

# **F2802x0 Peripheral Driver Library**

## **USER'S GUIDE**



---

# Copyright

Copyright © 2023 Texas Instruments Incorporated. All rights reserved. Other names and brands may be claimed as the property of others.

 Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this document.

Texas Instruments  
13905 University Boulevard  
Sugar Land, TX 77479  
<http://www.ti.com/c2000>



## Revision Information

This is version 3.05.00.00 of this document, last updated on Wed Jul 12 21:05:58 IST 2023.

# Table of Contents

<b>Copyright</b>	<b>2</b>
<b>Revision Information</b>	<b>2</b>
<b>1 Introduction</b>	<b>5</b>
<b>2 Programming Model</b>	<b>7</b>
<b>3 Analog to Digital Converter (ADC)</b>	<b>9</b>
3.1 ADC	9
<b>4 Capture (CAP)</b>	<b>15</b>
4.1 CAP	15
<b>5 Device Clocking (CLK)</b>	<b>27</b>
5.1 CLK	27
<b>6 Comparater (COMP)</b>	<b>45</b>
6.1 COMP	45
<b>7 Central Processing Unit (CPU)</b>	<b>51</b>
7.1 CPU	51
<b>8 Flash</b>	<b>59</b>
8.1 FLASH	59
<b>9 General Purpose Input/Output (GPIO)</b>	<b>69</b>
9.1 GPIO	69
<b>10 Oscillator (OSC)</b>	<b>81</b>
10.1 OSC	81
<b>11 Peripheral Interrupt Expansion Module (PIE)</b>	<b>87</b>
11.1 PIE	87
<b>12 Phase Locked Loop (PLL)</b>	<b>103</b>
12.1 PLL	103
<b>13 Pulse Width Modulator (PWM)</b>	<b>111</b>
13.1 PWM	111
<b>14 Power Control (PWR)</b>	<b>147</b>
14.1 PWR	147
<b>15 Serial Communications Interface (SCI)</b>	<b>153</b>
15.1 SCI	153
<b>16 Serial Peripheral Interface (SPI)</b>	<b>173</b>
16.1 SPI	173
<b>17 Timer</b>	<b>189</b>
17.1 TIMER	189
<b>18 Watchdog Timer</b>	<b>195</b>
18.1 WDOG	195
<b>IMPORTANT NOTICE</b>	<b>200</b>



# 1 Introduction

The Texas Instruments® C2000Ware® Peripheral Driver Library is a set of drivers for accessing the peripherals found on the Piccolo Entryline family of C2000 microcontrollers. While they are not drivers in the pure operating system sense (that is, they do not have a common interface and do not connect into a global device driver infrastructure), they do provide a mechanism that makes it easy to use the device's peripherals.

The capabilities and organization of the drivers are governed by the following design goals:

They are written entirely in C except where absolutely not possible.

They demonstrate how to use the peripheral in its common mode of operation.

They are easy to understand.

They are reasonably efficient in terms of memory and processor usage.

They are as self-contained as possible.

Where possible, computations that can be performed at compile time are done there instead of at run time.

Some consequences of these design goals are:

The drivers are not necessarily as efficient as they could be (from a code size and/or execution speed point of view). While the most efficient piece of code for operating a peripheral would be written in assembly and custom tailored to the specific requirements of the application, further size optimizations of the drivers would make them more difficult to understand.

The drivers do not support the full capabilities of the hardware. Some of the peripherals provide complex capabilities which cannot be utilized by the drivers in this library, though the existing code can be used as a reference upon which to add support for the additional capabilities.

For many applications, the drivers can be used as is. But in some cases, the drivers will have to be enhanced or rewritten in order to meet the functionality, memory, or processing requirements of the application. If so, the existing driver can be used as a reference on how to operate the peripheral.

This device support release also contains the traditional peripheral register header files and associated software. Please see chapter 2 of this guide for more information on this software. For future development we recommend the use of this driver infrastructure instead of the traditional header files.

## Source Code Overview

The following is an overview of the organization of the peripheral driver library source code.

`common/source/`

This directory contains the source code for the drivers.

`common/include/`

This directory contains the header files for the drivers. These headers define not only values used in the drivers and calls to drivers but also define the register structure of each peripheral.

`common/ccs/`

This directory contains the CCS project for the driver library. The driver library can be rebuilt if modifications are made by importing and building this project.

## 2 Programming Model

Introduction .....	??
Direct Register Access Model .....	??
Software Driver Model .....	??
Combining The Models .....	??

The peripheral driver library provides support for two programming models: the direct register access model and the software driver model. Each model can be used independently or combined, based on the needs of the application or the programming environment desired by the developer.

Each programming model has advantages and disadvantages. Use of the direct register access model generally results in smaller and more efficient code than using the software driver model. However, the direct register access model requires detailed knowledge of the operation of each register and bit field, as well as their interactions and any sequencing required for proper operation of the peripheral; the developer is insulated from these details by the software driver model, generally requiring less time to develop applications. In the direct register access model, the peripherals are programmed by the application by writing values directly into the peripheral's registers. Every register is defined in a peripheral's corresponding header file contained in `f2802x0/headers/include`. These headers define the locations of each register relative to the other registers in a peripheral as well as the bit fields within each register. All of this is implemented using structures. The header files only define these structures; they do not declare them. To declare the structures a C source file must be included in each project `f2802x0/headers/source/F2802x0_GlobalVariableDefs.c`. This file declares each structure (and in some cases multiple instances when there are multiple instances of a peripheral) as well as declaring and associating each structure with a code section for the linker. The final piece of the puzzle is a special linker command file that associates each section defined in `F2802x0_GlobalVariableDefs.c` with the physical memory of the device. There are two versions of this linker command file, one for use with SYS/BIOS and one for use without, but both can be found in `f2802x0/headers/cmd`. One of this linker command files as well as your normal application linker command file should be used to link your project.

Applications that wish to use the direct register access model should define the root of the `f2802x0` directory to be an include path and include the file `DSP28x_Project.h` in each source file where register accesses are made.

In practice accessing peripheral registers and bitfields is extremely easy. Below are a few examples: `GpioCtrlRegs.GPAPUD.bit.GPIO0 = 0;` `GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 1;`

In the software driver model, the API provided by the peripheral driver library is used by applications to control the peripherals. Because these drivers provide complete control of the peripherals in their normal mode of operation, it is possible to write an entire application without direct access to the hardware. This method provides for rapid development of the application without requiring detailed knowledge of how to program the peripherals.

Before a driver for a peripheral can be used that peripheral's driver header file should be included and a handle to that peripheral initialized: `GPIO_Handle myGpio; myGpio = GPIO_init((void*)GPIO_BASE_ADDR, sizeof(GPIO_Obj));`

In subsequent calls to that peripheral's driver the initialized handle as well as any parameters are passed to the function: `GPIO_setPullUp(myGpio, GPIO_Number0, GPIO_PullUp_Enable); GPIO_setMode(myGpio, GPIO_Number0, GPIO_`

As you can see from the above sample, using the driver library makes one's code substantially more readable.

In addition to including the appropriate header files for the drivers you are using, the project

should also have `driverlib.lib` linked into it. A CCS project as well as the lib file can be found in `f2802x0/common/ccs`. The driver library also contains some basic helper functions that many projects will use as well as the direct register access model source file `F2802x0_GlobalVariableDefs.c` which allows you to use the header files without having to directly compile this source file into your project.

The drivers in the peripheral driver library are described in the remaining chapters in this document. They combine to form the software driver model. The direct register access model and software driver model can be used together in a single application, allowing the most appropriate model to be applied as needed to any particular situation within the application. For example, the software driver model can be used to configure the peripherals (because this is not performance critical) and the direct register access model can be used for operation of the peripheral (which may be more performance critical). Or, the software driver model can be used for peripherals that are not performance critical (such as a UART used for data logging) and the direct register access model for performance critical peripherals.

To use both models interchangeably in your application:

Link `driverlib.lib` into your application

Include `DSP28x_Project.h` in files you wish to use the direct register access model

Add `C2000Ware_version/device_support/f2802x0/` to your projects include path (Right click on project, Build Properties, Include Path)

Include driver header files from `f2802x0/common/include` in any source file that makes calls to that driver.



## 3 Analog to Digital Converter (ADC)

[Introduction](#) .....??

[API Functions](#) 9 The Analog to Digital Converter (ADC) APIs provide a set of functions for accessing the Piccolo F2802x0 ADC modules.

This driver is contained in `f2802x0/common/source/adc.c`, with `f2802x0/common/include/adc.h` containing the API definitions for use by applications.

### 3.1 ADC

#### Data Structures

[\\_ADC\\_Obj](#)

#### Macros

[ADC\\_ADCCTL1\\_ADCBGPWD\\_BITS](#)

[ADC\\_ADCCTL1\\_ADCBSY\\_BITS](#)

[ADC\\_ADCCTL1\\_ADCBSYCHAN\\_BITS](#)

[ADC\\_ADCCTL1\\_ADCENABLE\\_BITS](#)

[ADC\\_ADCCTL1\\_ADCPWDN\\_BITS](#)

[ADC\\_ADCCTL1\\_ADCREFPWD\\_BITS](#)

[ADC\\_ADCCTL1\\_ADCREFSEL\\_BITS](#)

[ADC\\_ADCCTL1\\_INTPULSEPOS\\_BITS](#)

[ADC\\_ADCCTL1\\_RESET\\_BITS](#)

[ADC\\_ADCCTL1\\_TEMPCONV\\_BITS](#)

[ADC\\_ADCCTL1\\_VREFLOCONV\\_BITS](#)

[ADC\\_ADCSAMPLEMODE\\_SEPARATE\\_FLAG](#)

[ADC\\_ADCSAMPLEMODE\\_SIMULEN0\\_BITS](#)

[ADC\\_ADCSAMPLEMODE\\_SIMULEN10\\_BITS](#)

[ADC\\_ADCSAMPLEMODE\\_SIMULEN12\\_BITS](#)

[ADC\\_ADCSAMPLEMODE\\_SIMULEN14\\_BITS](#)

[ADC\\_ADCSAMPLEMODE\\_SIMULEN2\\_BITS](#)

ADC\_ADCSAMPLEMODE\_SIMULEN4\_BITS  
ADC\_ADCSAMPLEMODE\_SIMULEN6\_BITS  
ADC\_ADCSAMPLEMODE\_SIMULEN8\_BITS  
ADC\_ADCSOCxCTL\_ACQPS\_BITS  
ADC\_ADCSOCxCTL\_CHSEL\_BITS  
ADC\_ADCSOCxCTL\_TRIGSEL\_BITS  
ADC\_BASE\_ADDR  
ADC\_dataBias  
ADC\_DELAY\_usec  
ADC\_INTSELxNy\_INTCONT\_BITS  
ADC\_INTSELxNy\_INTE\_BITS  
ADC\_INTSELxNy\_INTSEL\_BITS  
ADC\_INTSELxNy\_LOG2\_NUMBITS\_PER\_REG  
ADC\_INTSELxNy\_NUMBITS\_PER\_REG

## Enumerations

ADC\_IntMode\_e  
ADC\_IntNumber\_e  
ADC\_IntPulseGenMode\_e  
ADC\_IntSrc\_e  
ADC\_ResultNumber\_e  
ADC\_SampleMode\_e  
ADC\_SocChanNumber\_e  
ADC\_SocNumber\_e  
ADC\_SocSampleWindow\_e  
ADC\_SocTrigSrc\_e  
ADC\_VoltageRefSrc\_e

## Functions

```
void ADC_clearIntFlag (ADC_Handle adcHandle, const ADC_IntNumber_e intNumber)

void ADC_disable (ADC_Handle adcHandle)

void ADC_disableBandGap (ADC_Handle adcHandle)

void ADC_disableInt (ADC_Handle adcHandle, const ADC_IntNumber_e intNumber)

void ADC_disableRefBuffers (ADC_Handle adcHandle)

void ADC_disableTempSensor (ADC_Handle adcHandle)

void ADC_enable (ADC_Handle adcHandle)

void ADC_enableBandGap (ADC_Handle adcHandle)

void ADC_enableInt (ADC_Handle adcHandle, const ADC_IntNumber_e intNumber)

void ADC_enableRefBuffers (ADC_Handle adcHandle)

void ADC_enableTempSensor (ADC_Handle adcHandle)

void ADC_forceConversion (ADC_Handle adcHandle, const ADC_SocNumber_e socNumber)

bool_t ADC_getIntStatus (ADC_Handle adcHandle, const ADC_IntNumber_e intNumber)

ADC_SocSampleWindow_e ADC_getSocSampleWindow (ADC_Handle adcHandle, const
ADC_SocNumber_e socNumber)

int16_t ADC_getTemperatureC (ADC_Handle adcHandle, int16_t sensorSample)

int16_t ADC_getTemperatureK (ADC_Handle adcHandle, int16_t sensorSample)

ADC_Handle ADC_init (void *pMemory, const size_t numBytes)

void ADC_powerDown (ADC_Handle adcHandle)

void ADC_powerUp (ADC_Handle adcHandle)

uint_least16_t ADC_readResult (ADC_Handle adcHandle, const ADC_ResultNumber_e result-
Number)

void ADC_reset (ADC_Handle adcHandle)

void ADC_setIntMode (ADC_Handle adcHandle, const ADC_IntNumber_e intNumber, const
ADC_IntMode_e intMode)

void ADC_setIntPulseGenMode (ADC_Handle adcHandle, const ADC_IntPulseGenMode_e pulse-
Mode)

void ADC_setIntSrc (ADC_Handle adcHandle, const ADC_IntNumber_e intNumber, const
ADC_IntSrc_e intSrc)

void ADC_setSampleMode (ADC_Handle adcHandle, const ADC_SampleMode_e sampleMode)
```

```
void ADC_setSocChanNumber (ADC_Handle adcHandle, const ADC_SocNumber_e socNumber,
const ADC_SocChanNumber_e chanNumber)
```

```
void ADC_setSocSampleWindow (ADC_Handle adcHandle, const ADC_SocNumber_e socNum-
ber, const ADC_SocSampleWindow_e sampleWindow)
```

```
void ADC_setSocTrigSrc (ADC_Handle adcHandle, const ADC_SocNumber_e socNumber, const
ADC_SocTrigSrc_e trigSrc)
```

```
void ADC_setVoltRefSrc (ADC_Handle adcHandle, const ADC_VoltageRefSrc_e voltRef)
```

### 3.1.1 Detailed Description

### 3.1.2 Data Structure Documentation

#### 3.1.2.1 \_ADC\_Obj\_

**Definition:**

```
typedef struct
{
    uint16_t ADCRESULT[16];
    uint16_t rsvd_1[26096];
    uint16_t ADCCTL1;
    uint16_t rsvd_2[3];
    uint16_t ADCINTFLG;
    uint16_t ADCINTFLGCLR;
    uint16_t ADCINTOVF;
    uint16_t ADCINTOVFCLR;
    uint16_t INTSELxNy[5];
    uint16_t rsvd_3[3];
    uint16_t SOCPRICTRL;
    uint16_t rsvd_4;
    uint16_t ADCSAMPLEMODE;
    uint16_t rsvd_5;
    uint16_t ADCINTSOCSEL1;
    uint16_t ADCINTSOCSEL2;
    uint16_t rsvd_6[2];
    uint16_t ADCSOCFLG1;
    uint16_t rsvd_7;
    uint16_t ADCSOCFRC1;
    uint16_t rsvd_8;
    uint16_t ADCSOCOVF1;
    uint16_t rsvd_9;
    uint16_t ADCSOCOVFCLR1;
    uint16_t rsvd_10;
    uint16_t ADCSOCxCTL[16];
    uint16_t rsvd_11[16];
    uint16_t ADCREFTRIM;
    uint16_t ADCOFFTRIM;
    uint16_t rsvd_12[13];
    uint16_t ADCREV;
```

```
}  
_ADC_Obj_
```

**Members:**

**ADCRESULT** ADC result registers.

**rsvd\_1** Reserved.

**ADCCTL1** ADC Control Register 1.

**rsvd\_2** Reserved.

**ADCINTFLG** ADC Interrupt Flag Register.

**ADCINTFLGCLR** ADC Interrupt Flag Clear Register.

**ADCINTOVF** ADC Interrupt Overflow Register.

**ADCINTOVFCLR** ADC Interrupt Overflow Clear Register.

**INTSELxNy** ADC Interrupt Select x and y Register.

**rsvd\_3** Reserved.

**SOCPRICTRL** ADC Start Of Conversion Priority Control Register.

**rsvd\_4** Reserved.

**ADCSAMPLEMODE** ADC Sample Mode Register.

**rsvd\_5** Reserved.

**ADCINTSOCSEL1** ADC Interrupt Trigger SOC Select 1 Register.

**ADCINTSOCSEL2** ADC Interrupt Trigger SOC Select 2 Register.

**rsvd\_6** Reserved.

**ADCSOCFLG1** ADC SOC Flag 1 Register.

**rsvd\_7** Reserved.

**ADCSOCFRC1** ADC SOC Force 1 Register.

**rsvd\_8** Reserved.

**ADCSOCOVF1** ADC SOC Overflow 1 Register.

**rsvd\_9** Reserved.

**ADCSOCOVFCLR1** ADC SOC Overflow Clear 1 Register.

**rsvd\_10** Reserved.

**ADCSOCxCTL** ADC SOCx Control Registers.

**rsvd\_11** Reserved.

**ADCREFTRIM** ADC Reference/Gain Trim Register.

**ADCOFFTRIM** ADC Offset Trim Register.

**rsvd\_12** Reserved.

**ADCREV** ADC Revision Register.

**Description:**

Defines the analog-to-digital converter (ADC) object.



## 4 Capture (CAP)

[Introduction](#) ..... ??

[API Functions](#) 15 The Enhanced Capture (CAP) API provides a set of functions and data structs for configuring and capturing data as well as generating PWMs with the capture peripheral.

This driver is contained in `f2802x0/common/source/cap.c`, with `f2802x0/common/include/cap.h` containing the API definitions and data structs for use by applications.

### 4.1 CAP

#### Data Structures

struct `_CAP_Obj`

#### Macros

```
#define CAP_ECCTL1_CAP1POL_BITS
#define CAP_ECCTL1_CAP2POL_BITS
#define CAP_ECCTL1_CAP3POL_BITS
#define CAP_ECCTL1_CAP4POL_BITS
#define CAP_ECCTL1_CAPLDEN_BITS
#define CAP_ECCTL1_CTRRST1_BITS
#define CAP_ECCTL1_CTRRST2_BITS
#define CAP_ECCTL1_CTRRST3_BITS
#define CAP_ECCTL1_CTRRST4_BITS
#define CAP_ECCTL1_FREESOFT_BITS
#define CAP_ECCTL1_PRESCALE_BITS
#define CAP_ECCTL2_APWMPOL_BITS
#define CAP_ECCTL2_CAPAPWM_BITS
#define CAP_ECCTL2_CONTONESHOT_BITS
#define CAP_ECCTL2_REARM_BITS
#define CAP_ECCTL2_STOP_WRAP_BITS
#define CAP_ECCTL2_SWSYNC_BITS
```

```
#define CAP_ECCTL2_SYNCIEN_BITS
#define CAP_ECCTL2_SYNCOSSEL_BITS
#define CAP_ECCTL2_TSCTRSTOP_BITS
#define CAP_ECCxxx_C EVT1_BITS
#define CAP_ECCxxx_C EVT2_BITS
#define CAP_ECCxxx_C EVT3_BITS
#define CAP_ECCxxx_C EVT4_BITS
#define CAP_ECCxxx_C TRCOMP_BITS
#define CAP_ECCxxx_C TROVF_BITS
#define CAP_ECCxxx_C TRPRD_BITS
#define CAP_ECCxxx_INT_BITS
#define CAPA_BASE_ADDR
```

## Typedefs

```
typedef struct _CAP_Obj * CAP_Handle
typedef struct _CAP_Obj CAP_Obj
```

## Enumerations

```
enum CAP_Event_e { CAP_Event_1, CAP_Event_2, CAP_Event_3, CAP_Event_4 }

enum CAP_Int_Type_e {
    CAP_Int_Type_CTR_CMP,          CAP_Int_Type_CTR_PRD,          CAP_Int_Type_CTR_OVF,
    CAP_Int_Type_C EVT4,
    CAP_Int_Type_C EVT3, CAP_Int_Type_C EVT2, CAP_Int_Type_C EVT1, CAP_Int_Type_Global,
    CAP_Int_Type_All }

enum CAP_Polarity_e { CAP_Polarity_Rising, CAP_Polarity_Falling }

enum CAP_Prescale_e {
    CAP_Prescale_By_1, CAP_Prescale_By_2, CAP_Prescale_By_4, CAP_Prescale_By_6,
    CAP_Prescale_By_8, CAP_Prescale_By_10, CAP_Prescale_By_12, CAP_Prescale_By_14,
    CAP_Prescale_By_16, CAP_Prescale_By_18, CAP_Prescale_By_20, CAP_Prescale_By_22,
    CAP_Prescale_By_24, CAP_Prescale_By_26, CAP_Prescale_By_28, CAP_Prescale_By_30,
    CAP_Prescale_By_32, CAP_Prescale_By_34, CAP_Prescale_By_36, CAP_Prescale_By_38,
    CAP_Prescale_By_40, CAP_Prescale_By_42, CAP_Prescale_By_44, CAP_Prescale_By_46,
    CAP_Prescale_By_48, CAP_Prescale_By_50, CAP_Prescale_By_52, CAP_Prescale_By_54,
    CAP_Prescale_By_56, CAP_Prescale_By_58, CAP_Prescale_By_60, CAP_Prescale_By_62 }
```



```
enum CAP_Reset_e { CAP_Reset_Disable, CAP_Reset_Enable }

enum CAP_RunMode_e

enum CAP_Stop_Wrap_e { CAP_Stop_Wrap_C EVT1, CAP_Stop_Wrap_C EVT2,
CAP_Stop_Wrap_C EVT3, CAP_Stop_Wrap_C EVT4 }

enum CAP_SyncOut_e { CAP_SyncOut_SyncIn, CAP_SyncOut_CTRPRD,
CAP_SyncOut_Disable }
```

## Functions

```
void CAP_clearInt (CAP_Handle capHandle, const CAP_Int_Type_e intType)

void CAP_disableCaptureLoad (CAP_Handle capHandle)

void CAP_disableInt (CAP_Handle capHandle, const CAP_Int_Type_e intType)

void CAP_disableSyncIn (CAP_Handle capHandle)

void CAP_disableTimestampCounter (CAP_Handle capHandle)

void CAP_enableCaptureLoad (CAP_Handle capHandle)

void CAP_enableInt (CAP_Handle capHandle, const CAP_Int_Type_e intType)

void CAP_enableSyncIn (CAP_Handle capHandle)

void CAP_enableTimestampCounter (CAP_Handle capHandle)

uint32_t CAP_getCap1 (CAP_Handle capHandle)

uint32_t CAP_getCap2 (CAP_Handle capHandle)

uint32_t CAP_getCap3 (CAP_Handle capHandle)

uint32_t CAP_getCap4 (CAP_Handle capHandle)

CAP_Handle CAP_init (void *pMemory, const size_t numBytes)

void CAP_rearm (CAP_Handle capHandle)

void CAP_setApwmCompare (CAP_Handle capHandle, const uint32_t compare)

void CAP_setApwmPeriod (CAP_Handle capHandle, const uint32_t period)

void CAP_setApwmShadowPeriod (CAP_Handle capHandle, const uint32_t shadowPeriod)

void CAP_setCapContinuous (CAP_Handle capHandle)

void CAP_setCapEvtPolarity (CAP_Handle capHandle, const CAP_Event_e event, const
CAP_Polarity_e polarity)

void CAP_setCapEvtReset (CAP_Handle capHandle, const CAP_Event_e event, const
CAP_Reset_e reset)
```

```
void CAP_setCapOneShot (CAP_Handle capHandle)
void CAP_setModeApwm (CAP_Handle capHandle)
void CAP_setModeCap (CAP_Handle capHandle)
void CAP_setStopWrap (CAP_Handle capHandle, const CAP_Stop_Wrap_e stopWrap)
void CAP_setSyncOut (CAP_Handle capHandle, const CAP_SyncOut_e syncOut)
```

#### 4.1.1 Detailed Description

#### 4.1.2 Enumeration Type Documentation

##### 4.1.2.1 enum **CAP\_Event\_e**

Enumeration to define the capture (CAP) events.

**Enumerator**

**CAP\_Event\_1** Capture Event 1.  
**CAP\_Event\_2** Capture Event 2.  
**CAP\_Event\_3** Capture Event 3.  
**CAP\_Event\_4** Capture Event 4.

##### 4.1.2.2 enum **CAP\_Int\_Type\_e**

Enumeration to define the capture (CAP) interrupts.

**Enumerator**

**CAP\_Int\_Type\_CTR\_CMP** Denotes CTR = CMP interrupt.  
**CAP\_Int\_Type\_CTR\_PRD** Denotes CTR = PRD interrupt.  
**CAP\_Int\_Type\_CTR\_OVF** Denotes CTROVF interrupt.  
**CAP\_Int\_Type\_C EVT4** Denotes CEVT4 interrupt.  
**CAP\_Int\_Type\_C EVT3** Denotes CEVT3 interrupt.  
**CAP\_Int\_Type\_C EVT2** Denotes CEVT2 interrupt.  
**CAP\_Int\_Type\_C EVT1** Denotes CEVT1 interrupt.  
**CAP\_Int\_Type\_Global** Denotes Capture global interrupt.  
**CAP\_Int\_Type\_All** Denotes All interrupts.

##### 4.1.2.3 enum **CAP\_Polarity\_e**

Enumeration to define the capture (CAP) event polarities.

**Enumerator**

**CAP\_Polarity\_Rising** Rising Edge Triggered.  
**CAP\_Polarity\_Falling** Falling Edge Triggered.

#### 4.1.2.4 enum **CAP\_Prescale\_e**

Enumeration to define the capture (CAP) prescaler values.

##### Enumerator

**CAP\_Prescale\_By\_1** Divide by 1.  
**CAP\_Prescale\_By\_2** Divide by 2.  
**CAP\_Prescale\_By\_4** Divide by 4.  
**CAP\_Prescale\_By\_6** Divide by 6.  
**CAP\_Prescale\_By\_8** Divide by 8.  
**CAP\_Prescale\_By\_10** Divide by 10.  
**CAP\_Prescale\_By\_12** Divide by 12.  
**CAP\_Prescale\_By\_14** Divide by 14.  
**CAP\_Prescale\_By\_16** Divide by 16.  
**CAP\_Prescale\_By\_18** Divide by 18.  
**CAP\_Prescale\_By\_20** Divide by 20.  
**CAP\_Prescale\_By\_22** Divide by 22.  
**CAP\_Prescale\_By\_24** Divide by 24.  
**CAP\_Prescale\_By\_26** Divide by 26.  
**CAP\_Prescale\_By\_28** Divide by 28.  
**CAP\_Prescale\_By\_30** Divide by 30.  
**CAP\_Prescale\_By\_32** Divide by 32.  
**CAP\_Prescale\_By\_34** Divide by 34.  
**CAP\_Prescale\_By\_36** Divide by 36.  
**CAP\_Prescale\_By\_38** Divide by 38.  
**CAP\_Prescale\_By\_40** Divide by 40.  
**CAP\_Prescale\_By\_42** Divide by 42.  
**CAP\_Prescale\_By\_44** Divide by 44.  
**CAP\_Prescale\_By\_46** Divide by 46.  
**CAP\_Prescale\_By\_48** Divide by 48.  
**CAP\_Prescale\_By\_50** Divide by 50.  
**CAP\_Prescale\_By\_52** Divide by 52.  
**CAP\_Prescale\_By\_54** Divide by 54.  
**CAP\_Prescale\_By\_56** Divide by 56.  
**CAP\_Prescale\_By\_58** Divide by 58.  
**CAP\_Prescale\_By\_60** Divide by 60.  
**CAP\_Prescale\_By\_62** Divide by 62.

#### 4.1.2.5 enum **CAP\_Reset\_e**

Enumeration to define the capture (CAP) event resets.

##### Enumerator

**CAP\_Reset\_Disable** Disable counter reset on capture event.  
**CAP\_Reset\_Enable** Enable counter reset on capture event.

#### 4.1.2.6 enum **CAP\_Stop\_Wrap\_e**

Enumeration to define the capture (CAP) Stop/Wrap modes.

##### Enumerator

**CAP\_Stop\_Wrap\_C EVT1** Stop/Wrap after Capture Event 1.  
**CAP\_Stop\_Wrap\_C EVT2** Stop/Wrap after Capture Event 2.  
**CAP\_Stop\_Wrap\_C EVT3** Stop/Wrap after Capture Event 3.  
**CAP\_Stop\_Wrap\_C EVT4** Stop/Wrap after Capture Event 4.

#### 4.1.2.7 enum **CAP\_SyncOut\_e**

Enumeration to define the Sync Out options.

##### Enumerator

**CAP\_SyncOut\_SyncIn** Sync In used for Sync Out.  
**CAP\_SyncOut\_CTRPRD** CTR = PRD used for Sync Out.  
**CAP\_SyncOut\_Disable** Disables Sync Out.

### 4.1.3 Function Documentation

#### 4.1.3.1 void CAP\_clearInt (

**CAP\_Handle** capHandle,

const **CAP\_Int\_Type\_e** intType ) [inline]

Clears capture (CAP) interrupt flag.

[1]Parameters in *capHandle* The capture (CAP) object handle

---

in *intType* The capture interrupt to be cleared

---

References *\_CAP\_Obj\_::ECECLR*.

#### 4.1.3.2 void CAP\_disableCaptureLoad (

**CAP\_Handle** capHandle )

Disables loading of CAP1-4 on capture event.

[1]Parameters in *capHandle* The capture (CAP) object handle

#### 4.1.3.3 void CAP\_disableInt (

**CAP\_Handle** capHandle,

const **CAP\_Int\_Type\_e** intType )

Disables capture (CAP) interrupt source.

[1]Parameters in *capHandle* The capture (CAP) object handle

---

in *intType* The capture interrupt type to be disabled

---

4.1.3.4 void CAP\_disableSyncIn (

**CAP\_Handle** capHandle )

Disables counter synchronization.

[1]Parameters in *capHandle* The capture (CAP) object handle

---

4.1.3.5 void CAP\_disableTimestampCounter (

**CAP\_Handle** capHandle )

Disables Time Stamp counter from running.

[1]Parameters in *capHandle* The capture (CAP) object handle

---

4.1.3.6 void CAP\_enableCaptureLoad (

**CAP\_Handle** capHandle )

Enables loading of CAP1-4 on capture event.

[1]Parameters in *capHandle* The capture (CAP) object handle

---

4.1.3.7 void CAP\_enableInt (

**CAP\_Handle** capHandle,

const **CAP\_Int\_Type\_e** intType )

Enables capture (CAP) interrupt source.

[1]Parameters in *capHandle* The capture (CAP) object handle

---

in *intType* The capture interrupt type to be enabled

---

4.1.3.8 void CAP\_enableSyncln (  
CAP\_Handle capHandle )

Enables counter synchronization.

[1]Parameters in *capHandle* The capture (CAP) object handle

---

4.1.3.9 void CAP\_enableTimestampCounter (  
CAP\_Handle capHandle )

Enables Time Stamp counter to running.

[1]Parameters in *capHandle* The capture (CAP) object handle

---

4.1.3.10 uint32\_t CAP\_getCap1 (  
CAP\_Handle capHandle ) [inline]

Gets the CAP1 register value.

[1]Parameters in *capHandle* The capture (CAP) object handle

---

References \_CAP\_Obj\_::CAP1.

4.1.3.11 uint32\_t CAP\_getCap2 (  
CAP\_Handle capHandle ) [inline]

Gets the CAP2 register value.

[1]Parameters in *capHandle* The capture (CAP) object handle

---

References \_CAP\_Obj\_::CAP2.

4.1.3.12 uint32\_t CAP\_getCap3 (  
CAP\_Handle capHandle ) [inline]

Gets the CAP3 register value.

[1]Parameters in *capHandle* The capture (CAP) object handle

---

References \_CAP\_Obj\_::CAP3.

4.1.3.13 `uint32_t CAP_getCap4 (`  
`CAP_Handle capHandle ) [inline]`

Gets the CAP4 register value.

[1]Parameters in *capHandle* The capture (CAP) object handle

---

References `_CAP_Obj_::CAP4`.

4.1.3.14 `CAP_Handle CAP_init (`  
`void * pMemory,`  
`const size_t numBytes )`

Initializes the capture (CAP) object handle.

[1]Parameters in *pMemory* A pointer to the base address of the CAP registers

---

in *numBytes* The number of bytes allocated for the CAP object, bytes

---

Returns The capture (CAP) object handle

4.1.3.15 `void CAP_rearm (`  
`CAP_Handle capHandle ) [inline]`

(Re-)Arm the capture module

[1]Parameters in *capHandle* The capture (CAP) object handle

---

References `CAP_ECCTL2_REARM_BITS`, and `_CAP_Obj_::ECCTL2`.

4.1.3.16 `void CAP_setApwmCompare (`  
`CAP_Handle capHandle,`  
`const uint32_t compare ) [inline]`

Sets the APWM compare value.

[1]Parameters in *capHandle* The capture (CAP) object handle

---

in *compare* The APWM compare value

---

References `_CAP_Obj_::CAP2`.





722.7

*in polarity* The polarity to configure the event for

---

4.1.3.21 void CAP\_setCapEvtReset (  
    **CAP\_Handle** capHandle,  
    const **CAP\_Event\_e** event,  
    const **CAP\_Reset\_e** reset )

Sets the capture event counter reset configuration.

[1]Parameters *in capHandle* The capture (CAP) object handle

---

*in event* The event to configure

---

*in reset* Whether the event should reset the counter or not

---

4.1.3.22 void CAP\_setCapOneShot (  
    **CAP\_Handle** capHandle )

Sets up for one-shot Capture.

[1]Parameters *in capHandle* The capture (CAP) object handle

---

4.1.3.23 void CAP\_setModeApwm (  
    **CAP\_Handle** capHandle )

Sets capture peripheral up for APWM mode.

[1]Parameters *in capHandle* The capture (CAP) object handle

---

4.1.3.24 void CAP\_setModeCap (  
    **CAP\_Handle** capHandle )

Sets capture peripheral up for capture mode.

[1]Parameters *in capHandle* The capture (CAP) object handle

---

4.1.3.25 void CAP\_setStopWrap (  
    **CAP\_Handle** capHandle,  
    const **CAP\_Stop\_Wrap\_e** stopWrap )

Set the stop/wrap mode.

[1]Parameters in *capHandle* The capture (CAP) object handle

---

in *stopWrap* The stop/wrap mode to set

---

4.1.3.26 void CAP\_setSyncOut (  
    **CAP\_Handle** capHandle,  
    const **CAP\_SyncOut\_e** syncOut )

Set the sync out mode.

[1]Parameters in *capHandle* The capture (CAP) object handle

---

in *syncOut* The sync out mode to set

---

## 5 Device Clocking (CLK)

Introduction .....	??
API Functions .....	27

The CLK API provides functions to control the clocking subsystem of the device. Clock dividers and prescalers as well as peripheral clocks can all be set or enabled via this API.

This driver is contained in `f2802x0/common/source/clk.c`, with `f2802x0/common/include/clk.h` containing the API definitions for use by applications.

### 5.1 CLK

#### Data Structures

struct `_CLK_Obj_`

#### Macros

```
#define CLK_BASE_ADDR
#define CLK_CLKCTL_INTOSC1HALTI_BITS
#define CLK_CLKCTL_INTOSC1OFF_BITS
#define CLK_CLKCTL_INTOSC2HALTI_BITS
#define CLK_CLKCTL_INTOSC2OFF_BITS
#define CLK_CLKCTL_NMIRESETSEL_BITS
#define CLK_CLKCTL_OSCCLKSRC2SEL_BITS
#define CLK_CLKCTL_OSCCLKSRCSEL_BITS
#define CLK_CLKCTL_TMR2CLKPRESCALE_BITS
#define CLK_CLKCTL_TMR2CLKSRCSEL_BITS
#define CLK_CLKCTL_WDCLKSRCSEL_BITS
#define CLK_CLKCTL_WDHALTI_BITS
#define CLK_CLKCTL_XCLKINOFF_BITS
#define CLK_CLKCTL_XTALOSCOFF_BITS
#define CLK_LOSPCP_LSPCLK_BITS
#define CLK_PCLKCR0_ADCENCLK_BITS
```

```
#define CLK_PCLKCR0_ECANAENCLK_BITS
#define CLK_PCLKCR0_HRPWMENCLK_BITS
#define CLK_PCLKCR0_I2CAENCLK_BITS
#define CLK_PCLKCR0_LINAENCLK_BITS
#define CLK_PCLKCR0_SCIAENCLK_BITS
#define CLK_PCLKCR0_SPIAENCLK_BITS
#define CLK_PCLKCR0_SPIBENCLK_BITS
#define CLK_PCLKCR0_TBCLKSYNC_BITS
#define CLK_PCLKCR1_ECAP1ENCLK_BITS
#define CLK_PCLKCR1_EPWM1ENCLK_BITS
#define CLK_PCLKCR1_EPWM2ENCLK_BITS
#define CLK_PCLKCR1_EPWM3ENCLK_BITS
#define CLK_PCLKCR1_EPWM4ENCLK_BITS
#define CLK_PCLKCR1_EPWM5ENCLK_BITS
#define CLK_PCLKCR1_EPWM6ENCLK_BITS
#define CLK_PCLKCR1_EPWM7ENCLK_BITS
#define CLK_PCLKCR1_EQEP1ENCLK_BITS
#define CLK_PCLKCR3_CLA1ENCLK_BITS
#define CLK_PCLKCR3_COMP1ENCLK_BITS
#define CLK_PCLKCR3_COMP2ENCLK_BITS
#define CLK_PCLKCR3_COMP3ENCLK_BITS
#define CLK_PCLKCR3_CPUTIMER0ENCLK_BITS
#define CLK_PCLKCR3_CPUTIMER1ENCLK_BITS
#define CLK_PCLKCR3_CPUTIMER2ENCLK_BITS
#define CLK_PCLKCR3_GPIOINENCLK_BITS
#define CLK_XCLK_XCLKINSEL_BITS
#define CLK_XCLK_XCLKOUTDIV_BITS
```

## Typedefs

```
typedef struct _CLK_Obj * CLK_Handle
```

```
typedef struct _CLK_Obj CLK_Obj
```

## Enumerations

```
enum CLK_ClkInSrc_e
```

```
enum CLK_ClkOutPreScaler_e { CLK_ClkOutPreScaler_SysClkOut_by_4,  
CLK_ClkOutPreScaler_SysClkOut_by_2, CLK_ClkOutPreScaler_SysClkOut_by_1,  
CLK_ClkOutPreScaler_Off }
```

```
enum CLK_CompNumber_e { CLK_CompNumber_1, CLK_CompNumber_2,  
CLK_CompNumber_3 }
```

```
enum CLK_CpuTimerNumber_e { CLK_CpuTimerNumber_0, CLK_CpuTimerNumber_1,  
CLK_CpuTimerNumber_2 }
```

```
enum CLK_LowSpdPreScaler_e {  
CLK_LowSpdPreScaler_SysClkOut_by_1, CLK_LowSpdPreScaler_SysClkOut_by_2,  
CLK_LowSpdPreScaler_SysClkOut_by_4, CLK_LowSpdPreScaler_SysClkOut_by_6,  
CLK_LowSpdPreScaler_SysClkOut_by_8, CLK_LowSpdPreScaler_SysClkOut_by_10,  
CLK_LowSpdPreScaler_SysClkOut_by_12, CLK_LowSpdPreScaler_SysClkOut_by_14 }
```

```
enum CLK_Osc2Src_e { CLK_Osc2Src_Internal, CLK_Osc2Src_External }
```

```
enum CLK_OscSrc_e { CLK_OscSrc_Internal, CLK_OscSrc_External }
```

```
enum CLK_Timer2PreScaler_e {  
CLK_Timer2PreScaler_by_1, CLK_Timer2PreScaler_by_2, CLK_Timer2PreScaler_by_4,  
CLK_Timer2PreScaler_by_8,  
CLK_Timer2PreScaler_by_16 }
```

```
enum CLK_Timer2Src_e { CLK_Timer2Src_SysClk, CLK_Timer2Src_ExtOsc,  
CLK_Timer2Src_IntOsc1, CLK_Timer2Src_IntOsc2 }
```

```
enum CLK_WdClkSrc_e { CLK_WdClkSrc_IntOsc1, CLK_WdClkSrc_ExtOscOrIntOsc2 }
```

## Functions

```
void CLK_disableAdcClock (CLK_Handle clkHandle)
```

```
void CLK_disableClaClock (CLK_Handle clkHandle)
```

```
void CLK_disableCkIn (CLK_Handle clkHandle)
```

```
void CLK_disableCompClock (CLK_Handle clkHandle, const CLK_CompNumber_e compNumber)
```

```
void CLK_disableCpuTimerClock (CLK_Handle clkHandle, const CLK_CpuTimerNumber_e  
cpuTimerNumber)
```

```
void CLK_disableCrystalOsc (CLK_Handle clkHandle)
void CLK_disableEcanaClock (CLK_Handle clkHandle)
void CLK_disableEcap1Clock (CLK_Handle clkHandle)
void CLK_disableEqep1Clock (CLK_Handle clkHandle)
void CLK_disableGpioInputClock (CLK_Handle clkHandle)
void CLK_disableHrPwmClock (CLK_Handle clkHandle)
void CLK_disableI2cClock (CLK_Handle clkHandle)
void CLK_disableLinAClock (CLK_Handle clkHandle)
void CLK_disableOsc1 (CLK_Handle clkHandle)
void CLK_disableOsc1HaltMode (CLK_Handle clkHandle)
void CLK_disableOsc2 (CLK_Handle clkHandle)
void CLK_disableOsc2HaltMode (CLK_Handle clkHandle)
void CLK_disablePwmClock (CLK_Handle clkHandle, const PWM_Number_e pwmNumber)
void CLK_disableSciaClock (CLK_Handle clkHandle)
void CLK_disableSpiaClock (CLK_Handle clkHandle)
void CLK_disableSpibClock (CLK_Handle clkHandle)
void CLK_disableTbClockSync (CLK_Handle clkHandle)
void CLK_disableWatchDogHaltMode (CLK_Handle clkHandle)
void CLK_enableAdcClock (CLK_Handle clkHandle)
void CLK_enableCiaClock (CLK_Handle clkHandle)
void CLK_enableClkIn (CLK_Handle clkHandle)
void CLK_enableCompClock (CLK_Handle clkHandle, const CLK_CompNumber_e compNumber)
void CLK_enableCpuTimerClock (CLK_Handle clkHandle, const CLK_CpuTimerNumber_e
cpuTimerNumber)
void CLK_enableCrystalOsc (CLK_Handle clkHandle)
void CLK_enableEcanaClock (CLK_Handle clkHandle)
void CLK_enableEcap1Clock (CLK_Handle clkHandle)
void CLK_enableEqep1Clock (CLK_Handle clkHandle)
void CLK_enableGpioInputClock (CLK_Handle clkHandle)
void CLK_enableHrPwmClock (CLK_Handle clkHandle)
```

```
void CLK_enableI2cClock (CLK_Handle clkHandle)
void CLK_enableLinAClock (CLK_Handle clkHandle)
void CLK_enableOsc1 (CLK_Handle clkHandle)
void CLK_enableOsc1HaltMode (CLK_Handle clkHandle)
void CLK_enableOsc2 (CLK_Handle clkHandle)
void CLK_enableOsc2HaltMode (CLK_Handle clkHandle)
void CLK_enablePwmClock (CLK_Handle clkHandle, const PWM_Number_e pwmNumber)
void CLK_enableSciaClock (CLK_Handle clkHandle)
void CLK_enableSpiaClock (CLK_Handle clkHandle)
void CLK_enableSpibClock (CLK_Handle clkHandle)
void CLK_enableTbClockSync (CLK_Handle clkHandle)
void CLK_enableWatchDogHaltMode (CLK_Handle clkHandle)
CLK_Handle CLK_init (void *pMemory, const size_t numBytes)
void CLK_setClkOutPreScaler (CLK_Handle clkHandle, const CLK_ClkOutPreScaler_e preScaler)
void CLK_setLowSpdPreScaler (CLK_Handle clkHandle, const CLK_LowSpdPreScaler_e preScaler)
void CLK_setOsc2Src (CLK_Handle clkHandle, const CLK_Osc2Src_e src)
void CLK_setOscSrc (CLK_Handle clkHandle, const CLK_OscSrc_e src)
void CLK_setTimer2PreScaler (CLK_Handle clkHandle, const CLK_Timer2PreScaler_e preScaler)
void CLK_setTimer2Src (CLK_Handle clkHandle, const CLK_Timer2Src_e src)
void CLK_setWatchDogSrc (CLK_Handle clkHandle, const CLK_WdClkSrc_e src)
```

### 5.1.1 Detailed Description

### 5.1.2 Enumeration Type Documentation

#### 5.1.2.1 enum **CLK\_ClkOutPreScaler\_e**

Enumeration to define the external clock output frequency.

##### Enumerator

**CLK\_ClkOutPreScaler\_SysClkOut\_by\_4** Denotes XCLKOUT = SYSCLKOUT/4.

**CLK\_ClkOutPreScaler\_SysClkOut\_by\_2** Denotes XCLKOUT = SYSCLKOUT/2.

**CLK\_ClkOutPreScaler\_SysClkOut\_by\_1** Denotes XCLKOUT = SYSCLKOUT/1.

**CLK\_ClkOutPreScaler\_Off** Denotes XCLKOUT = Off.

#### 5.1.2.2 enum **CLK\_CompNumber\_e**

Enumeration to define the comparator numbers.

##### Enumerator

**CLK\_CompNumber\_1** Denotes comparator number 1.

**CLK\_CompNumber\_2** Denotes comparator number 2.

**CLK\_CompNumber\_3** Denotes comparator number 3.

#### 5.1.2.3 enum **CLK\_CpuTimerNumber\_e**

Enumeration to define the CPU timer numbers.

##### Enumerator

**CLK\_CpuTimerNumber\_0** Denotes CPU timer number 0.

**CLK\_CpuTimerNumber\_1** Denotes CPU timer number 1.

**CLK\_CpuTimerNumber\_2** Denotes CPU timer number 2.

#### 5.1.2.4 enum **CLK\_LowSpdPreScaler\_e**

Enumeration to define the low speed clock prescaler, which sets the clock frequency.

##### Enumerator

**CLK\_LowSpdPreScaler\_SysClkOut\_by\_1** Denotes Low Speed Clock = SYSCLKOUT/1.

**CLK\_LowSpdPreScaler\_SysClkOut\_by\_2** Denotes Low Speed Clock = SYSCLKOUT/2.

**CLK\_LowSpdPreScaler\_SysClkOut\_by\_4** Denotes Low Speed Clock = SYSCLKOUT/4.

**CLK\_LowSpdPreScaler\_SysClkOut\_by\_6** Denotes Low Speed Clock = SYSCLKOUT/6.

**CLK\_LowSpdPreScaler\_SysClkOut\_by\_8** Denotes Low Speed Clock = SYSCLKOUT/8.

**CLK\_LowSpdPreScaler\_SysClkOut\_by\_10** Denotes Low Speed Clock = SYSCLKOUT/10.

**CLK\_LowSpdPreScaler\_SysClkOut\_by\_12** Denotes Low Speed Clock = SYSCLKOUT/12.

**CLK\_LowSpdPreScaler\_SysClkOut\_by\_14** Denotes Low Speed Clock = SYSCLKOUT/14.

#### 5.1.2.5 enum **CLK\_Osc2Src\_e**

Enumeration to define the clock oscillator 2 source.

##### Enumerator

**CLK\_Osc2Src\_Internal** Denotes an internal oscillator 2 source.

**CLK\_Osc2Src\_External** Denotes an external oscillator 2 source.



### 5.1.2.6 enum **CLK\_OscSrc\_e**

Enumeration to define the clock oscillator source.

#### Enumerator

**CLK\_OscSrc\_Internal** Denotes an internal oscillator source.

**CLK\_OscSrc\_External** Denotes an external oscillator source.

### 5.1.2.7 enum **CLK\_Timer2PreScaler\_e**

Enumeration to define the timer 2 prescaler, which sets the timer 2 frequency.

#### Enumerator

**CLK\_Timer2PreScaler\_by\_1** Denotes a CPU timer 2 clock pre-scaler value of divide by 1.

**CLK\_Timer2PreScaler\_by\_2** Denotes a CPU timer 2 clock pre-scaler value of divide by 2.

**CLK\_Timer2PreScaler\_by\_4** Denotes a CPU timer 2 clock pre-scaler value of divide by 4.

**CLK\_Timer2PreScaler\_by\_8** Denotes a CPU timer 2 clock pre-scaler value of divide by 8.

**CLK\_Timer2PreScaler\_by\_16** Denotes a CPU timer 2 clock pre-scaler value of divide by 16.

### 5.1.2.8 enum **CLK\_Timer2Src\_e**

Enumeration to define the timer 2 source.

#### Enumerator

**CLK\_Timer2Src\_SysClk** Denotes the CPU timer 2 clock source is SYSCLKOUT.

**CLK\_Timer2Src\_ExtOsc** Denotes the CPU timer 2 clock source is external oscillator.

**CLK\_Timer2Src\_IntOsc1** Denotes the CPU timer 2 clock source is internal oscillator 1.

**CLK\_Timer2Src\_IntOsc2** Denotes the CPU timer 2 clock source is internal oscillator 2.

### 5.1.2.9 enum **CLK\_WdClkSrc\_e**

Enumeration to define the watchdog clock source.

#### Enumerator

**CLK\_WdClkSrc\_IntOsc1** Denotes the watchdog clock source is internal oscillator 1.

**CLK\_WdClkSrc\_ExtOscOrIntOsc2** Denotes the watchdog clock source is external oscillator or internal oscillator 2.

## 5.1.3 Function Documentation

### 5.1.3.1 void CLK\_disableAdcClock (

**CLK\_Handle** clkHandle )

Disables the ADC clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

5.1.3.2 void CLK\_disableClaClock (  
    **CLK\_Handle** clkHandle )

Disables the CLA clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

5.1.3.3 void CLK\_disableClkIn (  
    **CLK\_Handle** clkHandle )

Disables the XCLKIN oscillator input.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

5.1.3.4 void CLK\_disableCompClock (  
    **CLK\_Handle** clkHandle,  
    const **CLK\_CompNumber\_e** compNumber )

Disables the comparator clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

in *compNumber* The comparator number

---

5.1.3.5 void CLK\_disableCpuTimerClock (  
    **CLK\_Handle** clkHandle,  
    const **CLK\_CpuTimerNumber\_e** cpuTimerNumber )

Disables the CPU timer clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

in *cpuTimerNumber* The CPU timer number

---

5.1.3.6 void CLK\_disableCrystalOsc (

**CLK\_Handle** clkHandle )

Disables the crystal oscillator.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

## 5.1.3.7 void CLK\_disableEcanaClock (

**CLK\_Handle** clkHandle )

Disables the ECANA clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

## 5.1.3.8 void CLK\_disableEcap1Clock (

**CLK\_Handle** clkHandle )

Disables the ECAP1 clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

## 5.1.3.9 void CLK\_disableEqep1Clock (

**CLK\_Handle** clkHandle )

Disables the EQEP1 clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

## 5.1.3.10 void CLK\_disableGpioInputClock (

**CLK\_Handle** clkHandle )

Disables the GPIO input clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

## 5.1.3.11 void CLK\_disableHrPwmClock (

**CLK\_Handle** clkHandle )

Disables the HRPWM clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

#### 5.1.3.12 void CLK\_disableI2cClock (

**CLK\_Handle** clkHandle )

Disables the I2C clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

#### 5.1.3.13 void CLK\_disableLinAClock (

**CLK\_Handle** clkHandle )

Disables the LIN-A clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

#### 5.1.3.14 void CLK\_disableOsc1 (

**CLK\_Handle** clkHandle )

Disables internal oscillator 1.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

#### 5.1.3.15 void CLK\_disableOsc1HaltMode (

**CLK\_Handle** clkHandle )

Disables internal oscillator 1 halt mode ignore.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

#### 5.1.3.16 void CLK\_disableOsc2 (

**CLK\_Handle** clkHandle )

Disables internal oscillator 2.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

#### 5.1.3.17 void CLK\_disableOsc2HaltMode (

**CLK\_Handle** clkHandle )

Disables internal oscillator 2 halt mode ignore.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

5.1.3.18 void CLK\_disablePwmClock (  
    **CLK\_Handle** clkHandle,  
    const **PWM\_Number\_e** pwmNumber )

Disables the pwm clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

in *pwmNumber* The PWM number

---

5.1.3.19 void CLK\_disableSciaClock (  
    **CLK\_Handle** clkHandle )

Disables the SCI-A clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

5.1.3.20 void CLK\_disableSpiaClock (  
    **CLK\_Handle** clkHandle )

Disables the SPI-A clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

5.1.3.21 void CLK\_disableSpibClock (  
    **CLK\_Handle** clkHandle )

Disables the SPI-B clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

5.1.3.22 void CLK\_disableTbClockSync (  
    **CLK\_Handle** clkHandle )

Disables the ePWM module time base clock sync signal.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

#### 5.1.3.23 void CLK\_disableWatchDogHaltMode (

**CLK\_Handle** clkHandle )

Disables the watchdog halt mode ignore.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

#### 5.1.3.24 void CLK\_enableAdcClock (

**CLK\_Handle** clkHandle )

Enables the ADC clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

#### 5.1.3.25 void CLK\_enableClaClock (

**CLK\_Handle** clkHandle )

Enables the CLA clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

#### 5.1.3.26 void CLK\_enableClkIn (

**CLK\_Handle** clkHandle )

Enables the XCLKIN oscillator input.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

#### 5.1.3.27 void CLK\_enableCompClock (

**CLK\_Handle** clkHandle,

const **CLK\_CompNumber\_e** compNumber )

Enables the comparator clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

in *compNumber* The comparator number

---

5.1.3.28 void CLK\_enableCpuTimerClock (

**CLK\_Handle** clkHandle,

const **CLK\_CpuTimerNumber\_e** cpuTimerNumber )

Enables the CPU timer clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

in *cpuTimerNumber* The CPU timer number

---

5.1.3.29 void CLK\_enableCrystalOsc (

**CLK\_Handle** clkHandle )

Enables the crystal oscillator.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

5.1.3.30 void CLK\_enableEcanaClock (

**CLK\_Handle** clkHandle )

Enables the ECANA clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

5.1.3.31 void CLK\_enableEcap1Clock (

**CLK\_Handle** clkHandle )

Enables the ECAP1 clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

5.1.3.32 void CLK\_enableEqep1Clock (

**CLK\_Handle** clkHandle )

Enables the EQEP1 clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

#### 5.1.3.33 void CLK\_enableGpioInputClock (

**CLK\_Handle** clkHandle )

Enables the GPIO input clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

#### 5.1.3.34 void CLK\_enableHrPwmClock (

**CLK\_Handle** clkHandle )

Enables the HRPWM clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

#### 5.1.3.35 void CLK\_enableI2cClock (

**CLK\_Handle** clkHandle )

Enables the I2C clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

#### 5.1.3.36 void CLK\_enableLinAClock (

**CLK\_Handle** clkHandle )

Enables the LIN-A clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

#### 5.1.3.37 void CLK\_enableOsc1 (

**CLK\_Handle** clkHandle )

Enables internal oscillator 1.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

#### 5.1.3.38 void CLK\_enableOsc1HaltMode (

**CLK\_Handle** clkHandle )

Enables internal oscillator 1 halt mode ignore.



[1]Parameters in *clkHandle* The clock (CLK) object handle

---

5.1.3.39 void CLK\_enableOsc2 (

**CLK\_Handle** clkHandle )

Enables internal oscillator 2.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

5.1.3.40 void CLK\_enableOsc2HaltMode (

**CLK\_Handle** clkHandle )

Enables internal oscillator 2 halt mode ignore.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

5.1.3.41 void CLK\_enablePwmClock (

**CLK\_Handle** clkHandle,

const **PWM\_Number\_e** pwmNumber )

Enables the pwm clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

in *pwmNumber* The PWM number

---

5.1.3.42 void CLK\_enableSciaClock (

**CLK\_Handle** clkHandle )

Enables the SCI-A clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

5.1.3.43 void CLK\_enableSpiaClock (

**CLK\_Handle** clkHandle )

Enables the SPI-A clock.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---



722.7

*in preScaler* The prescaler value

---

5.1.3.49 void CLK\_setLowSpdPreScaler (  
    **CLK\_Handle** clkHandle,  
    const **CLK\_LowSpdPreScaler\_e** preScaler )

Sets the low speed peripheral clock prescaler.

[1]Parameters *in clkHandle* The clock (CLK) object handle

---

*in preScaler* The prescaler value

---

5.1.3.50 void CLK\_setOsc2Src (  
    **CLK\_Handle** clkHandle,  
    const **CLK\_Osc2Src\_e** src )

Sets the oscillator 2 clock source.

[1]Parameters *in clkHandle* The clock (CLK) object handle

---

*in src* The oscillator 2 clock source

---

5.1.3.51 void CLK\_setOscSrc (  
    **CLK\_Handle** clkHandle,  
    const **CLK\_OscSrc\_e** src )

Sets the oscillator clock source.

[1]Parameters *in clkHandle* The clock (CLK) object handle

---

*in src* The oscillator clock source

---

5.1.3.52 void CLK\_setTimer2PreScaler (  
    **CLK\_Handle** clkHandle,  
    const **CLK\_Timer2PreScaler\_e** preScaler )

Sets the timer 2 clock prescaler.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

in *preScaler* The prescaler value

---

#### 5.1.3.53 void CLK\_setTimer2Src (

**CLK\_Handle** clkHandle,

const **CLK\_Timer2Src\_e** src )

Sets the timer 2 clock source.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

in *src* The timer 2 clock source

---

#### 5.1.3.54 void CLK\_setWatchDogSrc (

**CLK\_Handle** clkHandle,

const **CLK\_WdClkSrc\_e** src )

Sets the watchdog clock source.

[1]Parameters in *clkHandle* The clock (CLK) object handle

---

in *src* The watchdog clock source

---

## 6 Comparater (COMP)

[Introduction](#) .....??

[API Functions](#) 45 The Comparator (COMP) API provides the set of functions to configure the analog comparators present on this device.

This driver is contained in `f2802x0/common/source/comp.c`, with `f2802x0/common/include/comp.h` containing the API definitions for use by applications.

### 6.1 COMP

#### Data Structures

struct [\\_COMP\\_Obj](#)

#### Macros

```
#define COMP1_BASE_ADDR
#define COMP2_BASE_ADDR
#define COMP_COMPCTL_CMPINV_BITS
#define COMP_COMPCTL_COMPDACE_BITS
#define COMP_COMPCTL_COMPSOURCE_BITS
#define COMP_COMPCTL_QUALSEL_BITS
#define COMP_COMPCTL_SYNCSEL_BITS
#define COMP_COMPSTS_COMPSTS_BITS
#define COMP_DACCTL_DACSOURCE_BITS
#define COMP_DACCTL_FREESOFT_BITS
#define COMP_DACCTL_RAMPSOURCE_BITS
```

#### Typedefs

```
typedef struct \_COMP\_Obj * COMP_Handle
typedef struct \_COMP\_Obj COMP_Obj
```

## Enumerations

```
enum COMP_QualSel_e {  
    COMP_QualSel_Sync,           COMP_QualSel_Qual_2,           COMP_QualSel_Qual_3,  
    COMP_QualSel_Qual_4,         COMP_QualSel_Qual_6,           COMP_QualSel_Qual_7,  
    COMP_QualSel_Qual_8,         COMP_QualSel_Qual_10,         COMP_QualSel_Qual_11,  
    COMP_QualSel_Qual_9,         COMP_QualSel_Qual_12,         COMP_QualSel_Qual_13,  
    COMP_QualSel_Qual_13,        COMP_QualSel_Qual_14,         COMP_QualSel_Qual_15,  
    COMP_QualSel_Qual_16,        COMP_QualSel_Qual_17,        COMP_QualSel_Qual_18,  
    COMP_QualSel_Qual_17,        COMP_QualSel_Qual_18,        COMP_QualSel_Qual_19,  
    COMP_QualSel_Qual_20,        COMP_QualSel_Qual_21,        COMP_QualSel_Qual_22,  
    COMP_QualSel_Qual_21,        COMP_QualSel_Qual_22,        COMP_QualSel_Qual_23,  
    COMP_QualSel_Qual_24,        COMP_QualSel_Qual_25,        COMP_QualSel_Qual_26,  
    COMP_QualSel_Qual_25,        COMP_QualSel_Qual_26,        COMP_QualSel_Qual_27,  
    COMP_QualSel_Qual_28,        COMP_QualSel_Qual_29,        COMP_QualSel_Qual_30,  
    COMP_QualSel_Qual_29,        COMP_QualSel_Qual_30,        COMP_QualSel_Qual_31,  
    COMP_QualSel_Qual_32,        COMP_QualSel_Qual_33 }  
  
enum COMP_RampSyncSrc_e { COMP_RampSyncSrc_PWMSYNC1,  
    COMP_RampSyncSrc_PWMSYNC2, COMP_RampSyncSrc_PWMSYNC3,  
    COMP_RampSyncSrc_PWMSYNC4 }
```

## Functions

```
void COMP_disable (COMP_Handle compHandle)  
void COMP_disableDac (COMP_Handle compHandle)  
void COMP_enable (COMP_Handle compHandle)  
void COMP_enableDac (COMP_Handle compHandle)  
COMP_Handle COMP_init (void *pMemory, const size_t numBytes)  
void COMP_setDacValue (COMP_Handle compHandle, uint16_t dacValue)
```

### 6.1.1 Detailed Description

### 6.1.2 Enumeration Type Documentation

#### 6.1.2.1 enum **COMP\_QualSel\_e**

Enumeration to define the comparator (COMP) output qualification.

##### Enumerator

**COMP\_QualSel\_Sync** Synchronize comparator output.

**COMP\_QualSel\_Qual\_2** Qualify comparator output with 2 cycles.  
**COMP\_QualSel\_Qual\_3** Qualify comparator output with 3 cycles.  
**COMP\_QualSel\_Qual\_4** Qualify comparator output with 4 cycles.  
**COMP\_QualSel\_Qual\_5** Qualify comparator output with 5 cycles.  
**COMP\_QualSel\_Qual\_6** Qualify comparator output with 6 cycles.  
**COMP\_QualSel\_Qual\_7** Qualify comparator output with 7 cycles.  
**COMP\_QualSel\_Qual\_8** Qualify comparator output with 8 cycles.  
**COMP\_QualSel\_Qual\_9** Qualify comparator output with 9 cycles.  
**COMP\_QualSel\_Qual\_10** Qualify comparator output with 10 cycles.  
**COMP\_QualSel\_Qual\_11** Qualify comparator output with 11 cycles.  
**COMP\_QualSel\_Qual\_12** Qualify comparator output with 12 cycles.  
**COMP\_QualSel\_Qual\_13** Qualify comparator output with 13 cycles.  
**COMP\_QualSel\_Qual\_14** Qualify comparator output with 14 cycles.  
**COMP\_QualSel\_Qual\_15** Qualify comparator output with 15 cycles.  
**COMP\_QualSel\_Qual\_16** Qualify comparator output with 16 cycles.  
**COMP\_QualSel\_Qual\_17** Qualify comparator output with 17 cycles.  
**COMP\_QualSel\_Qual\_18** Qualify comparator output with 18 cycles.  
**COMP\_QualSel\_Qual\_19** Qualify comparator output with 19 cycles.  
**COMP\_QualSel\_Qual\_20** Qualify comparator output with 20 cycles.  
**COMP\_QualSel\_Qual\_21** Qualify comparator output with 21 cycles.  
**COMP\_QualSel\_Qual\_22** Qualify comparator output with 22 cycles.  
**COMP\_QualSel\_Qual\_23** Qualify comparator output with 23 cycles.  
**COMP\_QualSel\_Qual\_24** Qualify comparator output with 24 cycles.  
**COMP\_QualSel\_Qual\_25** Qualify comparator output with 25 cycles.  
**COMP\_QualSel\_Qual\_26** Qualify comparator output with 26 cycles.  
**COMP\_QualSel\_Qual\_27** Qualify comparator output with 27 cycles.  
**COMP\_QualSel\_Qual\_28** Qualify comparator output with 28 cycles.  
**COMP\_QualSel\_Qual\_29** Qualify comparator output with 29 cycles.  
**COMP\_QualSel\_Qual\_30** Qualify comparator output with 30 cycles.  
**COMP\_QualSel\_Qual\_31** Qualify comparator output with 31 cycles.  
**COMP\_QualSel\_Qual\_32** Qualify comparator output with 32 cycles.  
**COMP\_QualSel\_Qual\_33** Qualify comparator output with 33 cycles.

#### 6.1.2.2 enum **COMP\_RampSyncSrc\_e**

Enumeration to define the comparator (COMP) ramp generator sync source.

##### Enumerator

**COMP\_RampSyncSrc\_PWMSYNC1** PWMSync1 used as Ramp Sync.  
**COMP\_RampSyncSrc\_PWMSYNC2** PWMSync2 used as Ramp Sync.  
**COMP\_RampSyncSrc\_PWMSYNC3** PWMSync3 used as Ramp Sync.  
**COMP\_RampSyncSrc\_PWMSYNC4** PWMSync4 used as Ramp Sync.

## 6.1.3 Function Documentation

### 6.1.3.1 void COMP\_disable (

**COMP\_Handle** compHandle )

Disables the comparator (COMP)

[1]Parameters in *compHandle* The comparator (COMP) object handle

---

### 6.1.3.2 void COMP\_disableDac (

**COMP\_Handle** compHandle )

Disables the DAC.

[1]Parameters in *compHandle* The comparator (COMP) object handle

---

### 6.1.3.3 void COMP\_enable (

**COMP\_Handle** compHandle )

Enables the comparator (COMP)

[1]Parameters in *compHandle* The comparator (COMP) object handle

---

### 6.1.3.4 void COMP\_enableDac (

**COMP\_Handle** compHandle )

Enables the DAC.

[1]Parameters in *compHandle* The comparator (COMP) object handle

---

### 6.1.3.5 **COMP\_Handle** COMP\_init (

void \* pMemory,

const size\_t numBytes )

Initializes the comparator (COMP) object handle.

[1]Parameters in *pMemory* A pointer to the base address of the COMP registers

---

in *numBytes* The number of bytes allocated for the COMP object, bytes

---

Returns The comparator (COMP) object handle

---



6.1.3.6    void COMP\_setDacValue (  
          **COMP\_Handle** compHandle,  
          uint16\_t dacValue ) [inline]

Sets the DAC's value.

[1]Parameters in *compHandle* The comparator (COMP) object handle

---

in *dacValue* The DAC's value

---

References \_COMP\_Obj\_:DACVAL.

---



## 7 Central Processing Unit (CPU)

Introduction ..... ??  
 API Functions ..... 51 The CPU API provides a set of functions for controlling the central processing unit of this device. This driver is unique in that when initialized `NULL` should be passed as the peripheral base address.

This driver is contained in `f2802x0/common/source/cpu.c`, with `f2802x0/common/include/cpu.h` containing the API definitions for use by applications.

### 7.1 CPU

#### Data Structures

`struct _CPU_Obj_`

#### Macros

```
#define CPU_DBGIER_DLOGINT_BITS
#define CPU_DBGIER_INT10_BITS
#define CPU_DBGIER_INT11_BITS
#define CPU_DBGIER_INT12_BITS
#define CPU_DBGIER_INT13_BITS
#define CPU_DBGIER_INT14_BITS
#define CPU_DBGIER_INT1_BITS
#define CPU_DBGIER_INT2_BITS
#define CPU_DBGIER_INT3_BITS
#define CPU_DBGIER_INT4_BITS
#define CPU_DBGIER_INT5_BITS
#define CPU_DBGIER_INT6_BITS
#define CPU_DBGIER_INT7_BITS
#define CPU_DBGIER_INT8_BITS
#define CPU_DBGIER_INT9_BITS
#define CPU_DBGIER_RTOSINT_BITS
```

```
#define CPU_IER_DLOGINT_BITS
#define CPU_IER_INT10_BITS
#define CPU_IER_INT11_BITS
#define CPU_IER_INT12_BITS
#define CPU_IER_INT13_BITS
#define CPU_IER_INT14_BITS
#define CPU_IER_INT1_BITS
#define CPU_IER_INT2_BITS
#define CPU_IER_INT3_BITS
#define CPU_IER_INT4_BITS
#define CPU_IER_INT5_BITS
#define CPU_IER_INT6_BITS
#define CPU_IER_INT7_BITS
#define CPU_IER_INT8_BITS
#define CPU_IER_INT9_BITS
#define CPU_IER_RTOSINT_BITS
#define CPU_IFR_DLOGINT_BITS
#define CPU_IFR_INT10_BITS
#define CPU_IFR_INT11_BITS
#define CPU_IFR_INT12_BITS
#define CPU_IFR_INT13_BITS
#define CPU_IFR_INT14_BITS
#define CPU_IFR_INT1_BITS
#define CPU_IFR_INT2_BITS
#define CPU_IFR_INT3_BITS
#define CPU_IFR_INT4_BITS
#define CPU_IFR_INT5_BITS
#define CPU_IFR_INT6_BITS
#define CPU_IFR_INT7_BITS
```

```
#define CPU_IFR_INT8_BITS
#define CPU_IFR_INT9_BITS
#define CPU_IFR_RTOSINT_BITS
#define CPU_ST0_C_BITS
#define CPU_ST0_N_BITS
#define CPU_ST0_OVCOVCU_BITS
#define CPU_ST0_OVM_BITS
#define CPU_ST0_PW_BITS
#define CPU_ST0_SXM_BITS
#define CPU_ST0_TC_BITS
#define CPU_ST0_V_BITS
#define CPU_ST0_Z_BITS
#define CPU_ST1_AMODE_BITS
#define CPU_ST1_ARP_BITS
#define CPU_ST1_DBGM_BITS
#define CPU_ST1_EALLOW_BITS
#define CPU_ST1_IDLESTAT_BITS
#define CPU_ST1_INTM_BITS
#define CPU_ST1_LOOP_BITS
#define CPU_ST1_MOM1MAP_BITS
#define CPU_ST1_OBJMODE_BITS
#define CPU_ST1_PAGE0_BITS
#define CPU_ST1_SPA_BITS
#define CPU_ST1_VMAP_BITS
#define CPU_ST1_XF_BITS
#define DINT
#define DISABLE_INTERRUPTS
#define DISABLE_PROTECTED_REGISTER_WRITE_MODE
#define DRTM
```

```
#define EALLOW
#define EDIS
#define EINT
#define ENABLE_INTERRUPTS
#define ENABLE_PROTECTED_REGISTER_WRITE_MODE
#define ERTM
#define ESTOP0
#define IDLE
```

## Typedefs

```
typedef struct _CPU_Obj * CPU_Handle
typedef struct _CPU_Obj CPU_Obj
```

## Enumerations

```
enum CPU_ExtIntNumber_e { CPU_ExtIntNumber_1, CPU_ExtIntNumber_2,
CPU_ExtIntNumber_3 }

enum CPU_IntNumber_e {
CPU_IntNumber_1, CPU_IntNumber_2, CPU_IntNumber_3, CPU_IntNumber_4,
CPU_IntNumber_5, CPU_IntNumber_6, CPU_IntNumber_7, CPU_IntNumber_8,
CPU_IntNumber_9, CPU_IntNumber_10, CPU_IntNumber_11, CPU_IntNumber_12,
CPU_IntNumber_13, CPU_IntNumber_14 }
```

## Functions

```
void CPU_clearIntFlags (CPU_Handle cpuHandle)
void CPU_disableDebugInt (CPU_Handle cpuHandle)
void CPU_disableGlobalInts (CPU_Handle cpuHandle)
void CPU_disableInt (CPU_Handle cpuHandle, const CPU_IntNumber_e intNumber)
void CPU_disableInts (CPU_Handle cpuHandle)
void CPU_disableProtectedRegisterWrite (CPU_Handle cpuHandle)
void CPU_enableDebugInt (CPU_Handle cpuHandle)
void CPU_enableGlobalInts (CPU_Handle cpuHandle)
```

```
void CPU_enableInt (CPU_Handle cpuHandle, const CPU_IntNumber_e intNumber)
```

```
void CPU_enableProtectedRegisterWrite (CPU_Handle cpuHandle)
```

```
CPU_Handle CPU_init (void *pMemory, const size_t numBytes)
```

## Variables

`CPU_Obj` `cpu`

register volatile unsigned int `IER`

register volatile unsigned int `IFR`

### 7.1.1 Detailed Description

### 7.1.2 Enumeration Type Documentation

#### 7.1.2.1 enum `CPU_ExtIntNumber_e`

Enumeration to define the external interrupt numbers.

##### Enumerator

**`CPU_ExtIntNumber_1`** Denotes external interrupt number 1.

**`CPU_ExtIntNumber_2`** Denotes external interrupt number 2.

**`CPU_ExtIntNumber_3`** Denotes external interrupt number 3.

#### 7.1.2.2 enum `CPU_IntNumber_e`

Enumeration to define the interrupt numbers.

##### Enumerator

**`CPU_IntNumber_1`** Denotes interrupt number 1.

**`CPU_IntNumber_2`** Denotes interrupt number 2.

**`CPU_IntNumber_3`** Denotes interrupt number 3.

**`CPU_IntNumber_4`** Denotes interrupt number 4.

**`CPU_IntNumber_5`** Denotes interrupt number 5.

**`CPU_IntNumber_6`** Denotes interrupt number 6.

**`CPU_IntNumber_7`** Denotes interrupt number 7.

**`CPU_IntNumber_8`** Denotes interrupt number 8.

**`CPU_IntNumber_9`** Denotes interrupt number 9.

**`CPU_IntNumber_10`** Denotes interrupt number 10.

**`CPU_IntNumber_11`** Denotes interrupt number 11.

**`CPU_IntNumber_12`** Denotes interrupt number 12.

**`CPU_IntNumber_13`** Denotes interrupt number 13.

**`CPU_IntNumber_14`** Denotes interrupt number 14.

## 7.1.3 Function Documentation

### 7.1.3.1 void CPU\_clearIntFlags (

**CPU\_Handle** cpuHandle )

Clears all interrupt flags.

[1]Parameters in *cpuHandle* The central processing unit (CPU) object handle

---

### 7.1.3.2 void CPU\_disableDebugInt (

**CPU\_Handle** cpuHandle )

Disables the debug interrupt.

[1]Parameters in *cpuHandle* The central processing unit (CPU) object handle

---

### 7.1.3.3 void CPU\_disableGlobalInts (

**CPU\_Handle** cpuHandle )

Disables global interrupts.

[1]Parameters in *cpuHandle* The CPU handle

---

### 7.1.3.4 void CPU\_disableInt (

**CPU\_Handle** cpuHandle,

const **CPU\_IntNumber\_e** intNumber )

Disables a specified interrupt number.

[1]Parameters in *cpuHandle* The central processing unit (CPU) object handle

---

in *intNumber* The interrupt number

---

### 7.1.3.5 void CPU\_disableInts (

**CPU\_Handle** cpuHandle )

Disables all interrupts.

[1]Parameters in *cpuHandle* The central processing unit (CPU) object handle

---



7.1.3.6 void CPU\_disableProtectedRegisterWrite (  
    **CPU\_Handle** cpuHandle )

Disables protected register writes.

[1]Parameters in *cpuHandle* The central processing unit (CPU) object handle

---

7.1.3.7 void CPU\_enableDebugInt (  
    **CPU\_Handle** cpuHandle )

Enables the debug interrupt.

[1]Parameters in *cpuHandle* The CPU handle

---

7.1.3.8 void CPU\_enableGlobalInts (  
    **CPU\_Handle** cpuHandle )

Enables global interrupts.

[1]Parameters in *cpuHandle* The CPU handle

---

7.1.3.9 void CPU\_enableInt (  
    **CPU\_Handle** cpuHandle,  
    const **CPU\_IntNumber\_e** intNumber )

Enables a specified interrupt number.

[1]Parameters in *cpuHandle* The central processing unit (CPU) object handle

---

in *intNumber* The interrupt number

---

7.1.3.10 void CPU\_enableProtectedRegisterWrite (  
    **CPU\_Handle** cpuHandle )

Enables protected register writes.

[1]Parameters in *cpuHandle* The central processing unit (CPU) object handle

---

#### 7.1.3.11 **CPU\_Handle** CPU\_init (

void \* pMemory,

const size\_t numBytes )

Initializes the central processing unit (CPU) object handle.

[1]Parameters in *pMemory* A pointer to the memory for the CPU object

---

in *numBytes* The number of bytes allocated for the CPU object, bytes

---

Returns The central processing unit (CPU) object handle

## 8 Flash

Introduction .....	??
API Functions .....	59

The Flash API contains functions for configuring the wait states as well as run and sleep modes of flash in the device.

**CAUTION:** The flash function(s) should only be run from RAM. Please copy the function to RAM using the memcpy function found in the run time support library before calling any of them.

This driver is contained in `f2802x0/common/source/flash.c`, with `f2802x0/common/include/flash.h` containing the API definitions for use by applications.

### 8.1 FLASH

#### Data Structures

struct `_FLASH_Obj_`

#### Macros

```
#define FLASH_ACTIVE_WAIT_COUNT_DEFAULT
#define FLASH_BASE_ADDR
#define FLASH_FACTIVEWAIT_ACTIVWAIT_BITS
#define FLASH_FBANKWAIT_PAGEWAIT_BITS
#define FLASH_FBANKWAIT_RANDWAIT_BITS
#define FLASH_FOPT_ENPIPE_BITS
#define FLASH_FOTPWAIT_OTPWAIT_BITS
#define FLASH_FPWR_PWR_BITS
#define FLASH_FSTATUS_3VSTAT_BITS
#define FLASH_FSTATUS_ACTIVWAITS_BITS
#define FLASH_FSTATUS_PWRS_BITS
#define FLASH_FSTATUS_STDBYWAITS_BITS
#define FLASH_FSTDBYWAIT_STDBYWAIT_BITS
#define FLASH_STANDBY_WAIT_COUNT_DEFAULT
```

## Typedefs

```
typedef struct _FLASH_Obj_ * FLASH_Handle
```

```
typedef struct _FLASH_Obj_ FLASH_Obj
```

## Enumerations

```
enum FLASH_3VStatus_e { FLASH_3VStatus_InRange, FLASH_3VStatus_OutOfRange }
```

```
enum FLASH_CounterStatus_e { FLASH_CounterStatus_NotCounting, FLASH_CounterStatus_Counting }
```

```
enum FLASH_NumOtpWaitStates_e {  
    FLASH_NumOtpWaitStates_1, FLASH_NumOtpWaitStates_2, FLASH_NumOtpWaitStates_3,  
    FLASH_NumOtpWaitStates_4,  
    FLASH_NumOtpWaitStates_5, FLASH_NumOtpWaitStates_6, FLASH_NumOtpWaitStates_7,  
    FLASH_NumOtpWaitStates_8,  
    FLASH_NumOtpWaitStates_9, FLASH_NumOtpWaitStates_10, FLASH_NumOtpWaitStates_11,  
    FLASH_NumOtpWaitStates_12,  
    FLASH_NumOtpWaitStates_13, FLASH_NumOtpWaitStates_14, FLASH_NumOtpWaitStates_15  
}
```

```
enum FLASH_NumPagedWaitStates_e {  
    FLASH_NumPagedWaitStates_0, FLASH_NumPagedWaitStates_1,  
    FLASH_NumPagedWaitStates_2, FLASH_NumPagedWaitStates_3,  
    FLASH_NumPagedWaitStates_4, FLASH_NumPagedWaitStates_5,  
    FLASH_NumPagedWaitStates_6, FLASH_NumPagedWaitStates_7,  
    FLASH_NumPagedWaitStates_8, FLASH_NumPagedWaitStates_9,  
    FLASH_NumPagedWaitStates_10, FLASH_NumPagedWaitStates_11,  
    FLASH_NumPagedWaitStates_12, FLASH_NumPagedWaitStates_13,  
    FLASH_NumPagedWaitStates_14, FLASH_NumPagedWaitStates_15 }
```

```
enum FLASH_NumRandomWaitStates_e {  
    FLASH_NumRandomWaitStates_1, FLASH_NumRandomWaitStates_2,  
    FLASH_NumRandomWaitStates_3, FLASH_NumRandomWaitStates_4,  
    FLASH_NumRandomWaitStates_5, FLASH_NumRandomWaitStates_6,  
    FLASH_NumRandomWaitStates_7, FLASH_NumRandomWaitStates_8,  
    FLASH_NumRandomWaitStates_9, FLASH_NumRandomWaitStates_10,  
    FLASH_NumRandomWaitStates_11, FLASH_NumRandomWaitStates_12,  
    FLASH_NumRandomWaitStates_13, FLASH_NumRandomWaitStates_14,  
    FLASH_NumRandomWaitStates_15 }
```

```
enum FLASH_PowerMode_e { FLASH_PowerMode_PumpAndBankSleep, FLASH_PowerMode_PumpAndBankStandby, FLASH_PowerMode_PumpAndBankActive }
```

## Functions

```
void FLASH_clear3VStatus (FLASH_Handle flashHandle)
```

```
void FLASH_disablePipelineMode (FLASH_Handle flashHandle)
```

```

void FLASH_enablePipelineMode (FLASH_Handle flashHandle)

FLASH_3VStatus_e FLASH_get3VStatus (FLASH_Handle flashHandle)

uint16_t FLASH_getActiveWaitCount (FLASH_Handle flashHandle)

FLASH_CounterStatus_e FLASH_getActiveWaitStatus (FLASH_Handle flashHandle)

FLASH_PowerMode_e FLASH_getPowerMode (FLASH_Handle flashHandle)

uint16_t FLASH_getStandbyWaitCount (FLASH_Handle flashHandle)

FLASH_CounterStatus_e FLASH_getStandbyWaitStatus (FLASH_Handle flashHandle)

FLASH_Handle FLASH_init (void *pMemory, const size_t numBytes)

void FLASH_setActiveWaitCount (FLASH_Handle flashHandle, const uint16_t count)

void FLASH_setNumPagedReadWaitStates (FLASH_Handle flashHandle, const
FLASH_NumPagedWaitStates_e numStates)

void FLASH_setNumRandomReadWaitStates (FLASH_Handle flashHandle, const
FLASH_NumRandomWaitStates_e numStates)

void FLASH_setOtpWaitStates (FLASH_Handle flashHandle, const FLASH_NumOtpWaitStates_e
numStates)

void FLASH_setPowerMode (FLASH_Handle flashHandle, const FLASH_PowerMode_e mode)

void FLASH_setStandbyWaitCount (FLASH_Handle flashHandle, const uint16_t count)

void FLASH_setup (FLASH_Handle flashHandle)

```

### 8.1.1 Detailed Description

### 8.1.2 Enumeration Type Documentation

#### 8.1.2.1 enum **FLASH\_3VStatus\_e**

Enumeration to define the 3V status.

##### Enumerator

***FLASH\_3VStatus\_InRange*** Denotes the 3V flash voltage is in range.

***FLASH\_3VStatus\_OutOfRange*** Denotes the 3V flash voltage went out of range.

#### 8.1.2.2 enum **FLASH\_CounterStatus\_e**

Enumeration to define the counter status.

##### Enumerator

***FLASH\_CounterStatus\_NotCounting*** Denotes the flash counter is not counting.

***FLASH\_CounterStatus\_Counting*** Denotes the flash counter is counting.

### 8.1.2.3 enum **FLASH\_NumOtpWaitStates\_e**

Enumeration to define the number of one-time programmable wait states.

#### Enumerator

- FLASH\_NumOtpWaitStates\_1** Denotes the number of one-time programmable (OTP) wait states is 1.
- FLASH\_NumOtpWaitStates\_2** Denotes the number of one-time programmable (OTP) wait states is 2.
- FLASH\_NumOtpWaitStates\_3** Denotes the number of one-time programmable (OTP) wait states is 3.
- FLASH\_NumOtpWaitStates\_4** Denotes the number of one-time programmable (OTP) wait states is 4.
- FLASH\_NumOtpWaitStates\_5** Denotes the number of one-time programmable (OTP) wait states is 5.
- FLASH\_NumOtpWaitStates\_6** Denotes the number of one-time programmable (OTP) wait states is 6.
- FLASH\_NumOtpWaitStates\_7** Denotes the number of one-time programmable (OTP) wait states is 7.
- FLASH\_NumOtpWaitStates\_8** Denotes the number of one-time programmable (OTP) wait states is 8.
- FLASH\_NumOtpWaitStates\_9** Denotes the number of one-time programmable (OTP) wait states is 9.
- FLASH\_NumOtpWaitStates\_10** Denotes the number of one-time programmable (OTP) wait states is 10.
- FLASH\_NumOtpWaitStates\_11** Denotes the number of one-time programmable (OTP) wait states is 11.
- FLASH\_NumOtpWaitStates\_12** Denotes the number of one-time programmable (OTP) wait states is 12.
- FLASH\_NumOtpWaitStates\_13** Denotes the number of one-time programmable (OTP) wait states is 13.
- FLASH\_NumOtpWaitStates\_14** Denotes the number of one-time programmable (OTP) wait states is 14.
- FLASH\_NumOtpWaitStates\_15** Denotes the number of one-time programmable (OTP) wait states is 15.

### 8.1.2.4 enum **FLASH\_NumPagedWaitStates\_e**

Enumeration to define the number of paged wait states.

#### Enumerator

- FLASH\_NumPagedWaitStates\_0** Denotes the number of paged read wait states is 0.
- FLASH\_NumPagedWaitStates\_1** Denotes the number of paged read wait states is 1.
- FLASH\_NumPagedWaitStates\_2** Denotes the number of paged read wait states is 2.
- FLASH\_NumPagedWaitStates\_3** Denotes the number of paged read wait states is 3.
- FLASH\_NumPagedWaitStates\_4** Denotes the number of paged read wait states is 4.
- FLASH\_NumPagedWaitStates\_5** Denotes the number of paged read wait states is 5.
- FLASH\_NumPagedWaitStates\_6** Denotes the number of paged read wait states is 6.

**FLASH\_NumPagedWaitStates\_7** Denotes the number of paged read wait states is 7.  
**FLASH\_NumPagedWaitStates\_8** Denotes the number of paged read wait states is 8.  
**FLASH\_NumPagedWaitStates\_9** Denotes the number of paged read wait states is 9.  
**FLASH\_NumPagedWaitStates\_10** Denotes the number of paged read wait states is 10.  
**FLASH\_NumPagedWaitStates\_11** Denotes the number of paged read wait states is 11.  
**FLASH\_NumPagedWaitStates\_12** Denotes the number of paged read wait states is 12.  
**FLASH\_NumPagedWaitStates\_13** Denotes the number of paged read wait states is 13.  
**FLASH\_NumPagedWaitStates\_14** Denotes the number of paged read wait states is 14.  
**FLASH\_NumPagedWaitStates\_15** Denotes the number of paged read wait states is 15.

#### 8.1.2.5 enum **FLASH\_NumRandomWaitStates\_e**

Enumeration to define the number of random wait states.

##### Enumerator

**FLASH\_NumRandomWaitStates\_1** Denotes the number of random read wait states is 1.  
**FLASH\_NumRandomWaitStates\_2** Denotes the number of random read wait states is 2.  
**FLASH\_NumRandomWaitStates\_3** Denotes the number of random read wait states is 3.  
**FLASH\_NumRandomWaitStates\_4** Denotes the number of random read wait states is 4.  
**FLASH\_NumRandomWaitStates\_5** Denotes the number of random read wait states is 5.  
**FLASH\_NumRandomWaitStates\_6** Denotes the number of random read wait states is 6.  
**FLASH\_NumRandomWaitStates\_7** Denotes the number of random read wait states is 7.  
**FLASH\_NumRandomWaitStates\_8** Denotes the number of random read wait states is 8.  
**FLASH\_NumRandomWaitStates\_9** Denotes the number of random read wait states is 9.  
**FLASH\_NumRandomWaitStates\_10** Denotes the number of random read wait states is 10.  
**FLASH\_NumRandomWaitStates\_11** Denotes the number of random read wait states is 11.  
**FLASH\_NumRandomWaitStates\_12** Denotes the number of random read wait states is 12.  
**FLASH\_NumRandomWaitStates\_13** Denotes the number of random read wait states is 13.  
**FLASH\_NumRandomWaitStates\_14** Denotes the number of random read wait states is 14.  
**FLASH\_NumRandomWaitStates\_15** Denotes the number of random read wait states is 15.

#### 8.1.2.6 enum **FLASH\_PowerMode\_e**

Enumeration to define the power modes.

##### Enumerator

**FLASH\_PowerMode\_PumpAndBankSleep** Denotes a pump and bank sleep power mode.  
**FLASH\_PowerMode\_PumpAndBankStandby** Denotes a pump and bank standby power mode.  
**FLASH\_PowerMode\_PumpAndBankActive** Denotes a pump and bank active power mode.

## 8.1.3 Function Documentation

### 8.1.3.1 void FLASH\_clear3VStatus (

**FLASH\_Handle** flashHandle )

Clears the 3V status.

[1]Parameters in *flashHandle* The flash (FLASH) object handle

---

### 8.1.3.2 void FLASH\_disablePipelineMode (

**FLASH\_Handle** flashHandle )

Disables the pipeline mode.

[1]Parameters in *flashHandle* The flash (FLASH) object handle

---

### 8.1.3.3 void FLASH\_enablePipelineMode (

**FLASH\_Handle** flashHandle )

Enables the pipeline mode.

[1]Parameters in *flashHandle* The flash (FLASH) object handle

---

### 8.1.3.4 **FLASH\_3VStatus\_e** FLASH\_get3VStatus (

**FLASH\_Handle** flashHandle )

Gets the 3V status.

[1]Parameters in *flashHandle* The flash (FLASH) object handle

---

Returns The 3V status

---

### 8.1.3.5 uint16\_t FLASH\_getActiveWaitCount (

**FLASH\_Handle** flashHandle )

Gets the active wait count.

[1]Parameters in *flashHandle* The flash (FLASH) object handle

---

Returns The active wait count

---





722.7

*in numBytes* The number of bytes allocated for the FLASH object, bytes

---

Returns The flash (FLASH) object handle

---

8.1.3.11 void FLASH\_setActiveWaitCount (

**FLASH\_Handle** flashHandle,

const uint16\_t count )

Sets the active wait count.

[1]Parameters *in flashHandle* The flash (FLASH) object handle

---

*in count* The active wait count

---

8.1.3.12 void FLASH\_setNumPagedReadWaitStates (

**FLASH\_Handle** flashHandle,

const **FLASH\_NumPagedWaitStates\_e** numStates )

Sets the number of paged read wait states.

[1]Parameters *in flashHandle* The flash (FLASH) object handle

---

*in numStates* The number of paged read wait states

---

8.1.3.13 void FLASH\_setNumRandomReadWaitStates (

**FLASH\_Handle** flashHandle,

const **FLASH\_NumRandomWaitStates\_e** numStates )

Sets the number of random read wait states.

[1]Parameters *in flashHandle* The flash (FLASH) object handle

---

*in numStates* The number of random read wait states

---

8.1.3.14 void FLASH\_setOtpWaitStates (

**FLASH\_Handle** flashHandle,

---

const **FLASH\_NumOtpWaitStates\_e** numStates )

Sets the number of one-time programmable (OTP) wait states.

[1]Parameters in *flashHandle* The flash (FLASH) object handle

---

in *numStates* The number of one-time programmable (OTP) wait states

---

8.1.3.15 void FLASH\_setPowerMode (

**FLASH\_Handle** flashHandle,

const **FLASH\_PowerMode\_e** mode )

Sets the power mode.

[1]Parameters in *flashHandle* The flash (FLASH) object handle

---

in *mode* The power mode

---

8.1.3.16 void FLASH\_setStandbyWaitCount (

**FLASH\_Handle** flashHandle,

const uint16\_t count )

Sets the standby wait count.

[1]Parameters in *flashHandle* The flash (FLASH) object handle

---

in *count* The standby wait count

---

8.1.3.17 void FLASH\_setup (

**FLASH\_Handle** flashHandle )

Setup flash for optimal performance.

[1]Parameters in *flashHandle* The flash (FLASH) object handle

---



## 9 General Purpose Input/Output (GPIO)

[Introduction](#) .....??

[API Functions](#) 69 The GPIO API provides functions to configure and control the GPIO of this device. Pins can be set as inputs, outputs, or a peripheral function and their value set or read via this API.

This driver is contained in `f2802x0/common/source/gpio.c`, with `f2802x0/common/include/gpio.h` containing the API definitions for use by applications.

### 9.1 GPIO

#### Data Structures

`struct _GPIO_Obj_`

#### Macros

```
#define GPIO_BASE_ADDR
#define GPIO_GPMUX_CONFIG_BITS
#define GPIO_GPMUX_NUMGPIOs_PER_REG
#define GPIO_GPxCtrl_QUALPRDx_BITS
#define GPIO_GPxCtrl_QUALPRDx_NUMBITS_PER_REG
#define GPIO_GPxQSELx_NUMGPIOs_PER_REG
#define GPIO_GPxQSELy_GPIOx_BITS
```

#### Typedefs

```
typedef struct _GPIO_Obj_ * GPIO_Handle
typedef struct _GPIO_Obj_ GPIO_Obj
```

#### Enumerations

```
enum GPIO_Direction_e { GPIO_Direction_Input, GPIO_Direction_Output }

enum GPIO_Mode_e {
    GPIO_0_Mode_GeneralPurpose,    GPIO_0_Mode_EPWM1A,    GPIO_0_Mode_Rsvd_2,
    GPIO_0_Mode_Rsvd_3,            GPIO_1_Mode_EPWM1B,    GPIO_1_Mode_Rsvd_2,
    GPIO_1_Mode_GeneralPurpose,
```

```
GPIO_1_Mode_COMP1OUT,
GPIO_2_Mode_GeneralPurpose, GPIO_2_Mode_EPWM2A, GPIO_2_Mode_Rsvd_2,
GPIO_2_Mode_Rsvd_3,
GPIO_3_Mode_GeneralPurpose, GPIO_3_Mode_EPWM2B, GPIO_3_Mode_Rsvd_2,
GPIO_3_Mode_COMP2OUT,
GPIO_4_Mode_GeneralPurpose, GPIO_4_Mode_EPWM3A, GPIO_4_Mode_Rsvd_2,
GPIO_4_Mode_Rsvd_3,
GPIO_5_Mode_GeneralPurpose, GPIO_5_Mode_EPWM3B, GPIO_5_Mode_Rsvd_2,
GPIO_5_Mode_ECAP1,
GPIO_6_Mode_GeneralPurpose, GPIO_6_Mode_Rsvd_1, GPIO_6_Mode_EPWMSYNCO,
GPIO_6_Mode_EPWMSYNCO,
GPIO_7_Mode_GeneralPurpose, GPIO_7_Mode_Rsvd_1, GPIO_7_Mode_SCIRXDA,
GPIO_7_Mode_Rsvd_3,
GPIO_12_Mode_GeneralPurpose, GPIO_12_Mode_TZ1_NOT, GPIO_12_Mode_SCITXDA,
GPIO_12_Mode_Rsvd_3,
GPIO_16_Mode_GeneralPurpose, GPIO_16_Mode_SPISIMOA, GPIO_16_Mode_Rsvd_2,
GPIO_16_Mode_TZ2_NOT,
GPIO_17_Mode_GeneralPurpose, GPIO_17_Mode_SPISOMIA, GPIO_17_Mode_Rsvd_2,
GPIO_17_Mode_TZ3_NOT,
GPIO_18_Mode_GeneralPurpose, GPIO_18_Mode_SPICLKA, GPIO_18_Mode_SCITXDA,
GPIO_18_Mode_XCLKOUT,
GPIO_19_Mode_GeneralPurpose, GPIO_19_Mode_SPISTEA_NOT, GPIO_19_Mode_SCIRXDA,
GPIO_19_Mode_ECAP1,
GPIO_28_Mode_GeneralPurpose, GPIO_28_Mode_SCIRXDA, GPIO_28_Mode_SDDA,
GPIO_28_Mode_TZ2_NOT,
GPIO_29_Mode_GeneralPurpose, GPIO_29_Mode_SCITXDA, GPIO_29_Mode_SCLA,
GPIO_29_Mode_TZ3_NOT,
GPIO_32_Mode_GeneralPurpose, GPIO_32_Mode_SDAA, GPIO_32_Mode_EPWMSYNCO,
GPIO_32_Mode_ADCSOCBO_NOT,
GPIO_33_Mode_GeneralPurpose, GPIO_33_Mode_SCLA, GPIO_33_Mode_EPWMSYNCO,
GPIO_33_Mode_ADCSOCBO_NOT,
GPIO_34_Mode_GeneralPurpose, GPIO_34_Mode_COMP2OUT, GPIO_34_Mode_Rsvd_2,
GPIO_34_Mode_Rsvd_3,
GPIO_35_Mode_JTAG_TDI, GPIO_35_Mode_Rsvd_1, GPIO_35_Mode_Rsvd_2,
GPIO_35_Mode_Rsvd_3,
GPIO_36_Mode_JTAG_TMS, GPIO_36_Mode_Rsvd_1, GPIO_36_Mode_Rsvd_2,
GPIO_36_Mode_Rsvd_3,
GPIO_37_Mode_JTAG_TDO, GPIO_37_Mode_Rsvd_1, GPIO_37_Mode_Rsvd_2,
GPIO_37_Mode_Rsvd_3,
GPIO_38_Mode_JTAG_TCK, GPIO_38_Mode_Rsvd_1, GPIO_38_Mode_Rsvd_2,
GPIO_38_Mode_Rsvd_3 }

enum GPIO_Number_e {
GPIO_Number_0, GPIO_Number_1, GPIO_Number_2, GPIO_Number_3,
GPIO_Number_4, GPIO_Number_5, GPIO_Number_6, GPIO_Number_7,
GPIO_Rsvd_8, GPIO_Rsvd_9, GPIO_Rsvd_10, GPIO_Rsvd_11,
GPIO_Number_12, GPIO_Rsvd_13, GPIO_Rsvd_14, GPIO_Rsvd_15,
GPIO_Number_16, GPIO_Number_17, GPIO_Number_18, GPIO_Number_19,
GPIO_Rsvd_20, GPIO_Rsvd_21, GPIO_Rsvd_22, GPIO_Rsvd_23,
GPIO_Rsvd_24, GPIO_Rsvd_25, GPIO_Rsvd_26, GPIO_Rsvd_27,
GPIO_Number_28, GPIO_Number_29, GPIO_Rsvd_30, GPIO_Rsvd_31,
GPIO_Number_32, GPIO_Number_33, GPIO_Number_34, GPIO_Number_35,
GPIO_Number_36, GPIO_Number_37, GPIO_Number_38 }
```

```
enum GPIO_Port_e { GPIO_Port_A, GPIO_Port_B }

enum GPIO_PullUp_e { GPIO_PullUp_Enable, GPIO_PullUp_Disable }

enum GPIO_Qual_e { GPIO_Qual_Sync, GPIO_Qual_Sample_3, GPIO_Qual_Sample_6,
GPIO_Qual_ASync }
```

## Functions

```
uint16_t GPIO_getData (GPIO_Handle gpioHandle, const GPIO_Number_e gpioNumber)

uint32_t GPIO_getPortData (GPIO_Handle gpioHandle, const GPIO_Port_e gpioPort)

GPIO_Handle GPIO_init (void *pMemory, const size_t numBytes)

void GPIO_lpmSelect (GPIO_Handle gpioHandle, const GPIO_Number_e gpioNumber)

void GPIO_setDirection (GPIO_Handle gpioHandle, const GPIO_Number_e gpioNumber, const
GPIO_Direction_e direction)

void GPIO_setExtInt (GPIO_Handle gpioHandle, const GPIO_Number_e gpioNumber, const
CPU_ExtIntNumber_e intNumber)

void GPIO_setHigh (GPIO_Handle gpioHandle, const GPIO_Number_e gpioNumber)

void GPIO_setLow (GPIO_Handle gpioHandle, const GPIO_Number_e gpioNumber)

void GPIO_setMode (GPIO_Handle gpioHandle, const GPIO_Number_e gpioNumber, const
GPIO_Mode_e mode)

void GPIO_setPortData (GPIO_Handle gpioHandle, const GPIO_Port_e gpioPort, const uint32_t
data)

void GPIO_setPullUp (GPIO_Handle gpioHandle, const GPIO_Number_e gpioNumber, const
GPIO_PullUp_e pullUp)

void GPIO_setQualification (GPIO_Handle gpioHandle, const GPIO_Number_e gpioNumber, const
GPIO_Qual_e qualification)

void GPIO_setQualificationPeriod (GPIO_Handle gpioHandle, const GPIO_Number_e gpioNum-
ber, const uint8_t period)

void GPIO_toggle (GPIO_Handle gpioHandle, const GPIO_Number_e gpioNumber)
```

### 9.1.1 Detailed Description

### 9.1.2 Enumeration Type Documentation

#### 9.1.2.1 enum **GPIO\_Direction\_e**

Enumeration to define the general purpose I/O (GPIO) directions.

**Enumerator**

**GPIO\_Direction\_Input** Denotes an input direction.  
**GPIO\_Direction\_Output** Denotes an output direction.

9.1.2.2 enum **GPIO\_Mode\_e**

Enumeration to define the general purpose I/O (GPIO) modes for each pin.

**Enumerator**

**GPIO\_0\_Mode\_GeneralPurpose** Denotes a general purpose function.  
**GPIO\_0\_Mode\_EPWM1A** Denotes a EPWM1A function.  
**GPIO\_0\_Mode\_Rsvd\_2** Denotes a reserved function.  
**GPIO\_0\_Mode\_Rsvd\_3** Denotes a reserved function.  
**GPIO\_1\_Mode\_GeneralPurpose** Denotes a general purpose function.  
**GPIO\_1\_Mode\_EPWM1B** Denotes a EPWM1B function.  
**GPIO\_1\_Mode\_Rsvd\_2** Denotes a reserved function.  
**GPIO\_1\_Mode\_COMP1OUT** Denotes a COMP1OUT function.  
**GPIO\_2\_Mode\_GeneralPurpose** Denotes a general purpose function.  
**GPIO\_2\_Mode\_EPWM2A** Denotes a EPWM2A function.  
**GPIO\_2\_Mode\_Rsvd\_2** Denotes a reserved function.  
**GPIO\_2\_Mode\_Rsvd\_3** Denotes a reserved function.  
**GPIO\_3\_Mode\_GeneralPurpose** Denotes a general purpose function.  
**GPIO\_3\_Mode\_EPWM2B** Denotes a EPWM2B function.  
**GPIO\_3\_Mode\_Rsvd\_2** Denotes a reserved function.  
**GPIO\_3\_Mode\_COMP2OUT** Denotes a COMP2OUT function.  
**GPIO\_4\_Mode\_GeneralPurpose** Denotes a general purpose function.  
**GPIO\_4\_Mode\_EPWM3A** Denotes a EPWM3A function.  
**GPIO\_4\_Mode\_Rsvd\_2** Denotes a reserved function.  
**GPIO\_4\_Mode\_Rsvd\_3** Denotes a reserved function.  
**GPIO\_5\_Mode\_GeneralPurpose** Denotes a general purpose function.  
**GPIO\_5\_Mode\_EPWM3B** Denotes a EPWM3B function.  
**GPIO\_5\_Mode\_Rsvd\_2** Denotes a reserved function.  
**GPIO\_5\_Mode\_ECAP1** Denotes a ECAP1 function.  
**GPIO\_6\_Mode\_GeneralPurpose** Denotes a general purpose function.  
**GPIO\_6\_Mode\_Rsvd\_1** Denotes a reserved function.  
**GPIO\_6\_Mode\_EPWMSYNCl** Denotes a EPWMSYNCl function.  
**GPIO\_6\_Mode\_EPWMSYNCO** Denotes a EPWMSYNCO function.  
**GPIO\_7\_Mode\_GeneralPurpose** Denotes a general purpose function.  
**GPIO\_7\_Mode\_Rsvd\_1** Denotes a reserved function.  
**GPIO\_7\_Mode\_SCIRXDA** Denotes a SCIRXDA function.  
**GPIO\_7\_Mode\_Rsvd\_3** Denotes a reserved function.  
**GPIO\_12\_Mode\_GeneralPurpose** Denotes a general purpose function.  
**GPIO\_12\_Mode\_TZ1\_NOT** Denotes a TZ1\_NOT function.  
**GPIO\_12\_Mode\_SCITXDA** Denotes a SCITXDA function.  
**GPIO\_12\_Mode\_Rsvd\_3** Denotes a reserved function.



**GPIO\_16\_Mode\_GeneralPurpose** Denotes a general purpose function.  
**GPIO\_16\_Mode\_SPISIMOA** Denotes a SPISIMOA function.  
**GPIO\_16\_Mode\_Rsvd\_2** Denotes a reserved function.  
**GPIO\_16\_Mode\_TZ2\_NOT** Denotes a TZ2\_NOT function.  
**GPIO\_17\_Mode\_GeneralPurpose** Denotes a general purpose function.  
**GPIO\_17\_Mode\_SPISOMIA** Denotes a SPISOMIA function.  
**GPIO\_17\_Mode\_Rsvd\_2** Denotes a reserved function.  
**GPIO\_17\_Mode\_TZ3\_NOT** Denotes a TZ3\_NOT function.  
**GPIO\_18\_Mode\_GeneralPurpose** Denotes a general purpose function.  
**GPIO\_18\_Mode\_SPICLKA** Denotes a SPICLKA function.  
**GPIO\_18\_Mode\_SCITXDA** Denotes a SCITXDA function.  
**GPIO\_18\_Mode\_XCLKOUT** Denotes a XCLKOUT function.  
**GPIO\_19\_Mode\_GeneralPurpose** Denotes a general purpose function.  
**GPIO\_19\_Mode\_SPISTEA\_NOT** Denotes a SPISTEA\_NOT function.  
**GPIO\_19\_Mode\_SCIRXDA** Denotes a SCIRXDA function.  
**GPIO\_19\_Mode\_ECAP1** Denotes a ECAP1 function.  
**GPIO\_28\_Mode\_GeneralPurpose** Denotes a general purpose function.  
**GPIO\_28\_Mode\_SCIRXDA** Denotes a SCIRXDA function.  
**GPIO\_28\_Mode\_SDDA** Denotes a SDDA function.  
**GPIO\_28\_Mode\_TZ2\_NOT** Denotes a TZ2\_NOT function.  
**GPIO\_29\_Mode\_GeneralPurpose** Denotes a general purpose function.  
**GPIO\_29\_Mode\_SCITXDA** Denotes a SCITXDA function.  
**GPIO\_29\_Mode\_SCLA** Denotes a SCLA function.  
**GPIO\_29\_Mode\_TZ3\_NOT** Denotes a TZ2\_NOT function.  
**GPIO\_32\_Mode\_GeneralPurpose** Denotes a general purpose function.  
**GPIO\_32\_Mode\_SDAA** Denotes a SDDA function.  
**GPIO\_32\_Mode\_EPWMSYNCI** Denotes a EPWMSYNCI function.  
**GPIO\_32\_Mode\_ADCSOCOA\_NOT** Denotes a ADCSOCOA\_NOT function.  
**GPIO\_33\_Mode\_GeneralPurpose** Denotes a general purpose function.  
**GPIO\_33\_Mode\_SCLA** Denotes a SCLA function.  
**GPIO\_33\_Mode\_EPWMSYNCO** Denotes a EPWMSYNCO function.  
**GPIO\_33\_Mode\_ADCSOCBO\_NOT** Denotes a ADCSOCBO\_NOT function.  
**GPIO\_34\_Mode\_GeneralPurpose** Denotes a general purpose function.  
**GPIO\_34\_Mode\_COMP2OUT** Denotes a COMP2OUT function.  
**GPIO\_34\_Mode\_Rsvd\_2** Denotes a reserved function.  
**GPIO\_34\_Mode\_Rsvd\_3** Denotes a reserved function.  
**GPIO\_35\_Mode\_JTAG\_TDI** Denotes a JTAG\_TDI function.  
**GPIO\_35\_Mode\_Rsvd\_1** Denotes a reserved function.  
**GPIO\_35\_Mode\_Rsvd\_2** Denotes a reserved function.  
**GPIO\_35\_Mode\_Rsvd\_3** Denotes a reserved function.  
**GPIO\_36\_Mode\_JTAG\_TMS** Denotes a JTAG\_TMS function.  
**GPIO\_36\_Mode\_Rsvd\_1** Denotes a reserved function.  
**GPIO\_36\_Mode\_Rsvd\_2** Denotes a reserved function.  
**GPIO\_36\_Mode\_Rsvd\_3** Denotes a reserved function.  
**GPIO\_37\_Mode\_JTAG\_TDO** Denotes a JTAG\_TDO function.  
**GPIO\_37\_Mode\_Rsvd\_1** Denotes a reserved function.

**GPIO\_37\_Mode\_Rsvd\_2** Denotes a reserved function.  
**GPIO\_37\_Mode\_Rsvd\_3** Denotes a reserved function.  
**GPIO\_38\_Mode\_JTAG\_TCK** Denotes a JTAG\_TCK function.  
**GPIO\_38\_Mode\_Rsvd\_1** Denotes a reserved function.  
**GPIO\_38\_Mode\_Rsvd\_2** Denotes a reserved function.  
**GPIO\_38\_Mode\_Rsvd\_3** Denotes a reserved function.

### 9.1.2.3 enum **GPIO\_Number\_e**

Enumeration to define the general purpose I/O (GPIO) numbers.

#### Enumerator

**GPIO\_Number\_0** Denotes GPIO number 0.  
**GPIO\_Number\_1** Denotes GPIO number 1.  
**GPIO\_Number\_2** Denotes GPIO number 2.  
**GPIO\_Number\_3** Denotes GPIO number 3.  
**GPIO\_Number\_4** Denotes GPIO number 4.  
**GPIO\_Number\_5** Denotes GPIO number 5.  
**GPIO\_Number\_6** Denotes GPIO number 6.  
**GPIO\_Number\_7** Denotes GPIO number 7.  
**GPIO\_Rsvd\_8** This GPIO not present.  
**GPIO\_Rsvd\_9** This GPIO not present.  
**GPIO\_Rsvd\_10** This GPIO not present.  
**GPIO\_Rsvd\_11** This GPIO not present.  
**GPIO\_Number\_12** Denotes GPIO number 12.  
**GPIO\_Rsvd\_13** This GPIO not present.  
**GPIO\_Rsvd\_14** This GPIO not present.  
**GPIO\_Rsvd\_15** This GPIO not present.  
**GPIO\_Number\_16** Denotes GPIO number 16.  
**GPIO\_Number\_17** Denotes GPIO number 17.  
**GPIO\_Number\_18** Denotes GPIO number 18.  
**GPIO\_Number\_19** Denotes GPIO number 19.  
**GPIO\_Rsvd\_20** This GPIO not present.  
**GPIO\_Rsvd\_21** This GPIO not present.  
**GPIO\_Rsvd\_22** This GPIO not present.  
**GPIO\_Rsvd\_23** This GPIO not present.  
**GPIO\_Rsvd\_24** This GPIO not present.  
**GPIO\_Rsvd\_25** This GPIO not present.  
**GPIO\_Rsvd\_26** This GPIO not present.  
**GPIO\_Rsvd\_27** This GPIO not present.  
**GPIO\_Number\_28** Denotes GPIO number 28.  
**GPIO\_Number\_29** Denotes GPIO number 29.  
**GPIO\_Rsvd\_30** This GPIO not present.  
**GPIO\_Rsvd\_31** This GPIO not present.  
**GPIO\_Number\_32** Denotes GPIO number 32.

**GPIO\_Number\_33** Denotes GPIO number 33.  
**GPIO\_Number\_34** Denotes GPIO number 34.  
**GPIO\_Number\_35** Denotes GPIO number 35.  
**GPIO\_Number\_36** Denotes GPIO number 36.  
**GPIO\_Number\_37** Denotes GPIO number 37.  
**GPIO\_Number\_38** Denotes GPIO number 38.

#### 9.1.2.4 enum **GPIO\_Port\_e**

Enumeration to define the general purpose I/O (GPIO) ports.

##### Enumerator

**GPIO\_Port\_A** GPIO Port A.  
**GPIO\_Port\_B** GPIO Port B.

#### 9.1.2.5 enum **GPIO\_PullUp\_e**

Enumeration to define the general purpose I/O (GPIO) pull ups.

##### Enumerator

**GPIO\_PullUp\_Enable** Denotes pull up will be enabled.  
**GPIO\_PullUp\_Disable** Denotes pull up will be disabled.

#### 9.1.2.6 enum **GPIO\_Qual\_e**

Enumeration to define the general purpose I/O (GPIO) qualification.

##### Enumerator

**GPIO\_Qual\_Sync** Denotes input will be synchronized to SYSCLK.  
**GPIO\_Qual\_Sample\_3** Denotes input is qualified with 3 samples.  
**GPIO\_Qual\_Sample\_6** Denotes input is qualified with 6 samples.  
**GPIO\_Qual\_ASync** Denotes input is asynchronous.

### 9.1.3 Function Documentation

#### 9.1.3.1 uint16\_t GPIO\_getData (

**GPIO\_Handle** gpioHandle,  
const **GPIO\_Number\_e** gpioNumber )

Returns the data value present on a pin (either input or output)

[1]Parameters in *gpioHandle* The general purpose I/O (GPIO) object handle

*in gpioNumber* The GPIO number

---

Returns The boolean state of a pin (high/low)

#### 9.1.3.2 uint32\_t GPIO\_getPortData (

**GPIO\_Handle** gpioHandle,

const **GPIO\_Port\_e** gpioPort )

Returns the data value present on a GPIO port.

[1]Parameters *in gpioHandle* The general purpose I/O (GPIO) object handle

---

*in gpioPort* The GPIO port

---

Returns The data values for the specified port

#### 9.1.3.3 **GPIO\_Handle** GPIO\_init (

void \* pMemory,

const size\_t numBytes )

Initializes the general purpose I/O (GPIO) object handle.

[1]Parameters *in pMemory* A pointer to the base address of the GPIO registers

---

*in numBytes* The number of bytes allocated for the GPIO object, bytes

---

Returns The general purpose I/O (GPIO) object handle

#### 9.1.3.4 void GPIO\_lpmSelect (

**GPIO\_Handle** gpioHandle,

const **GPIO\_Number\_e** gpioNumber )

Selects a gpio pin to wake up device from STANDBY and HALT LPM.

[1]Parameters *in gpioHandle* The general purpose I/O (GPIO) object handle

---

*in gpioNumber* The GPIO number

---

#### 9.1.3.5 void GPIO\_setDirection (

**GPIO\_Handle** gpioHandle,

```
const GPIO_Number_e gpioNumber,  
const GPIO_Direction_e direction )
```

Sets the general purpose I/O (GPIO) signal direction.

[1]Parameters in *gpioHandle* The general purpose I/O (GPIO) object handle

---

in *gpioNumber* The GPIO number

---

in *direction* The signal direction

---

```
9.1.3.6 void GPIO_setExtInt (  
    GPIO_Handle gpioHandle,  
    const GPIO_Number_e gpioNumber,  
    const CPU_ExtIntNumber_e intNumber )
```

Sets the general purpose I/O (GPIO) external interrupt number.

[1]Parameters in *gpioHandle* The general purpose I/O (GPIO) object handle

---

in *gpioNumber* The GPIO number

---

in *intNumber* The interrupt number

---

```
9.1.3.7 void GPIO_setHigh (  
    GPIO_Handle gpioHandle,  
    const GPIO_Number_e gpioNumber )
```

Sets the specified general purpose I/O (GPIO) signal high.

[1]Parameters in *gpioHandle* The general purpose I/O (GPIO) object handle

---

in *gpioNumber* The GPIO number

---

```
9.1.3.8 void GPIO_setLow (  
    GPIO_Handle gpioHandle,  
    const GPIO_Number_e gpioNumber )
```

Sets the specified general purpose I/O (GPIO) signal low.

[1]Parameters *in gpioHandle* The general purpose I/O (GPIO) object handle

---

*in gpioNumber* The GPIO number

---

9.1.3.9 void GPIO\_setMode (  
    **GPIO\_Handle** gpioHandle,  
    const **GPIO\_Number\_e** gpioNumber,  
    const **GPIO\_Mode\_e** mode )

Sets the mode for the specified general purpose I/O (GPIO) signal.

[1]Parameters *in gpioHandle* The general purpose I/O (GPIO) object handle

---

*in gpioNumber* The GPIO number

---

*in mode* The mode

---

9.1.3.10 void GPIO\_setPortData (  
    **GPIO\_Handle** gpioHandle,  
    const **GPIO\_Port\_e** gpioPort,  
    const uint32\_t data )

Sets data output on a given GPIO port.

[1]Parameters *in gpioHandle* The general purpose I/O (GPIO) object handle

---

*in gpioPort* The GPIO number

---

*in data* The data to write to the port

---

9.1.3.11 void GPIO\_setPullUp (  
    **GPIO\_Handle** gpioHandle,  
    const **GPIO\_Number\_e** gpioNumber,  
    const **GPIO\_PullUp\_e** pullUp )

Sets the general purpose I/O (GPIO) signal pullups.

[1]Parameters *in gpioHandle* The general purpose I/O (GPIO) object handle

---

*in gpioNumber* The GPIO number

---

*in pullUp* The pull up enable or disable

---

#### 9.1.3.12 void GPIO\_setQualification (

**GPIO\_Handle** gpioHandle,  
const **GPIO\_Number\_e** gpioNumber,  
const **GPIO\_Qual\_e** qualification )

Sets the qualification for the specified general purpose I/O (GPIO)

[1]Parameters *in gpioHandle* The general purpose I/O (GPIO) object handle

---

*in gpioNumber* The GPIO number

---

*in qualification* The desired input qualification

---

#### 9.1.3.13 void GPIO\_setQualificationPeriod (

**GPIO\_Handle** gpioHandle,  
const **GPIO\_Number\_e** gpioNumber,  
const uint8\_t period )

Sets the qualification period for the specified general purpose I/O block (8 I/O's per block)

[1]Parameters *in gpioHandle* The general purpose I/O (GPIO) object handle

---

*in gpioNumber* The GPIO number

---

*in period* The desired input qualification period

---

#### 9.1.3.14 void GPIO\_toggle (

**GPIO\_Handle** gpioHandle,  
const **GPIO\_Number\_e** gpioNumber )

Toggles the specified general purpose I/O (GPIO) signal.

[1]Parameters *in gpioHandle* The general purpose I/O (GPIO) object handle

---

*in gpioNumber* The GPIO number

---





## 10 Oscillator (OSC)

[Introduction](#) ..... ??

[OSC API Drivers](#) ..... 81

The oscillator (OSC) API provides functions for configuring an external or internal oscillator as well as compensating the internal oscillator for temperature drift.

This driver is contained in `f2802x0/common/source/osc.c`, with `f2802x0/common/include/osc.h` containing the API definitions for use by applications.

### 10.1 OSC

#### Data Structures

`struct _OSC_Obj`

#### Macros

`#define OSC_BASE_ADDR`

`#define OSC_INTOSCnTRIM_COARSE_BITS`

`#define OSC_INTOSCnTRIM_FINE_BITS`

`#define OSC_OTP_COURSE_TRIM1`

`#define OSC_OTP_COURSE_TRIM2`

`#define OSC_OTP_FINE_TRIM_OFFSET1`

`#define OSC_OTP_FINE_TRIM_OFFSET2`

`#define OSC_OTP_FINE_TRIM_SLOPE1`

`#define OSC_OTP_FINE_TRIM_SLOPE2`

`#define OSC_OTP_REF_TEMP_OFFSET`

#### Typedefs

`typedef struct _OSC_Obj * OSC_Handle`

`typedef struct _OSC_Obj OSC_Obj`

## Enumerations

enum [OSC\\_Number\\_e](#) { [OSC\\_Number\\_1](#), [OSC\\_Number\\_2](#) }

enum [OSC\\_Osc2Src\\_e](#) { [OSC\\_Osc2Src\\_Internal](#), [OSC\\_Osc2Src\\_External](#) }

enum [OSC\\_Src\\_e](#) { [OSC\\_Src\\_Internal](#), [OSC\\_Src\\_External](#) }

## Functions

int16\_t [OSC\\_getCourseTrim1](#) ([OSC\\_Handle](#) oscHandle)

int16\_t [OSC\\_getCourseTrim2](#) ([OSC\\_Handle](#) oscHandle)

int16\_t [OSC\\_getFineTrimOffset1](#) ([OSC\\_Handle](#) oscHandle)

int16\_t [OSC\\_getFineTrimOffset2](#) ([OSC\\_Handle](#) oscHandle)

int16\_t [OSC\\_getFineTrimSlope1](#) ([OSC\\_Handle](#) oscHandle)

int16\_t [OSC\\_getFineTrimSlope2](#) ([OSC\\_Handle](#) oscHandle)

int16\_t [OSC\\_getRefTempOffset](#) ([OSC\\_Handle](#) oscHandle)

[OSC\\_Handle](#) [OSC\\_init](#) (void \*pMemory, const size\_t numBytes)

void [OSC\\_setCoarseTrim](#) ([OSC\\_Handle](#) oscHandle, const [OSC\\_Number\\_e](#) oscNumber, const uint8\_t trimValue)

void [OSC\\_setFineTrim](#) ([OSC\\_Handle](#) oscHandle, const [OSC\\_Number\\_e](#) oscNumber, const uint8\_t trimValue)

### 10.1.1 Detailed Description

### 10.1.2 Enumeration Type Documentation

#### 10.1.2.1 enum [OSC\\_Number\\_e](#)

Enumeration to define the oscillator (OSC) number.

##### Enumerator

**[OSC\\_Number\\_1](#)** Denotes oscillator number 1.

**[OSC\\_Number\\_2](#)** Denotes oscillator number 2.

#### 10.1.2.2 enum [OSC\\_Osc2Src\\_e](#)

Enumeration to define the oscillator (OSC) 2 source.

**Enumerator**

***OSC\_Osc2Src\_Internal*** Denotes an internal oscillator source for oscillator 2.

***OSC\_Osc2Src\_External*** Denotes an external oscillator source for oscillator 2.

10.1.2.3 enum **OSC\_Src\_e**

Enumeration to define the oscillator (OSC) source.

**Enumerator**

***OSC\_Src\_Internal*** Denotes an internal oscillator.

***OSC\_Src\_External*** Denotes an external oscillator.

## 10.1.3 Function Documentation

## 10.1.3.1 int16\_t OSC\_getCourseTrim1 (

**OSC\_Handle** oscHandle ) [inline]

Gets the fine trim offset for oscillator 1.

[1]Parameters in *oscHandle* The oscillator (OSC) object handle

---

Returns The fine trim offset for oscillator 1

References OSC\_OTP\_COURSE\_TRIM1.

## 10.1.3.2 int16\_t OSC\_getCourseTrim2 (

**OSC\_Handle** oscHandle ) [inline]

Gets the fine trim offset for oscillator 2.

[1]Parameters in *oscHandle* The oscillator (OSC) object handle

---

Returns The fine trim offset for oscillator 2

References OSC\_OTP\_COURSE\_TRIM2.

## 10.1.3.3 int16\_t OSC\_getFineTrimOffset1 (

**OSC\_Handle** oscHandle ) [inline]

Gets the fine trim offset for oscillator 1.

[1]Parameters in *oscHandle* The oscillator (OSC) object handle

---

Returns The fine trim offset for oscillator 1

References OSC\_OTP\_FINE\_TRIM\_OFFSET1.

#### 10.1.3.4 int16\_t OSC\_getFineTrimOffset2 (

**OSC\_Handle** oscHandle ) [inline]

Gets the fine trim offset for oscillator 2.

[1]Parameters in *oscHandle* The oscillator (OSC) object handle

---

Returns The fine trim offset for oscillator 2

References OSC\_OTP\_FINE\_TRIM\_OFFSET2.

#### 10.1.3.5 int16\_t OSC\_getFineTrimSlope1 (

**OSC\_Handle** oscHandle ) [inline]

Gets the fine trim offset for oscillator 1.

[1]Parameters in *oscHandle* The oscillator (OSC) object handle

---

Returns The fine trim offset for oscillator 1

References OSC\_OTP\_FINE\_TRIM\_SLOPE1.

#### 10.1.3.6 int16\_t OSC\_getFineTrimSlope2 (

**OSC\_Handle** oscHandle ) [inline]

Gets the fine trim offset for oscillator 2.

[1]Parameters in *oscHandle* The oscillator (OSC) object handle

---

Returns The fine trim offset for oscillator 2

References OSC\_OTP\_FINE\_TRIM\_SLOPE2.

#### 10.1.3.7 int16\_t OSC\_getRefTempOffset (

**OSC\_Handle** oscHandle ) [inline]

Gets the reference temperature offset.

[1]Parameters in *oscHandle* The oscillator (OSC) object handle

---

Returns The reference temperature offset

References OSC\_OTP\_REF\_TEMP\_OFFSET.

#### 10.1.3.8 **OSC\_Handle** OSC\_init (

void \* pMemory,

---

```
const size_t numBytes )
```

Initializes the oscillator (OSC) handle.

[1]Parameters *in pMemory* A pointer to the base address of the FLASH registers

---

*in numBytes* The number of bytes allocated for the FLASH object, bytes

---

Returns The flash (FLASH) object handle

---

10.1.3.9 void OSC\_setCoarseTrim (

**OSC\_Handle** oscHandle,

const **OSC\_Number\_e** oscNumber,

const uint8\_t trimValue )

Sets the coarse trim value for a specified oscillator.

[1]Parameters *in oscHandle* The oscillator (OSC) object handle

---

*in oscNumber* The oscillator number

---

*in trimValue* The coarse trim value

---

10.1.3.10 void OSC\_setFineTrim (

**OSC\_Handle** oscHandle,

const **OSC\_Number\_e** oscNumber,

const uint8\_t trimValue )

Sets the fine trim value for a specified oscillator.

[1]Parameters *in oscHandle* The oscillator (OSC) object handle

---

*in oscNumber* The oscillator number

---

*in trimValue* The fine trim value

---



# 11 Peripheral Interrupt Expansion Module (PIE)

[Introduction](#) .....??

[API Functions](#) .....87 The Peripheral Interrupt Expansion controller (PIE) API provides functions for configuring and using the PIE on Piccolo devices. This API provides functions for enabling and disabling individual interrupt sources as well as acknowledging interrupts that have occurred.

This driver is contained in `f2802x0/common/source/pie.c`, with `f2802x0/common/include/pie.h` containing the API definitions for use by applications.

## 11.1 PIE

### Data Structures

`struct _PIE_IERIFR_t`

`struct _PIE_Obj_`

### Macros

`#define PIE_BASE_ADDR`

`#define PIE_DBGIER_DLOGINT_BITS`

`#define PIE_DBGIER_INT10_BITS`

`#define PIE_DBGIER_INT11_BITS`

`#define PIE_DBGIER_INT12_BITS`

`#define PIE_DBGIER_INT13_BITS`

`#define PIE_DBGIER_INT14_BITS`

`#define PIE_DBGIER_INT1_BITS`

`#define PIE_DBGIER_INT2_BITS`

`#define PIE_DBGIER_INT3_BITS`

`#define PIE_DBGIER_INT4_BITS`

`#define PIE_DBGIER_INT5_BITS`

`#define PIE_DBGIER_INT6_BITS`

`#define PIE_DBGIER_INT7_BITS`

`#define PIE_DBGIER_INT8_BITS`

```
#define PIE_DBGIER_INT9_BITS
#define PIE_DBGIER_RTOSINT_BITS
#define PIE_IER_DLOGINT_BITS
#define PIE_IER_INT10_BITS
#define PIE_IER_INT11_BITS
#define PIE_IER_INT12_BITS
#define PIE_IER_INT13_BITS
#define PIE_IER_INT14_BITS
#define PIE_IER_INT1_BITS
#define PIE_IER_INT2_BITS
#define PIE_IER_INT3_BITS
#define PIE_IER_INT4_BITS
#define PIE_IER_INT5_BITS
#define PIE_IER_INT6_BITS
#define PIE_IER_INT7_BITS
#define PIE_IER_INT8_BITS
#define PIE_IER_INT9_BITS
#define PIE_IER_RTOSINT_BITS
#define PIE_IERx_INTx1_BITS
#define PIE_IERx_INTx2_BITS
#define PIE_IERx_INTx3_BITS
#define PIE_IERx_INTx4_BITS
#define PIE_IERx_INTx5_BITS
#define PIE_IERx_INTx6_BITS
#define PIE_IERx_INTx7_BITS
#define PIE_IERx_INTx8_BITS
#define PIE_IFR_DLOGINT_BITS
#define PIE_IFR_INT10_BITS
#define PIE_IFR_INT11_BITS
```



```
#define PIE_IFR_INT12_BITS
#define PIE_IFR_INT13_BITS
#define PIE_IFR_INT14_BITS
#define PIE_IFR_INT1_BITS
#define PIE_IFR_INT2_BITS
#define PIE_IFR_INT3_BITS
#define PIE_IFR_INT4_BITS
#define PIE_IFR_INT5_BITS
#define PIE_IFR_INT6_BITS
#define PIE_IFR_INT7_BITS
#define PIE_IFR_INT8_BITS
#define PIE_IFR_INT9_BITS
#define PIE_IFR_RTOSINT_BITS
#define PIE_IFRx_INTx1_BITS
#define PIE_IFRx_INTx2_BITS
#define PIE_IFRx_INTx3_BITS
#define PIE_IFRx_INTx4_BITS
#define PIE_IFRx_INTx5_BITS
#define PIE_IFRx_INTx6_BITS
#define PIE_IFRx_INTx7_BITS
#define PIE_IFRx_INTx8_BITS
#define PIE_PIEACK_GROUP10_BITS
#define PIE_PIEACK_GROUP11_BITS
#define PIE_PIEACK_GROUP12_BITS
#define PIE_PIEACK_GROUP1_BITS
#define PIE_PIEACK_GROUP2_BITS
#define PIE_PIEACK_GROUP3_BITS
#define PIE_PIEACK_GROUP4_BITS
#define PIE_PIEACK_GROUP5_BITS
```

```
#define PIE_PIEACK_GROUP6_BITS
#define PIE_PIEACK_GROUP7_BITS
#define PIE_PIEACK_GROUP8_BITS
#define PIE_PIEACK_GROUP9_BITS
#define PIE_PIECTRL_ENPIE_BITS
#define PIE_PIECTRL_PIEVECT_BITS
```

## Typedefs

```
typedef interrupt void(* intVec_t)(void)
typedef struct _PIE_Obj * PIE_Handle
typedef struct _PIE_IERIFR_t PIE_IERIFR_t
typedef struct _PIE_Obj PIE_Obj
```

## Enumerations

```
enum PIE_ExtIntPolarity_e { PIE_ExtIntPolarity_FallingEdge, PIE_ExtIntPolarity_RisingEdge,
PIE_ExtIntPolarity_RisingAndFallingEdge }

enum PIE_GroupNumber_e {
PIE_GroupNumber_1, PIE_GroupNumber_2, PIE_GroupNumber_3, PIE_GroupNumber_4,
PIE_GroupNumber_5, PIE_GroupNumber_6, PIE_GroupNumber_7, PIE_GroupNumber_8,
PIE_GroupNumber_9, PIE_GroupNumber_10, PIE_GroupNumber_11, PIE_GroupNumber_12 }

enum PIE_InterruptSource_e {
PIE_InterruptSource_ADCINT_1_1, PIE_InterruptSource_ADCINT_1_2,
PIE_InterruptSource_XINT_1, PIE_InterruptSource_XINT_2,
PIE_InterruptSource_ADCINT_9, PIE_InterruptSource_TIMER_0, PIE_InterruptSource_WAKE,
PIE_InterruptSource_TZ1,
PIE_InterruptSource_TZ2, PIE_InterruptSource_TZ3, PIE_InterruptSource_EPWM1,
PIE_InterruptSource_EPWM2,
PIE_InterruptSource_EPWM3, PIE_InterruptSource_ECAP1, PIE_InterruptSource_SPIARX,
PIE_InterruptSource_SPIATX,
PIE_InterruptSource_I2CA1, PIE_InterruptSource_I2CA2, PIE_InterruptSource_SCIARX,
PIE_InterruptSource_SCIATX,
PIE_InterruptSource_ADCINT_10_1, PIE_InterruptSource_ADCINT_10_2,
PIE_InterruptSource_ADCINT_3, PIE_InterruptSource_ADCINT_4,
PIE_InterruptSource_ADCINT_5, PIE_InterruptSource_ADCINT_6,
PIE_InterruptSource_ADCINT_7, PIE_InterruptSource_ADCINT_8,
PIE_InterruptSource_XINT_3 }

enum PIE_SubGroupNumber_e {
PIE_SubGroupNumber_1, PIE_SubGroupNumber_2, PIE_SubGroupNumber_3,
```

```
PIE_SubGroupNumber_4,  
PIE_SubGroupNumber_5,    PIE_SubGroupNumber_6,    PIE_SubGroupNumber_7,  
PIE_SubGroupNumber_8 }  
  
enum PIE_SystemInterrupts_e {  
    PIE_SystemInterrupts_Reset,    PIE_SystemInterrupts_INT1,    PIE_SystemInterrupts_INT2,  
    PIE_SystemInterrupts_INT3,    PIE_SystemInterrupts_INT5,    PIE_SystemInterrupts_INT6,  
    PIE_SystemInterrupts_INT4,    PIE_SystemInterrupts_INT7,    PIE_SystemInterrupts_INT8,  
    PIE_SystemInterrupts_INT9,    PIE_SystemInterrupts_INT10,  
    PIE_SystemInterrupts_INT11,    PIE_SystemInterrupts_TINT1,    PIE_SystemInterrupts_TINT2,  
    PIE_SystemInterrupts_DATALOG,  
    PIE_SystemInterrupts_RTOSINT,    PIE_SystemInterrupts_EMUINT,    PIE_SystemInterrupts_NMI,  
    PIE_SystemInterrupts_ILLEGAL,  
    PIE_SystemInterrupts_USER1,    PIE_SystemInterrupts_USER2,    PIE_SystemInterrupts_USER3,  
    PIE_SystemInterrupts_USER4,    PIE_SystemInterrupts_USER5,    PIE_SystemInterrupts_USER6,  
    PIE_SystemInterrupts_USER7,    PIE_SystemInterrupts_USER8,  
    PIE_SystemInterrupts_USER9,    PIE_SystemInterrupts_USER10,    PIE_SystemInterrupts_USER11,  
    PIE_SystemInterrupts_USER12 }  
}
```

## Functions

```
void PIE_clearAllFlags (PIE_Handle pieHandle)  
  
void PIE_clearAllInts (PIE_Handle pieHandle)  
  
void PIE_clearInt (PIE_Handle pieHandle, const PIE_GroupNumber_e groupNumber)  
  
void PIE_disable (PIE_Handle pieHandle)  
  
void PIE_disableAllInts (PIE_Handle pieHandle)  
  
void PIE_disableCaptureInt (PIE_Handle pieHandle)  
  
void PIE_disableInt (PIE_Handle pieHandle, const PIE_GroupNumber_e group, const  
PIE_InterruptSource_e intSource)  
  
void PIE_enable (PIE_Handle pieHandle)  
  
void PIE_enableAdcInt (PIE_Handle pieHandle, const ADC_IntNumber_e intNumber)  
  
void PIE_enableCaptureInt (PIE_Handle pieHandle)  
  
void PIE_enableExtInt (PIE_Handle pieHandle, const CPU_ExtIntNumber_e intNumber)  
  
void PIE_enableInt (PIE_Handle pieHandle, const PIE_GroupNumber_e group, const  
PIE_InterruptSource_e intSource)  
  
void PIE_enablePwmInt (PIE_Handle pieHandle, const PWM_Number_e pwmNumber)  
  
void PIE_enablePwmTzInt (PIE_Handle pieHandle, const PWM_Number_e pwmNumber)
```

```
void PIE_enableTimer0Int (PIE_Handle pieHandle)

void PIE_forceInt (PIE_Handle pieHandle, const PIE_GroupNumber_e group, const
PIE_InterruptSource_e intSource)

uint16_t PIE_getExtIntCount (PIE_Handle pieHandle, const CPU_ExtIntNumber_e intNumber)

uint16_t PIE_getIntEnables (PIE_Handle pieHandle, const PIE_GroupNumber_e group)

uint16_t PIE_getIntFlags (PIE_Handle pieHandle, const PIE_GroupNumber_e group)

interrupt void PIE_illegalSr (void)

PIE_Handle PIE_init (void *pMemory, const size_t numBytes)

void PIE_registerPieIntHandler (PIE_Handle pieHandle, const PIE_GroupNumber_e groupNum-
ber, const PIE_SubGroupNumber_e subGroupNumber, const intVec_t vector)

void PIE_registerSystemIntHandler (PIE_Handle pieHandle, const PIE_SystemInterrupts_e sys-
temInt, const intVec_t vector)

void PIE_setDefaultIntVectorTable (PIE_Handle pieHandle)

void PIE_setExtIntPolarity (PIE_Handle pieHandle, const CPU_ExtIntNumber_e intNumber, const
PIE_ExtIntPolarity_e polarity)

void PIE_unregisterPieIntHandler (PIE_Handle pieHandle, const PIE_GroupNumber_e groupNum-
ber, const PIE_SubGroupNumber_e subGroupNumber)

void PIE_unregisterSystemIntHandler (PIE_Handle pieHandle, const PIE_SystemInterrupts_e sys-
temInt)
```

### 11.1.1 Detailed Description

### 11.1.2 Enumeration Type Documentation

#### 11.1.2.1 enum **PIE\_ExtIntPolarity\_e**

Enumeration to define the external interrupt polarity.

##### Enumerator

**PIE\_ExtIntPolarity\_FallingEdge** Denotes an interrupt is generated on the falling edge.

**PIE\_ExtIntPolarity\_RisingEdge** Denotes an interrupt is generated on the rising edge.

**PIE\_ExtIntPolarity\_RisingAndFallingEdge** Denotes an interrupt is generated on the falling and rising edges.

#### 11.1.2.2 enum **PIE\_GroupNumber\_e**

Enumeration to define the peripheral interrupt expansion (PIE) group numbers.

**Enumerator**

**PIE\_GroupNumber\_1** Denotes PIE group number 1.  
**PIE\_GroupNumber\_2** Denotes PIE group number 2.  
**PIE\_GroupNumber\_3** Denotes PIE group number 3.  
**PIE\_GroupNumber\_4** Denotes PIE group number 4.  
**PIE\_GroupNumber\_5** Denotes PIE group number 5.  
**PIE\_GroupNumber\_6** Denotes PIE group number 6.  
**PIE\_GroupNumber\_7** Denotes PIE group number 7.  
**PIE\_GroupNumber\_8** Denotes PIE group number 8.  
**PIE\_GroupNumber\_9** Denotes PIE group number 9.  
**PIE\_GroupNumber\_10** Denotes PIE group number 10.  
**PIE\_GroupNumber\_11** Denotes PIE group number 11.  
**PIE\_GroupNumber\_12** Denotes PIE group number 12.

**11.1.2.3 enum PIE\_InterruptSource\_e**

Enumeration to define the peripheral interrupt expansion (PIE) individual interrupt sources.

**Enumerator**

**PIE\_InterruptSource\_ADCINT\_1\_1** Group 1 ADC Interrupt 1.  
**PIE\_InterruptSource\_ADCINT\_1\_2** Group 1 ADC Interrupt 2.  
**PIE\_InterruptSource\_XINT\_1** External Interrupt 1.  
**PIE\_InterruptSource\_XINT\_2** External Interrupt 2.  
**PIE\_InterruptSource\_ADCINT\_9** ADC Interrupt 9.  
**PIE\_InterruptSource\_TIMER\_0** Timer Interrupt 0.  
**PIE\_InterruptSource\_WAKE** Wake Up Interrupt.  
**PIE\_InterruptSource\_TZ1** EPWM TZ1 Interrupt.  
**PIE\_InterruptSource\_TZ2** EPWM TZ2 Interrupt.  
**PIE\_InterruptSource\_TZ3** EPWM TZ3 Interrupt.  
**PIE\_InterruptSource\_EPWM1** EPWM 1 Interrupt.  
**PIE\_InterruptSource\_EPWM2** EPWM 2 Interrupt.  
**PIE\_InterruptSource\_EPWM3** EPWM 3 Interrupt.  
**PIE\_InterruptSource\_ECAP1** ECAP 1 Interrupt.  
**PIE\_InterruptSource\_SPIARX** SPI A RX Interrupt.  
**PIE\_InterruptSource\_SPIATX** SPI A TX Interrupt.  
**PIE\_InterruptSource\_I2CA1** I2C A Interrupt 1.  
**PIE\_InterruptSource\_I2CA2** I2C A Interrupt 2.  
**PIE\_InterruptSource\_SCIARX** SCI A RX Interrupt.  
**PIE\_InterruptSource\_SCIATX** SCI A TX Interrupt.  
**PIE\_InterruptSource\_ADCINT\_10\_1** Group 10 ADC Interrupt 1.  
**PIE\_InterruptSource\_ADCINT\_10\_2** Group 10 ADC Interrupt 2.  
**PIE\_InterruptSource\_ADCINT\_3** ADC Interrupt 3.  
**PIE\_InterruptSource\_ADCINT\_4** ADC Interrupt 4.  
**PIE\_InterruptSource\_ADCINT\_5** ADC Interrupt 5.  
**PIE\_InterruptSource\_ADCINT\_6** ADC Interrupt 6.  
**PIE\_InterruptSource\_ADCINT\_7** ADC Interrupt 7.  
**PIE\_InterruptSource\_ADCINT\_8** ADC Interrupt 8.  
**PIE\_InterruptSource\_XINT\_3** External Interrupt 3.

#### 11.1.2.4 enum **PIE\_SubGroupNumber\_e**

Enumeration to define the peripheral interrupt expansion (PIE) sub-group numbers.

##### Enumerator

- PIE\_SubGroupNumber\_1** Denotes PIE group number 1.
- PIE\_SubGroupNumber\_2** Denotes PIE group number 2.
- PIE\_SubGroupNumber\_3** Denotes PIE group number 3.
- PIE\_SubGroupNumber\_4** Denotes PIE group number 4.
- PIE\_SubGroupNumber\_5** Denotes PIE group number 5.
- PIE\_SubGroupNumber\_6** Denotes PIE group number 6.
- PIE\_SubGroupNumber\_7** Denotes PIE group number 7.
- PIE\_SubGroupNumber\_8** Denotes PIE group number 8.

#### 11.1.2.5 enum **PIE\_SystemInterrupts\_e**

Enumeration to define the system interrupts.

##### Enumerator

- PIE\_SystemInterrupts\_Reset** Reset interrupt vector.
- PIE\_SystemInterrupts\_INT1** INT1 interrupt vector.
- PIE\_SystemInterrupts\_INT2** INT2 interrupt vector.
- PIE\_SystemInterrupts\_INT3** INT3 interrupt vector.
- PIE\_SystemInterrupts\_INT4** INT4 interrupt vector.
- PIE\_SystemInterrupts\_INT5** INT5 interrupt vector.
- PIE\_SystemInterrupts\_INT6** INT6 interrupt vector.
- PIE\_SystemInterrupts\_INT7** INT7 interrupt vector.
- PIE\_SystemInterrupts\_INT8** INT8 interrupt vector.
- PIE\_SystemInterrupts\_INT9** INT9 interrupt vector.
- PIE\_SystemInterrupts\_INT10** INT10 interrupt vector.
- PIE\_SystemInterrupts\_INT11** INT11 interrupt vector.
- PIE\_SystemInterrupts\_INT12** INT12 interrupt vector.
- PIE\_SystemInterrupts\_TINT1** INT13 interrupt vector.
- PIE\_SystemInterrupts\_TINT2** INT14 interrupt vector.
- PIE\_SystemInterrupts\_DATALOG** DATALOG interrupt vector.
- PIE\_SystemInterrupts\_RTOSINT** RTOSINT interrupt vector.
- PIE\_SystemInterrupts\_EMUINT** EMUINT interrupt vector.
- PIE\_SystemInterrupts\_NMI** NMI interrupt vector.
- PIE\_SystemInterrupts\_ILLEGAL** ILLEGAL interrupt vector.
- PIE\_SystemInterrupts\_USER1** USER1 interrupt vector.
- PIE\_SystemInterrupts\_USER2** USER2 interrupt vector.
- PIE\_SystemInterrupts\_USER3** USER3 interrupt vector.
- PIE\_SystemInterrupts\_USER4** USER4 interrupt vector.
- PIE\_SystemInterrupts\_USER5** USER5 interrupt vector.
- PIE\_SystemInterrupts\_USER6** USER6 interrupt vector.
- PIE\_SystemInterrupts\_USER7** USER7 interrupt vector.

**PIE\_SystemInterrupts\_USER8** USER8 interrupt vector.  
**PIE\_SystemInterrupts\_USER9** USER9 interrupt vector.  
**PIE\_SystemInterrupts\_USER10** USER10 interrupt vector.  
**PIE\_SystemInterrupts\_USER11** USER11 interrupt vector.  
**PIE\_SystemInterrupts\_USER12** USER12 interrupt vector.

### 11.1.3 Function Documentation

#### 11.1.3.1 void PIE\_clearAllFlags (

**PIE\_Handle** pieHandle )

Clears all the interrupt flags.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) object handle

---

#### 11.1.3.2 void PIE\_clearAllInts (

**PIE\_Handle** pieHandle )

Clears all the interrupts.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) object handle

---

#### 11.1.3.3 void PIE\_clearInt (

**PIE\_Handle** pieHandle,

const **PIE\_GroupNumber\_e** groupNumber ) [inline]

Clears an interrupt defined by group number.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) object handle

---

in *groupNumber* The group number

---

References `_PIE_Obj_::PIEACK`.

---

#### 11.1.3.4 void PIE\_disable (

**PIE\_Handle** pieHandle )

Disables the peripheral interrupt expansion (PIE)

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) object handle

---





722.7

---

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) object handle

---

in *intNumber* The interrupt number

---

11.1.3.10 void **PIE\_enableCaptureInt** (  
**PIE\_Handle** pieHandle )

Enables the capture interrupt.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) object handle

---

11.1.3.11 void **PIE\_enableExtInt** (  
**PIE\_Handle** pieHandle,  
const **CPU\_ExtIntNumber\_e** intNumber )

Enables the prescribed external interrupt.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) handle

---

in *intNumber* The interrupt number

---

11.1.3.12 void **PIE\_enableInt** (  
**PIE\_Handle** pieHandle,  
const **PIE\_GroupNumber\_e** group,  
const **PIE\_InterruptSource\_e** intSource )

Enable a specific PIE interrupt.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) object handle

---

in *group* The PIE group an interrupt belongs to

---

in *intSource* The specific interrupt source to enable

---

11.1.3.13 void **PIE\_enablePwmInt** (  
**PIE\_Handle** pieHandle,

const **PWM\_Number\_e** pwmNumber )

Enables the PWM interrupt.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) handle

---

in *pwmNumber* The PWM number

---

11.1.3.14 void PIE\_enablePwmTzInt (

**PIE\_Handle** pieHandle,

const **PWM\_Number\_e** pwmNumber )

Enables the PWM Trip Zone interrupt.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) handle

---

in *pwmNumber* The PWM number

---

11.1.3.15 void PIE\_enableTimer0Int (

**PIE\_Handle** pieHandle )

Enables the Cpu Timer 0 interrupt.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) handle

---

11.1.3.16 void PIE\_forceInt (

**PIE\_Handle** pieHandle,

const **PIE\_GroupNumber\_e** group,

const **PIE\_InterruptSource\_e** intSource )

Force a specific PIE interrupt.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) object handle

---

in *group* The PIE group an interrupt belongs to

---

in *intSource* The specific interrupt source to force

---

11.1.3.17 uint16\_t PIE\_getExtIntCount (

**PIE\_Handle** pieHandle,

const **CPU\_ExtIntNumber\_e** intNumber )

Gets the external interrupt count value.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) handle

---

in *intNumber* The external interrupt number

---

Returns The count value

#### 11.1.3.18 uint16\_t PIE\_getIntEnables (

**PIE\_Handle** pieHandle,

const **PIE\_GroupNumber\_e** group )

Gets PIE interrupt enable values.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) object handle

---

in *group* The PIE group the flags belong to

---

#### 11.1.3.19 uint16\_t PIE\_getIntFlags (

**PIE\_Handle** pieHandle,

const **PIE\_GroupNumber\_e** group )

Gets PIE interrupt flag values.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) object handle

---

in *group* The PIE group the flags belong to

---

#### 11.1.3.20 **PIE\_Handle** PIE\_init (

void \* pMemory,

const size\_t numBytes )

Initializes the peripheral interrupt expansion (PIE) object handle.

[1]Parameters in *pMemory* A pointer to the memory for the PIE object

---

in *numBytes* The number of bytes allocated for the PIE object, bytes

---

Returns The peripheral interrupt expansion (PIE) object handle

## 11.1.3.21 void PIE\_registerPieIntHandler (

**PIE\_Handle** pieHandle,  
const **PIE\_GroupNumber\_e** groupNumber,  
const **PIE\_SubGroupNumber\_e** subGroupNumber,  
const **intVec\_t** vector )

Registers a handler for a PIE interrupt.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) object handle

---

in *groupNumber* The PIE group an interrupt belongs to

---

in *subGroupNumber* The PIE subgroup an interrupt belongs to

---

in *vector* The specific interrupt handler

---

## 11.1.3.22 void PIE\_registerSystemIntHandler (

**PIE\_Handle** pieHandle,  
const **PIE\_SystemInterrupts\_e** systemInt,  
const **intVec\_t** vector )

Registers a handler for a PIE interrupt.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) object handle

---

in *systemInt* The system interrupt to register this handler to

---

in *vector* The specific interrupt handler

---

## 11.1.3.23 void PIE\_setDefaultIntVectorTable (

**PIE\_Handle** pieHandle )

Initializes the vector table with illegal ISR handlers.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) object handle

---

## 11.1.3.24 void PIE\_setExtIntPolarity (

**PIE\_Handle** pieHandle,

```
const CPU_ExtIntNumber_e intNumber,  
const PIE_ExtIntPolarity_e polarity )
```

Sets the external interrupt polarity.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) handle

---

in *intNumber* The external interrupt number

---

in *polarity* The signal polarity

---

#### 11.1.3.25 void PIE\_unregisterPieIntHandler (

```
PIE_Handle pieHandle,  
const PIE_GroupNumber_e groupNumber,  
const PIE_SubGroupNumber_e subGroupNumber )
```

Unregisters a handler for a PIE interrupt.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) object handle

---

in *groupNumber* The PIE group an interrupt belongs to

---

in *subGroupNumber* The PIE subgroup an interrupt belongs to

---

#### 11.1.3.26 void PIE\_unregisterSystemIntHandler (

```
PIE_Handle pieHandle,  
const PIE_SystemInterrupts_e systemInt )
```

Unregisters a handler for a PIE interrupt.

[1]Parameters in *pieHandle* The peripheral interrupt expansion (PIE) object handle

---

in *systemInt* The system interrupt to unregister

---



## 12 Phase Locked Loop (PLL)

[Introduction](#) .....??

[API Functions](#) 103 The Phase Locked Loop (PLL) API provides functions for configuring the device's PLL as well as other miscellaneous clock functions.

This driver is contained in `f2802x0/common/source/pll.c`, with `f2802x0/common/include/pll.h` containing the API definitions for use by applications.

### 12.1 PLL

#### Data Structures

struct [\\_PLL\\_Obj](#)

#### Macros

#define [PLL\\_BASE\\_ADDR](#)

#define [PLL\\_PLLCR\\_DIV\\_BITS](#)

#define [PLL\\_PLLSTS\\_DIVSEL\\_BITS](#)

#define [PLL\\_PLLSTS\\_MCLKCLR\\_BITS](#)

#define [PLL\\_PLLSTS\\_MCLKOFF\\_BITS](#)

#define [PLL\\_PLLSTS\\_MCLKSTS\\_BITS](#)

#define [PLL\\_PLLSTS\\_NORMRDYE\\_BITS](#)

#define [PLL\\_PLLSTS\\_OSCOFF\\_BITS](#)

#define [PLL\\_PLLSTS\\_PLLOCKES\\_BITS](#)

#define [PLL\\_PLLSTS\\_PLLOFF\\_BITS](#)

#### Typedefs

typedef struct [\\_PLL\\_Obj](#) \* [PLL\\_Handle](#)

typedef struct [\\_PLL\\_Obj](#) [PLL\\_Obj](#)

#### Enumerations

enum [PLL\\_ClkStatus\\_e](#) { [PLL\\_ClkStatus\\_Normal](#), [PLL\\_ClkStatus\\_Missing](#) }

```
enum PLL_DivideSelect_e { PLL_DivideSelect_ClkIn_by_4, PLL_DivideSelect_ClkIn_by_2,  
PLL_DivideSelect_ClkIn_by_1 }  
  
enum PLL_LockStatus_e { PLL_LockStatus_Locking, PLL_LockStatus_Done }  
  
enum PLL_Multiplier_e {  
PLL_Multiplier_1, PLL_Multiplier_2, PLL_Multiplier_3, PLL_Multiplier_4,  
PLL_Multiplier_5, PLL_Multiplier_6, PLL_Multiplier_7, PLL_Multiplier_8,  
PLL_Multiplier_9, PLL_Multiplier_10, PLL_Multiplier_11, PLL_Multiplier_12 }
```

## Functions

```
void PLL_disable (PLL_Handle pllHandle)  
void PLL_disableClkDetect (PLL_Handle pllHandle)  
void PLL_disableNormRdy (PLL_Handle pllHandle)  
void PLL_disableOsc (PLL_Handle pllHandle)  
void PLL_enable (PLL_Handle pllHandle)  
void PLL_enableClkDetect (PLL_Handle pllHandle)  
void PLL_enableNormRdy (PLL_Handle pllHandle)  
void PLL_enableOsc (PLL_Handle pllHandle)  
PLL_ClkStatus_e PLL_getClkStatus (PLL_Handle pllHandle)  
PLL_DivideSelect_e PLL_getDivider (PLL_Handle pllHandle)  
PLL_LockStatus_e PLL_getLockStatus (PLL_Handle pllHandle)  
PLL_Multiplier_e PLL_getMultiplier (PLL_Handle pllHandle)  
PLL_Handle PLL_init (void *pMemory, const size_t numBytes)  
void PLL_resetClkDetect (PLL_Handle pllHandle)  
void PLL_setDivider (PLL_Handle pllHandle, const PLL_DivideSelect_e divSelect)  
void PLL_setLockPeriod (PLL_Handle pllHandle, const uint16_t lockPeriod)  
void PLL_setMultiplier (PLL_Handle pllHandle, const PLL_Multiplier_e freq)  
void PLL_setup (PLL_Handle pllHandle, const PLL_Multiplier_e clkMult, const PLL_DivideSelect_e  
divSelect)
```



## 12.1.1 Detailed Description

## 12.1.2 Enumeration Type Documentation

### 12.1.2.1 enum **PLL\_ClkStatus\_e**

Enumeration to define the phase lock loop (PLL) clock status.

#### Enumerator

**PLL\_ClkStatus\_Normal** Denotes a normal clock.

**PLL\_ClkStatus\_Missing** Denotes a missing clock.

### 12.1.2.2 enum **PLL\_DivideSelect\_e**

Enumeration to define the phase lock loop (PLL) divide select.

#### Enumerator

**PLL\_DivideSelect\_ClkIn\_by\_4** Denotes a divide select of CLKIN/4.

**PLL\_DivideSelect\_ClkIn\_by\_2** Denotes a divide select of CLKIN/2.

**PLL\_DivideSelect\_ClkIn\_by\_1** Denotes a divide select of CLKIN/1.

### 12.1.2.3 enum **PLL\_LockStatus\_e**

Enumeration to define the phase lock loop (PLL) clock lock status.

#### Enumerator

**PLL\_LockStatus\_Locking** Denotes that the system is locking to the clock.

**PLL\_LockStatus\_Done** Denotes that the system is locked to the clock.

### 12.1.2.4 enum **PLL\_Multiplier\_e**

Enumeration to define the phase lock loop (PLL) clock frequency.

#### Enumerator

**PLL\_Multiplier\_1** Denotes a multiplier of 1.

**PLL\_Multiplier\_2** Denotes a multiplier of 2.

**PLL\_Multiplier\_3** Denotes a multiplier of 3.

**PLL\_Multiplier\_4** Denotes a multiplier of 4.

**PLL\_Multiplier\_5** Denotes a multiplier of 5.

**PLL\_Multiplier\_6** Denotes a multiplier of 6.

**PLL\_Multiplier\_7** Denotes a multiplier of 7.

**PLL\_Multiplier\_8** Denotes a multiplier of 8.

**PLL\_Multiplier\_9** Denotes a multiplier of 9.

**PLL\_Multiplier\_10** Denotes a multiplier of 10.

**PLL\_Multiplier\_11** Denotes a multiplier of 11.

**PLL\_Multiplier\_12** Denotes a multiplier of 12.

### 12.1.3 Function Documentation

#### 12.1.3.1 void PLL\_disable (

**PLL\_Handle** pllHandle )

Disables the phase lock loop (PLL)

[1]Parameters in *pllHandle* The phase lock loop (PLL) object handle

---

#### 12.1.3.2 void PLL\_disableClkDetect (

**PLL\_Handle** pllHandle )

Disables the clock detect logic.

[1]Parameters in *pllHandle* The phase lock loop (PLL) object handle

---

#### 12.1.3.3 void PLL\_disableNormRdy (

**PLL\_Handle** pllHandle )

Disables the NORMRDY signal.

[1]Parameters in *pllHandle* The phase lock loop (PLL) object handle

---

#### 12.1.3.4 void PLL\_disableOsc (

**PLL\_Handle** pllHandle )

Disables the oscillator.

[1]Parameters in *pllHandle* The phase lock loop (PLL) object handle

---

#### 12.1.3.5 void PLL\_enable (

**PLL\_Handle** pllHandle )

Enables the phase lock loop (PLL)

[1]Parameters in *pllHandle* The phase lock loop (PLL) object handle

---

#### 12.1.3.6 void PLL\_enableClkDetect (



722.7

[1]Parameters in *pllHandle* The phase lock loop (PLL) object handle

---

Returns The lock status

#### 12.1.3.12 **PLL\_Multiplier\_e** PLL\_getMultiplier ( **PLL\_Handle** pllHandle )

Gets the phase lock loop (PLL) clock frequency.

[1]Parameters in *pllHandle* The phase lock loop (PLL) object handle

---

Returns The clock frequency

#### 12.1.3.13 **PLL\_Handle** PLL\_init ( void \* pMemory, const size\_t numBytes )

Initializes the phase lock loop (PLL) object handle.

[1]Parameters in *pMemory* A pointer to the base address of the PLL registers

---

in *numBytes* The number of bytes allocated for the PLL object, bytes

---

Returns The phase lock loop (PLL) object handle

#### 12.1.3.14 void PLL\_resetClkDetect ( **PLL\_Handle** pllHandle )

Resets the phase lock loop (PLL) clock detect logic.

[1]Parameters in *pllHandle* The phase lock loop (PLL) object handle

---

#### 12.1.3.15 void PLL\_setDivider ( **PLL\_Handle** pllHandle, const **PLL\_DivideSelect\_e** divSelect )

Sets the phase lock loop (PLL) divide select value.

[1]Parameters in *pllHandle* The phase lock loop (PLL) object handle

---

in *divSelect* The divide select value

---

## 12.1.3.16 void PLL\_setLockPeriod (

**PLL\_Handle** pllHandle,

const uint16\_t lockPeriod )

Sets the phase lock loop (PLL) lock time.

[1]Parameters in *pllHandle* The phase lock loop (PLL) object handle

---

*in lockPeriod* The lock period, cycles

---

## 12.1.3.17 void PLL\_setMultiplier (

**PLL\_Handle** pllHandle,const **PLL\_Multiplier\_e** freq )

Sets the phase lock loop (PLL) clock frequency.

[1]Parameters in *pllHandle* The phase lock loop (PLL) object handle

---

*in freq* The clock frequency

---

## 12.1.3.18 void PLL\_setup (

**PLL\_Handle** pllHandle,const **PLL\_Multiplier\_e** clkMult,const **PLL\_DivideSelect\_e** divSelect )

Sets the phase lock loop (PLL) divider and multiplier.

[1]Parameters in *pllHandle* The phase lock loop (PLL) object handle

---

*in clkMult* The clock multiplier value

---

*in divSelect* The divide select value

---



# 13 Pulse Width Modulator (PWM)

[Introduction](#) ..... [??](#)

[API Functions](#) ..... 111 The pulse width modulation peripheral (PWM) APIs provide functions for configuring and updating the PWM peripherals on this device.

This driver is contained in `f2802x0/common/source/pwm.c`, with `f2802x0/common/include/pwm.h` containing the API definitions for use by applications.

## 13.1 PWM

### Data Structures

`struct _PWM_Obj_`

### Macros

```
#define PWM_AQCTL_CAD_BITS
#define PWM_AQCTL_CAU_BITS
#define PWM_AQCTL_CBD_BITS
#define PWM_AQCTL_CBU_BITS
#define PWM_AQCTL_PRD_BITS
#define PWM_AQCTL_ZRO_BITS
#define PWM_CMPCTL_LOADAMODE_BITS
#define PWM_CMPCTL_LOADBMODE_BITS
#define PWM_CMPCTL_SHDWAFULL_BITS
#define PWM_CMPCTL_SHDWAMODE_BITS
#define PWM_CMPCTL_SHDWBFULL_BITS
#define PWM_CMPCTL_SHDWBMODE_BITS
#define PWM_DBCTL_HALFCYCLE_BITS
#define PWM_DBCTL_INMODE_BITS
#define PWM_DBCTL_OUTMODE_BITS
#define PWM_DBCTL_POLSEL_BITS
#define PWM_DCFCTL_BLANKE_BITS
```

```
#define PWM_DCFCTL_BLANKINV_BITS
#define PWM_DCFCTL_PULSESEL_BITS
#define PWM_DCFCTL_SRCSEL_BITS
#define PWM_DCTRISEL_DCAHCOMPSEL_BITS
#define PWM_DCTRISEL_DCALCOMPSEL_BITS
#define PWM_DCTRISEL_DCBHCOMPSEL_BITS
#define PWM_DCTRISEL_DCBLCOMPSEL_BITS
#define PWM_ePWM1_BASE_ADDR
#define PWM_ePWM2_BASE_ADDR
#define PWM_ePWM3_BASE_ADDR
#define PWM_ETCLR_INT_BITS
#define PWM_ETCLR_SOCA_BITS
#define PWM_ETCLR_SOCB_BITS
#define PWM_ETPS_INTCNT_BITS
#define PWM_ETPS_INTPRD_BITS
#define PWM_ETPS_SOCACNT_BITS
#define PWM_ETPS_SOCAPRD_BITS
#define PWM_ETPS_SOCBCNT_BITS
#define PWM_ETPS_SOCBPRD_BITS
#define PWM_ETSEL_INTEN_BITS
#define PWM_ETSEL_INTSEL_BITS
#define PWM_ETSEL_SOCAEN_BITS
#define PWM_ETSEL_SOCASEL_BITS
#define PWM_ETSEL_SOCBEN_BITS
#define PWM_ETSEL_SOCBSEL_BITS
#define PWM_PCCTL_CHPDUTY_BITS
#define PWM_PCCTL_CHPEN_BITS
#define PWM_PCCTL_CHPFREQ_BITS
#define PWM_PCCTL_OSHTWTH_BITS
```



```
#define PWM_TBCTL_CLKDIV_BITS
#define PWM_TBCTL_CTRMODE_BITS
#define PWM_TBCTL_FREESOFT_BITS
#define PWM_TBCTL_HSPCLKDIV_BITS
#define PWM_TBCTL_PHSDIR_BITS
#define PWM_TBCTL_PHSSEN_BITS
#define PWM_TBCTL_PRDLT_BITS
#define PWM_TBCTL_SWFSYNC_BITS
#define PWM_TBCTL_SYNCSEL_BITS
#define PWM_TZCLR_CBC_BITS
#define PWM_TZCLR_DCAEV1_BITS
#define PWM_TZCLR_DCAEV2_BITS
#define PWM_TZCLR_DCBEV1_BITS
#define PWM_TZCLR_DCBEV2_BITS
#define PWM_TZCLR_INT_BITS
#define PWM_TZCLR_OST_BITS
#define PWM_TZCTL_DCAEV1_BITS
#define PWM_TZCTL_DCAEV2_BITS
#define PWM_TZCTL_DCBEV1_BITS
#define PWM_TZCTL_DCBEV2_BITS
#define PWM_TZCTL_TZA_BITS
#define PWM_TZCTL_TZB_BITS
#define PWM_TZDCSEL_DCAEV1_BITS
#define PWM_TZDCSEL_DCAEV2_BITS
#define PWM_TZDCSEL_DCBEV1_BITS
#define PWM_TZDCSEL_DCBEV2_BITS
#define PWM_TZFRC_CBC_BITS
#define PWM_TZFRC_DCAEV1_BITS
#define PWM_TZFRC_DCAEV2_BITS
```

```
#define PWM_TZFRC_DCBEVT1_BITS  
#define PWM_TZFRC_DCBEVT2_BITS  
#define PWM_TZFRC_OST_BITS
```

## Typedefs

```
typedef struct _PWM_Obj_ * PWM_Handle  
typedef struct _PWM_Obj_ PWM_Obj
```

## Enumerations

```
enum PWM_ActionQual_e  
enum PWM_ChoppingClkFreq_e  
enum PWM_ChoppingDutyCycle_e  
enum PWM_ChoppingPulseWidth_e  
enum PWM_ClkDiv_e  
enum PWM_CounterMode_e  
enum PWM_DeadBandInputMode_e  
enum PWM_DeadBandOutputMode_e  
enum PWM_DeadBandPolarity_e  
enum PWM_DigitalCompare_FilterSrc_e  
enum PWM_DigitalCompare_Input_e  
enum PWM_DigitalCompare_InputSel_e  
enum PWM_DigitalCompare_PulseSel_e  
enum PWM_HspClkDiv_e  
enum PWM_IntMode_e  
enum PWM_IntPeriod_e  
enum PWM_LoadMode_e  
enum PWM_Number_e  
enum PWM_PeriodLoad_e  
enum PWM_PhaseDir_e
```

```
enum PWM_RunMode_e

enum PWM_ShadowMode_e

enum PWM_ShadowStatus_e

enum PWM_SocPeriod_e

enum PWM_SocPulseSrc_e

enum PWM_SyncMode_e

enum PWM_TripZoneDCEventSel_e {
    PWM_TripZoneDCEventSel_Disabled,          PWM_TripZoneDCEventSel_DCxHL_DCxLX,
    PWM_TripZoneDCEventSel_DCxHH_DCxLX, PWM_TripZoneDCEventSel_DCxHx_DCxLL,
    PWM_TripZoneDCEventSel_DCxHx_DCxLH, PWM_TripZoneDCEventSel_DCxHL_DCxLH }

enum PWM_TripZoneFlag_e {
    PWM_TripZoneFlag_Global,          PWM_TripZoneFlag_CBC,          PWM_TripZoneFlag_OST,
    PWM_TripZoneFlag_DCAEVT1,
    PWM_TripZoneFlag_DCAEVT2, PWM_TripZoneFlag_DCBEVT1, PWM_TripZoneFlag_DCBEVT2
}

enum PWM_TripZoneSrc_e

enum PWM_TripZoneState_e
```

## Functions

```
void PWM_clearIntFlag (PWM_Handle pwmHandle)

void PWM_clearOneShotTrip (PWM_Handle pwmHandle)

void PWM_clearSocAFlag (PWM_Handle pwmHandle)

void PWM_clearSocBFlag (PWM_Handle pwmHandle)

void PWM_clearTripZone (PWM_Handle pwmHandle, const PWM_TripZoneFlag_e tripZoneFlag)

void PWM_decrementDeadBandFallingEdgeDelay (PWM_Handle pwmHandle)

void PWM_decrementDeadBandRisingEdgeDelay (PWM_Handle pwmHandle)

void PWM_disableChopping (PWM_Handle pwmHandle)

void PWM_disableCounterLoad (PWM_Handle pwmHandle)

void PWM_disableDeadBand (PWM_Handle pwmHandle)

void PWM_disableDeadBandHalfCycle (PWM_Handle pwmHandle)

void PWM_disableDigitalCompareBlankingWindow (PWM_Handle pwmHandle)

void PWM_disableDigitalCompareBlankingWindowInversion (PWM_Handle pwmHandle)
```

```
void PWM_disableInt (PWM_Handle pwmHandle)

void PWM_disableSocAPulse (PWM_Handle pwmHandle)

void PWM_disableSocBPulse (PWM_Handle pwmHandle)

void PWM_disableTripZoneInt (PWM_Handle pwmHandle, const PWM_TripZoneFlag_e interrupt-
Source)

void PWM_disableTripZones (PWM_Handle pwmHandle)

void PWM_disableTripZoneSrc (PWM_Handle pwmHandle, const PWM_TripZoneSrc_e src)

void PWM_enableChopping (PWM_Handle pwmHandle)

void PWM_enableCounterLoad (PWM_Handle pwmHandle)

void PWM_enableDeadBandHalfCycle (PWM_Handle pwmHandle)

void PWM_enableDigitalCompareBlankingWindow (PWM_Handle pwmHandle)

void PWM_enableDigitalCompareBlankingWindowInversion (PWM_Handle pwmHandle)

void PWM_enableInt (PWM_Handle pwmHandle)

void PWM_enableSocAPulse (PWM_Handle pwmHandle)

void PWM_enableSocBPulse (PWM_Handle pwmHandle)

void PWM_enableTripZoneInt (PWM_Handle pwmHandle, const PWM_TripZoneFlag_e interrupt-
Source)

void PWM_enableTripZoneSrc (PWM_Handle pwmHandle, const PWM_TripZoneSrc_e src)

uint16_t PWM_getCmpA (PWM_Handle pwmHandle)

uint16_t PWM_getCmpB (PWM_Handle pwmHandle)

uint16_t PWM_getDeadBandFallingEdgeDelay (PWM_Handle pwmHandle)

uint16_t PWM_getDeadBandRisingEdgeDelay (PWM_Handle pwmHandle)

uint16_t PWM_getIntCount (PWM_Handle pwmHandle)

uint16_t PWM_getPeriod (PWM_Handle pwmHandle)

uint16_t PWM_getSocACount (PWM_Handle pwmHandle)

uint16_t PWM_getSocBCount (PWM_Handle pwmHandle)

void PWM_incrementDeadBandFallingEdgeDelay (PWM_Handle pwmHandle)

void PWM_incrementDeadBandRisingEdgeDelay (PWM_Handle pwmHandle)

PWM_Handle PWM_init (void *pMemory, const size_t numBytes)

void PWM_setActionQual_CntDown_CmpA_PwmA (PWM_Handle pwmHandle, const
PWM_ActionQual_e actionQual)
```

---

```

void PWM_setActionQual_CntDown_CmpA_PwmB (PWM_Handle pwmHandle, const
PWM_ActionQual_e actionQual)

void PWM_setActionQual_CntDown_CmpB_PwmA (PWM_Handle pwmHandle, const
PWM_ActionQual_e actionQual)

void PWM_setActionQual_CntDown_CmpB_PwmB (PWM_Handle pwmHandle, const
PWM_ActionQual_e actionQual)

void PWM_setActionQual_CntUp_CmpA_PwmA (PWM_Handle pwmHandle, const
PWM_ActionQual_e actionQual)

void PWM_setActionQual_CntUp_CmpA_PwmB (PWM_Handle pwmHandle, const
PWM_ActionQual_e actionQual)

void PWM_setActionQual_CntUp_CmpB_PwmA (PWM_Handle pwmHandle, const
PWM_ActionQual_e actionQual)

void PWM_setActionQual_CntUp_CmpB_PwmB (PWM_Handle pwmHandle, const
PWM_ActionQual_e actionQual)

void PWM_setActionQual_Period_PwmA (PWM_Handle pwmHandle, const PWM_ActionQual_e
actionQual)

void PWM_setActionQual_Period_PwmB (PWM_Handle pwmHandle, const PWM_ActionQual_e
actionQual)

void PWM_setActionQual_Zero_PwmA (PWM_Handle pwmHandle, const PWM_ActionQual_e ac-
tionQual)

void PWM_setActionQual_Zero_PwmB (PWM_Handle pwmHandle, const PWM_ActionQual_e ac-
tionQual)

void PWM_setChoppingClkFreq (PWM_Handle pwmHandle, const PWM_ChoppingClkFreq_e clk-
Freq)

void PWM_setChoppingDutyCycle (PWM_Handle pwmHandle, const
PWM_ChoppingDutyCycle_e dutyCycle)

void PWM_setChoppingPulseWidth (PWM_Handle pwmHandle, const
PWM_ChoppingPulseWidth_e pulseWidth)

void PWM_setClkDiv (PWM_Handle pwmHandle, const PWM_ClkDiv_e clkDiv)

void PWM_setCmpA (PWM_Handle pwmHandle, const uint16_t pwmData)

void PWM_setCmpB (PWM_Handle pwmHandle, const uint16_t pwmData)

void PWM_setCount (PWM_Handle pwmHandle, const uint16_t count)

void PWM_setCounterMode (PWM_Handle pwmHandle, const PWM_CounterMode_e counter-
Mode)

void PWM_setDeadBandFallingEdgeDelay (PWM_Handle pwmHandle, const uint16_t delay)

void PWM_setDeadBandInputMode (PWM_Handle pwmHandle, const
PWM_DeadBandInputMode_e inputMode)

```

```
void PWM_setDeadBandOutputMode (PWM_Handle pwmHandle, const
PWM_DeadBandOutputMode_e outputMode)

void PWM_setDeadBandPolarity (PWM_Handle pwmHandle, const PWM_DeadBandPolarity_e
polarity)

void PWM_setDeadBandRisingEdgeDelay (PWM_Handle pwmHandle, const uint16_t delay)

void PWM_setDigitalCompareAEvent1 (PWM_Handle pwmHandle, const bool_t selectFilter, const
bool_t disableSync, const bool_t enableSoc, const bool_t generateSync)

void PWM_setDigitalCompareAEvent2 (PWM_Handle pwmHandle, const bool_t selectFilter, const
bool_t disableSync)

void PWM_setDigitalCompareBEvent1 (PWM_Handle pwmHandle, const bool_t selectFilter, const
bool_t disableSync, const bool_t enableSoc, const bool_t generateSync)

void PWM_setDigitalCompareBEvent2 (PWM_Handle pwmHandle, const bool_t selectFilter, const
bool_t disableSync)

void PWM_setDigitalCompareBlankingPulse (PWM_Handle pwmHandle, const
PWM_DigitalCompare_PulseSel_e pulseSelect)

void PWM_setDigitalCompareFilterOffset (PWM_Handle pwmHandle, const uint16_t offset)

void PWM_setDigitalCompareFilterSource (PWM_Handle pwmHandle, const
PWM_DigitalCompare_FilterSrc_e input)

void PWM_setDigitalCompareFilterWindow (PWM_Handle pwmHandle, const uint16_t window)

void PWM_setDigitalCompareInput (PWM_Handle pwmHandle, const
PWM_DigitalCompare_Input_e input, const PWM_DigitalCompare_InputSel_e inputSel)

void PWM_setHighSpeedClkDiv (PWM_Handle pwmHandle, const PWM_HspClkDiv_e clkDiv)

void PWM_setIntMode (PWM_Handle pwmHandle, const PWM_IntMode_e intMode)

void PWM_setIntPeriod (PWM_Handle pwmHandle, const PWM_IntPeriod_e intPeriod)

void PWM_setLoadMode_CmpA (PWM_Handle pwmHandle, const PWM_LoadMode_e load-
Mode)

void PWM_setLoadMode_CmpB (PWM_Handle pwmHandle, const PWM_LoadMode_e load-
Mode)

void PWM_setOneShotTrip (PWM_Handle pwmHandle)

void PWM_setPeriod (PWM_Handle pwmHandle, const uint16_t period)

void PWM_setPeriodLoad (PWM_Handle pwmHandle, const PWM_PeriodLoad_e periodLoad)

void PWM_setPhase (PWM_Handle pwmHandle, const uint16_t phase)

void PWM_setPhaseDir (PWM_Handle pwmHandle, const PWM_PhaseDir_e phaseDir)

void PWM_setRunMode (PWM_Handle pwmHandle, const PWM_RunMode_e runMode)
```

```
void PWM_setShadowMode_CmpA (PWM_Handle pwmHandle, const PWM_ShadowMode_e
shadowMode)

void PWM_setShadowMode_CmpB (PWM_Handle pwmHandle, const PWM_ShadowMode_e
shadowMode)

void PWM_setSocAPeriod (PWM_Handle pwmHandle, const PWM_SocPeriod_e intPeriod)

void PWM_setSocAPulseSrc (PWM_Handle pwmHandle, const PWM_SocPulseSrc_e pulseSrc)

void PWM_setSocBPeriod (PWM_Handle pwmHandle, const PWM_SocPeriod_e intPeriod)

void PWM_setSocBPulseSrc (PWM_Handle pwmHandle, const PWM_SocPulseSrc_e pulseSrc)

void PWM_setSwSync (PWM_Handle pwmHandle)

void PWM_setSyncMode (PWM_Handle pwmHandle, const PWM_SyncMode_e syncMode)

void PWM_setTripZoneDCEventSelect_DCAEVT1 (PWM_Handle pwmHandle, const
PWM_TripZoneDCEventSel_e tripZoneEvent)

void PWM_setTripZoneDCEventSelect_DCAEVT2 (PWM_Handle pwmHandle, const
PWM_TripZoneDCEventSel_e tripZoneEvent)

void PWM_setTripZoneDCEventSelect_DCBEVT1 (PWM_Handle pwmHandle, const
PWM_TripZoneDCEventSel_e tripZoneEvent)

void PWM_setTripZoneDCEventSelect_DCBEVT2 (PWM_Handle pwmHandle, const
PWM_TripZoneDCEventSel_e tripZoneEvent)

void PWM_setTripZoneState_DCAEVT1 (PWM_Handle pwmHandle, const
PWM_TripZoneState_e tripZoneState)

void PWM_setTripZoneState_DCAEVT2 (PWM_Handle pwmHandle, const
PWM_TripZoneState_e tripZoneState)

void PWM_setTripZoneState_DCBEVT1 (PWM_Handle pwmHandle, const
PWM_TripZoneState_e tripZoneState)

void PWM_setTripZoneState_DCBEVT2 (PWM_Handle pwmHandle, const
PWM_TripZoneState_e tripZoneState)

void PWM_setTripZoneState_TZA (PWM_Handle pwmHandle, const PWM_TripZoneState_e trip-
ZoneState)

void PWM_setTripZoneState_TZB (PWM_Handle pwmHandle, const PWM_TripZoneState_e trip-
ZoneState)

void PWM_write_CmpA (PWM_Handle pwmHandle, const int16_t pwmData)

void PWM_write_CmpB (PWM_Handle pwmHandle, const int16_t pwmData)
```

### 13.1.1 Detailed Description

### 13.1.2 Enumeration Type Documentation

#### 13.1.2.1 enum **PWM\_TripZoneDCEventSel\_e**

Enumeration to define the pulse width modulation (PWM) trip zone event selections.

##### Enumerator

**PWM\_TripZoneDCEventSel\_Disabled** Event Disabled.

**PWM\_TripZoneDCEventSel\_DCxHL\_DCxLX** Compare H = Low, Compare L = Don't Care.

**PWM\_TripZoneDCEventSel\_DCxHH\_DCxLX** Compare H = High, Compare L = Don't Care.

**PWM\_TripZoneDCEventSel\_DCxHx\_DCxLL** Compare H = Don't Care, Compare L = Low.

**PWM\_TripZoneDCEventSel\_DCxHx\_DCxLH** Compare H = Don't Care, Compare L = High.

**PWM\_TripZoneDCEventSel\_DCxHL\_DCxLH** Compare H = Low, Compare L = High.

#### 13.1.2.2 enum **PWM\_TripZoneFlag\_e**

Enumeration to define the pulse width modulation (PWM) trip zone states.

##### Enumerator

**PWM\_TripZoneFlag\_Global** Global Trip Zone flag.

**PWM\_TripZoneFlag\_CBC** Cycle by cycle Trip Zone flag.

**PWM\_TripZoneFlag\_OST** One Shot Trip Zone flag.

**PWM\_TripZoneFlag\_DCAEVT1** Digital Compare A Event 1 Trip Zone flag.

**PWM\_TripZoneFlag\_DCAEVT2** Digital Compare A Event 2 Trip Zone flag.

**PWM\_TripZoneFlag\_DCBEVT1** Digital Compare B Event 1 Trip Zone flag.

**PWM\_TripZoneFlag\_DCBEVT2** Digital Compare B Event 2 Trip Zone flag.

### 13.1.3 Function Documentation

#### 13.1.3.1 void PWM\_clearIntFlag (

**PWM\_Handle** pwmHandle ) [inline]

Clears the pulse width modulation (PWM) interrupt flag.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

References *\_PWM\_Obj\_::ETCLR*, and *PWM\_ETCLR\_INT\_BITS*.

#### 13.1.3.2 void PWM\_clearOneShotTrip (



**PWM\_Handle** pwmHandle ) [inline]

Clears the pulse width modulation (PWM) one shot trip.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

References DISABLE\_PROTECTED\_REGISTER\_WRITE\_MODE, EN-  
 ABLE\_PROTECTED\_REGISTER\_WRITE\_MODE, PWM\_TZCLR\_OST\_BITS, and  
 \_PWM\_Obj::TZCLR.

## 13.1.3.3 void PWM\_clearSocAFlag (

**PWM\_Handle** pwmHandle ) [inline]

Clears the pulse width modulation (PWM) start of conversion (SOC) A flag.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

References \_PWM\_Obj::ETCLR, and PWM\_ETCLR\_SOCA\_BITS.

## 13.1.3.4 void PWM\_clearSocBFlag (

**PWM\_Handle** pwmHandle ) [inline]

Clears the pulse width modulation (PWM) start of conversion (SOC) B flag.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

References \_PWM\_Obj::ETCLR, and PWM\_ETCLR\_SOCB\_BITS.

## 13.1.3.5 void PWM\_clearTripZone (

**PWM\_Handle** pwmHandle,const **PWM\_TripZoneFlag\_e** tripZoneFlag )

Clears the trip zone (TZ) flag specified.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *tripZoneFlag* The trip zone flag to clear

## 13.1.3.6 void PWM\_decrementDeadBandFallingEdgeDelay (

**PWM\_Handle** pwmHandle )

Decrement the dead band falling edge delay.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

#### 13.1.3.7 void PWM\_decrementDeadBandRisingEdgeDelay (

**PWM\_Handle** pwmHandle )

Decrement the dead band rising edge delay.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

#### 13.1.3.8 void PWM\_disableChopping (

**PWM\_Handle** pwmHandle )

Disables the pulse width modulation (PWM) chopping.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

#### 13.1.3.9 void PWM\_disableCounterLoad (

**PWM\_Handle** pwmHandle )

Disables the pulse width modulation (PWM) counter loading from the phase register.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

#### 13.1.3.10 void PWM\_disableDeadBand (

**PWM\_Handle** pwmHandle )

Disables the pulse width modulation (PWM) deadband.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

#### 13.1.3.11 void PWM\_disableDeadBandHalfCycle (

**PWM\_Handle** pwmHandle )

Disables the pulse width modulation (PWM) deadband half cycle clocking.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---



722.7

const **PWM\_TripZoneFlag\_e** interruptSource )

Disables the pulse width modulation (PWM) trip zones interrupts.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *interruptSource* The interrupt source to disable

---

13.1.3.18 void PWM\_disableTripZones (

**PWM\_Handle** pwmHandle )

Disables the pulse width modulation (PWM) trip zones.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

13.1.3.19 void PWM\_disableTripZoneSrc (

**PWM\_Handle** pwmHandle,

const **PWM\_TripZoneSrc\_e** src )

Disable the pulse width modulation (PWM) trip zone source.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *src* The pulse width modulation (PWM) trip zone source

---

13.1.3.20 void PWM\_enableChopping (

**PWM\_Handle** pwmHandle )

Enables the pulse width modulation (PWM) chopping.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

13.1.3.21 void PWM\_enableCounterLoad (

**PWM\_Handle** pwmHandle )

Enables the pulse width modulation (PWM) counter loading from the phase register.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

**13.1.3.22 void PWM\_enableDeadBandHalfCycle (****PWM\_Handle** pwmHandle )

Enables the pulse width modulation (PWM) deadband half cycle clocking.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

**13.1.3.23 void PWM\_enableDigitalCompareBlankingWindow (****PWM\_Handle** pwmHandle )

Enables the pulse width modulation (PWM) digital compare blanking window.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

**13.1.3.24 void PWM\_enableDigitalCompareBlankingWindowInversion (****PWM\_Handle** pwmHandle )

Enables the pulse width modulation (PWM) digital compare blanking window inversion.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

**13.1.3.25 void PWM\_enableInt (****PWM\_Handle** pwmHandle )

Enables the pulse width modulation (PWM) interrupt.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

**13.1.3.26 void PWM\_enableSocAPulse (****PWM\_Handle** pwmHandle )

Enables the pulse width modulation (PWM) start of conversion (SOC) A pulse generation.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

**13.1.3.27 void PWM\_enableSocBPulse (****PWM\_Handle** pwmHandle )

Enables the pulse width modulation (PWM) start of conversion (SOC) B pulse generation.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

#### 13.1.3.28 void PWM\_enableTripZoneInt (

**PWM\_Handle** pwmHandle,  
const **PWM\_TripZoneFlag\_e** interruptSource )

Enables the pulse width modulation (PWM) trip zones interrupts.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *interruptSource* The interrupt source to enable

---

#### 13.1.3.29 void PWM\_enableTripZoneSrc (

**PWM\_Handle** pwmHandle,  
const **PWM\_TripZoneSrc\_e** src )

Enable the pulse width modulation (PWM) trip zone source.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *src* The pulse width modulation (PWM) trip zone source

---

#### 13.1.3.30 uint16\_t PWM\_getCmpA (

**PWM\_Handle** pwmHandle ) [inline]

Gets the pulse width modulation (PWM) data value from the Counter Compare A hardware.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

Returns The PWM compare data value

References *\_PWM\_Obj\_::CMPA*.

---

#### 13.1.3.31 uint16\_t PWM\_getCmpB (

**PWM\_Handle** pwmHandle ) [inline]

Gets the pulse width modulation (PWM) data value from the Counter Compare B hardware.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

Returns The PWM compare data value

References *\_PWM\_Obj\_::CMPB*.

---

## 13.1.3.32 uint16\_t PWM\_getDeadBandFallingEdgeDelay (

**PWM\_Handle** pwmHandle )

Gets the pulse width modulation (PWM) deadband falling edge delay.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

Returns The delay

## 13.1.3.33 uint16\_t PWM\_getDeadBandRisingEdgeDelay (

**PWM\_Handle** pwmHandle )

Gets the pulse width modulation (PWM) deadband rising edge delay.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

Returns The delay

## 13.1.3.34 uint16\_t PWM\_getIntCount (

**PWM\_Handle** pwmHandle )

Gets the pulse width modulation (PWM) interrupt event count.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

Returns The interrupt event count

## 13.1.3.35 uint16\_t PWM\_getPeriod (

**PWM\_Handle** pwmHandle ) [inline]

Gets the pulse width modulation (PWM) period value.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

Returns The pwm period value

References `_PWM_Obj::TBPRD`.

## 13.1.3.36 uint16\_t PWM\_getSocACount (

**PWM\_Handle** pwmHandle )

Gets the pulse width modulation (PWM) start of conversion (SOC) A count.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

Returns The SOC A count

#### 13.1.3.37 uint16\_t PWM\_getSocBCount (

**PWM\_Handle** pwmHandle )

Gets the pulse width modulation (PWM) start of conversion (SOC) B count.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

Returns The SOC B count

#### 13.1.3.38 void PWM\_incrementDeadBandFallingEdgeDelay (

**PWM\_Handle** pwmHandle )

Increment the dead band falling edge delay.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

#### 13.1.3.39 void PWM\_incrementDeadBandRisingEdgeDelay (

**PWM\_Handle** pwmHandle )

Increment the dead band rising edge delay.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

#### 13.1.3.40 **PWM\_Handle** PWM\_init (

void \* pMemory,

const size\_t numBytes )

Initializes the pulse width modulation (PWM) object handle.

[1]Parameters in *pMemory* A pointer to the base address of the PWM registers

---

in *numBytes* The number of bytes allocated for the PWM object, bytes

---

Returns The pulse width modulation (PWM) object handle

#### 13.1.3.41 void PWM\_setActionQual\_CntDown\_CmpA\_PwmA (

**PWM\_Handle** pwmHandle,



---

```
const PWM_ActionQual_e actionQual )
```

Sets the pulse width modulation (PWM) object action for PWM A when the counter equals CMPA and the counter is decrementing.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *actionQual* The action qualifier

---

```
13.1.3.42 void PWM_setActionQual_CntDown_CmpA_PwmB (
```

```
    PWM_Handle pwmHandle,
```

```
    const PWM_ActionQual_e actionQual )
```

Sets the pulse width modulation (PWM) object action for PWM B when the counter equals CMPA and the counter is decrementing.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *actionQual* The action qualifier

---

```
13.1.3.43 void PWM_setActionQual_CntDown_CmpB_PwmA (
```

```
    PWM_Handle pwmHandle,
```

```
    const PWM_ActionQual_e actionQual )
```

Sets the pulse width modulation (PWM) object action for PWM A when the counter equals CMPB and the counter is decrementing.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *actionQual* The action qualifier

---

```
13.1.3.44 void PWM_setActionQual_CntDown_CmpB_PwmB (
```

```
    PWM_Handle pwmHandle,
```

```
    const PWM_ActionQual_e actionQual )
```

Sets the pulse width modulation (PWM) object action for PWM B when the counter equals CMPB and the counter is decrementing.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *actionQual* The action qualifier

---

#### 13.1.3.45 void PWM\_setActionQual\_CntUp\_CmpA\_PwmA (

**PWM\_Handle** pwmHandle,

const **PWM\_ActionQual\_e** actionQual )

Sets the pulse width modulation (PWM) object action for PWM A when the counter equals CMPA and the counter is incrementing.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *actionQual* The action qualifier

---

#### 13.1.3.46 void PWM\_setActionQual\_CntUp\_CmpA\_PwmB (

**PWM\_Handle** pwmHandle,

const **PWM\_ActionQual\_e** actionQual )

Sets the pulse width modulation (PWM) object action for PWM B when the counter equals CMPA and the counter is incrementing.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *actionQual* The action qualifier

---

#### 13.1.3.47 void PWM\_setActionQual\_CntUp\_CmpB\_PwmA (

**PWM\_Handle** pwmHandle,

const **PWM\_ActionQual\_e** actionQual )

Sets the pulse width modulation (PWM) object action for PWM A when the counter equals CMPB and the counter is incrementing.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *actionQual* The action qualifier

---

#### 13.1.3.48 void PWM\_setActionQual\_CntUp\_CmpB\_PwmB (

**PWM\_Handle** pwmHandle,

const **PWM\_ActionQual\_e** actionQual )

Sets the pulse width modulation (PWM) object action for PWM B when the counter equals CMPB and the counter is incrementing.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *actionQual* The action qualifier

---

13.1.3.49 void PWM\_setActionQual\_Period\_PwmA (

**PWM\_Handle** pwmHandle,

const **PWM\_ActionQual\_e** actionQual )

Sets the pulse width modulation (PWM) object action for PWM A when the counter equals the period.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *actionQual* The action qualifier

---

13.1.3.50 void PWM\_setActionQual\_Period\_PwmB (

**PWM\_Handle** pwmHandle,

const **PWM\_ActionQual\_e** actionQual )

Sets the pulse width modulation (PWM) object action for PWM B when the counter equals the period.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *actionQual* The action qualifier

---

13.1.3.51 void PWM\_setActionQual\_Zero\_PwmA (

**PWM\_Handle** pwmHandle,

const **PWM\_ActionQual\_e** actionQual )

Sets the pulse width modulation (PWM) object action for PWM A when the counter equals the zero.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *actionQual* The action qualifier

---

13.1.3.52 void PWM\_setActionQual\_Zero\_PwmB (

**PWM\_Handle** pwmHandle,

const **PWM\_ActionQual\_e** actionQual )

Sets the pulse width modulation (PWM) object action for PWM B when the counter equals the zero.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *actionQual* The action qualifier

---

13.1.3.53 void PWM\_setChoppingClkFreq (

**PWM\_Handle** pwmHandle,

const **PWM\_ChoppingClkFreq\_e** clkFreq )

Sets the pulse width modulation (PWM) chopping clock frequency.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *clkFreq* The clock frequency

---

13.1.3.54 void PWM\_setChoppingDutyCycle (

**PWM\_Handle** pwmHandle,

const **PWM\_ChoppingDutyCycle\_e** dutyCycle )

Sets the pulse width modulation (PWM) chopping clock duty cycle.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *dutyCycle* The duty cycle

---

13.1.3.55 void PWM\_setChoppingPulseWidth (

**PWM\_Handle** pwmHandle,

const **PWM\_ChoppingPulseWidth\_e** pulseWidth )

Sets the pulse width modulation (PWM) chopping clock pulse width.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *pulseWidth* The pulse width

---

13.1.3.56 void PWM\_setClkDiv (

**PWM\_Handle** pwmHandle,

const **PWM\_ClkDiv\_e** clkDiv )

Sets the pulse width modulation (PWM) clock divisor.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *clkDiv* The clock divisor

---

13.1.3.57 void PWM\_setCmpA (

**PWM\_Handle** pwmHandle,

const uint16\_t pwmData ) [inline]

Writes the pulse width modulation (PWM) data value to the Counter Compare A hardware.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *pwmData* The PWM data value

---

References \_PWM\_Obj\_::CMPA.

13.1.3.58 void PWM\_setCmpB (

**PWM\_Handle** pwmHandle,

const uint16\_t pwmData ) [inline]

Writes the pulse width modulation (PWM) data value to the Counter Compare B hardware.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *pwmData* The PWM data value

---

References \_PWM\_Obj\_::CMPB.

13.1.3.59 void PWM\_setCount (

**PWM\_Handle** pwmHandle,

const uint16\_t count )

Sets the pulse width modulation (PWM) count.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *count* The count

---

#### 13.1.3.60 void PWM\_setCounterMode (

**PWM\_Handle** pwmHandle,  
const **PWM\_CounterMode\_e** counterMode )

Sets the pulse width modulation (PWM) counter mode.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *counterMode* The count mode

---

#### 13.1.3.61 void PWM\_setDeadBandFallingEdgeDelay (

**PWM\_Handle** pwmHandle,  
const uint16\_t delay )

Sets the pulse width modulation (PWM) deadband falling edge delay.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *delay* The delay

---

#### 13.1.3.62 void PWM\_setDeadBandInputMode (

**PWM\_Handle** pwmHandle,  
const **PWM\_DeadBandInputMode\_e** inputMode )

Sets the pulse width modulation (PWM) deadband input mode.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *inputMode* The input mode

---

#### 13.1.3.63 void PWM\_setDeadBandOutputMode (

**PWM\_Handle** pwmHandle,  
const **PWM\_DeadBandOutputMode\_e** outputMode )

Sets the pulse width modulation (PWM) deadband output mode.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *outputMode* The output mode

---

## 13.1.3.64 void PWM\_setDeadBandPolarity (

**PWM\_Handle** pwmHandle,  
const **PWM\_DeadBandPolarity\_e** polarity )

Sets the pulse width modulation (PWM) deadband polarity.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *polarity* The polarity

---

## 13.1.3.65 void PWM\_setDeadBandRisingEdgeDelay (

**PWM\_Handle** pwmHandle,  
const uint16\_t delay )

Sets the pulse width modulation (PWM) deadband rising edge delay.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *delay* The delay

---

## 13.1.3.66 void PWM\_setDigitalCompareAEvent1 (

**PWM\_Handle** pwmHandle,  
const bool\_t selectFilter,  
const bool\_t disableSync,  
const bool\_t enableSoc,  
const bool\_t generateSync )

Sets the pulse width modulation (PWM) digital compare A event 1 source parameters.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *selectFilter* Select filter output if true

---



---

in *disableSync* Asynchronous if true

---



---

in *enableSoc* Enable SOC generation if true

---



---

in *generateSync* Generate SYNC if true

---





722.7

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

in *selectFilter* Select filter output if true

in *disableSync* Asynchronous if true

13.1.3.70 void PWM\_setDigitalCompareBlankingPulse (

**PWM\_Handle** pwmHandle,

const **PWM\_DigitalCompare\_PulseSel\_e** pulseSelect )

Sets the pulse width modulation (PWM) digital compare blanking pulse.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

in *pulseSelect* The pulse selection

13.1.3.71 void PWM\_setDigitalCompareFilterOffset (

**PWM\_Handle** pwmHandle,

const uint16\_t offset )

Sets the pulse width modulation (PWM) digital compare filter offset.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

in *offset* The offset

13.1.3.72 void PWM\_setDigitalCompareFilterSource (

**PWM\_Handle** pwmHandle,

const **PWM\_DigitalCompare\_FilterSrc\_e** input )

Sets the pulse width modulation (PWM) digital compare filter source.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

in *input* The filter's source

13.1.3.73 void PWM\_setDigitalCompareFilterWindow (

**PWM\_Handle** pwmHandle,

const uint16\_t window )

Sets the pulse width modulation (PWM) digital compare filter offset.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *window* The window

---

13.1.3.74 void PWM\_setDigitalCompareInput (

**PWM\_Handle** pwmHandle,

const **PWM\_DigitalCompare\_Input\_e** input,

const **PWM\_DigitalCompare\_InputSel\_e** inputSel )

Sets the pulse width modulation (PWM) digital compare input.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *input* Comparator input to change

---

in *inputSel* Input selection for designated input

---

13.1.3.75 void PWM\_setHighSpeedClkDiv (

**PWM\_Handle** pwmHandle,

const **PWM\_HspClkDiv\_e** clkDiv )

Sets the pulse width modulation (PWM) high speed clock divisor.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *clkDiv* The clock divisor

---

13.1.3.76 void PWM\_setIntMode (

**PWM\_Handle** pwmHandle,

const **PWM\_IntMode\_e** intMode )

Sets the pulse width modulation (PWM) interrupt mode.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *intMode* The interrupt mode

---

## 13.1.3.77 void PWM\_setIntPeriod (

**PWM\_Handle** pwmHandle,  
const **PWM\_IntPeriod\_e** intPeriod )

Sets the pulse width modulation (PWM) interrupt period.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

in *intPeriod* The interrupt period

## 13.1.3.78 void PWM\_setLoadMode\_CmpA (

**PWM\_Handle** pwmHandle,  
const **PWM\_LoadMode\_e** loadMode )

Sets the pulse width modulation (PWM) load mode for CMPA.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

in *loadMode* The load mode

## 13.1.3.79 void PWM\_setLoadMode\_CmpB (

**PWM\_Handle** pwmHandle,  
const **PWM\_LoadMode\_e** loadMode )

Sets the pulse width modulation (PWM) load mode for CMPB.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

in *loadMode* The load mode

## 13.1.3.80 void PWM\_setOneShotTrip (

**PWM\_Handle** pwmHandle ) [inline]

Sets the pulse width modulation (PWM) one shot trip.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

References DISABLE\_PROTECTED\_REGISTER\_WRITE\_MODE, EN-  
ABLE\_PROTECTED\_REGISTER\_WRITE\_MODE, PWM\_TZFRC\_OST\_BITS, and  
\_PWM\_Obj\_::TZFRC.

#### 13.1.3.81 void PWM\_setPeriod (

**PWM\_Handle** pwmHandle,

const uint16\_t period )

Sets the pulse width modulation (PWM) period.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *period* The period

---

#### 13.1.3.82 void PWM\_setPeriodLoad (

**PWM\_Handle** pwmHandle,

const **PWM\_PeriodLoad\_e** periodLoad )

Sets the pulse width modulation (PWM) period load mode.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *periodLoad* The period load mode

---

#### 13.1.3.83 void PWM\_setPhase (

**PWM\_Handle** pwmHandle,

const uint16\_t phase )

Sets the pulse width modulation (PWM) phase.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *phase* The phase

---

#### 13.1.3.84 void PWM\_setPhaseDir (

**PWM\_Handle** pwmHandle,

const **PWM\_PhaseDir\_e** phaseDir )

Sets the pulse width modulation (PWM) phase direction.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *phaseDir* The phase direction

---

## 13.1.3.85 void PWM\_setRunMode (

**PWM\_Handle** pwmHandle,const **PWM\_RunMode\_e** runMode )

Sets the pulse width modulation (PWM) run mode.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handlein *runMode* The run mode

---

## 13.1.3.86 void PWM\_setShadowMode\_CmpA (

**PWM\_Handle** pwmHandle,const **PWM\_ShadowMode\_e** shadowMode )

Sets the pulse width modulation (PWM) shadow mode for CMPA.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handlein *shadowMode* The shadow mode

---

## 13.1.3.87 void PWM\_setShadowMode\_CmpB (

**PWM\_Handle** pwmHandle,const **PWM\_ShadowMode\_e** shadowMode )

Sets the pulse width modulation (PWM) shadow mode for CMPB.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handlein *shadowMode* The shadow mode

---

## 13.1.3.88 void PWM\_setSocAPeriod (

**PWM\_Handle** pwmHandle,const **PWM\_SocPeriod\_e** intPeriod )

Sets the pulse width modulation (PWM) start of conversion (SOC) A interrupt period.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handlein *intPeriod* The interrupt period

---

#### 13.1.3.89 void PWM\_setSocAPulseSrc (

**PWM\_Handle** pwmHandle,

const **PWM\_SocPulseSrc\_e** pulseSrc )

Sets the pulse width modulation (PWM) start of conversion (SOC) A interrupt pulse source.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *pulseSrc* The interrupt pulse source

---

#### 13.1.3.90 void PWM\_setSocBPeriod (

**PWM\_Handle** pwmHandle,

const **PWM\_SocPeriod\_e** intPeriod )

Sets the pulse width modulation (PWM) start of conversion (SOC) B interrupt period.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *intPeriod* The interrupt period

---

#### 13.1.3.91 void PWM\_setSocBPulseSrc (

**PWM\_Handle** pwmHandle,

const **PWM\_SocPulseSrc\_e** pulseSrc )

Sets the pulse width modulation (PWM) start of conversion (SOC) B interrupt pulse source.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *pulseSrc* The interrupt pulse source

---

#### 13.1.3.92 void PWM\_setSwSync (

**PWM\_Handle** pwmHandle )

Sets the pulse width modulation (PWM) software sync.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

#### 13.1.3.93 void PWM\_setSyncMode (

**PWM\_Handle** pwmHandle,

---

```
const PWM_SyncMode_e syncMode )
```

Sets the pulse width modulation (PWM) sync mode.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *syncMode* The sync mode

---

13.1.3.94 void PWM\_setTripZoneDCEventSelect\_DCAEVT1 (

**PWM\_Handle** pwmHandle,

const **PWM\_TripZoneDCEventSel\_e** tripZoneEvent )

Sets the pulse width modulation (PWM) trip zone digital compare event select for Digital Compare Output A Event 1 (DCAEVT1)

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *tripZoneEvent* The trip zone digital compare event

---

13.1.3.95 void PWM\_setTripZoneDCEventSelect\_DCAEVT2 (

**PWM\_Handle** pwmHandle,

const **PWM\_TripZoneDCEventSel\_e** tripZoneEvent )

Sets the pulse width modulation (PWM) trip zone digital compare event select for Digital Compare Output A Event 2 (DCAEVT2)

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *tripZoneEvent* The trip zone digital compare event

---

13.1.3.96 void PWM\_setTripZoneDCEventSelect\_DCBEVT1 (

**PWM\_Handle** pwmHandle,

const **PWM\_TripZoneDCEventSel\_e** tripZoneEvent )

Sets the pulse width modulation (PWM) trip zone digital compare event select for Digital Compare Output B Event 1 (DCBEVT1)

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *tripZoneEvent* The trip zone digital compare event

---

13.1.3.97 void PWM\_setTripZoneDCEventSelect\_DCBEVT2 (

**PWM\_Handle** pwmHandle,

const **PWM\_TripZoneDCEventSel\_e** tripZoneEvent )

Sets the pulse width modulation (PWM) trip zone digital compare event select for Digital Compare Output B Event 2 (DCBEVT2)

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *tripZoneEvent* The trip zone digital compare event

---

13.1.3.98 void PWM\_setTripZoneState\_DCAEVT1 (

**PWM\_Handle** pwmHandle,

const **PWM\_TripZoneState\_e** tripZoneState )

Sets the pulse width modulation (PWM) trip zone state for Digital Compare Output A Event 1 (DCAEVT1)

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *tripZoneState* The trip zone state

---

13.1.3.99 void PWM\_setTripZoneState\_DCAEVT2 (

**PWM\_Handle** pwmHandle,

const **PWM\_TripZoneState\_e** tripZoneState )

Sets the pulse width modulation (PWM) trip zone state for Digital Compare Output A Event 2 (DCAEVT2)

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *tripZoneState* The trip zone state

---

13.1.3.100 void PWM\_setTripZoneState\_DCBEVT1 (

**PWM\_Handle** pwmHandle,

const **PWM\_TripZoneState\_e** tripZoneState )

Sets the pulse width modulation (PWM) trip zone state for Digital Compare Output B Event 1 (DCBEVT1)



[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *tripZoneState* The trip zone state

---

13.1.3.101void PWM\_setTripZoneState\_DCBEVT2 (

**PWM\_Handle** pwmHandle,

const **PWM\_TripZoneState\_e** tripZoneState )

Sets the pulse width modulation (PWM) trip zone state for Digital Compare Output B Event 2 (DCBEVT1)

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *tripZoneState* The trip zone state

---

13.1.3.102void PWM\_setTripZoneState\_TZA (

**PWM\_Handle** pwmHandle,

const **PWM\_TripZoneState\_e** tripZoneState )

Sets the pulse width modulation (PWM) trip zone state for Output A (TZA)

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *tripZoneState* The trip zone state

---

13.1.3.103void PWM\_setTripZoneState\_TZB (

**PWM\_Handle** pwmHandle,

const **PWM\_TripZoneState\_e** tripZoneState )

Sets the pulse width modulation (PWM) trip zone state for Output B (TZB)

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *tripZoneState* The trip zone state

---

13.1.3.104void PWM\_write\_CmpA (

**PWM\_Handle** pwmHandle,

```
const int16_t pwmData ) [inline]
```

Writes the pulse width modulation (PWM) data value to the Counter Compare A hardware.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *pwmData* The PWM data value

---

References \_PWM\_Obj\_::CMPA, and \_PWM\_Obj\_::TBPRD.

13.1.3.105void PWM\_write\_CmpB (

**PWM\_Handle** pwmHandle,

```
const int16_t pwmData ) [inline]
```

Writes the pulse width modulation (PWM) data value to the Counter Compare B hardware.

[1]Parameters in *pwmHandle* The pulse width modulation (PWM) object handle

---

in *pwmData* The PWM data value

---

References \_PWM\_Obj\_::CMPB, and \_PWM\_Obj\_::TBPRD.

# 14 Power Control (PWR)

[Introduction](#) .....??

[API Functions](#) .. 147 The power API is a set of functions for configuring low power modes as well as other power related functions such as brown out.

This driver is contained in `f2802x0/common/source/pwr.c`, with `f2802x0/common/include/pwr.h` containing the API definitions for use by applications.

## 14.1 PWR

### Data Structures

struct [\\_PWR\\_Obj\\_](#)

### Macros

`#define` [PWR\\_BASE\\_ADDR](#)

`#define` [PWR\\_BORCFG\\_BORENZ\\_BITS](#)

`#define` [PWR\\_LPMCR0\\_LPM\\_BITS](#)

`#define` [PWR\\_LPMCR0\\_QUALSTDBY\\_BITS](#)

`#define` [PWR\\_LPMCR0\\_WDINTE\\_BITS](#)

### Typedefs

`typedef struct` [\\_PWR\\_Obj\\_](#) \* [PWR\\_Handle](#)

`typedef struct` [\\_PWR\\_Obj\\_](#) [PWR\\_Obj](#)

### Enumerations

`enum` [PWR\\_LowPowerMode\\_e](#) { [PWR\\_LowPowerMode\\_Idle](#), [PWR\\_LowPowerMode\\_Standby](#), [PWR\\_LowPowerMode\\_Halt](#) }

`enum` [PWR\\_NumStandByClocks\\_e](#) {  
[PWR\\_NumStandByClocks\\_2](#), [PWR\\_NumStandByClocks\\_3](#), [PWR\\_NumStandByClocks\\_4](#),  
[PWR\\_NumStandByClocks\\_5](#),  
[PWR\\_NumStandByClocks\\_6](#), [PWR\\_NumStandByClocks\\_7](#), [PWR\\_NumStandByClocks\\_8](#),  
[PWR\\_NumStandByClocks\\_9](#),  
[PWR\\_NumStandByClocks\\_10](#), [PWR\\_NumStandByClocks\\_11](#), [PWR\\_NumStandByClocks\\_12](#),

PWR\_NumStandByClocks\_13,  
PWR\_NumStandByClocks\_14, PWR\_NumStandByClocks\_15, PWR\_NumStandByClocks\_16,  
PWR\_NumStandByClocks\_17, PWR\_NumStandByClocks\_19, PWR\_NumStandByClocks\_20,  
PWR\_NumStandByClocks\_21, PWR\_NumStandByClocks\_23, PWR\_NumStandByClocks\_24,  
PWR\_NumStandByClocks\_25, PWR\_NumStandByClocks\_27, PWR\_NumStandByClocks\_28,  
PWR\_NumStandByClocks\_29, PWR\_NumStandByClocks\_31, PWR\_NumStandByClocks\_32,  
PWR\_NumStandByClocks\_30, PWR\_NumStandByClocks\_33, PWR\_NumStandByClocks\_35, PWR\_NumStandByClocks\_36,  
PWR\_NumStandByClocks\_34, PWR\_NumStandByClocks\_37, PWR\_NumStandByClocks\_39, PWR\_NumStandByClocks\_40,  
PWR\_NumStandByClocks\_38, PWR\_NumStandByClocks\_41, PWR\_NumStandByClocks\_43, PWR\_NumStandByClocks\_44,  
PWR\_NumStandByClocks\_42, PWR\_NumStandByClocks\_45, PWR\_NumStandByClocks\_47, PWR\_NumStandByClocks\_48,  
PWR\_NumStandByClocks\_46, PWR\_NumStandByClocks\_49, PWR\_NumStandByClocks\_51, PWR\_NumStandByClocks\_52,  
PWR\_NumStandByClocks\_50, PWR\_NumStandByClocks\_53, PWR\_NumStandByClocks\_55, PWR\_NumStandByClocks\_56,  
PWR\_NumStandByClocks\_54, PWR\_NumStandByClocks\_57, PWR\_NumStandByClocks\_59, PWR\_NumStandByClocks\_60,  
PWR\_NumStandByClocks\_58, PWR\_NumStandByClocks\_61, PWR\_NumStandByClocks\_63, PWR\_NumStandByClocks\_64,  
PWR\_NumStandByClocks\_62, PWR\_NumStandByClocks\_65 }

## Functions

void PWR\_disableBrownOutReset (PWR\_Handle pwrHandle)

void PWR\_disableWatchDogInt (PWR\_Handle pwrHandle)

void PWR\_enableBrownOutReset (PWR\_Handle pwrHandle)

void PWR\_enableWatchDogInt (PWR\_Handle pwrHandle)

PWR\_Handle PWR\_init (void \*pMemory, const size\_t numBytes)

void PWR\_setLowPowerMode (PWR\_Handle pwrHandle, const PWR\_LowPowerMode\_e lowPowerMode)

void PWR\_setNumStandByClocks (PWR\_Handle pwrHandle, const PWR\_NumStandByClocks\_e numClkCycles)

### 14.1.1 Detailed Description

### 14.1.2 Enumeration Type Documentation

#### 14.1.2.1 enum **PWR\_LowPowerMode\_e**

Enumeration to define the power (PWR) low power modes.

**Enumerator**

***PWR\_LowPowerMode\_Idle*** Denotes the idle mode.

***PWR\_LowPowerMode\_Standby*** Denotes the standby mode.

***PWR\_LowPowerMode\_Halt*** Denotes the halt mode.

#### 14.1.2.2 enum **PWR\_NumStandByClocks\_e**

Enumeration to define the power (PWR) number of standby clock cycles.

**Enumerator**

***PWR\_NumStandByClocks\_2*** Denotes 2 standby clock cycles.

***PWR\_NumStandByClocks\_3*** Denotes 3 standby clock cycles.

***PWR\_NumStandByClocks\_4*** Denotes 4 standby clock cycles.

***PWR\_NumStandByClocks\_5*** Denotes 5 standby clock cycles.

***PWR\_NumStandByClocks\_6*** Denotes 6 standby clock cycles.

***PWR\_NumStandByClocks\_7*** Denotes 7 standby clock cycles.

***PWR\_NumStandByClocks\_8*** Denotes 8 standby clock cycles.

***PWR\_NumStandByClocks\_9*** Denotes 9 standby clock cycles.

***PWR\_NumStandByClocks\_10*** Denotes 10 standby clock cycles.

***PWR\_NumStandByClocks\_11*** Denotes 11 standby clock cycles.

***PWR\_NumStandByClocks\_12*** Denotes 12 standby clock cycles.

***PWR\_NumStandByClocks\_13*** Denotes 13 standby clock cycles.

***PWR\_NumStandByClocks\_14*** Denotes 14 standby clock cycles.

***PWR\_NumStandByClocks\_15*** Denotes 15 standby clock cycles.

***PWR\_NumStandByClocks\_16*** Denotes 16 standby clock cycles.

***PWR\_NumStandByClocks\_17*** Denotes 17 standby clock cycles.

***PWR\_NumStandByClocks\_18*** Denotes 18 standby clock cycles.

***PWR\_NumStandByClocks\_19*** Denotes 19 standby clock cycles.

***PWR\_NumStandByClocks\_20*** Denotes 20 standby clock cycles.

***PWR\_NumStandByClocks\_21*** Denotes 21 standby clock cycles.

***PWR\_NumStandByClocks\_22*** Denotes 22 standby clock cycles.

***PWR\_NumStandByClocks\_23*** Denotes 23 standby clock cycles.

***PWR\_NumStandByClocks\_24*** Denotes 24 standby clock cycles.

***PWR\_NumStandByClocks\_25*** Denotes 25 standby clock cycles.

***PWR\_NumStandByClocks\_26*** Denotes 26 standby clock cycles.

***PWR\_NumStandByClocks\_27*** Denotes 27 standby clock cycles.

***PWR\_NumStandByClocks\_28*** Denotes 28 standby clock cycles.

<b><i>PWR_NumStandByClocks_29</i></b>	Denotes 29 standby clock cycles.
<b><i>PWR_NumStandByClocks_30</i></b>	Denotes 30 standby clock cycles.
<b><i>PWR_NumStandByClocks_31</i></b>	Denotes 31 standby clock cycles.
<b><i>PWR_NumStandByClocks_32</i></b>	Denotes 32 standby clock cycles.
<b><i>PWR_NumStandByClocks_33</i></b>	Denotes 33 standby clock cycles.
<b><i>PWR_NumStandByClocks_34</i></b>	Denotes 34 standby clock cycles.
<b><i>PWR_NumStandByClocks_35</i></b>	Denotes 35 standby clock cycles.
<b><i>PWR_NumStandByClocks_36</i></b>	Denotes 36 standby clock cycles.
<b><i>PWR_NumStandByClocks_37</i></b>	Denotes 37 standby clock cycles.
<b><i>PWR_NumStandByClocks_38</i></b>	Denotes 38 standby clock cycles.
<b><i>PWR_NumStandByClocks_39</i></b>	Denotes 39 standby clock cycles.
<b><i>PWR_NumStandByClocks_40</i></b>	Denotes 40 standby clock cycles.
<b><i>PWR_NumStandByClocks_41</i></b>	Denotes 41 standby clock cycles.
<b><i>PWR_NumStandByClocks_42</i></b>	Denotes 42 standby clock cycles.
<b><i>PWR_NumStandByClocks_43</i></b>	Denotes 43 standby clock cycles.
<b><i>PWR_NumStandByClocks_44</i></b>	Denotes 44 standby clock cycles.
<b><i>PWR_NumStandByClocks_45</i></b>	Denotes 45 standby clock cycles.
<b><i>PWR_NumStandByClocks_46</i></b>	Denotes 46 standby clock cycles.
<b><i>PWR_NumStandByClocks_47</i></b>	Denotes 47 standby clock cycles.
<b><i>PWR_NumStandByClocks_48</i></b>	Denotes 48 standby clock cycles.
<b><i>PWR_NumStandByClocks_49</i></b>	Denotes 49 standby clock cycles.
<b><i>PWR_NumStandByClocks_50</i></b>	Denotes 50 standby clock cycles.
<b><i>PWR_NumStandByClocks_51</i></b>	Denotes 51 standby clock cycles.
<b><i>PWR_NumStandByClocks_52</i></b>	Denotes 52 standby clock cycles.
<b><i>PWR_NumStandByClocks_53</i></b>	Denotes 53 standby clock cycles.
<b><i>PWR_NumStandByClocks_54</i></b>	Denotes 54 standby clock cycles.
<b><i>PWR_NumStandByClocks_55</i></b>	Denotes 55 standby clock cycles.
<b><i>PWR_NumStandByClocks_56</i></b>	Denotes 56 standby clock cycles.
<b><i>PWR_NumStandByClocks_57</i></b>	Denotes 57 standby clock cycles.
<b><i>PWR_NumStandByClocks_58</i></b>	Denotes 58 standby clock cycles.
<b><i>PWR_NumStandByClocks_59</i></b>	Denotes 59 standby clock cycles.
<b><i>PWR_NumStandByClocks_60</i></b>	Denotes 60 standby clock cycles.
<b><i>PWR_NumStandByClocks_61</i></b>	Denotes 61 standby clock cycles.
<b><i>PWR_NumStandByClocks_62</i></b>	Denotes 62 standby clock cycles.
<b><i>PWR_NumStandByClocks_63</i></b>	Denotes 63 standby clock cycles.
<b><i>PWR_NumStandByClocks_64</i></b>	Denotes 64 standby clock cycles.
<b><i>PWR_NumStandByClocks_65</i></b>	Denotes 65 standby clock cycles.

### 14.1.3 Function Documentation

#### 14.1.3.1 void PWR\_disableBrownOutReset (

**PWR\_Handle** pwrHandle )

Disables the brownout reset functions.

[1]Parameters in *pwrHandle* The power (PWR) object handle



722.7

*in lowPowerMode* The low power mode

---

14.1.3.7 void PWR\_setNumStandByClocks (

**PWR\_Handle** pwrHandle,

const **PWR\_NumStandByClocks\_e** numClkCycles )

Sets the number of standby clock cycles.

[1]Parameters *in pwrHandle* The power (PWR) object handle

---

*in numClkCycles* The number of standby clock cycles

---



# 15 Serial Communications Interface (SCI)

[Introduction](#) ..... [??](#)

[API Functions](#) ..... 153 The Serial Communication Interface (SCI) API provides a set of functions for configuring and using the SCI peripheral(s) on this device.

This driver is contained in `f2802x0/common/source/sci.c`, with `f2802x0/common/include/sci.h` containing the API definitions for use by applications.

## 15.1 SCI

### Data Structures

struct `_SCI_Obj`

### Macros

```
#define SCI_SCICCR_CHAR_LENGTH_BITS
#define SCI_SCICCR_LB_ENA_BITS
#define SCI_SCICCR_MODE_BITS
#define SCI_SCICCR_PARITY_BITS
#define SCI_SCICCR_PARITY_ENA_BITS
#define SCI_SCICCR_STOP_BITS
#define SCI_SCICTL1_RESET_BITS
#define SCI_SCICTL1_RX_ERR_INT_ENA_BITS
#define SCI_SCICTL1_RXENA_BITS
#define SCI_SCICTL1_SLEEP_BITS
#define SCI_SCICTL1_TXENA_BITS
#define SCI_SCICTL1_TXWAKE_BITS
#define SCI_SCICTL2_RX_INT_ENA_BITS
#define SCI_SCICTL2_TX_INT_ENA_BITS
#define SCI_SCICTL2_TXEMPTY_BITS
#define SCI_SCICTL2_TXRDY_BITS
#define SCI_SCIFFCT_ABD_BITS
```

```
#define SCI_SCIFFCT_ABDCLR_BITS
#define SCI_SCIFFCT_CDC_BITS
#define SCI_SCIFFCT_DELAY_BITS
#define SCI_SCIFFRX_FIFO_OVF_BITS
#define SCI_SCIFFRX_FIFO_OVFCLR_BITS
#define SCI_SCIFFRX_FIFO_RESET_BITS
#define SCI_SCIFFRX_FIFO_ST_BITS
#define SCI_SCIFFRX_IENA_BITS
#define SCI_SCIFFRX_IL_BITS
#define SCI_SCIFFRX_INT_BITS
#define SCI_SCIFFRX_INTCLR_BITS
#define SCI_SCIFFTX_CHAN_RESET_BITS
#define SCI_SCIFFTX_FIFO_ENA_BITS
#define SCI_SCIFFTX_FIFO_RESET_BITS
#define SCI_SCIFFTX_FIFO_ST_BITS
#define SCI_SCIFFTX_IENA_BITS
#define SCI_SCIFFTX_IL_BITS
#define SCI_SCIFFTX_INT_BITS
#define SCI_SCIFFTX_INTCLR_BITS
#define SCI_SCIRXST_BRKDT_BITS
#define SCI_SCIRXST_FE_BITS
#define SCI_SCIRXST_OE_BITS
#define SCI_SCIRXST_PE_BITS
#define SCI_SCIRXST_RXERROR_BITS
#define SCI_SCIRXST_RXRDY_BITS
#define SCI_SCIRXST_RXWAKE_BITS
#define SCIA_BASE_ADDR
```

## Typedefs

```
typedef struct _SCI_Obj * SCI_Handle
```

```
typedef struct _SCI_Obj SCI_Obj
```

## Enumerations

```
enum SCI_BaudRate_e { SCI_BaudRate_9_6_kBaud, SCI_BaudRate_19_2_kBaud,  
SCI_BaudRate_57_6_kBaud, SCI_BaudRate_115_2_kBaud }
```

```
enum SCI_CharLength_e {  
SCI_CharLength_1_Bit, SCI_CharLength_2_Bits, SCI_CharLength_3_Bits,  
SCI_CharLength_4_Bits, SCI_CharLength_5_Bits, SCI_CharLength_6_Bits, SCI_CharLength_7_Bits,  
SCI_CharLength_8_Bits }
```

```
enum SCI_FifoLevel_e {  
SCI_FifoLevel_Empty, SCI_FifoLevel_1_Word, SCI_FifoLevel_2_Words, SCI_FifoLevel_3_Words,  
SCI_FifoLevel_4_Words }
```

```
enum SCI_FifoStatus_e {  
SCI_FifoStatus_Empty, SCI_FifoStatus_1_Word, SCI_FifoStatus_2_Words,  
SCI_FifoStatus_3_Words, SCI_FifoStatus_4_Words }
```

```
enum SCI_Mode_e { SCI_Mode_IdleLine, SCI_Mode_AddressBit }
```

```
enum SCI_NumStopBits_e { SCI_NumStopBits_One, SCI_NumStopBits_Two }
```

```
enum SCI_Parity_e { SCI_Parity_Odd, SCI_Parity_Even }
```

```
enum SCI_Priority_e { SCI_Priority_Immediate, SCI_Priority_FreeRun,  
SCI_Priority_AfterRxRxSeq }
```

## Functions

```
void SCI_clearAutoBaudDetect (SCI_Handle sciHandle)
```

```
void SCI_clearRxFifoInt (SCI_Handle sciHandle)
```

```
void SCI_clearRxFifoOvf (SCI_Handle sciHandle)
```

```
void SCI_clearTxFifoInt (SCI_Handle sciHandle)
```

```
void SCI_disable (SCI_Handle sciHandle)
```

```
void SCI_disableAutoBaudAlign (SCI_Handle sciHandle)
```

```
void SCI_disableFifoEnh (SCI_Handle sciHandle)
```

```
void SCI_disableLoopBack (SCI_Handle sciHandle)
```

```
void SCI_disableParity (SCI_Handle sciHandle)
void SCI_disableRx (SCI_Handle sciHandle)
void SCI_disableRxErrorInt (SCI_Handle sciHandle)
void SCI_disableRxFifoInt (SCI_Handle sciHandle)
void SCI_disableRxInt (SCI_Handle sciHandle)
void SCI_disableSleep (SCI_Handle sciHandle)
void SCI_disableTx (SCI_Handle sciHandle)
void SCI_disableTxFifoInt (SCI_Handle sciHandle)
void SCI_disableTxInt (SCI_Handle sciHandle)
void SCI_disableTxWake (SCI_Handle sciHandle)
void SCI_enable (SCI_Handle sciHandle)
void SCI_enableAutoBaudAlign (SCI_Handle sciHandle)
void SCI_enableFifoEnh (SCI_Handle sciHandle)
void SCI_enableLoopBack (SCI_Handle sciHandle)
void SCI_enableParity (SCI_Handle sciHandle)
void SCI_enableRx (SCI_Handle sciHandle)
void SCI_enableRxErrorInt (SCI_Handle sciHandle)
void SCI_enableRxFifoInt (SCI_Handle sciHandle)
void SCI_enableRxInt (SCI_Handle sciHandle)
void SCI_enableSleep (SCI_Handle sciHandle)
void SCI_enableTx (SCI_Handle sciHandle)
void SCI_enableTxFifoInt (SCI_Handle sciHandle)
void SCI_enableTxInt (SCI_Handle sciHandle)
void SCI_enableTxWake (SCI_Handle sciHandle)
uint16_t SCI_getData (SCI_Handle sciHandle)
uint16_t SCI_getDataBlocking (SCI_Handle sciHandle)
uint16_t SCI_getDataNonBlocking (SCI_Handle sciHandle, uint16_t *success)
SCI_FifoStatus_e SCI_getRxFifoStatus (SCI_Handle sciHandle)
SCI_FifoStatus_e SCI_getTxFifoStatus (SCI_Handle sciHandle)
```

```
SCI_Handle SCI_init (void *pMemory, const size_t numBytes)

bool_t SCI_isRxDataReady (SCI_Handle sciHandle)

bool_t SCI_isTxReady (SCI_Handle sciHandle)

void SCI_putData (SCI_Handle sciHandle, const uint16_t data)

void SCI_putDataBlocking (SCI_Handle sciHandle, uint16_t data)

uint16_t SCI_putDataNonBlocking (SCI_Handle sciHandle, uint16_t data)

void SCI_reset (SCI_Handle sciHandle)

void SCI_resetChannels (SCI_Handle sciHandle)

void SCI_resetRxFifo (SCI_Handle sciHandle)

void SCI_resetTxFifo (SCI_Handle sciHandle)

void SCI_setBaudRate (SCI_Handle sciHandle, const SCI_BaudRate_e baudRate)

void SCI_setCharLength (SCI_Handle sciHandle, const SCI_CharLength_e charLength)

void SCI_setMode (SCI_Handle sciHandle, const SCI_Mode_e mode)

void SCI_setNumStopBits (SCI_Handle sciHandle, const SCI_NumStopBits_e numBits)

void SCI_setParity (SCI_Handle sciHandle, const SCI_Parity_e parity)

void SCI_setPriority (SCI_Handle sciHandle, const SCI_Priority_e priority)

void SCI_setRxFifoIntLevel (SCI_Handle sciHandle, const SCI_FifoLevel_e fifoLevel)

void SCI_setTxDelay (SCI_Handle sciHandle, const uint8_t delay)

void SCI_setTxFifoIntLevel (SCI_Handle sciHandle, const SCI_FifoLevel_e fifoLevel)
```

### 15.1.1 Detailed Description

### 15.1.2 Enumeration Type Documentation

#### 15.1.2.1 enum **SCI\_BaudRate\_e**

Enumeration to define the serial communications interface (SCI) baud rates. This enumeration assume a device clock of 50Mhz and a LSPCLK of 12.5 MHz.

##### Enumerator

**SCI\_BaudRate\_9\_6\_kBaud** Denotes 9.6 kBaud.  
**SCI\_BaudRate\_19\_2\_kBaud** Denotes 19.2 kBaud.  
**SCI\_BaudRate\_57\_6\_kBaud** Denotes 57.6 kBaud.  
**SCI\_BaudRate\_115\_2\_kBaud** Denotes 115.2 kBaud.

### 15.1.2.2 enum **SCI\_CharLength\_e**

Enumeration to define the serial communications interface (SCI) character lengths.

#### Enumerator

- SCI\_CharLength\_1\_Bit** Denotes a character length of 1 bit.
- SCI\_CharLength\_2\_Bits** Denotes a character length of 2 bits.
- SCI\_CharLength\_3\_Bits** Denotes a character length of 3 bits.
- SCI\_CharLength\_4\_Bits** Denotes a character length of 4 bits.
- SCI\_CharLength\_5\_Bits** Denotes a character length of 5 bits.
- SCI\_CharLength\_6\_Bits** Denotes a character length of 6 bits.
- SCI\_CharLength\_7\_Bits** Denotes a character length of 7 bits.
- SCI\_CharLength\_8\_Bits** Denotes a character length of 8 bits.

### 15.1.2.3 enum **SCI\_FifoLevel\_e**

Enumeration to define the serial communications interface (SCI) FIFO level.

#### Enumerator

- SCI\_FifoLevel\_Empty** Denotes the fifo is empty.
- SCI\_FifoLevel\_1\_Word** Denotes the fifo contains 1 word.
- SCI\_FifoLevel\_2\_Words** Denotes the fifo contains 2 words.
- SCI\_FifoLevel\_3\_Words** Denotes the fifo contains 3 words.
- SCI\_FifoLevel\_4\_Words** Denotes the fifo contains 4 words.

### 15.1.2.4 enum **SCI\_FifoStatus\_e**

Enumeration to define the serial communications interface (SCI) FIFO status.

#### Enumerator

- SCI\_FifoStatus\_Empty** Denotes the fifo is empty.
- SCI\_FifoStatus\_1\_Word** Denotes the fifo contains 1 word.
- SCI\_FifoStatus\_2\_Words** Denotes the fifo contains 2 words.
- SCI\_FifoStatus\_3\_Words** Denotes the fifo contains 3 words.
- SCI\_FifoStatus\_4\_Words** Denotes the fifo contains 4 words.

### 15.1.2.5 enum **SCI\_Mode\_e**

Enumeration to define the serial communications interface (SCI) multiprocessor protocol mode.

#### Enumerator

- SCI\_Mode\_IdleLine** Denotes idle-line mode protocol.
- SCI\_Mode\_AddressBit** Denotes address-bit mode protocol.

### 15.1.2.6 enum **SCI\_NumStopBits\_e**

Enumeration to define the serial communications interface (SCI) number of stop bits.

**Enumerator**

**SCI\_NumStopBits\_One** Denotes 1 stop bit.

**SCI\_NumStopBits\_Two** Denotes 2 stop bits.

### 15.1.2.7 enum **SCI\_Parity\_e**

Enumeration to define the serial communications interface (SCI) parity.

**Enumerator**

**SCI\_Parity\_Odd** Denotes odd parity.

**SCI\_Parity\_Even** Denotes even parity.

### 15.1.2.8 enum **SCI\_Priority\_e**

Enumeration to define the serial communications interface (SCI) emulation suspend priority.

**Enumerator**

**SCI\_Priority\_Immediate** Denotes an immediate stop.

**SCI\_Priority\_FreeRun** Denotes free running.

**SCI\_Priority\_AfterRxRxSeq** Denotes that a stop after the current receive/transmit sequence.

## 15.1.3 Function Documentation

### 15.1.3.1 void SCI\_clearAutoBaudDetect (

**SCI\_Handle** sciHandle )

Clears the auto baud detect mode.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

### 15.1.3.2 void SCI\_clearRxFifoInt (

**SCI\_Handle** sciHandle )

Clears the Rx FIFO interrupt flag.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.3 void SCI\_clearRxFifoOvf (

**SCI\_Handle** sciHandle )

Clears the Rx FIFO overflow flag.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.4 void SCI\_clearTxFifoInt (

**SCI\_Handle** sciHandle )

Clears the Tx FIFO interrupt flag.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.5 void SCI\_disable (

**SCI\_Handle** sciHandle )

Disables the serial communications interface (SCI) interrupt.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.6 void SCI\_disableAutoBaudAlign (

**SCI\_Handle** sciHandle )

Disable the serial communications interface (SCI) auto baud alignment.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.7 void SCI\_disableFifoEnh (

**SCI\_Handle** sciHandle )

Disables the serial communications interface (SCI) FIFO enhancements.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.8 void SCI\_disableLoopBack (

**SCI\_Handle** sciHandle )

Disables the serial peripheral interface (SCI) loop back mode.



[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.9 void SCI\_disableParity (

**SCI\_Handle** sciHandle )

Disable the parity.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.10 void SCI\_disableRx (

**SCI\_Handle** sciHandle )

Disables the serial communications interface (SCI) master/slave receive mode.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.11 void SCI\_disableRxErrorInt (

**SCI\_Handle** sciHandle )

Disables the serial communications interface (SCI) receive error interrupt.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.12 void SCI\_disableRxFifoInt (

**SCI\_Handle** sciHandle )

Disables the serial communications interface (SCI) receive FIFO interrupt.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.13 void SCI\_disableRxInt (

**SCI\_Handle** sciHandle )

Disables the serial communications interface (SCI) receive interrupt.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.14 void SCI\_disableSleep (

**SCI\_Handle** sciHandle )

Disables the serial communications interface (SCI) sleep mode.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.15 void SCI\_disableTx (

**SCI\_Handle** sciHandle )

Disables the serial communications interface (SCI) master/slave transmit mode.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.16 void SCI\_disableTxFifoInt (

**SCI\_Handle** sciHandle )

Disables the serial communications interface (SCI) transmit FIFO interrupt.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.17 void SCI\_disableTxInt (

**SCI\_Handle** sciHandle )

Disables the serial communications interface (SCI) transmit interrupt.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.18 void SCI\_disableTxWake (

**SCI\_Handle** sciHandle )

Disables the serial communications interface (SCI) wakeup method.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.19 void SCI\_enable (

**SCI\_Handle** sciHandle )

Enables the serial communications interface (SCI)

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.20 void SCI\_enableAutoBaudAlign (

**SCI\_Handle** sciHandle )

Enable the serial communications interface (SCI) auto baud alignment.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.21 void SCI\_enableFifoEnh (

**SCI\_Handle** sciHandle )

Enables the serial communications interface (SCI) FIFO enhancements.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.22 void SCI\_enableLoopBack (

**SCI\_Handle** sciHandle )

Enables the serial peripheral interface (SCI) loop back mode.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.23 void SCI\_enableParity (

**SCI\_Handle** sciHandle )

Enables the serial peripheral interface (SCI) parity.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.24 void SCI\_enableRx (

**SCI\_Handle** sciHandle )

Enables the serial peripheral interface (SCI) receiver.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.25 void SCI\_enableRxErrorInt (

**SCI\_Handle** sciHandle )

Enables the serial communications interface (SCI) receive error interrupt.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.26 void SCI\_enableRxFifoInt (

**SCI\_Handle** sciHandle )

Enables the serial communications interface (SCI) receive FIFO interrupt.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.27 void SCI\_enableRxInt (

**SCI\_Handle** sciHandle )

Enables the serial communications interface (SCI) receive interrupt.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.28 void SCI\_enableSleep (

**SCI\_Handle** sciHandle )

Enables the serial communications interface (SCI) sleep mode.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.29 void SCI\_enableTx (

**SCI\_Handle** sciHandle )

Enables the serial communications interface (SCI) master/slave transmit mode.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

#### 15.1.3.30 void SCI\_enableTxFifoInt (

**SCI\_Handle** sciHandle )

Enables the serial communications interface (SCI) transmit FIFO interrupt.



722.7

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

out *success* Pointer to a variable which will house whether the read was successful or not (true on success)

---

Returns Data if successful, or NULL if no characters

---

#### 15.1.3.36 **SCI\_FifoStatus\_e** SCI\_getRxFifoStatus (

**SCI\_Handle** sciHandle )

Gets the serial communications interface (SCI) receive FIFO status.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

Returns The receive FIFO status

---

#### 15.1.3.37 **SCI\_FifoStatus\_e** SCI\_getTxFifoStatus (

**SCI\_Handle** sciHandle )

Gets the serial communications interface (SCI) transmit FIFO status.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

Returns The transmit FIFO status

---

#### 15.1.3.38 **SCI\_Handle** SCI\_init (

void \* pMemory,

const size\_t numBytes )

Initializes the serial communications interface (SCI) object handle.

[1]Parameters in *pMemory* A pointer to the base address of the SCI registers

---

in *numBytes* The number of bytes allocated for the SCI object, bytes

---

Returns The serial communications interface (SCI) object handle

---

#### 15.1.3.39 **bool\_t** SCI\_isRxDataReady (

**SCI\_Handle** sciHandle ) [inline]

Determines if the serial communications interface (SCI) has receive data ready.

---

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

Returns The receive data status

References SCI\_SCIRXST\_RXRDY\_BITS, and \_SCI\_Obj\_::SCIRXST.

#### 15.1.3.40 bool\_t SCI\_isTxReady (

**SCI\_Handle** sciHandle ) [inline]

Determines if the serial communications interface (SCI) is ready to transmit.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

Returns The transmit status

References SCI\_SCICTL2\_TXRDY\_BITS, and \_SCI\_Obj\_::SCICTL2.

#### 15.1.3.41 void SCI\_putData (

**SCI\_Handle** sciHandle,

const uint16\_t data ) [inline]

Writes data to the serial communications interface (SCI)

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

in *data* The data value

---

References \_SCI\_Obj\_::SCITXBUF.

#### 15.1.3.42 void SCI\_putDataBlocking (

**SCI\_Handle** sciHandle,

uint16\_t data )

Writes data to the serial communications interface (Blocking)

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

in *data* The data value

---

#### 15.1.3.43 uint16\_t SCI\_putDataNonBlocking (

**SCI\_Handle** sciHandle,

uint16\_t data )

Writes data to the serial communications interface (Non-Blocking)

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

in *data* The data value

---

Returns True on successful write, false if no space is available in the transmit buffer

15.1.3.44 void SCI\_reset (

**SCI\_Handle** sciHandle )

Resets the serial communications interface (SCI)

[1]Parameters in *sciHandle* The serial communication interface (SCI) object handle

---

15.1.3.45 void SCI\_resetChannels (

**SCI\_Handle** sciHandle )

Resets the serial communications interface (SCI) transmit and receive channels.

[1]Parameters in *sciHandle* The serial communication interface (SCI) object handle

---

15.1.3.46 void SCI\_resetRxFifo (

**SCI\_Handle** sciHandle )

Resets the serial communications interface (SCI) receive FIFO.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

15.1.3.47 void SCI\_resetTxFifo (

**SCI\_Handle** sciHandle )

Resets the serial communications interface (SCI) transmit FIFO.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

15.1.3.48 void SCI\_setBaudRate (

**SCI\_Handle** sciHandle,



const **SCI\_BaudRate\_e** baudRate )

Sets the serial communications interface (SCI) baud rate.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

in *baudRate* The baud rate

---

15.1.3.49 void SCI\_setCharLength (

**SCI\_Handle** sciHandle,

const **SCI\_CharLength\_e** charLength )

Sets the serial communications interface (SCI) character length.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

in *charLength* The character length

---

15.1.3.50 void SCI\_setMode (

**SCI\_Handle** sciHandle,

const **SCI\_Mode\_e** mode )

Sets the serial communications interface (SCI) multiprocessor mode.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

in *mode* The multiprocessor mode

---

15.1.3.51 void SCI\_setNumStopBits (

**SCI\_Handle** sciHandle,

const **SCI\_NumStopBits\_e** numBits )

Sets the serial communications interface (SCI) number of stop bits.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

in *numBits* The number of bits

---

15.1.3.52 void SCI\_setParity (

**SCI\_Handle** sciHandle,

const **SCI\_Parity\_e** parity )

Sets the serial communications interface (SCI) parity.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

in *parity* The parity

---

15.1.3.53 void SCI\_setPriority (

**SCI\_Handle** sciHandle,

const **SCI\_Priority\_e** priority )

Sets the serial communications interface (SCI) priority.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

in *priority* The priority

---

15.1.3.54 void SCI\_setRxFifoIntLevel (

**SCI\_Handle** sciHandle,

const **SCI\_FifoLevel\_e** fifoLevel )

Sets the serial communications interface (SCI) receive FIFO level for generating an interrupt.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

in *fifoLevel* The FIFO level

---

15.1.3.55 void SCI\_setTxDelay (

**SCI\_Handle** sciHandle,

const uint8\_t delay )

Sets the serial communications interface (SCI) transmit delay.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

in *delay* The transmit delay

---

15.1.3.56 void SCI\_setTxFifoIntLevel (

**SCI\_Handle** sciHandle,

const **SCI\_FifoLevel\_e** fifoLevel )

Sets the serial communications interface (SCI) transmit FIFO level for generating an interrupt.

[1]Parameters in *sciHandle* The serial communications interface (SCI) object handle

---

in *fifoLevel* The FIFO level

---



## 16 Serial Peripheral Interface (SPI)

[Introduction](#) ..... [??](#)  
[API Functions](#) ..... 173 The Serial Peripheral Interface (SPI) API provides a set of functions for configuring and using the device's SPI peripheral(s).

This driver is contained in `f2802x0/common/source/spi.c`, with `f2802x0/common/include/spi.h` containing the API definitions for use by applications.

### 16.1 SPI

#### Data Structures

struct `_SPI_Obj`

#### Macros

```
#define SPI_SPICCR_CHAR_LENGTH_BITS
#define SPI_SPICCR_CLKPOL_BITS
#define SPI_SPICCR_RESET_BITS
#define SPI_SPICCR_SPILBK_BITS
#define SPI_SPICTL_CLK_PHASE_BITS
#define SPI_SPICTL_INT_ENA_BITS
#define SPI_SPICTL_MODE_BITS
#define SPI_SPICTL_OVRRUN_INT_ENA_BITS
#define SPI_SPICTL_TALK_BITS
#define SPI_SPIFFRX_FIFO_OVF_BITS
#define SPI_SPIFFRX_FIFO_OVFCLR_BITS
#define SPI_SPIFFRX_FIFO_RESET_BITS
#define SPI_SPIFFRX_FIFO_ST_BITS
#define SPI_SPIFFRX_IENA_BITS
#define SPI_SPIFFRX_IL_BITS
#define SPI_SPIFFRX_INT_BITS
#define SPI_SPIFFRX_INTCLR_BITS
```

```
#define SPI_SPIFFTX_CHAN_RESET_BITS
#define SPI_SPIFFTX_FIFO_ENA_BITS
#define SPI_SPIFFTX_FIFO_RESET_BITS
#define SPI_SPIFFTX_FIFO_ST_BITS
#define SPI_SPIFFTX_IENA_BITS
#define SPI_SPIFFTX_IL_BITS
#define SPI_SPIFFTX_INT_BITS
#define SPI_SPIFFTX_INTCLR_BITS
#define SPIA_BASE_ADDR
```

## Typedefs

```
typedef struct _SPI_Obj * SPI_Handle
typedef struct _SPI_Obj SPI_Obj
```

## Enumerations

```
enum SPI_BaudRate_e { SPI_BaudRate_500_KBaud, SPI_BaudRate_1_MBaud }

enum SPI_CharLength_e {
SPI_CharLength_1_Bit,          SPI_CharLength_2_Bits,          SPI_CharLength_3_Bits,
SPI_CharLength_4_Bits,          SPI_CharLength_6_Bits,          SPI_CharLength_7_Bits,
SPI_CharLength_5_Bits,          SPI_CharLength_8_Bits,          SPI_CharLength_11_Bits,
SPI_CharLength_9_Bits,          SPI_CharLength_10_Bits,         SPI_CharLength_12_Bits,
SPI_CharLength_13_Bits,         SPI_CharLength_14_Bits,         SPI_CharLength_15_Bits,
SPI_CharLength_16_Bits }

enum SPI_ClkPhase_e { SPI_ClkPhase_Normal, SPI_ClkPhase_Delayed }

enum SPI_ClkPolarity_e { SPI_ClkPolarity_OutputRisingEdge_InputFallingEdge,
SPI_ClkPolarity_OutputFallingEdge_InputRisingEdge }

enum SPI_FifoLevel_e {
SPI_FifoLevel_Empty, SPI_FifoLevel_1_Word, SPI_FifoLevel_2_Words, SPI_FifoLevel_3_Words,
SPI_FifoLevel_4_Words }

enum SPI_FifoStatus_e {
SPI_FifoStatus_Empty,          SPI_FifoStatus_1_Word,          SPI_FifoStatus_2_Words,
SPI_FifoStatus_3_Words,
SPI_FifoStatus_4_Words }
```

```
enum SPI_Mode_e { SPI_Mode_Slave, SPI_Mode_Master }

enum SPI_Priority_e { SPI_Priority_Immediate, SPI_Priority_FreeRun, SPI_Priority_AfterRxRxSeq
}

enum SPI_StelInv_e { SPI_StelInv_ActiveLow, SPI_StelInv_ActiveHigh }

enum SPI_TriWire_e { SPI_TriWire_NormalFourWire, SPI_TriWire_ThreeWire }
```

## Functions

```
void SPI_clearRxFifoInt (SPI_Handle spiHandle)
void SPI_clearRxFifoOvf (SPI_Handle spiHandle)
void SPI_clearTxFifoInt (SPI_Handle spiHandle)
void SPI_disable (SPI_Handle spiHandle)
void SPI_disableChannels (SPI_Handle spiHandle)
void SPI_disableInt (SPI_Handle spiHandle)
void SPI_disableLoopBack (SPI_Handle spiHandle)
void SPI_disableOverRunInt (SPI_Handle spiHandle)
void SPI_disableRxFifo (SPI_Handle spiHandle)
void SPI_disableRxFifoInt (SPI_Handle spiHandle)
void SPI_disableTx (SPI_Handle spiHandle)
void SPI_disableTxFifo (SPI_Handle spiHandle)
void SPI_disableTxFifoEnh (SPI_Handle spiHandle)
void SPI_disableTxFifoInt (SPI_Handle spiHandle)
void SPI_enable (SPI_Handle spiHandle)
void SPI_enableChannels (SPI_Handle spiHandle)
void SPI_enableFifoEnh (SPI_Handle spiHandle)
void SPI_enableInt (SPI_Handle spiHandle)
void SPI_enableLoopBack (SPI_Handle spiHandle)
void SPI_enableOverRunInt (SPI_Handle spiHandle)
void SPI_enableRxFifo (SPI_Handle spiHandle)
void SPI_enableRxFifoInt (SPI_Handle spiHandle)
void SPI_enableTx (SPI_Handle spiHandle)
```

```
void SPI_enableTxFifo (SPI_Handle spiHandle)
void SPI_enableTxFifoInt (SPI_Handle spiHandle)
SPI_FifoStatus_e SPI_getRxFifoStatus (SPI_Handle spiHandle)
SPI_FifoStatus_e SPI_getTxFifoStatus (SPI_Handle spiHandle)
SPI_Handle SPI_init (void *pMemory, const size_t numBytes)
uint16_t SPI_read (SPI_Handle spiHandle)
void SPI_reset (SPI_Handle spiHandle)
void SPI_resetChannels (SPI_Handle spiHandle)
void SPI_resetRxFifo (SPI_Handle spiHandle)
void SPI_resetTxFifo (SPI_Handle spiHandle)
void SPI_setBaudRate (SPI_Handle spiHandle, const SPI_BaudRate_e baudRate)
void SPI_setCharLength (SPI_Handle spiHandle, const SPI_CharLength_e length)
void SPI_setClkPhase (SPI_Handle spiHandle, const SPI_ClkPhase_e clkPhase)
void SPI_setClkPolarity (SPI_Handle spiHandle, const SPI_ClkPolarity_e polarity)
void SPI_setMode (SPI_Handle spiHandle, const SPI_Mode_e mode)
void SPI_setPriority (SPI_Handle spiHandle, const SPI_Priority_e priority)
void SPI_setRxFifoIntLevel (SPI_Handle spiHandle, const SPI_FifoLevel_e fifoLevel)
void SPI_setStelInv (SPI_Handle spiHandle, const SPI_StelInv_e stelInv)
void SPI_setTriWire (SPI_Handle spiHandle, const SPI_TriWire_e triWire)
void SPI_setTxDelay (SPI_Handle spiHandle, const uint8_t delay)
void SPI_setTxFifoIntLevel (SPI_Handle spiHandle, const SPI_FifoLevel_e fifoLevel)
void SPI_write (SPI_Handle spiHandle, const uint16_t data)
void SPI_write8 (SPI_Handle spiHandle, const uint16_t data)
```

## 16.1.1 Detailed Description

## 16.1.2 Enumeration Type Documentation

### 16.1.2.1 enum **SPI\_BaudRate\_e**

Enumeration to define the serial peripheral interface (SPI) baud rates. These assume a LSCLK of 12.5MHz.



**Enumerator*****SPI\_BaudRate\_500\_KBaud*** Denotes 500 KBaud.***SPI\_BaudRate\_1\_MBaud*** Denotes 1 MBaud.16.1.2.2 enum **SPI\_CharLength\_e**

Enumeration to define the serial peripheral interface (SPI) character lengths.

**Enumerator*****SPI\_CharLength\_1\_Bit*** Denotes a character length of 1 bit.***SPI\_CharLength\_2\_Bits*** Denotes a character length of 2 bits.***SPI\_CharLength\_3\_Bits*** Denotes a character length of 3 bits.***SPI\_CharLength\_4\_Bits*** Denotes a character length of 4 bits.***SPI\_CharLength\_5\_Bits*** Denotes a character length of 5 bits.***SPI\_CharLength\_6\_Bits*** Denotes a character length of 6 bits.***SPI\_CharLength\_7\_Bits*** Denotes a character length of 7 bits.***SPI\_CharLength\_8\_Bits*** Denotes a character length of 8 bits.***SPI\_CharLength\_9\_Bits*** Denotes a character length of 9 bits.***SPI\_CharLength\_10\_Bits*** Denotes a character length of 10 bits.***SPI\_CharLength\_11\_Bits*** Denotes a character length of 11 bits.***SPI\_CharLength\_12\_Bits*** Denotes a character length of 12 bits.***SPI\_CharLength\_13\_Bits*** Denotes a character length of 13 bits.***SPI\_CharLength\_14\_Bits*** Denotes a character length of 14 bits.***SPI\_CharLength\_15\_Bits*** Denotes a character length of 15 bits.***SPI\_CharLength\_16\_Bits*** Denotes a character length of 16 bits.16.1.2.3 enum **SPI\_ClkPhase\_e**

Enumeration to define the serial peripheral interface (SPI) clock phase.

**Enumerator*****SPI\_ClkPhase\_Normal*** Denotes a normal clock scheme.***SPI\_ClkPhase\_Delayed*** Denotes that the SPICLK signal is delayed by one half-cycle.16.1.2.4 enum **SPI\_ClkPolarity\_e**

Enumeration to define the serial peripheral interface (SPI) clock polarity for the input and output data.

**Enumerator*****SPI\_ClkPolarity\_OutputRisingEdge\_InputFallingEdge*** Denotes that the tx data is output on the rising edge, the rx data is latched on the falling edge.***SPI\_ClkPolarity\_OutputFallingEdge\_InputRisingEdge*** Denotes that the tx data is output on the falling edge, the rx data is latched on the rising edge.

#### 16.1.2.5 enum **SPI\_FifoLevel\_e**

Enumeration to define the serial peripheral interface (SPI) FIFO level.

##### Enumerator

- SPI\_FifoLevel\_Empty** Denotes the fifo is empty.
- SPI\_FifoLevel\_1\_Word** Denotes the fifo contains 1 word.
- SPI\_FifoLevel\_2\_Words** Denotes the fifo contains 2 words.
- SPI\_FifoLevel\_3\_Words** Denotes the fifo contains 3 words.
- SPI\_FifoLevel\_4\_Words** Denotes the fifo contains 4 words.

#### 16.1.2.6 enum **SPI\_FifoStatus\_e**

Enumeration to define the serial peripheral interface (SPI) FIFO status.

##### Enumerator

- SPI\_FifoStatus\_Empty** Denotes the fifo is empty.
- SPI\_FifoStatus\_1\_Word** Denotes the fifo contains 1 word.
- SPI\_FifoStatus\_2\_Words** Denotes the fifo contains 2 words.
- SPI\_FifoStatus\_3\_Words** Denotes the fifo contains 3 words.
- SPI\_FifoStatus\_4\_Words** Denotes the fifo contains 4 words.

#### 16.1.2.7 enum **SPI\_Mode\_e**

Enumeration to define the serial peripheral interface (SPI) network mode control.

##### Enumerator

- SPI\_Mode\_Slave** Denotes slave mode.
- SPI\_Mode\_Master** Denotes master mode.

#### 16.1.2.8 enum **SPI\_Priority\_e**

##### Enumerator

- SPI\_Priority\_Immediate** Stops immediately after EMU halt.
- SPI\_Priority\_FreeRun** Doesn't stop after EMU halt.
- SPI\_Priority\_AfterRxRxSeq** Stops after EMU halt and next rx/rx sequence.

#### 16.1.2.9 enum **SPI\_Stelnv\_e**

##### Enumerator

- SPI\_Stelnv\_ActiveLow** Denotes active low STE pin.
- SPI\_Stelnv\_ActiveHigh** Denotes active high STE pin.



722.7

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

16.1.3.6 void SPI\_disableInt (  
**SPI\_Handle** spiHandle )

Disables the serial peripheral interface (SPI) interrupt.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

16.1.3.7 void SPI\_disableLoopBack (  
**SPI\_Handle** spiHandle )

Disables the serial peripheral interface (SPI) loop back mode.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

16.1.3.8 void SPI\_disableOverRunInt (  
**SPI\_Handle** spiHandle )

Disables the serial peripheral interface (SPI) over-run interrupt.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

16.1.3.9 void SPI\_disableRxFifo (  
**SPI\_Handle** spiHandle )

Disables the serial peripheral interface (SPI) receive FIFO.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

16.1.3.10 void SPI\_disableRxFifoInt (  
**SPI\_Handle** spiHandle )

Disables the serial peripheral interface (SPI) receive FIFO interrupt.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

16.1.3.11 void SPI\_disableTx (  
    **SPI\_Handle** spiHandle )

Disables the serial peripheral interface (SPI) master/slave transmit mode.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

16.1.3.12 void SPI\_disableTxFifo (  
    **SPI\_Handle** spiHandle )

Disables the serial peripheral interface (SPI) transmit FIFO.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

16.1.3.13 void SPI\_disableTxFifoEnh (  
    **SPI\_Handle** spiHandle )

Disables the serial peripheral interface (SPI) transmit FIFO enhancements.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

16.1.3.14 void SPI\_disableTxFifoInt (  
    **SPI\_Handle** spiHandle )

Disables the serial peripheral interface (SPI) transmit FIFO interrupt.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

16.1.3.15 void SPI\_enable (  
    **SPI\_Handle** spiHandle )

Enables the serial peripheral interface (SPI)

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

16.1.3.16 void SPI\_enableChannels (  
    **SPI\_Handle** spiHandle )

Enables the serial peripheral interface (SPI) transmit and receive channels.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

#### 16.1.3.17 void SPI\_enableFifoEnh (

**SPI\_Handle** spiHandle )

Enables the serial peripheral interface (SPI) transmit FIFO enhancements.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

#### 16.1.3.18 void SPI\_enableInt (

**SPI\_Handle** spiHandle )

Enables the serial peripheral interface (SPI) interrupt.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

#### 16.1.3.19 void SPI\_enableLoopBack (

**SPI\_Handle** spiHandle )

Enables the serial peripheral interface (SPI) loop back mode.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

#### 16.1.3.20 void SPI\_enableOverRunInt (

**SPI\_Handle** spiHandle )

Enables the serial peripheral interface (SPI) over-run interrupt.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

#### 16.1.3.21 void SPI\_enableRxFifo (

**SPI\_Handle** spiHandle )

Enables the serial peripheral interface (SPI) receive FIFO.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---



722.7

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

Returns The transmit FIFO status

#### 16.1.3.28 **SPI\_Handle** SPI\_init (

void \* pMemory,

const size\_t numBytes )

Initializes the serial peripheral interface (SPI) object handle.

[1]Parameters in *pMemory* A pointer to the base address of the SPI registers

---

in *numBytes* The number of bytes allocated for the SPI object, bytes

---

Returns The serial peripheral interface (SPI) object handle

#### 16.1.3.29 uint16\_t SPI\_read (

**SPI\_Handle** spiHandle ) [inline]

Reads data from the serial peripheral interface (SPI)

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

Returns The received data value

References \_SPI\_Obj\_::SPIRXBUF.

#### 16.1.3.30 void SPI\_reset (

**SPI\_Handle** spiHandle )

Resets the serial peripheral interface (SPI)

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

#### 16.1.3.31 void SPI\_resetChannels (

**SPI\_Handle** spiHandle )

Resets the serial peripheral interface (SPI) transmit and receive channels.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---





722.7

[1]Parameters *in spiHandle* The serial peripheral interface (SPI) object handle

---

*in clkPhase* The clock phase

---

16.1.3.37 void SPI\_setClkPolarity (

**SPI\_Handle** spiHandle,

const **SPI\_ClkPolarity\_e** polarity )

Sets the serial peripheral interface (SPI) clock polarity.

[1]Parameters *in spiHandle* The serial peripheral interface (SPI) object handle

---

*in polarity* The clock polarity

---

16.1.3.38 void SPI\_setMode (

**SPI\_Handle** spiHandle,

const **SPI\_Mode\_e** mode )

Sets the serial peripheral interface (SPI) network mode.

[1]Parameters *in spiHandle* The serial peripheral interface (SPI) object handle

---

*in mode* The network mode

---

16.1.3.39 void SPI\_setPriority (

**SPI\_Handle** spiHandle,

const **SPI\_Priority\_e** priority )

Sets the priority of the SPI port vis-a-vis the EMU.

[1]Parameters *in spiHandle* The serial peripheral interface (SPI) object handle

---

*in priority* The priority of the SPI port vis-a-vis the EMU

---

16.1.3.40 void SPI\_setRxFifoIntLevel (

**SPI\_Handle** spiHandle,

const **SPI\_FifoLevel\_e** fifoLevel )

Sets the serial peripheral interface (SPI) receive FIFO level for generating an interrupt.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

in *fifoLevel* The FIFO level

---

16.1.3.41 void SPI\_setStelInv (

**SPI\_Handle** spiHandle,

const **SPI\_StelInv\_e** stelinv )

Controls pin inversion of STE pin.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

in *stelinv* Polarity of STE pin

---

16.1.3.42 void SPI\_setTriWire (

**SPI\_Handle** spiHandle,

const **SPI\_TriWire\_e** triwire )

Sets SPI port operating mode.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

in *triwire* 3 or 4 wire mode

---

16.1.3.43 void SPI\_setTxDelay (

**SPI\_Handle** spiHandle,

const uint8\_t delay )

Sets the serial peripheral interface (SPI) transmit delay.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

in *delay* The delay value

---

16.1.3.44 void SPI\_setTxFifoIntLevel (

**SPI\_Handle** spiHandle,

const **SPI\_FifoLevel\_e** fifoLevel )

Sets the serial peripheral interface (SPI) transmit FIFO level for generating an interrupt.

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

in *fifoLevel* The FIFO level

---

16.1.3.45 void SPI\_write (

**SPI\_Handle** spiHandle,

const uint16\_t data ) [inline]

Writes data to the serial peripheral interface (SPI)

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

in *data* The data value

---

References \_SPI\_Obj\_::SPITXBUF.

---

16.1.3.46 void SPI\_write8 (

**SPI\_Handle** spiHandle,

const uint16\_t data ) [inline]

Writes a byte of data to the serial peripheral interface (SPI)

[1]Parameters in *spiHandle* The serial peripheral interface (SPI) object handle

---

in *data* The data value

---

References \_SPI\_Obj\_::SPITXBUF.

---

# 17 Timer

[Introduction](#) ..... [??](#)

[API Functions](#) . 189 The timer API provides a set of functions for configuring, starting, stopping, and reading the CPU timers.

This driver is contained in `f2802x0/common/source/timer.c`, with `f2802x0/common/include/timer.h` containing the API definitions for use by applications.

## 17.1 TIMER

### Data Structures

```
struct \_TIMER\_Obj
```

### Macros

```
#define TIMER0\_BASE\_ADDR
#define TIMER1\_BASE\_ADDR
#define TIMER2\_BASE\_ADDR
#define TIMER\_TCR\_FREESOFT\_BITS
#define TIMER\_TCR\_TIE\_BITS
#define TIMER\_TCR\_TIF\_BITS
#define TIMER\_TCR\_TRB\_BITS
#define TIMER\_TCR\_TSS\_BITS
```

### Typedefs

```
typedef struct \_TIMER\_Obj * TIMER\_Handle
typedef struct \_TIMER\_Obj TIMER\_Obj
```

### Enumerations

```
enum TIMER\_EmulationMode\_e { TIMER\_EmulationMode\_StopAfterNextDecrement,
TIMER\_EmulationMode\_StopAtZero, TIMER\_EmulationMode\_RunFree }

enum TIMER\_Status\_e { TIMER\_Status\_CntIsNotZero, TIMER\_Status\_CntIsZero }
```

## Functions

void [TIMER\\_clearFlag](#) ([TIMER\\_Handle](#) timerHandle)

void [TIMER\\_disableInt](#) ([TIMER\\_Handle](#) timerHandle)

void [TIMER\\_enableInt](#) ([TIMER\\_Handle](#) timerHandle)

uint32\_t [TIMER\\_getCount](#) ([TIMER\\_Handle](#) timerHandle)

[TIMER\\_Status\\_e](#) [TIMER\\_getStatus](#) ([TIMER\\_Handle](#) timerHandle)

[TIMER\\_Handle](#) [TIMER\\_init](#) (void \*pMemory, const size\_t numBytes)

void [TIMER\\_reload](#) ([TIMER\\_Handle](#) timerHandle)

void [TIMER\\_setDecimationFactor](#) ([TIMER\\_Handle](#) timerHandle, const uint16\_t decFactor)

void [TIMER\\_setEmulationMode](#) ([TIMER\\_Handle](#) timerHandle, const [TIMER\\_EmulationMode\\_e](#) mode)

void [TIMER\\_setPeriod](#) ([TIMER\\_Handle](#) timerHandle, const uint32\_t period)

void [TIMER\\_setPreScaler](#) ([TIMER\\_Handle](#) timerHandle, const uint16\_t preScaler)

void [TIMER\\_start](#) ([TIMER\\_Handle](#) timerHandle)

void [TIMER\\_stop](#) ([TIMER\\_Handle](#) timerHandle)

### 17.1.1 Detailed Description

### 17.1.2 Enumeration Type Documentation

#### 17.1.2.1 enum [TIMER\\_EmulationMode\\_e](#)

Enumeration to define the timer (TIMER) emulation mode.

##### Enumerator

**[TIMER\\_EmulationMode\\_StopAfterNextDecrement](#)** Denotes that the timer will stop after the next decrement.

**[TIMER\\_EmulationMode\\_StopAtZero](#)** Denotes that the timer will stop when it reaches zero.

**[TIMER\\_EmulationMode\\_RunFree](#)** Denotes that the timer will run free.

#### 17.1.2.2 enum [TIMER\\_Status\\_e](#)

Enumeration to define the timer (TIMER) status.

##### Enumerator

**[TIMER\\_Status\\_CntIsNotZero](#)** Denotes that the counter is non-zero.

**[TIMER\\_Status\\_CntIsZero](#)** Denotes that the counter is zero.

### 17.1.3 Function Documentation

17.1.3.1 void `TIMER_clearFlag (`  
**`TIMER_Handle`** `timerHandle )`

Clears the timer (TIMER) flag.

[1]Parameters in *timerHandle* The timer (TIMER) object handle

---

17.1.3.2 void `TIMER_disableInt (`  
**`TIMER_Handle`** `timerHandle )`

Disables the timer (TIMER) interrupt.

[1]Parameters in *timerHandle* The timer (TIMER) object handle

---

17.1.3.3 void `TIMER_enableInt (`  
**`TIMER_Handle`** `timerHandle )`

Enables the timer (TIMER) interrupt.

[1]Parameters in *timerHandle* The timer (TIMER) object handle

---

17.1.3.4 uint32\_t `TIMER_getCount (`  
**`TIMER_Handle`** `timerHandle )` `[inline]`

Gets the timer (TIMER) count.

[1]Parameters in *timerHandle* The timer (TIMER) object handle

---

Returns The timer (TIMER) count

References `_TIMER_Obj_::TIM`.

---

17.1.3.5 **`TIMER_Status_e`** `TIMER_getStatus (`  
**`TIMER_Handle`** `timerHandle )`

Gets the timer (TIMER) status.

[1]Parameters in *timerHandle* The timer (TIMER) object handle

---

Returns The timer (TIMER) status

---

**17.1.3.6 TIMER\_Handle** TIMER\_init (

void \* pMemory,  
const size\_t numBytes )

Initializes the timer (TIMER) object handle.

[1]Parameters in *pMemory* A pointer to the base address of the TIMER registers

---

in *numBytes* The number of bytes allocated for the TIMER object, bytes

---

Returns The timer (CLK) object handle

**17.1.3.7 void** TIMER\_reload (

**TIMER\_Handle** timerHandle )

Reloads the timer (TIMER) value.

[1]Parameters in *timerHandle* The timer (TIMER) object handle

---

**17.1.3.8 void** TIMER\_setDecimationFactor (

**TIMER\_Handle** timerHandle,

const uint16\_t decFactor )

Sets the timer (TIMER) decimation factor.

[1]Parameters in *timerHandle* The timer (TIMER) object handle

---

in *decFactor* The timer decimation factor

---

**17.1.3.9 void** TIMER\_setEmulationMode (

**TIMER\_Handle** timerHandle,

const **TIMER\_EmulationMode\_e** mode )

Sets the timer (TIMER) emulation mode.

[1]Parameters in *timerHandle* The timer (TIMER) object handle

---

in *mode* The emulation mode

---



---

#### 17.1.3.10 void TIMER\_setPeriod (

**TIMER\_Handle** timerHandle,

const uint32\_t period )

Sets the timer (TIMER) period.

[1]Parameters in *timerHandle* The timer (TIMER) object handle

---

in *period* The period

---

#### 17.1.3.11 void TIMER\_setPreScaler (

**TIMER\_Handle** timerHandle,

const uint16\_t preScaler )

Sets the timer (TIMER) prescaler.

[1]Parameters in *timerHandle* The timer (TIMER) object handle

---

in *preScaler* The preScaler value

---

#### 17.1.3.12 void TIMER\_start (

**TIMER\_Handle** timerHandle )

Starts the timer (TIMER)

[1]Parameters in *timerHandle* The timer (TIMER) object handle

---

#### 17.1.3.13 void TIMER\_stop (

**TIMER\_Handle** timerHandle )

Stops the timer (TIMER)

[1]Parameters in *timerHandle* The timer (TIMER) object handle

---



# 18 Watchdog Timer

[Introduction](#) ..... [??](#)

[API Functions](#) ..... 195 The Watchdog Timer API provides a set of functions for using the watchdog module.

This driver is contained in `f2802x0/common/source/watchdog.c`, with `f2802x0/common/include/watchdog.h` containing the API definitions for use by applications.

## 18.1 WDOG

### Data Structures

struct [\\_WD OG\\_Obj\\_](#)

### Macros

`#define` [WDOG\\_BASE\\_ADDR](#)

`#define` [WDOG\\_SCSR\\_WDENINT\\_BITS](#)

`#define` [WDOG\\_SCSR\\_WDINTS\\_BITS](#)

`#define` [WDOG\\_SCSR\\_WDOVERRIDE\\_BITS](#)

`#define` [WDOG\\_WDCNTR\\_BITS](#)

`#define` [WDOG\\_WDCR\\_WDCHK\\_BITS](#)

`#define` [WDOG\\_WDCR\\_WDDIS\\_BITS](#)

`#define` [WDOG\\_WDCR\\_WDFLAG\\_BITS](#)

`#define` [WDOG\\_WDCR\\_WDPS\\_BITS](#)

`#define` [WDOG\\_WDCR\\_WRITE\\_ENABLE](#)

`#define` [WDOG\\_WDKEY\\_BITS](#)

### Typedefs

`typedef struct` [\\_WD OG\\_Obj\\_](#) \* [WDOG\\_Handle](#)

`typedef struct` [\\_WD OG\\_Obj\\_](#) [WDOG\\_Obj](#)

## Enumerations

```
enum WDOG_IntStatus_e { WDOG_IntStatus_Active, WDOG_IntStatus_InActive }

enum WDOG_PreScaler_e {
    WDOG_PreScaler_OscClk_by_512_by_1,          WDOG_PreScaler_OscClk_by_512_by_2,
    WDOG_PreScaler_OscClk_by_512_by_4, WDOG_PreScaler_OscClk_by_512_by_8,
    WDOG_PreScaler_OscClk_by_512_by_16,         WDOG_PreScaler_OscClk_by_512_by_32,
    WDOG_PreScaler_OscClk_by_512_by_64 }
```

## Functions

```
void WDOG_clearCounter (WDOG_Handle wdogHandle)

void WDOG_disable (WDOG_Handle wdogHandle)

void WDOG_disableInt (WDOG_Handle wdogHandle)

void WDOG_enable (WDOG_Handle wdogHandle)

void WDOG_enableInt (WDOG_Handle wdogHandle)

void WDOG_enableOverRide (WDOG_Handle wdogHandle)

WDOG_IntStatus_e WDOG_getIntStatus (WDOG_Handle wdogHandle)

WDOG_Handle WDOG_init (void *pMemory, const size_t numBytes)

void WDOG_setCount (WDOG_Handle wdogHandle, const uint8_t count)

void WDOG_setPreScaler (WDOG_Handle wdogHandle, const WDOG_PreScaler_e preScaler)
```

### 18.1.1 Detailed Description

### 18.1.2 Enumeration Type Documentation

#### 18.1.2.1 enum **WDOG\_IntStatus\_e**

Enumeration to define the watchdog (WDOG) interrupt status.

##### Enumerator

**WDOG\_IntStatus\_Active** Denotes an active interrupt status.

**WDOG\_IntStatus\_InActive** Denotes an in-active interrupt status.

#### 18.1.2.2 enum **WDOG\_PreScaler\_e**

Enumeration to define the watchdog (WDOG) timer clock prescaler, which sets the clock frequency.

**Enumerator**

**WDOG\_PreScaler\_OscClk\_by\_512\_by\_1** Denotes WDCLK = OSCCLK/512/1.  
**WDOG\_PreScaler\_OscClk\_by\_512\_by\_2** Denotes WDCLK = OSCCLK/512/2.  
**WDOG\_PreScaler\_OscClk\_by\_512\_by\_4** Denotes WDCLK = OSCCLK/512/4.  
**WDOG\_PreScaler\_OscClk\_by\_512\_by\_8** Denotes WDCLK = OSCCLK/512/8.  
**WDOG\_PreScaler\_OscClk\_by\_512\_by\_16** Denotes WDCLK = OSCCLK/512/16.  
**WDOG\_PreScaler\_OscClk\_by\_512\_by\_32** Denotes WDCLK = OSCCLK/512/32.  
**WDOG\_PreScaler\_OscClk\_by\_512\_by\_64** Denotes WDCLK = OSCCLK/512/64.

### 18.1.3 Function Documentation

#### 18.1.3.1 void WDOG\_clearCounter (

**WDOG\_Handle** wdogHandle )

Clears the watchdog (WDOG) counter.

[1]Parameters in *wdogHandle* The watchdog (WDOG) timer object handle

---

#### 18.1.3.2 void WDOG\_disable (

**WDOG\_Handle** wdogHandle )

Disables the watchdog (WDOG) timer.

[1]Parameters in *wdogHandle* The watchdog (WDOG) timer object handle

---

#### 18.1.3.3 void WDOG\_disableInt (

**WDOG\_Handle** wdogHandle )

Disables the watchdog (WDOG) timer interrupt.

[1]Parameters in *wdogHandle* The watchdog (WDOG) timer object handle

---

#### 18.1.3.4 void WDOG\_enable (

**WDOG\_Handle** wdogHandle )

Enables the watchdog (WDOG) timer.

[1]Parameters in *wdogHandle* The watchdog (WDOG) timer object handle

---

18.1.3.5 void WDOG\_enableInt (  
    **WDOG\_Handle** wdogHandle )

Enables the watchdog (WDOG) timer interrupt.

[1]Parameters in *wdogHandle* The watchdog (WDOG) timer object handle

---

18.1.3.6 void WDOG\_enableOverRide (  
    **WDOG\_Handle** wdogHandle )

Enables the watchdog (WDOG) timer override.

[1]Parameters in *wdogHandle* The watchdog (WDOG) timer object handle

---

18.1.3.7 **WDOG\_IntStatus\_e** WDOG\_getIntStatus (  
    **WDOG\_Handle** wdogHandle )

Gets the watchdog (WDOG) interrupt status.

[1]Parameters in *wdogHandle* The watchdog (WDOG) timer object handle

---

Returns The interrupt status

---

18.1.3.8 **WDOG\_Handle** WDOG\_init (  
    void \* pMemory,  
    const size\_t numBytes )

Initializes the watchdog (WDOG) object handle.

[1]Parameters in *pMemory* A pointer to the base address of the WDOG registers

---

in *numBytes* The number of bytes allocated for the WDOG object, bytes

---

Returns The watchdog (WDOG) object handle

---

18.1.3.9 void WDOG\_setCount (  
    **WDOG\_Handle** wdogHandle,  
    const uint8\_t count )

Sets the watchdog (WDOG) counter.

[1]Parameters *in wdogHandle* The watchdog (WDOG) timer object handle

---

*in count* The count

---

#### 18.1.3.10 void WDOG\_setPreScaler (

**WDOG\_Handle** wdogHandle,

const **WDOG\_PreScaler\_e** preScaler )

Sets the watchdog (WDOG) timer clock prescaler.

[1]Parameters *in wdogHandle* The watchdog (WDOG) timer object handle

---

*in preScaler* The prescaler

---

---

# IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

## Products

Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>
DLP® Products	<a href="http://www.dlp.com">www.dlp.com</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>
Clocks and Timers	<a href="http://www.ti.com/clocks">www.ti.com/clocks</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>
RFID	<a href="http://www.ti-rfid.com">www.ti-rfid.com</a>
RF/IF and ZigBee® Solutions	<a href="http://www.ti.com/lprf">www.ti.com/lprf</a>

## Applications

Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Automotive	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
Broadband	<a href="http://www.ti.com/broadband">www.ti.com/broadband</a>
Digital Control	<a href="http://www.ti.com/digitalcontrol">www.ti.com/digitalcontrol</a>
Medical	<a href="http://www.ti.com/medical">www.ti.com/medical</a>
Military	<a href="http://www.ti.com/military">www.ti.com/military</a>
Optical Networking	<a href="http://www.ti.com/opticalnetwork">www.ti.com/opticalnetwork</a>
Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
Telephony	<a href="http://www.ti.com/telephony">www.ti.com/telephony</a>
Video & Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>
Wireless	<a href="http://www.ti.com/wireless">www.ti.com/wireless</a>

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2023, Texas Instruments Incorporated