HR_PWM_DAC_DRV

Texas Instruments – C2000 DSP System Applications Group

Table of contents

Table of Figures

Index of Tables

1 Overview

This software module directly controls the EPWM peripherals on the 280x devices to simulate a digital to analog converter, in conjunction with an output filter. It generates appropriate high resolution PWM signals which, when filtered, represent the input values pointed to by the input pointer. This is a useful tool for output of analog values and for visualization.

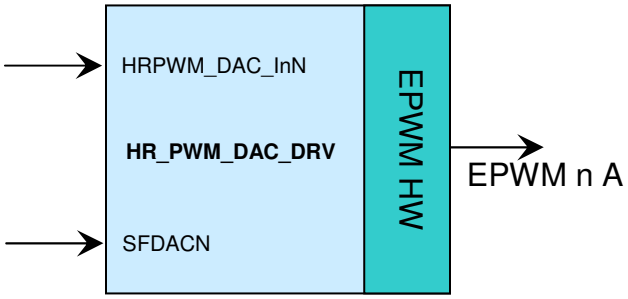

Figure 1. High resolution PWM based DAC

2 Module Properties

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

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

Table 1. HR_PWM_DAC_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. HR_PWM_DAC_DRV module components

The HR_PWM_DAC_DRV module has the following miscellaneous properties:

Property name	Property value
Multiple instance support	Yes
Reentrant	No
Accessible from 'C' environment	Yes
Full configuration from 'C' environment	Yes
Input / Output connection	Pointer to signal net.

Table 3. HR_PWM_DAC_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. HR_PWM_DAC_DRV module component files

3 Module Input and Output Definitions

3.1 Module inputs

Input name	Description	Format	Range
HRPWM_DAC_InN	Duty cycle control	Pointer to 16-bit fixed point input	Q15: [0, 1] or [0, 32767]
SFDACN	Scale factor input for HR PWM MEP unit	Pointer to 16-bit fixed point input	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

[†] 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.

specifications.

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 HrPwmDacDrvCnf

Function Name:	HrPwmDacDrvCnf
Prototype:	void HrPwmDacDrvCnf(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 HrPwmDacDrvCnf 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. If EPWM modules 5 or 6 are selected, the high resolution function is not applicable.
- Period**:** Specifies the PWM period in cycles, corresponding to the high speed peripheral clock.
  - **Valid Range:** 1 to 32767.

Example: Call the HrPwmDacDrvCnf function to use the EPWM1 module as a DAC.

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

4.2 HR_PWM_DAC_INIT

Function Name: HRPWM_DAC_DRV_INIT

Prototype: HRPWM_DAC_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 HRPWM_DAC_DRV_INIT to initalize EPWM1 module.

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

4.3 HRPWM_DAC_DRV

Function Name: HRPWM_DAC_DRV

Prototype: HRPWM_DAC_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 HRPWM_DAC_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 HRPWM_DAC_DRV in an assembler ISR

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

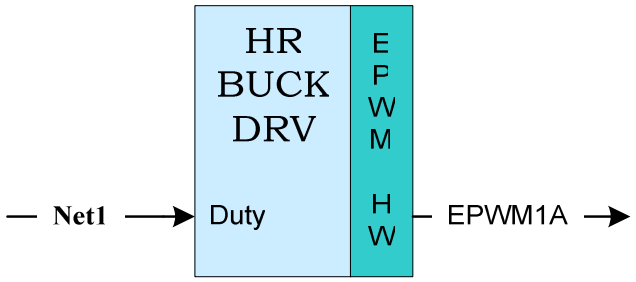

Figure 2. Connecting the high resolution buck converter

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

```
HrPwmDacDrvCnf(1, 100);
```

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

```
; Instantiate the init macro

          HRPWM_DAC_DRV_INIT  1
```

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

```
; “call” the main macro

          HRPWM_DAC_DRV       1
```

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

```
int16     Net1;

; Note Net1 can be simply a global integer variable
```

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

```
// HRBUCK_DRV terminal pointers, external references

extern int16     *PWMDAC_InAn, *PWMDAC_InAn, ;
```

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

```
// HRBUCK_DRV connections

     PWMDAC_InAn = &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.
```