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 High resolution PWM signals to control a Buck converter using only a single EPWMx module. This module forms the interface between the control software and the device PWM pins. It provides duty cycle control from 0-100%, and is geared toward application using high frequency PWM (200kHz – 2MHz).

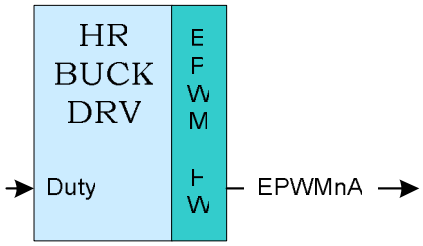

Figure 1. High resolution buck converter, PWM driver module

2 Module Properties

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

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

Table 1. HRBUCK_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. HRBUCK_DRV module components

The HRBUCK_DRV module has the following miscellaneous properties:

Property name	Property value
Multiple instance support	Yes (limited by number of physical EPWM modules on a given device).
Reentrant	No
Accessible from ‘C’ environment	Yes
Full configuration from ‘C’ environment	Yes
Input / Output connection	Pointer to signal net.

Table 3. HRBUCK_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. HRBUCK_DRV module component files

3 Module Input and Output Definitions

3.1 Module Inputs

Input name	Description	Format	Range
Duty	Duty cycle control (0-100%)	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.

[†] 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 HrBuckDrvCnf

Function Name:	HrBuckDrvCnf
Prototype:	void HrBuckDrvCnf(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 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.

Note: Only EPWM 1-4 allow for high resolution operation on the TMS320F2801, TMS320F2806, and TMS320F2808. EPWM5-6 will operate with standard resolution (1/clock cycle).

- Period**:** Specifies the PWM period in cycles, corresponding to the high speed peripheral clock.
  - **Valid Range:** 1 to 32767 TBCLK cycles.

$$PWM\,Frequency = \frac{HSPCLK(High\,Speed\,Per\,CLK)}{Period}$$

Example: Call the HrBuckDrvCnf function to initalize EPWM1 module.

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

HrBuckDrvCnfV2

Function Name: HrBuckDrvCnfV2

Prototype: void HrBuckDrvCnfV2 (int16 nEPwmModule,
int16 period,
int16 mode,
int16 phase);

Return value: None.

Preconditions: The following preconditions must be satisfied:

The appropriate EPWM module clock must be enabled in the PCLKCR1 register.

The HrBuckDrvCnfV2 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. This function allows the HRBUCK driver to be configured with additional flexibility.

- **nEPwmModule:** Specifies which EPWM module is intialized.
  - **Valid Range:** 1-6, corresponding to EPWM1-6.
- **Period:** Specifies the PWM period in cycles, corresponding to the high speed peripheral clock.
  - **Valid Range:** 1 to 32767.
- **Mode**: Specifies whether the EPWM module is configured as a master or as a slave.
  - **1**: EPWM module is configured as a master.
  - **0**: EPWM module is configured as a slave.
- **Phase**: Phase offset from the master module, applicable only if the module is in slave mode.

Example: Call the HrBuckDrvCnfV2 function to initalize EPWM1 module as a master, and EPWM2 as a slave with zero phase.

```
//-----------------------------------------------------------------
// ePWM1 target, 1000KHz PWM, master mode
// (100 clock period with a 100MHz High speed peripheral clock
//-----------------------------------------------------------------
HrBuckDrvCnfV2(1, 100, 1, 0); // ePWM1 Master
HrBuckDrvCnfV2(2, 100, 0, 0); // ePWM2 Slave
```

4.2 HRBUCK_DRV_INIT

Function Name: HRBUCK_DRV_INIT

Prototype: HRBUCK_DRV_INIT nEPwmModule

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.

Example: Call the HRBUCK_DRV_INIT to initalize EPWM1 module.

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

4.3 HRBUCK_DRV

Function Name: HRBUCK_DRV

Prototype: HRBUCK_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 HRBUCK_DRV_INIT 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.

Example: Call the HRBUCK_DRV in an assembler ISR

```
;---------------------------------------------------------
; Runtime interrupt service routine
;---------------------------------------------------------
_ISR_Run:   CONTEXT_SAVE            ;call macro
            HRBUCK_DRV 1
;---------------------------------------------------------
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:

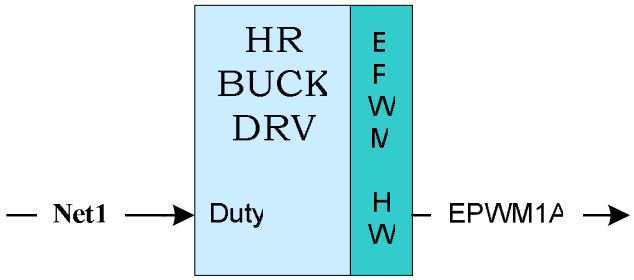

Figure 2. Connecting the high resolution buck converter

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

```
     HrBuckDrvCnf(1,200);
     HrBuckDrvCnf(2,200);
```

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

```
; Instantiate the init macro
     HRBUCK_DRV_INIT     1
     HRBUCK_DRV_INIT     2
```

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

```
; “call” the main macro
     HRBUCK_DRV      1
     HRBUCK_DRV      2
```

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

```
//Note: Net1, Net2 can be simply a global integer variable
int16     Net1, Net2;
```

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

```
// HRBUCK_DRV terminal pointers, external references
extern int16     *HRBUCK_Duty1, *HRBUCK_Duty2;
```

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

```
// HRBUCK_DRV connections
     HRBUCK_Duty1 = &Net1;
     HRBUCK_Duty2 = &Net2;

// 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

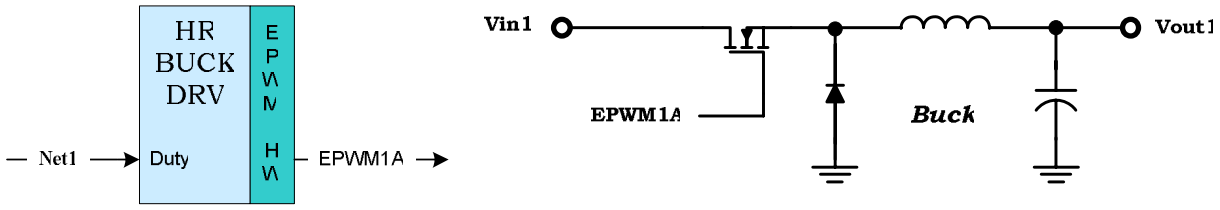

Figure 3. High resolution buck converter

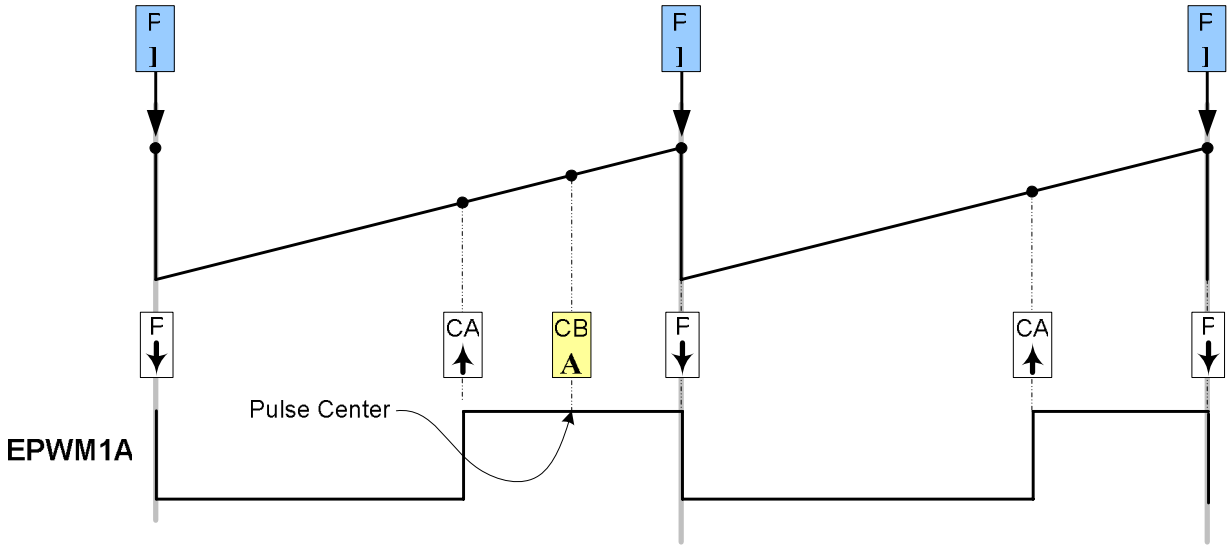

Figure 4.PWM generation with the F280x EPWM module.

HRBUCK_DRV

(Asymmetrical - Up count)

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

Duty cycle calculation table

		MPH3IL_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 and will give 0% duty.