Texas Instruments – C2000 DSP & Analog SPM groups

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 ........................................................................................................................ 3
4 Module API Description.................................................................................................................. 4
   4.1 BuckDrvCnf .............................................................................................................................. 4
   4.2 BUCK_DRV_INIT .................................................................................................................. 5
   4.3 BUCK_DRV ............................................................................................................................ 6
5 Usage Example............................................................................................................................ 7
6 Detailed description....................................................................................................................... 8

Table of Figures

Figure 1. Buck converter, PWM driver module ................................................................................. 2
Figure 2. Connecting the high resolution buck converter ................................................................. 7
Figure 3. High resolution buck converter ..................................................................................... 8
Figure 4. PWM generation with the F280x EPWM module ........................................................... 8

Index of Tables

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

![Buck converter, PWM driver module](image)

2 Module Properties

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

<table>
<thead>
<tr>
<th>Module</th>
<th>Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU dependency</td>
<td>C28x</td>
</tr>
<tr>
<td>Device dependency</td>
<td>x2801 / x2806 / x2808 members only</td>
</tr>
</tbody>
</table>

*Table 1. BUCK_DRV module dependencies*

The BUCK_DRV module has the following components:

<table>
<thead>
<tr>
<th>Component</th>
<th>Present</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-based initialization</td>
<td>Yes</td>
</tr>
<tr>
<td>ASM interrupt initialization</td>
<td>Yes</td>
</tr>
<tr>
<td>ASM runtime macro</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*Table 2. BUCK_DRV module components*

The BUCK_DRV module has the following miscellaneous properties:

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple instance support</td>
<td>No</td>
</tr>
<tr>
<td>Reentrant</td>
<td>No</td>
</tr>
<tr>
<td>Accessible from ‘C’ environment</td>
<td>Yes</td>
</tr>
<tr>
<td>Full configuration from ‘C’</td>
<td>Yes</td>
</tr>
</tbody>
</table>
3 Module Input and Output Definitions

3.1 Module Inputs

<table>
<thead>
<tr>
<th>Input name</th>
<th>Description</th>
<th>Format</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>BuckInAn</td>
<td>Duty cycle control</td>
<td>Pointer to 16-bit fixed point input data</td>
<td>Q15: [0, 1] or [0, 32767]</td>
</tr>
<tr>
<td>BuckInBn</td>
<td>Duty cycle control</td>
<td>Pointer to 16-bit fixed point input data</td>
<td>Q15: [0, 1] or [0, 32767]</td>
</tr>
</tbody>
</table>

3.2 Module Outputs

<table>
<thead>
<tr>
<th>Output name</th>
<th>Description</th>
<th>Format</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPWMnA</td>
<td>F280x/C280x PWM output pin</td>
<td>Pulse width modulated output.</td>
<td>See device datasheet for electrical specifications.</td>
</tr>
<tr>
<td>EPWMnB</td>
<td>F280x/C280x PWM output pin</td>
<td>Pulse width modulated output.</td>
<td>See device datasheet for electrical specifications.</td>
</tr>
</tbody>
</table>

† 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

<table>
<thead>
<tr>
<th>Function Name:</th>
<th>BuckDrvCnf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prototype:</td>
<td>void BuckDrvCnf(int16 nEPwmModule, int16 Period);</td>
</tr>
<tr>
<td>Return value:</td>
<td>None.</td>
</tr>
<tr>
<td>Preconditions:</td>
<td>The following preconditions must be satisfied:</td>
</tr>
<tr>
<td></td>
<td>The appropriate EPWM module clock must be enabled in the PCLKCR1 register.</td>
</tr>
</tbody>
</table>

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 initialized.
  - **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 initialize EPWM1 module in standard PWM resolution mode.

```c
// 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 initialized.
  - Valid Range: 1-6, corresponding to EPWM1-6.

Example: Call the BUCK_DRV_INIT to initialize 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 initialized.
  - Valid Range: 1-6, corresponding to EPWM1-6.

Example: Call the BUCK_DRV in an assembler ISR

```plaintext
; 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](image)

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

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

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

```assembly
; Instantiate the init macro
BUCK_DRV_INIT
```

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

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

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

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

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

```c
extern int16   *BUCK_InAn, *BUCK_InBn;
```

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

```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.