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 buck converter using only a single EPWM module. This module forms the interface between the control software and the device PWM pins.

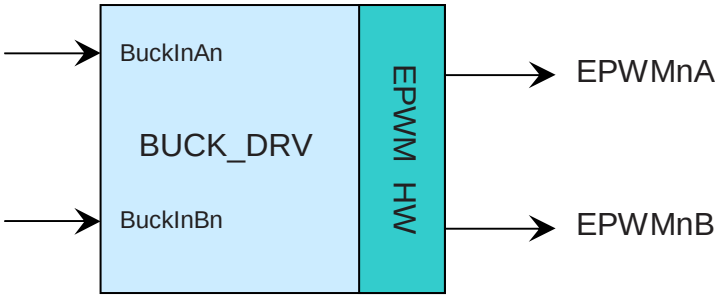

Figure 1. Buck converter, PWM driver module

2 Module Properties

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

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

Table 1. BUCK_DRV module dependencies

The BUCK_DRV module has the following components:

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

Table 2. BUCK_DRV module components

The BUCK_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. BUCK_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. BUCK_DRV module component files

3 Module Input and Output Definitions

3.1 Module Inputs

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

[†] 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 BuckDrvCnf

Function Name:	BuckDrvCnf
Prototype:	void BuckDrvCnf(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 BuckDrvCnf 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.
- Period**:** Specifies the PWM period in cycles, corresponding to the high speed peripheral clock.
  - **Valid Range:** 1 to 32767.

Example: Call the BuckDrvCnf function to initalize EPWM1 module in standard PWM resolution mode.

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

4.2 BUCK_DRV_INIT

Function Name: BUCK_DRV_INIT

Prototype: BUCK_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 BUCK_DRV_INIT to initalize EPWM1 module.

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

4.3 BUCK_DRV

Function Name: BUCK_DRV

Prototype: BUCK_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 BUCK_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 BUCK_DRV in an assembler ISR

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

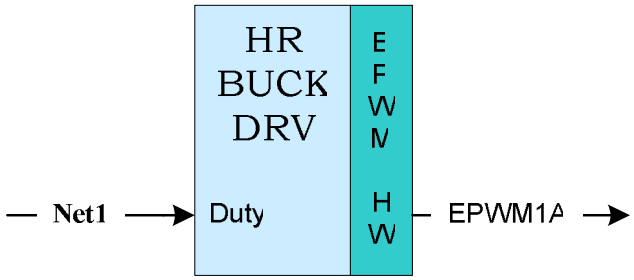

Figure 2. Connecting the high resolution buck converter

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

```
     BuckDrvCnf(,,,);
```

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

```
; Instantiate the init macro
     BUCK_DRV_INIT       1
```

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

```
; “call” the main macro
     BUCK_DRV       1
```

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

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

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

```
// BUCK_DRV terminal pointers, external references
extern int16     *BUCK_InAn, *BUCK_InBn;
```

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

```
// BUCK_DRV connections
     BUCK_InAn = &Net1, &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

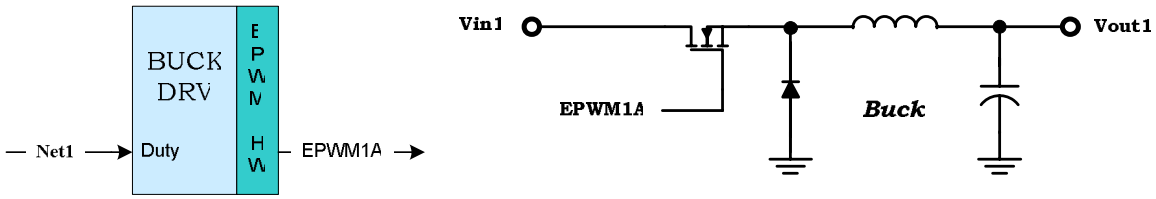

Figure 3. Buck converter driven by EPWM module

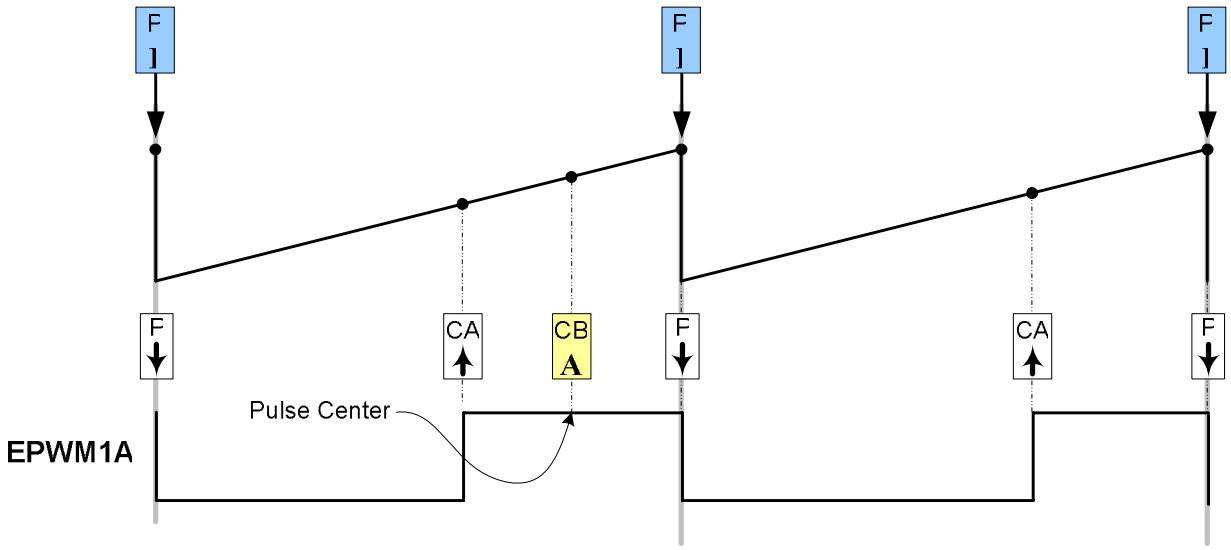

Figure 4.PWM generation with the F280x EPWM module.