

## Table of contents

1	Overview.....	2
2	Module Properties.....	2
3	Module Input and Output Definitions.....	3
3.1	Module Inputs.....	3
3.2	Module Outputs .....	4
4	Module API Description.....	4
4.1	HrPSFBDrvCnf .....	4
4.2	HR_PSFB_DRV_INIT .....	5
4.3	HR_PSFB_DRV.....	5
5	Usage Example:.....	7
6	Detailed description.....	8

## Table of Figures

Figure 1.	High resolution phase shifted full bridge driver .....	2
Figure 2.	Full bridge power converter.....	8
Figure 3.	Phase shifted PWM generation with the F280x EPWM module. ....	8

## Index of Tables

Table 1.	HR_PSFB_DRV module dependencies.....	2
Table 2.	HR_PSFB_DRV module components .....	2
Table 3.	HR_PSFB_DRV module miscellaneous properties.....	3
Table 4.	HR_PSFB_DRV module component files .....	3

## 1 Overview

This module controls the PWM generators in the EV peripheral to control a full bridge by using the phase shifting approach, whereby providing the zero voltage switching capabilities. This module forms the interface between the control software and the device PWM pins. In addition to phase control, the module offers control over left and right leg dead-band amounts.

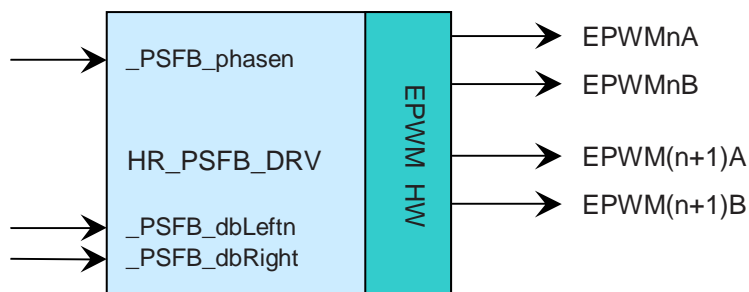


Figure 1. High resolution phase shifted full bridge driver

## 2 Module Properties

This section describes module properties, such as compatible devices, components, invocation etc. The HR\_PSFB\_DRV module has the following dependencies:

Module	Dependency
CPU dependency	C28x
Device dependency	x2801 / x2806 / x2808 members only

Table 1. HR\_PSFB\_DRV module dependencies

The HRBUCK\_DRV module has the following components:

Component	Present
C-based initialization	Yes
ASM interrupt initialization	Yes
ASM runtime macro	Yes

Table 2. HR\_PSFB\_DRV module components

The HR\_PSFB\_DRV module has the following miscellaneous properties:

Property name	Property value
Multiple instance support	No
Reentrant	No
Accessible from 'C' environment	Yes
Full configuration from 'C' environment	Yes
Input / Output connection	Pointer to signal net.

*Table 3. HR\_PSFB\_DRV module miscellaneous properties*

Component	Files
Macro source:	C:\tidcs\DPS_C280x\vXYZ <sup>†</sup> \dplib280x\dplib280x.inc
C Interface:	C:\tidcs\DPS_C280x\vXYZ\dplib280x\dplib280x.h
C source	C:\tidcs\DPS_C280x\vXYZ\dplib280x\dplib280x.src
Object file archive	C:\tidcs\DPS_C280x\vXYZ\dplib280x\dplib280x.lib

*Table 4. HR\_PSFB\_DRV module component files*

## 3 Module Input and Output Definitions

### 3.1 Module Inputs

Input name	Description	Format	Range
Phase	Output phase control	Pointer to a 16-bit fixed point input	Q15: [0, 1] or [0, 32767]
_PSFB_dbLeftn	Left leg dead-band (i.e. between EPWMnA & EPWMnB)	Pointer to a 16-bit fixed point input	Q15: [0, 1] or [0, 32767]
_PSFB_dbRightn	Right leg dead-band (i.e. between EPWM(n+1)A & EPWM(n+1)B)	Pointer to a 16-bit fixed point input	Q15: [0, 1] or [0, 32767]
SFPSFB	Scale factor input for the high resolution PWM MEP unit	Pointer to a 16-bit fixed point input	Q15: [0, 1] or [0, 32767]

<sup>†</sup> The xyz represents the version number directory level. For instance, a 1.00 release would have v100 in its directory path, and v210 would indicate a release 2.10.

## 3.2 Module Outputs

Output name	Description	Format	Range
EPWMnA, EPWMnB, EPWM(n+1)A, EPWM(n+1)B	F280x/C280x PWM output pins	Pulse width modulated output.	See device datasheet for electrical specifications.

## 4 Module API Description

This module has three executable code components, as described in Table 2. Each of these components is described in this section.

### 4.1 HrPSFBDrvCnf

**Function Name:** HrPSFBDrvCnf

**Prototype:** void HrPSFBDrvCnf(int16 nEPwmModule, int16 Period);

**Return value:** None.

**Preconditions:** The following preconditions must be satisfied:  
The appropriate EPWM module clock must be enabled in the PCLKCR1 register.

The HrPSFBDrvCnf function is called from the C environment, and performs driver configuration, including the selection of the target EPWM module, and the PWM period. This function should be executed once during the startup process.

❑ nEPwmModule: Specifies which two consecutive EPWM modules are initialized. The driver must be called with a valid nEPwmModule value, there exists no verification mechanisms to prevent incorrect execution with invalid arguments.

- **Valid Range:** 1-3, corresponding to EPWM1-3. The table below shows the modules selected.

nEPwmModule	Modules selected
1	EPWM1 and EPWM2
2	EPWM2 and EPWM3
3	EPWM3 and EPWM4

❑ Period: Specifies the PWM period in cycles, corresponding to the high speed peripheral clock.

- **Valid Range:** 1 to 32767.

**Example:** Call the HrPSFBDrvCnf function to initialize EPWM modules 1 and 2 to drive a phase shifted full bridge configuration, with a PWM frequency of 1000kHz

```
//-----  
// ePWM1 and 2 target, 1000KHz PWM (100 clock period with a 100MHz High  
// speed peripheral clock  
//-----  
HrPSFBDrvCnf(1, 100);
```

## 4.2 HR\_PSFB\_DRV\_INIT

<b>Function Name:</b>	HR_PSFB_DRV_INIT
<b>Prototype:</b>	HR_PSFB_DRV_INIT nEPwmModule
<b>Return value:</b>	None.
<b>Preconditions:</b>	The following preconditions must be satisfied: <ul style="list-style-type: none"><li>▪ The appropriate EPWM module clock must be enabled in the PCLKCR1 register, and the C language init routine must be called.</li><li>▪ The HrPSFBDrvCnf function must be called to initialize the peripheral registers etc.</li></ul>

This function is the assembler initialization macro, and must be called in addition to the C language initialization routine, for proper operation of the runtime macro routine. This initialization routine must be executed as part of an assembler initialization routine. This macro routine declares variables, initializes variables to known values, and sets up constants for the runtime macro routines.

**Example:** Call the HR\_PSFB\_DRV\_INIT to initialize the PSFB driver module.

```
;-----  
; ISR Initialisation  
;-----  
_ISR_Init:  HR_PSFB_DRV_INIT  
            LRETR
```

## 4.3 HR\_PSFB\_DRV

<b>Function Name:</b>	HR_PSFB_DRV
<b>Prototype:</b>	HR_PSFB_DRV
<b>Return value:</b>	None.
<b>Preconditions:</b>	The following preconditions must be satisfied: <ul style="list-style-type: none"><li>▪ The appropriate EPWM module clock must be enabled in the PCLKCR1 register.</li><li>▪ C language init routine must be called.</li><li>▪ The ISR initialization macro HR_PSFB_DRV must be instanced in an assembler initialization routine.</li></ul>

This function is the assembler run time macro, and this creates code that forms a bridge between software controllers and the PWM output. This routine writes values into the PWM control registers to control the PWM duty cycle.

**Example:** Call the HR\_PSFb\_DRV in an assembler ISR

```
-----  
;-----  
; Runtime interrupt service routine  
;-----  
_ISR_Run:    CONTEXT_SAVE                ;call macro  
  
            HR_PSFb_DRV  
;-----  
EXIT_ISR: ;Interrupt management before exit  
;-----  
            MOVW    DP,#ETCLR1>>6  
            MOV     @ETCLR1,#0x01        ; Clear EPWM1 Int flag  
  
;-----  
; Restore context & return  
;-----  
            CONTEXT_REST  
            IRET
```

## 5 Usage Example:

Step1. Instantiate the INIT macro in assembly (this is one-time pass through code)

```
; Instantiate the init macro  
  
    HR_PSFB_DRV_INIT
```

Step2. Instantiate the run time macro in assembly (this is usually looped or ISR code)

```
; "call" the main macro  
  
    HR_PSFB_DRV
```

Step3. (optional) Declare "Signal Nets" to "connect" the module to in "C"

```
int  Net1, Net2, Net3, Net4;
```

Step4. Declare the module "Terminal pointers" in "C"

```
// HR_PSFB_DRV terminal pointers, external references  
  
extern int  *PSFB_phase, *PSFB_dbLeft, *PSFB_dbRight, *SFPSFB;
```

Step5. "Connect" the module terminals to the Signal Nets in "C".

```
// ZVSFB_DRV connections  
  
    PSFB_phase    = &Net1;  
    PSFB_dbLeft   = &Net2;  
    PSFB_dbRight  = &Net3;  
    SFPSFB        = &Net4;  
  
// Note this can be done once during init, or dynamically during  
// run time operation, i.e. module connections can be  
// re-configured to other Nets as required by the application.
```

## 6 Detailed description

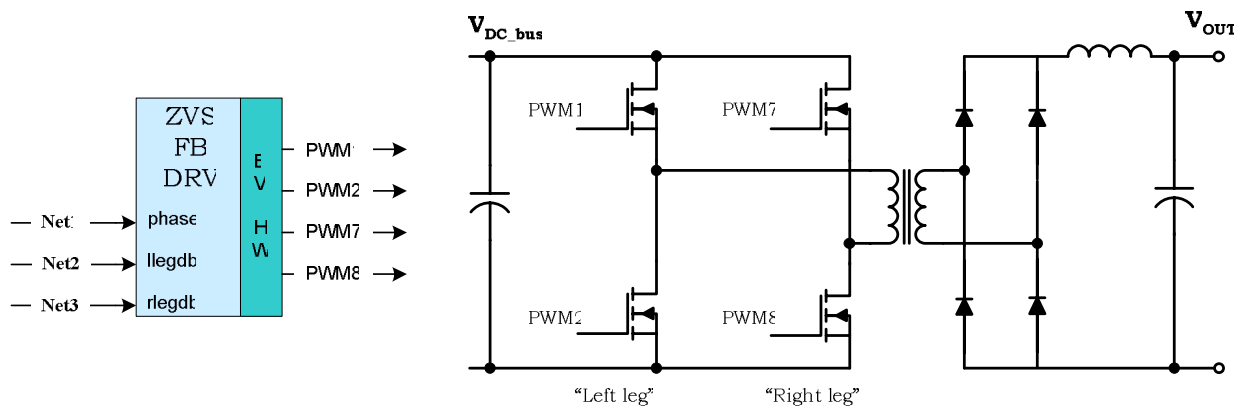


Figure 2. Full bridge power converter

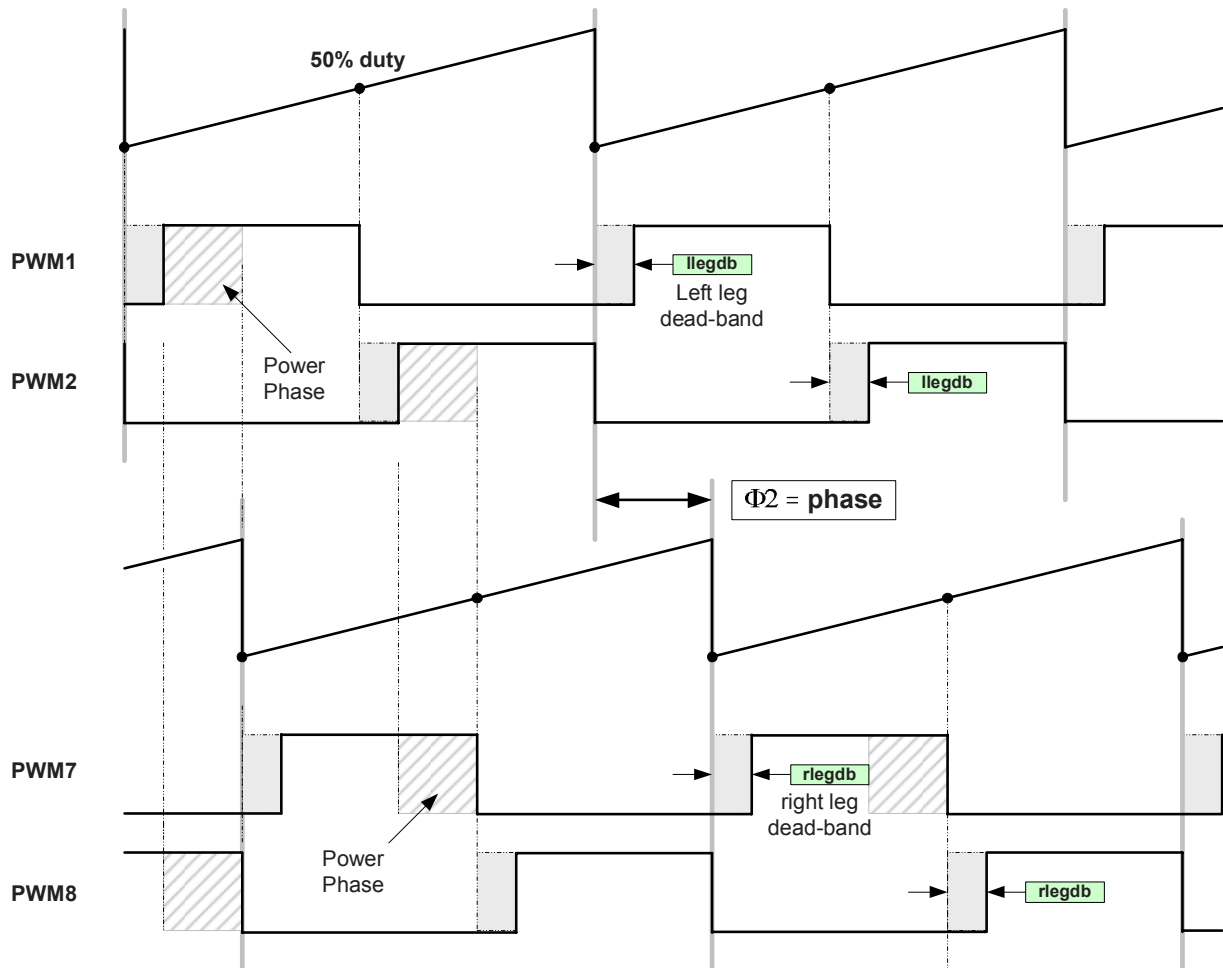


Figure 3. Phase shifted PWM generation with the F280x EPWM module.



**Phase calculation table**

			ZVSFB_phase	
Angle (deg)	per unit	%	Q15 (Dec)	Q15 (Hex)
180	1.00	100%	32767	7FFF
160	0.89	89%	29126	71C6
140	0.78	78%	25485	638D
120	0.67	67%	21845	5554
100	0.56	56%	18204	471B
90	0.50	50%	16384	3FFF
70	0.39	39%	12743	31C6
50	0.28	28%	9102	238D
30	0.17	17%	5461	1555
10	0.06	6%	1820	071C
0	0.00	0%	0	0000
-10	-0.06	-6%	63716	F8E3
-30	-0.17	-17%	60075	EAAA
-50	-0.28	-28%	56434	DC71
-70	-0.39	-39%	52793	CE38
-90	-0.50	-50%	49152	C000
-100	-0.56	-56%	47332	B8E3
-120	-0.67	-67%	43691	AAAA
-140	-0.78	-78%	40050	9C71
-160	-0.89	-89%	36409	8E38
-180	-1.00	-100%	32768	8000

**Dead band calculation table (nS)**

HSPCLK = **150** 1.00E+06 MHz (Note: this is the EV clock)

ZVSFB_xlegdb		ZVSFB_dbpscale (hex)					
DBT[3:0] (dec)	DBT[3:0] (hex)	00 1	04 2	08 4	0C 8	10 16	14 32
0	0	0	0	0	0	0	0
1	1	7	13	27	53	107	213
2	2	13	27	53	107	213	427
3	3	20	40	80	160	320	640
4	4	27	53	107	213	427	853
5	5	33	67	133	267	533	1067
6	6	40	80	160	320	640	1280
7	7	47	93	187	373	747	1493
8	8	53	107	213	427	853	1707
9	9	60	120	240	480	960	1920
10	A	67	133	267	533	1067	2133
11	B	73	147	293	587	1173	2347
12	C	80	160	320	640	1280	2560
13	D	87	173	347	693	1387	2773
14	E	93	187	373	747	1493	2987
15	F	100	200	400	800	1600	3200