Module Document: PFC_2PHIL

Texas Instruments – C2000 DSP System Applications Group

Contents

Figures

Tables

1 PFC_2PHIL Module Documentation

This module controls the EPWM1 and 2 generators to drive a 2 phase interleaved PFC stage. This module forms the interface between the control software and the device PWM pins.

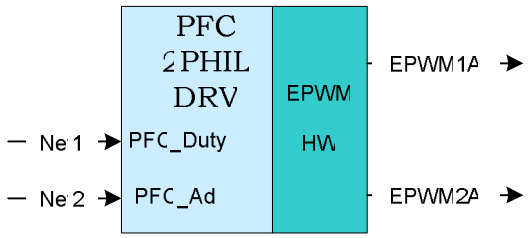

Figure 1. Two phase interleaved duty cycle Driver

1.1 Module Properties

This section describes module properties, such as compatible devices, components, invocation etc. The PFC2PHIL_DRV module has the following dependencies:

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

Table 1. PFC2PHIL_DRV module dependencies

The PFC2PHIL_DRV module has the following components:

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

Table 2. PFC2PHIL_DRV module components

The PFC2PHIL_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. PFC2PHIL_DRV module miscellaneous properties

Component	Files
Macro source:	C:\tidcs\DPS_C280x\vXYZ[1]\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. PFC2PHIL_DRV module component files

1.2 Module Input Definitions

Input name	Description	Data Format	Range
PfcDuty	Duty cycle control	Q15	Q15: [-1, 1) or [-32768, 32767]
PfcAdj	PFC current share adjustment** **In the current version this must be always zero**	Q15	Q15: [-1, 1) or [-32768, 32767]

Table 5. PFC2PHIL_DRV module inputs

1.3 Module Output Definitions

Output name	Description	Data Format	Output data Range
EPWM1A	PWM pin	PWM output	0-100%

Table 6. PFC2PHIL_DRV module outputs

[1] 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.

1.4 Module API Description: Pfc2phildrvCnf

This module has three executable code components, as described in **Error! Reference source not found.**. Each of these components is described in this section.

Function Name: Pfc2phildrvCnf

Prototype: void Pfc2phildrvCnf(nEPWMChannel, nPeriod);

Return value: None.

Preconditions: The following preconditions must be satisfied:

The EPWM modules' clocks must be enabled in the PCLKCR0 register.

The Pfc2phildrvCnf function is called from the C environment, and performs driver configuration, including configuration of ADC registers.

- **nEPWMChannel:** First Channel to be used for EPWM output.
  - **Valid Range:** 0-4, depending on the specific F280x device chosen.
- **nPeriod**: Specifies the PWM period in clock cycles.

Example: Call the ADC5CONTCnf function to initialize EPWM1 module.

```
//-----------------------------------------------------------------
// Set up 2Phase Interleaved PWM output on EPWM1 and EPWM2 with a
// 1000
//-----------------------------------------------------------------
       Pfc2phildrvCnf(1, 1000);
```

1.5 PFC2PHIL_DRV_INIT

Function Name: PFC2PHIL_DRV_INIT

Prototype: PFC2PHIL_DRV_INIT

Return value: None.

Preconditions: The following preconditions must be satisfied:

The ADC module clock must be enabled in the PCLKCR0 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.

Example: **Call the PFC2PHIL_DRV_INIT to initialize the PFC2PHIL_DRV module**.

```
;-------------------------------------------------------
; ISR Initialisation
;-------------------------------------------------------
_ISR_Init:          PFC2PHIL_DRV_INIT
                    LRETR
```

1.6 PFC2PHIL_DRV

Function Name: PFC2PHIL_DRV

Prototype: PFC2PHIL_DRV

Return value: None.

Preconditions: The following preconditions must be satisfied:

- The module clock must be enabled in the PCLKCRx register.
- C language init routine must be called.
- The ISR initialization macro PFC2PHIL_DRV_INIT must be instanced and executed in an assembler initialization routine.

This function is the assembler run time macro, and this creates code that forms a bridge between software and the EPWM register hardware output. The output is written to the duty cycle control registers, which control the PWM generation.

Example: Call the PFC2PHIL_DRV in an assembler ISR

```
;---------------------------------------------------------
; Runtime interrupt service routine
;---------------------------------------------------------
_ISR_Run:           CONTEXT_SAVE              ;call macro
                    PFC2PHIL_DRV
                    . . . .                                  ; other code
;---------------------------------------------------------
EXIT_ISR: ;Interrupt management before exit
;---------------------------------------------------------
                    MOVW        DP,#ETCLR1>>6
                    MOV         @ETCLR1,#0x01; Clear EPWM1 Int flag

;---------------------------------------------------------
; Restore context & return
;---------------------------------------------------------
                    CONTEXT_REST
                    IRET
```

1.7 Usage Example:

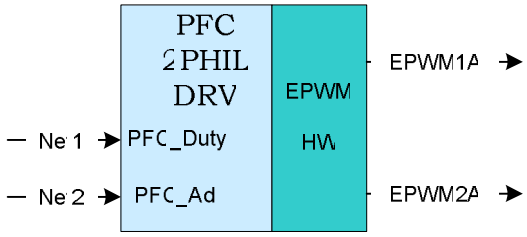

Figure 2. Connecting the one channel PFC2PHIL driver

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

```
       Pfc2philDrvCnf(1, 1000);   // ePWM 1 & 2, Freq = 100 KHz
```

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

```
; Instantiate the init macro
       PFC2PHIL_DRV_INIT
```

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

```
; “call” the main macro
       PFC2PHIL_DRV
```

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

```
int16   PfcDuty;
; Note this is just a global integer variable
```

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

```
// PFC2PHIL_DRV terminal pointers, external references
extern int16        *PFC2PHIL_Duty, *PFC2PHIL_Adj;
```

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

```
// Connect the PFC PWM generator block:
       PFC2PHIL_Duty = &PfcDuty;              // Point to the duty cycle net
       PFC2PHIL_Adj  = &PfcShareAdj;          // Point to the adj factor net
```