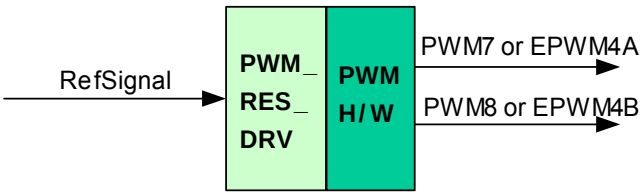
PWM_RES_DRV	*PWM Driver for Excitation Signal of Resolver Sensor*

Description

This module uses the duty ratio information and calculates the compare value for generating two PWM outputs (PWM 7 and 8 for x281x or EPWM4A and EPWM4B for x280x) for excitation signal of resolver sensor. The compare value is used in the full compare unit in 281x event manager (EVB). This also allows PWM period modulation.



Availability

This 16-bit module is available in one interface format:

1) The C interface version RefSignal

Module Properties

Type: Target Dependent, Application Independent

Target Devices: x281x or x280x

C Version File Names: f281xpwm_res.c, f281xpwm_res.h (for x281x)
f280xpwm_res.c, f280xpwm_res.h (for x280x)

IQmath library files for C: N/A

Item	C version	Comments
Code Size□ (x281x/x280x)	47/109 words	
Data RAM	0 words•	
xDAIS ready	No	
XDAIS component	No	IALG layer not implemented
Multiple instances	Yes	
Reentrancy	Yes	

• Each pre-initialized RESOLVER_PWM structure consumes 6 words in the data memory

□ Code size mentioned here is the size of the ***init()*** and ***update()*** functions

C Interface

Object Definition

The structure of RESOLVER_PWM object is defined by following structure definition

```
typedef struct {   int16 RefSignal;         // Input: Excitation reference signal (Q15)
                   Uint16 PeriodMax;     // Parameter: PWM Half-Period in CPU clock cycles (Q0)
                   void (*init)();           // Pointer to the init function
                   void (*update)();     // Pointer to the update function
               } RESOLVER_PWM;

typedef RESOLVER_PWM * RESOLVER_PWM_handle;
```

Item	Name	Description	Format	Range(Hex)
Input	RefSignal	PWM duty cycle ratio	Q15	8000-7FFF
Outputs	PWM7-8 (x281x) EPWM4A-4B (x280x)	Output signals from the 2 PWM pins in EVB on the x2812eZdsp or EPWM4A/4B pins on the x2808eZdsp.	N/A	0-3.3 V
PWM_RES parameter	PeriodMax	PWM Half-Period in CPU clock cycles	Q0	8000-7FFF

Special Constants and Data types

RESOLVER_PWM
The module definition is created as a data type. This makes it convenient to instance an interface to the RESOLVER_PWM driver. To create multiple instances of the module simply declare variables of type RESOLVER_PWM.

RESOLVER_PWM_handle
User defined Data type of pointer to RESOLVER_PWM module

RESOLVER_PWM_DEFAULTS
Structure symbolic constant to initialize RESOLVER_PWM module. This provides the initial values to the terminal variables as well as method pointers.

Methods

void F281X_EV2_Resolver_PWM_Init(RESOLVER_PWM *);
void F281X_EV2_Resolver_PWM_Update(RESOLVER_PWM *);

void F280X_Resolver_PWM_Init(RESOLVER_PWM *);
void F280X_Resolver_PWM_Update(RESOLVER_PWM *);

This default definition of the object implements two methods – the initialization and the runtime compute function for RESOLVER_PWM generation. This is implemented by means of a function pointer, and the initializer sets this to F281X_EV2_Resolver_PWM_Init and F281X_EV2_Resolver_PWM_Update functions for x281x or F280X_Resolver_PWM_Init and F280X_Resolver_PWM_Update functions for x280x. The argument to this function is the address of the RESOLVER_PWM object.

Module Usage

Instantiation
The following example instances one RESOLVER_PWM object
RESOLVER_PWM pwm_res1;

Initialization
To Instance pre-initialized objects
RESOLVER_PWM pwm_res1 = RESOLVER_PWM_DEFAULTS;

Invoking the computation function
pwm_res1.init(&pwm_res1);
pwm_res1.update(&pwm_res1);

Example

The following pseudo code provides the information about the module usage.

```
main()
{

        pwm_res1.PeriodMax = 469;      // PWM frequency = 160 kHz, clock = 150 MHz
        pwm_res1.init(&pwm_res1);      // Call init function for pwm_res1
}

void interrupt periodic_interrupt_isr()
{
        // sig_ref is the sinusoidal excitation signal with 2-10 kHz (defined as floating point)
        pwm_res1.RefSignal = _IQtoIQ15(sig_ref);
        pwm_res1.update(&pwm_res1);                  // Call update function for pwm_res1
}
```