

# **F2802x Peripheral Driver Library**

## **USER'S GUIDE**



---

# Copyright

Copyright © 2012 Texas Instruments Incorporated. All rights reserved. ControlSUITE is a registered trademark of Texas Instruments. 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  
12203 Southwest Freeway  
Houston, TX 77477  
<http://www.ti.com/c2000>



## Revision Information

This is version 200 of this document, last updated on Tue Jul 24 10:01:42 CDT 2012.

# 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>
2.1 Introduction	7
2.2 Direct Register Access Model	7
2.3 Software Driver Model	8
2.4 Combining The Models	8
<b>3 Analog to Digital Converter (ADC)</b>	<b>9</b>
3.1 Introduction	9
3.2 ADC	9
<b>4 Capture (CAP)</b>	<b>33</b>
4.1 Introduction	33
4.2 CAP	33
<b>5 Device Clocking (CLK)</b>	<b>49</b>
5.1 Introduction	49
5.2 CLK	49
<b>6 Comparater (COMP)</b>	<b>73</b>
6.1 Introduction	73
6.2 COMP	73
<b>7 Central Processing Unit (CPU)</b>	<b>81</b>
7.1 Introduction	81
7.2 CPU	81
<b>8 Flash</b>	<b>103</b>
8.1 Introduction	103
8.2 FLASH	103
<b>9 General Purpose Input/Output (GPIO)</b>	<b>115</b>
9.1 Introduction	115
9.2 GPIO	115
<b>10 Oscillator (OSC)</b>	<b>127</b>
10.1 Introduction	127
10.2 OSC	127
<b>11 Peripheral Interrupt Expansion Module (PIE)</b>	<b>135</b>
11.1 Introduction	135
11.2 PIE	135
<b>12 Phase Locked Loop (PLL)</b>	<b>167</b>
12.1 Introduction	167
12.2 PLL	167
<b>13 Pulse Width Modulator (PWM)</b>	<b>177</b>
13.1 Introduction	177
13.2 PWM	177
<b>14 Power Control (PWR)</b>	<b>229</b>
14.1 Introduction	229
14.2 PWR	229

<b>15</b>	<b>Serial Communications Interface (SCI)</b>	<b>235</b>
15.1	Introduction	235
15.2	SCI	235
<b>16</b>	<b>Serial Peripheral Interface (SPI)</b>	<b>261</b>
16.1	Introduction	261
16.2	SPI	261
<b>17</b>	<b>Timer</b>	<b>279</b>
17.1	Introduction	279
17.2	TIMER	279
<b>18</b>	<b>Watchdog Timer</b>	<b>287</b>
18.1	Introduction	287
18.2	WDOG	287
	<b>IMPORTANT NOTICE</b>	<b>294</b>

# 1 Introduction

The Texas Instruments® ControlSUITE® 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.

`f2802x0_common/src/` This directory contains the source code for the drivers.

`f2802x0_common/inc/` 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.

f2802x0\_common/project 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 .....	7
Direct Register Access Model .....	7
Software Driver Model .....	8
Combining The Models .....	8

### 2.1 Introduction

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.

### 2.2 Direct Register Access Model

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 peripherals corresponding header file contained in `f2802x0_header/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 structure; 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 version 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` version 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;
```

## 2.3 Software Driver Model

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 peripherals 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_Number_0, GPIO_PullUp_Enable);  
GPIO_setMode(myGpio, GPIO_Number_0, GPIO_0_Mode_EPWM1A);
```

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/project`. 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.

## 2.4 Combining The Models

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 `/controlsuite/device_support/f2802x0/version/` 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 .....	9
API Functions .....	9

### 3.1 Introduction

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.

### 3.2 ADC

#### Data Structures

- [\\_ADC\\_Obj](#)

#### Defines

- [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` resultNumber)
- 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` pulseMode)
- 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` socNumber, 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)

## 3.2.1 Data Structure Documentation

### 3.2.1.1 `_ADC_Obj_`

**Definition:**

```
typedef struct
{
    uint16_t ADCRESULT[16];
    uint16_t resvd_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 resvd_12[13];
uint16_t ADCREV;
}
__ADC_Obj__

```

**Members:**

**ADCRESULT** ADC result registers.

**resvd\_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.

**resvd\_12** Reserved.

**ADCREV** ADC Revision Register.

**Description:**

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

## 3.2.2 Define Documentation

### 3.2.2.1 ADC\_ADCCTL1\_ADCBGPWD\_BITS

**Definition:**

```
#define ADC_ADCCTL1_ADCBGPWD_BITS
```

**Description:**

Defines the location of the ADCBGPWD bits in the ADCTL1 register.

### 3.2.2.2 ADC\_ADCCTL1\_ADCBSY\_BITS

**Definition:**

```
#define ADC_ADCCTL1_ADCBSY_BITS
```

**Description:**

Defines the location of the ADCBSY bits in the ADCTL1 register.

### 3.2.2.3 ADC\_ADCCTL1\_ADCBSYCHAN\_BITS

**Definition:**

```
#define ADC_ADCCTL1_ADCBSYCHAN_BITS
```

**Description:**

Defines the location of the ADCBSYCHAN bits in the ADCTL1 register.

### 3.2.2.4 ADC\_ADCCTL1\_ADCENABLE\_BITS

**Definition:**

```
#define ADC_ADCCTL1_ADCENABLE_BITS
```

**Description:**

Defines the location of the ADCENABLE bits in the ADCTL1 register.

### 3.2.2.5 ADC\_ADCCTL1\_ADCPWDN\_BITS

**Definition:**

```
#define ADC_ADCCTL1_ADCPWDN_BITS
```

**Description:**

Defines the location of the ADCPWDN bits in the ADCTL1 register.

### 3.2.2.6 ADC\_ADCCTL1\_ADCREFPWD\_BITS

**Definition:**

```
#define ADC_ADCCTL1_ADCREFPWD_BITS
```

**Description:**

Defines the location of the ADCREFPWD bits in the ADCTL1 register.

### 3.2.2.7 ADC\_ADCCTL1\_ADCREFSEL\_BITS

**Definition:**

```
#define ADC_ADCCTL1_ADCREFSEL_BITS
```

**Description:**

Defines the location of the ADCREFSEL bits in the ADCTL1 register.

### 3.2.2.8 ADC\_ADCCTL1\_INTPULSEPOS\_BITS

**Definition:**

```
#define ADC_ADCCTL1_INTPULSEPOS_BITS
```

**Description:**

Defines the location of the INTPULSEPOS bits in the ADCTL1 register.

### 3.2.2.9 ADC\_ADCCTL1\_RESET\_BITS

**Definition:**

```
#define ADC_ADCCTL1_RESET_BITS
```

**Description:**

Defines the location of the RESET bits in the ADCTL1 register.

### 3.2.2.10 ADC\_ADCCTL1\_TEMPCONV\_BITS

**Definition:**

```
#define ADC_ADCCTL1_TEMPCONV_BITS
```

**Description:**

Defines the location of the TEMPCONV bits in the ADCTL1 register.

### 3.2.2.11 ADC\_ADCCTL1\_VREFLOCONV\_BITS

**Definition:**

```
#define ADC_ADCCTL1_VREFLOCONV_BITS
```

**Description:**

Defines the location of the VREFLOCONV bits in the ADCTL1 register.

### 3.2.2.12 ADC\_ADCSAMPLEMODE\_SEPARATE\_FLAG

**Definition:**

```
#define ADC_ADCSAMPLEMODE_SEPARATE_FLAG
```

**Description:**

Define for the channel separate flag.

### 3.2.2.13 ADC\_ADCSAMPLEMODE\_SIMULEN0\_BITS

**Definition:**

```
#define ADC_ADCSAMPLEMODE_SIMULEN0_BITS
```

**Description:**

Defines the location of the SIMULEN0 bits in the ADCSAMPLEMODE register.

### 3.2.2.14 ADC\_ADCSAMPLEMODE\_SIMULEN10\_BITS

**Definition:**

```
#define ADC_ADCSAMPLEMODE_SIMULEN10_BITS
```

**Description:**

Defines the location of the SIMULEN10 bits in the ADCSAMPLEMODE register.

### 3.2.2.15 ADC\_ADCSAMPLEMODE\_SIMULEN12\_BITS

**Definition:**

```
#define ADC_ADCSAMPLEMODE_SIMULEN12_BITS
```

**Description:**

Defines the location of the SIMULEN12 bits in the ADCSAMPLEMODE register.

### 3.2.2.16 ADC\_ADCSAMPLEMODE\_SIMULEN14\_BITS

**Definition:**

```
#define ADC_ADCSAMPLEMODE_SIMULEN14_BITS
```

**Description:**

Defines the location of the SIMULEN14 bits in the ADCSAMPLEMODE register.

### 3.2.2.17 ADC\_ADCSAMPLEMODE\_SIMULEN2\_BITS

**Definition:**

```
#define ADC_ADCSAMPLEMODE_SIMULEN2_BITS
```

**Description:**

Defines the location of the SIMULEN2 bits in the ADCSAMPLEMODE register.

### 3.2.2.18 ADC\_ADCSAMPLEMODE\_SIMULEN4\_BITS

**Definition:**

```
#define ADC_ADCSAMPLEMODE_SIMULEN4_BITS
```

**Description:**

Defines the location of the SIMULEN4 bits in the ADCSAMPLEMODE register.

### 3.2.2.19 ADC\_ADCSAMPLEMODE\_SIMULEN6\_BITS

**Definition:**

```
#define ADC_ADCSAMPLEMODE_SIMULEN6_BITS
```

**Description:**

Defines the location of the SIMULEN6 bits in the ADCSAMPLEMODE register.

### 3.2.2.20 ADC\_ADCSAMPLEMODE\_SIMULEN8\_BITS

**Definition:**

```
#define ADC_ADCSAMPLEMODE_SIMULEN8_BITS
```

**Description:**

Defines the location of the SIMULEN8 bits in the ADCSAMPLEMODE register.

### 3.2.2.21 ADC\_ADCSOCxCTL\_ACQPS\_BITS

**Definition:**

```
#define ADC_ADCSOCxCTL_ACQPS_BITS
```

**Description:**

Defines the location of the ACQPS bits in the ADCSOCxCTL register.

### 3.2.2.22 ADC\_ADCSOCxCTL\_CHSEL\_BITS

**Definition:**

```
#define ADC_ADCSOCxCTL_CHSEL_BITS
```

**Description:**

Defines the location of the CHSEL bits in the ADCSOCxCTL register.

### 3.2.2.23 ADC\_ADCSOCxCTL\_TRIGSEL\_BITS

**Definition:**

```
#define ADC_ADCSOCxCTL_TRIGSEL_BITS
```

**Description:**

Defines the location of the TRIGSEL bits in the ADCSOCxCTL register.



### 3.2.2.24 ADC\_BASE\_ADDR

**Definition:**

```
#define ADC_BASE_ADDR
```

**Description:**

Defines the base address of the analog-to-digital converter (ADC) registers.

### 3.2.2.25 ADC\_dataBias

**Definition:**

```
#define ADC_dataBias
```

**Description:**

Integer value bias corresponding to voltage 1.65V bias of input data on 3.3V, 12 bit ADC.

### 3.2.2.26 ADC\_DELAY\_usec

**Definition:**

```
#define ADC_DELAY_usec
```

**Description:**

Defines the ADC delay for part of the ADC initialization.

### 3.2.2.27 ADC\_INTSELxNy\_INTCONT\_BITS

**Definition:**

```
#define ADC_INTSELxNy_INTCONT_BITS
```

**Description:**

Defines the location of the INTCONT bits in the INTSELxNy register.

### 3.2.2.28 ADC\_INTSELxNy\_INTE\_BITS

**Definition:**

```
#define ADC_INTSELxNy_INTE_BITS
```

**Description:**

Defines the location of the INTE bits in the INTSELxNy register.

### 3.2.2.29 ADC\_INTSELxNy\_INTSEL\_BITS

**Definition:**

```
#define ADC_INTSELxNy_INTSEL_BITS
```

**Description:**

Defines the location of the INTSEL bits in the INTSELxNy register.

### 3.2.2.30 ADC\_INTSELxNy\_LOG2\_NUMBITS\_PER\_REG

**Definition:**

```
#define ADC_INTSELxNy_LOG2_NUMBITS_PER_REG
```

**Description:**

Defines the log2() of the number of bits per INTSELxNy register.

### 3.2.2.31 ADC\_INTSELxNy\_NUMBITS\_PER\_REG

**Definition:**

```
#define ADC_INTSELxNy_NUMBITS_PER_REG
```

**Description:**

Defines the number of bits per INTSELxNy register.

## 3.2.3 Typedef Documentation

### 3.2.3.1 ADC\_Handle

**Definition:**

```
typedef struct ADC_Obj *ADC_Handle
```

**Description:**

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

### 3.2.3.2 ADC\_Obj

**Definition:**

```
typedef struct _ADC_Obj_ ADC_Obj
```

**Description:**

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

## 3.2.4 Enumeration Documentation

### 3.2.4.1 ADC\_IntMode\_e

**Description:**

Enumeration to define the analog-to-digital converter (ADC) interrupt mode.

**Enumerators:**

**ADC\_IntMode\_ClearFlag** Denotes that a new interrupt will not be generated until the interrupt flag is cleared.

**ADC\_IntMode\_EOC** Denotes that a new interrupt will be generated on the next end of conversion (EOC).

### 3.2.4.2 ADC\_IntNumber\_e

**Description:**

Enumeration to define the analog-to-digital converter (ADC) interrupt number.

**Enumerators:**

- ADC\_IntNumber\_1** Denotes ADCINT1.
- ADC\_IntNumber\_2** Denotes ADCINT2.
- ADC\_IntNumber\_3** Denotes ADCINT3.
- ADC\_IntNumber\_4** Denotes ADCINT4.
- ADC\_IntNumber\_5** Denotes ADCINT5.
- ADC\_IntNumber\_6** Denotes ADCINT6.
- ADC\_IntNumber\_7** Denotes ADCINT7.
- ADC\_IntNumber\_8** Denotes ADCINT8.
- ADC\_IntNumber\_9** Denotes ADCINT9.

### 3.2.4.3 ADC\_IntPulseGenMode\_e

**Description:**

Enumeration to define the analog-to-digital converter (ADC) interrupt pulse generation mode.

**Enumerators:**

- ADC\_IntPulseGenMode\_During** Denotes that interrupt pulse generation occurs when the ADC begins conversion.
- ADC\_IntPulseGenMode\_Prior** Denotes that interrupt pulse generation occurs 1 cycle prior to the ADC result latching.

### 3.2.4.4 ADC\_IntSrc\_e

**Description:**

Enumeration to define the analog-to-digital converter (ADC) interrupt source.

**Enumerators:**

- ADC\_IntSrc\_EOC0** Denotes that interrupt source is the end of conversion for SOC0.
- ADC\_IntSrc\_EOC1** Denotes that interrupt source is the end of conversion for SOC1.
- ADC\_IntSrc\_EOC2** Denotes that interrupt source is the end of conversion for SOC2.
- ADC\_IntSrc\_EOC3** Denotes that interrupt source is the end of conversion for SOC3.
- ADC\_IntSrc\_EOC4** Denotes that interrupt source is the end of conversion for SOC4.
- ADC\_IntSrc\_EOC5** Denotes that interrupt source is the end of conversion for SOC5.
- ADC\_IntSrc\_EOC6** Denotes that interrupt source is the end of conversion for SOC6.
- ADC\_IntSrc\_EOC7** Denotes that interrupt source is the end of conversion for SOC7.
- ADC\_IntSrc\_EOC8** Denotes that interrupt source is the end of conversion for SOC8.
- ADC\_IntSrc\_EOC9** Denotes that interrupt source is the end of conversion for SOC9.
- ADC\_IntSrc\_EOC10** Denotes that interrupt source is the end of conversion for SOC10.
- ADC\_IntSrc\_EOC11** Denotes that interrupt source is the end of conversion for SOC11.
- ADC\_IntSrc\_EOC12** Denotes that interrupt source is the end of conversion for SOC12.
- ADC\_IntSrc\_EOC13** Denotes that interrupt source is the end of conversion for SOC13.

**ADC\_IntSrc\_EOC14** Denotes that interrupt source is the end of conversion for SOC14.

**ADC\_IntSrc\_EOC15** Denotes that interrupt source is the end of conversion for SOC15.

#### 3.2.4.5 ADC\_ResultNumber\_e

**Description:**

Enumeration to define the analog-to-digital converter (ADC) result number.

**Enumerators:**

**ADC\_ResultNumber\_0** Denotes ADCRESULT0.

**ADC\_ResultNumber\_1** Denotes ADCRESULT1.

**ADC\_ResultNumber\_2** Denotes ADCRESULT2.

**ADC\_ResultNumber\_3** Denotes ADCRESULT3.

**ADC\_ResultNumber\_4** Denotes ADCRESULT4.

**ADC\_ResultNumber\_5** Denotes ADCRESULT5.

**ADC\_ResultNumber\_6** Denotes ADCRESULT6.

**ADC\_ResultNumber\_7** Denotes ADCRESULT7.

**ADC\_ResultNumber\_8** Denotes ADCRESULT8.

**ADC\_ResultNumber\_9** Denotes ADCRESULT9.

**ADC\_ResultNumber\_10** Denotes ADCRESULT10.

**ADC\_ResultNumber\_11** Denotes ADCRESULT11.

**ADC\_ResultNumber\_12** Denotes ADCRESULT12.

**ADC\_ResultNumber\_13** Denotes ADCRESULT13.

**ADC\_ResultNumber\_14** Denotes ADCRESULT14.

**ADC\_ResultNumber\_15** Denotes ADCRESULT15.

#### 3.2.4.6 ADC\_SampleMode\_e

**Description:**

Enumeration to define the analog-to-digital converter (ADC) sample modes.

**Enumerators:**

**ADC\_SampleMode\_SOC0\_and\_SOC1\_Separate** Denotes SOC0 and SOC1 are sampled separately.

**ADC\_SampleMode\_SOC2\_and\_SOC3\_Separate** Denotes SOC2 and SOC3 are sampled separately.

**ADC\_SampleMode\_SOC4\_and\_SOC5\_Separate** Denotes SOC4 and SOC5 are sampled separately.

**ADC\_SampleMode\_SOC6\_and\_SOC7\_Separate** Denotes SOC6 and SOC7 are sampled separately.

**ADC\_SampleMode\_SOC8\_and\_SOC9\_Separate** Denotes SOC8 and SOC9 are sampled separately.

**ADC\_SampleMode\_SOC10\_and\_SOC11\_Separate** Denotes SOC10 and SOC11 are sampled separately.

**ADC\_SampleMode\_SOC12\_and\_SOC13\_Separate** Denotes SOC12 and SOC13 are sampled separately.

- ADC\_SampleMode\_SOC14\_and\_SOC15\_Separate** Denotes SOC14 and SOC15 are sampled separately.
- ADC\_SampleMode\_SOC0\_and\_SOC1\_Together** Denotes SOC0 and SOC1 are sampled together.
- ADC\_SampleMode\_SOC2\_and\_SOC3\_Together** Denotes SOC2 and SOC3 are sampled together.
- ADC\_SampleMode\_SOC4\_and\_SOC5\_Together** Denotes SOC4 and SOC5 are sampled together.
- ADC\_SampleMode\_SOC6\_and\_SOC7\_Together** Denotes SOC6 and SOC7 are sampled together.
- ADC\_SampleMode\_SOC8\_and\_SOC9\_Together** Denotes SOC8 and SOC9 are sampled together.
- ADC\_SampleMode\_SOC10\_and\_SOC11\_Together** Denotes SOC10 and SOC11 are sampled together.
- ADC\_SampleMode\_SOC12\_and\_SOC13\_Together** Denotes SOC12 and SOC13 are sampled together.
- ADC\_SampleMode\_SOC14\_and\_SOC15\_Together** Denotes SOC14 and SOC15 are sampled together.

### 3.2.4.7 ADC\_SocChanNumber\_e

#### Description:

Enumeration to define the start of conversion (SOC) channel numbers.

#### Enumerators:

- ADC\_SocChanNumber\_A0** Denotes SOC channel number A0.
- ADC\_SocChanNumber\_A1** Denotes SOC channel number A1.
- ADC\_SocChanNumber\_A2** Denotes SOC channel number A2.
- ADC\_SocChanNumber\_A3** Denotes SOC channel number A3.
- ADC\_SocChanNumber\_A4** Denotes SOC channel number A4.
- ADC\_SocChanNumber\_A5** Denotes SOC channel number A5.
- ADC\_SocChanNumber\_A6** Denotes SOC channel number A6.
- ADC\_SocChanNumber\_A7** Denotes SOC channel number A7.
- ADC\_SocChanNumber\_B0** Denotes SOC channel number B0.
- ADC\_SocChanNumber\_B1** Denotes SOC channel number B1.
- ADC\_SocChanNumber\_B2** Denotes SOC channel number B2.
- ADC\_SocChanNumber\_B3** Denotes SOC channel number B3.
- ADC\_SocChanNumber\_B4** Denotes SOC channel number B4.
- ADC\_SocChanNumber\_B5** Denotes SOC channel number B5.
- ADC\_SocChanNumber\_B6** Denotes SOC channel number B6.
- ADC\_SocChanNumber\_B7** Denotes SOC channel number B7.
- ADC\_SocChanNumber\_A0\_and\_B0\_Together** Denotes SOC channel number A0 and B0 together.
- ADC\_SocChanNumber\_A1\_and\_B1\_Together** Denotes SOC channel number A0 and B0 together.
- ADC\_SocChanNumber\_A2\_and\_B2\_Together** Denotes SOC channel number A0 and B0 together.

- ADC\_SocChanNumber\_A3\_and\_B3\_Together*** Denotes SOC channel number A0 and B0 together.
- ADC\_SocChanNumber\_A4\_and\_B4\_Together*** Denotes SOC channel number A0 and B0 together.
- ADC\_SocChanNumber\_A5\_and\_B5\_Together*** Denotes SOC channel number A0 and B0 together.
- ADC\_SocChanNumber\_A6\_and\_B6\_Together*** Denotes SOC channel number A0 and B0 together.
- ADC\_SocChanNumber\_A7\_and\_B7\_Together*** Denotes SOC channel number A0 and B0 together.

#### 3.2.4.8 ADC\_SocNumber\_e

**Description:**

Enumeration to define the start of conversion (SOC) numbers.

**Enumerators:**

- ADC\_SocNumber\_0*** Denotes SOC0.
- ADC\_SocNumber\_1*** Denotes SOC1.
- ADC\_SocNumber\_2*** Denotes SOC2.
- ADC\_SocNumber\_3*** Denotes SOC3.
- ADC\_SocNumber\_4*** Denotes SOC4.
- ADC\_SocNumber\_5*** Denotes SOC5.
- ADC\_SocNumber\_6*** Denotes SOC6.
- ADC\_SocNumber\_7*** Denotes SOC7.
- ADC\_SocNumber\_8*** Denotes SOC8.
- ADC\_SocNumber\_9*** Denotes SOC9.
- ADC\_SocNumber\_10*** Denotes SOC10.
- ADC\_SocNumber\_11*** Denotes SOC11.
- ADC\_SocNumber\_12*** Denotes SOC12.
- ADC\_SocNumber\_13*** Denotes SOC13.
- ADC\_SocNumber\_14*** Denotes SOC14.
- ADC\_SocNumber\_15*** Denotes SOC15.

#### 3.2.4.9 ADC\_SocSampleWindow\_e

**Description:**

Enumeration to define the start of conversion (SOC) sample delays.

**Enumerators:**

- ADC\_SocSampleWindow\_7\_cycles*** Denotes an SOC sample window of 7 cycles.
- ADC\_SocSampleWindow\_8\_cycles*** Denotes an SOC sample window of 8 cycles.
- ADC\_SocSampleWindow\_9\_cycles*** Denotes an SOC sample window of 9 cycles.
- ADC\_SocSampleWindow\_10\_cycles*** Denotes an SOC sample window of 10 cycles.
- ADC\_SocSampleWindow\_11\_cycles*** Denotes an SOC sample window of 11 cycles.
- ADC\_SocSampleWindow\_12\_cycles*** Denotes an SOC sample window of 12 cycles.

$$= \frac{1}{2} \left( \frac{1}{2} + \frac{1}{2} \right) = \frac{1}{2}$$

**ADC\_SocSampleWindow\_59\_cycles** Denotes an SOC sample window of 59 cycles.  
**ADC\_SocSampleWindow\_60\_cycles** Denotes an SOC sample window of 60 cycles.  
**ADC\_SocSampleWindow\_61\_cycles** Denotes an SOC sample window of 61 cycles.  
**ADC\_SocSampleWindow\_62\_cycles** Denotes an SOC sample window of 62 cycles.  
**ADC\_SocSampleWindow\_63\_cycles** Denotes an SOC sample window of 63 cycles.  
**ADC\_SocSampleWindow\_64\_cycles** Denotes an SOC sample window of 64 cycles.

### 3.2.4.10 ADC\_SocTrigSrc\_e

**Description:**

Enumeration to define the start of conversion (SOC) trigger source.

**Enumerators:**

**ADC\_SocTrigSrc\_Sw** Denotes a software trigger source for the SOC flag.  
**ADC\_SocTrigSrc\_CpuTimer\_0** Denotes a CPUTIMER0 trigger source for the SOC flag.  
**ADC\_SocTrigSrc\_CpuTimer\_1** Denotes a CPUTIMER1 trigger source for the SOC flag.  
**ADC\_SocTrigSrc\_CpuTimer\_2** Denotes a CPUTIMER2 trigger source for the SOC flag.  
**ADC\_SocTrigSrc\_XINT2\_XINT2SOC** Denotes a XINT2, XINT2SOC trigger source for the SOC flag.  
**ADC\_SocTrigSrc\_EPWM1\_ADCSOCA** Denotes a EPWM1, ADCSOCA trigger source for the SOC flag.  
**ADC\_SocTrigSrc\_EPWM1\_ADCSOCB** Denotes a EPWM1, ADCSOCB trigger source for the SOC flag.  
**ADC\_SocTrigSrc\_EPWM2\_ADCSOCA** Denotes a EPWM2, ADCSOCA trigger source for the SOC flag.  
**ADC\_SocTrigSrc\_EPWM2\_ADCSOCB** Denotes a EPWM2, ADCSOCB trigger source for the SOC flag.  
**ADC\_SocTrigSrc\_EPWM3\_ADCSOCA** Denotes a EPWM3, ADCSOCA trigger source for the SOC flag.  
**ADC\_SocTrigSrc\_EPWM3\_ADCSOCB** Denotes a EPWM3, ADCSOCB trigger source for the SOC flag.  
**ADC\_SocTrigSrc\_EPWM4\_ADCSOCA** Denotes a EPWM4, ADCSOCA trigger source for the SOC flag.  
**ADC\_SocTrigSrc\_EPWM4\_ADCSOCB** Denotes a EPWM4, ADCSOCB trigger source for the SOC flag.  
**ADC\_SocTrigSrc\_EPWM5\_ADCSOCA** Denotes a EPWM5, ADCSOCA trigger source for the SOC flag.  
**ADC\_SocTrigSrc\_EPWM5\_ADCSOCB** Denotes a EPWM5, ADCSOCB trigger source for the SOC flag.  
**ADC\_SocTrigSrc\_EPWM6\_ADCSOCA** Denotes a EPWM6, ADCSOCA trigger source for the SOC flag.  
**ADC\_SocTrigSrc\_EPWM6\_ADCSOCB** Denotes a EPWM7, ADCSOCB trigger source for the SOC flag.  
**ADC\_SocTrigSrc\_EPWM7\_ADCSOCA** Denotes a EPWM7, ADCSOCA trigger source for the SOC flag.  
**ADC\_SocTrigSrc\_EPWM7\_ADCSOCB** Denotes a EPWM7, ADCSOCB trigger source for the SOC flag.



### 3.2.4.11 ADC\_VoltageRefSrc\_e

**Description:**

Enumeration to define the voltage reference source.

**Enumerators:**

**ADC\_VoltageRefSrc\_Int** Denotes an internal voltage reference source.

**ADC\_VoltageRefSrc\_Ext** Denotes an external voltage reference source.

## 3.2.5 Function Documentation

### 3.2.5.1 ADC\_clearIntFlag

Clears the analog-to-digital converter (ADC) interrupt flag.

**Prototype:**

```
void  
ADC_clearIntFlag(ADC_Handle adcHandle,  
                 const ADC_IntNumber_e intNumber) [inline]
```

**Parameters:**

← **adcHandle** The analog-to-digital converter (ADC) object handle

← **intNumber** The ADC interrupt number

### 3.2.5.2 void ADC\_disable (ADC\_Handle adcHandle)

Disables the analog-to-digital converter (ADC).

**Parameters:**

← **adcHandle** The analog-to-digital converter (ADC) object handle

### 3.2.5.3 void ADC\_disableBandGap (ADC\_Handle adcHandle)

Disables the analog-to-digital converter (ADC) band gap circuit.

**Parameters:**

← **adcHandle** The analog-to-digital converter (ADC) object handle

### 3.2.5.4 void ADC\_disableInt (ADC\_Handle adcHandle, const ADC\_IntNumber\_e intNumber)

Disables the analog-to-digital converter (ADC) interrupt.

**Parameters:**

← **adcHandle** The analog-to-digital converter (ADC) object handle

← **intNumber** The interrupt number

3.2.5.5 void ADC\_disableRefBuffers ([ADC\\_Handle](#) *adcHandle*)

Disables the analog-to-digital converter (ADC) reference buffers circuit.

**Parameters:**

← ***adcHandle*** The analog-to-digital converter (ADC) object handle

3.2.5.6 void ADC\_disableTempSensor ([ADC\\_Handle](#) *adcHandle*)

Disables temperature sensor for conversion on A5.

**Parameters:**

← ***adcHandle*** The analog-to-digital converter (ADC) object handle

3.2.5.7 void ADC\_enable ([ADC\\_Handle](#) *adcHandle*)

Enables the analog-to-digital converter (ADC).

**Parameters:**

← ***adcHandle*** The analog-to-digital converter (ADC) object handle

3.2.5.8 void ADC\_enableBandGap ([ADC\\_Handle](#) *adcHandle*)

Enables the analog-to-digital converter (ADC) band gap circuit.

**Parameters:**

← ***adcHandle*** The analog-to-digital converter (ADC) object handle

3.2.5.9 void ADC\_enableInt ([ADC\\_Handle](#) *adcHandle*, const [ADC\\_IntNumber\\_e](#) *intNumber*)

Enables the analog-to-digital converter (ADC) interrupt.

**Parameters:**

← ***adcHandle*** The analog-to-digital converter (ADC) object handle

← ***intNumber*** The interrupt number

3.2.5.10 void ADC\_enableRefBuffers ([ADC\\_Handle](#) *adcHandle*)

Enables the analog-to-digital converter (ADC) reference buffers circuit.

**Parameters:**

← ***adcHandle*** The analog-to-digital converter (ADC) object handle

**3.2.5.11** void ADC\_enableTempSensor ([ADC\\_Handle](#) *adcHandle*)

Enables temperature sensor for conversion on A5.

**Parameters:**

← ***adcHandle*** The analog-to-digital converter (ADC) object handle

**3.2.5.12** void ADC\_forceConversion ([ADC\\_Handle](#) *adcHandle*, const [ADC\\_SocNumber\\_e](#) *socNumber*) [inline]

Reads the specified ADC result (i.e. value).

**Parameters:**

← ***adcHandle*** The ADC handle

← ***resultNumber*** The result number for the ADCRESULT registers

**Returns:**

The ADC result

**3.2.5.13** bool\_t ADC\_getIntStatus ([ADC\\_Handle](#) *adcHandle*, const [ADC\\_IntNumber\\_e](#) *intNumber*) [inline]

Reads the status for a given ADC interrupt.

**Parameters:**

← ***adcHandle*** The analog-to-digital converter (ADC) object handle

← ***intNumber*** The ADC interrupt number

**Returns:**

Interrupt status for selected ADC interrupt

**3.2.5.14** [ADC\\_SocSampleWindow\\_e](#) ADC\_getSocSampleWindow ([ADC\\_Handle](#) *adcHandle*, const [ADC\\_SocNumber\\_e](#) *socNumber*) [inline]

Gets the analog-to-digital converter (ADC) start-of-conversion (SOC) sample delay value.

**Parameters:**

← ***adcHandle*** The analog-to-digital converter (ADC) object handle

← ***socNumber*** The SOC number

**Returns:**

The ADC sample delay value

3.2.5.15 `int16_t ADC_getTemperatureC (ADC_Handle adcHandle, int16_t sensorSample)`  
[inline]

Converts a temperature sensor sample into a temperature in Celcius.

**Parameters:**

← **adcHandle** The analog-to-digital converter (ADC) object handle

**Returns:**

Temperature in degrees Celcius

3.2.5.16 `int16_t ADC_getTemperatureK (ADC_Handle adcHandle, int16_t sensorSample)`  
[inline]

Converts a temperature sensor sample into a temperature in Kelvin.

**Parameters:**

← **adcHandle** The analog-to-digital converter (ADC) object handle

**Returns:**

Temperature in degrees Kelvin

3.2.5.17 `ADC_Handle ADC_init (void * pMemory, const size_t numBytes)`

Initializes the analog-to-digital converter (ADC) object handle.

**Parameters:**

← **pMemory** A pointer to the base address of the ADC registers

← **numBytes** The number of bytes allocated for the ADC object, bytes

**Returns:**

The analog-to-digital converter (ADC) object handle

3.2.5.18 `void ADC_powerDown (ADC_Handle adcHandle)`

Powers down the analog-to-digital converter (ADC).

**Parameters:**

← **adcHandle** The analog-to-digital converter (ADC) object handle

3.2.5.19 `void ADC_powerUp (ADC_Handle adcHandle)`

Powers up the analog-to-digital converter (ADC).

**Parameters:**

← **adcHandle** The analog-to-digital converter (ADC) object handle

3.2.5.20 `uint_least16_t ADC_readResult (ADC_Handle adcHandle, const ADC_ResultNumber_e resultNumber) [inline]`

Reads the specified ADC result (i.e. value).

**Parameters:**

- ← **adcHandle** The ADC handle
- ← **resultNumber** The result number for the ADCRESULT registers

**Returns:**

The ADC result

3.2.5.21 `void ADC_reset (ADC_Handle adcHandle)`

Resets the analog-to-digital converter (ADC).

**Parameters:**

- ← **adcHandle** The analog-to-digital converter (ADC) object handle

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

Sets the interrupt mode.

**Parameters:**

- ← **adcHandle** The analog-to-digital converter (ADC) object handle
- ← **intNumber** The interrupt number
- ← **intMode** The interrupt mode

3.2.5.23 `void ADC_setIntPulseGenMode (ADC_Handle adcHandle, const ADC_IntPulseGenMode_e pulseMode)`

Sets the interrupt pulse generation mode.

**Parameters:**

- ← **adcHandle** The analog-to-digital converter (ADC) object handle
- ← **pulseMode** The pulse generation mode

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

Sets the interrupt source.

**Parameters:**

- ← **adcHandle** The analog-to-digital converter (ADC) object handle

- ← **intNumber** The interrupt number
- ← **intSrc** The interrupt source

3.2.5.25 void ADC\_setSampleMode (ADC\_Handle adcHandle, const ADC\_SampleMode\_e sampleMode)

Sets the sample mode.

**Parameters:**

- ← **adcHandle** The analog-to-digital converter (ADC) object handle
- ← **sampleMode** The sample mode

3.2.5.26 void ADC\_setSocChanNumber (ADC\_Handle adcHandle, const ADC\_SocNumber\_e socNumber, const ADC\_SocChanNumber\_e chanNumber)

Sets the start-of-conversion (SOC) channel number.

**Parameters:**

- ← **adcHandle** The analog-to-digital converter (ADC) object handle
- ← **socNumber** The SOC number
- ← **chanNumber** The channel number

3.2.5.27 void ADC\_setSocSampleWindow (ADC\_Handle adcHandle, const ADC\_SocNumber\_e socNumber, const ADC\_SocSampleWindow\_e sampleWindow)

Sets the start-of-conversion (SOC) sample delay.

**Parameters:**

- ← **adcHandle** The analog-to-digital converter (ADC) object handle
- ← **socNumber** The SOC number
- ← **sampleDelay** The sample delay

3.2.5.28 void ADC\_setSocTrigSrc (ADC\_Handle adcHandle, const ADC\_SocNumber\_e socNumber, const ADC\_SocTrigSrc\_e trigSrc)

Sets the start-of-conversion (SOC) trigger source.

**Parameters:**

- ← **adcHandle** The analog-to-digital converter (ADC) object handle
- ← **socNumber** The SOC number
- ← **trigSrc** The trigger delay

3.2.5.29 void ADC\_setVoltRefSrc ([ADC\\_Handle](#) *adcHandle*, const [ADC\\_VoltageRefSrc\\_e](#) *voltRef*)

Sets the voltage reference source.

**Parameters:**

- ← ***adcHandle*** The analog-to-digital converter (ADC) object handle
- ← ***voltRef*** The voltage reference source





## 4 Capture (CAP)

Introduction .....	33
API Functions .....	33

### 4.1 Introduction

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.

### 4.2 CAP

#### Data Structures

- [\\_CAP\\_Obj](#)

#### Defines

- [CAP\\_ECCTL1\\_CAP1POL\\_BITS](#)
- [CAP\\_ECCTL1\\_CAP2POL\\_BITS](#)
- [CAP\\_ECCTL1\\_CAP3POL\\_BITS](#)
- [CAP\\_ECCTL1\\_CAP4POL\\_BITS](#)
- [CAP\\_ECCTL1\\_CAPLDEN\\_BITS](#)
- [CAP\\_ECCTL1\\_CTRRST1\\_BITS](#)
- [CAP\\_ECCTL1\\_CTRRST2\\_BITS](#)
- [CAP\\_ECCTL1\\_CTRRST3\\_BITS](#)
- [CAP\\_ECCTL1\\_CTRRST4\\_BITS](#)
- [CAP\\_ECCTL1\\_FREESOFT\\_BITS](#)
- [CAP\\_ECCTL1\\_PRESCALE\\_BITS](#)
- [CAP\\_ECCTL2\\_APWMPOL\\_BITS](#)
- [CAP\\_ECCTL2\\_CAPAPWM\\_BITS](#)
- [CAP\\_ECCTL2\\_CONTONESHOT\\_BITS](#)
- [CAP\\_ECCTL2\\_REARM\\_BITS](#)
- [CAP\\_ECCTL2\\_STOP\\_WRAP\\_BITS](#)
- [CAP\\_ECCTL2\\_SWSYNC\\_BITS](#)
- [CAP\\_ECCTL2\\_SYNCIEN\\_BITS](#)
- [CAP\\_ECCTL2\\_SYNCOSEL\\_BITS](#)
- [CAP\\_ECCTL2\\_TSCTRSTOP\\_BITS](#)
- [CAP\\_ECCxxx\\_C EVT1\\_BITS](#)

- CAP\_ECCxxx\_CEV2\_BITS
- CAP\_ECCxxx\_CEV3\_BITS
- CAP\_ECCxxx\_CEV4\_BITS
- CAP\_ECCxxx\_CTRCOMP\_BITS
- CAP\_ECCxxx\_CTROVF\_BITS
- CAP\_ECCxxx\_CTRPRD\_BITS
- CAP\_ECCxxx\_INT\_BITS
- CAPA\_BASE\_ADDR

## Enumerations

- CAP\_Event\_e
- CAP\_Int\_Type\_e
- CAP\_Polarity\_e
- CAP\_Prescale\_e
- CAP\_Reset\_e
- CAP\_RunMode\_e
- CAP\_SyncOut\_e

## 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\_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)

## 4.2.1 Data Structure Documentation

### 4.2.1.1 `_CAP_Obj_`

**Definition:**

```
typedef struct
{
    uint32_t TSCTR;
    uint32_t CTRPHS;
    uint32_t CAP1;
    uint32_t CAP2;
    uint32_t CAP3;
    uint32_t CAP4;
    uint16_t Rsvd_1[8];
    uint16_t ECCTL1;
    uint16_t ECCTL2;
    uint16_t ECEINT;
    uint16_t ECEFLG;
    uint16_t ECECLR;
    uint16_t ECEFRC;
}
_CAP_Obj_
```

**Members:**

**TSCTR** Time-stamp Counter.  
**CTRPHS** Counter Phase Offset Value Register.  
**CAP1** Capture 1 Register.  
**CAP2** Capture 2 Register.  
**CAP3** Capture 3 Register.  
**CAP4** Capture 4 Register.  
**Rsvd\_1** Reserved.  
**ECCTL1** Capture Control Register 1.  
**ECCTL2** Capture Control Register 2.  
**ECEINT** Capture Interrupt Enable Register.  
**ECEFLG** Capture Interrupt Flag Register.  
**ECECLR** Capture Interrupt Clear Register.  
**ECEFRC** Capture Interrupt Force Register.

**Description:**

Defines the capture (CAP) object.

## 4.2.2 Define Documentation

### 4.2.2.1 CAP\_ECCTL1\_CAP1POL\_BITS

**Definition:**

```
#define CAP_ECCTL1_CAP1POL_BITS
```

**Description:**

Defines the location of the CAP1POL bits in the ECCTL1 register.

### 4.2.2.2 CAP\_ECCTL1\_CAP2POL\_BITS

**Definition:**

```
#define CAP_ECCTL1_CAP2POL_BITS
```

**Description:**

Defines the location of the CAP2POL bits in the ECCTL1 register.

### 4.2.2.3 CAP\_ECCTL1\_CAP3POL\_BITS

**Definition:**

```
#define CAP_ECCTL1_CAP3POL_BITS
```

**Description:**

Defines the location of the CAP3POL bits in the ECCTL1 register.

### 4.2.2.4 CAP\_ECCTL1\_CAP4POL\_BITS

**Definition:**

```
#define CAP_ECCTL1_CAP4POL_BITS
```

**Description:**

Defines the location of the CAP4POL bits in the ECCTL1 register.

### 4.2.2.5 CAP\_ECCTL1\_CAPLDEN\_BITS

**Definition:**

```
#define CAP_ECCTL1_CAPLDEN_BITS
```

**Description:**

Defines the location of the CAPLDEN bits in the ECCTL1 register.

#### 4.2.2.6 CAP\_ECCTL1\_CTRRST1\_BITS

**Definition:**

```
#define CAP_ECCTL1_CTRRST1_BITS
```

**Description:**

Defines the location of the CTRRST1 bits in the ECCTL1 register.

#### 4.2.2.7 CAP\_ECCTL1\_CTRRST2\_BITS

**Definition:**

```
#define CAP_ECCTL1_CTRRST2_BITS
```

**Description:**

Defines the location of the CTRRST2 bits in the ECCTL1 register.

#### 4.2.2.8 CAP\_ECCTL1\_CTRRST3\_BITS

**Definition:**

```
#define CAP_ECCTL1_CTRRST3_BITS
```

**Description:**

Defines the location of the CTRRST3 bits in the ECCTL1 register.

#### 4.2.2.9 CAP\_ECCTL1\_CTRRST4\_BITS

**Definition:**

```
#define CAP_ECCTL1_CTRRST4_BITS
```

**Description:**

Defines the location of the CTRRST4 bits in the ECCTL1 register.

#### 4.2.2.10 CAP\_ECCTL1\_FREESOFT\_BITS

**Definition:**

```
#define CAP_ECCTL1_FREESOFT_BITS
```

**Description:**

Defines the location of the FREE/SOFT bits in the ECCTL1 register.

#### 4.2.2.11 CAP\_ECCTL1\_PRESCALE\_BITS

**Definition:**

```
#define CAP_ECCTL1_PRESCALE_BITS
```

**Description:**

Defines the location of the PRESCALE bits in the ECCTL1 register.

#### 4.2.2.12 CAP\_ECCTL2\_APWMPOL\_BITS

**Definition:**

```
#define CAP_ECCTL2_APWMPOL_BITS
```

**Description:**

Defines the location of the APWMPOL bits in the ECCTL2 register.

#### 4.2.2.13 CAP\_ECCTL2\_CAPAPWM\_BITS

**Definition:**

```
#define CAP_ECCTL2_CAPAPWM_BITS
```

**Description:**

Defines the location of the CAP/APWM bits in the ECCTL2 register.

#### 4.2.2.14 CAP\_ECCTL2\_CONTONESHOT\_BITS

**Definition:**

```
#define CAP_ECCTL2_CONTONESHOT_BITS
```

**Description:**

Defines the location of the CONT/ONESHOT bits in the ECCTL2 register.

#### 4.2.2.15 CAP\_ECCTL2\_REARM\_BITS

**Definition:**

```
#define CAP_ECCTL2_REARM_BITS
```

**Description:**

Defines the location of the REARM bits in the ECCTL2 register.

#### 4.2.2.16 CAP\_ECCTL2\_STOP\_WRAP\_BITS

**Definition:**

```
#define CAP_ECCTL2_STOP_WRAP_BITS
```

**Description:**

Defines the location of the STOP\_WRAP bits in the ECCTL2 register.

#### 4.2.2.17 CAP\_ECCTL2\_SWSYNC\_BITS

**Definition:**

```
#define CAP_ECCTL2_SWSYNC_BITS
```

**Description:**

Defines the location of the SWSYNC bits in the ECCTL2 register.

#### 4.2.2.18 CAP\_ECCTL2\_SYNCIEN\_BITS

**Definition:**

```
#define CAP_ECCTL2_SYNCIEN_BITS
```

**Description:**

Defines the location of the SYNCI\_EN bits in the ECCTL2 register.

#### 4.2.2.19 CAP\_ECCTL2\_SYNCOSEL\_BITS

**Definition:**

```
#define CAP_ECCTL2_SYNCOSEL_BITS
```

**Description:**

Defines the location of the SYNCO\_SEL bits in the ECCTL2 register.

#### 4.2.2.20 CAP\_ECCTL2\_TSCTRSTOP\_BITS

**Definition:**

```
#define CAP_ECCTL2_TSCTRSTOP_BITS
```

**Description:**

Defines the location of the TSCTRSTOP bits in the ECCTL2 register.

#### 4.2.2.21 CAP\_ECCxxx\_C EVT1\_BITS

**Definition:**

```
#define CAP_ECCxxx_C EVT1_BITS
```

**Description:**

Defines the location of the CEVT4 bits in the ECCxxx register.

#### 4.2.2.22 CAP\_ECCxxx\_C EVT2\_BITS

**Definition:**

```
#define CAP_ECCxxx_C EVT2_BITS
```

**Description:**

Defines the location of the CEVT4 bits in the ECCxxx register.

#### 4.2.2.23 CAP\_ECCxxx\_C EVT3\_BITS

**Definition:**

```
#define CAP_ECCxxx_C EVT3_BITS
```

**Description:**

Defines the location of the CEVT4 bits in the ECCxxx register.

#### 4.2.2.24 CAP\_ECCxxx\_C EVT4\_BITS

**Definition:**

```
#define CAP_ECCxxx_C EVT4_BITS
```

**Description:**

Defines the location of the CEVT4 bits in the ECCxxx register.

#### 4.2.2.25 CAP\_ECCxxx\_C TRCOMP\_BITS

**Definition:**

```
#define CAP_ECCxxx_C TRCOMP_BITS
```

**Description:**

Defines the location of the CTR=COMP bits in the ECCxxx register.

#### 4.2.2.26 CAP\_ECCxxx\_C TROVF\_BITS

**Definition:**

```
#define CAP_ECCxxx_C TROVF_BITS
```

**Description:**

Defines the location of the CTROVF bits in the ECCxxx register.

#### 4.2.2.27 CAP\_ECCxxx\_C TRPRD\_BITS

**Definition:**

```
#define CAP_ECCxxx_C TRPRD_BITS
```

**Description:**

Defines the location of the CTR=PRD bits in the ECCxxx register.

#### 4.2.2.28 CAP\_ECCxxx\_C INT\_BITS

**Definition:**

```
#define CAP_ECCxxx_C INT_BITS
```

**Description:**

Defines the location of the INT bits in the ECCxxx register.

#### 4.2.2.29 CAPA\_BASE\_ADDR

**Definition:**

```
#define CAPA_BASE_ADDR
```

**Description:**

Defines the base address of the capture (CAP) A registers.



## 4.2.3 Typedef Documentation

### 4.2.3.1 CAP\_Handle

**Definition:**

```
typedef struct CAP_Obj *CAP_Handle
```

**Description:**

Defines the capture (CAP) handle.

### 4.2.3.2 CAP\_Obj

**Definition:**

```
typedef struct _CAP_Obj_ CAP_Obj
```

**Description:**

Defines the capture (CAP) object.

## 4.2.4 Enumeration Documentation

### 4.2.4.1 CAP\_Event\_e

**Description:**

Enumeration to define the capture (CAP) events.

**Enumerators:**

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

### 4.2.4.2 CAP\_Int\_Type\_e

**Description:**

Enumeration to define the capture (CAP) interrupts.

**Enumerators:**

- 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.

#### 4.2.4.3 CAP\_Polarity\_e

**Description:**

Enumeration to define the capture (CAP) event polarities.

**Enumerators:**

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

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

#### 4.2.4.4 CAP\_Prescale\_e

**Description:**

Enumeration to define the capture (CAP) prescaler values.

**Enumerators:**

- 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.

#### 4.2.4.5 CAP\_Reset\_e

**Description:**

Enumeration to define the capture (CAP) event resets.

**Enumerators:**

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

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

#### 4.2.4.6 CAP\_RunMode\_e

**Description:**

Enumeration to define the pulse width modulation (PWM) run modes.

#### 4.2.4.7 enum CAP\_Stop\_Wrap\_e

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

**Enumerators:**

**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.

#### 4.2.4.8 CAP\_SyncOut\_e

**Description:**

Enumeration to define the Sync Out options.

**Enumerators:**

**CAP\_SyncOut\_SyncIn** Sync In used for Sync Out.

**CAP\_SyncOut\_CTRPRD** CTR = PRD used for Sync Out.

**CAP\_SyncOut\_Disable** Disables Sync Out.

### 4.2.5 Function Documentation

#### 4.2.5.1 CAP\_clearInt

Clears capture (CAP) interrupt flag.

**Prototype:**

```
void  
CAP_clearInt(CAP_Handle capHandle,  
             const CAP_Int_Type_e intType) [inline]
```

**Parameters:**

← **capHandle** The capture (CAP) object handle

← **intType** The capture interrupt to be cleared

#### 4.2.5.2 void CAP\_disableCaptureLoad (CAP\_Handle capHandle)

Disables loading of CAP1-4 on capture event.

**Parameters:**

← **capHandle** The capture (CAP) object handle

#### 4.2.5.3 void CAP\_disableInt (CAP\_Handle capHandle, const CAP\_Int\_Type\_e intType)

Disables capture (CAP) interrupt source.

**Parameters:**

← **capHandle** The capture (CAP) object handle

← **intType** The capture interrupt type to be disabled

#### 4.2.5.4 void CAP\_disableSyncln (CAP\_Handle capHandle)

Disables counter synchronization.

**Parameters:**

← **capHandle** The capture (CAP) object handle

#### 4.2.5.5 void CAP\_disableTimestampCounter (CAP\_Handle capHandle)

Disables Time Stamp counter from running.

**Parameters:**

← **capHandle** The capture (CAP) object handle

#### 4.2.5.6 void CAP\_enableCaptureLoad (CAP\_Handle capHandle)

Enables loading of CAP1-4 on capture event.

**Parameters:**

← **capHandle** The capture (CAP) object handle

#### 4.2.5.7 void CAP\_enableInt (CAP\_Handle capHandle, const CAP\_Int\_Type\_e intType)

Enables capture (CAP) interrupt source.

**Parameters:**

← **capHandle** The capture (CAP) object handle

← **intType** The capture interrupt type to be enabled

#### 4.2.5.8 void CAP\_enableSyncIn (CAP\_Handle capHandle)

Enables counter synchronization.

**Parameters:**

← **capHandle** The capture (CAP) object handle

#### 4.2.5.9 void CAP\_enableTimestampCounter (CAP\_Handle capHandle)

Enables Time Stamp counter to running.

**Parameters:**

← **capHandle** The capture (CAP) object handle

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

Gets the CAP1 register value.

**Parameters:**

← **capHandle** The capture (CAP) object handle

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

Gets the CAP2 register value.

**Parameters:**

← **capHandle** The capture (CAP) object handle

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

Gets the CAP3 register value.

**Parameters:**

← **capHandle** The capture (CAP) object handle

#### 4.2.5.13 uint32\_t CAP\_getCap4 (CAP\_Handle capHandle) [inline]

Gets the CAP4 register value.

**Parameters:**

← **capHandle** The capture (CAP) object handle

#### 4.2.5.14 CAP\_Handle CAP\_init (void \* *pMemory*, const size\_t *numBytes*)

Initializes the capture (CAP) object handle.

**Parameters:**

- ← ***pMemory*** A pointer to the base address of the CAP registers
- ← ***numBytes*** The number of bytes allocated for the CAP object, bytes

**Returns:**

The capture (CAP) object handle

#### 4.2.5.15 void CAP\_rearm (CAP\_Handle *capHandle*) [inline]

(Re-)Arm the capture module

**Parameters:**

- ← ***capHandle*** The capture (CAP) object handle

#### 4.2.5.16 void CAP\_setApwmCompare (CAP\_Handle *capHandle*, const uint32\_t *compare*) [inline]

Sets the APWM compare value.

**Parameters:**

- ← ***capHandle*** The capture (CAP) object handle
- ← ***compare*** The APWM compare value

#### 4.2.5.17 void CAP\_setApwmPeriod (CAP\_Handle *capHandle*, const uint32\_t *period*) [inline]

Sets the APWM period.

**Parameters:**

- ← ***capHandle*** The capture (CAP) object handle
- ← ***period*** The APWM period

#### 4.2.5.18 void CAP\_setCapContinuous (CAP\_Handle *capHandle*)

Sets up for continuous Capture.

**Parameters:**

- ← ***capHandle*** The capture (CAP) object handle

4.2.5.19 void CAP\_setCapEvtPolarity ([CAP\\_Handle](#) capHandle, const [CAP\\_Event\\_e](#) event, const [CAP\\_Polarity\\_e](#) polarity)

Sets the capture event polarity.

**Parameters:**

- ← **capHandle** The capture (CAP) object handle
- ← **event** The event to configure
- ← **polarity** The polarity to configure the event for

4.2.5.20 void CAP\_setCapEvtReset ([CAP\\_Handle](#) capHandle, const [CAP\\_Event\\_e](#) event, const [CAP\\_Reset\\_e](#) reset)

Sets the capture event counter reset configuration.

**Parameters:**

- ← **capHandle** The capture (CAP) object handle
- ← **event** The event to configure
- ← **reset** Whether the event should reset the counter or not

4.2.5.21 void CAP\_setCapOneShot ([CAP\\_Handle](#) capHandle)

Sets up for one-shot Capture.

**Parameters:**

- ← **capHandle** The capture (CAP) object handle

4.2.5.22 void CAP\_setModeApwm ([CAP\\_Handle](#) capHandle)

Sets capture peripheral up for APWM mode.

**Parameters:**

- ← **capHandle** The capture (CAP) object handle

4.2.5.23 void CAP\_setModeCap ([CAP\\_Handle](#) capHandle)

Sets capture peripheral up for capture mode.

**Parameters:**

- ← **capHandle** The capture (CAP) object handle

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

Set the stop/wrap mode.

**Parameters:**

- ← **capHandle** The capture (CAP) object handle
- ← **stopWrap** The stop/wrap mode to set

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

Set the sync out mode.

**Parameters:**

- ← **capHandle** The capture (CAP) object handle
- ← **syncOut** The sync out mode to set



## 5 Device Clocking (CLK)

Introduction .....	49
API Functions .....	49

### 5.1 Introduction

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 `f280x0_common/source/clk.c`, with `<tf280x0_common/include/clk.h` containing the API definitions for use by applications.

### 5.2 CLK

#### Data Structures

- [\\_CLK\\_Obj\\_](#)

#### Defines

- [CLK\\_BASE\\_ADDR](#)
- [CLK\\_CLKCTL\\_INTOSC1HALTI\\_BITS](#)
- [CLK\\_CLKCTL\\_INTOSC1OFF\\_BITS](#)
- [CLK\\_CLKCTL\\_INTOSC2HALTI\\_BITS](#)
- [CLK\\_CLKCTL\\_INTOSC2OFF\\_BITS](#)
- [CLK\\_CLKCTL\\_NMIRESETSEL\\_BITS](#)
- [CLK\\_CLKCTL\\_OSCCLKSRC2SEL\\_BITS](#)
- [CLK\\_CLKCTL\\_OSCCLKSRCSEL\\_BITS](#)
- [CLK\\_CLKCTL\\_TMR2CLKPRESCALE\\_BITS](#)
- [CLK\\_CLKCTL\\_TMR2CLKSRCSEL\\_BITS](#)
- [CLK\\_CLKCTL\\_WDCLKSRCSEL\\_BITS](#)
- [CLK\\_CLKCTL\\_WDHALTI\\_BITS](#)
- [CLK\\_CLKCTL\\_XCLKINOFF\\_BITS](#)
- [CLK\\_CLKCTL\\_XTALOSCOFF\\_BITS](#)
- [CLK\\_LOSPCP\\_LSPCLK\\_BITS](#)
- [CLK\\_PCLKCR0\\_ADCENCLK\\_BITS](#)
- [CLK\\_PCLKCR0\\_ECANAENCLK\\_BITS](#)
- [CLK\\_PCLKCR0\\_HRPWMENCLK\\_BITS](#)
- [CLK\\_PCLKCR0\\_I2CAENCLK\\_BITS](#)
- [CLK\\_PCLKCR0\\_LINAENCLK\\_BITS](#)
- [CLK\\_PCLKCR0\\_SCIAENCLK\\_BITS](#)

- [CLK\\_PCLKCR0\\_SPIAENCLK\\_BITS](#)
- [CLK\\_PCLKCR0\\_SPIBENCLK\\_BITS](#)
- [CLK\\_PCLKCR0\\_TBCLKSYNC\\_BITS](#)
- [CLK\\_PCLKCR1\\_ECAP1ENCLK\\_BITS](#)
- [CLK\\_PCLKCR1\\_EPWM1ENCLK\\_BITS](#)
- [CLK\\_PCLKCR1\\_EPWM2ENCLK\\_BITS](#)
- [CLK\\_PCLKCR1\\_EPWM3ENCLK\\_BITS](#)
- [CLK\\_PCLKCR1\\_EPWM4ENCLK\\_BITS](#)
- [CLK\\_PCLKCR1\\_EPWM5ENCLK\\_BITS](#)
- [CLK\\_PCLKCR1\\_EPWM6ENCLK\\_BITS](#)
- [CLK\\_PCLKCR1\\_EPWM7ENCLK\\_BITS](#)
- [CLK\\_PCLKCR1\\_EQEP1ENCLK\\_BITS](#)
- [CLK\\_PCLKCR3\\_CLA1ENCLK\\_BITS](#)
- [CLK\\_PCLKCR3\\_COMP1ENCLK\\_BITS](#)
- [CLK\\_PCLKCR3\\_COMP2ENCLK\\_BITS](#)
- [CLK\\_PCLKCR3\\_COMP3ENCLK\\_BITS](#)
- [CLK\\_PCLKCR3\\_CPUTIMER0ENCLK\\_BITS](#)
- [CLK\\_PCLKCR3\\_CPUTIMER1ENCLK\\_BITS](#)
- [CLK\\_PCLKCR3\\_CPUTIMER2ENCLK\\_BITS](#)
- [CLK\\_PCLKCR3\\_GPIOINENCLK\\_BITS](#)
- [CLK\\_XCLK\\_XCLKINSEL\\_BITS](#)
- [CLK\\_XCLK\\_XCLKOUTDIV\\_BITS](#)

## Enumerations

- [CLK\\_ClkInSrc\\_e](#)
- [CLK\\_CompNumber\\_e](#)
- [CLK\\_CpuTimerNumber\\_e](#)
- [CLK\\_LowSpdPreScaler\\_e](#)
- [CLK\\_Osc2Src\\_e](#)
- [CLK\\_OscSrc\\_e](#)
- [CLK\\_Timer2PreScaler\\_e](#)
- [CLK\\_Timer2Src\\_e](#)
- [CLK\\_WdClkSrc\\_e](#)

## Functions

- void [CLK\\_disableAdcClock](#) ([CLK\\_Handle](#) clkHandle)
- void [CLK\\_disableClaClock](#) ([CLK\\_Handle](#) clkHandle)
- void [CLK\\_disableClkIn](#) ([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\_enableClaClock (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)

## 5.2.1 Data Structure Documentation

### 5.2.1.1 \_CLK\_Obj\_

**Definition:**

```
typedef struct
{
    uint16_t XCLK;
    uint16_t rsvd_1;
    uint16_t CLKCTL;
    uint16_t rsvd_2[8];
    uint16_t LOSPCP;
    uint16_t PCLKCR0;
    uint16_t PCLKCR1;
    uint16_t rsvd_3[2];
    uint16_t PCLKCR3;
}
_CLK_Obj_
```

**Members:**

**XCLK** XCLKOUT/XCLKIN Control.

**rsvd\_1** Reserved.

**CLKCTL** Clock Control Register.

**rsvd\_2** Reserved.

**LOSPCP** Low-Speed Peripheral Clock Pre-Scaler Register.

**PCLKCR0** Peripheral Clock Control Register 0.

**PCLKCR1** Peripheral Clock Control Register 1.

**rsvd\_3** Reserved.

**PCLKCR3** Peripheral Clock Control Register 3.

**Description:**

Defines the clock (CLK) object.

## 5.2.2 Define Documentation

### 5.2.2.1 CLK\_BASE\_ADDR

**Definition:**

```
#define CLK_BASE_ADDR
```

**Description:**

Defines the base address of the clock (CLK) registers.

### 5.2.2.2 CLK\_CLKCTL\_INTOSC1HALTI\_BITS

**Definition:**

```
#define CLK_CLKCTL_INTOSC1HALTI_BITS
```

**Description:**

Defines the location of the INTOSC1HALTI bits in the CLKCTL register.

### 5.2.2.3 CLK\_CLKCTL\_INTOSC1OFF\_BITS

**Definition:**

```
#define CLK_CLKCTL_INTOSC1OFF_BITS
```

**Description:**

Defines the location of the INTOSC1OFF bits in the CLKCTL register.

### 5.2.2.4 CLK\_CLKCTL\_INTOSC2HALTI\_BITS

**Definition:**

```
#define CLK_CLKCTL_INTOSC2HALTI_BITS
```

**Description:**

Defines the location of the INTOSC2HALTI bits in the CLKCTL register.

### 5.2.2.5 CLK\_CLKCTL\_INTOSC2OFF\_BITS

**Definition:**

```
#define CLK_CLKCTL_INTOSC2OFF_BITS
```

**Description:**

Defines the location of the INTOSC2OFF bits in the CLKCTL register.

### 5.2.2.6 CLK\_CLKCTL\_NMIRESETSEL\_BITS

**Definition:**

```
#define CLK_CLKCTL_NMIRESETSEL_BITS
```

**Description:**

Defines the location of the NMIRESETSEL bits in the CLKCTL register.

### 5.2.2.7 CLK\_CLKCTL\_OSCCLKSRC2SEL\_BITS

**Definition:**

```
#define CLK_CLKCTL_OSCCLKSRC2SEL_BITS
```

**Description:**

Defines the location of the OSCCLKSRC2SEL bits in the CLKCTL register.

#### 5.2.2.8 CLK\_CLKCTL\_OSCCLKSRCSEL\_BITS

**Definition:**

```
#define CLK_CLKCTL_OSCCLKSRCSEL_BITS
```

**Description:**

Defines the location of the OSCCLKSRCSEL bits in the CLKCTL register.

#### 5.2.2.9 CLK\_CLKCTL\_TMR2CLKPRESCALE\_BITS

**Definition:**

```
#define CLK_CLKCTL_TMR2CLKPRESCALE_BITS
```

**Description:**

Defines the location of the TMR2CLKPRESCALE bits in the CLKCTL register.

#### 5.2.2.10 CLK\_CLKCTL\_TMR2CLKSRCSEL\_BITS

**Definition:**

```
#define CLK_CLKCTL_TMR2CLKSRCSEL_BITS
```

**Description:**

Defines the location of the TMR2CLKSRCSEL bits in the CLKCTL register.

#### 5.2.2.11 CLK\_CLKCTL\_WDCLKSRCSEL\_BITS

**Definition:**

```
#define CLK_CLKCTL_WDCLKSRCSEL_BITS
```

**Description:**

Defines the location of the WDCLKSRCSEL bits in the CLKCTL register.

#### 5.2.2.12 CLK\_CLKCTL\_WDHALTI\_BITS

**Definition:**

```
#define CLK_CLKCTL_WDHALTI_BITS
```

**Description:**

Defines the location of the WDHALTI bits in the CLKCTL register.

#### 5.2.2.13 CLK\_CLKCTL\_XCLKINOFF\_BITS

**Definition:**

```
#define CLK_CLKCTL_XCLKINOFF_BITS
```

**Description:**

Defines the location of the XCLKINOFF bits in the CLKCTL register.

#### 5.2.2.14 CLK\_CLKCTL\_XTALOSCOFF\_BITS

**Definition:**

```
#define CLK_CLKCTL_XTALOSCOFF_BITS
```

**Description:**

Defines the location of the XTALOSCOFF bits in the CLKCTL register.

#### 5.2.2.15 CLK\_LOSPCP\_LSPCLK\_BITS

**Definition:**

```
#define CLK_LOSPCP_LSPCLK_BITS
```

**Description:**

Defines the location of the LSPCLK bits in the LOSPCP register.

#### 5.2.2.16 CLK\_PCLKCR0\_ADCENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR0_ADCENCLK_BITS
```

**Description:**

Defines the location of the ADCENCLK bits in the PCLKCR0 register.

#### 5.2.2.17 CLK\_PCLKCR0\_ECANAENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR0_ECANAENCLK_BITS
```

**Description:**

Defines the location of the ECANAENCLK bits in the PCLKCR0 register.

#### 5.2.2.18 CLK\_PCLKCR0\_HRPWMENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR0_HRPWMENCLK_BITS
```

**Description:**

Defines the location of the HRPWMENCLK bits in the PCLKCR0 register.

#### 5.2.2.19 CLK\_PCLKCR0\_I2CAENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR0_I2CAENCLK_BITS
```

**Description:**

Defines the location of the I2CAENCLK bits in the PCLKCR0 register.

#### 5.2.2.20 CLK\_PCLKCR0\_LINAENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR0_LINAENCLK_BITS
```

**Description:**

Defines the location of the LINAENCLK bits in the PCLKCR0 register.

#### 5.2.2.21 CLK\_PCLKCR0\_SCIAENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR0_SCIAENCLK_BITS
```

**Description:**

Defines the location of the SCIAENCLK bits in the PCLKCR0 register.

#### 5.2.2.22 CLK\_PCLKCR0\_SPIAENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR0_SPIAENCLK_BITS
```

**Description:**

Defines the location of the SPIAENCLK bits in the PCLKCR0 register.

#### 5.2.2.23 CLK\_PCLKCR0\_SPIBENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR0_SPIBENCLK_BITS
```

**Description:**

Defines the location of the SPIBENCLK bits in the PCLKCR0 register.

#### 5.2.2.24 CLK\_PCLKCR0\_TBCLKSYNC\_BITS

**Definition:**

```
#define CLK_PCLKCR0_TBCLKSYNC_BITS
```

**Description:**

Defines the location of the TBCLKSYNC bits in the PCLKCR0 register.

#### 5.2.2.25 CLK\_PCLKCR1\_ECAP1ENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR1_ECAP1ENCLK_BITS
```

**Description:**

Defines the location of the ECAP1ENCLK bits in the PCLKCR1 register.



#### 5.2.2.26 CLK\_PCLKCR1\_EPWM1ENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR1_EPWM1ENCLK_BITS
```

**Description:**

Defines the location of the EPWM1ENCLK bits in the PCLKCR1 register.

#### 5.2.2.27 CLK\_PCLKCR1\_EPWM2ENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR1_EPWM2ENCLK_BITS
```

**Description:**

Defines the location of the EPWM2ENCLK bits in the PCLKCR1 register.

#### 5.2.2.28 CLK\_PCLKCR1\_EPWM3ENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR1_EPWM3ENCLK_BITS
```

**Description:**

Defines the location of the EPWM3ENCLK bits in the PCLKCR1 register.

#### 5.2.2.29 CLK\_PCLKCR1\_EPWM4ENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR1_EPWM4ENCLK_BITS
```

**Description:**

Defines the location of the EPWM4ENCLK bits in the PCLKCR1 register.

#### 5.2.2.30 CLK\_PCLKCR1\_EPWM5ENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR1_EPWM5ENCLK_BITS
```

**Description:**

Defines the location of the EPWM5ENCLK bits in the PCLKCR1 register.

#### 5.2.2.31 CLK\_PCLKCR1\_EPWM6ENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR1_EPWM6ENCLK_BITS
```

**Description:**

Defines the location of the EPWM6ENCLK bits in the PCLKCR1 register.

#### 5.2.2.32 CLK\_PCLKCR1\_EPWM7ENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR1_EPWM7ENCLK_BITS
```

**Description:**

Defines the location of the EPWM7ENCLK bits in the PCLKCR1 register.

#### 5.2.2.33 CLK\_PCLKCR1\_EQEP1ENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR1_EQEP1ENCLK_BITS
```

**Description:**

Defines the location of the EQEP1ENCLK bits in the PCLKCR1 register.

#### 5.2.2.34 CLK\_PCLKCR3\_CLA1ENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR3_CLA1ENCLK_BITS
```

**Description:**

Defines the location of the CLA1ENCLK bits in the PCLKCR3 register.

#### 5.2.2.35 CLK\_PCLKCR3\_COMP1ENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR3_COMP1ENCLK_BITS
```

**Description:**

Defines the location of the COMP1ENCLK bits in the PCLKCR3 register.

#### 5.2.2.36 CLK\_PCLKCR3\_COMP2ENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR3_COMP2ENCLK_BITS
```

**Description:**

Defines the location of the COMP2ENCLK bits in the PCLKCR3 register.

#### 5.2.2.37 CLK\_PCLKCR3\_COMP3ENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR3_COMP3ENCLK_BITS
```

**Description:**

Defines the location of the COMP3ENCLK bits in the PCLKCR3 register.

### 5.2.2.38 CLK\_PCLKCR3\_CPUTIMER0ENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR3_CPUTIMER0ENCLK_BITS
```

**Description:**

Defines the location of the CPUTIMER0ENCLK bits in the PCLKCR3 register.

### 5.2.2.39 CLK\_PCLKCR3\_CPUTIMER1ENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR3_CPUTIMER1ENCLK_BITS
```

**Description:**

Defines the location of the CPUTIMER1ENCLK bits in the PCLKCR3 register.

### 5.2.2.40 CLK\_PCLKCR3\_CPUTIMER2ENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR3_CPUTIMER2ENCLK_BITS
```

**Description:**

Defines the location of the CPUTIMER2ENCLK bits in the PCLKCR3 register.

### 5.2.2.41 CLK\_PCLKCR3\_GPIOINENCLK\_BITS

**Definition:**

```
#define CLK_PCLKCR3_GPIOINENCLK_BITS
```

**Description:**

Defines the location of the GPIOINENCLK bits in the PCLKCR3 register.

### 5.2.2.42 CLK\_XCLK\_XCLKINSEL\_BITS

**Definition:**

```
#define CLK_XCLK_XCLKINSEL_BITS
```

**Description:**

Defines the location of the XCLKINSEL bits in the XCLK register.

### 5.2.2.43 CLK\_XCLK\_XCLKOUTDIV\_BITS

**Definition:**

```
#define CLK_XCLK_XCLKOUTDIV_BITS
```

**Description:**

Defines the location of the XCLKOUTDIV bits in the XCLK register.

## 5.2.3 Typedef Documentation

### 5.2.3.1 CLK\_Handle

**Definition:**

```
typedef struct CLK_Obj *CLK_Handle
```

**Description:**

Defines the clock (CLK) handle.

### 5.2.3.2 CLK\_Obj

**Definition:**

```
typedef struct _CLK_Obj_ CLK_Obj
```

**Description:**

Defines the clock (CLK) object.

## 5.2.4 Enumeration Documentation

### 5.2.4.1 CLK\_ClkInSrc\_e

**Description:**

Enumeration to define the clock in source.

### 5.2.4.2 enum CLK\_ClkOutPreScaler\_e

Enumeration to define the external clock output frequency.

**Enumerators:**

**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.

### 5.2.4.3 CLK\_CompNumber\_e

**Description:**

Enumeration to define the comparator numbers.

**Enumerators:**

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

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

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

#### 5.2.4.4 CLK\_CpuTimerNumber\_e

**Description:**

Enumeration to define the CPU timer numbers.

**Enumerators:**

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

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

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

#### 5.2.4.5 CLK\_LowSpdPreScaler\_e

**Description:**

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

**Enumerators:**

**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.

#### 5.2.4.6 CLK\_Osc2Src\_e

**Description:**

Enumeration to define the clock oscillator 2 source.

**Enumerators:**

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

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

#### 5.2.4.7 CLK\_OscSrc\_e

**Description:**

Enumeration to define the clock oscillator source.

**Enumerators:**

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

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

#### 5.2.4.8 CLK\_Timer2PreScaler\_e

**Description:**

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

**Enumerators:**

- 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.

#### 5.2.4.9 CLK\_Timer2Src\_e

**Description:**

Enumeration to define the timer 2 source.

**Enumerators:**

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

#### 5.2.4.10 CLK\_WdClkSrc\_e

**Description:**

Enumeration to define the watchdog clock source.

**Enumerators:**

- 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.

### 5.2.5 Function Documentation

#### 5.2.5.1 CLK\_disableAdcClock

Disables the ADC clock.

**Prototype:**

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

**Parameters:**

← **clkHandle** The clock (CLK) object handle

#### 5.2.5.2 void CLK\_disableClaClock (CLK\_Handle *clkHandle*)

Disables the CLA clock.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

#### 5.2.5.3 void CLK\_disableClkIn (CLK\_Handle *clkHandle*)

Disables the XCLKIN oscillator input.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

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

Disables the comparator clock.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

← ***compNumber*** The comparator number

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

Disables the CPU timer clock.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

← ***cpuTimerNumber*** The CPU timer number

#### 5.2.5.6 void CLK\_disableCrystalOsc (CLK\_Handle *clkHandle*)

Disables the crystal oscillator.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

#### 5.2.5.7 void CLK\_disableEcanaClock (CLK\_Handle *clkHandle*)

Disables the ECANA clock.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

#### 5.2.5.8 void CLK\_disableEcap1Clock (CLK\_Handle clkHandle)

Disables the ECAP1 clock.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

#### 5.2.5.9 void CLK\_disableEqep1Clock (CLK\_Handle clkHandle)

Disables the EQEP1 clock.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

#### 5.2.5.10 void CLK\_disableGpioInputClock (CLK\_Handle clkHandle)

Disables the GPIO input clock.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

#### 5.2.5.11 void CLK\_disableHrPwmClock (CLK\_Handle clkHandle)

Disables the HRPWM clock.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

#### 5.2.5.12 void CLK\_disableI2cClock (CLK\_Handle clkHandle)

Disables the I2C clock.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

#### 5.2.5.13 void CLK\_disableLinAClock (CLK\_Handle clkHandle)

Disables the LIN-A clock.

**Parameters:**

← **clkHandle** The clock (CLK) object handle



#### 5.2.5.14 void CLK\_disableOsc1 (CLK\_Handle clkHandle)

Disables internal oscillator 1.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

#### 5.2.5.15 void CLK\_disableOsc1HaltMode (CLK\_Handle clkHandle)

Disables internal oscillator 1 halt mode ignore.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

#### 5.2.5.16 void CLK\_disableOsc2 (CLK\_Handle clkHandle)

Disables internal oscillator 2.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

#### 5.2.5.17 void CLK\_disableOsc2HaltMode (CLK\_Handle clkHandle)

Disables internal oscillator 2 halt mode ignore.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

#### 5.2.5.18 void CLK\_disablePwmClock (CLK\_Handle clkHandle, const PWM\_Number\_e pwmNumber)

Disables the pwm clock.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

← **pwmNumber** The PWM number

#### 5.2.5.19 void CLK\_disableSciaClock (CLK\_Handle clkHandle)

Disables the SCI-A clock.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

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

Disables the SPI-A clock.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

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

Disables the SPI-B clock.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

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

Disables the ePWM module time base clock sync signal.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

5.2.5.23 void CLK\_disableWatchDogHaltMode (CLK\_Handle *clkHandle*)

Disables the watchdog halt mode ignore.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

5.2.5.24 void CLK\_enableAdcClock (CLK\_Handle *clkHandle*)

Enables the ADC clock.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

5.2.5.25 void CLK\_enableClaClock (CLK\_Handle *clkHandle*)

Enables the CLA clock.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

**5.2.5.26** void CLK\_enableClkIn (CLK\_Handle *clkHandle*)

Enables the XCLKIN oscillator input.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

**5.2.5.27** void CLK\_enableCompClock (CLK\_Handle *clkHandle*, const CLK\_CompNumber\_e *compNumber*)

Enables the comparator clock.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

← ***compNumber*** The comparator number

**5.2.5.28** void CLK\_enableCpuTimerClock (CLK\_Handle *clkHandle*, const CLK\_CpuTimerNumber\_e *cpuTimerNumber*)

Enables the CPU timer clock.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

← ***cpuTimerNumber*** The CPU timer number

**5.2.5.29** void CLK\_enableCrystalOsc (CLK\_Handle *clkHandle*)

Enables the crystal oscillator.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

**5.2.5.30** void CLK\_enableEcanaClock (CLK\_Handle *clkHandle*)

Enables the ECANA clock.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

**5.2.5.31** void CLK\_enableEcap1Clock (CLK\_Handle *clkHandle*)

Enables the ECAP1 clock.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

5.2.5.32 void CLK\_enableEqep1Clock (CLK\_Handle clkHandle)

Enables the EQEP1 clock.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

5.2.5.33 void CLK\_enableGpioInputClock (CLK\_Handle clkHandle)

Enables the GPIO input clock.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

5.2.5.34 void CLK\_enableHrPwmClock (CLK\_Handle clkHandle)

Enables the HRPWM clock.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

5.2.5.35 void CLK\_enableI2cClock (CLK\_Handle clkHandle)

Enables the I2C clock.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

5.2.5.36 void CLK\_enableLinAClock (CLK\_Handle clkHandle)

Enables the LIN-A clock.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

5.2.5.37 void CLK\_enableOsc1 (CLK\_Handle clkHandle)

Enables internal oscillator 1.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

5.2.5.38 void CLK\_enableOsc1HaltMode (CLK\_Handle clkHandle)

Enables internal oscillator 1 halt mode ignore.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

5.2.5.39 void CLK\_enableOsc2 (CLK\_Handle clkHandle)

Enables internal oscillator 2.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

5.2.5.40 void CLK\_enableOsc2HaltMode (CLK\_Handle clkHandle)

Enables internal oscillator 2 halt mode ignore.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

5.2.5.41 void CLK\_enablePwmClock (CLK\_Handle clkHandle, const PWM\_Number\_e pwmNumber)

Enables the pwm clock.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

← **pwmNumber** The PWM number

5.2.5.42 void CLK\_enableSciaClock (CLK\_Handle clkHandle)

Enables the SCI-A clock.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

5.2.5.43 void CLK\_enableSpiaClock (CLK\_Handle clkHandle)

Enables the SPI-A clock.

**Parameters:**

← **clkHandle** The clock (CLK) object handle

#### 5.2.5.44 void CLK\_enableSpibClock (CLK\_Handle *clkHandle*)

Enables the SPI-B clock.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

#### 5.2.5.45 void CLK\_enableTbClockSync (CLK\_Handle *clkHandle*)

Enables the ePWM module time base clock sync signal.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

#### 5.2.5.46 void CLK\_enableWatchDogHaltMode (CLK\_Handle *clkHandle*)

Enables the watchdog halt mode ignore.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

#### 5.2.5.47 CLK\_Handle CLK\_init (void \* *pMemory*, const size\_t *numBytes*)

Initializes the clock (CLK) object handle.

**Parameters:**

← ***pMemory*** A pointer to the base address of the CLK registers

← ***numBytes*** The number of bytes allocated for the CLK object, bytes

**Returns:**

The clock (CLK) object handle

#### 5.2.5.48 void CLK\_setClkOutPreScaler (CLK\_Handle *clkHandle*, const CLK\_ClkOutPreScaler\_e *preScaler*)

Sets the external clock out prescaler.

**Parameters:**

← ***clkHandle*** The clock (CLK) object handle

← ***preScaler*** The prescaler value

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

Sets the low speed peripheral clock prescaler.

**Parameters:**

- ← ***clkHandle*** The clock (CLK) object handle
- ← ***preScaler*** The prescaler value

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

Sets the oscillator 2 clock source.

**Parameters:**

- ← ***clkHandle*** The clock (CLK) object handle
- ← ***src*** The oscillator 2 clock source

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

Sets the oscillator clock source.

**Parameters:**

- ← ***clkHandle*** The clock (CLK) object handle
- ← ***src*** The oscillator clock source

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

Sets the timer 2 clock prescaler.

**Parameters:**

- ← ***clkHandle*** The clock (CLK) object handle
- ← ***preScaler*** The prescaler value

5.2.5.53 void CLK\_setTimer2Src (CLK\_Handle *clkHandle*, const CLK\_Timer2Src\_e *src*)

Sets the timer 2 clock source.

**Parameters:**

- ← ***clkHandle*** The clock (CLK) object handle
- ← ***src*** The timer 2 clock source

5.2.5.54 void CLK\_setWatchDogSrc (CLK\_Handle *clkHandle*, const CLK\_WdClkSrc\_e *src*)

Sets the watchdog clock source.

**Parameters:**

- ← ***clkHandle*** The clock (CLK) object handle
- ← ***src*** The watchdog clock source



## 6 Comparater (COMP)

Introduction .....	73
API Functions .....	73

### 6.1 Introduction

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

This driver is contained in `f280x0_common/source/comp.c`, with `f280x0_common/include/comp.h` containing the API definitions for use by applications.

### 6.2 COMP

#### Data Structures

- [\\_COMP\\_Obj\\_](#)

#### Defines

- [COMP1\\_BASE\\_ADDR](#)
- [COMP2\\_BASE\\_ADDR](#)
- [COMP\\_COMPCTL\\_CMPINV\\_BITS](#)
- [COMP\\_COMPCTL\\_COMPDACE\\_BITS](#)
- [COMP\\_COMPCTL\\_COMPSOURCE\\_BITS](#)
- [COMP\\_COMPCTL\\_QUALSEL\\_BITS](#)
- [COMP\\_COMPCTL\\_SYNCSEL\\_BITS](#)
- [COMP\\_COMPSTS\\_COMPSTS\\_BITS](#)
- [COMP\\_DACCTL\\_DACSOURCE\\_BITS](#)
- [COMP\\_DACCTL\\_FREESOFT\\_BITS](#)
- [COMP\\_DACCTL\\_RAMPSOURCE\\_BITS](#)

#### Enumerations

- [COMP\\_QualSel\\_e](#)
- [COMP\\_RampSyncSrc\\_e](#)

#### 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)

## 6.2.1 Data Structure Documentation

### 6.2.1.1 \_COMP\_Obj\_

**Definition:**

```
typedef struct
{
    uint16_t COMPCTL;
    uint16_t rsvd_1;
    uint16_t COMPSTS;
    uint16_t rsvd_2;
    uint16_t DACCTL;
    uint16_t rsvd_3;
    uint16_t DACVAL;
    uint16_t rsvd_4;
    uint16_t RAMPMAXREF_ACTIVE;
    uint16_t rsvd_5;
    uint16_t RAMPMAXREF_SHADOW;
    uint16_t rsvd_6;
    uint16_t RAMPDECVAL_ACTIVE;
    uint16_t rsvd_7;
    uint16_t RAMPDECVAL_SHADOW;
    uint16_t rsvd_8;
    uint16_t RAMPSTS;
}
_COMP_Obj_
```

**Members:**

**COMPCTL** COMP Control Register.

**rsvd\_1** Reserved.

**COMPSTS** COMP Status Register.

**rsvd\_2** Reserved.

**DACCTL** DAC Control Register.

**rsvd\_3** Reserved.

**DACVAL** DAC Value Register.

**rsvd\_4** Reserved.

**RAMPMAXREF\_ACTIVE** Ramp Generator Maximum Reference (Active).

**rsvd\_5** Reserved.

**RAMPMAXREF\_SHADOW** Ramp Generator Maximum Reference (Shadow).

**rsvd\_6** Reserved.

**RAMPDECVAL\_ACTIVE** Ramp Generator Decrement Value (Active).

**rsvd\_7** Reserved.

**RAMPDECVAL\_SHADOW** Ramp Generator Decrement Value (Shadow).

**rsvd\_8** Reserved.

**RAMPSTS** Ramp Generator Status.

**Description:**

Defines the comparator (COMP) object.

## 6.2.2 Define Documentation

### 6.2.2.1 COMP1\_BASE\_ADDR

**Definition:**

```
#define COMP1_BASE_ADDR
```

**Description:**

Defines the base address of the comparator (COMP) registers.

### 6.2.2.2 COMP2\_BASE\_ADDR

**Definition:**

```
#define COMP2_BASE_ADDR
```

**Description:**

Defines the base address of the comparator (COMP) registers.

### 6.2.2.3 COMP\_COMPCTL\_CMPINV\_BITS

**Definition:**

```
#define COMP_COMPCTL_CMPINV_BITS
```

**Description:**

Defines the location of the CMPINV bits in the COMPCTL register.

### 6.2.2.4 COMP\_COMPCTL\_COMPDACE\_BITS

**Definition:**

```
#define COMP_COMPCTL_COMPDACE_BITS
```

**Description:**

Defines the location of the COMPDACE bits in the COMPCTL register.

### 6.2.2.5 COMP\_COMPCTL\_COMPSOURCE\_BITS

**Definition:**

```
#define COMP_COMPCTL_COMPSOURCE_BITS
```

**Description:**

Defines the location of the COMPSOURCE bits in the COMPCTL register.

#### 6.2.2.6 COMP\_COMPCTL\_QUALSEL\_BITS

**Definition:**

```
#define COMP_COMPCTL_QUALSEL_BITS
```

**Description:**

Defines the location of the QUALSEL bits in the COMPCTL register.

#### 6.2.2.7 COMP\_COMPCTL\_SYNCSEL\_BITS

**Definition:**

```
#define COMP_COMPCTL_SYNCSEL_BITS
```

**Description:**

Defines the location of the SYNCSEL bits in the COMPCTL register.

#### 6.2.2.8 COMP\_COMPSTS\_COMPSTS\_BITS

**Definition:**

```
#define COMP_COMPSTS_COMPSTS_BITS
```

**Description:**

Defines the location of the COMPSTS bits in the COMPSTS register.

#### 6.2.2.9 COMP\_DACCTL\_DACSOURCE\_BITS

**Definition:**

```
#define COMP_DACCTL_DACSOURCE_BITS
```

**Description:**

Defines the location of the DACSOURCE bits in the DACCTL register.

#### 6.2.2.10 COMP\_DACCTL\_FREESOFT\_BITS

**Definition:**

```
#define COMP_DACCTL_FREESOFT_BITS
```

**Description:**

Defines the location of the FREE:SOFT bits in the DACCTL register.

### 6.2.2.11 COMP\_DACCTL\_RAMPSOURCE\_BITS

**Definition:**

```
#define COMP_DACCTL_RAMPSOURCE_BITS
```

**Description:**

Defines the location of the RAMPSOURCE bits in the DACCTL register.

## 6.2.3 Typedef Documentation

### 6.2.3.1 COMP\_Handle

**Definition:**

```
typedef struct COMP_Obj *COMP_Handle
```

**Description:**

Defines the comparator (COMP) handle.

### 6.2.3.2 COMP\_Obj

**Definition:**

```
typedef struct _COMP_Obj_ COMP_Obj
```

**Description:**

Defines the comparator (COMP) object.

## 6.2.4 Enumeration Documentation

### 6.2.4.1 COMP\_QualSel\_e

**Description:**

Enumeration to define the comparator (COMP) output qualification.

**Enumerators:**

**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.

#### 6.2.4.2 COMP\_RampSyncSrc\_e

**Description:**

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

**Enumerators:**

**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.

### 6.2.5 Function Documentation

#### 6.2.5.1 COMP\_disable

Disables the comparator (COMP).

**Prototype:**

```
void  
COMP_disable(COMP\_Handle compHandle)
```

**Parameters:**

← **compHandle** The comparator (COMP) object handle

**6.2.5.2** void COMP\_disableDac (**COMP\_Handle** compHandle)

Disables the DAC.

**Parameters:**

← **compHandle** The comparator (COMP) object handle

**6.2.5.3** void COMP\_enable (**COMP\_Handle** compHandle)

Enables the comparator (COMP).

**Parameters:**

← **compHandle** The comparator (COMP) object handle

**6.2.5.4** void COMP\_enableDac (**COMP\_Handle** compHandle)

Enables the DAC.

**Parameters:**

← **compHandle** The comparator (COMP) object handle

**6.2.5.5** **COMP\_Handle** COMP\_init (void \* pMemory, const size\_t numBytes)

Initializes the comparator (COMP) object handle.

**Parameters:**

← **pMemory** A pointer to the base address of the COMP registers

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

**Returns:**

The comparator (COMP) object handle

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

Sets the DAC's value.

**Parameters:**

← **compHandle** The comparator (COMP) object handle

← **dacValue** The DAC's value





## 7 Central Processing Unit (CPU)

Introduction .....	81
API Functions .....	81

### 7.1 Introduction

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.

### 7.2 CPU

#### Data Structures

- `_CPU_Obj_`

#### Defines

- `CPU_DBGIER_DLOGINT_BITS`
- `CPU_DBGIER_INT10_BITS`
- `CPU_DBGIER_INT11_BITS`
- `CPU_DBGIER_INT12_BITS`
- `CPU_DBGIER_INT13_BITS`
- `CPU_DBGIER_INT14_BITS`
- `CPU_DBGIER_INT1_BITS`
- `CPU_DBGIER_INT2_BITS`
- `CPU_DBGIER_INT3_BITS`
- `CPU_DBGIER_INT4_BITS`
- `CPU_DBGIER_INT5_BITS`
- `CPU_DBGIER_INT6_BITS`
- `CPU_DBGIER_INT7_BITS`
- `CPU_DBGIER_INT8_BITS`
- `CPU_DBGIER_INT9_BITS`
- `CPU_DBGIER_RTOSINT_BITS`
- `CPU_IER_DLOGINT_BITS`
- `CPU_IER_INT10_BITS`
- `CPU_IER_INT11_BITS`
- `CPU_IER_INT12_BITS`
- `CPU_IER_INT13_BITS`

- CPU\_IER\_INT14\_BITS
- CPU\_IER\_INT1\_BITS
- CPU\_IER\_INT2\_BITS
- CPU\_IER\_INT3\_BITS
- CPU\_IER\_INT4\_BITS
- CPU\_IER\_INT5\_BITS
- CPU\_IER\_INT6\_BITS
- CPU\_IER\_INT7\_BITS
- CPU\_IER\_INT8\_BITS
- CPU\_IER\_INT9\_BITS
- CPU\_IER\_RTOSINT\_BITS
- CPU\_IFR\_DLOGINT\_BITS
- CPU\_IFR\_INT10\_BITS
- CPU\_IFR\_INT11\_BITS
- CPU\_IFR\_INT12\_BITS
- CPU\_IFR\_INT13\_BITS
- CPU\_IFR\_INT14\_BITS
- CPU\_IFR\_INT1\_BITS
- CPU\_IFR\_INT2\_BITS
- CPU\_IFR\_INT3\_BITS
- CPU\_IFR\_INT4\_BITS
- CPU\_IFR\_INT5\_BITS
- CPU\_IFR\_INT6\_BITS
- CPU\_IFR\_INT7\_BITS
- CPU\_IFR\_INT8\_BITS
- CPU\_IFR\_INT9\_BITS
- CPU\_IFR\_RTOSINT\_BITS
- CPU\_ST0\_C\_BITS
- CPU\_ST0\_N\_BITS
- CPU\_ST0\_OVCOVCU\_BITS
- CPU\_ST0\_OVM\_BITS
- CPU\_ST0\_PW\_BITS
- CPU\_ST0\_SXM\_BITS
- CPU\_ST0\_TC\_BITS
- CPU\_ST0\_V\_BITS
- CPU\_ST0\_Z\_BITS
- CPU\_ST1\_AMODE\_BITS
- CPU\_ST1\_ARP\_BITS
- CPU\_ST1\_DBGM\_BITS
- CPU\_ST1\_EALLOW\_BITS
- CPU\_ST1\_IDLESTAT\_BITS
- CPU\_ST1\_INTM\_BITS
- CPU\_ST1\_LOOP\_BITS
- CPU\_ST1\_MOM1MAP\_BITS
- CPU\_ST1\_OBJMODE\_BITS
- CPU\_ST1\_PAGE0\_BITS

- CPU\_ST1\_SPA\_BITS
- CPU\_ST1\_VMAP\_BITS
- CPU\_ST1\_XF\_BITS
- DINT
- DISABLE\_INTERRUPTS
- DISABLE\_PROTECTED\_REGISTER\_WRITE\_MODE
- DRTM
- EALLOW
- EDIS
- EINT
- ENABLE\_INTERRUPTS
- ENABLE\_PROTECTED\_REGISTER\_WRITE\_MODE
- ERTM
- ESTOP0
- IDLE

## Enumerations

- CPU\_ExtIntNumber\_e
- CPU\_IntNumber\_e

## 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

## 7.2.1 Data Structure Documentation

### 7.2.1.1 `_CPU_Obj_`

**Definition:**

```
typedef struct
{
    HASH(0x2841838) argsstring;
}
_CPU_Obj_
```

**Members:**

***argsstring***

**Description:**

Defines the central processing unit (CPU) object.

## 7.2.2 Define Documentation

### 7.2.2.1 `CPU_DBGIER_DLOGINT_BITS`

**Definition:**

```
#define CPU_DBGIER_DLOGINT_BITS
```

**Description:**

Defines the location of the DLOGINT bits in the DBGIER register.

### 7.2.2.2 `CPU_DBGIER_INT10_BITS`

**Definition:**

```
#define CPU_DBGIER_INT10_BITS
```

**Description:**

Defines the location of the INT10 bits in the DBGIER register.

### 7.2.2.3 `CPU_DBGIER_INT11_BITS`

**Definition:**

```
#define CPU_DBGIER_INT11_BITS
```

**Description:**

Defines the location of the INT11 bits in the DBGIER register.

### 7.2.2.4 `CPU_DBGIER_INT12_BITS`

**Definition:**

```
#define CPU_DBGIER_INT12_BITS
```

**Description:**

Defines the location of the INT12 bits in the DBGIER register.

#### 7.2.2.5 CPU\_DBGIER\_INT13\_BITS

**Definition:**

```
#define CPU_DBGIER_INT13_BITS
```

**Description:**

Defines the location of the INT13 bits in the DBGIER register.

#### 7.2.2.6 CPU\_DBGIER\_INT14\_BITS

**Definition:**

```
#define CPU_DBGIER_INT14_BITS
```

**Description:**

Defines the location of the INT14 bits in the DBGIER register.

#### 7.2.2.7 CPU\_DBGIER\_INT1\_BITS

**Definition:**

```
#define CPU_DBGIER_INT1_BITS
```

**Description:**

Defines the location of the INT1 bits in the DBGIER register.

#### 7.2.2.8 CPU\_DBGIER\_INT2\_BITS

**Definition:**

```
#define CPU_DBGIER_INT2_BITS
```

**Description:**

Defines the location of the INT2 bits in the DBGIER register.

#### 7.2.2.9 CPU\_DBGIER\_INT3\_BITS

**Definition:**

```
#define CPU_DBGIER_INT3_BITS
```

**Description:**

Defines the location of the INT3 bits in the DBGIER register.

#### 7.2.2.10 CPU\_DBGIER\_INT4\_BITS

**Definition:**

```
#define CPU_DBGIER_INT4_BITS
```

**Description:**

Defines the location of the INT4 bits in the DBGIER register.

#### 7.2.2.11 CPU\_DBGIER\_INT5\_BITS

**Definition:**

```
#define CPU_DBGIER_INT5_BITS
```

**Description:**

Defines the location of the INT5 bits in the DBGIER register.

#### 7.2.2.12 CPU\_DBGIER\_INT6\_BITS

**Definition:**

```
#define CPU_DBGIER_INT6_BITS
```

**Description:**

Defines the location of the INT6 bits in the DBGIER register.

#### 7.2.2.13 CPU\_DBGIER\_INT7\_BITS

**Definition:**

```
#define CPU_DBGIER_INT7_BITS
```

**Description:**

Defines the location of the INT7 bits in the DBGIER register.

#### 7.2.2.14 CPU\_DBGIER\_INT8\_BITS

**Definition:**

```
#define CPU_DBGIER_INT8_BITS
```

**Description:**

Defines the location of the INT8 bits in the DBGIER register.

#### 7.2.2.15 CPU\_DBGIER\_INT9\_BITS

**Definition:**

```
#define CPU_DBGIER_INT9_BITS
```

**Description:**

Defines the location of the INT9 bits in the DBGIER register.

#### 7.2.2.16 CPU\_DBGIER\_RTOSINT\_BITS

**Definition:**

```
#define CPU_DBGIER_RTOSINT_BITS
```

**Description:**

Defines the location of the RTOSINT bits in the DBGIER register.

#### 7.2.2.17 CPU\_IER\_DLOGINT\_BITS

**Definition:**

```
#define CPU_IER_DLOGINT_BITS
```

**Description:**

Defines the location of the DLOGINT bits in the IER register.

#### 7.2.2.18 CPU\_IER\_INT10\_BITS

**Definition:**

```
#define CPU_IER_INT10_BITS
```

**Description:**

Defines the location of the INT10 bits in the IER register.

#### 7.2.2.19 CPU\_IER\_INT11\_BITS

**Definition:**

```
#define CPU_IER_INT11_BITS
```

**Description:**

Defines the location of the INT11 bits in the IER register.

#### 7.2.2.20 CPU\_IER\_INT12\_BITS

**Definition:**

```
#define CPU_IER_INT12_BITS
```

**Description:**

Defines the location of the INT12 bits in the IER register.

#### 7.2.2.21 CPU\_IER\_INT13\_BITS

**Definition:**

```
#define CPU_IER_INT13_BITS
```

**Description:**

Defines the location of the INT13 bits in the IER register.

#### 7.2.2.22 CPU\_IER\_INT14\_BITS

**Definition:**

```
#define CPU_IER_INT14_BITS
```

**Description:**

Defines the location of the INT14 bits in the IER register.

#### 7.2.2.23 CPU\_IER\_INT1\_BITS

**Definition:**

```
#define CPU_IER_INT1_BITS
```

**Description:**

Defines the location of the INT1 bits in the IER register.

#### 7.2.2.24 CPU\_IER\_INT2\_BITS

**Definition:**

```
#define CPU_IER_INT2_BITS
```

**Description:**

Defines the location of the INT2 bits in the IER register.

#### 7.2.2.25 CPU\_IER\_INT3\_BITS

**Definition:**

```
#define CPU_IER_INT3_BITS
```

**Description:**

Defines the location of the INT3 bits in the IER register.

#### 7.2.2.26 CPU\_IER\_INT4\_BITS

**Definition:**

```
#define CPU_IER_INT4_BITS
```

**Description:**

Defines the location of the INT4 bits in the IER register.

#### 7.2.2.27 CPU\_IER\_INT5\_BITS

**Definition:**

```
#define CPU_IER_INT5_BITS
```

**Description:**

Defines the location of the INT5 bits in the IER register.



#### 7.2.2.28 CPU\_IER\_INT6\_BITS

**Definition:**

```
#define CPU_IER_INT6_BITS
```

**Description:**

Defines the location of the INT6 bits in the IER register.

#### 7.2.2.29 CPU\_IER\_INT7\_BITS

**Definition:**

```
#define CPU_IER_INT7_BITS
```

**Description:**

Defines the location of the INT7 bits in the IER register.

#### 7.2.2.30 CPU\_IER\_INT8\_BITS

**Definition:**

```
#define CPU_IER_INT8_BITS
```

**Description:**

Defines the location of the INT8 bits in the IER register.

#### 7.2.2.31 CPU\_IER\_INT9\_BITS

**Definition:**

```
#define CPU_IER_INT9_BITS
```

**Description:**

Defines the location of the INT9 bits in the IER register.

#### 7.2.2.32 CPU\_IER\_RTOSINT\_BITS

**Definition:**

```
#define CPU_IER_RTOSINT_BITS
```

**Description:**

Defines the location of the RTOSINT bits in the IER register.

#### 7.2.2.33 CPU\_IFR\_DLOGINT\_BITS

**Definition:**

```
#define CPU_IFR_DLOGINT_BITS
```

**Description:**

Defines the location of the DLOGINT bits in the IFR register.

#### 7.2.2.34 CPU\_IFR\_INT10\_BITS

**Definition:**

```
#define CPU_IFR_INT10_BITS
```

**Description:**

Defines the location of the INT10 bits in the IER register.

#### 7.2.2.35 CPU\_IFR\_INT11\_BITS

**Definition:**

```
#define CPU_IFR_INT11_BITS
```

**Description:**

Defines the location of the INT11 bits in the IER register.

#### 7.2.2.36 CPU\_IFR\_INT12\_BITS

**Definition:**

```
#define CPU_IFR_INT12_BITS
```

**Description:**

Defines the location of the INT12 bits in the IER register.

#### 7.2.2.37 CPU\_IFR\_INT13\_BITS

**Definition:**

```
#define CPU_IFR_INT13_BITS
```

**Description:**

Defines the location of the INT13 bits in the IER register.

#### 7.2.2.38 CPU\_IFR\_INT14\_BITS

**Definition:**

```
#define CPU_IFR_INT14_BITS
```

**Description:**

Defines the location of the INT14 bits in the IER register.

#### 7.2.2.39 CPU\_IFR\_INT1\_BITS

**Definition:**

```
#define CPU_IFR_INT1_BITS
```

**Description:**

Defines the location of the INT1 bits in the IER register.

#### 7.2.2.40 CPU\_IFR\_INT2\_BITS

**Definition:**

```
#define CPU_IFR_INT2_BITS
```

**Description:**

Defines the location of the INT2 bits in the IER register.

#### 7.2.2.41 CPU\_IFR\_INT3\_BITS

**Definition:**

```
#define CPU_IFR_INT3_BITS
```

**Description:**

Defines the location of the INT3 bits in the IER register.

#### 7.2.2.42 CPU\_IFR\_INT4\_BITS

**Definition:**

```
#define CPU_IFR_INT4_BITS
```

**Description:**

Defines the location of the INT4 bits in the IER register.

#### 7.2.2.43 CPU\_IFR\_INT5\_BITS

**Definition:**

```
#define CPU_IFR_INT5_BITS
```

**Description:**

Defines the location of the INT5 bits in the IER register.

#### 7.2.2.44 CPU\_IFR\_INT6\_BITS

**Definition:**

```
#define CPU_IFR_INT6_BITS
```

**Description:**

Defines the location of the INT6 bits in the IER register.

#### 7.2.2.45 CPU\_IFR\_INT7\_BITS

**Definition:**

```
#define CPU_IFR_INT7_BITS
```

**Description:**

Defines the location of the INT7 bits in the IER register.

#### 7.2.2.46 CPU\_IFR\_INT8\_BITS

**Definition:**

```
#define CPU_IFR_INT8_BITS
```

**Description:**

Defines the location of the INT8 bits in the IER register.

#### 7.2.2.47 CPU\_IFR\_INT9\_BITS

**Definition:**

```
#define CPU_IFR_INT9_BITS
```

**Description:**

Defines the location of the INT9 bits in the IER register.

#### 7.2.2.48 CPU\_IFR\_RTOSINT\_BITS

**Definition:**

```
#define CPU_IFR_RTOSINT_BITS
```

**Description:**

Defines the location of the RTOSINT bits in the IFR register.

#### 7.2.2.49 CPU\_ST0\_C\_BITS

**Definition:**

```
#define CPU_ST0_C_BITS
```

**Description:**

Defines the location of the C bits in the ST0 register.

#### 7.2.2.50 CPU\_ST0\_N\_BITS

**Definition:**

```
#define CPU_ST0_N_BITS
```

**Description:**

Defines the location of the N bits in the ST0 register.

#### 7.2.2.51 CPU\_ST0\_OVCOVCU\_BITS

**Definition:**

```
#define CPU_ST0_OVCOVCU_BITS
```

**Description:**

Defines the location of the OVCOVCU bits in the ST0 register.

#### 7.2.2.52 CPU\_ST0\_OVM\_BITS

**Definition:**

```
#define CPU_ST0_OVM_BITS
```

**Description:**

Defines the location of the OVM bits in the ST0 register.

#### 7.2.2.53 CPU\_ST0\_PW\_BITS

**Definition:**

```
#define CPU_ST0_PW_BITS
```

**Description:**

Defines the location of the PW bits in the ST0 register.

#### 7.2.2.54 CPU\_ST0\_SXM\_BITS

**Definition:**

```
#define CPU_ST0_SXM_BITS
```

**Description:**

Defines the location of the SXM bits in the ST0 register.

#### 7.2.2.55 CPU\_ST0\_TC\_BITS

**Definition:**

```
#define CPU_ST0_TC_BITS
```

**Description:**

Defines the location of the T bits in the ST0 register.

#### 7.2.2.56 CPU\_ST0\_V\_BITS

**Definition:**

```
#define CPU_ST0_V_BITS
```

**Description:**

Defines the location of the V bits in the ST0 register.

#### 7.2.2.57 CPU\_ST0\_Z\_BITS

**Definition:**

```
#define CPU_ST0_Z_BITS
```

**Description:**

Defines the location of the Z bits in the ST0 register.

#### 7.2.2.58 CPU\_ST1\_AMODE\_BITS

**Definition:**

```
#define CPU_ST1_AMODE_BITS
```

**Description:**

Defines the location of the AMODE bits in the ST1 register.

#### 7.2.2.59 CPU\_ST1\_ARP\_BITS

**Definition:**

```
#define CPU_ST1_ARP_BITS
```

**Description:**

Defines the location of the ARP bits in the ST1 register.

#### 7.2.2.60 CPU\_ST1\_DBGM\_BITS

**Definition:**

```
#define CPU_ST1_DBGM_BITS
```

**Description:**

Defines the location of the DBGM bits in the ST1 register.

#### 7.2.2.61 CPU\_ST1\_EALLOW\_BITS

**Definition:**

```
#define CPU_ST1_EALLOW_BITS
```

**Description:**

Defines the location of the EALLOW bits in the ST1 register.

#### 7.2.2.62 CPU\_ST1\_IDLESTAT\_BITS

**Definition:**

```
#define CPU_ST1_IDLESTAT_BITS
```

**Description:**

Defines the location of the IDLESTAT bits in the ST1 register.

#### 7.2.2.63 CPU\_ST1\_INTM\_BITS

**Definition:**

```
#define CPU_ST1_INTM_BITS
```

**Description:**

Defines the location of the INTM bits in the ST1 register.

#### 7.2.2.64 CPU\_ST1\_LOOP\_BITS

**Definition:**

```
#define CPU_ST1_LOOP_BITS
```

**Description:**

Defines the location of the LOOP bits in the ST1 register.

#### 7.2.2.65 CPU\_ST1\_MOM1MAP\_BITS

**Definition:**

```
#define CPU_ST1_MOM1MAP_BITS
```

**Description:**

Defines the location of the MOM1MAP bits in the ST1 register.

#### 7.2.2.66 CPU\_ST1\_OBJMODE\_BITS

**Definition:**

```
#define CPU_ST1_OBJMODE_BITS
```

**Description:**

Defines the location of the OBJMODE bits in the ST1 register.

#### 7.2.2.67 CPU\_ST1\_PAGE0\_BITS

**Definition:**

```
#define CPU_ST1_PAGE0_BITS
```

**Description:**

Defines the location of the PAGE0 bits in the ST1 register.

#### 7.2.2.68 CPU\_ST1\_SPA\_BITS

**Definition:**

```
#define CPU_ST1_SPA_BITS
```

**Description:**

Defines the location of the SPA bits in the ST1 register.

#### 7.2.2.69 CPU\_ST1\_VMAP\_BITS

**Definition:**

```
#define CPU_ST1_VMAP_BITS
```

**Description:**

Defines the location of the VMAP bits in the ST1 register.

#### 7.2.2.70 CPU\_ST1\_XF\_BITS

**Definition:**

```
#define CPU_ST1_XF_BITS
```

**Description:**

Defines the location of the XF bits in the ST1 register.

#### 7.2.2.71 DINT

**Definition:**

```
#define DINT
```

**Description:**

Define to disable interrupts (legacy).

#### 7.2.2.72 DISABLE\_INTERRUPTS

**Definition:**

```
#define DISABLE_INTERRUPTS
```

**Description:**

Define to disable interrupts.

#### 7.2.2.73 DISABLE\_PROTECTED\_REGISTER\_WRITE\_MODE

**Definition:**

```
#define DISABLE_PROTECTED_REGISTER_WRITE_MODE
```

**Description:**

Define to disable protected register writes.

#### 7.2.2.74 DRTM

**Definition:**

```
#define DRTM
```

**Description:**

Define to disable debug events.

#### 7.2.2.75 EALLOW

**Definition:**

```
#define EALLOW
```

**Description:**

Define to allow protected register writes (legacy).



#### 7.2.2.76 EDIS

**Definition:**

```
#define EDIS
```

**Description:**

Define to disable protected register writes (legacy).

#### 7.2.2.77 EINT

**Definition:**

```
#define EINT
```

**Description:**

Define to enable interrupts (legacy).

#### 7.2.2.78 ENABLE\_INTERRUPTS

**Definition:**

```
#define ENABLE_INTERRUPTS
```

**Description:**

Define to enable interrupts.

#### 7.2.2.79 ENABLE\_PROTECTED\_REGISTER\_WRITE\_MODE

**Definition:**

```
#define ENABLE_PROTECTED_REGISTER_WRITE_MODE
```

**Description:**

Define to allow protected register writes.

#### 7.2.2.80 ERTM

**Definition:**

```
#define ERTM
```

**Description:**

Define to enable debug events.

#### 7.2.2.81 ESTOP0

**Definition:**

```
#define ESTOP0
```

**Description:**

Define for emulation stop 0.

### 7.2.2.82 IDLE

**Definition:**

```
#define IDLE
```

**Description:**

Define for entering IDLE mode.

## 7.2.3 Typedef Documentation

### 7.2.3.1 CPU\_Handle

**Definition:**

```
typedef struct CPU_Obj *CPU_Handle
```

**Description:**

Defines the central processing unit (CPU) handle.

### 7.2.3.2 CPU\_Obj

**Definition:**

```
typedef struct _CPU_Obj_ CPU_Obj
```

**Description:**

Defines the central processing unit (CPU) object.

## 7.2.4 Enumeration Documentation

### 7.2.4.1 CPU\_ExtIntNumber\_e

**Description:**

Enumeration to define the external interrupt numbers.

**Enumerators:**

***CPU\_ExtIntNumber\_1*** Denotes external interrupt number 1.

***CPU\_ExtIntNumber\_2*** Denotes external interrupt number 2.

***CPU\_ExtIntNumber\_3*** Denotes external interrupt number 3.

### 7.2.4.2 CPU\_IntNumber\_e

**Description:**

Enumeration to define the interrupt numbers.

**Enumerators:**

***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.

## 7.2.5 Function Documentation

### 7.2.5.1 CPU\_clearIntFlags

Clears all interrupt flags.

**Prototype:**

```
void  
CPU_clearIntFlags(CPU_Handle cpuHandle)
```

**Parameters:**

← ***cpuHandle*** The central processing unit (CPU) object handle

### 7.2.5.2 void CPU\_disableDebugInt (CPU\_Handle cpuHandle)

Disables the debug interrupt.

**Parameters:**

← ***cpuHandle*** The central processing unit (CPU) object handle

### 7.2.5.3 void CPU\_disableGlobalInts (CPU\_Handle cpuHandle)

Disables global interrupts.

**Parameters:**

← ***cpuHandle*** The CPU handle

### 7.2.5.4 void CPU\_disableInt (CPU\_Handle cpuHandle, const CPU\_IntNumber\_e intNumber)

Disables a specified interrupt number.

**Parameters:**

- ← ***cpuHandle*** The central processing unit (CPU) object handle
- ← ***intNumber*** The interrupt number

7.2.5.5 void CPU\_disableInts (**CPU\_Handle** *cpuHandle*)

Disables all interrupts.

**Parameters:**

- ← ***cpuHandle*** The central processing unit (CPU) object handle

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

Disables protected register writes.

**Parameters:**

- ← ***cpuHandle*** The central processing unit (CPU) object handle

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

Enables the debug interrupt.

**Parameters:**

- ← ***cpuHandle*** The CPU handle

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

Enables global interrupts.

**Parameters:**

- ← ***cpuHandle*** The CPU handle

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

Enables a specified interrupt number.

**Parameters:**

- ← ***cpuHandle*** The central processing unit (CPU) object handle
- ← ***intNumber*** The interrupt number

#### 7.2.5.10 void CPU\_enableProtectedRegisterWrite ([CPU\\_Handle](#) *cpuHandle*)

Enables protected register writes.

**Parameters:**

← ***cpuHandle*** The central processing unit (CPU) object handle

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

Initializes the central processing unit (CPU) object handle.

**Parameters:**

← ***pMemory*** A pointer to the memory for the CPU object

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

**Returns:**

The central processing unit (CPU) object handle

### 7.2.6 Variable Documentation

#### 7.2.6.1 [CPU\\_Obj](#) *cpu*

Defines the CPU object.

#### 7.2.6.2 cregister volatile unsigned int [IER](#)

External reference to the interrupt enable register (IER) register.

#### 7.2.6.3 cregister volatile unsigned int [IFR](#)

External reference to the interrupt flag register (IFR) register.



## 8 Flash

Introduction .....	103
API Functions .....	103

### 8.1 Introduction

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.

### 8.2 FLASH

#### Data Structures

- [\\_FLASH\\_Obj\\_](#)

#### Defines

- [FLASH\\_ACTIVE\\_WAIT\\_COUNT\\_DEFAULT](#)
- [FLASH\\_BASE\\_ADDR](#)
- [FLASH\\_FACTIVEWAIT\\_ACTIVWAIT\\_BITS](#)
- [FLASH\\_FBANKWAIT\\_PAGEWAIT\\_BITS](#)
- [FLASH\\_FBANKWAIT\\_RANDWAIT\\_BITS](#)
- [FLASH\\_FOPT\\_ENPIPE\\_BITS](#)
- [FLASH\\_FOTPWAIT\\_OTPWAIT\\_BITS](#)
- [FLASH\\_FPWR\\_PWR\\_BITS](#)
- [FLASH\\_FSTATUS\\_3VSTAT\\_BITS](#)
- [FLASH\\_FSTATUS\\_ACTIVWAITS\\_BITS](#)
- [FLASH\\_FSTATUS\\_PWRS\\_BITS](#)
- [FLASH\\_FSTATUS\\_STDBYWAITS\\_BITS](#)
- [FLASH\\_FSTDBYWAIT\\_STDBYWAIT\\_BITS](#)
- [FLASH\\_STANDBY\\_WAIT\\_COUNT\\_DEFAULT](#)

#### Enumerations

- [FLASH\\_3VStatus\\_e](#)

- FLASH\_CounterStatus\_e
- FLASH\_NumOtpWaitStates\_e
- FLASH\_NumPagedWaitStates\_e
- FLASH\_NumRandomWaitStates\_e
- FLASH\_PowerMode\_e

## 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)

## 8.2.1 Data Structure Documentation

### 8.2.1.1 \_FLASH\_Obj\_

**Definition:**

```
typedef struct
{
    uint16_t FOPT;
    uint16_t rsvd_1;
    uint16_t FPWR;
    uint16_t FSTATUS;
    uint16_t FSTDBYWAIT;
    uint16_t FACTIVEWAIT;
    uint16_t FBANKWAIT;
    uint16_t FOTPWAIT;
}
_FLASH_Obj_
```



**Members:**

**FOPT** Flash Option Register.

**rsvd\_1** Reserved.

**FPWR** Flash Power Modes Register.

**FSTATUS** Status Register.

**FSTDBYWAIT** Flash Sleep To Standby Wait Register.

**FACTIVEWAIT** Flash Standby to Active Wait Register.

**FBANKWAIT** Flash Read Access Wait State Register.

**FOTPWAIT** OTP Read Access Wait State Register.

**Description:**

Defines the flash (FLASH) object.

## 8.2.2 Define Documentation

### 8.2.2.1 FLASH\_ACTIVE\_WAIT\_COUNT\_DEFAULT

**Definition:**

```
#define FLASH_ACTIVE_WAIT_COUNT_DEFAULT
```

**Description:**

Defines the default active wait count in units of SYSCLKOUT cycles.

### 8.2.2.2 FLASH\_BASE\_ADDR

**Definition:**

```
#define FLASH_BASE_ADDR
```

**Description:**

Defines the base address of the flash (FLASH) registers.

### 8.2.2.3 FLASH\_FACTIVEWAIT\_ACTIVEWAIT\_BITS

**Definition:**

```
#define FLASH_FACTIVEWAIT_ACTIVEWAIT_BITS
```

**Description:**

Defines the location of the ACTIVEWAIT bits in the FACTIVEWAIT register.

### 8.2.2.4 FLASH\_FBANKWAIT\_PAGEWAIT\_BITS

**Definition:**

```
#define FLASH_FBANKWAIT_PAGEWAIT_BITS
```

**Description:**

Defines the location of the PAGEWAIT bits in the FBANKWAIT register.

#### 8.2.2.5 FLASH\_FBANKWAIT\_RANDWAIT\_BITS

**Definition:**

```
#define FLASH_FBANKWAIT_RANDWAIT_BITS
```

**Description:**

Defines the location of the RANDWAIT bits in the FBANKWAIT register.

#### 8.2.2.6 FLASH\_FOPT\_ENPIPE\_BITS

**Definition:**

```
#define FLASH_FOPT_ENPIPE_BITS
```

**Description:**

Defines the location of the ENPIPE bits in the FOPT register.

#### 8.2.2.7 FLASH\_FOTPWAIT\_OTPWAIT\_BITS

**Definition:**

```
#define FLASH_FOTPWAIT_OTPWAIT_BITS
```

**Description:**

Defines the location of the OTPWAIT bits in the FOTPWAIT register.

#### 8.2.2.8 FLASH\_FPWR\_PWR\_BITS

**Definition:**

```
#define FLASH_FPWR_PWR_BITS
```

**Description:**

Defines the location of the PWR bits in the FPWR register.

#### 8.2.2.9 FLASH\_FSTATUS\_3VSTAT\_BITS

**Definition:**

```
#define FLASH_FSTATUS_3VSTAT_BITS
```

**Description:**

Defines the location of the 3VSTAT bits in the FSTATUS register.

#### 8.2.2.10 FLASH\_FSTATUS\_ACTIVEWAITS\_BITS

**Definition:**

```
#define FLASH_FSTATUS_ACTIVEWAITS_BITS
```

**Description:**

Defines the location of the ACTIVEWAITS bits in the FSTATUS register.

### 8.2.2.11 FLASH\_FSTATUS\_PWRS\_BITS

**Definition:**

```
#define FLASH_FSTATUS_PWRS_BITS
```

**Description:**

Defines the location of the PWRS bits in the FSTATUS register.

### 8.2.2.12 FLASH\_FSTATUS\_STDBYWAITS\_BITS

**Definition:**

```
#define FLASH_FSTATUS_STDBYWAITS_BITS
```

**Description:**

Defines the location of the STDBYWAITS bits in the FSTATUS register.

### 8.2.2.13 FLASH\_FSTDBYWAIT\_STDBYWAIT\_BITS

**Definition:**

```
#define FLASH_FSTDBYWAIT_STDBYWAIT_BITS
```

**Description:**

Defines the location of the STDBYWAIT bits in the FSTDBYWAIT register.

### 8.2.2.14 FLASH\_STANDBY\_WAIT\_COUNT\_DEFAULT

**Definition:**

```
#define FLASH_STANDBY_WAIT_COUNT_DEFAULT
```

**Description:**

Defines the default standby wait count in units of SYSCLKOUT cycles.

## 8.2.3 Typedef Documentation

### 8.2.3.1 FLASH\_Handle

**Definition:**

```
typedef struct FLASH_Obj *FLASH_Handle
```

**Description:**

Defines the flash (FLASH) handle.

### 8.2.3.2 FLASH\_Obj

**Definition:**

```
typedef struct _FLASH_Obj_ FLASH_Obj
```

**Description:**

Defines the flash (FLASH) object.

## 8.2.4 Enumeration Documentation

### 8.2.4.1 FLASH\_3VStatus\_e

**Description:**

Enumeration to define the 3V status.

**Enumerators:**

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

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

### 8.2.4.2 FLASH\_CounterStatus\_e

**Description:**

Enumeration to define the counter status.

**Enumerators:**

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

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

### 8.2.4.3 FLASH\_NumOtpWaitStates\_e

**Description:**

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

**Enumerators:**

**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.

#### 8.2.4.4 FLASH\_NumPagedWaitStates\_e

**Description:**

Enumeration to define the number of paged wait states.

**Enumerators:**

- 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.

#### 8.2.4.5 FLASH\_NumRandomWaitStates\_e

**Description:**

Enumeration to define the number of random wait states.

**Enumerators:**

- 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.

#### 8.2.4.6 FLASH\_PowerMode\_e

**Description:**

Enumeration to define the power modes.

**Enumerators:**

**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.

### 8.2.5 Function Documentation

#### 8.2.5.1 FLASH\_clear3VStatus

Clears the 3V status.

**Prototype:**

```
void  
FLASH_clear3VStatus(FLASH\_Handle flashHandle)
```

**Parameters:**

← **flashHandle** The flash (FLASH) object handle

---

#### 8.2.5.2 void FLASH\_disablePipelineMode (FLASH\_Handle flashHandle)

Disables the pipeline mode.

**Parameters:**

← **flashHandle** The flash (FLASH) object handle

#### 8.2.5.3 void FLASH\_enablePipelineMode (FLASH\_Handle flashHandle)

Enables the pipeline mode.

**Parameters:**

← **flashHandle** The flash (FLASH) object handle

#### 8.2.5.4 FLASH\_3VStatus\_e FLASH\_get3VStatus (FLASH\_Handle flashHandle)

Gets the 3V status.

**Parameters:**

← **flashHandle** The flash (FLASH) object handle

**Returns:**

The 3V status

#### 8.2.5.5 uint16\_t FLASH\_getActiveWaitCount (FLASH\_Handle flashHandle)

Gets the active wait count.

**Parameters:**

← **flashHandle** The flash (FLASH) object handle

**Returns:**

The active wait count

#### 8.2.5.6 FLASH\_CounterStatus\_e FLASH\_getActiveWaitStatus (FLASH\_Handle flashHandle)

Gets the active wait counter status.

**Parameters:**

← **flashHandle** The flash (FLASH) object handle

**Returns:**

The active wait counter status

**8.2.5.7**    **FLASH\_PowerMode\_e** FLASH\_getPowerMode (**FLASH\_Handle** *flashHandle*)

Gets the power mode.

**Parameters:**

← **flashHandle** The flash (FLASH) object handle

**Returns:**

The power mode

**8.2.5.8**    **uint16\_t** FLASH\_getStandbyWaitCount (**FLASH\_Handle** *flashHandle*)

Gets the standby wait count.

**Parameters:**

← **flashHandle** The flash (FLASH) object handle

**Returns:**

The standby wait count

**8.2.5.9**    **FLASH\_CounterStatus\_e** FLASH\_getStandbyWaitStatus (**FLASH\_Handle** *flashHandle*)

Gets the standby wait counter status.

**Parameters:**

← **flashHandle** The flash (FLASH) object handle

**Returns:**

The standby wait counter status

**8.2.5.10**    **FLASH\_Handle** FLASH\_init (void \* *pMemory*, const size\_t *numBytes*)

Initializes the flash (FLASH) handle.

**Parameters:**

← **pMemory** A pointer to the base address of the FLASH registers

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

**Returns:**

The flash (FLASH) object handle

**8.2.5.11**    void FLASH\_setActiveWaitCount (**FLASH\_Handle** *flashHandle*, const uint16\_t *count*)

Sets the active wait count.



**Parameters:**

- ← **flashHandle** The flash (FLASH) object handle
- ← **count** The active wait count

8.2.5.12 void FLASH\_setNumPagedReadWaitStates ([FLASH\\_Handle](#) flashHandle, const [FLASH\\_NumPagedWaitStates\\_e](#) numStates)

Sets the number of paged read wait states.

**Parameters:**

- ← **flashHandle** The flash (FLASH) object handle
- ← **numStates** The number of paged read wait states

8.2.5.13 void FLASH\_setNumRandomReadWaitStates ([FLASH\\_Handle](#) flashHandle, const [FLASH\\_NumRandomWaitStates\\_e](#) numStates)

Sets the number of random read wait states.

**Parameters:**

- ← **flashHandle** The flash (FLASH) object handle
- ← **numStates** The number of random read wait states

8.2.5.14 void FLASH\_setOtpWaitStates ([FLASH\\_Handle](#) flashHandle, const [FLASH\\_NumOtpWaitStates\\_e](#) numStates)

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

**Parameters:**

- ← **flashHandle** The flash (FLASH) object handle
- ← **numStates** The number of one-time programmable (OTP) wait states

8.2.5.15 void FLASH\_setPowerMode ([FLASH\\_Handle](#) flashHandle, const [FLASH\\_PowerMode\\_e](#) mode)

Sets the power mode.

**Parameters:**

- ← **flashHandle** The flash (FLASH) object handle
- ← **mode** The power mode

8.2.5.16 void FLASH\_setStandbyWaitCount ([FLASH\\_Handle](#) *flashHandle*, const uint16\_t *count*)

Sets the standby wait count.

**Parameters:**

- ← ***flashHandle*** The flash (FLASH) object handle
- ← ***count*** The standby wait count

8.2.5.17 void FLASH\_setup ([FLASH\\_Handle](#) *flashHandle*)

Setup flash for optimal performance.

**Parameters:**

- ← ***flashHandle*** The flash (FLASH) object handle

## 9 General Purpose Input/Output (GPIO)

Introduction .....	115
API Functions .....	115

### 9.1 Introduction

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.

### 9.2 GPIO

#### Data Structures

- [\\_GPIO\\_Obj](#)

#### Defines

- [GPIO\\_BASE\\_ADDR](#)
- [GPIO\\_GPMUX\\_CONFIG\\_BITS](#)

#### Enumerations

- [GPIO\\_Direction\\_e](#)
- [GPIO\\_Mode\\_e](#)
- [GPIO\\_Number\\_e](#)
- [GPIO\\_Port\\_e](#)
- [GPIO\\_PullUp\\_e](#)
- [GPIO\\_Qual\\_e](#)

#### Functions

- `uint16_t` [GPIO\\_getData](#) ([GPIO\\_Handle](#) gpioHandle, const [GPIO\\_Number\\_e](#) gpioNumber)
- `uint16_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 [uint16\\_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](#) gpioNumber, const [uint8\\_t](#) period)
- void [GPIO\\_toggle](#) ([GPIO\\_Handle](#) gpioHandle, const [GPIO\\_Number\\_e](#) gpioNumber)

## 9.2.1 Data Structure Documentation

### 9.2.1.1 [\\_GPIO\\_Obj\\_](#)

**Definition:**

```
typedef struct
{
    uint32_t  GPACTRL;
    uint32_t  GPAQSEL1;
    uint32_t  GPAQSEL2;
    uint32_t  GPAMUX1;
    uint32_t  GPAMUX2;
    uint32_t  GPADIR;
    uint32_t  GPAPUD;
    uint16_t  rsvd_1[2];
    uint32_t  GPBCTRL;
    uint32_t  GPBQSEL1;
    uint16_t  rsvd_2[2];
    uint32_t  GPBMUX1;
    uint16_t  rsvd_3[2];
    uint32_t  GPBDIR;
    uint32_t  GPBPUD;
    uint16_t  rsvd_4[24];
    uint32_t  AIOMUX1;
    uint16_t  rsvd_5[2];
    uint32_t  AIODIR;
    uint16_t  rsvd_6[4];
    uint32_t  GPADAT;
    uint32_t  GPASET;
    uint32_t  GPACLEAR;
    uint32_t  GPATOGGLE;
    uint32_t  GPBDAT;
    uint32_t  GPBSET;
    uint32_t  GPBCLEAR;
```

```

uint32_t GPBTOGGLE;
uint16_t rsvd_7[8];
uint32_t AIODAT;
uint32_t AIOSET;
uint32_t AIOCLEAR;
uint32_t AIOTOGGLE;
uint16_t GPIOXINTnSEL[3];
uint16_t rsvd_8[5];
uint32_t GPIOLPMSEL;
}
__GPIO_Obj_

```

**Members:**

**GPACTRL** GPIO A Control Register.  
**GPAQSEL1** GPIO A Qualifier Select 1 Register.  
**GPAQSEL2** GPIO A Qualifier Select 2 Register.  
**GPAMUX1** GPIO A MUX 1 Register.  
**GPAMUX2** GPIO A MUX 2 Register.  
**GPADIR** GPIO A Direction Register.  
**GPAPUD** GPIO A Pull Up Disable Register.  
**rsvd\_1** Reserved.  
**GPBCTRL** GPIO B Control Register.  
**GPBQSEL1** GPIO B Qualifier Select 1 Register.  
**rsvd\_2** Reserved.  
**GPBMUX1** GPIO B MUX 1 Register.  
**rsvd\_3** Reserved.  
**GPBDIR** GPIO B Direction Register.  
**GPBPUD** GPIO B Pull Up Disable Register.  
**rsvd\_4** Reserved.  
**AIOMUX1** Analog, I/O Mux 1 Register.  
**rsvd\_5** Reserved.  
**AIODIR** Analog, I/O Direction Register.  
**rsvd\_6** Reserved.  
**GPADAT** GPIO A Data Register.  
**GPASET** GPIO A Set Register.  
**GPACLEAR** GPIO A Clear Register.  
**GPATOGGLE** GPIO A Toggle Register.  
**GPBDAT** GPIO B Data Register.  
**GPBSET** GPIO B Set Register.  
**GPBCLEAR** GPIO B Clear Register.  
**GPBTOGGLE** GPIO B Toggle Register.  
**rsvd\_7** Reserved.  
**AIODAT** Analog I/O Data Register.  
**AIOSET** Analog I/O Data Set Register.  
**AIOCLEAR** Analog I/O Clear Register.  
**AIOTOGGLE** Analog I/O Toggle Register.  
**GPIOXINTnSEL** XINT1-3 Source Select Registers.  
**rsvd\_8** Reserved.

**GPIO\_LPMSEL** GPIO Low Power Mode Wakeup Select Register.

**Description:**

Defines the General Purpose I/O (GPIO) object.

## 9.2.2 Define Documentation

### 9.2.2.1 GPIO\_BASE\_ADDR

**Definition:**

```
#define GPIO_BASE_ADDR
```

**Description:**

Defines the base address of the general purpose I/O (GPIO) registers.

### 9.2.2.2 GPIO\_GPMUX\_CONFIG\_BITS

**Definition:**

```
#define GPIO_GPMUX_CONFIG_BITS
```

**Description:**

Defines the location of the CONFIG bits in the GPMUX register.

## 9.2.3 Typedef Documentation

### 9.2.3.1 GPIO\_Handle

**Definition:**

```
typedef struct GPIO_Obj *GPIO_Handle
```

**Description:**

Defines the general purpose I/O (GPIO) handle.

### 9.2.3.2 GPIO\_Obj

**Definition:**

```
typedef struct _GPIO_Obj_ GPIO_Obj
```

**Description:**

Defines the General Purpose I/O (GPIO) object.

## 9.2.4 Enumeration Documentation

### 9.2.4.1 GPIO\_Direction\_e

**Description:**

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

**Enumerators:**

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

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

## 9.2.4.2 GPIO\_Mode\_e

**Description:**

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

**Enumerators:**

**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\_EPWM4A** Denotes a EPWM4A 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\_EPWM4B** Denotes a EPWM4B 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\_SDA** Denotes a SDA function.  
**GPIO\_32\_Mode\_EPWMSYNCI** Denotes a EPWMSYNCI function.  
**GPIO\_32\_Mode\_ADCSOCBO\_NOT** Denotes a ADCSOCBO\_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.

#### 9.2.4.3 GPIO\_Number\_e

**Description:**

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

**Enumerators:**

**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.

#### 9.2.4.4 GPIO\_Port\_e

**Description:**

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

**Enumerators:**

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

#### 9.2.4.5 GPIO\_PullUp\_e

**Description:**

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

**Enumerators:**

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

#### 9.2.4.6 GPIO\_Qual\_e

**Description:**

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

**Enumerators:**

**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.

### 9.2.5 Function Documentation

#### 9.2.5.1 GPIO\_getData

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

**Prototype:**

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

**Parameters:**

- ← **gpioHandle** The general purpose I/O (GPIO) object handle
- ← **gpioNumber** The GPIO number

**Returns:**

The boolean state of a pin (high/low)

#### 9.2.5.2 uint16\_t GPIO\_getPortData (GPIO\_Handle gpioHandle, const GPIO\_Port\_e gpioPort)

Returns the data value present on a GPIO port.

**Parameters:**

- ← **gpioHandle** The general purpose I/O (GPIO) object handle
- ← **gpioPort** The GPIO port

**Returns:**

The data values for the specified port

#### 9.2.5.3 GPIO\_Handle GPIO\_init (void \* pMemory, const size\_t numBytes)

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

**Parameters:**

- ← **pMemory** A pointer to the base address of the GPIO registers
- ← **numBytes** The number of bytes allocated for the GPIO object, bytes

**Returns:**

The general purpose I/O (GPIO) object handle

#### 9.2.5.4 void GPIO\_lpmSelect (GPIO\_Handle gpioHandle, const GPIO\_Number\_e gpioNumber)

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

**Parameters:**

- ← **gpioHandle** The general purpose I/O (GPIO) object handle
- ← **gpioNumber** The GPIO number

9.2.5.5 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.

**Parameters:**

- ← **gpioHandle** The general purpose I/O (GPIO) object handle
- ← **gpioNumber** The GPIO number
- ← **direction** The signal direction

9.2.5.6 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.

**Parameters:**

- ← **gpioHandle** The general purpose I/O (GPIO) object handle
- ← **gpioNumber** The GPIO number
- ← **intNumber** The interrupt number

9.2.5.7 void GPIO\_setHigh (GPIO\_Handle gpioHandle, const GPIO\_Number\_e gpioNumber)

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

**Parameters:**

- ← **gpioHandle** The general purpose I/O (GPIO) object handle
- ← **gpioNumber** The GPIO number

9.2.5.8 void GPIO\_setLow (GPIO\_Handle gpioHandle, const GPIO\_Number\_e gpioNumber)

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

**Parameters:**

- ← **gpioHandle** The general purpose I/O (GPIO) object handle
- ← **gpioNumber** The GPIO number

9.2.5.9 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.

**Parameters:**

- ← **gpioHandle** The general purpose I/O (GPIO) object handle

- ← **gpioNumber** The GPIO number
- ← **mode** The mode

9.2.5.10 void GPIO\_setPortData (GPIO\_Handle gpioHandle, const GPIO\_Port\_e gpioPort, const uint16\_t data)

Sets data output on a given GPIO port.

**Parameters:**

- ← **gpioHandle** The general purpose I/O (GPIO) object handle
- ← **gpioPort** The GPIO number
- ← **data** The data to write to the port

9.2.5.11 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.

**Parameters:**

- ← **gpioHandle** The general purpose I/O (GPIO) object handle
- ← **gpioNumber** The GPIO number
- ← **pullUp** The pull up enable or disable

9.2.5.12 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).

**Parameters:**

- ← **gpioHandle** The general purpose I/O (GPIO) object handle
- ← **gpioNumber** The GPIO number
- ← **qualification** The desired input qualification

9.2.5.13 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).

**Parameters:**

- ← **gpioHandle** The general purpose I/O (GPIO) object handle
- ← **gpioNumber** The GPIO number
- ← **period** The desired input qualification period

9.2.5.14 void GPIO\_toggle (GPIO\_Handle gpioHandle, const GPIO\_Number\_e gpioNumber)

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

**Parameters:**

- ← **gpioHandle** The general purpose I/O (GPIO) object handle
- ← **gpioNumber** The GPIO number

## 10 Oscillator (OSC)

Introduction .....	127
OSC API Drivers .....	127

### 10.1 Introduction

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.

### 10.2 OSC

#### Data Structures

- [\\_OSC\\_Obj](#)

#### Defines

- [OSC\\_BASE\\_ADDR](#)
- [OSC\\_INTOSCnTRIM\\_COARSE\\_BITS](#)
- [OSC\\_INTOSCnTRIM\\_FINE\\_BITS](#)
- [OSC\\_OTP\\_COURSE\\_TRIM1](#)
- [OSC\\_OTP\\_COURSE\\_TRIM2](#)
- [OSC\\_OTP\\_FINE\\_TRIM\\_OFFSET1](#)
- [OSC\\_OTP\\_FINE\\_TRIM\\_OFFSET2](#)
- [OSC\\_OTP\\_FINE\\_TRIM\\_SLOPE1](#)
- [OSC\\_OTP\\_FINE\\_TRIM\\_SLOPE2](#)
- [OSC\\_OTP\\_REF\\_TEMP\\_OFFSET](#)

#### Enumerations

- [OSC\\_Number\\_e](#)
- [OSC\\_Osc2Src\\_e](#)
- [OSC\\_Src\\_e](#)

## 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)`

## 10.2.1 Data Structure Documentation

### 10.2.1.1 \_OSC\_Obj\_

**Definition:**

```
typedef struct
{
    uint16_t INTOSC1TRIM;
    uint16_t rsvd_1;
    uint16_t INTOSC2TRIM;
}
_OSC_Obj_
```

**Members:**

**INTOSC1TRIM** Internal Oscillator 1 Trim Register.

**rsvd\_1** Reserved.

**INTOSC2TRIM** Internal Oscillator 2 Trim Register.

**Description:**

Defines the oscillator (OSC) object.

## 10.2.2 Define Documentation

### 10.2.2.1 OSC\_BASE\_ADDR

**Definition:**

```
#define OSC_BASE_ADDR
```

**Description:**

Defines the base address of the oscillator (OSC) registers.



### 10.2.2.2 OSC\_INTOSCnTRIM\_COARSE\_BITS

**Definition:**

```
#define OSC_INTOSCnTRIM_COARSE_BITS
```

**Description:**

Defines the location of the COARSETRIM bits in the INTOSCnTRIM register.

### 10.2.2.3 OSC\_INTOSCnTRIM\_FINE\_BITS

**Definition:**

```
#define OSC_INTOSCnTRIM_FINE_BITS
```

**Description:**

Defines the location of the FINETRIM bits in the INTOSCnTRIM register.

### 10.2.2.4 OSC\_OTP\_COURSE\_TRIM1

**Definition:**

```
#define OSC_OTP_COURSE_TRIM1
```

**Description:**

Defines the address of the Course Trim 1 in OTP.

### 10.2.2.5 OSC\_OTP\_COURSE\_TRIM2

**Definition:**

```
#define OSC_OTP_COURSE_TRIM2
```

**Description:**

Defines the address of the Course Trim 1 in OTP.

### 10.2.2.6 OSC\_OTP\_FINE\_TRIM\_OFFSET1

**Definition:**

```
#define OSC_OTP_FINE_TRIM_OFFSET1
```

**Description:**

Defines the address of the Fine Trim Offset 1 in OTP.

### 10.2.2.7 OSC\_OTP\_FINE\_TRIM\_OFFSET2

**Definition:**

```
#define OSC_OTP_FINE_TRIM_OFFSET2
```

**Description:**

Defines the address of the Fine Trim Offset 1 in OTP.

#### 10.2.2.8 OSC\_OTP\_FINE\_TRIM\_SLOPE1

**Definition:**

```
#define OSC_OTP_FINE_TRIM_SLOPE1
```

**Description:**

Defines the address of the Fine Trim Slope 1 in OTP.

#### 10.2.2.9 OSC\_OTP\_FINE\_TRIM\_SLOPE2

**Definition:**

```
#define OSC_OTP_FINE_TRIM_SLOPE2
```

**Description:**

Defines the address of the Fine Trim Slope 1 in OTP.

#### 10.2.2.10 OSC\_OTP\_REF\_TEMP\_OFFSET

**Definition:**

```
#define OSC_OTP_REF_TEMP_OFFSET
```

**Description:**

Defines the address of the Reference Temp Offset in OTP.

### 10.2.3 Typedef Documentation

#### 10.2.3.1 OSC\_Handle

**Definition:**

```
typedef struct OSC_Obj *OSC_Handle
```

**Description:**

Defines the oscillator (OSC) handle.

#### 10.2.3.2 OSC\_Obj

**Definition:**

```
typedef struct _OSC_Obj_ OSC_Obj
```

**Description:**

Defines the oscillator (OSC) object.

## 10.2.4 Enumeration Documentation

### 10.2.4.1 OSC\_Number\_e

**Description:**

Enumeration to define the oscillator (OSC) number.

**Enumerators:**

***OSC\_Number\_1*** Denotes oscillator number 1.

***OSC\_Number\_2*** Denotes oscillator number 2.

### 10.2.4.2 OSC\_Osc2Src\_e

**Description:**

Enumeration to define the oscillator (OSC) 2 source.

**Enumerators:**

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

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

### 10.2.4.3 OSC\_Src\_e

**Description:**

Enumeration to define the oscillator (OSC) source.

**Enumerators:**

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

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

## 10.2.5 Function Documentation

### 10.2.5.1 OSC\_getCourseTrim1

Gets the fine trim offset for oscillator 1.

**Prototype:**

```
int16_t  
OSC_getCourseTrim1(OSC_Handle oscHandle) [inline]
```

**Parameters:**

← ***clkHandle*** The oscillator (OSC) object handle

**Returns:**

The fine trim offset for oscillator 1

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

Gets the fine trim offset for oscillator 2.

**Parameters:**

← **clkHandle** The oscillator (OSC) object handle

**Returns:**

The fine trim offset for oscillator 2

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

Gets the fine trim offset for oscillator 1.

**Parameters:**

← **clkHandle** The oscillator (OSC) object handle

**Returns:**

The fine trim offset for oscillator 1

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

Gets the fine trim offset for oscillator 2.

**Parameters:**

← **clkHandle** The oscillator (OSC) object handle

**Returns:**

The fine trim offset for oscillator 2

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

Gets the fine trim offset for oscillator 1.

**Parameters:**

← **clkHandle** The oscillator (OSC) object handle

**Returns:**

The fine trim offset for oscillator 1

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

Gets the fine trim offset for oscillator 2.

**Parameters:**

← **clkHandle** The oscillator (OSC) object handle

**Returns:**

The fine trim offset for oscillator 2

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

Gets the reference temperature offset.

**Parameters:**

← **clkHandle** The oscillator (OSC) object handle

**Returns:**

The reference temperature offset

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

Initializes the oscillator (OSC) handle.

**Parameters:**

← **pMemory** A pointer to the base address of the FLASH registers

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

**Returns:**

The flash (FLASH) object handle

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

Sets the coarse trim value for a specified oscillator.

**Parameters:**

← **oscHandle** The oscillator (OSC) object handle

← **oscNumber** The oscillator number

← **trimValue** The coarse trim value

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

Sets the fine trim value for a specified oscillator.

**Parameters:**

← **oscHandle** The oscillator (OSC) object handle

← **oscNumber** The oscillator number

← **trimValue** The fine trim value



# 11 Peripheral Interrupt Expansion Module (PIE)

Introduction .....	135
API Functions .....	135

## 11.1 Introduction

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.

## 11.2 PIE

### Data Structures

- [\\_PIE\\_IERIFR\\_t](#)
- [\\_PIE\\_Obj\\_](#)

### Defines

- [PIE\\_BASE\\_ADDR](#)
- [PIE\\_DBGIER\\_DLOGINT\\_BITS](#)
- [PIE\\_DBGIER\\_INT10\\_BITS](#)
- [PIE\\_DBGIER\\_INT11\\_BITS](#)
- [PIE\\_DBGIER\\_INT12\\_BITS](#)
- [PIE\\_DBGIER\\_INT13\\_BITS](#)
- [PIE\\_DBGIER\\_INT14\\_BITS](#)
- [PIE\\_DBGIER\\_INT1\\_BITS](#)
- [PIE\\_DBGIER\\_INT2\\_BITS](#)
- [PIE\\_DBGIER\\_INT3\\_BITS](#)
- [PIE\\_DBGIER\\_INT4\\_BITS](#)
- [PIE\\_DBGIER\\_INT5\\_BITS](#)
- [PIE\\_DBGIER\\_INT6\\_BITS](#)
- [PIE\\_DBGIER\\_INT7\\_BITS](#)
- [PIE\\_DBGIER\\_INT8\\_BITS](#)
- [PIE\\_DBGIER\\_INT9\\_BITS](#)
- [PIE\\_DBGIER\\_RTOSINT\\_BITS](#)
- [PIE\\_IER\\_DLOGINT\\_BITS](#)
- [PIE\\_IER\\_INT10\\_BITS](#)

- [PIE\\_IER\\_INT11\\_BITS](#)
- [PIE\\_IER\\_INT12\\_BITS](#)
- [PIE\\_IER\\_INT13\\_BITS](#)
- [PIE\\_IER\\_INT14\\_BITS](#)
- [PIE\\_IER\\_INT1\\_BITS](#)
- [PIE\\_IER\\_INT2\\_BITS](#)
- [PIE\\_IER\\_INT3\\_BITS](#)
- [PIE\\_IER\\_INT4\\_BITS](#)
- [PIE\\_IER\\_INT5\\_BITS](#)
- [PIE\\_IER\\_INT6\\_BITS](#)
- [PIE\\_IER\\_INT7\\_BITS](#)
- [PIE\\_IER\\_INT8\\_BITS](#)
- [PIE\\_IER\\_INT9\\_BITS](#)
- [PIE\\_IER\\_RTOSINT\\_BITS](#)
- [PIE\\_IERx\\_INTx1\\_BITS](#)
- [PIE\\_IERx\\_INTx2\\_BITS](#)
- [PIE\\_IERx\\_INTx3\\_BITS](#)
- [PIE\\_IERx\\_INTx4\\_BITS](#)
- [PIE\\_IERx\\_INTx5\\_BITS](#)
- [PIE\\_IERx\\_INTx6\\_BITS](#)
- [PIE\\_IERx\\_INTx7\\_BITS](#)
- [PIE\\_IERx\\_INTx8\\_BITS](#)
- [PIE\\_IFR\\_DLOGINT\\_BITS](#)
- [PIE\\_IFR\\_INT10\\_BITS](#)
- [PIE\\_IFR\\_INT11\\_BITS](#)
- [PIE\\_IFR\\_INT12\\_BITS](#)
- [PIE\\_IFR\\_INT13\\_BITS](#)
- [PIE\\_IFR\\_INT14\\_BITS](#)
- [PIE\\_IFR\\_INT1\\_BITS](#)
- [PIE\\_IFR\\_INT2\\_BITS](#)
- [PIE\\_IFR\\_INT3\\_BITS](#)
- [PIE\\_IFR\\_INT4\\_BITS](#)
- [PIE\\_IFR\\_INT5\\_BITS](#)
- [PIE\\_IFR\\_INT6\\_BITS](#)
- [PIE\\_IFR\\_INT7\\_BITS](#)
- [PIE\\_IFR\\_INT8\\_BITS](#)
- [PIE\\_IFR\\_INT9\\_BITS](#)
- [PIE\\_IFR\\_RTOSINT\\_BITS](#)
- [PIE\\_IFRx\\_INTx1\\_BITS](#)
- [PIE\\_IFRx\\_INTx2\\_BITS](#)
- [PIE\\_IFRx\\_INTx3\\_BITS](#)
- [PIE\\_IFRx\\_INTx4\\_BITS](#)
- [PIE\\_IFRx\\_INTx5\\_BITS](#)
- [PIE\\_IFRx\\_INTx6\\_BITS](#)
- [PIE\\_IFRx\\_INTx7\\_BITS](#)
- [PIE\\_IFRx\\_INTx8\\_BITS](#)



- [PIE\\_PIEACK\\_GROUP10\\_BITS](#)
- [PIE\\_PIEACK\\_GROUP11\\_BITS](#)
- [PIE\\_PIEACK\\_GROUP12\\_BITS](#)
- [PIE\\_PIEACK\\_GROUP1\\_BITS](#)
- [PIE\\_PIEACK\\_GROUP2\\_BITS](#)
- [PIE\\_PIEACK\\_GROUP3\\_BITS](#)
- [PIE\\_PIEACK\\_GROUP4\\_BITS](#)
- [PIE\\_PIEACK\\_GROUP5\\_BITS](#)
- [PIE\\_PIEACK\\_GROUP6\\_BITS](#)
- [PIE\\_PIEACK\\_GROUP7\\_BITS](#)
- [PIE\\_PIEACK\\_GROUP8\\_BITS](#)
- [PIE\\_PIEACK\\_GROUP9\\_BITS](#)
- [PIE\\_PIECTRL\\_ENPIE\\_BITS](#)
- [PIE\\_PIECTRL\\_PIEVECT\\_BITS](#)

## Enumerations

- [PIE\\_ExtIntPolarity\\_e](#)
- [PIE\\_GroupNumber\\_e](#)
- [PIE\\_InterruptSource\\_e](#)
- [PIE\\_SubGroupNumber\\_e](#)
- [PIE\\_SystemInterrupts\\_e](#)

## 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_illegallsr` (void)
- `PIE_Handle` `PIE_init` (void \*pMemory, const `size_t` numBytes)
- void `PIE_registerPieIntHandler` (`PIE_Handle` pieHandle, const `PIE_GroupNumber_e` groupNumber, const `PIE_SubGroupNumber_e` subGroupNumber, const `intVec_t` vector)
- void `PIE_registerSystemIntHandler` (`PIE_Handle` pieHandle, const `PIE_SystemInterrupts_e` systemInt, 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` groupNumber, const `PIE_SubGroupNumber_e` subGroupNumber)
- void `PIE_unregisterSystemIntHandler` (`PIE_Handle` pieHandle, const `PIE_SystemInterrupts_e` systemInt)

## 11.2.1 Data Structure Documentation

### 11.2.1.1 `_PIE_IERIFR_t`

**Definition:**

```
typedef struct
{
    uint16_t IER;
    uint16_t IFR;
}
_PIE_IERIFR_t
```

**Members:**

***IER*** the Interrupt Enable Register (IER)  
***IFR*** the Interrupt Flag Register (IFR)

**Description:**

Defines the `PIE_IERIFR_t` data type.

### 11.2.1.2 `_PIE_Obj_`

**Definition:**

```
typedef struct
{
    uint16_t PIELCTRL;
    uint16_t PIEACK;
    PIE_IERIFR_t PIEIER_PIEIFR[12];
    uint16_t rsvd_1[6];
    intVec_t Reset;
    intVec_t INT1;
```

```
intVec_t INT2;
intVec_t INT3;
intVec_t INT4;
intVec_t INT5;
intVec_t INT6;
intVec_t INT7;
intVec_t INT8;
intVec_t INT9;
intVec_t INT10;
intVec_t INT11;
intVec_t INT12;
intVec_t TINT1;
intVec_t TINT2;
intVec_t DATALOG;
intVec_t RTOSINT;
intVec_t EMUINT;
intVec_t NMI;
intVec_t ILLEGAL;
intVec_t USER1;
intVec_t USER2;
intVec_t USER3;
intVec_t USER4;
intVec_t USER5;
intVec_t USER6;
intVec_t USER7;
intVec_t USER8;
intVec_t USER9;
intVec_t USER10;
intVec_t USER11;
intVec_t USER12;
intVec_t rsvd1_1;
intVec_t rsvd1_2;
intVec_t rsvd1_3;
intVec_t XINT1;
intVec_t XINT2;
intVec_t ADCINT9;
intVec_t TINT0;
intVec_t WAKEINT;
intVec_t EPWM1_TZINT;
intVec_t EPWM2_TZINT;
intVec_t EPWM3_TZINT;
intVec_t EPWM4_TZINT;
intVec_t rsvd2_5;
intVec_t rsvd2_6;
intVec_t rsvd2_7;
intVec_t rsvd2_8;
intVec_t EPWM1_INT;
intVec_t EPWM2_INT;
intVec_t EPWM3_INT;
intVec_t EPWM4_INT;
intVec_t rsvd3_5;
intVec_t rsvd3_6;
```

```
intVec_t  rsvd3_7;
intVec_t  rsvd3_8;
intVec_t  ECAP1_INT;
intVec_t  rsvd4_2;
intVec_t  rsvd4_3;
intVec_t  rsvd4_4;
intVec_t  rsvd4_5;
intVec_t  rsvd4_6;
intVec_t  rsvd4_7;
intVec_t  rsvd4_8;
intVec_t  rsvd5_1;
intVec_t  rsvd5_2;
intVec_t  rsvd5_3;
intVec_t  rsvd5_4;
intVec_t  rsvd5_5;
intVec_t  rsvd5_6;
intVec_t  rsvd5_7;
intVec_t  rsvd5_8;
intVec_t  SPIRXINTA;
intVec_t  SPITXINTA;
intVec_t  rsvd6_3;
intVec_t  rsvd6_4;
intVec_t  rsvd6_5;
intVec_t  rsvd6_6;
intVec_t  rsvd6_7;
intVec_t  rsvd6_8;
intVec_t  rsvd7_1;
intVec_t  rsvd7_2;
intVec_t  rsvd7_3;
intVec_t  rsvd7_4;
intVec_t  rsvd7_5;
intVec_t  rsvd7_6;
intVec_t  rsvd7_7;
intVec_t  rsvd7_8;
intVec_t  I2CINT1A;
intVec_t  I2CINT2A;
intVec_t  rsvd8_3;
intVec_t  rsvd8_4;
intVec_t  rsvd8_5;
intVec_t  rsvd8_6;
intVec_t  rsvd8_7;
intVec_t  rsvd8_8;
intVec_t  SCIRXINTA;
intVec_t  SCITXINTA;
intVec_t  rsvd9_3;
intVec_t  rsvd9_4;
intVec_t  rsvd9_5;
intVec_t  rsvd9_6;
intVec_t  rsvd9_7;
intVec_t  rsvd9_8;
intVec_t  ADCINT1;
intVec_t  ADCINT2;
```

```
intVec_t ADCINT3;  
intVec_t ADCINT4;  
intVec_t ADCINT5;  
intVec_t ADCINT6;  
intVec_t ADCINT7;  
intVec_t ADCINT8;  
intVec_t rsvd11_1;  
intVec_t rsvd11_2;  
intVec_t rsvd11_3;  
intVec_t rsvd11_4;  
intVec_t rsvd11_5;  
intVec_t rsvd11_6;  
intVec_t rsvd11_7;  
intVec_t rsvd11_8;  
intVec_t XINT3;  
intVec_t rsvd12_2;  
intVec_t rsvd12_3;  
intVec_t rsvd12_4;  
intVec_t rsvd12_5;  
intVec_t rsvd12_6;  
intVec_t rsvd12_7;  
intVec_t rsvd12_8;  
uint16_t rsvd13[25200];  
uint16_t XINTnCR[3];  
uint16_t rsvd14[5];  
uint16_t XINTnCTR[3];  
}  
_PIE_Obj_
```

**Members:**

**PIECTRL** PIE Control Register.

**PIEACK** PIE Acknowledge Register.

**PIEIER\_PIEIFR** PIE Interrupt Enable Register and PIE Interrupt Flag Register.

**rsvd\_1** Reserved.

**Reset** Reset interrupt vector.

**INT1** INT1 interrupt vector.

**INT2** INT2 interrupt vector.

**INT3** INT3 interrupt vector.

**INT4** INT4 interrupt vector.

**INT5** INT5 interrupt vector.

**INT6** INT6 interrupt vector.

**INT7** INT7 interrupt vector.

**INT8** INT8 interrupt vector.

**INT9** INT9 interrupt vector.

**INT10** INT10 interrupt vector.

**INT11** INT11 interrupt vector.

**INT12** INT12 interrupt vector.

**TINT1** INT13 interrupt vector.

**TINT2** INT14 interrupt vector.

**DATALOG** DATALOG interrupt vector.

**RTOSINT** RTOSINT interrupt vector.  
**EMUINT** EMUINT interrupt vector.  
**NMI** NMI interrupt vector.  
**ILLEGAL** ILLEGAL interrupt vector.  
**USER1** USER1 interrupt vector.  
**USER2** USER2 interrupt vector.  
**USER3** USER3 interrupt vector.  
**USER4** USER4 interrupt vector.  
**USER5** USER5 interrupt vector.  
**USER6** USER6 interrupt vector.  
**USER7** USER7 interrupt vector.  
**USER8** USER8 interrupt vector.  
**USER9** USER9 interrupt vector.  
**USER10** USER10 interrupt vector.  
**USER11** USER11 interrupt vector.  
**USER12** USER12 interrupt vector.  
**rsvd1\_1** Reserved (Note: using ADCINT\_1 in group 10).  
**rsvd1\_2** Reserved (Note: using ADCINT\_2 in group 10).  
**rsvd1\_3** Reserved.  
**XINT1** XINT1 interrupt vector.  
**XINT2** XINT2 interrupt vector.  
**ADCINT9** ADCINT9 interrupt vector.  
**TINT0** TINT0 interrupt vector.  
**WAKEINT** WAKEINT interrupt vector.  
**EPWM1\_TZINT** EPWM1\_TZINT interrupt vector.  
**EPWM2\_TZINT** EPWM2\_TZINT interrupt vector.  
**EPWM3\_TZINT** EPWM3\_TZINT interrupt vector.  
**EPWM4\_TZINT** EPWM4\_TZINT interrupt vector.  
**rsvd2\_5** Reserved.  
**rsvd2\_6** Reserved.  
**rsvd2\_7** Reserved.  
**rsvd2\_8** Reserved.  
**EPWM1\_INT** EPWM1 interrupt vector.  
**EPWM2\_INT** EPWM2 interrupt vector.  
**EPWM3\_INT** EPWM3 interrupt vector.  
**EPWM4\_INT** EPWM4 interrupt vector.  
**rsvd3\_5** Reserved.  
**rsvd3\_6** Reserved.  
**rsvd3\_7** Reserved.  
**rsvd3\_8** Reserved.  
**ECAP1\_INT** ECAP1\_INT interrupt vector.  
**rsvd4\_2** Reserved.  
**rsvd4\_3** Reserved.  
**rsvd4\_4** Reserved.  
**rsvd4\_5** Reserved.  
**rsvd4\_6** Reserved.

*rsvd4\_7* Reserved.  
*rsvd4\_8* Reserved.  
*rsvd5\_1* Reserved.  
*rsvd5\_2* Reserved.  
*rsvd5\_3* Reserved.  
*rsvd5\_4* Reserved.  
*rsvd5\_5* Reserved.  
*rsvd5\_6* Reserved.  
*rsvd5\_7* Reserved.  
*rsvd5\_8* Reserved.  
**SPIRXINTA** SPIRXINTA interrupt vector.  
**SPITXINTA** SPITXINTA interrupt vector.  
*rsvd6\_3* Reserved.  
*rsvd6\_4* Reserved.  
*rsvd6\_5* Reserved.  
*rsvd6\_6* Reserved.  
*rsvd6\_7* Reserved.  
*rsvd6\_8* Reserved.  
*rsvd7\_1* Reserved.  
*rsvd7\_2* Reserved.  
*rsvd7\_3* Reserved.  
*rsvd7\_4* Reserved.  
*rsvd7\_5* Reserved.  
*rsvd7\_6* Reserved.  
*rsvd7\_7* Reserved.  
*rsvd7\_8* Reserved.  
**I2CINT1A** I2CINT1A interrupt vector.  
**I2CINT2A** I2CINT2A interrupt vector.  
*rsvd8\_3* Reserved.  
*rsvd8\_4* Reserved.  
*rsvd8\_5* Reserved.  
*rsvd8\_6* Reserved.  
*rsvd8\_7* Reserved.  
*rsvd8\_8* Reserved.  
**SCIRXINTA** SCIRXINTA interrupt vector.  
**SCITXINTA** SCITXINTA interrupt vector.  
*rsvd9\_3* Reserved.  
*rsvd9\_4* Reserved.  
*rsvd9\_5* Reserved.  
*rsvd9\_6* Reserved.  
*rsvd9\_7* Reserved.  
*rsvd9\_8* Reserved.  
**ADCINT1** ADCINT1 interrupt vector.  
**ADCINT2** ADCINT2 interrupt vector.  
**ADCINT3** ADCINT3 interrupt vector.  
**ADCINT4** ADCINT4 interrupt vector.

**ADCINT5** ADCINT5 interrupt vector.  
**ADCINT6** ADCINT6 interrupt vector.  
**ADCINT7** ADCINT7 interrupt vector.  
**ADCINT8** ADCINT8 interrupt vector.  
**rsvd11\_1** Reserved.  
**rsvd11\_2** Reserved.  
**rsvd11\_3** Reserved.  
**rsvd11\_4** Reserved.  
**rsvd11\_5** Reserved.  
**rsvd11\_6** Reserved.  
**rsvd11\_7** Reserved.  
**rsvd11\_8** Reserved.  
**XINT3** XINT3 interrupt vector.  
**rsvd12\_2** Reserved.  
**rsvd12\_3** Reserved.  
**rsvd12\_4** Reserved.  
**rsvd12\_5** Reserved.  
**rsvd12\_6** Reserved.  
**rsvd12\_7** Reserved.  
**rsvd12\_8** Reserved.  
**rsvd13** Reserved.  
**XINTnCR** External Interrupt n Control Register.  
**rsvd14** Reserved.  
**XINTnCTR** External Interrupt n Counter Register.

**Description:**

Defines the peripheral interrupt expansion (PIE) object.

## 11.2.2 Define Documentation

### 11.2.2.1 PIE\_BASE\_ADDR

**Definition:**

```
#define PIE_BASE_ADDR
```

**Description:**

Defines the base address of the peripheral interrupt expansion (PIE) registers.

### 11.2.2.2 PIE\_DBGIER\_DLOGINT\_BITS

**Definition:**

```
#define PIE_DBGIER_DLOGINT_BITS
```

**Description:**

Defines the location of the DLOGINT bits in the DBGIER register.



### 11.2.2.3 PIE\_DBGIER\_INT10\_BITS

**Definition:**

```
#define PIE_DBGIER_INT10_BITS
```

**Description:**

Defines the location of the INT10 bits in the DBGIER register.

### 11.2.2.4 PIE\_DBGIER\_INT11\_BITS

**Definition:**

```
#define PIE_DBGIER_INT11_BITS
```

**Description:**

Defines the location of the INT11 bits in the DBGIER register.

### 11.2.2.5 PIE\_DBGIER\_INT12\_BITS

**Definition:**

```
#define PIE_DBGIER_INT12_BITS
```

**Description:**

Defines the location of the INT12 bits in the DBGIER register.

### 11.2.2.6 PIE\_DBGIER\_INT13\_BITS

**Definition:**

```
#define PIE_DBGIER_INT13_BITS
```

**Description:**

Defines the location of the INT13 bits in the DBGIER register.

### 11.2.2.7 PIE\_DBGIER\_INT14\_BITS

**Definition:**

```
#define PIE_DBGIER_INT14_BITS
```

**Description:**

Defines the location of the INT14 bits in the DBGIER register.

### 11.2.2.8 PIE\_DBGIER\_INT1\_BITS

**Definition:**

```
#define PIE_DBGIER_INT1_BITS
```

**Description:**

Defines the location of the INT1 bits in the DBGIER register.

#### 11.2.2.9 PIE\_DBGIER\_INT2\_BITS

**Definition:**

```
#define PIE_DBGIER_INT2_BITS
```

**Description:**

Defines the location of the INT2 bits in the DBGIER register.

#### 11.2.2.10 PIE\_DBGIER\_INT3\_BITS

**Definition:**

```
#define PIE_DBGIER_INT3_BITS
```

**Description:**

Defines the location of the INT3 bits in the DBGIER register.

#### 11.2.2.11 PIE\_DBGIER\_INT4\_BITS

**Definition:**

```
#define PIE_DBGIER_INT4_BITS
```

**Description:**

Defines the location of the INT4 bits in the DBGIER register.

#### 11.2.2.12 PIE\_DBGIER\_INT5\_BITS

**Definition:**

```
#define PIE_DBGIER_INT5_BITS
```

**Description:**

Defines the location of the INT5 bits in the DBGIER register.

#### 11.2.2.13 PIE\_DBGIER\_INT6\_BITS

**Definition:**

```
#define PIE_DBGIER_INT6_BITS
```

**Description:**

Defines the location of the INT6 bits in the DBGIER register.

#### 11.2.2.14 PIE\_DBGIER\_INT7\_BITS

**Definition:**

```
#define PIE_DBGIER_INT7_BITS
```

**Description:**

Defines the location of the INT7 bits in the DBGIER register.

### 11.2.2.15 PIE\_DBGIER\_INT8\_BITS

**Definition:**

```
#define PIE_DBGIER_INT8_BITS
```

**Description:**

Defines the location of the INT8 bits in the DBGIER register.

### 11.2.2.16 PIE\_DBGIER\_INT9\_BITS

**Definition:**

```
#define PIE_DBGIER_INT9_BITS
```

**Description:**

Defines the location of the INT9 bits in the DBGIER register.

### 11.2.2.17 PIE\_DBGIER\_RTOSINT\_BITS

**Definition:**

```
#define PIE_DBGIER_RTOSINT_BITS
```

**Description:**

Defines the location of the RTOSINT bits in the DBGIER register.

### 11.2.2.18 PIE\_IER\_DLOGINT\_BITS

**Definition:**

```
#define PIE_IER_DLOGINT_BITS
```

**Description:**

Defines the location of the DLOGINT bits in the IER register.

### 11.2.2.19 PIE\_IER\_INT10\_BITS

**Definition:**

```
#define PIE_IER_INT10_BITS
```

**Description:**

Defines the location of the INT10 bits in the IER register.

### 11.2.2.20 PIE\_IER\_INT11\_BITS

**Definition:**

```
#define PIE_IER_INT11_BITS
```

**Description:**

Defines the location of the INT11 bits in the IER register.

#### 11.2.2.21 PIE\_IER\_INT12\_BITS

**Definition:**

```
#define PIE_IER_INT12_BITS
```

**Description:**

Defines the location of the INT12 bits in the IER register.

#### 11.2.2.22 PIE\_IER\_INT13\_BITS

**Definition:**

```
#define PIE_IER_INT13_BITS
```

**Description:**

Defines the location of the INT13 bits in the IER register.

#### 11.2.2.23 PIE\_IER\_INT14\_BITS

**Definition:**

```
#define PIE_IER_INT14_BITS
```

**Description:**

Defines the location of the INT14 bits in the IER register.

#### 11.2.2.24 PIE\_IER\_INT1\_BITS

**Definition:**

```
#define PIE_IER_INT1_BITS
```

**Description:**

Defines the location of the INT1 bits in the IER register.

#### 11.2.2.25 PIE\_IER\_INT2\_BITS

**Definition:**

```
#define PIE_IER_INT2_BITS
```

**Description:**

Defines the location of the INT2 bits in the IER register.

#### 11.2.2.26 PIE\_IER\_INT3\_BITS

**Definition:**

```
#define PIE_IER_INT3_BITS
```

**Description:**

Defines the location of the INT3 bits in the IER register.

#### 11.2.2.27 PIE\_IER\_INT4\_BITS

**Definition:**

```
#define PIE_IER_INT4_BITS
```

**Description:**

Defines the location of the INT4 bits in the IER register.

#### 11.2.2.28 PIE\_IER\_INT5\_BITS

**Definition:**

```
#define PIE_IER_INT5_BITS
```

**Description:**

Defines the location of the INT5 bits in the IER register.

#### 11.2.2.29 PIE\_IER\_INT6\_BITS

**Definition:**

```
#define PIE_IER_INT6_BITS
```

**Description:**

Defines the location of the INT6 bits in the IER register.

#### 11.2.2.30 PIE\_IER\_INT7\_BITS

**Definition:**

```
#define PIE_IER_INT7_BITS
```

**Description:**

Defines the location of the INT7 bits in the IER register.

#### 11.2.2.31 PIE\_IER\_INT8\_BITS

**Definition:**

```
#define PIE_IER_INT8_BITS
```

**Description:**

Defines the location of the INT8 bits in the IER register.

#### 11.2.2.32 PIE\_IER\_INT9\_BITS

**Definition:**

```
#define PIE_IER_INT9_BITS
```

**Description:**

Defines the location of the INT9 bits in the IER register.

#### 11.2.2.33 PIE\_IER\_RTOSINT\_BITS

**Definition:**

```
#define PIE_IER_RTOSINT_BITS
```

**Description:**

Defines the location of the RTOSINT bits in the IER register.

#### 11.2.2.34 PIE\_IERx\_INTx1\_BITS

**Definition:**

```
#define PIE_IERx_INTx1_BITS
```

**Description:**

Defines the location of the INTx1 bits in the IERx register.

#### 11.2.2.35 PIE\_IERx\_INTx2\_BITS

**Definition:**

```
#define PIE_IERx_INTx2_BITS
```

**Description:**

Defines the location of the INTx2 bits in the IERx register.

#### 11.2.2.36 PIE\_IERx\_INTx3\_BITS

**Definition:**

```
#define PIE_IERx_INTx3_BITS
```

**Description:**

Defines the location of the INTx3 bits in the IERx register.

#### 11.2.2.37 PIE\_IERx\_INTx4\_BITS

**Definition:**

```
#define PIE_IERx_INTx4_BITS
```

**Description:**

Defines the location of the INTx4 bits in the IERx register.

#### 11.2.2.38 PIE\_IERx\_INTx5\_BITS

**Definition:**

```
#define PIE_IERx_INTx5_BITS
```

**Description:**

Defines the location of the INTx5 bits in the IERx register.

#### 11.2.2.39 PIE\_IERx\_INTx6\_BITS

**Definition:**

```
#define PIE_IERx_INTx6_BITS
```

**Description:**

Defines the location of the INTx6 bits in the IERx register.

#### 11.2.2.40 PIE\_IERx\_INTx7\_BITS

**Definition:**

```
#define PIE_IERx_INTx7_BITS
```

**Description:**

Defines the location of the INTx7 bits in the IERx register.

#### 11.2.2.41 PIE\_IERx\_INTx8\_BITS

**Definition:**

```
#define PIE_IERx_INTx8_BITS
```

**Description:**

Defines the location of the INTx8 bits in the IERx register.

#### 11.2.2.42 PIE\_IFR\_DLOGINT\_BITS

**Definition:**

```
#define PIE_IFR_DLOGINT_BITS
```

**Description:**

Defines the location of the DLOGINT bits in the IFR register.

#### 11.2.2.43 PIE\_IFR\_INT10\_BITS

**Definition:**

```
#define PIE_IFR_INT10_BITS
```

**Description:**

Defines the location of the INT10 bits in the IFR register.

#### 11.2.2.44 PIE\_IFR\_INT11\_BITS

**Definition:**

```
#define PIE_IFR_INT11_BITS
```

**Description:**

Defines the location of the INT11 bits in the IFR register.

#### 11.2.2.45 PIE\_IFR\_INT12\_BITS

**Definition:**

```
#define PIE_IFR_INT12_BITS
```

**Description:**

Defines the location of the INT12 bits in the IFR register.

#### 11.2.2.46 PIE\_IFR\_INT13\_BITS

**Definition:**

```
#define PIE_IFR_INT13_BITS
```

**Description:**

Defines the location of the INT13 bits in the IFR register.

#### 11.2.2.47 PIE\_IFR\_INT14\_BITS

**Definition:**

```
#define PIE_IFR_INT14_BITS
```

**Description:**

Defines the location of the INT14 bits in the IFR register.

#### 11.2.2.48 PIE\_IFR\_INT1\_BITS

**Definition:**

```
#define PIE_IFR_INT1_BITS
```

**Description:**

Defines the location of the INT1 bits in the IFR register.

#### 11.2.2.49 PIE\_IFR\_INT2\_BITS

**Definition:**

```
#define PIE_IFR_INT2_BITS
```

**Description:**

Defines the location of the INT2 bits in the IFR register.

#### 11.2.2.50 PIE\_IFR\_INT3\_BITS

**Definition:**

```
#define PIE_IFR_INT3_BITS
```

**Description:**

Defines the location of the INT3 bits in the IFR register.



#### 11.2.2.51 PIE\_IFR\_INT4\_BITS

**Definition:**

```
#define PIE_IFR_INT4_BITS
```

**Description:**

Defines the location of the INT4 bits in the IFR register.

#### 11.2.2.52 PIE\_IFR\_INT5\_BITS

**Definition:**

```
#define PIE_IFR_INT5_BITS
```

**Description:**

Defines the location of the INT5 bits in the IFR register.

#### 11.2.2.53 PIE\_IFR\_INT6\_BITS

**Definition:**

```
#define PIE_IFR_INT6_BITS
```

**Description:**

Defines the location of the INT6 bits in the IFR register.

#### 11.2.2.54 PIE\_IFR\_INT7\_BITS

**Definition:**

```
#define PIE_IFR_INT7_BITS
```

**Description:**

Defines the location of the INT7 bits in the IFR register.

#### 11.2.2.55 PIE\_IFR\_INT8\_BITS

**Definition:**

```
#define PIE_IFR_INT8_BITS
```

**Description:**

Defines the location of the INT8 bits in the IFR register.

#### 11.2.2.56 PIE\_IFR\_INT9\_BITS

**Definition:**

```
#define PIE_IFR_INT9_BITS
```

**Description:**

Defines the location of the INT9 bits in the IFR register.

#### 11.2.2.57 PIE\_IFR\_RTOSINT\_BITS

**Definition:**

```
#define PIE_IFR_RTOSINT_BITS
```

**Description:**

Defines the location of the RTOSINT bits in the IFR register.

#### 11.2.2.58 PIE\_IFRx\_INTx1\_BITS

**Definition:**

```
#define PIE_IFRx_INTx1_BITS
```

**Description:**

Defines the location of the INTx1 bits in the IFRx register.

#### 11.2.2.59 PIE\_IFRx\_INTx2\_BITS

**Definition:**

```
#define PIE_IFRx_INTx2_BITS
```

**Description:**

Defines the location of the INTx2 bits in the IFRx register.

#### 11.2.2.60 PIE\_IFRx\_INTx3\_BITS

**Definition:**

```
#define PIE_IFRx_INTx3_BITS
```

**Description:**

Defines the location of the INTx3 bits in the IFRx register.

#### 11.2.2.61 PIE\_IFRx\_INTx4\_BITS

**Definition:**

```
#define PIE_IFRx_INTx4_BITS
```

**Description:**

Defines the location of the INTx4 bits in the IFRx register.

#### 11.2.2.62 PIE\_IFRx\_INTx5\_BITS

**Definition:**

```
#define PIE_IFRx_INTx5_BITS
```

**Description:**

Defines the location of the INTx5 bits in the IFRx register.

#### 11.2.2.63 PIE\_IFRx\_INTx6\_BITS

**Definition:**

```
#define PIE_IFRx_INTx6_BITS
```

**Description:**

Defines the location of the INTx6 bits in the IFRx register.

#### 11.2.2.64 PIE\_IFRx\_INTx7\_BITS

**Definition:**

```
#define PIE_IFRx_INTx7_BITS
```

**Description:**

Defines the location of the INTx7 bits in the IFRx register.

#### 11.2.2.65 PIE\_IFRx\_INTx8\_BITS

**Definition:**

```
#define PIE_IFRx_INTx8_BITS
```

**Description:**

Defines the location of the INTx8 bits in the IFRx register.

#### 11.2.2.66 PIE\_PIEACK\_GROUP10\_BITS

**Definition:**

```
#define PIE_PIEACK_GROUP10_BITS
```

**Description:**

Defines the location of the GROUP10 bits in the PIEACK register.

#### 11.2.2.67 PIE\_PIEACK\_GROUP11\_BITS

**Definition:**

```
#define PIE_PIEACK_GROUP11_BITS
```

**Description:**

Defines the location of the GROUP11 bits in the PIEACK register.

#### 11.2.2.68 PIE\_PIEACK\_GROUP12\_BITS

**Definition:**

```
#define PIE_PIEACK_GROUP12_BITS
```

**Description:**

Defines the location of the GROUP12 bits in the PIEACK register.

#### 11.2.2.69 PIE\_PIEACK\_GROUP1\_BITS

**Definition:**

```
#define PIE_PIEACK_GROUP1_BITS
```

**Description:**

Defines the location of the GROUP1 bits in the PIEACK register.

#### 11.2.2.70 PIE\_PIEACK\_GROUP2\_BITS

**Definition:**

```
#define PIE_PIEACK_GROUP2_BITS
```

**Description:**

Defines the location of the GROUP2 bits in the PIEACK register.

#### 11.2.2.71 PIE\_PIEACK\_GROUP3\_BITS

**Definition:**

```
#define PIE_PIEACK_GROUP3_BITS
```

**Description:**

Defines the location of the GROUP3 bits in the PIEACK register.

#### 11.2.2.72 PIE\_PIEACK\_GROUP4\_BITS

**Definition:**

```
#define PIE_PIEACK_GROUP4_BITS
```

**Description:**

Defines the location of the GROUP4 bits in the PIEACK register.

#### 11.2.2.73 PIE\_PIEACK\_GROUP5\_BITS

**Definition:**

```
#define PIE_PIEACK_GROUP5_BITS
```

**Description:**

Defines the location of the GROUP5 bits in the PIEACK register.

#### 11.2.2.74 PIE\_PIEACK\_GROUP6\_BITS

**Definition:**

```
#define PIE_PIEACK_GROUP6_BITS
```

**Description:**

Defines the location of the GROUP6 bits in the PIEACK register.

### 11.2.2.75 PIE\_PIEACK\_GROUP7\_BITS

**Definition:**

```
#define PIE_PIEACK_GROUP7_BITS
```

**Description:**

Defines the location of the GROUP7 bits in the PIEACK register.

### 11.2.2.76 PIE\_PIEACK\_GROUP8\_BITS

**Definition:**

```
#define PIE_PIEACK_GROUP8_BITS
```

**Description:**

Defines the location of the GROUP8 bits in the PIEACK register.

### 11.2.2.77 PIE\_PIEACK\_GROUP9\_BITS

**Definition:**

```
#define PIE_PIEACK_GROUP9_BITS
```

**Description:**

Defines the location of the GROUP9 bits in the PIEACK register.

### 11.2.2.78 PIE\_PIECTRL\_ENPIE\_BITS

**Definition:**

```
#define PIE_PIECTRL_ENPIE_BITS
```

**Description:**

Defines the location of the ENPIE bits in the PIECTRL register.

### 11.2.2.79 PIE\_PIECTRL\_PIEVECT\_BITS

**Definition:**

```
#define PIE_PIECTRL_PIEVECT_BITS
```

**Description:**

Defines the location of the PIEVECT bits in the PIECTRL register.

## 11.2.3 Typedef Documentation

### 11.2.3.1 intVec\_t

**Definition:**

```
typedef interrupt void(*) intVec_t (void)
```

**Description:**

Defines the type for an interrupt vector.

### 11.2.3.2 PIE\_Handle

**Definition:**

```
typedef struct PIE\_Obj *PIE_Handle
```

**Description:**

Defines the peripheral interrupt expansion (PIE) handle.

### 11.2.3.3 PIE\_IERIFR\_t

**Definition:**

```
typedef struct \_PIE\_IERIFR\_t PIE\_IERIFR\_t
```

**Description:**

Defines the PIE\_IERIFR\_t data type.

### 11.2.3.4 PIE\_Obj

**Definition:**

```
typedef struct \_PIE\_Obj\_ PIE\_Obj
```

**Description:**

Defines the peripheral interrupt expansion (PIE) object.

## 11.2.4 Enumeration Documentation

### 11.2.4.1 PIE\_ExtIntPolarity\_e

**Description:**

Enumeration to define the external interrupt polarity.

**Enumerators:**

***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.

### 11.2.4.2 PIE\_GroupNumber\_e

**Description:**

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

**Enumerators:**

***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.

### 11.2.4.3 `PIE_InterruptSource_e`

**Description:**

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

**Enumerators:**

***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\_TZ4*** EPWM TZ4 Interrupt.  
***PIE\_InterruptSource\_EPWM1*** EPWM 1 Interrupt.  
***PIE\_InterruptSource\_EPWM2*** EPWM 2 Interrupt.  
***PIE\_InterruptSource\_EPWM3*** EPWM 3 Interrupt.  
***PIE\_InterruptSource\_EPWM4*** EPWM 4 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.

#### 11.2.4.4 **PIE\_SubGroupNumber\_e**

**Description:**

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

**Enumerators:**

***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.

#### 11.2.4.5 **PIE\_SystemInterrupts\_e**

**Description:**

Enumeration to define the system interrupts.

**Enumerators:**

***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.

## 11.2.5 Function Documentation

### 11.2.5.1 PIE\_clearAllFlags

Clears all the interrupt flags.

**Prototype:**

```
void  
PIE_clearAllFlags(PIE\_Handle pieHandle)
```

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) object handle

### 11.2.5.2 void PIE\_clearAllInts ([PIE\\_Handle](#) pieHandle)

Clears all the interrupts.

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) object handle

### 11.2.5.3 void PIE\_clearInt ([PIE\\_Handle](#) pieHandle, const [PIE\\_GroupNumber\\_e](#) groupNumber) [inline]

Clears an interrupt defined by group number.

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) object handle

← **groupNumber** The group number

#### 11.2.5.4 void PIE\_disable (PIE\_Handle pieHandle)

Disables the peripheral interrupt expansion (PIE).

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) object handle

#### 11.2.5.5 void PIE\_disableAllInts (PIE\_Handle pieHandle)

Disables all of the interrupts.

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) object handle

#### 11.2.5.6 void PIE\_disableCaptureInt (PIE\_Handle pieHandle)

Disables the capture interrupt.

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) object handle

#### 11.2.5.7 void PIE\_disableInt (PIE\_Handle pieHandle, const PIE\_GroupNumber\_e group, const PIE\_InterruptSource\_e intSource)

Disable a specific PIE interrupt.

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) object handle

← **group** The PIE group an interrupt belongs to

← **intSource** The specific interrupt source to disable

#### 11.2.5.8 void PIE\_enable (PIE\_Handle pieHandle)

Enables the peripheral interrupt expansion (PIE).

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) object handle

#### 11.2.5.9 void PIE\_enableAdcInt (PIE\_Handle pieHandle, const ADC\_IntNumber\_e intNumber)

Enables the specified ADC interrupt.

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) object handle

← **intNumber** The interrupt number

#### 11.2.5.10 void PIE\_enableCaptureInt (PIE\_Handle pieHandle)

Enables the capture interrupt.

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) object handle

#### 11.2.5.11 void PIE\_enableExtInt (PIE\_Handle pieHandle, const CPU\_ExtIntNumber\_e intNumber)

Enables the prescribed external interrupt.

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) handle

← **intNumber** The interrupt number

#### 11.2.5.12 void PIE\_enableInt (PIE\_Handle pieHandle, const PIE\_GroupNumber\_e group, const PIE\_InterruptSource\_e intSource)

Enable a specific PIE interrupt.

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) object handle

← **group** The PIE group an interrupt belongs to

← **intSource** The specific interrupt source to enable

#### 11.2.5.13 void PIE\_enablePwmInt (PIE\_Handle pieHandle, const PWM\_Number\_e pwmNumber)

Enables the PWM interrupt.

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) handle

← **pwmNumber** The PWM number

#### 11.2.5.14 void PIE\_enablePwmTzInt (PIE\_Handle pieHandle, const PWM\_Number\_e pwmNumber)

Enables the PWM Trip Zone interrupt.

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) handle

← **pwmNumber** The PWM number

#### 11.2.5.15 void PIE\_enableTimer0Int (PIE\_Handle pieHandle)

Enables the Cpu Timer 0 interrupt.

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) handle

#### 11.2.5.16 void PIE\_forceInt (PIE\_Handle pieHandle, const PIE\_GroupNumber\_e group, const PIE\_InterruptSource\_e intSource)

Force a specific PIE interrupt.

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) object handle

← **group** The PIE group an interrupt belongs to

← **intSource** The specific interrupt source to force

#### 11.2.5.17 uint16\_t PIE\_getExtIntCount (PIE\_Handle pieHandle, const CPU\_ExtIntNumber\_e intNumber)

Gets the external interrupt count value.

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) handle

← **intNumber** The external interrupt number

**Returns:**

The count value

#### 11.2.5.18 uint16\_t PIE\_getIntEnables (PIE\_Handle pieHandle, const PIE\_GroupNumber\_e group)

Gets PIE interrupt enable values.

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) object handle

← **group** The PIE group the flags belong to

#### 11.2.5.19 uint16\_t PIE\_getIntFlags (PIE\_Handle pieHandle, const PIE\_GroupNumber\_e group)

Gets PIE interrupt flag values.

**Parameters:**

← **pieHandle** The peripheral interrupt expansion (PIE) object handle

← **group** The PIE group the flags belong to

#### 11.2.5.20 interrupt void PIE\_illegalIsr (void)

Defines an illegal interrupt service routine - if the program pointer references this function, there is an incorrect mapping in the PIE interrupt table.

#### 11.2.5.21 `PIE_Handle` PIE\_init (void \* *pMemory*, const size\_t *numBytes*)

Initializes the peripheral interrupt expansion (PIE) object handle.

**Parameters:**

- ← ***pMemory*** A pointer to the memory for the PIE object
- ← ***numBytes*** The number of bytes allocated for the PIE object, bytes

**Returns:**

The peripheral interrupt expansion (PIE) object handle

#### 11.2.5.22 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.

**Parameters:**

- ← ***pieHandle*** The peripheral interrupt expansion (PIE) object handle
- ← ***groupNumber*** The PIE group an interrupt belongs to
- ← ***subGroupNumber*** The PIE subgroup an interrupt belongs to
- ← ***vector*** The specific interrupt handler

#### 11.2.5.23 void PIE\_registerSystemIntHandler (`PIE_Handle` *pieHandle*, const `PIE_SystemInterrupts_e` *systemInt*, const `intVec_t` *vector*)

Registers a handler for a PIE interrupt.

**Parameters:**

- ← ***pieHandle*** The peripheral interrupt expansion (PIE) object handle
- ← ***systemInt*** The system interrupt to register this handler to
- ← ***vector*** The specific interrupt handler

#### 11.2.5.24 void PIE\_setDefaultIntVectorTable (`PIE_Handle` *pieHandle*)

Initializes the vector table with illegal ISR handlers.

**Parameters:**

- ← ***pieHandle*** The peripheral interrupt expansion (PIE) object handle

11.2.5.25 void PIE\_setExtIntPolarity (PIE\_Handle *pieHandle*, const CPU\_ExtIntNumber\_e *intNumber*, const PIE\_ExtIntPolarity\_e *polarity*)

Sets the external interrupt polarity.

**Parameters:**

- ← ***pieHandle*** The peripheral interrupt expansion (PIE) handle
- ← ***intNumber*** The external interrupt number
- ← ***polarity*** The signal polarity

11.2.5.26 void PIE\_unregisterPieIntHandler (PIE\_Handle *pieHandle*, const PIE\_GroupNumber\_e *groupNumber*, const PIE\_SubGroupNumber\_e *subGroupNumber*)

Unregisters a handler for a PIE interrupt.

**Parameters:**

- ← ***pieHandle*** The peripheral interrupt expansion (PIE) object handle
- ← ***groupNumber*** The PIE group an interrupt belongs to
- ← ***subGroupNumber*** The PIE subgroup an interrupt belongs to

11.2.5.27 void PIE\_unregisterSystemIntHandler (PIE\_Handle *pieHandle*, const PIE\_SystemInterrupts\_e *systemInt*)

Unregisters a handler for a PIE interrupt.

**Parameters:**

- ← ***pieHandle*** The peripheral interrupt expansion (PIE) object handle
- ← ***systemInt*** The system interrupt to unregister

## 12 Phase Locked Loop (PLL)

Introduction .....	167
API Functions .....	167

### 12.1 Introduction

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.

### 12.2 PLL

#### Data Structures

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

#### Defines

- [PLL\\_BASE\\_ADDR](#)
- [PLL\\_PLLCR\\_DIV\\_BITS](#)
- [PLL\\_PLLSTS\\_DIVSEL\\_BITS](#)
- [PLL\\_PLLSTS\\_MCLKCLR\\_BITS](#)
- [PLL\\_PLLSTS\\_MCLKOFF\\_BITS](#)
- [PLL\\_PLLSTS\\_MCLKSTS\\_BITS](#)
- [PLL\\_PLLSTS\\_NORMRDYE\\_BITS](#)
- [PLL\\_PLLSTS\\_OSCOFF\\_BITS](#)
- [PLL\\_PLLSTS\\_PLLLOCKS\\_BITS](#)
- [PLL\\_PLLSTS\\_PLLOFF\\_BITS](#)

#### Enumerations

- [PLL\\_ClkStatus\\_e](#)
- [PLL\\_DivideSelect\\_e](#)
- [PLL\\_LockStatus\\_e](#)
- [PLL\\_Multiplier\\_e](#)

## 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)

## 12.2.1 Data Structure Documentation

### 12.2.1.1 `_PLL_Obj`

#### Definition:

```
typedef struct
{
    uint16_t PLLSTS;
    uint16_t rsvd_1;
    uint16_t PLLLOCKPRD;
    uint16_t rsvd_2[13];
    uint16_t PLLCR;
}
_PLL_Obj
```

#### Members:

***PLLSTS*** PLL Status Register.  
***rsvd\_1*** Reserved.  
***PLLLOCKPRD*** PLL Lock Period Register.  
***rsvd\_2*** Reserved.  
***PLLCR*** PLL Control Register.

#### Description:

Defines the phase lock loop (PLL) object.



## 12.2.2 Define Documentation

### 12.2.2.1 PLL\_BASE\_ADDR

**Definition:**

```
#define PLL_BASE_ADDR
```

**Description:**

Defines the base address of the phase lock loop (PLL) registers.

### 12.2.2.2 PLL\_PLLCR\_DIV\_BITS

**Definition:**

```
#define PLL_PLLCR_DIV_BITS
```

**Description:**

Defines the location of the DIV bits in the PLLCR register.

### 12.2.2.3 PLL\_PLLSTS\_DIVSEL\_BITS

**Definition:**

```
#define PLL_PLLSTS_DIVSEL_BITS
```

**Description:**

Defines the location of the DIVSEL bits in the PLLSTS register.

### 12.2.2.4 PLL\_PLLSTS\_MCLKCLR\_BITS

**Definition:**

```
#define PLL_PLLSTS_MCLKCLR_BITS
```

**Description:**

Defines the location of the MCLKCLR bits in the PLLSTS register.

### 12.2.2.5 PLL\_PLLSTS\_MCLKOFF\_BITS

**Definition:**

```
#define PLL_PLLSTS_MCLKOFF_BITS
```

**Description:**

Defines the location of the MCLKOFF bits in the PLLSTS register.

#### 12.2.2.6 PLL\_PLLSTS\_MCLKSTS\_BITS

**Definition:**

```
#define PLL_PLLSTS_MCLKSTS_BITS
```

**Description:**

Defines the location of the MCLKSTS bits in the PLLSTS register.

#### 12.2.2.7 PLL\_PLLSTS\_NORMRDYE\_BITS

**Definition:**

```
#define PLL_PLLSTS_NORMRDYE_BITS
```

**Description:**

Defines the location of the NORMRDYE bits in the PLLSTS register.

#### 12.2.2.8 PLL\_PLLSTS\_OSCOFF\_BITS

**Definition:**

```
#define PLL_PLLSTS_OSCOFF_BITS
```

**Description:**

Defines the location of the OSCOFF bits in the PLLSTS register.

#### 12.2.2.9 PLL\_PLLSTS\_PLLLOCKS\_BITS

**Definition:**

```
#define PLL_PLLSTS_PLLLOCKS_BITS
```

**Description:**

Defines the location of the PLLLOCKS bits in the PLLSTS register.

#### 12.2.2.10 PLL\_PLLSTS\_PLLOFF\_BITS

**Definition:**

```
#define PLL_PLLSTS_PLLOFF_BITS
```

**Description:**

Defines the location of the PLLOFF bits in the PLLSTS register.

### 12.2.3 Typedef Documentation

#### 12.2.3.1 PLL\_Handle

**Definition:**

```
typedef struct PLL_Obj *PLL_Handle
```

**Description:**

Defines the phase lock loop (PLL) handle.

### 12.2.3.2 PLL\_Obj

**Definition:**

```
typedef struct _PLL_Obj_ PLL_Obj
```

**Description:**

Defines the phase lock loop (PLL) object.

## 12.2.4 Enumeration Documentation

### 12.2.4.1 PLL\_ClkStatus\_e

**Description:**

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

**Enumerators:**

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

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

### 12.2.4.2 PLL\_DivideSelect\_e

**Description:**

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

**Enumerators:**

**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.

### 12.2.4.3 PLL\_LockStatus\_e

**Description:**

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

**Enumerators:**

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

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

#### 12.2.4.4 PLL\_Multiplier\_e

**Description:**

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

**Enumerators:**

- 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.

### 12.2.5 Function Documentation

#### 12.2.5.1 PLL\_disable

Disables the phase lock loop (PLL).

**Prototype:**

```
void  
PLL_disable(PLL_Handle pllHandle)
```

**Parameters:**

← **pllHandle** The phase lock loop (PLL) object handle

#### 12.2.5.2 void PLL\_disableClkDetect (PLL\_Handle pllHandle)

Disables the clock detect logic.

**Parameters:**

← **pllHandle** The phase lock loop (PLL) object handle

#### 12.2.5.3 void PLL\_disableNormRdy (PLL\_Handle pllHandle)

Disables the NORMRDY signal.

**Parameters:**

← **pllHandle** The phase lock loop (PLL) object handle

#### 12.2.5.4 void PLL\_disableOsc (PLL\_Handle pllHandle)

Disables the oscillator.

**Parameters:**

← **pllHandle** The phase lock loop (PLL) object handle

#### 12.2.5.5 void PLL\_enable (PLL\_Handle pllHandle)

Enables the phase lock loop (PLL).

**Parameters:**

← **pllHandle** The phase lock loop (PLL) object handle

#### 12.2.5.6 void PLL\_enableClkDetect (PLL\_Handle pllHandle)

Enables the clock detect logic.

**Parameters:**

← **pllHandle** The phase lock loop (PLL) object handle

#### 12.2.5.7 void PLL\_enableNormRdy (PLL\_Handle pllHandle)

Enables the NORMRDY signal.

**Parameters:**

← **pllHandle** The phase lock loop (PLL) object handle

#### 12.2.5.8 void PLL\_enableOsc (PLL\_Handle pllHandle)

Enables the oscillator.

**Parameters:**

← **pllHandle** The phase lock loop (PLL) object handle

#### 12.2.5.9 PLL\_ClkStatus\_e PLL\_getClkStatus (PLL\_Handle pllHandle)

Gets the phase lock loop (PLL) clock status.

**Parameters:**

← **pllHandle** The phase lock loop (PLL) object handle

**Returns:**

The clock status

#### 12.2.5.10 [PLL\\_DivideSelect\\_e](#) PLL\_getDivider ([PLL\\_Handle](#) *pllHandle*)

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

**Parameters:**

← ***pllHandle*** The phase lock loop (PLL) object handle

**Returns:**

The divide select value

#### 12.2.5.11 [PLL\\_LockStatus\\_e](#) PLL\_getLockStatus ([PLL\\_Handle](#) *pllHandle*)

Gets the phase lock loop (PLL) lock status.

**Parameters:**

← ***pllHandle*** The phase lock loop (PLL) object handle

**Returns:**

The lock status

#### 12.2.5.12 [PLL\\_Multiplier\\_e](#) PLL\_getMultiplier ([PLL\\_Handle](#) *pllHandle*)

Gets the phase lock loop (PLL) clock frequency.

**Parameters:**

← ***pllHandle*** The phase lock loop (PLL) object handle

**Returns:**

The clock frequency

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

Initializes the phase lock loop (PLL) object handle.

**Parameters:**

← ***pMemory*** A pointer to the base address of the PLL registers

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

**Returns:**

The phase lock loop (PLL) object handle

#### 12.2.5.14 void PLL\_resetClkDetect ([PLL\\_Handle](#) *pllHandle*)

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

**Parameters:**

← ***pllHandle*** The phase lock loop (PLL) object handle

12.2.5.15 void PLL\_setDivider (PLL\_Handle pllHandle, const PLL\_DivideSelect\_e divSelect)

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

**Parameters:**

- ← **pllHandle** The phase lock loop (PLL) object handle
- ← **divSelect** The divide select value

12.2.5.16 void PLL\_setLockPeriod (PLL\_Handle pllHandle, const uint16\_t lockPeriod)

Sets the phase lock loop (PLL) lock time.

**Parameters:**

- ← **pllHandle** The phase lock loop (PLL) object handle
- ← **lockPeriod** The lock period, cycles

12.2.5.17 void PLL\_setMultiplier (PLL\_Handle pllHandle, const PLL\_Multiplier\_e freq)

Sets the phase lock loop (PLL) clock frequency.

**Parameters:**

- ← **pllHandle** The phase lock loop (PLL) object handle
- ← **freq** The clock frequency

12.2.5.18 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.

**Parameters:**

- ← **pllHandle** The phase lock loop (PLL) object handle
- ← **clkMult** The clock multiplier value
- ← **divSelect** The divide select value





# 13 Pulse Width Modulator (PWM)

Introduction .....	177
API Functions .....	177

## 13.1 Introduction

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.

## 13.2 PWM

### Data Structures

- [\\_PWM\\_Obj\\_](#)

### Defines

- [PWM\\_AQCTL\\_CAD\\_BITS](#)
- [PWM\\_AQCTL\\_CAU\\_BITS](#)
- [PWM\\_AQCTL\\_CBD\\_BITS](#)
- [PWM\\_AQCTL\\_CBU\\_BITS](#)
- [PWM\\_AQCTL\\_PRD\\_BITS](#)
- [PWM\\_AQCTL\\_ZRO\\_BITS](#)
- [PWM\\_CMPCTL\\_LOADAMODE\\_BITS](#)
- [PWM\\_CMPCTL\\_LOADBMODE\\_BITS](#)
- [PWM\\_CMPCTL\\_SHDWAFULL\\_BITS](#)
- [PWM\\_CMPCTL\\_SHDWAMODE\\_BITS](#)
- [PWM\\_CMPCTL\\_SHDWBFULL\\_BITS](#)
- [PWM\\_CMPCTL\\_SHDWBMODE\\_BITS](#)
- [PWM\\_DBCTL\\_HALFCYCLE\\_BITS](#)
- [PWM\\_DBCTL\\_INMODE\\_BITS](#)
- [PWM\\_DBCTL\\_OUTMODE\\_BITS](#)
- [PWM\\_DBCTL\\_POLSEL\\_BITS](#)
- [PWM\\_DCFCTL\\_BLANKE\\_BITS](#)
- [PWM\\_DCFCTL\\_BLANKINV\\_BITS](#)
- [PWM\\_DCFCTL\\_PULSESEL\\_BITS](#)
- [PWM\\_DCFCTL\\_SRCSEL\\_BITS](#)
- [PWM\\_DCTRIPSEL\\_DCAHCOMPSEL\\_BITS](#)

- PWM\_DCTRISEL\_DCALCOMPSEL\_BITS
- PWM\_DCTRISEL\_DCBHCOMPSEL\_BITS
- PWM\_DCTRISEL\_DCBLCOMPSEL\_BITS
- PWM\_ePWM1\_BASE\_ADDR
- PWM\_ePWM2\_BASE\_ADDR
- PWM\_ePWM3\_BASE\_ADDR
- PWM\_ePWM4\_BASE\_ADDR
- PWM\_ETCLR\_INT\_BITS
- PWM\_ETCLR\_SOCA\_BITS
- PWM\_ETCLR\_SOCB\_BITS
- PWM\_ETPS\_INTCNT\_BITS
- PWM\_ETPS\_INTPRD\_BITS
- PWM\_ETPS\_SOCACNT\_BITS
- PWM\_ETPS\_SOCAPRD\_BITS
- PWM\_ETPS\_SOCBCNT\_BITS
- PWM\_ETPS\_SOCBPRD\_BITS
- PWM\_ETSEL\_INTEN\_BITS
- PWM\_ETSEL\_INTSEL\_BITS
- PWM\_ETSEL\_SOCAEN\_BITS
- PWM\_ETSEL\_SOCASEL\_BITS
- PWM\_ETSEL\_SOCBEN\_BITS
- PWM\_ETSEL\_SOCBSEL\_BITS
- PWM\_HRCNFG\_AUTOCONV\_BITS
- PWM\_HRCNFG\_CTLMODE\_BITS
- PWM\_HRCNFG\_EDGMODE\_BITS
- PWM\_HRCNFG\_HRLOAD\_BITS
- PWM\_HRCNFG\_SELOUTB\_BITS
- PWM\_HRCNFG\_SWAPAB\_BITS
- PWM\_HRPCTL\_HRPE\_BITS
- PWM\_HRPCTL\_PWMSYNCSEL\_BITS
- PWM\_HRPCTL\_TBPHSHRLOADE\_BITS
- PWM\_PCCTL\_CHPDUTY\_BITS
- PWM\_PCCTL\_CHPEN\_BITS
- PWM\_PCCTL\_CHPFREQ\_BITS
- PWM\_PCCTL\_OSHTWTH\_BITS
- PWM\_TBCTL\_CLKDIV\_BITS
- PWM\_TBCTL\_CTRMODE\_BITS
- PWM\_TBCTL\_FREESOFT\_BITS
- PWM\_TBCTL\_HSPCLKDIV\_BITS
- PWM\_TBCTL\_PHSDIR\_BITS
- PWM\_TBCTL\_PHSN\_BITS
- PWM\_TBCTL\_PRDLT\_BITS
- PWM\_TBCTL\_SWFSYNC\_BITS
- PWM\_TBCTL\_SYNCSEL\_BITS
- PWM\_TZCLR\_CBC\_BITS
- PWM\_TZCLR\_DCAEVT1\_BITS

- [PWM\\_TZCLR\\_DCAEVT2\\_BITS](#)
- [PWM\\_TZCLR\\_DCBEVT1\\_BITS](#)
- [PWM\\_TZCLR\\_DCBEVT2\\_BITS](#)
- [PWM\\_TZCLR\\_INT\\_BITS](#)
- [PWM\\_TZCLR\\_OST\\_BITS](#)
- [PWM\\_TZCTL\\_DCAEVT1\\_BITS](#)
- [PWM\\_TZCTL\\_DCAEVT2\\_BITS](#)
- [PWM\\_TZCTL\\_DCBEVT1\\_BITS](#)
- [PWM\\_TZCTL\\_DCBEVT2\\_BITS](#)
- [PWM\\_TZCTL\\_TZA\\_BITS](#)
- [PWM\\_TZCTL\\_TZB\\_BITS](#)
- [PWM\\_TZDCSEL\\_DCAEVT1\\_BITS](#)
- [PWM\\_TZDCSEL\\_DCAEVT2\\_BITS](#)
- [PWM\\_TZDCSEL\\_DCBEVT1\\_BITS](#)
- [PWM\\_TZDCSEL\\_DCBEVT2\\_BITS](#)
- [PWM\\_TZFRC\\_CBC\\_BITS](#)
- [PWM\\_TZFRC\\_DCAEVT1\\_BITS](#)
- [PWM\\_TZFRC\\_DCAEVT2\\_BITS](#)
- [PWM\\_TZFRC\\_DCBEVT1\\_BITS](#)
- [PWM\\_TZFRC\\_DCBEVT2\\_BITS](#)
- [PWM\\_TZFRC\\_OST\\_BITS](#)

## Enumerations

- [PWM\\_ActionQual\\_e](#)
- [PWM\\_TripZoneFlag\\_e](#)
- [PWM\\_TripZoneSrc\\_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\\_disableAutoConvert](#) ([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\\_disableHrPeriod](#) (PWM\_Handle pwmHandle)
- void [PWM\\_disableHrPhaseSync](#) (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](#) interruptSource)
- void [PWM\\_disableTripZones](#) (PWM\_Handle pwmHandle)
- void [PWM\\_disableTripZoneSrc](#) (PWM\_Handle pwmHandle, const [PWM\\_TripZoneSrc\\_e](#) src)
- void [PWM\\_enableAutoConvert](#) (PWM\_Handle pwmHandle)
- 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\\_enableHrPeriod](#) (PWM\_Handle pwmHandle)
- void [PWM\\_enableHrPhaseSync](#) (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](#) interruptSource)
- void [PWM\\_enableTripZoneSrc](#) (PWM\_Handle pwmHandle, const [PWM\\_TripZoneSrc\\_e](#) src)
- void [PWM\\_forceSync](#) (PWM\_Handle pwmHandle)
- uint16\_t [PWM\\_getCmpA](#) (PWM\_Handle pwmHandle)
- uint16\_t [PWM\\_getCmpAHr](#) (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 actionQual)
- void PWM\_setActionQual\_Zero\_PwmB (PWM\_Handle pwmHandle, const PWM\_ActionQual\_e actionQual)
- void PWM\_setChoppingClkFreq (PWM\_Handle pwmHandle, const PWM\_ChoppingClkFreq\_e clkFreq)
- 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\_setCmpAhr (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 counterMode)
- 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_setHrControlMode` (`PWM_Handle` pwmHandle, const `PWM_HrControlMode_e` controlMode)
- void `PWM_setHrEdgeMode` (`PWM_Handle` pwmHandle, const `PWM_HrEdgeMode_e` edgeMode)
- void `PWM_setHrShadowMode` (`PWM_Handle` pwmHandle, const `PWM_HrShadowMode_e` shadowMode)
- 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` loadMode)
- void `PWM_setLoadMode_CmpB` (`PWM_Handle` pwmHandle, const `PWM_LoadMode_e` loadMode)
- void `PWM_setOneShotTrip` (`PWM_Handle` pwmHandle)
- void `PWM_setPeriod` (`PWM_Handle` pwmHandle, const uint16\_t period)
- void `PWM_setPeriodHr` (`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](#) tripZoneState)
- void [PWM\\_setTripZoneState\\_TZB](#) ([PWM\\_Handle](#) pwmHandle, const [PWM\\_TripZoneState\\_e](#) tripZoneState)
- void [PWM\\_write\\_CmpA](#) ([PWM\\_Handle](#) pwmHandle, const int16\_t pwmData)
- void [PWM\\_write\\_CmpB](#) ([PWM\\_Handle](#) pwmHandle, const int16\_t pwmData)

## 13.2.1 Data Structure Documentation

### 13.2.1.1 \_PWM\_Obj\_

**Definition:**

```
typedef struct
{
    uint16_t TBCTL;
    uint16_t TBSTS;
    uint16_t TBPHSHR;
    uint16_t TBPHS;
    uint16_t TBCTR;
    uint16_t TBPRD;
    uint16_t TBPRDHR;
    uint16_t CMPCTL;
    uint16_t CMPAHR;
    uint16_t CMPA;
    uint16_t CMPB;
    uint16_t AQCTLA;
    uint16_t AQCTLB;
    uint16_t AQSFRC;
    uint16_t AQCSFRC;
    uint16_t DBCTL;
    uint16_t DBRED;
    uint16_t DBFED;
    uint16_t TZSEL;
    uint16_t TZDCSEL;
    uint16_t TZCTL;
    uint16_t TZEINT;
    uint16_t TZFLG;
    uint16_t TZCLR;
    uint16_t TZFRC;
    uint16_t ETSEL;
    uint16_t ETPS;
    uint16_t ETFLG;
    uint16_t ETCLR;
    uint16_t ETFRC;
```

```
uint16_t PCCTL;  
uint16_t rsvd_1;  
uint16_t HRCNFG;  
uint16_t HRPWR;  
uint16_t rsvd_2[4];  
uint16_t HRMSTEP;  
uint16_t rsvd_3;  
uint16_t HRPCTL;  
uint16_t rsvd_4;  
uint16_t TBPRDHRM;  
uint16_t TBPRDM;  
uint16_t CMPAHRM;  
uint16_t CMPAM;  
uint16_t rsvd_5[2];  
uint16_t DCTRIPSEL;  
uint16_t DCACTL;  
uint16_t DCBCTL;  
uint16_t DCFCTL;  
uint16_t DCCAPCTL;  
uint16_t DCFOFFSET;  
uint16_t DCFOFFSETCNT;  
uint16_t DCFWINDOW;  
uint16_t DCFWINDOWCNT;  
uint16_t DCCAP;  
}  
__PWM_Obj__
```

**Members:**

**TBCTL** Time-Base Control Register.  
**TBSTS** Time-Base Status Register.  
**TBPHSHR** Extension for the HRPWM Phase Register.  
**TBPHS** Time-Base Phase Register.  
**TBCTR** Time-Base Counter.  
**TBPRD** Time-Base Period register set.  
**TBPRDHR** Time-Base Period High Resolution Register.  
**CMPCTL** Counter-Compare Control Register.  
**CMPAHR** Extension of HRPWM Counter-Compare A Register.  
**CMPA** Counter-Compare A Register.  
**CMPB** Counter-Compare B Register.  
**AQCTLA** Action-Qualifier Control Register for Output A (EPWMxA).  
**AQCTLB** Action-Qualifier Control Register for Output B (EPWMxB).  
**AQSFRC** Action qual SW force.  
**AQCSFRC** Action qualifier continuous SW force.  
**DBCTL** Dead-band control.  
**DBRED** Dead-band rising edge delay.  
**DBFED** Dead-band falling edge delay.  
**TZSEL** Trip zone select.  
**TZDCSEL** Trip zone digital comparator select.  
**TZCTL** Trip zone control.  
**TZEINT** Trip zone interrupt enable.



**TZFLG** Trip zone interrupt flags.  
**TZCLR** Trip zone clear.  
**TZFRC** Trip zone force interrupt.  
**ETSEL** Event trigger selection.  
**ETPS** Event trigger pre-scaler.  
**ETFLG** Event trigger flags.  
**ETCLR** Event trigger clear.  
**ETFRC** Event trigger force.  
**PCCTL** PWM chopper control.  
**rsvd\_1** Reserved.  
**HRCNFG** HRPWM Config Reg.  
**HRPWR** HRPWM Power Register.  
**rsvd\_2** Reserved.  
**HRMSTEP** HRPWM MEP Step Register.  
**rsvd\_3** Reserved.  
**HRPCTL** High Resolution Period Control.  
**rsvd\_4** Reserved.  
**TBPRDHrm** Time base period High Resolution register mirror.  
**TBPRDM** Time base period register mirror.  
**CMPAHRM** Compare A High Resolution register mirror.  
**CMPAM** Compare A register mirror.  
**rsvd\_5** Reserved.  
**DCTRIPSEL** Digital Compare Trip Select.  
**DCACTL** Digital Compare A Control.  
**DCBCTL** Digital Compare B Control.  
**DCFCTL** Digital Compare Filter Control.  
**DCCAPCTL** Digital Compare Capture Control.  
**DCFOFFSET** Digital Compare Filter Offset.  
**DCFOFFSETCNT** Digital Compare Filter Offset Counter.  
**DCFWINDOW** Digital Compare Filter Window.  
**DCFWINDOWCNT** Digital Compare Filter Window Counter.  
**DCCAP** Digital Compare Filter Counter Capture.

**Description:**

Defines the pulse width modulation (PWM) object.

## 13.2.2 Define Documentation

### 13.2.2.1 PWM\_AQCTL\_CAD\_BITS

**Definition:**

```
#define PWM_AQCTL_CAD_BITS
```

**Description:**

Defines the location of the CAD bits in the AQCTL register.

### 13.2.2.2 PWM\_AQCTL\_CAU\_BITS

**Definition:**

```
#define PWM_AQCTL_CAU_BITS
```

**Description:**

Defines the location of the CAU bits in the AQCTL register.

### 13.2.2.3 PWM\_AQCTL\_CBD\_BITS

**Definition:**

```
#define PWM_AQCTL_CBD_BITS
```

**Description:**

Defines the location of the CBD bits in the AQCTL register.

### 13.2.2.4 PWM\_AQCTL\_CBU\_BITS

**Definition:**

```
#define PWM_AQCTL_CBU_BITS
```

**Description:**

Defines the location of the CBU bits in the AQCTL register.

### 13.2.2.5 PWM\_AQCTL\_PRD\_BITS

**Definition:**

```
#define PWM_AQCTL_PRD_BITS
```

**Description:**

Defines the location of the PRD bits in the AQCTL register.

### 13.2.2.6 PWM\_AQCTL\_ZRO\_BITS

**Definition:**

```
#define PWM_AQCTL_ZRO_BITS
```

**Description:**

Defines the location of the ZRO bits in the AQCTL register.

### 13.2.2.7 PWM\_CMPCTL\_LOADAMODE\_BITS

**Definition:**

```
#define PWM_CMPCTL_LOADAMODE_BITS
```

**Description:**

Defines the location of the LOADAMODE bits in the CMPCTL register.

### 13.2.2.8 PWM\_CMPCTL\_LOADBMODE\_BITS

**Definition:**

```
#define PWM_CMPCTL_LOADBMODE_BITS
```

**Description:**

Defines the location of the LOADBMODE bits in the CMPCTL register.

### 13.2.2.9 PWM\_CMPCTL\_SHDWAFULL\_BITS

**Definition:**

```
#define PWM_CMPCTL_SHDWAFULL_BITS
```

**Description:**

Defines the location of the SHDWAFULL bits in the CMPCTL register.

### 13.2.2.10 PWM\_CMPCTL\_SHDWAMODE\_BITS

**Definition:**

```
#define PWM_CMPCTL_SHDWAMODE_BITS
```

**Description:**

Defines the location of the SHDWAMODE bits in the CMPCTL register.

### 13.2.2.11 PWM\_CMPCTL\_SHDWBFULL\_BITS

**Definition:**

```
#define PWM_CMPCTL_SHDWBFULL_BITS
```

**Description:**

Defines the location of the SHDWBFULL bits in the CMPCTL register.

### 13.2.2.12 PWM\_CMPCTL\_SHDWBMODE\_BITS

**Definition:**

```
#define PWM_CMPCTL_SHDWBMODE_BITS
```

**Description:**

Defines the location of the SHDWBMODE bits in the CMPCTL register.

### 13.2.2.13 PWM\_DBCTL\_HALFCYCLE\_BITS

**Definition:**

```
#define PWM_DBCTL_HALFCYCLE_BITS
```

**Description:**

Defines the location of the HALFCYCLE bits in the DBCTL register.

#### 13.2.2.14 PWM\_DBCTL\_INMODE\_BITS

**Definition:**

```
#define PWM_DBCTL_INMODE_BITS
```

**Description:**

Defines the location of the INMODE bits in the DBCTL register.

#### 13.2.2.15 PWM\_DBCTL\_OUTMODE\_BITS

**Definition:**

```
#define PWM_DBCTL_OUTMODE_BITS
```

**Description:**

Defines the location of the OUTMODE bits in the DBCTL register.

#### 13.2.2.16 PWM\_DBCTL\_POLSEL\_BITS

**Definition:**

```
#define PWM_DBCTL_POLSEL_BITS
```

**Description:**

Defines the location of the POLSEL bits in the DBCTL register.

#### 13.2.2.17 PWM\_DCFCTL\_BLANKE\_BITS

**Definition:**

```
#define PWM_DCFCTL_BLANKE_BITS
```

**Description:**

Defines the location of the BLANKE bits in the DCFCTL register.

#### 13.2.2.18 PWM\_DCFCTL\_BLANKINV\_BITS

**Definition:**

```
#define PWM_DCFCTL_BLANKINV_BITS
```

**Description:**

Defines the location of the BLANKINV bits in the DCFCTL register.

#### 13.2.2.19 PWM\_DCFCTL\_PULSESEL\_BITS

**Definition:**

```
#define PWM_DCFCTL_PULSESEL_BITS
```

**Description:**

Defines the location of the PULSESEL bits in the DCFCTL register.

#### 13.2.2.20 PWM\_DCFCTL\_SRCSEL\_BITS

**Definition:**

```
#define PWM_DCFCTL_SRCSEL_BITS
```

**Description:**

Defines the location of the SRCSEL bits in the DCFCTL register.

#### 13.2.2.21 PWM\_DCTRIPSEL\_DCAHCOMPSEL\_BITS

**Definition:**

```
#define PWM_DCTRIPSEL_DCAHCOMPSEL_BITS
```

**Description:**

Defines the location of the DCAHCOMPSEL bits in the DCTRIPSEL register.

#### 13.2.2.22 PWM\_DCTRIPSEL\_DCALCOMPSEL\_BITS

**Definition:**

```
#define PWM_DCTRIPSEL_DCALCOMPSEL_BITS
```

**Description:**

Defines the location of the DCALCOMPSEL bits in the DCTRIPSEL register.

#### 13.2.2.23 PWM\_DCTRIPSEL\_DCBHCOMPSEL\_BITS

**Definition:**

```
#define PWM_DCTRIPSEL_DCBHCOMPSEL_BITS
```

**Description:**

Defines the location of the DCBHCOMPSEL bits in the DCTRIPSEL register.

#### 13.2.2.24 PWM\_DCTRIPSEL\_DCBLCOMPSEL\_BITS

**Definition:**

```
#define PWM_DCTRIPSEL_DCBLCOMPSEL_BITS
```

**Description:**

Defines the location of the DCBLCOMPSEL bits in the DCTRIPSEL register.

#### 13.2.2.25 PWM\_ePWM1\_BASE\_ADDR

**Definition:**

```
#define PWM_ePWM1_BASE_ADDR
```

**Description:**

Defines the base address of the pulse width modulation (PWM) 1 registers.

#### 13.2.2.26 PWM\_ePWM2\_BASE\_ADDR

**Definition:**

```
#define PWM_ePWM2_BASE_ADDR
```

**Description:**

Defines the base address of the pulse width modulation (PWM) 2 registers.

#### 13.2.2.27 PWM\_ePWM3\_BASE\_ADDR

**Definition:**

```
#define PWM_ePWM3_BASE_ADDR
```

**Description:**

Defines the base address of the pulse width modulation (PWM) 3 registers.

#### 13.2.2.28 PWM\_ePWM4\_BASE\_ADDR

**Definition:**

```
#define PWM_ePWM4_BASE_ADDR
```

**Description:**

Defines the base address of the pulse width modulation (PWM) 4 registers.

#### 13.2.2.29 PWM\_ETCLR\_INT\_BITS

**Definition:**

```
#define PWM_ETCLR_INT_BITS
```

**Description:**

Defines the location of the ETCR bits in the ETCLR register.

#### 13.2.2.30 PWM\_ETCLR\_SOCA\_BITS

**Definition:**

```
#define PWM_ETCLR_SOCA_BITS
```

**Description:**

Defines the location of the SOCA bits in the ETCLR register.

#### 13.2.2.31 PWM\_ETCLR\_SOCB\_BITS

**Definition:**

```
#define PWM_ETCLR_SOCB_BITS
```

**Description:**

Defines the location of the SOCB bits in the ETCLR register.

### 13.2.2.32 PWM\_ETPS\_INTCNT\_BITS

**Definition:**

```
#define PWM_ETPS_INTCNT_BITS
```

**Description:**

Defines the location of the INTCNT bits in the ETPS register.

### 13.2.2.33 PWM\_ETPS\_INTPRD\_BITS

**Definition:**

```
#define PWM_ETPS_INTPRD_BITS
```

**Description:**

Defines the location of the INTPRD bits in the ETPS register.

### 13.2.2.34 PWM\_ETPS\_SOCACNT\_BITS

**Definition:**

```
#define PWM_ETPS_SOCACNT_BITS
```

**Description:**

Defines the location of the SOCACNT bits in the ETPS register.

### 13.2.2.35 PWM\_ETPS\_SOCAPRD\_BITS

**Definition:**

```
#define PWM_ETPS_SOCAPRD_BITS
```

**Description:**

Defines the location of the SOCAPRD bits in the ETPS register.

### 13.2.2.36 PWM\_ETPS\_SOCBCNT\_BITS

**Definition:**

```
#define PWM_ETPS_SOCBCNT_BITS
```

**Description:**

Defines the location of the SOCBCNT bits in the ETPS register.

### 13.2.2.37 PWM\_ETPS\_SOCBPRD\_BITS

**Definition:**

```
#define PWM_ETPS_SOCBPRD_BITS
```

**Description:**

Defines the location of the SOCBPRD bits in the ETPS register.

### 13.2.2.38 PWM\_ETSEL\_INTEN\_BITS

**Definition:**

```
#define PWM_ETSEL_INTEN_BITS
```

**Description:**

Defines the location of the INTEN bits in the ETSEL register.

### 13.2.2.39 PWM\_ETSEL\_INTSEL\_BITS

**Definition:**

```
#define PWM_ETSEL_INTSEL_BITS
```

**Description:**

Defines the location of the INTSEL bits in the ETSEL register.

### 13.2.2.40 PWM\_ETSEL\_SOCAEN\_BITS

**Definition:**

```
#define PWM_ETSEL_SOCAEN_BITS
```

**Description:**

Defines the location of the SOCAEN bits in the ETSEL register.

### 13.2.2.41 PWM\_ETSEL\_SOCASEL\_BITS

**Definition:**

```
#define PWM_ETSEL_SOCASEL_BITS
```

**Description:**

Defines the location of the SOCASEL bits in the ETSEL register.

### 13.2.2.42 PWM\_ETSEL\_SOCBEN\_BITS

**Definition:**

```
#define PWM_ETSEL_SOCBEN_BITS
```

**Description:**

Defines the location of the SOCBEN bits in the ETSEL register.

### 13.2.2.43 PWM\_ETSEL\_SOCBSEL\_BITS

**Definition:**

```
#define PWM_ETSEL_SOCBSEL_BITS
```

**Description:**

Defines the location of the SOCBSEL bits in the ETSEL register.



#### 13.2.2.44 PWM\_HRCNFG\_AUTOCONV\_BITS

**Definition:**

```
#define PWM_HRCNFG_AUTOCONV_BITS
```

**Description:**

Defines the location of the AUTOCONV bits in the HRCNFG register.

#### 13.2.2.45 PWM\_HRCNFG\_CTLMODE\_BITS

**Definition:**

```
#define PWM_HRCNFG_CTLMODE_BITS
```

**Description:**

Defines the location of the CTLMODE bits in the HRCNFG register.

#### 13.2.2.46 PWM\_HRCNFG\_EDGMODE\_BITS

**Definition:**

```
#define PWM_HRCNFG_EDGMODE_BITS
```

**Description:**

Defines the location of the EDGMODE bits in the HRCNFG register.

#### 13.2.2.47 PWM\_HRCNFG\_HRLOAD\_BITS

**Definition:**

```
#define PWM_HRCNFG_HRLOAD_BITS
```

**Description:**

Defines the location of the HRLOAD bits in the HRCNFG register.

#### 13.2.2.48 PWM\_HRCNFG\_SELOUTB\_BITS

**Definition:**

```
#define PWM_HRCNFG_SELOUTB_BITS
```

**Description:**

Defines the location of the SELOUTB bits in the HRCNFG register.

#### 13.2.2.49 PWM\_HRCNFG\_SWAPAB\_BITS

**Definition:**

```
#define PWM_HRCNFG_SWAPAB_BITS
```

**Description:**

Defines the location of the SWAPAB bits in the HRCNFG register.

#### 13.2.2.50 PWM\_HRPCTL\_HRPE\_BITS

**Definition:**

```
#define PWM_HRPCTL_HRPE_BITS
```

**Description:**

Defines the location of the HRPE bits in the HRPCTL register.

#### 13.2.2.51 PWM\_HRPCTL\_PWMSYNCSEL\_BITS

**Definition:**

```
#define PWM_HRPCTL_PWMSYNCSEL_BITS
```

**Description:**

Defines the location of the PWMSYNCSEL bits in the HRPCTL register.

#### 13.2.2.52 PWM\_HRPCTL\_TBPHSHRLOADE\_BITS

**Definition:**

```
#define PWM_HRPCTL_TBPHSHRLOADE_BITS
```

**Description:**

Defines the location of the TBPHSHRLOADE bits in the HRPCTL register.

#### 13.2.2.53 PWM\_PCCTL\_CHPDUTY\_BITS

**Definition:**

```
#define PWM_PCCTL_CHPDUTY_BITS
```

**Description:**

Defines the location of the CHPDUTY bits in the PCCTL register.

#### 13.2.2.54 PWM\_PCCTL\_CHPEN\_BITS

**Definition:**

```
#define PWM_PCCTL_CHPEN_BITS
```

**Description:**

Defines the location of the CHPEN bits in the PCCTL register.

#### 13.2.2.55 PWM\_PCCTL\_CHPFREQ\_BITS

**Definition:**

```
#define PWM_PCCTL_CHPFREQ_BITS
```

**Description:**

Defines the location of the CHPFREQ bits in the PCCTL register.

#### 13.2.2.56 PWM\_PCCTL\_OSHTWTH\_BITS

**Definition:**

```
#define PWM_PCCTL_OSHTWTH_BITS
```

**Description:**

Defines the location of the OSHTWTH bits in the PCCTL register.

#### 13.2.2.57 PWM\_TBCTL\_CLKDIV\_BITS

**Definition:**

```
#define PWM_TBCTL_CLKDIV_BITS
```

**Description:**

Defines the location of the CLKDIV bits in the TBCTL register.

#### 13.2.2.58 PWM\_TBCTL\_CTRMODE\_BITS

**Definition:**

```
#define PWM_TBCTL_CTRMODE_BITS
```

**Description:**

Defines the location of the CTRMODE bits in the TBCTL register.

#### 13.2.2.59 PWM\_TBCTL\_FREESOFT\_BITS

**Definition:**

```
#define PWM_TBCTL_FREESOFT_BITS
```

**Description:**

Defines the location of the FREESOFT bits in the TBCTL register.

#### 13.2.2.60 PWM\_TBCTL\_HSPCLKDIV\_BITS

**Definition:**

```
#define PWM_TBCTL_HSPCLKDIV_BITS
```

**Description:**

Defines the location of the HSPCLKDIV bits in the TBCTL register.

#### 13.2.2.61 PWM\_TBCTL\_PHSDIR\_BITS

**Definition:**

```
#define PWM_TBCTL_PHSDIR_BITS
```

**Description:**

Defines the location of the PHSDIR bits in the TBCTL register.

#### 13.2.2.62 PWM\_TBCTL\_PHSEN\_BITS

**Definition:**

```
#define PWM_TBCTL_PHSEN_BITS
```

**Description:**

Defines the location of the PHSEN bits in the TBCTL register.

#### 13.2.2.63 PWM\_TBCTL\_PRDLD\_BITS

**Definition:**

```
#define PWM_TBCTL_PRDLD_BITS
```

**Description:**

Defines the location of the PRDLD bits in the TBCTL register.

#### 13.2.2.64 PWM\_TBCTL\_SWFSYNC\_BITS

**Definition:**

```
#define PWM_TBCTL_SWFSYNC_BITS
```

**Description:**

Defines the location of the SWFSYNC bits in the TBCTL register.

#### 13.2.2.65 PWM\_TBCTL\_SYNCOSSEL\_BITS

**Definition:**

```
#define PWM_TBCTL_SYNCOSSEL_BITS
```

**Description:**

Defines the location of the SYNCOSSEL bits in the TBCTL register.

#### 13.2.2.66 PWM\_TZCLR\_CBC\_BITS

**Definition:**

```
#define PWM_TZCLR_CBC_BITS
```

**Description:**

Defines the location of the CBC bits in the TXCLR register.

#### 13.2.2.67 PWM\_TZCLR\_DCAEVT1\_BITS

**Definition:**

```
#define PWM_TZCLR_DCAEVT1_BITS
```

**Description:**

Defines the location of the DCAEVT1 bits in the TXCLR register.

### 13.2.2.68 PWM\_TZCLR\_DCAEVT2\_BITS

**Definition:**

```
#define PWM_TZCLR_DCAEVT2_BITS
```

**Description:**

Defines the location of the DCAEVT2 bits in the TXCLR register.

### 13.2.2.69 PWM\_TZCLR\_DCBEVT1\_BITS

**Definition:**

```
#define PWM_TZCLR_DCBEVT1_BITS
```

**Description:**

Defines the location of the DCBEVT1 bits in the TXCLR register.

### 13.2.2.70 PWM\_TZCLR\_DCBEVT2\_BITS

**Definition:**

```
#define PWM_TZCLR_DCBEVT2_BITS
```

**Description:**

Defines the location of the DCBEVT2 bits in the TXCLR register.

### 13.2.2.71 PWM\_TZCLR\_INT\_BITS

**Definition:**

```
#define PWM_TZCLR_INT_BITS
```

**Description:**

Defines the location of the INT bits in the TXCLR register.

### 13.2.2.72 PWM\_TZCLR\_OST\_BITS

**Definition:**

```
#define PWM_TZCLR_OST_BITS
```

**Description:**

Defines the location of the OST bits in the TXCLR register.

### 13.2.2.73 PWM\_TZCTL\_DCAEVT1\_BITS

**Definition:**

```
#define PWM_TZCTL_DCAEVT1_BITS
```

**Description:**

Defines the location of the DCAEVT1 bits in the TZCTL register.

#### 13.2.2.74 PWM\_TZCTL\_DCAEVT2\_BITS

**Definition:**

```
#define PWM_TZCTL_DCAEVT2_BITS
```

**Description:**

Defines the location of the DCAEVT2 bits in the TZCTL register.

#### 13.2.2.75 PWM\_TZCTL\_DCBEVT1\_BITS

**Definition:**

```
#define PWM_TZCTL_DCBEVT1_BITS
```

**Description:**

Defines the location of the DCBEVT1 bits in the TZCTL register.

#### 13.2.2.76 PWM\_TZCTL\_DCBEVT2\_BITS

**Definition:**

```
#define PWM_TZCTL_DCBEVT2_BITS
```

**Description:**

Defines the location of the DCBEVT2 bits in the TZCTL register.

#### 13.2.2.77 PWM\_TZCTL\_TZA\_BITS

**Definition:**

```
#define PWM_TZCTL_TZA_BITS
```

**Description:**

Defines the location of the TZA bits in the TZCTL register.

#### 13.2.2.78 PWM\_TZCTL\_TZB\_BITS

**Definition:**

```
#define PWM_TZCTL_TZB_BITS
```

**Description:**

Defines the location of the TZB bits in the TZCTL register.

#### 13.2.2.79 PWM\_TZDCSEL\_DCAEVT1\_BITS

**Definition:**

```
#define PWM_TZDCSEL_DCAEVT1_BITS
```

**Description:**

Defines the location of the DCAEVT1 bits in the TZDCSEL register.

#### 13.2.2.80 PWM\_TZDCSEL\_DCAEVT2\_BITS

**Definition:**

```
#define PWM_TZDCSEL_DCAEVT2_BITS
```

**Description:**

Defines the location of the DCAEVT2 bits in the TZDCSEL register.

#### 13.2.2.81 PWM\_TZDCSEL\_DCBEVT1\_BITS

**Definition:**

```
#define PWM_TZDCSEL_DCBEVT1_BITS
```

**Description:**

Defines the location of the DCBEVT1 bits in the TZDCSEL register.

#### 13.2.2.82 PWM\_TZDCSEL\_DCBEVT2\_BITS

**Definition:**

```
#define PWM_TZDCSEL_DCBEVT2_BITS
```

**Description:**

Defines the location of the DCBEVT2 bits in the TZDCSEL register.

#### 13.2.2.83 PWM\_TZFRC\_CBC\_BITS

**Definition:**

```
#define PWM_TZFRC_CBC_BITS
```

**Description:**

Defines the location of the CBC bits in the TZFRC register.

#### 13.2.2.84 PWM\_TZFRC\_DCAEVT1\_BITS

**Definition:**

```
#define PWM_TZFRC_DCAEVT1_BITS
```

**Description:**

Defines the location of the DCAEVT1 bits in the TZFRC register.

#### 13.2.2.85 PWM\_TZFRC\_DCAEVT2\_BITS

**Definition:**

```
#define PWM_TZFRC_DCAEVT2_BITS
```

**Description:**

Defines the location of the DCAEVT2 bits in the TZFRC register.

### 13.2.2.86 PWM\_TZFRC\_DCBEVT1\_BITS

**Definition:**

```
#define PWM_TZFRC_DCBEVT1_BITS
```

**Description:**

Defines the location of the DCBEVT1 bits in the TZFRC register.

### 13.2.2.87 PWM\_TZFRC\_DCBEVT2\_BITS

**Definition:**

```
#define PWM_TZFRC_DCBEVT2_BITS
```

**Description:**

Defines the location of the DCBEVT2 bits in the TZFRC register.

### 13.2.2.88 PWM\_TZFRC\_OST\_BITS

**Definition:**

```
#define PWM_TZFRC_OST_BITS
```

**Description:**

Defines the location of the OST bits in the TZFRC register.

## 13.2.3 Typedef Documentation

### 13.2.3.1 PWM\_Handle

**Definition:**

```
typedef struct PWM_Obj *PWM_Handle
```

**Description:**

Defines the pulse width modulation (PWM) handle.

### 13.2.3.2 PWM\_Obj

**Definition:**

```
typedef struct _PWM_Obj_ PWM_Obj
```

**Description:**

Defines the pulse width modulation (PWM) object.



## 13.2.4 Enumeration Documentation

### 13.2.4.1 PWM\_ActionQual\_e

**Description:**

Enumeration to define the pulse width modulation (PWM) action qualifiers.

### 13.2.4.2 enum PWM\_ChoppingClkFreq\_e

Enumeration to define the pulse width modulation (PWM) chopping clock frequencies.

### 13.2.4.3 enum PWM\_ChoppingDutyCycle\_e

Enumeration to define the pulse width modulation (PWM) chopping clock duty cycles.

### 13.2.4.4 enum PWM\_ChoppingPulseWidth\_e

Enumeration to define the pulse width modulation (PWM) chopping clock pulse widths.

### 13.2.4.5 enum PWM\_ClkDiv\_e

Enumeration to define the pulse width modulation (PWM) clock dividers.

### 13.2.4.6 enum PWM\_CounterMode\_e

Enumeration to define the pulse width modulation (PWM) counter modes.

### 13.2.4.7 enum PWM\_DeadBandInputMode\_e

Enumeration to define the pulse width modulation (PWM) deadband options.

### 13.2.4.8 enum PWM\_DeadBandOutputMode\_e

Enumeration to define the pulse width modulation (PWM) deadband output modes.

### 13.2.4.9 enum PWM\_DeadBandPolarity\_e

Enumeration to define the pulse width modulation (PWM) deadband polarity.

### 13.2.4.10 enum PWM\_DigitalCompare\_FilterSrc\_e

Enumeration to define the pulse width modulation (PWM) digital compare filter sources.

13.2.4.11 enum [PWM\\_DigitalCompare\\_Input\\_e](#)

Enumeration to define the pulse width modulation (PWM) digital compare inputs.

13.2.4.12 enum [PWM\\_DigitalCompare\\_InputSel\\_e](#)

Enumeration to define the pulse width modulation (PWM) digital compare input choices.

13.2.4.13 enum [PWM\\_DigitalCompare\\_PulseSel\\_e](#)

Enumeration to define the pulse width modulation (PWM) digital compare blanking pulse select.

13.2.4.14 enum [PWM\\_HrControlMode\\_e](#)

Enumeration to define the pulse width modulation (PWM) high resolution control mode options.

13.2.4.15 enum [PWM\\_HrEdgeMode\\_e](#)

Enumeration to define the pulse width modulation (PWM) high resolution edge mode options.

13.2.4.16 enum [PWM\\_HrShadowMode\\_e](#)

Enumeration to define the pulse width modulation (PWM) high resolution shadow load mode options.

13.2.4.17 enum [PWM\\_HspClkDiv\\_e](#)

Enumeration to define the pulse width modulation (PWM) high speed clock divide options.

13.2.4.18 enum [PWM\\_IntMode\\_e](#)

Enumeration to define the pulse width modulation (PWM) interrupt generation modes.

13.2.4.19 enum [PWM\\_IntPeriod\\_e](#)

Enumeration to define the pulse width modulation (PWM) interrupt period options.

13.2.4.20 enum [PWM\\_LoadMode\\_e](#)

Enumeration to define the pulse width modulation (PWM) load modes.

13.2.4.21 enum [PWM\\_Number\\_e](#)

Enumeration to define the pulse width modulation (PWM) numbers.

13.2.4.22 enum [PWM\\_PeriodLoad\\_e](#)

Enumeration to define the pulse width modulation (PWM) period load options.

13.2.4.23 enum [PWM\\_PhaseDir\\_e](#)

Enumeration to define the pulse width modulation (PWM) phase direction modes.

13.2.4.24 enum [PWM\\_RunMode\\_e](#)

Enumeration to define the pulse width modulation (PWM) run modes.

13.2.4.25 enum [PWM\\_ShadowMode\\_e](#)

Enumeration to define the pulse width modulation (PWM) shadow modes.

13.2.4.26 enum [PWM\\_ShadowStatus\\_e](#)

Enumeration to define the pulse width modulation (PWM) shadow status options.

13.2.4.27 enum [PWM\\_SocPeriod\\_e](#)

Enumeration to define the pulse width modulation (PWM) start of conversion (SOC) period options.

13.2.4.28 enum [PWM\\_SocPulseSrc\\_e](#)

Enumeration to define the pulse width modulation (PWM) start of conversion (SOC) sources.

13.2.4.29 enum [PWM\\_SyncMode\\_e](#)

Enumeration to define the pulse width modulation (PWM) sync modes.

13.2.4.30 enum [PWM\\_TripZoneDCEventSel\\_e](#)

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

**Enumerators:**

***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.

#### 13.2.4.31 PWM\_TripZoneFlag\_e

**Description:**

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

**Enumerators:**

***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.

#### 13.2.4.32 PWM\_TripZoneSrc\_e

**Description:**

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

#### 13.2.4.33 enum [PWM\\_TripZoneState\\_e](#)

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

### 13.2.5 Function Documentation

#### 13.2.5.1 void PWM\_clearIntFlag ([PWM\\_Handle](#) pwmHandle) [inline]

Clears the pulse width modulation (PWM) interrupt flag.

**Parameters:**

← ***pwmHandle*** The pulse width modulation (PWM) object handle

13.2.5.2 void PWM\_clearOneShotTrip ([PWM\\_Handle](#) *pwmHandle*) [[inline](#)]

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

**Parameters:**

← ***pwmHandle*** The pulse width modulation (PWM) object handle

13.2.5.3 void PWM\_clearSocAFlag ([PWM\\_Handle](#) *pwmHandle*) [[inline](#)]

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

**Parameters:**

← ***pwmHandle*** The pulse width modulation (PWM) object handle

13.2.5.4 void PWM\_clearSocBFlag ([PWM\\_Handle](#) *pwmHandle*) [[inline](#)]

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

**Parameters:**

← ***pwmHandle*** The pulse width modulation (PWM) object handle

13.2.5.5 void PWM\_clearTripZone ([PWM\\_Handle](#) *pwmHandle*, const [PWM\\_TripZoneFlag\\_e](#) *tripZoneFlag*)

Clears the trip zone (TZ) flag specified.

**Parameters:**

← ***pwmHandle*** The pulse width modulation (PWM) object handle

← ***tripZoneFlag*** The trip zone flag to clear

13.2.5.6 void PWM\_decrementDeadBandFallingEdgeDelay ([PWM\\_Handle](#) *pwmHandle*)

Decrement the dead band falling edge delay.

**Parameters:**

← ***pwmHandle*** The pulse width modulation (PWM) object handle

13.2.5.7 void PWM\_decrementDeadBandRisingEdgeDelay ([PWM\\_Handle](#) *pwmHandle*)

Decrement the dead band rising edge delay.

**Parameters:**

← ***pwmHandle*** The pulse width modulation (PWM) object handle

#### 13.2.5.8 void PWM\_disableAutoConvert ([PWM\\_Handle](#) pwmHandle)

Disables auto conversion of delay line value.

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

#### 13.2.5.9 void PWM\_disableChopping ([PWM\\_Handle](#) pwmHandle)

Disables the pulse width modulation (PWM) chopping.

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

#### 13.2.5.10 void PWM\_disableCounterLoad ([PWM\\_Handle](#) pwmHandle)

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

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

#### 13.2.5.11 void PWM\_disableDeadBand ([PWM\\_Handle](#) pwmHandle)

Disables the pulse width modulation (PWM) deadband.

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

#### 13.2.5.12 void PWM\_disableDeadBandHalfCycle ([PWM\\_Handle](#) pwmHandle)

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

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

#### 13.2.5.13 void PWM\_disableDigitalCompareBlankingWindow ([PWM\\_Handle](#) pwmHandle)

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

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

13.2.5.14 void PWM\_disableDigitalCompareBlankingWindowInversion ([PWM\\_Handle](#) *pwmHandle*)

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

**Parameters:**

← ***pwmHandle*** The pulse width modulation (PWM) object handle

13.2.5.15 void PWM\_disableHrPeriod ([PWM\\_Handle](#) *pwmHandle*)

Disables high resolution period control.

**Parameters:**

← ***pwmHandle*** The pulse width modulation (PWM) object handle

13.2.5.16 void PWM\_disableHrPhaseSync ([PWM\\_Handle](#) *pwmHandle*)

Disables high resolution phase synchronization.

**Parameters:**

← ***pwmHandle*** The pulse width modulation (PWM) object handle

13.2.5.17 void PWM\_disableInt ([PWM\\_Handle](#) *pwmHandle*)

Disables the pulse width modulation (PWM) interrupt.

**Parameters:**

← ***pwmHandle*** The pulse width modulation (PWM) object handle

13.2.5.18 void PWM\_disableSocAPulse ([PWM\\_Handle](#) *pwmHandle*)

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

**Parameters:**

← ***pwmHandle*** The pulse width modulation (PWM) object handle

13.2.5.19 void PWM\_disableSocBPulse ([PWM\\_Handle](#) *pwmHandle*)

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

**Parameters:**

← ***pwmHandle*** The pulse width modulation (PWM) object handle

13.2.5.20 void PWM\_disableTripZoneInt ([PWM\\_Handle](#) pwmHandle, const [PWM\\_TripZoneFlag\\_e](#) interruptSource)

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

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **interrupt** The interrupt source to disable

13.2.5.21 void PWM\_disableTripZones ([PWM\\_Handle](#) pwmHandle)

Disables the pulse width modulation (PWM) trip zones.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle

13.2.5.22 void PWM\_disableTripZoneSrc ([PWM\\_Handle](#) pwmHandle, const [PWM\\_TripZoneSrc\\_e](#) src)

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

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **src** The pulse width modulation (PWM) trip zone source

13.2.5.23 void PWM\_enableAutoConvert ([PWM\\_Handle](#) pwmHandle)

Enables auto conversion of delay line value.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle

13.2.5.24 void PWM\_enableChopping ([PWM\\_Handle](#) pwmHandle)

Enables the pulse width modulation (PWM) chopping.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle

13.2.5.25 void PWM\_enableCounterLoad ([PWM\\_Handle](#) pwmHandle)

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

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle



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

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

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

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

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

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

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

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

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

**13.2.5.29 void PWM\_enableHrPeriod (PWM\_Handle pwmHandle)**

Enables high resolution period control.

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

**13.2.5.30 void PWM\_enableHrPhaseSync (PWM\_Handle pwmHandle)**

Enables high resolution phase synchronization.

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

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

Enables the pulse width modulation (PWM) interrupt.

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

#### 13.2.5.32 void PWM\_enableSocAPulse ([PWM\\_Handle](#) pwmHandle)

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

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

#### 13.2.5.33 void PWM\_enableSocBPulse ([PWM\\_Handle](#) pwmHandle)

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

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

#### 13.2.5.34 void PWM\_enableTripZoneInt ([PWM\\_Handle](#) pwmHandle, const [PWM\\_TripZoneFlag\\_e](#) interruptSource)

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

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

← **interrupt** The interrupt source to enable

#### 13.2.5.35 void PWM\_enableTripZoneSrc ([PWM\\_Handle](#) pwmHandle, const [PWM\\_TripZoneSrc\\_e](#) src)

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

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

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

#### 13.2.5.36 void PWM\_forceSync ([PWM\\_Handle](#) pwmHandle) [inline]

Force Synchronization.

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

#### 13.2.5.37 uint16\_t PWM\_getCmpA ([PWM\\_Handle](#) pwmHandle) [inline]

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

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

**Returns:**

The PWM compare data value

**13.2.5.38** uint16\_t PWM\_getCmpAHr ([PWM\\_Handle pwmHandle](#)) [inline]

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

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

**Returns:**

The PWM compare high resolution data value

**13.2.5.39** uint16\_t PWM\_getCmpB ([PWM\\_Handle pwmHandle](#)) [inline]

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

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

**Returns:**

The PWM compare data value

**13.2.5.40** uint16\_t PWM\_getDeadBandFallingEdgeDelay ([PWM\\_Handle pwmHandle](#))

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

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

**Returns:**

The delay

**13.2.5.41** uint16\_t PWM\_getDeadBandRisingEdgeDelay ([PWM\\_Handle pwmHandle](#))

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

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

**Returns:**

The delay

#### 13.2.5.42 uint16\_t PWM\_getIntCount ([PWM\\_Handle](#) pwmHandle)

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

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

**Returns:**

The interrupt event count

#### 13.2.5.43 uint16\_t PWM\_getPeriod ([PWM\\_Handle](#) pwmHandle) [inline]

Gets the pulse width modulation (PWM) period value.

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

**Returns:**

The pwm period value

#### 13.2.5.44 uint16\_t PWM\_getSocACount ([PWM\\_Handle](#) pwmHandle)

Gets the pulse width modulation (PWM) start of conversion (SOC) A count.

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

**Returns:**

The SOC A count

#### 13.2.5.45 uint16\_t PWM\_getSocBCount ([PWM\\_Handle](#) pwmHandle)

Gets the pulse width modulation (PWM) start of conversion (SOC) B count.

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

**Returns:**

The SOC B count

#### 13.2.5.46 void PWM\_incrementDeadBandFallingEdgeDelay ([PWM\\_Handle](#) pwmHandle)

Increment the dead band falling edge delay.

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

13.2.5.47 void PWM\_incrementDeadBandRisingEdgeDelay ([PWM\\_Handle](#) pwmHandle)

Increment the dead band rising edge delay.

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

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

Initializes the pulse width modulation (PWM) object handle.

**Parameters:**

← **pMemory** A pointer to the base address of the PWM registers

← **numBytes** The number of bytes allocated for the PWM object, bytes

**Returns:**

The pulse width modulation (PWM) object handle

13.2.5.49 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.

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

← **actionQual** The action qualifier

13.2.5.50 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.

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

← **actionQual** The action qualifier

13.2.5.51 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.

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

← **actionQual** The action qualifier

13.2.5.52 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.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **actionQual** The action qualifier

13.2.5.53 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.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **actionQual** The action qualifier

13.2.5.54 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.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **actionQual** The action qualifier

13.2.5.55 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.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **actionQual** The action qualifier

13.2.5.56 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.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***actionQual*** The action qualifier

13.2.5.57 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.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***actionQual*** The action qualifier

13.2.5.58 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.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***actionQual*** The action qualifier

13.2.5.59 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.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***actionQual*** The action qualifier

13.2.5.60 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.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***actionQual*** The action qualifier

13.2.5.61 void PWM\_setChoppingClkFreq ([PWM\\_Handle](#) *pwmHandle*, const [PWM\\_ChoppingClkFreq\\_e](#) *clkFreq*)

Sets the pulse width modulation (PWM) chopping clock frequency.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***clkFreq*** The clock frequency

13.2.5.62 void PWM\_setChoppingDutyCycle ([PWM\\_Handle](#) *pwmHandle*, const [PWM\\_ChoppingDutyCycle\\_e](#) *dutyCycle*)

Sets the pulse width modulation (PWM) chopping clock duty cycle.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***dutyCycle*** The duty cycle

13.2.5.63 void PWM\_setChoppingPulseWidth ([PWM\\_Handle](#) *pwmHandle*, const [PWM\\_ChoppingPulseWidth\\_e](#) *pulseWidth*)

Sets the pulse width modulation (PWM) chopping clock pulse width.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***pulseWidth*** The pulse width

13.2.5.64 void PWM\_setClkDiv ([PWM\\_Handle](#) *pwmHandle*, const [PWM\\_ClkDiv\\_e](#) *clkDiv*)

Sets the pulse width modulation (PWM) clock divisor.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***clkDiv*** The clock divisor

13.2.5.65 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.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***pwmData*** The PWM data value



13.2.5.66 void PWM\_setCmpAHr ([PWM\\_Handle](#) *pwmHandle*, const uint16\_t *pwmData*)  
[inline]

Writes the pulse width modulation (PWM) data value to the Counter Compare A Hr hardware.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***pwmData*** The PWM high resolution data value

13.2.5.67 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.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***pwmData*** The PWM data value

13.2.5.68 void PWM\_setCount ([PWM\\_Handle](#) *pwmHandle*, const uint16\_t *count*)

Sets the pulse width modulation (PWM) count.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***count*** The count

13.2.5.69 void PWM\_setCounterMode ([PWM\\_Handle](#) *pwmHandle*, const  
[PWM\\_CounterMode\\_e](#) *counterMode*)

Sets the pulse width modulation (PWM) counter mode.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***counterMode*** The count mode

13.2.5.70 void PWM\_setDeadBandFallingEdgeDelay ([PWM\\_Handle](#) *pwmHandle*, const  
uint16\_t *delay*)

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

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***delay*** The delay

13.2.5.71 void PWM\_setDeadBandInputMode ([PWM\\_Handle](#) *pwmHandle*, const [PWM\\_DeadBandInputMode\\_e](#) *inputMode*)

Sets the pulse width modulation (PWM) deadband input mode.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***inputMode*** The input mode

13.2.5.72 void PWM\_setDeadBandOutputMode ([PWM\\_Handle](#) *pwmHandle*, const [PWM\\_DeadBandOutputMode\\_e](#) *outputMode*)

Sets the pulse width modulation (PWM) deadband output mode.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***outputMode*** The output mode

13.2.5.73 void PWM\_setDeadBandPolarity ([PWM\\_Handle](#) *pwmHandle*, const [PWM\\_DeadBandPolarity\\_e](#) *polarity*)

Sets the pulse width modulation (PWM) deadband polarity.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***polarity*** The polarity

13.2.5.74 void PWM\_setDeadBandRisingEdgeDelay ([PWM\\_Handle](#) *pwmHandle*, const uint16\_t *delay*)

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

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***delay*** The delay

13.2.5.75 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.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***selectFilter*** Select filter output if true

- ← **disableSync** Asynchronous if true
- ← **enableSoc** Enable SOC generation if true
- ← **generateSync** Generate SYNC if true

13.2.5.76 void PWM\_setDigitalCompareAEvent2 ([PWM\\_Handle](#) pwmHandle, const bool\_t selectFilter, const bool\_t disableSync)

Sets the pulse width modulation (PWM) digital compare A event 2 source parameters.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **selectFilter** Select filter output if true
- ← **disableSync** Asynchronous if true

13.2.5.77 void PWM\_setDigitalCompareBEvent1 ([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 B event 1 source parameters.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **selectFilter** Select filter output if true
- ← **disableSync** Asynchronous if true
- ← **enableSoc** Enable SOC generation if true
- ← **generateSync** Generate SYNC if true

13.2.5.78 void PWM\_setDigitalCompareBEvent2 ([PWM\\_Handle](#) pwmHandle, const bool\_t selectFilter, const bool\_t disableSync)

Sets the pulse width modulation (PWM) digital compare B event 2 source parameters.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **selectFilter** Select filter output if true
- ← **disableSync** Asynchronous if true

13.2.5.79 void PWM\_setDigitalCompareBlankingPulse ([PWM\\_Handle](#) pwmHandle, const [PWM\\_DigitalCompare\\_PulseSel\\_e](#) pulseSelect)

Sets the pulse width modulation (PWM) digital compare blanking pulse.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **input** The pulse selection

13.2.5.80 void PWM\_setDigitalCompareFilterOffset ([PWM\\_Handle](#) *pwmHandle*, const uint16\_t *offset*)

Sets the pulse width modulation (PWM) digital compare filter offset.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***offset*** The offset

13.2.5.81 void PWM\_setDigitalCompareFilterSource ([PWM\\_Handle](#) *pwmHandle*, const [PWM\\_DigitalCompare\\_FilterSrc\\_e](#) *input*)

Sets the pulse width modulation (PWM) digital compare filter source.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***input*** The filter's source

13.2.5.82 void PWM\_setDigitalCompareFilterWindow ([PWM\\_Handle](#) *pwmHandle*, const uint16\_t *window*)

Sets the pulse width modulation (PWM) digital compare filter offset.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***window*** The window

13.2.5.83 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.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***input*** Comparator input to change
- ← ***inputSel*** Input selection for designated input

13.2.5.84 void PWM\_setHighSpeedClkDiv ([PWM\\_Handle](#) *pwmHandle*, const [PWM\\_HspClkDiv\\_e](#) *clkDiv*)

Sets the pulse width modulation (PWM) high speed clock divisor.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***clkDiv*** The clock divisor

13.2.5.85 void PWM\_setHrControlMode ([PWM\\_Handle](#) pwmHandle, const [PWM\\_HrControlMode\\_e](#) controlMode)

Set the High Resolution Control Mode.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **edgeMode** The control mode HRPWM should use

13.2.5.86 void PWM\_setHrEdgeMode ([PWM\\_Handle](#) pwmHandle, const [PWM\\_HrEdgeMode\\_e](#) edgeMode)

Set the High Resolution Edge Mode.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **edgeMode** The edge mode HRPWM should use

13.2.5.87 void PWM\_setHrShadowMode ([PWM\\_Handle](#) pwmHandle, const [PWM\\_HrShadowMode\\_e](#) shadowMode)

Set the High Resolution Shadow Load Mode.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **edgeMode** The shadow load mode HRPWM should use

13.2.5.88 void PWM\_setIntMode ([PWM\\_Handle](#) pwmHandle, const [PWM\\_IntMode\\_e](#) intMode)

Sets the pulse width modulation (PWM) interrupt mode.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **intMode** The interrupt mode

13.2.5.89 void PWM\_setIntPeriod ([PWM\\_Handle](#) pwmHandle, const [PWM\\_IntPeriod\\_e](#) intPeriod)

Sets the pulse width modulation (PWM) interrupt period.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **intPeriod** The interrupt period

13.2.5.90 void PWM\_setLoadMode\_CmpA ([PWM\\_Handle](#) pwmHandle, const [PWM\\_LoadMode\\_e](#) loadMode)

Sets the pulse width modulation (PWM) load mode for CMPA.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **loadMode** The load mode

13.2.5.91 void PWM\_setLoadMode\_CmpB ([PWM\\_Handle](#) pwmHandle, const [PWM\\_LoadMode\\_e](#) loadMode)

Sets the pulse width modulation (PWM) load mode for CMPB.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **loadMode** The load mode

13.2.5.92 void PWM\_setOneShotTrip ([PWM\\_Handle](#) pwmHandle) [inline]

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

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle

13.2.5.93 void PWM\_setPeriod ([PWM\\_Handle](#) pwmHandle, const uint16\_t period)

Sets the pulse width modulation (PWM) period.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **period** The period

13.2.5.94 void PWM\_setPeriodHr ([PWM\\_Handle](#) pwmHandle, const uint16\_t period)

Sets the pulse width modulation (PWM) high resolution period.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **period** The period

13.2.5.95 void PWM\_setPeriodLoad ([PWM\\_Handle](#) *pwmHandle*, const [PWM\\_PeriodLoad\\_e](#) *periodLoad*)

Sets the pulse width modulation (PWM) period load mode.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***periodLoad*** The period load mode

13.2.5.96 void PWM\_setPhase ([PWM\\_Handle](#) *pwmHandle*, const uint16\_t *phase*)

Sets the pulse width modulation (PWM) phase.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***phase*** The phase

13.2.5.97 void PWM\_setPhaseDir ([PWM\\_Handle](#) *pwmHandle*, const [PWM\\_PhaseDir\\_e](#) *phaseDir*)

Sets the pulse width modulation (PWM) phase direction.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***phaseDir*** The phase direction

13.2.5.98 void PWM\_setRunMode ([PWM\\_Handle](#) *pwmHandle*, const [PWM\\_RunMode\\_e](#) *runMode*)

Sets the pulse width modulation (PWM) run mode.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***runMode*** The run mode

13.2.5.99 void PWM\_setShadowMode\_CmpA ([PWM\\_Handle](#) *pwmHandle*, const [PWM\\_ShadowMode\\_e](#) *shadowMode*)

Sets the pulse width modulation (PWM) shadow mode for CMPA.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***shadowMode*** The shadow mode

13.2.5.100 `void PWM_setShadowMode_CmpB (PWM_Handle pwmHandle, const PWM_ShadowMode_e shadowMode)`

Sets the pulse width modulation (PWM) shadow mode for CMPB.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **shadowMode** The shadow mode

13.2.5.101 `void PWM_setSocAPeriod (PWM_Handle pwmHandle, const PWM_SocPeriod_e intPeriod)`

Sets the pulse width modulation (PWM) start of conversion (SOC) A interrupt period.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **intPeriod** The interrupt period

13.2.5.102 `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.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **pulseSrc** The interrupt pulse source

13.2.5.103 `void PWM_setSocBPeriod (PWM_Handle pwmHandle, const PWM_SocPeriod_e intPeriod)`

Sets the pulse width modulation (PWM) start of conversion (SOC) B interrupt period.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **intPeriod** The interrupt period

13.2.5.104 `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.

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **pulseSrc** The interrupt pulse source



**13.2.5.105** void PWM\_setSwSync ([PWM\\_Handle](#) pwmHandle)

Sets the pulse width modulation (PWM) software sync.

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

**13.2.5.106** void PWM\_setSyncMode ([PWM\\_Handle](#) pwmHandle, const [PWM\\_SyncMode\\_e](#) syncMode)

Sets the pulse width modulation (PWM) sync mode.

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

← **syncMode** The sync mode

**13.2.5.107** 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).

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

← **tripZoneEvent** The trip zone digital compare event

**13.2.5.108** 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).

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

← **tripZoneEvent** The trip zone digital compare event

**13.2.5.109** 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).

**Parameters:**

← **pwmHandle** The pulse width modulation (PWM) object handle

← **tripZoneEvent** The trip zone digital compare event

13.2.5.110 `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).

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **tripZoneEvent** The trip zone digital compare event

13.2.5.111 `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).

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **tripZoneState** The trip zone state

13.2.5.112 `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).

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **tripZoneState** The trip zone state

13.2.5.113 `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).

**Parameters:**

- ← **pwmHandle** The pulse width modulation (PWM) object handle
- ← **tripZoneState** The trip zone state

13.2.5.114 `void 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 (DCBEVT2).

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***tripZoneState*** The trip zone state

13.2.5.115 `void PWM_setTripZoneState_TZA (PWM_Handle pwmHandle, const PWM_TripZoneState_e tripZoneState)`

Sets the pulse width modulation (PWM) trip zone state for Output A (TZA).

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***tripZoneState*** The trip zone state

13.2.5.116 `void PWM_setTripZoneState_TZB (PWM_Handle pwmHandle, const PWM_TripZoneState_e tripZoneState)`

Sets the pulse width modulation (PWM) trip zone state for Output B (TZB).

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***tripZoneState*** The trip zone state

13.2.5.117 `void 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.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***pwmData*** The PWM data value

13.2.5.118 `void 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.

**Parameters:**

- ← ***pwmHandle*** The pulse width modulation (PWM) object handle
- ← ***pwmData*** The PWM data value



## 14 Power Control (PWR)

<a href="#">Introduction .....</a>	<a href="#">229</a>
<a href="#">API Functions .....</a>	<a href="#">229</a>

### 14.1 Introduction

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.

### 14.2 PWR

#### Data Structures

- [\\_PWR\\_Obj\\_](#)

#### Defines

- [PWR\\_BASE\\_ADDR](#)
- [PWR\\_BORCFG\\_BORENZ\\_BITS](#)
- [PWR\\_LPMCR0\\_LPM\\_BITS](#)
- [PWR\\_LPMCR0\\_QUALSTDBY\\_BITS](#)
- [PWR\\_LPMCR0\\_WDINTE\\_BITS](#)

#### Enumerations

- [PWR\\_LowPowerMode\\_e](#)
- [PWR\\_NumStandByClocks\\_e](#)

#### 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)
```

## 14.2.1 Data Structure Documentation

### 14.2.1.1 \_PWR\_Obj\_

**Definition:**

```
typedef struct  
{  
    uint16_t BORCFG;  
    uint16_t rsvd_1[26264];  
    uint16_t LPMCR0;  
}  
_PWR_Obj_
```

**Members:**

***BORCFG*** BOR (Brown Out Reset) Configuration Register.  
***rsvd\_1***  
***LPMCR0***

**Description:**

Defines the power (PWR) object.

## 14.2.2 Define Documentation

### 14.2.2.1 PWR\_BASE\_ADDR

**Definition:**

```
#define PWR_BASE_ADDR
```

**Description:**

Defines the base address of the power (PWR) registers.

### 14.2.2.2 PWR\_BORCFG\_BORENZ\_BITS

**Definition:**

```
#define PWR_BORCFG_BORENZ_BITS
```

**Description:**

Defines the location of the BORENZ bits in the BORCFG register.

### 14.2.2.3 PWR\_LPMCR0\_LPM\_BITS

**Definition:**

```
#define PWR_LPMCR0_LPM_BITS
```

**Description:**

Defines the location of the LPM bits in the LPMCR0 register.

#### 14.2.2.4 PWR\_LPMCR0\_QUALSTDBY\_BITS

**Definition:**

```
#define PWR_LPMCR0_QUALSTDBY_BITS
```

**Description:**

Defines the location of the QUALSTDBY bits in the LPMCR0 register.

#### 14.2.2.5 PWR\_LPMCR0\_WDINTE\_BITS

**Definition:**

```
#define PWR_LPMCR0_WDINTE_BITS
```

**Description:**

Defines the location of the WDINTE bits in the LPMCR0 register.

### 14.2.3 Typedef Documentation

#### 14.2.3.1 PWR\_Handle

**Definition:**

```
typedef struct PWR_Obj *PWR_Handle
```

**Description:**

Defines the power (PWR) handle.

#### 14.2.3.2 PWR\_Obj

**Definition:**

```
typedef struct _PWR_Obj_ PWR_Obj
```

**Description:**

Defines the power (PWR) object.

### 14.2.4 Enumeration Documentation

#### 14.2.4.1 PWR\_LowPowerMode\_e

**Description:**

Enumeration to define the power (PWR) low power modes.

**Enumerators:**

***PWR\_LowPowerMode\_Idle*** Denotes the idle mode.

***PWR\_LowPowerMode\_Standby*** Denotes the standby mode.

***PWR\_LowPowerMode\_Halt*** Denotes the halt mode.

#### 14.2.4.2 PWR\_NumStandByClocks\_e

**Description:**

Enumeration to define the power (PWR) number of standby clock cycles.

**Enumerators:**

***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.

***PWR\_NumStandByClocks\_29*** Denotes 29 standby clock cycles.

***PWR\_NumStandByClocks\_30*** Denotes 30 standby clock cycles.

***PWR\_NumStandByClocks\_31*** Denotes 31 standby clock cycles.

***PWR\_NumStandByClocks\_32*** Denotes 32 standby clock cycles.

***PWR\_NumStandByClocks\_33*** Denotes 33 standby clock cycles.

***PWR\_NumStandByClocks\_34*** Denotes 34 standby clock cycles.

***PWR\_NumStandByClocks\_35*** Denotes 35 standby clock cycles.

***PWR\_NumStandByClocks\_36*** Denotes 36 standby clock cycles.

***PWR\_NumStandByClocks\_37*** 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.

## 14.2.5 Function Documentation

### 14.2.5.1 PWR\_disableBrownOutReset

Disables the brownout reset functions.

**Prototype:**

```
void  
PWR_disableBrownOutReset (PWR_Handle pwrHandle)
```

**Parameters:**

← ***pwrHandle*** The power (PWR) object handle

### 14.2.5.2 void PWR\_disableWatchDogInt (PWR\_Handle pwrHandle)

Disables the watchdog interrupt.

**Parameters:**

← **pwrHandle** The power (PWR) object handle

14.2.5.3 void PWR\_enableBrownOutReset ([PWR\\_Handle](#) pwrHandle)

Enables the brownout reset functions.

**Parameters:**

← **pwrHandle** The power (PWR) object handle

14.2.5.4 void PWR\_enableWatchDogInt ([PWR\\_Handle](#) pwrHandle)

Enables the watchdog interrupt.

**Parameters:**

← **pwrHandle** The power (PWR) object handle

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

Initializes the power (PWR) object handle.

**Parameters:**

← **pMemory** A pointer to the base address of the PWR registers

← **numBytes** The number of bytes allocated for the PWR object, bytes

**Returns:**

The power (PWR) object handle

14.2.5.6 void PWR\_setLowPowerMode ([PWR\\_Handle](#) pwrHandle, const [PWR\\_LowPowerMode\\_e](#) lowPowerMode)

Sets the low power mode.

**Parameters:**

← **pwrHandle** The power (PWR) object handle

← **lowPowerMode** The low power mode

14.2.5.7 void PWR\_setNumStandByClocks ([PWR\\_Handle](#) pwrHandle, const [PWR\\_NumStandByClocks\\_e](#) numClkCycles)

Sets the number of standby clock cycles.

**Parameters:**

← **pwrHandle** The power (PWR) object handle

← **numClkCycles** The number of standby clock cycles

## 15 Serial Communications Interface (SCI)

Introduction .....	235
API Functions .....	235

### 15.1 Introduction

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.

### 15.2 SCI

#### Data Structures

- [\\_SCI\\_Obj](#)

#### Defines

- [SCI\\_SCICCR\\_CHAR\\_LENGTH\\_BITS](#)
- [SCI\\_SCICCR\\_LB\\_ENA\\_BITS](#)
- [SCI\\_SCICCR\\_MODE\\_BITS](#)
- [SCI\\_SCICCR\\_PARITY\\_BITS](#)
- [SCI\\_SCICCR\\_PARITY\\_ENA\\_BITS](#)
- [SCI\\_SCICCR\\_STOP\\_BITS](#)
- [SCI\\_SCICTL1\\_RESET\\_BITS](#)
- [SCI\\_SCICTL1\\_RX\\_ERR\\_INT\\_ENA\\_BITS](#)
- [SCI\\_SCICTL1\\_RXENA\\_BITS](#)
- [SCI\\_SCICTL1\\_SLEEP\\_BITS](#)
- [SCI\\_SCICTL1\\_TXENA\\_BITS](#)
- [SCI\\_SCICTL1\\_TXWAKE\\_BITS](#)
- [SCI\\_SCICTL2\\_RX\\_INT\\_ENA\\_BITS](#)
- [SCI\\_SCICTL2\\_TX\\_INT\\_ENA\\_BITS](#)
- [SCI\\_SCICTL2\\_TXEMPTY\\_BITS](#)
- [SCI\\_SCICTL2\\_TXRDY\\_BITS](#)
- [SCI\\_SCIFFCT\\_ABD\\_BITS](#)
- [SCI\\_SCIFFCT\\_ABDCLR\\_BITS](#)
- [SCI\\_SCIFFCT\\_CDC\\_BITS](#)
- [SCI\\_SCIFFCT\\_DELAY\\_BITS](#)
- [SCI\\_SCIFFRX\\_FIFO\\_OVF\\_BITS](#)

- [SCI\\_SCIFFRX\\_FIFO\\_OVFCLR\\_BITS](#)
- [SCI\\_SCIFFRX\\_FIFO\\_RESET\\_BITS](#)
- [SCI\\_SCIFFRX\\_FIFO\\_ST\\_BITS](#)
- [SCI\\_SCIFFRX\\_IENA\\_BITS](#)
- [SCI\\_SCIFFRX\\_IL\\_BITS](#)
- [SCI\\_SCIFFRX\\_INT\\_BITS](#)
- [SCI\\_SCIFFRX\\_INTCLR\\_BITS](#)
- [SCI\\_SCIFFTX\\_CHAN\\_RESET\\_BITS](#)
- [SCI\\_SCIFFTX\\_FIFO\\_ENA\\_BITS](#)
- [SCI\\_SCIFFTX\\_FIFO\\_RESET\\_BITS](#)
- [SCI\\_SCIFFTX\\_FIFO\\_ST\\_BITS](#)
- [SCI\\_SCIFFTX\\_IENA\\_BITS](#)
- [SCI\\_SCIFFTX\\_IL\\_BITS](#)
- [SCI\\_SCIFFTX\\_INT\\_BITS](#)
- [SCI\\_SCIFFTX\\_INTCLR\\_BITS](#)
- [SCI\\_SCIRXST\\_BRKDT\\_BITS](#)
- [SCI\\_SCIRXST\\_FE\\_BITS](#)
- [SCI\\_SCIRXST\\_OE\\_BITS](#)
- [SCI\\_SCIRXST\\_PE\\_BITS](#)
- [SCI\\_SCIRXST\\_RXERROR\\_BITS](#)
- [SCI\\_SCIRXST\\_RXRDY\\_BITS](#)
- [SCI\\_SCIRXST\\_RXWAKE\\_BITS](#)
- [SCIA\\_BASE\\_ADDR](#)

## Enumerations

- [SCI\\_BaudRate\\_e](#)
- [SCI\\_CharLength\\_e](#)
- [SCI\\_FifoLevel\\_e](#)
- [SCI\\_FifoStatus\\_e](#)
- [SCI\\_Mode\\_e](#)
- [SCI\\_NumStopBits\\_e](#)
- [SCI\\_Parity\\_e](#)
- [SCI\\_Priority\\_e](#)

## 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)

## 15.2.1 Data Structure Documentation

### 15.2.1.1 \_SCI\_Obj\_

**Definition:**

```
typedef struct
{
    uint16_t SCICCR;
    uint16_t SCICTL1;
    uint16_t SCIHBAUD;
    uint16_t SCILBAUD;
    uint16_t SCICTL2;
    uint16_t SCIRXST;
    uint16_t SCIRXEMU;
    uint16_t SCIRXBUF;
    uint16_t rsvd_1;
    uint16_t SCITXBUF;
    uint16_t SCIFFTX;
    uint16_t SCIFFRX;
    uint16_t SCIFFCT;
    uint16_t rsvd_2[2];
    uint16_t SCIPRI;
}
_SCI_Obj_
```

**Members:**

**SCICCR** SCI Configuration Control Register.  
**SCICTL1** SCI Control Register 1.  
**SCIHBAUD** SCI Baud Register, High Bits.  
**SCILBAUD** SCI Baud Register, Low Bits.  
**SCICTL2** SCI Control Register 2.  
**SCIRXST** SCI Receive Status Register.  
**SCIRXEMU** SCI Receive Emulation Data Buffer Register.  
**SCIRXBUF** SCI Receive Data Buffer Register.  
**rsvd\_1** Reserved.  
**SCITXBUF** SCI Transmit Data Buffer Register.  
**SCIFFTX** SCI FIFO Transmit Register.  
**SCIFFRX** SCI FIFO Receive Register.  
**SCIFFCT** SCI FIFO Control Register.  
**rsvd\_2** Reserved.  
**SCIPRI** SCI Priority Register.

**Description:**

Defines the serial communications interface (SCI) object.

## 15.2.2 Define Documentation

### 15.2.2.1 SCI\_SCICCR\_CHAR\_LENGTH\_BITS

**Definition:**

```
#define SCI_SCICCR_CHAR_LENGTH_BITS
```

**Description:**

Defines the location of the SCICCHAR2-0 bits in the SCICCR register.

### 15.2.2.2 SCI\_SCICCR\_LB\_ENA\_BITS

**Definition:**

```
#define SCI_SCICCR_LB_ENA_BITS
```

**Description:**

Defines the location of the LOOP BACK ENA bits in the SCICCR register.

### 15.2.2.3 SCI\_SCICCR\_MODE\_BITS

**Definition:**

```
#define SCI_SCICCR_MODE_BITS
```

**Description:**

Defines the location of the ADDR/IDLE MODE bits in the SCICCR register.

### 15.2.2.4 SCI\_SCICCR\_PARITY\_BITS

**Definition:**

```
#define SCI_SCICCR_PARITY_BITS
```

**Description:**

Defines the location of the EVEN/ODD PARITY bits in the SCICCR register.

### 15.2.2.5 SCI\_SCICCR\_PARITY\_ENA\_BITS

**Definition:**

```
#define SCI_SCICCR_PARITY_ENA_BITS
```

**Description:**

Defines the location of the PARITY ENABLE bits in the SCICCR register.

#### 15.2.2.6 SCI\_SCICCR\_STOP\_BITS

**Definition:**

```
#define SCI_SCICCR_STOP_BITS
```

**Description:**

Defines the location of the STOP bits in the SCICCR register.

#### 15.2.2.7 SCI\_SCICTL1\_RESET\_BITS

**Definition:**

```
#define SCI_SCICTL1_RESET_BITS
```

**Description:**

Defines the location of the SW RESET bits in the SCICTL1 register.

#### 15.2.2.8 SCI\_SCICTL1\_RX\_ERR\_INT\_ENA\_BITS

**Definition:**

```
#define SCI_SCICTL1_RX_ERR_INT_ENA_BITS
```

**Description:**

Defines the location of the RX ERR INT ENA bits in the SCICTL1 register.

#### 15.2.2.9 SCI\_SCICTL1\_RXENA\_BITS

**Definition:**

```
#define SCI_SCICTL1_RXENA_BITS
```

**Description:**

Defines the location of the RXENA bits in the SCICTL1 register.

#### 15.2.2.10 SCI\_SCICTL1\_SLEEP\_BITS

**Definition:**

```
#define SCI_SCICTL1_SLEEP_BITS
```

**Description:**

Defines the location of the SLEEP bits in the SCICTL1 register.

#### 15.2.2.11 SCI\_SCICTL1\_TXENA\_BITS

**Definition:**

```
#define SCI_SCICTL1_TXENA_BITS
```

**Description:**

Defines the location of the TXENA bits in the SCICTL1 register.



#### 15.2.2.12 SCI\_SCICTL1\_TXWAKE\_BITS

**Definition:**

```
#define SCI_SCICTL1_TXWAKE_BITS
```

**Description:**

Defines the location of the TXWAKE bits in the SCICTL1 register.

#### 15.2.2.13 SCI\_SCICTL2\_RX\_INT\_ENA\_BITS

**Definition:**

```
#define SCI_SCICTL2_RX_INT_ENA_BITS
```

**Description:**

Defines the location of the RX/BK INT ENA bits in the SCICTL2 register.

#### 15.2.2.14 SCI\_SCICTL2\_TX\_INT\_ENA\_BITS

**Definition:**

```
#define SCI_SCICTL2_TX_INT_ENA_BITS
```

**Description:**

Defines the location of the TX INT ENA bits in the SCICTL2 register.

#### 15.2.2.15 SCI\_SCICTL2\_TXEMPTY\_BITS

**Definition:**

```
#define SCI_SCICTL2_TXEMPTY_BITS
```

**Description:**

Defines the location of the TX EMPTY bits in the SCICTL2 register.

#### 15.2.2.16 SCI\_SCICTL2\_TXRDY\_BITS

**Definition:**

```
#define SCI_SCICTL2_TXRDY_BITS
```

**Description:**

Defines the location of the RX EMPTY bits in the SCICTL2 register.

#### 15.2.2.17 SCI\_SCIFFCT\_ABD\_BITS

**Definition:**

```
#define SCI_SCIFFCT_ABD_BITS
```

**Description:**

Defines the location of the ABD bits in the SCIFFCT register.

#### 15.2.2.18 SCI\_SCIFFCT\_ABDCLR\_BITS

**Definition:**

```
#define SCI_SCIFFCT_ABDCLR_BITS
```

**Description:**

Defines the location of the ABD CLR bits in the SCIFFCT register.

#### 15.2.2.19 SCI\_SCIFFCT\_CDC\_BITS

**Definition:**

```
#define SCI_SCIFFCT_CDC_BITS
```

**Description:**

Defines the location of the CDC bits in the SCIFFCT register.

#### 15.2.2.20 SCI\_SCIFFCT\_DELAY\_BITS

**Definition:**

```
#define SCI_SCIFFCT_DELAY_BITS
```

**Description:**

Defines the location of the FFTXDLY7-0 bits in the SCIFFCT register.

#### 15.2.2.21 SCI\_SCIFFRX\_FIFO\_OVF\_BITS

**Definition:**

```
#define SCI_SCIFFRX_FIFO_OVF_BITS
```

**Description:**

Defines the location of the RXFFOVF bits in the SCIFFRX register.

#### 15.2.2.22 SCI\_SCIFFRX\_FIFO\_OVFCLR\_BITS

**Definition:**

```
#define SCI_SCIFFRX_FIFO_OVFCLR_BITS
```

**Description:**

Defines the location of the RXFFOVF CLR bits in the SCIFFRX register.

#### 15.2.2.23 SCI\_SCIFFRX\_FIFO\_RESET\_BITS

**Definition:**

```
#define SCI_SCIFFRX_FIFO_RESET_BITS
```

**Description:**

Defines the location of the RXFIFO Reset bits in the SCIFFRX register.

#### 15.2.2.24 SCI\_SCIFFRX\_FIFO\_ST\_BITS

**Definition:**

```
#define SCI_SCIFFRX_FIFO_ST_BITS
```

**Description:**

Defines the location of the RXFFST4-0 bits in the SCIFFRX register.

#### 15.2.2.25 SCI\_SCIFFRX\_IENA\_BITS

**Definition:**

```
#define SCI_SCIFFRX_IENA_BITS
```

**Description:**

Defines the location of the RXFFIENA bits in the SCIFFRX register.

#### 15.2.2.26 SCI\_SCIFFRX\_IL\_BITS

**Definition:**

```
#define SCI_SCIFFRX_IL_BITS
```

**Description:**

Defines the location of the RXFFIL4-0 bits in the SCIFFRX register.

#### 15.2.2.27 SCI\_SCIFFRX\_INT\_BITS

**Definition:**

```
#define SCI_SCIFFRX_INT_BITS
```

**Description:**

Defines the location of the RXFFINT flag bits in the SCIFFRX register.

#### 15.2.2.28 SCI\_SCIFFRX\_INTCLR\_BITS

**Definition:**

```
#define SCI_SCIFFRX_INTCLR_BITS
```

**Description:**

Defines the location of the RXFFINT CLR bits in the SCIFFRX register.

#### 15.2.2.29 SCI\_SCIFFTX\_CHAN\_RESET\_BITS

**Definition:**

```
#define SCI_SCIFFTX_CHAN_RESET_BITS
```

**Description:**

Defines the location of the SCIRST bits in the SCIFFTX register.

#### 15.2.2.30 SCI\_SCIFFTX\_FIFO\_ENA\_BITS

**Definition:**

```
#define SCI_SCIFFTX_FIFO_ENA_BITS
```

**Description:**

Defines the location of the SCIFFENA bits in the SCIFFTX register.

#### 15.2.2.31 SCI\_SCIFFTX\_FIFO\_RESET\_BITS

**Definition:**

```
#define SCI_SCIFFTX_FIFO_RESET_BITS
```

**Description:**

Defines the location of the TXFIFO Reset bits in the SCIFFTX register.

#### 15.2.2.32 SCI\_SCIFFTX\_FIFO\_ST\_BITS

**Definition:**

```
#define SCI_SCIFFTX_FIFO_ST_BITS
```

**Description:**

Defines the location of the TXFFST4-0 bits in the SCIFFTX register.

#### 15.2.2.33 SCI\_SCIFFTX\_IENA\_BITS

**Definition:**

```
#define SCI_SCIFFTX_IENA_BITS
```

**Description:**

Defines the location of the TXFFIENA bits in the SCIFFTX register.

#### 15.2.2.34 SCI\_SCIFFTX\_IL\_BITS

**Definition:**

```
#define SCI_SCIFFTX_IL_BITS
```

**Description:**

Defines the location of the TXFFIL4-0 bits in the SCIFFTX register.

#### 15.2.2.35 SCI\_SCIFFTX\_INT\_BITS

**Definition:**

```
#define SCI_SCIFFTX_INT_BITS
```

**Description:**

Defines the location of the TXFFINT flag bits in the SCIFFTX register.

#### 15.2.2.36 SCI\_SCIFFTX\_INTCLR\_BITS

**Definition:**

```
#define SCI_SCIFFTX_INTCLR_BITS
```

**Description:**

Defines the location of the TXFFINT CLR bits in the SCIFFTX register.

#### 15.2.2.37 SCI\_SCIRXST\_BRKDT\_BITS

**Definition:**

```
#define SCI_SCIRXST_BRKDT_BITS
```

**Description:**

Defines the location of the BRKDT bits in the SCIRXST register.

#### 15.2.2.38 SCI\_SCIRXST\_FE\_BITS

**Definition:**

```
#define SCI_SCIRXST_FE_BITS
```

**Description:**

Defines the location of the FE bits in the SCIRXST register.

#### 15.2.2.39 SCI\_SCIRXST\_OE\_BITS

**Definition:**

```
#define SCI_SCIRXST_OE_BITS
```

**Description:**

Defines the location of the OE bits in the SCIRXST register.

#### 15.2.2.40 SCI\_SCIRXST\_PE\_BITS

**Definition:**

```
#define SCI_SCIRXST_PE_BITS
```

**Description:**

Defines the location of the PE bits in the SCIRXST register.

#### 15.2.2.41 SCI\_SCIRXST\_RXERROR\_BITS

**Definition:**

```
#define SCI_SCIRXST_RXERROR_BITS
```

**Description:**

Defines the location of the RX ERROR bits in the SCIRXST register.

#### 15.2.2.42 SCI\_SCIRXST\_RXRDY\_BITS

**Definition:**

```
#define SCI_SCIRXST_RXRDY_BITS
```

**Description:**

Defines the location of the RXRDY bits in the SCIRXST register.

#### 15.2.2.43 SCI\_SCIRXST\_RXWAKE\_BITS

**Definition:**

```
#define SCI_SCIRXST_RXWAKE_BITS
```

**Description:**

Defines the location of the RXWAKE bits in the SCIRXST register.

#### 15.2.2.44 SCIA\_BASE\_ADDR

**Definition:**

```
#define SCIA_BASE_ADDR
```

**Description:**

Defines the base address of the serial communications interface (SCI) A registers.

### 15.2.3 Typedef Documentation

#### 15.2.3.1 SCI\_Handle

**Definition:**

```
typedef struct SCI\_Obj *SCI\_Handle
```

**Description:**

Defines the serial communications interface (SCI) handle.

#### 15.2.3.2 SCI\_Obj

**Definition:**

```
typedef struct \_SCI\_Obj SCI\_Obj
```

**Description:**

Defines the serial communications interface (SCI) object.

## 15.2.4 Enumeration Documentation

### 15.2.4.1 SCI\_BaudRate\_e

**Description:**

Enumeration to define the serial communications interface (SCI) baud rates. This enumeration assume a device clock of 60Mhz and a LSPCLK of 15MHz.

**Enumerators:**

- 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.

### 15.2.4.2 SCI\_CharLength\_e

**Description:**

Enumeration to define the serial communications interface (SCI) character lengths.

**Enumerators:**

- 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.

### 15.2.4.3 SCI\_FifoLevel\_e

**Description:**

Enumeration to define the serial communications interface (SCI) FIFO level.

**Enumerators:**

- 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.

### 15.2.4.4 SCI\_FifoStatus\_e

**Description:**

Enumeration to define the serial communications interface (SCI) FIFO status.

**Enumerators:**

- 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.

#### 15.2.4.5 SCI\_Mode\_e

**Description:**

Enumeration to define the serial communications interface (SCI) multiprocessor protocol mode.

**Enumerators:**

- SCI\_Mode\_IdleLine*** Denotes idle-line mode protocol.
- SCI\_Mode\_AddressBit*** Denotes address-bit mode protocol.

#### 15.2.4.6 SCI\_NumStopBits\_e

**Description:**

Enumeration to define the serial communications interface (SCI) number of stop bits.

**Enumerators:**

- SCI\_NumStopBits\_One*** Denotes 1 stop bit.
- SCI\_NumStopBits\_Two*** Denotes 2 stop bits.

#### 15.2.4.7 SCI\_Parity\_e

**Description:**

Enumeration to define the serial communications interface (SCI) parity.

**Enumerators:**

- SCI\_Parity\_Odd*** Denotes odd parity.
- SCI\_Parity\_Even*** Denotes even parity.

#### 15.2.4.8 SCI\_Priority\_e

**Description:**

Enumeration to define the serial communications interface (SCI) emulation suspend priority.

**Enumerators:**

- 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.



## 15.2.5 Function Documentation

### 15.2.5.1 SCI\_clearAutoBaudDetect

Clears the auto baud detect mode.

**Prototype:**

```
void  
SCI_clearAutoBaudDetect(SCI_Handle sciHandle)
```

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

### 15.2.5.2 void SCI\_clearRxFifoInt (SCI\_Handle sciHandle)

Clears the Rx FIFO interrupt flag.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

### 15.2.5.3 void SCI\_clearRxFifoOvf (SCI\_Handle sciHandle)

Clears the Rx FIFO overflow flag.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

### 15.2.5.4 void SCI\_clearTxFifoInt (SCI\_Handle sciHandle)

Clears the Tx FIFO interrupt flag.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

### 15.2.5.5 void SCI\_disable (SCI\_Handle sciHandle)

Disables the serial communications interface (SCI) interrupt.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

#### 15.2.5.6 void SCI\_disableAutoBaudAlign (SCI\_Handle sciHandle)

Disable the serial communications interface (SCI) auto baud alignment.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

#### 15.2.5.7 void SCI\_disableFifoEnh (SCI\_Handle sciHandle)

Disables the serial communications interface (SCI) FIFO enhancements.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

#### 15.2.5.8 void SCI\_disableLoopBack (SCI\_Handle sciHandle)

Disables the serial peripheral interface (SCI) loop back mode.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

#### 15.2.5.9 void SCI\_disableParity (SCI\_Handle sciHandle)

Disable the parity.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

#### 15.2.5.10 void SCI\_disableRx (SCI\_Handle sciHandle)

Disables the serial communications interface (SCI) master/slave receive mode.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

#### 15.2.5.11 void SCI\_disableRxErrorInt (SCI\_Handle sciHandle)

Disables the serial communications interface (SCI) receive error interrupt.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

**15.2.5.12 void SCI\_disableRxFifoInt (SCI\_Handle sciHandle)**

Disables the serial communications interface (SCI) receive FIFO interrupt.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

**15.2.5.13 void SCI\_disableRxInt (SCI\_Handle sciHandle)**

Disables the serial communications interface (SCI) receive interrupt.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

**15.2.5.14 void SCI\_disableSleep (SCI\_Handle sciHandle)**

Disables the serial communications interface (SCI) sleep mode.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

**15.2.5.15 void SCI\_disableTx (SCI\_Handle sciHandle)**

Disables the serial communications interface (SCI) master/slave transmit mode.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

**15.2.5.16 void SCI\_disableTxFifoInt (SCI\_Handle sciHandle)**

Disables the serial communications interface (SCI) transmit FIFO interrupt.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

**15.2.5.17 void SCI\_disableTxInt (SCI\_Handle sciHandle)**

Disables the serial communications interface (SCI) transmit interrupt.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

#### 15.2.5.18 void SCI\_disableTxWake (SCI\_Handle sciHandle)

Disables the serial communications interface (SCI) wakeup method.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

#### 15.2.5.19 void SCI\_enable (SCI\_Handle sciHandle)

Enables the serial communications interface (SCI).

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

#### 15.2.5.20 void SCI\_enableAutoBaudAlign (SCI\_Handle sciHandle)

Enable the serial communications interface (SCI) auto baud alignment.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

#### 15.2.5.21 void SCI\_enableFifoEnh (SCI\_Handle sciHandle)

Enables the serial communications interface (SCI) FIFO enhancements.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

#### 15.2.5.22 void SCI\_enableLoopBack (SCI\_Handle sciHandle)

Enables the serial peripheral interface (SCI) loop back mode.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

#### 15.2.5.23 void SCI\_enableParity (SCI\_Handle sciHandle)

Enables the serial peripheral interface (SCI) parity.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

**15.2.5.24 void SCI\_enableRx (SCI\_Handle sciHandle)**

Enables the serial peripheral interface (SCI) receiver.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

**15.2.5.25 void SCI\_enableRxErrorInt (SCI\_Handle sciHandle)**

Enables the serial communications interface (SCI) receive error interrupt.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

**15.2.5.26 void SCI\_enableRxFifoInt (SCI\_Handle sciHandle)**

Enables the serial communications interface (SCI) receive FIFO interrupt.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

**15.2.5.27 void SCI\_enableRxInt (SCI\_Handle sciHandle)**

Enables the serial communications interface (SCI) receive interrupt.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

**15.2.5.28 void SCI\_enableSleep (SCI\_Handle sciHandle)**

Enables the serial communications interface (SCI) sleep mode.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

**15.2.5.29 void SCI\_enableTx (SCI\_Handle sciHandle)**

Enables the serial communications interface (SCI) master/slave transmit mode.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

#### 15.2.5.30 void SCI\_enableTxFifoInt (SCI\_Handle sciHandle)

Enables the serial communications interface (SCI) transmit FIFO interrupt.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

#### 15.2.5.31 void SCI\_enableTxInt (SCI\_Handle sciHandle)

Enables the serial communications interface (SCI) transmit interrupt.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

#### 15.2.5.32 void SCI\_enableTxWake (SCI\_Handle sciHandle)

Enables the serial communications interface (SCI) wakeup method.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

#### 15.2.5.33 uint16\_t SCI\_getData (SCI\_Handle sciHandle) [inline]

Reads data from the serial communications interface (SCI).

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

**Returns:**

The received data value

#### 15.2.5.34 uint16\_t SCI\_getDataBlocking (SCI\_Handle sciHandle)

Gets data from the serial communications interface (Blocking).

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

**Returns:**

Data from the serial peripheral

**15.2.5.35** `uint16_t SCI_getDataNonBlocking (SCI_Handle sciHandle, uint16_t * success)`

Read data from the serial communications interface (Non-Blocking).

**Parameters:**

- ← **sciHandle** The serial communications interface (SCI) object handle
- **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

**15.2.5.36** `SCI_FifoStatus_e SCI_getRxFifoStatus (SCI_Handle sciHandle)`

Gets the serial communications interface (SCI) receive FIFO status.

**Parameters:**

- ← **sciHandle** The serial communications interface (SCI) object handle

**Returns:**

The receive FIFO status

**15.2.5.37** `SCI_FifoStatus_e SCI_getTxFifoStatus (SCI_Handle sciHandle)`

Gets the serial communications interface (SCI) transmit FIFO status.

**Parameters:**

- ← **sciHandle** The serial communications interface (SCI) object handle

**Returns:**

The transmit FIFO status

**15.2.5.38** `SCI_Handle SCI_init (void * pMemory, const size_t numBytes)`

Initializes the serial communications interface (SCI) object handle.

**Parameters:**

- ← **pMemory** A pointer to the base address of the SCI registers
- ← **numBytes** The number of bytes allocated for the SCI object, bytes

**Returns:**

The serial communications interface (SCI) object handle

#### 15.2.5.39 bool\_t SCI\_isRxDataReady (SCI\_Handle sciHandle) [inline]

Determines if the serial communications interface (SCI) has receive data ready.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

**Returns:**

The receive data status

#### 15.2.5.40 bool\_t SCI\_isTxReady (SCI\_Handle sciHandle) [inline]

Determines if the serial communications interface (SCI) is ready to transmit.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

**Returns:**

The transmit status

#### 15.2.5.41 void SCI\_putData (SCI\_Handle sciHandle, const uint16\_t data) [inline]

Writes data to the serial communications interface (SCI).

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

← **data** The data value

#### 15.2.5.42 void SCI\_putDataBlocking (SCI\_Handle sciHandle, uint16\_t data)

Writes data to the serial communications interface (Blocking).

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

← **data** The data value

#### 15.2.5.43 uint16\_t SCI\_putDataNonBlocking (SCI\_Handle sciHandle, uint16\_t data)

Writes data to the serial communications interface (Non-Blocking).

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

← **data** The data value

**Returns:**

True on successful write, false if no space is available in the transmit buffer



**15.2.5.44 void SCI\_reset (SCI\_Handle sciHandle)**

Resets the serial communications interface (SCI).

**Parameters:**

← **sciHandle** The serial communication interface (SCI) object handle

**15.2.5.45 void SCI\_resetChannels (SCI\_Handle sciHandle)**

Resets the serial communications interface (SCI) transmit and receive channels.

**Parameters:**

← **sciHandle** The serial communication interface (SCI) object handle

**15.2.5.46 void SCI\_resetRxFifo (SCI\_Handle sciHandle)**

Resets the serial communications interface (SCI) receive FIFO.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

**15.2.5.47 void SCI\_resetTxFifo (SCI\_Handle sciHandle)**

Resets the serial communications interface (SCI) transmit FIFO.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

**15.2.5.48 void SCI\_setBaudRate (SCI\_Handle sciHandle, const SCI\_BaudRate\_e baudRate)**

Sets the serial communications interface (SCI) baud rate.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

← **baudRate** The baud rate

**15.2.5.49 void SCI\_setCharLength (SCI\_Handle sciHandle, const SCI\_CharLength\_e charLength)**

Sets the serial communications interface (SCI) character length.

**Parameters:**

← **sciHandle** The serial communications interface (SCI) object handle

← **charLength** The character length

15.2.5.50 void SCI\_setMode (SCI\_Handle sciHandle, const SCI\_Mode\_e mode)

Sets the serial communications interface (SCI) multiprocessor mode.

**Parameters:**

- ← **sciHandle** The serial communications interface (SCI) object handle
- ← **mode** The multiprocessor mode

15.2.5.51 void SCI\_setNumStopBits (SCI\_Handle sciHandle, const SCI\_NumStopBits\_e numBits)

Sets the serial communications interface (SCI) number of stop bits.

**Parameters:**

- ← **sciHandle** The serial communications interface (SCI) object handle
- ← **numBits** The number of bits

15.2.5.52 void SCI\_setParity (SCI\_Handle sciHandle, const SCI\_Parity\_e parity)

Sets the serial communications interface (SCI) parity.

**Parameters:**

- ← **sciHandle** The serial communications interface (SCI) object handle
- ← **parity** The parity

15.2.5.53 void SCI\_setPriority (SCI\_Handle sciHandle, const SCI\_Priority\_e priority)

Sets the serial communications interface (SCI) priority.

**Parameters:**

- ← **sciHandle** The serial communications interface (SCI) object handle
- ← **priority** The priority

15.2.5.54 void SCI\_setRxFifoIntLevel (SCI\_Handle sciHandle, const SCI\_FifoLevel\_e fifoLevel)

Sets the serial communications interface (SCI) receive FIFO level for generating an interrupt.

**Parameters:**

- ← **sciHandle** The serial communications interface (SCI) object handle
- ← **fifoLevel** The FIFO level

#### 15.2.5.55 void SCI\_setTxDelay ([SCI\\_Handle](#) *sciHandle*, const uint8\_t *delay*)

Sets the serial communications interface (SCI) transmit delay.

**Parameters:**

- ← ***sciHandle*** The serial communications interface (SCI) object handle
- ← ***delay*** The transmit delay

#### 15.2.5.56 void SCI\_setTxFifoIntLevel ([SCI\\_Handle](#) *sciHandle*, const [SCI\\_FifoLevel\\_e](#) *fifoLevel*)

Sets the serial communications interface (SCI) transmit FIFO level for generating an interrupt.

**Parameters:**

- ← ***sciHandle*** The serial communications interface (SCI) object handle
- ← ***fifoLevel*** The FIFO level



# 16 Serial Peripheral Interface (SPI)

Introduction .....	261
API Functions .....	261

## 16.1 Introduction

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.

## 16.2 SPI

### Defines

- `SPI_SPICCR_CHAR_LENGTH_BITS`
- `SPI_SPICCR_CLKPOL_BITS`
- `SPI_SPICCR_RESET_BITS`
- `SPI_SPICCR_SPILBK_BITS`
- `SPI_SPICTL_CLK_PHASE_BITS`
- `SPI_SPICTL_INT_ENA_BITS`
- `SPI_SPICTL_MODE_BITS`
- `SPI_SPICTL_OVRRUN_INT_ENA_BITS`
- `SPI_SPICTL_TALK_BITS`
- `SPI_SPIFFRX_FIFO_OVF_BITS`
- `SPI_SPIFFRX_FIFO_OVFCLR_BITS`
- `SPI_SPIFFRX_FIFO_RESET_BITS`
- `SPI_SPIFFRX_FIFO_ST_BITS`
- `SPI_SPIFFRX_IENA_BITS`
- `SPI_SPIFFRX_IL_BITS`
- `SPI_SPIFFRX_INT_BITS`
- `SPI_SPIFFRX_INTCLR_BITS`
- `SPI_SPIFFTX_CHAN_RESET_BITS`
- `SPI_SPIFFTX_FIFO_ENA_BITS`
- `SPI_SPIFFTX_FIFO_RESET_BITS`
- `SPI_SPIFFTX_FIFO_ST_BITS`
- `SPI_SPIFFTX_IENA_BITS`
- `SPI_SPIFFTX_IL_BITS`
- `SPI_SPIFFTX_INT_BITS`
- `SPI_SPIFFTX_INTCLR_BITS`
- `SPIA_BASE_ADDR`

## Enumerations

- [SPI\\_BaudRate\\_e](#)
- [SPI\\_CharLength\\_e](#)
- [SPI\\_ClkPhase\\_e](#)
- [SPI\\_ClkPolarity\\_e](#)
- [SPI\\_FifoLevel\\_e](#)
- [SPI\\_FifoStatus\\_e](#)
- [SPI\\_Mode\\_e](#)
- [SPI\\_Priority\\_e](#)
- [SPI\\_Stelnv\\_e](#)
- [SPI\\_TriWire\\_e](#)

## 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)

## 16.2.1 Define Documentation

### 16.2.1.1 SPI\_SPICCR\_CHAR\_LENGTH\_BITS

**Definition:**

```
#define SPI_SPICCR_CHAR_LENGTH_BITS
```

**Description:**

Defines the location of the SPICCHAR3-0 bits in the SPICCR register.

### 16.2.1.2 SPI\_SPICCR\_CLKPOL\_BITS

**Definition:**

```
#define SPI_SPICCR_CLKPOL_BITS
```

**Description:**

Defines the location of the CLOCK POLARITY bits in the SPICCR register.

### 16.2.1.3 SPI\_SPICCR\_RESET\_BITS

**Definition:**

```
#define SPI_SPICCR_RESET_BITS
```

**Description:**

Defines the location of the SPI SW Reset bits in the SPICCR register.

#### 16.2.1.4 SPI\_SPICCR\_SPILBK\_BITS

**Definition:**

```
#define SPI_SPICCR_SPILBK_BITS
```

**Description:**

Defines the location of the SPILBK bits in the SPICCR register.

#### 16.2.1.5 SPI\_SPICTL\_CLK\_PHASE\_BITS

**Definition:**

```
#define SPI_SPICTL_CLK_PHASE_BITS
```

**Description:**

Defines the location of the CLOCK PHASE bits in the SPICTL register.

#### 16.2.1.6 SPI\_SPICTL\_INT\_ENA\_BITS

**Definition:**

```
#define SPI_SPICTL_INT_ENA_BITS
```

**Description:**

Defines the location of the SPI INT ENA bits in the SPICTL register.

#### 16.2.1.7 SPI\_SPICTL\_MODE\_BITS

**Definition:**

```
#define SPI_SPICTL_MODE_BITS
```

**Description:**

Defines the location of the MASTER/SLAVE bits in the SPICTL register.

#### 16.2.1.8 SPI\_SPICTL\_OVRRUN\_INT\_ENA\_BITS

**Definition:**

```
#define SPI_SPICTL_OVRRUN_INT_ENA_BITS
```

**Description:**

Defines the location of the OVERRUN INT ENA bits in the SPICTL register.

#### 16.2.1.9 SPI\_SPICTL\_TALK\_BITS

**Definition:**

```
#define SPI_SPICTL_TALK_BITS
```

**Description:**

Defines the location of the TALK bits in the SPICTL register.



#### 16.2.1.10 SPI\_SPIFFRX\_FIFO\_OVF\_BITS

**Definition:**

```
#define SPI_SPIFFRX_FIFO_OVF_BITS
```

**Description:**

Defines the location of the RXFFOVF bits in the SPIFFRX register.

#### 16.2.1.11 SPI\_SPIFFRX\_FIFO\_OVFCLR\_BITS

**Definition:**

```
#define SPI_SPIFFRX_FIFO_OVFCLR_BITS
```

**Description:**

Defines the location of the RXFFOVF CLR bits in the SPIFFRX register.

#### 16.2.1.12 SPI\_SPIFFRX\_FIFO\_RESET\_BITS

**Definition:**

```
#define SPI_SPIFFRX_FIFO_RESET_BITS
```

**Description:**

Defines the location of the RXFIFO Reset bits in the SPIFFRX register.

#### 16.2.1.13 SPI\_SPIFFRX\_FIFO\_ST\_BITS

**Definition:**

```
#define SPI_SPIFFRX_FIFO_ST_BITS
```

**Description:**

Defines the location of the RXFFST4-0 bits in the SPIFFRX register.

#### 16.2.1.14 SPI\_SPIFFRX\_IENA\_BITS

**Definition:**

```
#define SPI_SPIFFRX_IENA_BITS
```

**Description:**

Defines the location of the RXFFIENA bits in the SPIFFRX register.

#### 16.2.1.15 SPI\_SPIFFRX\_IL\_BITS

**Definition:**

```
#define SPI_SPIFFRX_IL_BITS
```

**Description:**

Defines the location of the RXFFIL4-0 bits in the SPIFFRX register.

#### 16.2.1.16 SPI\_SPIFFRX\_INT\_BITS

**Definition:**

```
#define SPI_SPIFFRX_INT_BITS
```

**Description:**

Defines the location of the RXFFINT CLR bits in the SPIFFRX register.

#### 16.2.1.17 SPI\_SPIFFRX\_INTCLR\_BITS

**Definition:**

```
#define SPI_SPIFFRX_INTCLR_BITS
```

**Description:**

Defines the location of the RXFFINT CLR bits in the SPIFFRX register.

#### 16.2.1.18 SPI\_SPIFFTX\_CHAN\_RESET\_BITS

**Definition:**

```
#define SPI_SPIFFTX_CHAN_RESET_BITS
```

**Description:**

Defines the location of the SPIRST bits in the SPIFFTX register.

#### 16.2.1.19 SPI\_SPIFFTX\_FIFO\_ENA\_BITS

**Definition:**

```
#define SPI_SPIFFTX_FIFO_ENA_BITS
```

**Description:**

Defines the location of the SPIFFENA bits in the SPIFFTX register.

#### 16.2.1.20 SPI\_SPIFFTX\_FIFO\_RESET\_BITS

**Definition:**

```
#define SPI_SPIFFTX_FIFO_RESET_BITS
```

**Description:**

Defines the location of the TXFIFO Reset bits in the SPIFFTX register.

#### 16.2.1.21 SPI\_SPIFFTX\_FIFO\_ST\_BITS

**Definition:**

```
#define SPI_SPIFFTX_FIFO_ST_BITS
```

**Description:**

Defines the location of the TXFFST4-0 bits in the SPIFFTX register.

#### 16.2.1.22 SPI\_SPIFFTX\_IENA\_BITS

**Definition:**

```
#define SPI_SPIFFTX_IENA_BITS
```

**Description:**

Defines the location of the TXFFIENA bits in the SPIFFTX register.

#### 16.2.1.23 SPI\_SPIFFTX\_IL\_BITS

**Definition:**

```
#define SPI_SPIFFTX_IL_BITS
```

**Description:**

Defines the location of the TXFFIL4-0 bits in the SPIFFTX register.

#### 16.2.1.24 SPI\_SPIFFTX\_INT\_BITS

**Definition:**

```
#define SPI_SPIFFTX_INT_BITS
```

**Description:**

Defines the location of the TXFFINT bits in the SPIFFTX register.

#### 16.2.1.25 SPI\_SPIFFTX\_INTCLR\_BITS

**Definition:**

```
#define SPI_SPIFFTX_INTCLR_BITS
```

**Description:**

Defines the location of the TXFFINT CLR bits in the SPIFFTX register.

#### 16.2.1.26 SPIA\_BASE\_ADDR

**Definition:**

```
#define SPIA_BASE_ADDR
```

**Description:**

Defines the base address of the serial peripheral interface (SPI) A registers.

### 16.2.2 Typedef Documentation

#### 16.2.2.1 SPI\_Handle

**Definition:**

```
typedef struct SPI_Obj *SPI_Handle
```

**Description:**

Defines the serial peripheral interface (SPI) handle.

## 16.2.3 Enumeration Documentation

### 16.2.3.1 SPI\_BaudRate\_e

**Description:**

Enumeration to define the serial peripheral interface (SPI) baud rates. These assume a LSCLK of 12.5MHz.

**Enumerators:**

***SPI\_BaudRate\_500\_KBaud*** Denotes 500 KBaud.

***SPI\_BaudRate\_1\_MBaud*** Denotes 1 MBaud.

### 16.2.3.2 SPI\_CharLength\_e

**Description:**

Enumeration to define the serial peripheral interface (SPI) character lengths.

**Enumerators:**

***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.

### 16.2.3.3 SPI\_ClkPhase\_e

**Description:**

Enumeration to define the serial peripheral interface (SPI) clock phase.

**Enumerators:**

***SPI\_ClkPhase\_Normal*** Denotes a normal clock scheme.

***SPI\_ClkPhase\_Delayed*** Denotes that the SPICLK signal is delayed by one half-cycle.

#### 16.2.3.4 SPI\_ClkPolarity\_e

**Description:**

Enumeration to define the serial peripheral interface (SPI) clock polarity for the input and output data.

**Enumerators:**

**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.

#### 16.2.3.5 SPI\_FifoLevel\_e

**Description:**

Enumeration to define the serial peripheral interface (SPI) FIFO level.

**Enumerators:**

**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.

#### 16.2.3.6 SPI\_FifoStatus\_e

**Description:**

Enumeration to define the serial peripheral interface (SPI) FIFO status.

**Enumerators:**

**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.

#### 16.2.3.7 SPI\_Mode\_e

**Description:**

Enumeration to define the serial peripheral interface (SPI) network mode control.

**Enumerators:**

**SPI\_Mode\_Slave** Denotes slave mode.

**SPI\_Mode\_Master** Denotes master mode.

#### 16.2.3.8 SPI\_Priority\_e

**Description:**

**Enumerators:**

**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.

#### 16.2.3.9 SPI\_StelInv\_e

**Description:**

**Enumerators:**

**SPI\_StelInv\_ActiveLow** Denotes active low STE pin.

**SPI\_StelInv\_ActiveHigh** Denotes active high STE pin.

#### 16.2.3.10 SPI\_TriWire\_e

**Description:**

**Enumerators:**

**SPI\_TriWire\_NormalFourWire** Denotes 4 wire SPI mode.

**SPI\_TriWire\_ThreeWire** Denotes 3 wire SPI mode.

### 16.2.4 Function Documentation

#### 16.2.4.1 SPI\_clearRxFifoInt

Clears the Rx FIFO interrupt flag.

**Prototype:**

```
void  
SPI_clearRxFifoInt(SPI_Handle spiHandle)
```

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

#### 16.2.4.2 void SPI\_clearRxFifoOvf (SPI\_Handle spiHandle)

Clears the Rx FIFO overflow flag.

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

**16.2.4.3 void SPI\_clearTxFifoInt (SPI\_Handle spiHandle)**

Clears the Tx FIFO interrupt flag.

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

**16.2.4.4 void SPI\_disable (SPI\_Handle spiHandle)**

Disables the serial peripheral interface (SPI).

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

**16.2.4.5 void SPI\_disableChannels (SPI\_Handle spiHandle)**

Disables the serial peripheral interface (SPI) transmit and receive channels.

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

**16.2.4.6 void SPI\_disableInt (SPI\_Handle spiHandle)**

Disables the serial peripheral interface (SPI) interrupt.

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

**16.2.4.7 void SPI\_disableLoopBack (SPI\_Handle spiHandle)**

Disables the serial peripheral interface (SPI) loop back mode.

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

**16.2.4.8 void SPI\_disableOverRunInt (SPI\_Handle spiHandle)**

Disables the serial peripheral interface (SPI) over-run interrupt.

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

#### 16.2.4.9 void SPI\_disableRxFifo ([SPI\\_Handle](#) *spiHandle*)

Disables the serial peripheral interface (SPI) receive FIFO.

**Parameters:**

← ***spiHandle*** The serial peripheral interface (SPI) object handle

#### 16.2.4.10 void SPI\_disableRxFifoInt ([SPI\\_Handle](#) *spiHandle*)

Disables the serial peripheral interface (SPI) receive FIFO interrupt.

**Parameters:**

← ***spiHandle*** The serial peripheral interface (SPI) object handle

#### 16.2.4.11 void SPI\_disableTx ([SPI\\_Handle](#) *spiHandle*)

Disables the serial peripheral interface (SPI) master/slave transmit mode.

**Parameters:**

← ***spiHandle*** The serial peripheral interface (SPI) object handle

#### 16.2.4.12 void SPI\_disableTxFifo ([SPI\\_Handle](#) *spiHandle*)

Disables the serial peripheral interface (SPI) transmit FIFO.

**Parameters:**

← ***spiHandle*** The serial peripheral interface (SPI) object handle

#### 16.2.4.13 void SPI\_disableTxFifoEnh ([SPI\\_Handle](#) *spiHandle*)

Disables the serial peripheral interface (SPI) transmit FIFO enhancements.

**Parameters:**

← ***spiHandle*** The serial peripheral interface (SPI) object handle

#### 16.2.4.14 void SPI\_disableTxFifoInt ([SPI\\_Handle](#) *spiHandle*)

Disables the serial peripheral interface (SPI) transmit FIFO interrupt.

**Parameters:**

← ***spiHandle*** The serial peripheral interface (SPI) object handle



#### 16.2.4.15 void SPI\_enable (SPI\_Handle spiHandle)

Enables the serial peripheral interface (SPI).

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

#### 16.2.4.16 void SPI\_enableChannels (SPI\_Handle spiHandle)

Enables the serial peripheral interface (SPI) transmit and receive channels.

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

#### 16.2.4.17 void SPI\_enableFifoEnh (SPI\_Handle spiHandle)

Enables the serial peripheral interface (SPI) transmit FIFO enhancements.

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

#### 16.2.4.18 void SPI\_enableInt (SPI\_Handle spiHandle)

Enables the serial peripheral interface (SPI) interrupt.

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

#### 16.2.4.19 void SPI\_enableLoopBack (SPI\_Handle spiHandle)

Enables the serial peripheral interface (SPI) loop back mode.

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

#### 16.2.4.20 void SPI\_enableOverRunInt (SPI\_Handle spiHandle)

Enables the serial peripheral interface (SPI) over-run interrupt.

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

#### 16.2.4.21 void SPI\_enableRxFifo ([SPI\\_Handle](#) *spiHandle*)

Enables the serial peripheral interface (SPI) receive FIFO.

**Parameters:**

← ***spiHandle*** The serial peripheral interface (SPI) object handle

#### 16.2.4.22 void SPI\_enableRxFifoInt ([SPI\\_Handle](#) *spiHandle*)

Enables the serial peripheral interface (SPI) receive FIFO interrupt.

**Parameters:**

← ***spiHandle*** The serial peripheral interface (SPI) object handle

#### 16.2.4.23 void SPI\_enableTx ([SPI\\_Handle](#) *spiHandle*)

Enables the serial peripheral interface (SPI) master/slave transmit mode.

**Parameters:**

← ***spiHandle*** The serial peripheral interface (SPI) object handle

#### 16.2.4.24 void SPI\_enableTxFifo ([SPI\\_Handle](#) *spiHandle*)

Enables the serial peripheral interface (SPI) transmit FIFO.

**Parameters:**

← ***spiHandle*** The serial peripheral interface (SPI) object handle

#### 16.2.4.25 void SPI\_enableTxFifoInt ([SPI\\_Handle](#) *spiHandle*)

Enables the serial peripheral interface (SPI) transmit FIFO interrupt.

**Parameters:**

← ***spiHandle*** The serial peripheral interface (SPI) object handle

#### 16.2.4.26 [SPI\\_FifoStatus\\_e](#) SPI\_getRxFifoStatus ([SPI\\_Handle](#) *spiHandle*)

Gets the serial peripheral interface (SPI) receive FIFO status.

**Parameters:**

← ***spiHandle*** The serial peripheral interface (SPI) object handle

**Returns:**

The receive FIFO status

#### 16.2.4.27 `SPI_FifoStatus_e` `SPI_getTxFifoStatus (SPI_Handle spiHandle)`

Gets the serial peripheral interface (SPI) transmit FIFO status.

**Parameters:**

← ***spiHandle*** The serial peripheral interface (SPI) object handle

**Returns:**

The transmit FIFO status

#### 16.2.4.28 `SPI_Handle` `SPI_init (void * pMemory, const size_t numBytes)`

Initializes the serial peripheral interface (SPI) object handle.

**Parameters:**

← ***pMemory*** A pointer to the base address of the SPI registers

← ***numBytes*** The number of bytes allocated for the SPI object, bytes

**Returns:**

The serial peripheral interface (SPI) object handle

#### 16.2.4.29 `uint16_t` `SPI_read (SPI_Handle spiHandle) [inline]`

Reads data from the serial peripheral interface (SPI).

**Parameters:**

← ***spiHandle*** The serial peripheral interface (SPI) object handle

**Returns:**

The received data value

#### 16.2.4.30 `void` `SPI_reset (SPI_Handle spiHandle)`

Resets the serial peripheral interface (SPI).

**Parameters:**

← ***spiHandle*** The serial peripheral interface (SPI) object handle

#### 16.2.4.31 `void` `SPI_resetChannels (SPI_Handle spiHandle)`

Resets the serial peripheral interface (SPI) transmit and receive channels.

**Parameters:**

← ***spiHandle*** The serial peripheral interface (SPI) object handle

#### 16.2.4.32 void SPI\_resetRxFifo ([SPI\\_Handle spiHandle](#))

Resets the serial peripheral interface (SPI) receive FIFO.

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

#### 16.2.4.33 void SPI\_resetTxFifo ([SPI\\_Handle spiHandle](#))

Resets the serial peripheral interface (SPI) transmit FIFO.

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

#### 16.2.4.34 void SPI\_setBaudRate ([SPI\\_Handle spiHandle](#), const [SPI\\_BaudRate\\_e baudRate](#))

Sets the serial peripheral interface (SPI) baud rate.

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

← **baudRate** The baud rate

#### 16.2.4.35 void SPI\_setCharLength ([SPI\\_Handle spiHandle](#), const [SPI\\_CharLength\\_e length](#))

Sets the serial peripheral interface (SPI) character length.

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

← **length** The character length

#### 16.2.4.36 void SPI\_setClkPhase ([SPI\\_Handle spiHandle](#), const [SPI\\_ClkPhase\\_e clkPhase](#))

Sets the serial peripheral interface (SPI) clock phase.

**Parameters:**

← **spiHandle** The serial peripheral interface (SPI) object handle

← **clkPhase** The clock phase

**16.2.4.37 void SPI\_setClkPolarity (SPI\_Handle spiHandle, const SPI\_ClkPolarity\_e polarity)**

Sets the serial peripheral interface (SPI) clock polarity.

**Parameters:**

- ← **spiHandle** The serial peripheral interface (SPI) object handle
- ← **polarity** The clock polarity

**16.2.4.38 void SPI\_setMode (SPI\_Handle spiHandle, const SPI\_Mode\_e mode)**

Sets the serial peripheral interface (SPI) network mode.

**Parameters:**

- ← **spiHandle** The serial peripheral interface (SPI) object handle
- ← **mode** The network mode

**16.2.4.39 void SPI\_setPriority (SPI\_Handle spiHandle, const SPI\_Priority\_e priority)**

Sets the priority of the SPI port vis-a-vis the EMU.

**Parameters:**

- ← **spiHandle** The serial peripheral interface (SPI) object handle
- ← **priority** The priority of the SPI port vis-a-vis the EMU

**16.2.4.40 void SPI\_setRxFifoIntLevel (SPI\_Handle spiHandle, const SPI\_FifoLevel\_e fifoLevel)**

Sets the serial peripheral interface (SPI) receive FIFO level for generating an interrupt.

**Parameters:**

- ← **spiHandle** The serial peripheral interface (SPI) object handle
- ← **fifoLevel** The FIFO level

**16.2.4.41 void SPI\_setStelInv (SPI\_Handle spiHandle, const SPI\_StelInv\_e steinv)**

Controls pin inversion of STE pin.

**Parameters:**

- ← **spiHandle** The serial peripheral interface (SPI) object handle
- ← **steinv** Polarity of STE pin

16.2.4.42 void SPI\_setTriWire ([SPI\\_Handle](#) *spiHandle*, const [SPI\\_TriWire\\_e](#) *triwire*)

Sets SPI port operating mode.

**Parameters:**

- ← ***spiHandle*** The serial peripheral interface (SPI) object handle
- ← ***triwire*** 3 or 4 wire mode

16.2.4.43 void SPI\_setTxDelay ([SPI\\_Handle](#) *spiHandle*, const uint8\_t *delay*)

Sets the serial peripheral interface (SPI) transmit delay.

**Parameters:**

- ← ***spiHandle*** The serial peripheral interface (SPI) object handle
- ← ***delay*** The delay value

16.2.4.44 void SPI\_setTxFifoIntLevel ([SPI\\_Handle](#) *spiHandle*, const [SPI\\_FifoLevel\\_e](#) *fifoLevel*)

Sets the serial peripheral interface (SPI) transmit FIFO level for generating an interrupt.

**Parameters:**

- ← ***spiHandle*** The serial peripheral interface (SPI) object handle
- ← ***fifoLevel*** The FIFO level

16.2.4.45 void SPI\_write ([SPI\\_Handle](#) *spiHandle*, const uint16\_t *data*) [inline]

Writes data to the serial peripheral interface (SPI).

**Parameters:**

- ← ***spiHandle*** The serial peripheral interface (SPI) object handle
- ← ***data*** The data value

# 17 Timer

Introduction .....	279
API Functions .....	279

## 17.1 Introduction

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.

## 17.2 TIMER

### Data Structures

- [\\_TIMER\\_Obj](#)

### Defines

- [TIMER0\\_BASE\\_ADDR](#)
- [TIMER1\\_BASE\\_ADDR](#)
- [TIMER2\\_BASE\\_ADDR](#)
- [TIMER\\_TCR\\_FREESOFT\\_BITS](#)
- [TIMER\\_TCR\\_TIE\\_BITS](#)
- [TIMER\\_TCR\\_TIF\\_BITS](#)
- [TIMER\\_TCR\\_TRB\\_BITS](#)
- [TIMER\\_TCR\\_TSS\\_BITS](#)

### Enumerations

- [TIMER\\_EmulationMode\\_e](#)
- [TIMER\\_Status\\_e](#)

### 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)

## 17.2.1 Data Structure Documentation

### 17.2.1.1 `_TIMER_Obj_`

**Definition:**

```
typedef struct
{
    uint32_t TIM;
    uint32_t PRD;
    uint32_t TCR;
    uint32_t TPR;
}
_TIMER_Obj_
```

**Members:**

***TIM*** Timer Counter Register.  
***PRD*** Period Register.  
***TCR*** Timer Control Register.  
***TPR*** Timer Prescaler Register.

**Description:**

Defines the timer (TIMER) object.

## 17.2.2 Define Documentation

### 17.2.2.1 `TIMER0_BASE_ADDR`

**Definition:**

```
#define TIMER0_BASE_ADDR
```

**Description:**

Defines the base address of the timer (TIMER) 0 registers.

### 17.2.2.2 `TIMER1_BASE_ADDR`

**Definition:**

```
#define TIMER1_BASE_ADDR
```



**Description:**

Defines the base address of the timer (TIMER) 1 registers.

### 17.2.2.3 TIMER2\_BASE\_ADDR

**Definition:**

```
#define TIMER2_BASE_ADDR
```

**Description:**

Defines the base address of the timer (TIMER) 2 registers.

### 17.2.2.4 TIMER\_TCR\_FREESOFT\_BITS

**Definition:**

```
#define TIMER_TCR_FREESOFT_BITS
```

**Description:**

Defines the location of the FREESOFT bits in the TCR register.

### 17.2.2.5 TIMER\_TCR\_TIE\_BITS

**Definition:**

```
#define TIMER_TCR_TIE_BITS
```

**Description:**

Defines the location of the TIE bits in the TCR register.

### 17.2.2.6 TIMER\_TCR\_TIF\_BITS

**Definition:**

```
#define TIMER_TCR_TIF_BITS
```

**Description:**

Defines the location of the TIF bits in the TCR register.

### 17.2.2.7 TIMER\_TCR\_TRB\_BITS

**Definition:**

```
#define TIMER_TCR_TRB_BITS
```

**Description:**

Defines the location of the TRB bits in the TCR register.

### 17.2.2.8 TIMER\_TCR\_TSS\_BITS

**Definition:**

```
#define TIMER_TCR_TSS_BITS
```

**Description:**

Defines the location of the TSS bits in the TCR register.

## 17.2.3 Typedef Documentation

### 17.2.3.1 TIMER\_Handle

**Definition:**

```
typedef struct TIMER\_Obj *TIMER\_Handle
```

**Description:**

Defines the timer (TIMER) handle.

### 17.2.3.2 TIMER\_Obj

**Definition:**

```
typedef struct \_TIMER\_Obj\_ TIMER\_Obj
```

**Description:**

Defines the timer (TIMER) object.

## 17.2.4 Enumeration Documentation

### 17.2.4.1 TIMER\_EmulationMode\_e

**Description:**

Enumeration to define the timer (TIMER) emulation mode.

**Enumerators:**

***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.

### 17.2.4.2 TIMER\_Status\_e

**Description:**

Enumeration to define the timer (TIMER) status.

**Enumerators:**

***TIMER\_Status\_CntIsNotZero*** Denotes that the counter is non-zero.

***TIMER\_Status\_CntIsZero*** Denotes that the counter is zero.

---

## 17.2.5 Function Documentation

### 17.2.5.1 `TIMER_clearFlag`

Clears the timer (TIMER) flag.

**Prototype:**

```
void  
TIMER_clearFlag(TIMER_Handle timerHandle)
```

**Parameters:**

← ***timerHandle*** The timer (TIMER) object handle

### 17.2.5.2 `void TIMER_disableInt (TIMER_Handle timerHandle)`

Disables the timer (TIMER) interrupt.

**Parameters:**

← ***timerHandle*** The timer (TIMER) object handle

### 17.2.5.3 `void TIMER_enableInt (TIMER_Handle timerHandle)`

Enables the timer (TIMER) interrupt.

**Parameters:**

← ***timerHandle*** The timer (TIMER) object handle

### 17.2.5.4 `uint32_t TIMER_getCount (TIMER_Handle timerHandle) [inline]`

Gets the timer (TIMER) count.

**Parameters:**

← ***timerHandle*** The timer (TIMER) object handle

**Returns:**

The timer (TIMER) count

### 17.2.5.5 `TIMER_Status_e TIMER_getStatus (TIMER_Handle timerHandle)`

Gets the timer (TIMER) status.

**Parameters:**

← ***timerHandle*** The timer (TIMER) object handle

**Returns:**

The timer (TIMER) status

**17.2.5.6** `TIMER_Handle` `TIMER_init` (`void * pMemory`, `const size_t numBytes`)

Initializes the timer (TIMER) object handle.

**Parameters:**

- ← ***pMemory*** A pointer to the base address of the TIMER registers
- ← ***numBytes*** The number of bytes allocated for the TIMER object, bytes

**Returns:**

The timer (CLK) object handle

**17.2.5.7** `void` `TIMER_reload` (`TIMER_Handle timerHandle`)

Reloads the timer (TIMER) value.

**Parameters:**

- ← ***timerHandle*** The timer (TIMER) object handle

**17.2.5.8** `void` `TIMER_setDecimationFactor` (`TIMER_Handle timerHandle`, `const uint16_t decFactor`)

Sets the timer (TIMER) decimation factor.

**Parameters:**

- ← ***timerHandle*** The timer (TIMER) object handle
- ← ***decFactor*** The timer decimation factor

**17.2.5.9** `void` `TIMER_setEmulationMode` (`TIMER_Handle timerHandle`, `const TIMER_EmulationMode_e mode`)

Sets the timer (TIMER) emulation mode.

**Parameters:**

- ← ***timerHandle*** The timer (TIMER) object handle
- ← ***mode*** The emulation mode

**17.2.5.10** `void` `TIMER_setPeriod` (`TIMER_Handle timerHandle`, `const uint32_t period`)

Sets the timer (TIMER) period.

**Parameters:**

- ← ***timerHandle*** The timer (TIMER) object handle
- ← ***period*** The period

#### 17.2.5.11 void TIMER\_setPreScaler ([TIMER\\_Handle](#) *timerHandle*, const uint16\_t *preScaler*)

Sets the timer (TIMER) prescaler.

**Parameters:**

- ← ***timerHandle*** The timer (TIMER) object handle
- ← ***preScaler*** The preScaler value

#### 17.2.5.12 void TIMER\_start ([TIMER\\_Handle](#) *timerHandle*)

Starts the timer (TIMER).

**Parameters:**

- ← ***timerHandle*** The timer (TIMER) object handle

#### 17.2.5.13 void TIMER\_stop ([TIMER\\_Handle](#) *timerHandle*)

Stops the timer (TIMER).

**Parameters:**

- ← ***timerHandle*** The timer (TIMER) object handle



# 18 Watchdog Timer

Introduction .....	287
API Functions .....	287

## 18.1 Introduction

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.

## 18.2 WDOG

### Data Structures

- [\\_WDOG\\_Obj\\_](#)

### Defines

- [WDOG\\_BASE\\_ADDR](#)
- [WDOG\\_SCSR\\_WDENINT\\_BITS](#)
- [WDOG\\_SCSR\\_WDINTS\\_BITS](#)
- [WDOG\\_SCSR\\_WDOVERRIDE\\_BITS](#)
- [WDOG\\_WDCNTR\\_BITS](#)
- [WDOG\\_WDCR\\_WDCHK\\_BITS](#)
- [WDOG\\_WDCR\\_WDDIS\\_BITS](#)
- [WDOG\\_WDCR\\_WDFLAG\\_BITS](#)
- [WDOG\\_WDCR\\_WDPS\\_BITS](#)
- [WDOG\\_WDCR\\_WRITE\\_ENABLE](#)
- [WDOG\\_WDKEY\\_BITS](#)

### Enumerations

- [WDOG\\_IntStatus\\_e](#)

### Functions

- void [WDOG\\_clearCounter](#) ([WDOG\\_Handle](#) wdogHandle)
- void [WDOG\\_disable](#) ([WDOG\\_Handle](#) wdogHandle)
- void [WDOG\\_disableInt](#) ([WDOG\\_Handle](#) wdogHandle)

- void [WDOG\\_disableOverRide](#) ([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)

## 18.2.1 Data Structure Documentation

### 18.2.1.1 [\\_WDOG\\_Obj\\_](#)

**Definition:**

```
typedef struct
{
    uint16_t  SCSR;
    uint16_t  WDCNTR;
    uint16_t  rsvd_1;
    uint16_t  WDKEY;
    uint16_t  rsvd_2[3];
    uint16_t  WDCR;
}
_WDOG_Obj_
```

**Members:**

**SCSR** System Control Status Register.

**WDCNTR** Watchdog Counter Register.

**rsvd\_1** Reserved.

**WDKEY** Watchdog Reset Key Register.

**rsvd\_2** Reserved.

**WDCR** Watchdog Control Register.

**Description:**

Defines the watchdog (WDOG) object.

## 18.2.2 Define Documentation

### 18.2.2.1 [WDOG\\_BASE\\_ADDR](#)

**Definition:**

```
#define WDOG_BASE_ADDR
```

**Description:**

Defines the base address of the watchdog (WDOG) registers.



### 18.2.2.2 WDOG\_SCSR\_WDENINT\_BITS

**Definition:**

```
#define WDOG_SCSR_WDENINT_BITS
```

**Description:**

Defines the location of the WDENINT bits in the SCSR register.

### 18.2.2.3 WDOG\_SCSR\_WDINTS\_BITS

**Definition:**

```
#define WDOG_SCSR_WDINTS_BITS
```

**Description:**

Defines the location of the WDINTS bits in the SCSR register.

### 18.2.2.4 WDOG\_SCSR\_WDOVERRIDE\_BITS

**Definition:**

```
#define WDOG_SCSR_WDOVERRIDE_BITS
```

**Description:**

Defines the location of the WDOVERRIDE bits in the SCSR register.

### 18.2.2.5 WDOG\_WDCNTR\_BITS

**Definition:**

```
#define WDOG_WDCNTR_BITS
```

**Description:**

Defines the location of the WDCNTR bits in the WDCNTR register.

### 18.2.2.6 WDOG\_WDCR\_WDCHK\_BITS

**Definition:**

```
#define WDOG_WDCR_WDCHK_BITS
```

**Description:**

Defines the location of the WDCHK bits in the WDCR register.

### 18.2.2.7 WDOG\_WDCR\_WDDIS\_BITS

**Definition:**

```
#define WDOG_WDCR_WDDIS_BITS
```

**Description:**

Defines the location of the WDDIS bits in the WDCR register.

#### 18.2.2.8 WDOG\_WDCR\_WDFLAG\_BITS

**Definition:**

```
#define WDOG_WDCR_WDFLAG_BITS
```

**Description:**

Defines the location of the WDFLAG bits in the WDCR register.

#### 18.2.2.9 WDOG\_WDCR\_WDPS\_BITS

**Definition:**

```
#define WDOG_WDCR_WDPS_BITS
```

**Description:**

Defines the location of the WDPS bits in the WDCR register.

#### 18.2.2.10 WDOG\_WDCR\_WRITE\_ENABLE

**Definition:**

```
#define WDOG_WDCR_WRITE_ENABLE
```

**Description:**

Defines the watchdog write enable mode.

#### 18.2.2.11 WDOG\_WDKEY\_BITS

**Definition:**

```
#define WDOG_WDKEY_BITS
```

**Description:**

Defines the location of the WDKEY bits in the WDKEY register.

### 18.2.3 Typedef Documentation

#### 18.2.3.1 WDOG\_Handle

**Definition:**

```
typedef struct WDOG_Obj *WDOG_Handle
```

**Description:**

Defines the watchdog (WDOG) handle.

### 18.2.3.2 WDOG\_Obj

**Definition:**

```
typedef struct _WDOG_Obj_ WDOG_Obj
```

**Description:**

Defines the watchdog (WDOG) object.

## 18.2.4 Enumeration Documentation

### 18.2.4.1 WDOG\_IntStatus\_e

**Description:**

Enumeration to define the watchdog (WDOG) interrupt status.

### 18.2.4.2 enum WDOG\_Prescaler\_e

Enumeration to define the watchdog (WDOG) timer clock prescaler, which sets the clock frequency.

**Enumerators:**

**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  $WDCLK = OSCCLK/512/64$ .

## 18.2.5 Function Documentation

### 18.2.5.1 WDOG\_clearCounter

Clears the watchdog (WDOG) counter.

**Prototype:**

```
void  
WDOG_clearCounter(WDOG_Handle wdogHandle)
```

**Parameters:**

← **wdogHandle** The watchdog (WDOG) timer object handle

### 18.2.5.2 void WDOG\_disable (WDOG\_Handle wdogHandle)

Disables the watchdog (WDOG) timer.

**Parameters:**

← ***wdogHandle*** The watchdog (WDOG) timer object handle

18.2.5.3 void WDOG\_disableInt ([WDOG\\_Handle](#) *wdogHandle*)

Disables the watchdog (WDOG) timer interrupt.

**Parameters:**

← ***wdogHandle*** The watchdog (WDOG) timer object handle

18.2.5.4 void WDOG\_disableOverRide ([WDOG\\_Handle](#) *wdogHandle*)

Disables the timer override.

**Parameters:**

← ***wdogHandle*** The watchdog (WDOG) timer object handle

18.2.5.5 void WDOG\_enable ([WDOG\\_Handle](#) *wdogHandle*)

Enables the watchdog (WDOG) timer.

**Parameters:**

← ***wdogHandle*** The watchdog (WDOG) timer object handle

18.2.5.6 void WDOG\_enableInt ([WDOG\\_Handle](#) *wdogHandle*)

Enables the watchdog (WDOG) timer interrupt.

**Parameters:**

← ***wdogHandle*** The watchdog (WDOG) timer object handle

18.2.5.7 void WDOG\_enableOverRide ([WDOG\\_Handle](#) *wdogHandle*)

Enables the watchdog (WDOG) timer override.

**Parameters:**

← ***wdogHandle*** The watchdog (WDOG) timer object handle

18.2.5.8 [WDOG\\_IntStatus\\_e](#) WDOG\_getIntStatus ([WDOG\\_Handle](#) *wdogHandle*)

Gets the watchdog (WDOG) interrupt status.

**Parameters:**

← ***wdogHandle*** The watchdog (WDOG) timer object handle

**Returns:**

The interrupt status

**18.2.5.9** `WDOG_Handle` `WDOG_init` (`void * pMemory`, `const size_t numBytes`)

Initializes the watchdog (WDOG) object handle.

**Parameters:**

- ← ***pMemory*** A pointer to the base address of the WDOG registers
- ← ***numBytes*** The number of bytes allocated for the WDOG object, bytes

**Returns:**

The watchdog (WDOG) object handle

**18.2.5.10** `void` `WDOG_setCount` (`WDOG_Handle wdogHandle`, `const uint8_t count`)

Sets the watchdog (WDOG) counter.

**Parameters:**

- ← ***wdogHandle*** The watchdog (WDOG) timer object handle
- ← ***count*** The count

**18.2.5.11** `void` `WDOG_setPreScaler` (`WDOG_Handle wdogHandle`, `const WDOG_PreScaler_e preScaler`)

Sets the watchdog (WDOG) timer clock prescaler.

**Parameters:**

- ← ***wdogHandle*** The watchdog (WDOG) timer object handle
- ← ***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 © 2012, Texas Instruments Incorporated