Table of contents

Table of Figures

Index of Tables

1 Overview

This software module directly controls the EPWM peripherals on the 280x devices. It generates appropriate PWM signals to control a 3 phase interleaved buck by using 3 EPWM modules synchronized together with phase offset. This module forms the interface between the control software and the device PWM pins.

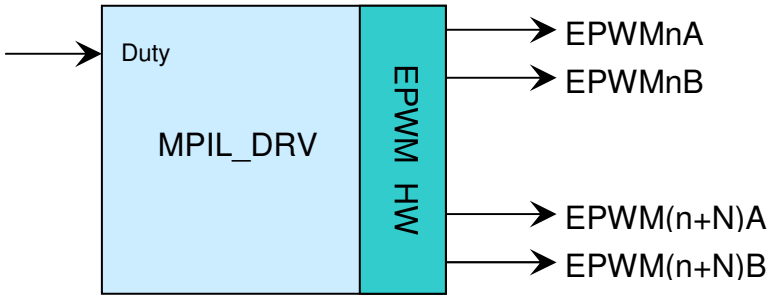


Figure 1. Multiphase buck converter, PWM driver module

2 Module Properties

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

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

Table 1. MPIL_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. MPIL_DRV module components

The MPIL_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. MPIL_DRV module miscellaneous properties

Component	Files
Macro source:	C:\tidcs\DPS_C280x\vXYZ[†]\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. MPIL_DRV module component files

3 Module Input and Output Definitions

3.1 Module Inputs

Input name	Description	Format	Range
Duty	Duty cycle control	Pointer to 16-bit fixed point input data	Q15: [0, 1] or [0, 32767]

3.2 Module Outputs

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

[†] 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.

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 MPILDrvCnf

Function Name:	MPILDrvCnf
Prototype:	void MPILDrvCnf(int16 n nEPwmModule, int16 nChCount, 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 HrBuckDrvCnf 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 EPWM module is intialized.
  - **Valid Range:** 1-6, corresponding to EPWM1-6.
- nChCount**:** Specifies how many EPWM modules are intialized.
  - **Valid Range:** (7-nEPwmModule).
- Period**:** Specifies the PWM period in cycles, corresponding to the high speed peripheral clock.
  - **Valid Range:** 1 to 32767.

Example: Call the MPILDrvCnf function to initalize EPWM1-3 modules.

```
//---------------------------------------------------------------------
// ePWM1-3 target, 1000KHz PWM (100 clock period with a 100MHz High
// speed peripheral clock
//---------------------------------------------------------------------
      MPILDrvCnf(1, 3, 100);
```

4.2 HRBUCK_DRV_INIT

Function Name: MPIL_DRV_INIT

Prototype: MPIL_DRV_INIT nEPwmModule, nChCount

Return value: None.

Preconditions: The following preconditions must be satisfied:

The appropriate EPWM module clock must be enabled in the PCLKCR1 register, and the C language init routine must be called.

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.

- nEPwmModule**:** Specifies which EPWM module is intialized.
  - **Valid Range:** 1-6, corresponding to EPWM1-6.
- nChCount**:** Specifies how many EPWM modules are intialized.
  - **Valid Range:** (7-nEPwmModule).

Example: Call the MPIL_DRV_INIT to initalize EPWM1-3 module.

```
;---------------------------------------------------------
; ISR Initialisation
;---------------------------------------------------------
_ISR_Init:  HRBUCK_DRV_INIT 1,3
            LRETR
```

4.3 MPIL_DRV

Function Name: MPIL_DRV

Prototype: MPIL_DRV nEPwmModule

Return value: None.

Preconditions: The following preconditions must be satisfied:

- The appropriate EPWM module clock must be enabled in the PCLKCR1 register.
- C language init routine must be called.
- The ISR initialization macro MPIL_DRV must be instanced in an assembler initialization routine.

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.

- nEPwmModule**:** Specifies which EPWM module is intialized.
  - **Valid Range:** 1-6, corresponding to EPWM1-6.
- nChCount**:** Specifies how many EPWM modules are intialized.
  - **Valid Range:** (7-nEPwmModule).

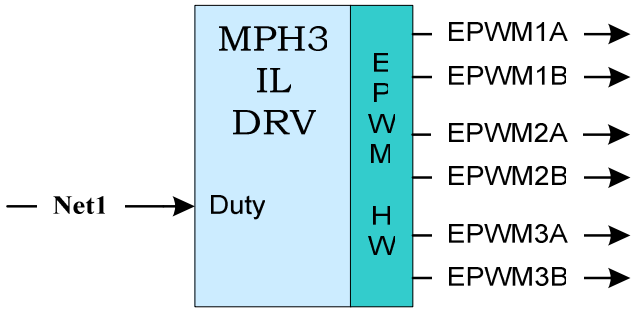
Example: Call the MPIL_DRV in an assembler ISR to implement runtime code for EPWM modules 1-3.

```
;---------------------------------------------------------
; Runtime interrupt service routine
;---------------------------------------------------------
_ISR_Run:   CONTEXT_SAVE            ;call macro
            MPIL_DRV 1,3
;---------------------------------------------------------
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:

Usage Example:



Step1. Call the driver configuration function in C (this is one-time pass through code)

```
     PWM_MPILDrvCnf();
```

Step2. Instantiate the INIT macro in assembly (this is one-time pass through code)

```
; Instantiate the init macro

     MPIL_DRV_INIT  nEPWMModule, nChCount
```

Step3. Instantiate the run time macro in assembly (this is usually looped or ISR code)

```
; “call” the main macro

     MPIL_DRV  nEPWMModule, nChCount
```

Step4. (optional) Declare “Signal Nets” to “connect” the module to in “C”

```
int  Net1;
```

Step5. Declare the module “Terminal pointers” in “C”

```
// MPH3IL_DRV terminal pointers, external references

extern int  *MPIL_Duty;
```

Step6. “Connect” the module terminals to the Signal Nets in “C”.

```
// MPIL_DRV connections

     MPIL_Duty = &Net1;

// 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

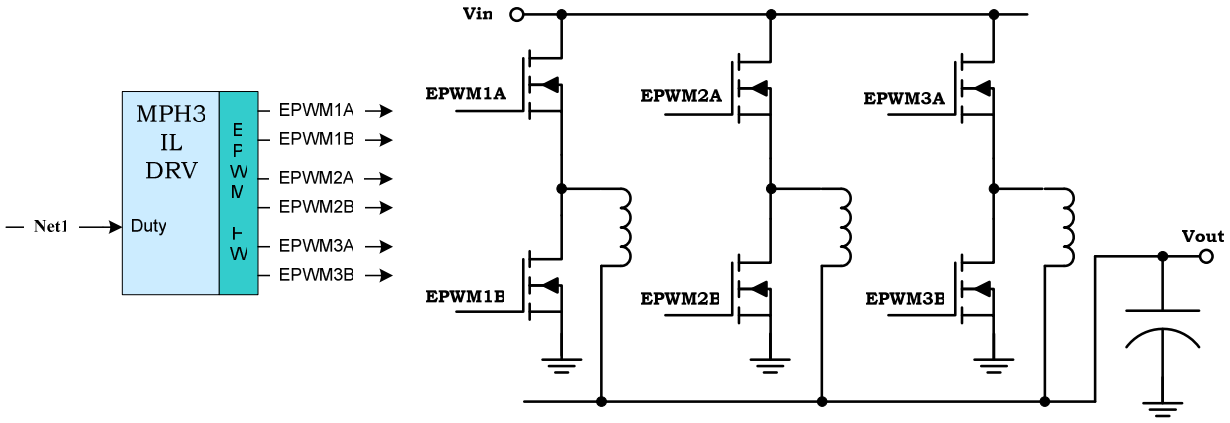


Figure 2. High resolution buck converter

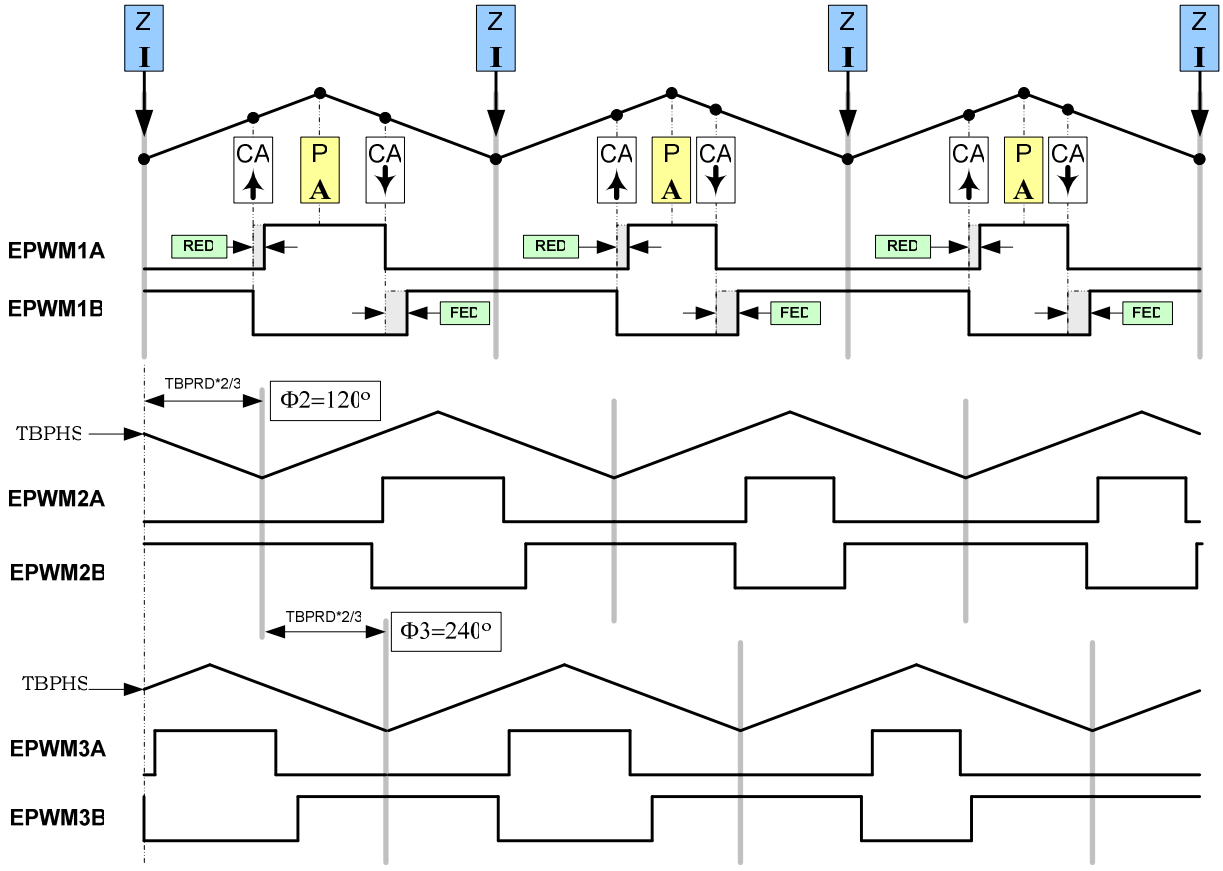


Figure 3.PWM generation with the F280x EPWM module.

MPIL_DRV

(symmetrical UpDown count)

HSPCLK =	**100**	1.00E+06	MHz	TBCLK = Sysclk * Pre-scale
MPH3IL Freq =	**100**	1.00E+03	KHz	

Duty cycle calculation table

		MPIL_duty		Period count (dec)	Period count (hex)
per unit	Duty(%)	Q15 (Dec)	Q15 (Hex)	500	01F4
				Compare Count (dec)	Compare Count (hex)
1.00	100%	32767	7FFF	500	01F4
0.90	90%	29490	7332	450	01C2
0.80	80%	26214	6665	400	0190
0.70	70%	22937	5998	350	015E
0.60	60%	19660	4CCC	300	012C
0.50	50%	16384	3FFF	250	00FA
0.40	40%	13107	3332	200	00C8
0.30	30%	9830	2666	150	0096
0.20	20%	6553	1999	100	0064
0.10	10%	3277	0CCC	50	0032
0.00	0%	0	0000	0	0000
-0.10	0%	62259	F333	*Not Defined*	
-0.20	0%	58982	E666		
-0.30	0%	55706	D999		
-0.40	0%	52429	CCCC		
-0.50	0%	49152	C000		
-0.60	0%	45875	B333		
-0.70	0%	42598	A666		
-0.70	0%	42598	A666		
-0.90	0%	36045	8CCC		
-1.00	0%	32768	8000		

Note: Negative duty cycle not defined.