC bus (I2C-bus) specification version 2.1 and connected by way of an I2C-bus.

SPRU722: — TMS320x280x Boot ROM Reference Guide

Describes the purpose and features of the bootloader (factory-programmed boot-loading software). It also describes other contents of the device on-chip boot ROM and identifies where all of the information is located within that memory.

SPRU790: — TMS320x280x Enhanced Quadrature Encoder Pulse (eQEP) Reference Guide

Describes the 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 and registers.

SPRU791: — TMS320x280x Enhanced Pulse Width Modulator (ePWM) Module Reference Guide

The PWM peripheral is an essential part of controlling many of the power related systems found in both commercial and industrial equipments. This guide describes the main areas that include digital motor control, switch mode power supply control, UPS (uninterruptible power supplies), and other forms of power conversion. The PWM peripheral can be considered as performing a DAC function, where the duty cycle is equivalent to a DAC analog value, it is sometimes referred to as a Power DAC.

SPRU807: — TMS320x280x Enhanced Capture (eCAP) Module Reference Guide

Describes the enhanced capture module. It includes the module description and registers.

Application Reports —

SPRAA58: — TMS320x281x to TMS320x280x Migration Overview describes differences between the Texas Instruments TMS320x281x and TMS320x280x DSPs to assist in application migration from the 281x to the 280x. While the main focus of this document is migration from 281x to 280x, users considering migrating in the reverse direction (280x to 281x) will also find this document useful.

SPRA550: — 3.3 V DSP for Digital Motor Control describes a scenario of a 3.3-V-only motor controller indicating that for most applications, no significant issue of interfacing between 3.3 V and 5 V exists. On-chip 3.3-V analog-to-digital converter (ADC) versus 5-V ADC is also discussed. Guidelines for component layout and printed circuit board (PCB) design that can reduce system noise and EMI effects are summarized.

SPRA820: — Online Stack Overflow Detection on the TMS320C28x DSP presents the methodology for online stack overflow detection on the TMS320C28x™ DSP. C-source code is provided that contains functions for implementing the overflow detection on both DSP/BIOS™ and non-DSP/BIOS applications.

SPRA861: — RAMDISK: A Sample User-Defined C I/O Driver provides an easy way to use the sophisticated buffering of the high-level CIO functions on an arbitrary device. This application report presents a sample implementation of a user-defined device driver.

SPRA953: — IC Package Thermal Metrics describes the traditional and new thermal metrics and will put their application in perspective with respect to system level junction temperature estimation.

SPRA963: — Reliability Data for TMS320LF24x and TMS320F281x Devices describes reliability data for TMS320LF24x and TMS320F281x devices.

SPRA991: — Simulation Fulfills its Promise for Enhancing Debug and Analysis - A White Paper describes simulation enhancements that enable developers to speed up the development cycle by allowing them to evaluate system alternatives more effectively.

Trademarks

C2000, TMS320C28x are trademarks of Texas Instruments.

High-Resolution Pulse Width Modulator (HRPWM)

This document is used in conjunction with the *TMS320x280x Enhanced Pulse Width Modulator (ePWM) Module Reference Guide* (literature number SPRU791).

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. This occurs at PWM frequencies greater than ~200 kHz when using a CPU/system clock of 100 MHz. The key features of HRPWM are:

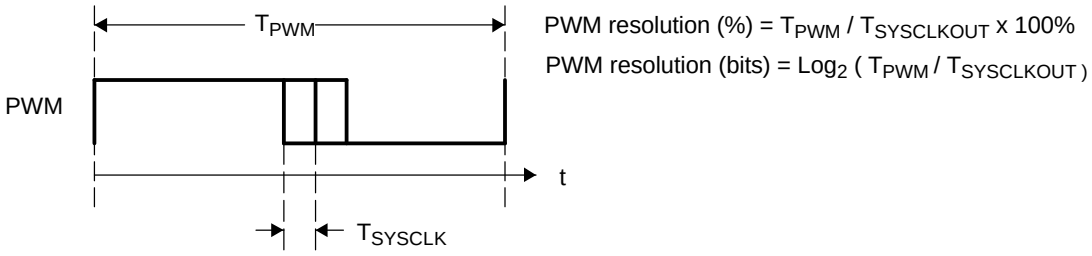
- Extended time resolution capability
- Used in both duty cycle and phase-shift control methods
- Finer time granularity control or edge positioning using extensions to the Compare A and Phase registers
- Implemented using the "A" signal path of PWM, i.e., on the EPWMxA output. EPWMxB output has conventional PWM capabilities
- Self-check diagnostics software mode to check if the micro edge positioner (MEP) logic is running optimally

Topic **Page**

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, where $T_{SYSCLKOUT}$ = 10 ns (i.e. 100 MHz clock), 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



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. Table 1 values assume a MEP step size of 180 ps. See the device data sheet for typical and maximum performance specifications for the MEP.

Table 1. Resolution for PWM and HRPWM

PWM Freq (kHz)	Regular Resolution (PWM)		High Resolution (HRPWM)	
	Bits	%	Bits	%
20	12.3	0.0	***18.1***	0.000
50	11.0	0.0	***16.8***	0.001
100	10.0	0.1	***15.8***	0.002
150	9.4	0.2	***15.2***	0.003
200	9.0	0.2	***14.8***	0.004
250	8.6	0.3	***14.4***	0.005
500	7.6	0.5	***13.8***	0.007
1000	6.6	1.0	***12.4***	0.018
1500	6.1	1.5	***11.9***	0.027
2000	5.6	2.0	***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:

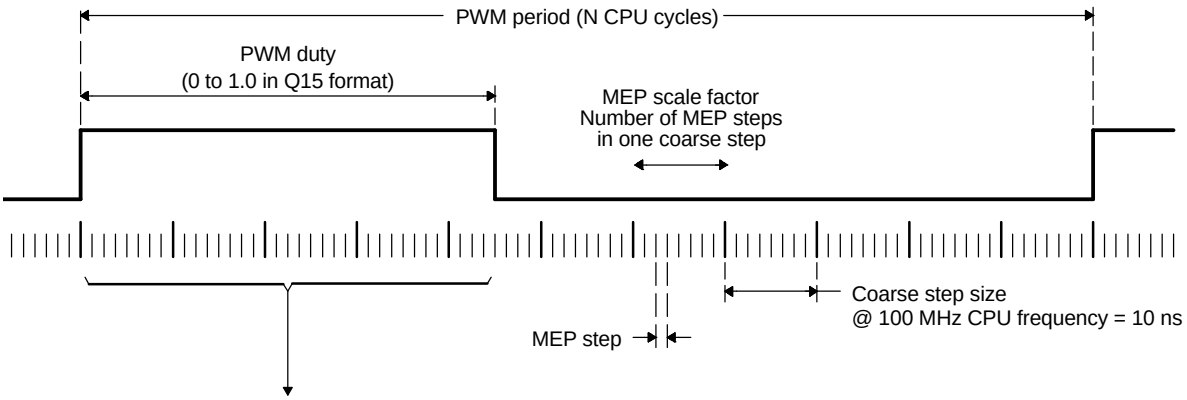
- Single-phase buck, boost, and flyback
- Multi-phase buck, boost, and flyback
- Phase-shifted full bridge
- Direct modulation of D-Class power amplifiers

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. 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



Number of coarse steps = integer(PWMduty * PWMperiod)
Number of MEP steps = fraction(PWMduty * PWMperiod) * (MEPScaleFactor)

16−bit CMPA register value	= number of coarse steps
16−bit CMPAHR register value	= (number of MEP steps) << 8 + 0x180 (rounding)†

† For MEP range and rounding adjustment.

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 *TMS320x280x Enhanced Pulse Width Modulator (ePWM) Module Reference Guide*(literature number SPRU791).

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)
CMPAHR	0x0008	Yes	Extension Register for HRPWM Duty (8 bits)
HRCNFG(1)	0x0020	No	HRPWM Configuration Register

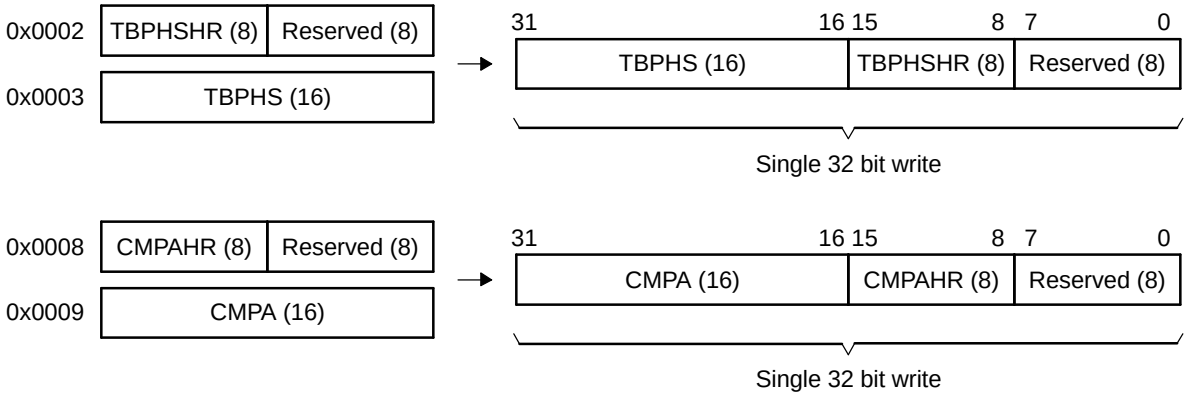
(1) This register is EALLOW protected.

2.1 Controlling the HRPWM Capabilities

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

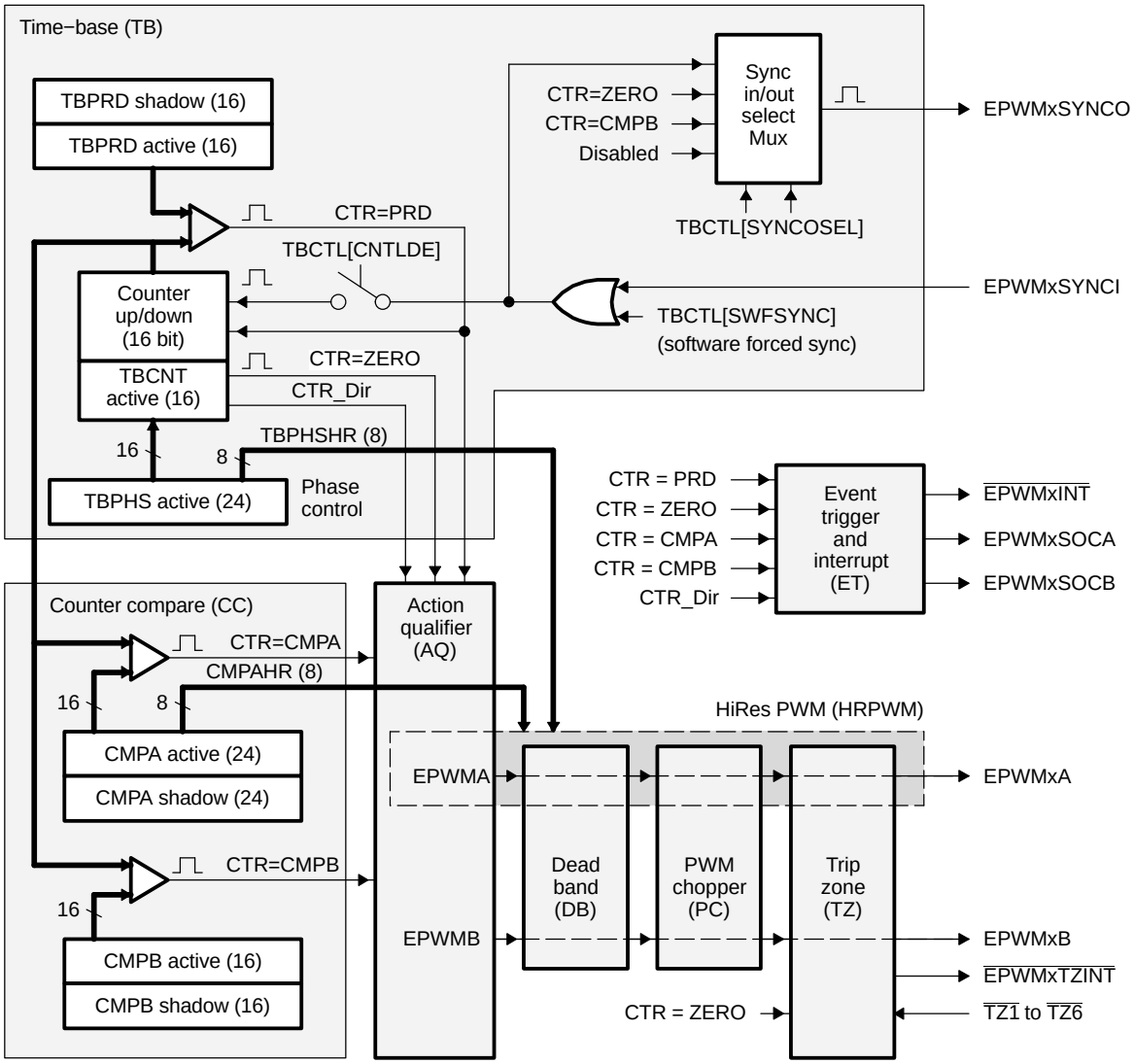
- TBPHSHR - Time Base Phase High Resolution Register
- CMPAHR - Counter Compare A High Resolution Register

Figure 3. HRPWM Extension Registers and Memory Configuration



HRPWM capabilities are controlled using the Channel A PWM signal path. Figure 4 shows how the HRPWM interfaces with the 8-bit extension registers.

Figure 4. HRPWM System Interface



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 configuration options for the following key operating modes:

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, while BE is used for topologies requiring phase shifting, e.g., phase shifted full bridge.

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.

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 register 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.

2.3 Principle of Operation

The MEP logic is capable of placing an edge in one of 255 (8 bits) discrete time steps, each of which has a time resolution on the order of 150 ps. 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 *TMS320F2808, TMS320F2806, TMS320F2801 Digital Signal Processors Data Manual* [literature number SPRS230]).
(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 100 MHz, the duty cycle choices are in the vicinity of 40.5%. In Figure 5, a compare value of 32 counts (i.e. duty = 40%) is the closest to 40.5% that you can attain. This is equivalent to an edge position of 320 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, 22 steps of the MEP (CMPAHR register) will position the edge at 323.96 ns, resulting in almost zero error. In this example, it is assumed that the MEP has a step resolution of 180 ns.

Figure 5. Required PWM Waveform for a Requested Duty = 40.5%

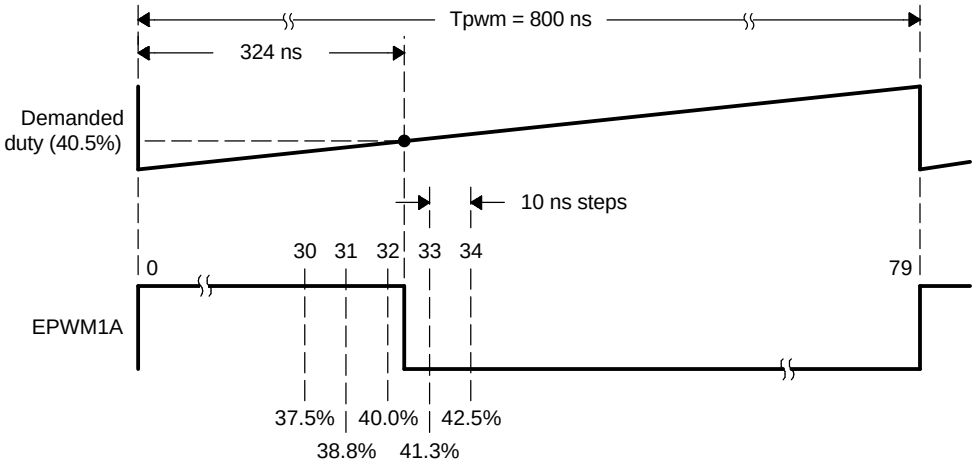


Table 4. CMPA vs Duty (left), and [CMPA:CMPAHR] vs Duty (right)

CMPA (count)[1][2][3]	DUTY %	High Time (ns)	CMPA (count)	CMPAHR (count)	Duty (%)	High Time (ns)
28	***35.0***	280	32	18	***40.405***	323.24
29	***36.3***	290	32	19	***40.428***	323.42
30	***37.5***	300	32	20	***40.450***	323.60
31	***38.8***	310	32	21	***40.473***	323.78
32	***40.0***	320	32	22	***40.495***	323.96
33	***41.3***	330	32	23	***40.518***	324.14
34	***42.5***	340	32	24	***40.540***	324.32
			32	25	***40.563***	324.50
Required			32	26	***40.585***	324.68
32.40	***40.5***	324	32	27	***40.608***	324.86

(1) System clock, SYSCLKOUT and TBCLK = 100 MHz, 10 ns
(2) For a PWM Period register value of 80 counts, PWM Period = 80 x 10 ns = 800 ns, PWM frequency = 1/800 ns = 1.25 MHz
(3) Assumed MEP step size for the above example = 180 ps
See the *TMS320F2808, TMS320F2806, TMS320F2801/UCD9501 Digital Signal Processors Data Manual* (literature number SPRS230) 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.

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	=	10 ns (100 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/10 ns)	=	80
Number of MEP steps per coarse step at 180 ps (10 ns/180 ps), **MEP_SF**	=	55
Value to keep CMPAHR within the range of 1-255 and fractional rounding constant (default value)	=	0180h

Step 1: Percentage Integer Duty value conversion for CMPA register

CMPA register value	=	int(**PWMDuty***PWMperiod**); int means integer part
	=	int(0.405*80)
	=	int(32.4)
CMPA register value	=	32 (20h)

Step 2: Fractional value conversion for CMPAHR register

CMPAHR register value	=	(frac(**PWMDuty***PWMperiod**)*MEP_SF**) << 8) + 0180h; frac means fractional part
	=	(frac(32.4)*55 <<8) + 0180h; Shift is to move the value as CMPAHR high byte
	=	((0.4*55) << 8) + 0180h
	=	(22<<8) + 0180h
	=	22*256 + 0180h ; Shifting left by 8 is the same multiplying by 256.
	=	5632 + 0180h
	=	1600h + 0180h
CMPAHR value	=	1780h; CMPAHR value = 1700h, lower 8 bits will be ignored by hardware.

Note: The MEP scale factor (MEP_SF) 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 280x CPU can write this as a single concatenated value, i.e. [CMPA:CMPAHR].

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 2.3.1.

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 TBCLK cycles after the period starts when diagnostics are disabled
- 6 TBCLK cycles after the period starts when SFO diagnostics are running

Duty cycle range limitations are illustrated in Figure 6. This limitation imposes a lower duty cycle limit on the MEP, for example, precision edge control is not available all the way down to 0% duty cycle. Although for the first 3 or 6 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.

Figure 6. Low % Duty Cycle Range Limitation Example When PWM Frequency = 1 MHz

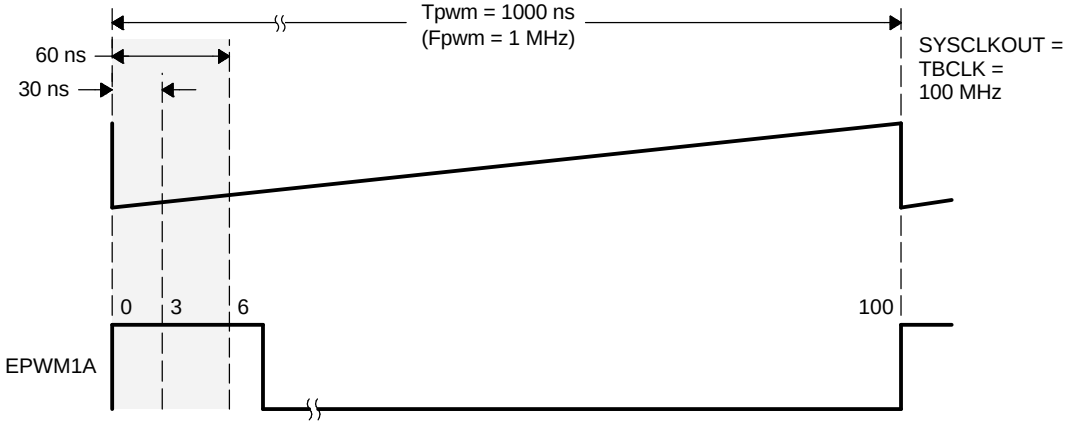


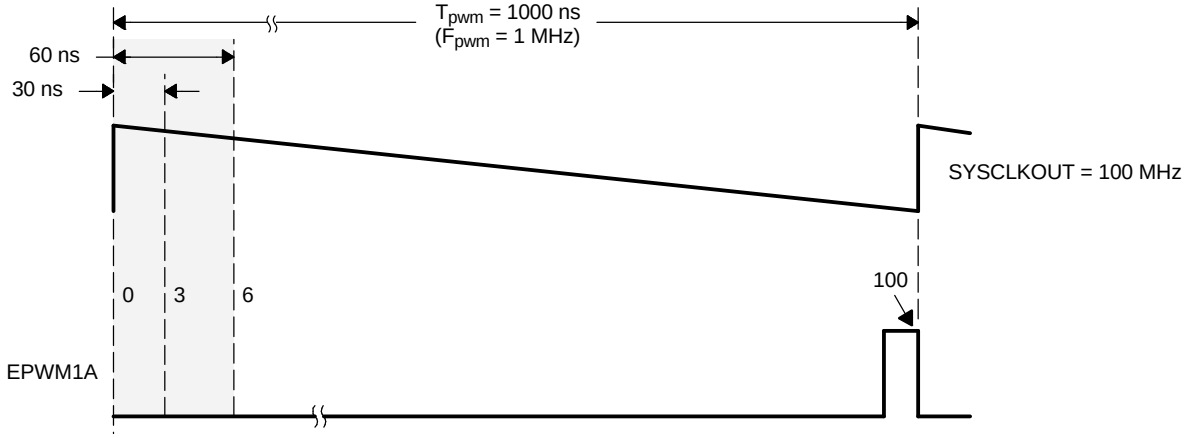
Table 5. Duty Cycle Range Limitation for 3 and 6 TBCLK Cycles

PWM Frequency[(1)] (kHz)	3 Cycles Minimum Duty	6 Cycles Minimum Duty
200	0.6%	1.2%
400	1.2%	2.4%
600	1.8%	3.6%
800	2.4%	4.8%
1000	3.0%	6.0%
1200	3.6%	7.2%
1400	4.2%	8.4%
1600	4.8%	9.6%
1800	5.4%	10.8%
2000	6.0%	12.0%

(1) System clock - $T_{SYSCLKOUT}$ = 10 ns
System clock = TBCLK = 100 MHz

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. This is illustrated in Figure 7. 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 7. High % Duty Cycle Range Limitation Example when PWM Frequency = 1 MHz



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. 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 functions. SFO functions help 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 to [CMPA:CMPAHR] mapping function (see Section 2.3.2), the correct value for the MEP scaling factor (MEP_SF) needs to be known by the software. To accomplish this, each HRPWM module has built in self-check and diagnostics capabilities that can be used to determine the optimum MEP_SF value for any operating condition. TI provides a C-callable library containing two SFO functions that utilize this hardware and determine the optimum MEP_SF. As such, MEP Control and Diagnostics registers are reserved for TI use.

Table 6 provides functional descriptions of the two SFO library routines.

Table 6. SFO Library Routines

Function	Description
SFO_MepDis(n)	Scale Factor Optimizer with MEP Disabled
	This routine runs faster, as the calibration logic works when HRPWM capabilities are disabled; therefore, HRPWM capabilities cannot be run concurrently when the ePWMn is being used. If SYSCLKOUT = TBCLK = 100 MHz and assuming MEP steps size is 150 ps Typical value at 100 MHz = 66 MEP steps per unit TBCLK (10 ns) The function returns a value in the variable array: MEP_ScaleFactor[n] = Number of MEP steps/SYSCLKOUT If TBCLK is not equal to SYSCLKOUT, then the returned value must be adjusted to reflect the correct TBCLK: MEP steps per TBCLK = MEP_ScaleFactor[n] * (SYSCLKOUT/TBCLK)(1) Example: If TBCLK =SYSCLKOUT/2, MEP steps per TBCLK = MEP_ScaleFactor[n] * (100/50) =66 * 2 = 132 (1) **Constraints when using this function:** SFO_MepDis(n) can be used with SYSCLKOUT from 50 MHz to 100 MHz. MEP diagnostics logic uses SYSCLKOUT not TBCLK and hence SYSCLKOUT restriction is an important constraint. SFO_MepDis(n) function does not require a starting Scale Factor value. **When to use** If one of the ePWM modules is not used in HRPWM mode, then it can be dedicated to run the SFO diagnostics for the modules that are running HRPWM mode. Here the single MEP_SF value obtained can be applied to other ePWM modules. This assumes that all HRPWM module's MEP steps are similar but may not be identical. The ePWM module that is not active in HRPWM mode is still fully operational in conventional PWM mode and can be used to drive PWM pins. The SFO function only makes use of the MEP diagnostics logic. The other ePWM modules operating in HRPWM mode incur only a 3-cycle minimum duty limitation.
SFO_MepEn(n)	Scale Factor Optimizer with MEP Enabled
	This routine runs slower as the calibration logic is used concurrently while HRPWM capabilities are being used by the ePWM module. If SYSCLKOUT = TBCLK = 100 MHz and assuming MEP steps size is 150 ps Typical value at 100 MHz = 66 MEP steps per unit TBCLK (10 ns) The function returns a value in the variable array: MEP_ScaleFactor[n](1) = Number of MEP steps/SYSCLKOUT = Number of MEP steps/TBCLK **Constraints when using this function:** SFO_MepEn(n) function is restricted to be used with SYSCLKOUT at 100 MHz only. MEP diagnostics logic uses SYSCLKOUT not TBCLK and hence SYSCLKOUT restriction is an important constraint. SFO_MepEn(n) function does require a starting Scale Factor value.MEP_ScaleFactor[0] needs to be initialized to a typical MEP step size value **When to use** If the application requires all ePWM modules to have HRPWM capability (i.e., MEP is operational), then the SFO_MepEn(n) function should run for each of the active ePWM modules with HRPWM capability. • In the above case, a 7-cycle MEP inactivity zone exists at the start of the PWM period. See Section 2.3.3 on duty cycle range limitation. • If all ePWM modules are using the same TBCLK prescaler, then it is also possible to run the SFO_MepEn(n) function for only one ePWM module and to use the SFO return value for the other modules. In this case only one ePWM module incurs the 7-cycle limitation, and remaining modules incur only a 3-cycle minimum duty limitation. See "Duty cycle limitation" section. This assumes that all HRPWM module's MEP steps are similar but may not be identical.

(1) n is the ePWM module number on which the SFO function operates.
e.g., n = 1, 2, 3, or 4 for the F2808. Check *TMS320F2808, TMS320F2806, TMS320F2801, UCD9501 Digital Signal Processors Data Manual* (SPRS230) for device configurations.

Both routines can be run as background tasks in a slow loop requiring negligible CPU cycles. In most applications only one of these routines will be needed. However, if the application has free HRPWM resources then both the routines could be used. The repetition rate at which an SFO function needs to be executed depends on the applications 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 HRPWM feature with no SFO diagnostics, HRPWM logic will not be active for the first 3 TBCLK cycles of the PWM period. While running the application in this configuration, if CMPA register value is less than 3 cycles, then its CMPAHR register must be cleared to zero. This would avoid any unexpected transitions on PWM signal.

However, if SFO diagnostic function SFO_MEPEn is used in the background, then HRPWM logic will not be active for the first 6 TBCLK cycles of PWM period. While using SFO_MEPEn function if CMPA register value is less than 6 cycles, then its CMPAHR register must be cleared to zero. This would avoid any unexpected transitions on PWM signal. Also note that the SFO_MEPDis function cannot be used concurrently with PWM signals see the previous section for details

2.4.1 Software Usage

Software library functions SFO_MepEn(int n) and SFO_MepDis(int n) calculate the MEP scale factor for ePWMn modules, where n = 1, 2, 3…etc. 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 array of integer variables of length 5 called MEP_ScaleFactor[5]. For example, see Table 7.

Table 7. Factor Values

Software function calls	Functional description	Updated Variable MEP_ScaleFactor[5](1)
SFO_MepDis(n)		
SFO_MepDis(1);	Returns the scale factor value to array index 1	MEP_ScaleFactor[1]
SFO_MepDis(2);	Returns the scale factor value to array index 2	MEP_ScaleFactor[2]
SFO_MepDis(3);	Returns the scale factor value to array index 3	MEP_ScaleFactor[3]
SFO_MepDis(4);	Returns the scale factor value to array index 4	MEP_ScaleFactor[4]
SFO_MepEn(n)		
SFO_MepEn(1);	Returns the scale factor value to array index	MEP_ScaleFactor[1]
SFO_MepEn(2);	Returns the scale factor value to array index 2	MEP_ScaleFactor[2]
SFO_MepEn(3);	Returns the scale factor value to array index 3	MEP_ScaleFactor[3]
SFO_MepEn(4);	Returns the scale factor value to array index 4	MEP_ScaleFactor[4]

(1) MEP_ScaleFactor[0] variable is a starting value and used by the SFO software functions internally

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

Step 1. Add Include Files

The SFO.h lib needs to be included as follows. This include file is mandatory while using the SFO library function. The *C280x C/C++ Header Files and Peripheral Examples* (literature number SPRC191) DSP280x_Device.h and DSP280x_PWM_defines.h are necessary as they are used with all TI software examples. These include files are optional if customized header files are used in the end applications.

Example 1. A Sample of How to Add Include Files

```
#include "DSP280x_Device.h"       // DSP280x Headerfile
#include "DSP280x_EPWM_defines.h" // init defines
#include "SFO.h                   // SFO lib functions (needed for HRPWM)
```

Step 2. Element Declaration

Declare a 5-element array of integer variables as follows:

Example 2. Declaring an Element

```
int    MEP_ScaleFactor[5] = {0,0,0,0,0};  // Scale factor values for ePWM1-4
int    MEP_SF1, MEP_SF2, MEP_SF3, MEP_SF4
volatile struct EPWM_REGS *ePWM[] = {&EPwm1Regs, &EPwm2Regs, &EPwm3Regs, &EPwm4Regs,
&EPwm5Regs, &EPwm6Regs};
```

Step 3. MEP_ScaleFactor Initialization

After power up, the SFO_MepEn(n) function needs a starting Scale Factor value. This value can be conveniently determined by using one of the ePWM modules to run the SFO_MepDis(n) function prior to configuring its PWM outputs for the application. SFO_MepDis(n) function does not require a starting Scale Factor value.

As part of the one-time initialization code, include the following:

Example 3. Initializing With a Scale Factor Value

```
//     MEP_ScaleFactor variables initialized using function SFO_MepDis
while (MEP_ScaleFactor[1] == 0) SFO_MepDis(1);    //SFO for HRPWM1
while (MEP_ScaleFactor[2] == 0) SFO_MepDis(2);    //SFO for HRPWM2
while (MEP_ScaleFactor[3] == 0) SFO_MepDis(3);    //SFO for HRPWM3
while (MEP_ScaleFactor[4] == 0) SFO_MepDis(4);    //SFO for HRPWM4

//    Initialize a common seed variable MEP_ScaleFactor[0]
//    required for all SFO functions
      MEP_ScaleFactor[0] = MEP_ScaleFactor[1];    //Common variable for SFOMepEN(n) function
```

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 *C280x C/C++ Header Files and Peripheral Examples* (SPRC191) available from the TI website.

Example 4. SFO Function Calls

```
main()
{
    //  User code
    // Case1: ePWM1,2,3,4 are running in HRPWM mode
        SFO_MepEn(1);                       // Each of these of function enables
        SFO_MepEn(2);                       // the respective MEP diagnostic logic
        SFO_MepEn(3);                       // and returns MEP Scale factor value
        SFO_MepEn(4);

        MEP_SF1 = MEP_ScaleFactor[1];       // used for ePWM1
        MEP_SF2 = MEP_ScaleFactor[2];       // used for ePWM2
        MEP_SF3 = MEP_ScaleFactor[3];       // used for ePWM3
        MEP_SF4 = MEP_ScaleFactor[4];       // used for ePWM4

    // Case2:ePWM1,2,3 only are running in HRPWM mode.
             One of the ePWM channel(as an example ePWM4) is used as for
             Scale factor calibration
   // Here minimum duty cycle limitation is 3 clock cycles.
   //
   // HRPWM 4 MEP diagnostics circuit is used to estimate the MEP steps
   // with the assumption that all HRPWM channels behave similarly
   // though may not be identical.
        SFO_MepDis(4);                      // MEP steps using ePWM4
        MEP_SF1 = MEP_ScaleFactor[4];       // used for ePWM1
        MEP_SF2 = MEP_SF1                   // used for ePWM2
        MEP_SF3 = MEP_SF1                   // used for ePWM3
        MEP_SF4 = MEP_SF1                   // used for ePWM4
 }
```

2.5 HRPWM Examples Using Optimized Assembly Code.

The best way to understand how to use the HRPWM capabilities is through 2 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 *TMS320x280x Enhanced Pulse Width Modulator (ePWM) Module Reference Guide* (literature number SPRU791) are also used.

Example 5 This example assumes MEP step size of 150 ps and does not use the SFO library.

Example 5. #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
```

2.5.1 Implementing a Simple Buck Converter

In this example, the PWM requirements are:

- PWM frequency = 1 MHz (i.e., TBPRD = 100)
- PWM mode = asymmetrical, up-count
- Resolution = 12.7 bits (with a MEP step size of 150 ps)

Figure 8 and Figure 9 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 8. Simple Buck Controlled Converter Using a Single PWM

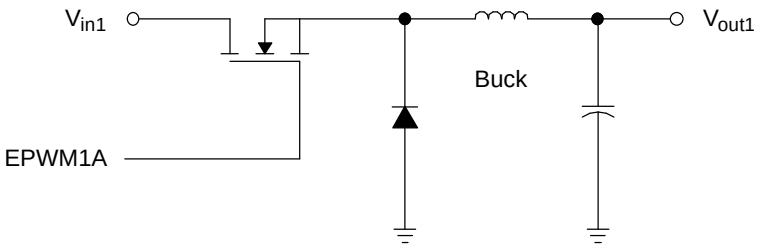
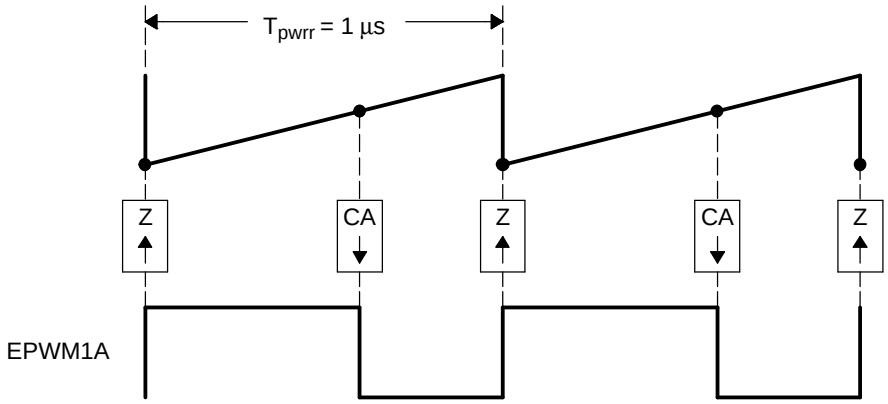


Figure 9. PWM Waveform Generated for Simple Buck Controlled Converter



The example code shown consists of two main parts:

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

Example 6 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 6. 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 1000 kHz PWM
     hrbuck_period = 200;                         // 2 x Period, 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.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_SF = 66*256;                              // Start with typical Scale Factor
                                                   //value for 100 MHz
                                // Note: Use SFO functions to update MEP_SF dynamically
}
```

Example 7 shows an assembly example of run-time code for the HRPWM buck converter.

Example 7. 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
     MOV   T,@_MEP_SF            ; 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, #0x180           ; 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 = 250)
- PWM mode = Asymmetrical, Up-count
- Resolution = 14 bits (MEP step size = 150 ps)

Figure 10 and Figure 11 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 10. Simple Reconstruction Filter for a PWM Based DAC

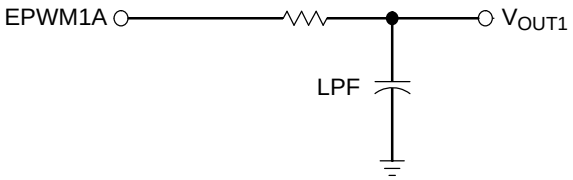
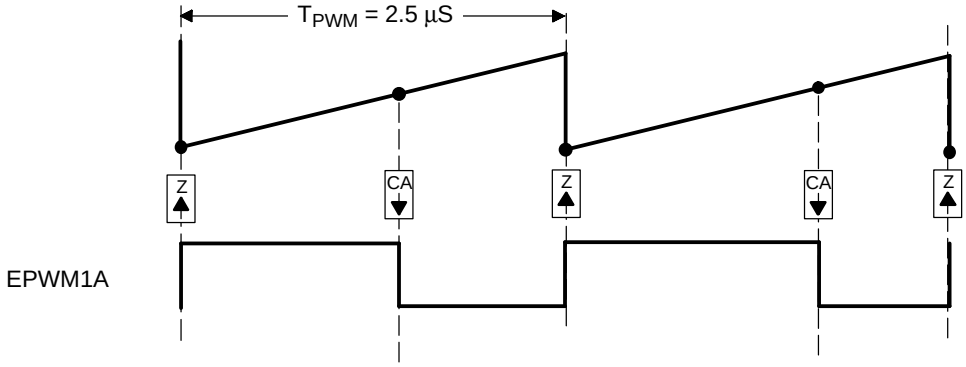


Figure 11. PWM Waveform Generated for the PWM DAC Function



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_SP and does not use the SFO library.

Example 8 shows the Initialization code. The first part is configured for conventional PWM. The second part sets up the HRPWM resources.

Example 8. PWM DAC Function Initialization Code

```
oid HrPwmDacDrvCnf(void)
{
     // Config for conventional PWM first
    EPwm1Regs.TBCTL.bit.PRDLD = TB_IMMEDIATE;       // Set Immediate load
    EPwm1Regs.TBPRD = 250;                          // Period set for 400 kHz PWM
    hrDAC_period = 250;                             // 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_SF = 66*256;                               // Start with typical Scale Factor
                                                   // value for 100 MHz.
                                                   // Use SFO functions to update MEP_SF
                                                   // dynamically.
}
```

Example 9 shows an assembly example of run-time code that can execute in a high-speed ISR loop.

Example 9. 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_SF                  ; 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, #0x180                 ; 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 Table 8.

Table 8. Register Descriptions

Name	Offset	Size (x16)	Description
Time Base Registers			
TBCTL	0x0000	1/0	Time Base Control Register
TBSTS	0x0001	1/0	Time Base Status Register
TBPHSHR	**TBPHSHR**	**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 [3]
Reserved	0x0006	1/0	
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 [4]
EPWM Registers			
ePWM	0x0000 to 0x001F	32	Other ePWM registers including the ones given above.
HRCNFG	**0x0020**	**1**	**HRPWM Configuration Register**
EPWM/HRPWM Test Registers			
Reserved	0x0030 0x003F	16	

3.2 Registers and Field Descriptions

Figure 12. HRPWM Configuration Register (HRCNFG)

15					8
Reserved					
R-0					
7	**4**	**3**	**2**	**1**	**0**
Reserved		HRLOAD	CTLMODE	EDGMODE	
R-0		R/W-0	R/W-0	R/W-0	

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

Table 9. HRPWM Configuration Register (HRCNFG) Field Descriptions

Bit	Field	Value	Description[(1)]
15-4	Reserved		Reserved
3	HRLOAD		Shadow mode bit: Selects the time event that loads the CMPAHR shadow value into the active register:
		00	CTR=ZRO (counter equal zero)
		01	CTR=PRD (counter equal period)
		10	CTR=ZRO, or CTR=PRD
		11	No loads, value is not made active
			Note: Load mode selection is valid only if CTLMODE=0 has been selected (bit 2). You should select this event to match the selection of the CMPA load mode (i.e., CMPCTL[LOADMODE] bits).
2	CTLMODE		Control Mode Bits: Selects the register (CMP or TBPHS) that controls the MEP:
		0	CMPAHR(8) Register controls the edge position (i.e., this is duty 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
		10	MEP control of falling edge
		11	MEP control of both edges

(1) This register is EALLOW protected.

Figure 13. Counter Compare A High Resolution Register (CMPAHR)

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

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

Table 10. Counter Compare A High Resolution Register (CMPAHR) Field Descriptions

Bit	Field	Value	Description
15-8	CMPAHR		Compare A High Resolution register bits for MEP step control. A minimum value of 0x0001 is needed to enable HRPWM capabilities. Valid MEP range of operation 1-255h.
7-0	Reserved		

Figure 14. TB Phase High Resolution Register (TBPHSHR)

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

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

Table 11. TB Phase High Resolution Register (TBPHSHR) Field Descriptions

Bit	Field	Value	Description
15-8	TBPHSH		Time base phase high resolution bits
7-0	Reserved		

Appendix A Revision History

This document was revised to SPRU924A from SPRU924. This appendix lists only revisions made in the most recent version. The scope of the revisions was limited to technical changes as shown in Table A-1.

Table A-1. Changes Made in Revision A

Location	Additions, Deletions, Modifications
Global	Reformatted to single chapter
Figure 4	Modified HRPWM System Interface figure
Section 2.4	Added section on Scale Factor Optimizing Software (SFO)
Example 2	Added a line of code to the example of initializing code with a scale factor value
Section 2.5	Changed header from "Using the HRPWM Module" to "HRPWM Examples Using Optimized Code".
Figure 6	Modified Low % Duty Cycle Range Limitation Example When PWM Frequency = 1 MHz figure
Table 5	Modified to reflect correct percentages
Figure 7	Modified High % Duty Cycle Range Limitation Example When PWM Frequency = 1 MHz figure

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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