# PWM Driver User Guide V1.00.01

# 1. Constant Definition

| Name | Value | Description |
|---|---|---|
| DRVPWM_TIMER0 | 0x00 | PWM Timer 0 |
| DRVPWM_TIMER1 | 0x01 | PWM Timer 1 |
| DRVPWM_CAP0 | 0x10 | PWM Capture 0 |
| DRVPWM_CAP1 | 0x11 | PWM Capture 1 |
| DRVPWM_CAP_ALL_INT | 3 | PWM Capture Rising and Falling Interrupt |
| DRVPWM_CAP_RISING_INT | 1 | PWM Capture Rising Interrupt |
| DRVPWM_CAP_FALLING_INT | 2 | PWM Capture Falling Interrupt |
| DRVPWM_CAP_RISING_FLAG | 6 | Capture rising interrupt flag |
| DRVPWM_CAP_FALLING_FLAG | 7 | Capture falling interrupt flag |
| DRVPWM_CLOCK_DIV_1 | 4 | Input clock divided by 1 |
| DRVPWM_CLOCK_DIV_2 | 0 | Input clock divided by 2 |
| DRVPWM_CLOCK_DIV_4 | 1 | Input clock divided by 4 |
| DRVPWM_CLOCK_DIV_8 | 2 | Input clock divided by 8 |
| DRVPWM_CLOCK_DIV_16 | 3 | Input clock divided by 16 |
| DRVPWM_TOGGLE_MODE | 1 | PWM Timer Toggle mode |
| DRVPWM_ONE_SHOT_MODE | 0 | PWM Timer One-shot mode |
| DRVPWM_10K | 0 | clock source from internal 10 kHz oscillator |
| DRVPWM_32K | 1 | clock source from external 32kHz crystal clock |
| DRVPWM_HCLKK | 2 | clock source from HCLK |
| DRVPWM_48M | 3 | clock source from internal OSC48M oscillator clock |

# 2. Functions

## *DrvPWM_IsTimerEnabled*

**Prototype**

int32_t DrvPWM_IsTimerEnabled(uint8_t u8Timer);

**Description**

This function is used to get PWM specified timer enable/disable state

**Parameter**

**u8Timer [in]**

Specify the timer.

DRVPWM_TIMER0: PWM timer 0.

DRVPWM_TIMER1: PWM timer 1.

**Include**

Driver/DrvPWM.h

**Return Value**

1: The specified timer is enabled.

0: The specified timer is disabled.

## *DrvPWM_SetTimerCounter*

**Prototype**

void DrvPWM_SetTimerCounter(uint8_t u8Timer, uint16_t u16Counter);

**Description**

This function is used to set the PWM specified timer counter.

**Parameter**

**u8Timer [in]**

Specify the timer.

DRVPWM_TIMER0: PWM timer 0.

DRVPWM_TIMER1: PWM timer 1.

**u16Counter [in]**

Specify the timer value. (0~65535)

**Include**

Driver/DrvPWM.h

**Return Value**

None

**Note** If the counter is set to 0, the timer will stop.

### *DrvPWM_GetTimerCounter*

**Prototype**

uint32_t DrvPWM_GetTimerCounter(uint8_t u8Timer);

**Description**

This function is used to get the PWM specified timer counter value

**Parameter**

**u8Timer [in]**

Specify the timer.

DRVPWM_TIMER0: PWM timer 0.

DRVPWM_TIMER1: PWM timer 1.

**Include**

Driver/DrvPWM.h

**Return Value**

The specified timer counter value.

### *DrvPWM_EnableInt*

**Prototype**

void DrvPWM_EnableInt(uint8_t u8Timer, uint8_t u8Int, PFN_DRVPWM_CALLBACK pfncallback);

**Description**

This function is used to enable the PWM timer/capture interrupt and install the call back function.

**Parameter**

**u8Timer [in]**

Specify the timer

DRVPWM_TIMER0: PWM timer 0.

DRVPWM_TIMER1: PWM timer 1.

or the capture.

DRVPWM_CAP0: PWM capture 0.

DRVPWM_CAP1: PWM capture 1.

**u8Int [in]**

Specify the capture interrupt type (The parameter is valid only when capture function)

DRVPWM_CAP_RISING_INT: The capture rising interrupt.

DRVPWM_CAP_FALLING_INT: The capture falling interrupt.

DRVPWM_CAP_ALL_INT: All capture interrupt.

**pfncallback [in]** The pointer of the callback function for specified timer / capture.

**Include**

Driver/DrvPWM.h

**Return Value**

None

### *DrvPWM_DisableInt*

**Prototype**

void DrvPWM_DisableInt(uint8_t u8Timer);

**Description**

This function is used to disable the PWM timer/capture interrupt.

**Parameter**

**u8Timer [in]**

Specify the timer

DRVPWM_TIMER0: PWM timer 0.

DRVPWM_TIMER1: PWM timer 1.

or the capture.

DRVPWM_CAP0: PWM capture 0.

DRVPWM_CAP1: PWM capture 1.

**Include**

Driver/DrvPWM.h

**Return Value**

None

### *DrvPWM_ClearInt*

**Prototype**

void DrvPWM_ClearInt(uint8_t u8Timer);

**Description**

This function is used to clear the PWM timer/capture interrupt.

**Parameter**

**u8Timer [in]**

Specify the timer

DRVPWM_TIMER0: PWM timer 0.

DRVPWM_TIMER1: PWM timer 1.

or the capture.

DRVPWM_CAP0: PWM capture 0.

DRVPWM_CAP1: PWM capture 1.

**Include**

Driver/DrvPWM.h

**Return Value**

None

## *DrvPWM_GetIntFlag*

**Prototype**

int32_t DrvPWM_GetIntFlag(uint8_t u8Timer);

**Description**

This function is used to get the PWM timer/capture interrupt flag

**Parameter**

**u8Timer [in]**

Specify the timer

DRVPWM_TIMER0: PWM timer 0.

DRVPWM_TIMER1: PWM timer 1.

or the capture.

DRVPWM_CAP0: PWM capture 0.

DRVPWM_CAP1: PWM capture 1.

**Include**

Driver/DrvPWM.h

**Return Value**

1: The specified interrupt occurs.

0: The specified interrupt doesn’t occur.

## *DrvPWM_GetRisingCounter*

**Prototype**

uint16_t DrvPWM_GetRisingCounter(uint8_t u8Capture);

**Description**

This function is used to get value which latches the counter when there is a rising transition.

**Parameter**

**u8Capture [in]**

Specify the capture.

DRVPWM_CAP0: PWM capture 0.

DRVPWM_CAP1: PWM capture 1.

**Include**

Driver/DrvPWM.h

**Return Value**

The value which latches the counter when there is a rising transition.

## *DrvPWM_GetFallingCounter*

**Prototype**

uint16_t DrvPWM_GetFallingCounter(uint8_t u8Capture);

**Description**

This function is used to get value which latches the counter when there‟s a falling transition.

**Parameter**

**u8Capture [in]**

Specify the capture.

DRVPWM_CAP0: PWM capture 0.

DRVPWM_CAP1: PWM capture 1.

**Include**

Driver/DrvPWM.h

**Return Value**

The value which latches the counter when there's a falling transition

## *DrvPWM_GetCaptureIntStatus*

**Prototype**

int32_t DrvPWM_GetCaptureIntStatus(uint8_t u8Capture, uint8_t u8IntType);

**Description**

Check if there's a rising / falling transition

**Parameter**

**u8Capture [in]**

Specify the capture.

DRVPWM_CAP0: PWM capture 0.

DRVPWM_CAP1: PWM capture 1.

**u8IntType [in]**

Specify the capture.

DRVPWM_CAP_RISING_FLAG: The capture rising interrupt flag.

DRVPWM_CAP_FALLING_FLAG: The capture falling interrupt flag.

**Include**

Driver/DrvPWM.h

**Return Value**

TRUE: The specified interrupt occurs.

FALSE: The specified interrupt doesn't occur.

## *DrvPWM_ClearCaptureIntStatus*

**Prototype**

void DrvPWM_ClearCaptureIntStatus(uint8_t u8Capture, uint8_t u8IntType);

**Description**

Clear the rising / falling transition interrupt flag

**Parameter**

**u8Capture [in]**

Specify the capture.

DRVPWM_CAP0: PWM capture 0.

DRVPWM_CAP1: PWM capture 1.

**u8IntType [in]**

Specify the capture.

DRVPWM_CAP_RISING_FLAG: The capture rising interrupt flag.

DRVPWM_CAP_FALLING_FLAG: The capture falling interrupt flag.

**Include**

Driver/DrvPWM.h

**Return Value**

None

## *DrvPWM_Open*

**Prototype**

void DrvPWM_Open(void);

**Description**

Enable PWM engine clock and reset PWM.

**Include**

Driver/DrvPWM.h

**Return Value**

None

## *DrvPWM_Close*

**Prototype**

void DrvPWM_Close(void);

**Description**

Disable PWM engine clock and the I/O enable

**Include**

Driver/DrvPWM.h

**Return Value**

None

## *DrvPWM_EnableDeadZone*

**Prototype**

void DrvPWM_EnableDeadZone(uint8_t u8Length, int32_t i32EnableDeadZone);

**Description**

This function is used to set the dead zone length and enable/disable Dead Zone function.

**Parameter**

**u8Length [in]**

Specify Dead Zone Length : 0~255.

**i32EnableDeadZone [in]**

Enable DeadZone (1) / Diasble DeadZone (0)

**Include**

Driver/DrvPWM.h

**Return Value**

None

## *DrvPWM_Enable*

**Prototype**

void DrvPWM_Enable(uint8_t u8Timer, int32_t i32Enable);

**Description**

This function is used to enable PWM timer / capture function

**Parameter**

**u8Timer [in]**

Specify the timer

DRVPWM_TIMER0: PWM timer 0.

DRVPWM_TIMER1: PWM timer 1.

or the capture.

DRVPWM_CAP0: PWM capture 0.

DRVPWM_CAP1: PWM capture 1.

**i32Enable [in]**

Enable (1) / Disable (0)

**Include**

Driver/DrvPWM.h

**Return Value**

None

## *DrvPWM_SetTimerClk*

**Prototype**

uint32_t DrvPWM_SetTimerClk(uint8_t u8Timer,
S_DRVPWM_TIME_DATA_T *sPt);

**Description**

This function is used to configure the frequency/pulse/mode/inverter function.

**Parameter**

**u8Timer [in]**

Specify the timer

DRVPWM_TIMER0: PWM timer 0.

DRVPWM_TIMER1: PWM timer 1.

or the capture.

DRVPWM_CAP0: PWM capture 0.

DRVPWM_CAP1: PWM capture 1.

***sPt [in]**

It includes the following parameter

***u8Frequency***: The timer/capture frequency

***u8HighPulseRatio***: High pulse ratio

***u8Mode***: DRVPWM_ONE_SHOT_MODE /DRVPWM_TOGGLE_MODE

***Inverter***: Inverter Enable (1) / Inverter Disable (0)

***u8ClockSelector***: Clock Selector

DRVPWM_CLOCK_DIV_1:

DRVPWM_CLOCK_DIV_2:

DRVPWM_CLOCK_DIV_4:

DRVPWM_CLOCK_DIV_8:

DRVPWM_CLOCK_DIV_16:

(The parameter takes effect when u8Frequency = 0)

***u8PreScale***: Prescale (2 ~ 256)

(The parameter takes effect when u8Frequency = 0)

***u32Duty***: Pulse duty

(The parameter takes effect when u8Frequency = 0 or u8Timer = DRVPWM_CAP0/DRVPWM_CAP1)

**Include**

Driver/DrvPWM.h

**Return Value**

The actual frequency.

**Note**

1. The function will set the frequency property automatically when user set a nonzero frequency value
2. When setting the frequency value to zero, user also can set frequency property (Clock selector/Prescale/Duty) by himself.
3. The function can set the proper frequency property (Clock selector/Prescale) for capture function and user needs to set the proper pulse duty by himself.

## *DrvPWM_SetTimerIO*

**Prototype**

void DrvPWM_SetTimerIO(uint8_t u8Timer, int32_t i32Enable);

**Description**

This function is used to enable/disable PWM timer/capture I/O function

**Parameter**

**u8Timer [in]**

Specify the timer

DRVPWM_TIMER0: PWM timer 0.

DRVPWM_TIMER1: PWM timer 1.

or the capture.

DRVPWM_CAP0: PWM capture 0.

DRVPWM_CAP1: PWM capture 1.

**i32Enable [in]**

Enable (1) / Disable (0)

**Include**

Driver/DrvPWM.h

**Return Value**

None

## *DrvPWM_SelectClockSource*

**Prototype**

void DrvPWM_SelectClockSource(uint8_t u8Timer, uint8_t u8ClockSourceSelector);

**Description**

This function is used to select PWM0/PWM1 engine clock source.

**Parameter**

**u8Timer [in]**

Specify the timer

DRVPWM_TIMER0/DRVPWM_TIMER1: PWM timer 0 or PWM timer 1.

**u8ClockSourceSelector [in]**

DRVPWM_10K/DRVPWM_32K/DRVPWM_HCLK/DRVPWM_48M

DRVPWM_10K: internal 10 KHz crystal clock

DRVPWM_32K: external 32 KHz crystal clock

DRVPWM_HCLK: HCLK

DRVPWM_48M: internal OSC48 MHz crystal clock

**Include**

Driver/DrvPWM.h

**Return Value**

None

## *DrvPWM_GetVersion*

**Prototype**

int32_t DrvPWM_GetVersion (void);

**Description**

Return the current version number of driver.

**Include**

Driver/DrvPWM.h

**Return Value**

Version number :

| 31:24 | 23:16 | 15:8 | 7:0 |
|---|---|---|---|
| 00000000 | MAJOR_NUM | MINOR_NUM | BUILD_NUM |

# 2. Revision History

| Version | Date | Description |
|---|---|---|
| 1.00.01 | Mar. 2011 | Preliminary PWM Driver User Guide of ISD9160 |
| | | |