F2802x Peripheral Driver Library

USER'S GUIDE

F2802x-DRL-UG-230

Copyright

Texas Instruments
12203 Southwest Freeway
Houston, TX 77477
http://www.ti.com/c2000

Revision Information

This is version 230 of this document, last updated on Fri May 8 07:43:11 CDT 2015.

Table of Contents

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.

`f2802x_common/src/`	This directory contains the source code for the drivers.
`f2802x_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.

f2802x_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

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 `f2802x_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 `f2802x_headers/source/F2802x_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 `F2802x_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 `f2802x_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 f2802x 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 `f2802x_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 `F2802x_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/f2802x/version/` to your projects include path (Right click on project, Build Properties, Include Path)
- Include driver header files from `f2802x_common/include` in any source file that makes calls to that driver.

3 Analog to Digital Converter (ADC)

3.1 Introduction

The Analog to Digital Converter (ADC) APIs provide a set of functions for accessing the Piccolo F2802x ADC modules.

This driver is contained in `f2802x_common/source/adc.c`, with `f2802x_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 rsvd_1[26096];
    uint16_t ADCCTL1;
    uint16_t rsvd_2[3];
    uint16_t ADCINTFLG;
    uint16_t ADCINTFLGCLR;
    uint16_t ADCINTOVF;
    uint16_t ADCINTOVFCLR;
    uint16_t INTSELxNy[5];
    uint16_t rsvd_3[3];
    uint16_t SOCPRICTRL;
    uint16_t rsvd_4;
    uint16_t ADCSAMPLEMODE;
    uint16_t rsvd_5;
    uint16_t ADCINTSOCSEL1;
    uint16_t ADCINTSOCSEL2;
    uint16_t rsvd_6[2];
    uint16_t ADCSOCFLG1;
    uint16_t rsvd_7;
    uint16_t ADCSOCFRC1;
    uint16_t rsvd_8;
```

```
    uint16_t ADCSOCOVF1;
    uint16_t rsvd_9;
    uint16_t ADCSOCOVFCLR1;
    uint16_t rsvd_10;
    uint16_t ADCSOCxCTL[16];
    uint16_t rsvd_11[16];
    uint16_t ADCREFTRIM;
    uint16_t ADCOFFTRIM;
    uint16_t rsvd_12[13];
    uint16_t ADCREV;
}
_ADC_Obj_
```

Members:

ADCRESULT ADC result registers.
rsvd_1 Reserved.
ADCCTL1 ADC Control Register 1.
rsvd_2 Reserved.
ADCINTFLG ADC Interrupt Flag Register.
ADCINTFLGCLR ADC Interrupt Flag Clear Register.
ADCINTOVF ADC Interrupt Overflow Register.
ADCINTOVFCLR ADC Interrupt Overflow Clear Register.
INTSELxNy ADC Interrupt Select x and y Register.
rsvd_3 Reserved.
SOCPRICTRL ADC Start Of Conversion Priority Control Register.
rsvd_4 Reserved.
ADCSAMPLEMODE ADC Sample Mode Register.
rsvd_5 Reserved.
ADCINTSOCSEL1 ADC Interrupt Trigger SOC Select 1 Register.
ADCINTSOCSEL2 ADC Interrupt Trigger SOC Select 2 Register.
rsvd_6 Reserved.
ADCSOCFLG1 ADC SOC Flag 1 Register.
rsvd_7 Reserved.
ADCSOCFRC1 ADC SOC Force 1 Register.
rsvd_8 Reserved.
ADCSOCOVF1 ADC SOC Overflow 1 Register.
rsvd_9 Reserved.
ADCSOCOVFCLR1 ADC SOC Overflow Clear 1 Register.
rsvd_10 Reserved.
ADCSOCxCTL ADC SOCx Control Registers.
rsvd_11 Reserved.
ADCREFTRIM ADC Reference/Gain Trim Register.
ADCOFFTRIM ADC Offset Trim Register.
rsvd_12 Reserved.
ADCREV ADC Revision Register.

Description:

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

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 with not be generated until the interrupt flag is cleared.

ADC_IntMode_EOC Denotes that a new interrupt with 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.

ADC_SocSampleWindow_13_cycles Denotes an SOC sample window of 13 cycles.
ADC_SocSampleWindow_14_cycles Denotes an SOC sample window of 14 cycles.
ADC_SocSampleWindow_15_cycles Denotes an SOC sample window of 15 cycles.
ADC_SocSampleWindow_16_cycles Denotes an SOC sample window of 16 cycles.
ADC_SocSampleWindow_23_cycles Denotes an SOC sample window of 23 cycles.
ADC_SocSampleWindow_24_cycles Denotes an SOC sample window of 24 cycles.
ADC_SocSampleWindow_25_cycles Denotes an SOC sample window of 25 cycles.
ADC_SocSampleWindow_26_cycles Denotes an SOC sample window of 26 cycles.
ADC_SocSampleWindow_27_cycles Denotes an SOC sample window of 27 cycles.
ADC_SocSampleWindow_28_cycles Denotes an SOC sample window of 28 cycles.
ADC_SocSampleWindow_29_cycles Denotes an SOC sample window of 29 cycles.
ADC_SocSampleWindow_35_cycles Denotes an SOC sample window of 35 cycles.
ADC_SocSampleWindow_36_cycles Denotes an SOC sample window of 36 cycles.
ADC_SocSampleWindow_37_cycles Denotes an SOC sample window of 37 cycles.
ADC_SocSampleWindow_38_cycles Denotes an SOC sample window of 38 cycles.
ADC_SocSampleWindow_39_cycles Denotes an SOC sample window of 39 cycles.
ADC_SocSampleWindow_40_cycles Denotes an SOC sample window of 40 cycles.
ADC_SocSampleWindow_41_cycles Denotes an SOC sample window of 41 cycles.
ADC_SocSampleWindow_42_cycles Denotes an SOC sample window of 42 cycles.
ADC_SocSampleWindow_48_cycles Denotes an SOC sample window of 48 cycles.
ADC_SocSampleWindow_49_cycles Denotes an SOC sample window of 49 cycles.
ADC_SocSampleWindow_50_cycles Denotes an SOC sample window of 50 cycles.
ADC_SocSampleWindow_51_cycles Denotes an SOC sample window of 51 cycles.
ADC_SocSampleWindow_52_cycles Denotes an SOC sample window of 52 cycles.
ADC_SocSampleWindow_53_cycles Denotes an SOC sample window of 53 cycles.
ADC_SocSampleWindow_54_cycles Denotes an SOC sample window of 54 cycles.
ADC_SocSampleWindow_55_cycles Denotes an SOC sample window of 55 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 internal 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)

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 `f2802x_common/source/cap.c`, with `f2802x_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_CEVT1_BITS

- CAP_ECCxxx_CEVT2_BITS
- CAP_ECCxxx_CEVT3_BITS
- CAP_ECCxxx_CEVT4_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_setApwmShadowPeriod (CAP_Handle capHandle, const uint32_t shadwoPeriod)
- 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_CEVT1_BITS

Definition:

```
#define CAP_ECCxxx_CEVT1_BITS
```

Description:

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

4.2.2.22 CAP_ECCxxx_CEVT2_BITS

Definition:

```
#define CAP_ECCxxx_CEVT2_BITS
```

Description:

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

4.2.2.23 CAP_ECCxxx_CEVT3_BITS

Definition:

```
#define CAP_ECCxxx_CEVT3_BITS
```

Description:

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

4.2.2.24 CAP_ECCxxx_CEVT4_BITS

Definition:

```
#define CAP_ECCxxx_CEVT4_BITS
```

Description:

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

4.2.2.25 CAP_ECCxxx_CTRCOMP_BITS

Definition:

```
#define CAP_ECCxxx_CTRCOMP_BITS
```

Description:

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

4.2.2.26 CAP_ECCxxx_CTROVF_BITS

Definition:

```
#define CAP_ECCxxx_CTROVF_BITS
```

Description:

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

4.2.2.27 CAP_ECCxxx_CTRPRD_BITS

Definition:

```
#define CAP_ECCxxx_CTRPRD_BITS
```

Description:

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

4.2.2.28 CAP_ECCxxx_INT_BITS

Definition:

```
#define CAP_ECCxxx_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_CEVT4 Denotes CEVT4 interrupt.
CAP_Int_Type_CEVT3 Denotes CEVT3 interrupt.
CAP_Int_Type_CEVT2 Denotes CEVT2 interrupt.
CAP_Int_Type_CEVT1 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_CEVT1 Stop/Wrap after Capture Event 1.
CAP_Stop_Wrap_CEVT2 Stop/Wrap after Capture Event 2.
CAP_Stop_Wrap_CEVT3 Stop/Wrap after Capture Event 3.
CAP_Stop_Wrap_CEVT4 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_disableSyncIn (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_setApwmShadowPeriod (CAP_Handle *capHandle*, const uint32_t *shadwoPeriod*) [inline]

Sets the APWM shadow period.

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

4.2.5.19 void CAP_setCapContinuous (CAP_Handle *capHandle*)

Sets up for continuous Capture.

Parameters:

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

4.2.5.20 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

4.2.5.21 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

4.2.5.22 void CAP_setCapOneShot (CAP_Handle *capHandle*)

Sets up for one-shot Capture.

Parameters:

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

4.2.5.23 void CAP_setModeApwm (CAP_Handle *capHandle*)

Sets capture peripheral up for APWM mode.

Parameters:

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

4.2.5.24 void CAP_setModeCap (CAP_Handle *capHandle*)

Sets capture peripheral up for capture mode.

Parameters:

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

4.2.5.25 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

4.2.5.26 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)

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 <ttf280x0_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 LSPNCLK 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 SYSCLKOUT.
CLK_Timer2Src_ExtOsc Denotes the CPU timer 2 clock source is external oscillator.
CLK_Timer2Src_IntOsc1 Denotes the CPU timer 2 clock source is internal oscillator 1.
CLK_Timer2Src_IntOsc2 Denotes the CPU timer 2 clock source is internal oscillator 2.

5.2.4.10 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)

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 Maxmimum 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 Syncronize 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)

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 `f2802x_common/source/cpu.c`, with `f2802x_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
- cregister volatile unsigned int IER
- cregister volatile unsigned int IFR

7.2.1 Data Structure Documentation

7.2.1.1 _CPU_Obj_

Definition:

```
typedef struct
{
    HASH(0x1120ea8) 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

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 `f2802x_common/source/flash.c`, with `f2802x_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_ACTIVEWAIT_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_ACTIVEWAITS_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 randowm read wait states is 1.

FLASH_NumRandomWaitStates_2 Denotes the number of randowm read wait states is 2.
FLASH_NumRandomWaitStates_3 Denotes the number of randowm read wait states is 3.
FLASH_NumRandomWaitStates_4 Denotes the number of randowm read wait states is 4.
FLASH_NumRandomWaitStates_5 Denotes the number of randowm read wait states is 5.
FLASH_NumRandomWaitStates_6 Denotes the number of randowm read wait states is 6.
FLASH_NumRandomWaitStates_7 Denotes the number of randowm read wait states is 7.
FLASH_NumRandomWaitStates_8 Denotes the number of randowm read wait states is 8.
FLASH_NumRandomWaitStates_9 Denotes the number of randowm read wait states is 9.
FLASH_NumRandomWaitStates_10 Denotes the number of randowm read wait states is 10.
FLASH_NumRandomWaitStates_11 Denotes the number of randowm read wait states is 11.
FLASH_NumRandomWaitStates_12 Denotes the number of randowm read wait states is 12.
FLASH_NumRandomWaitStates_13 Denotes the number of randowm read wait states is 13.
FLASH_NumRandomWaitStates_14 Denotes the number of randowm read wait states is 14.
FLASH_NumRandomWaitStates_15 Denotes the number of randowm 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)

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 `f2802x_common/source/gpio.c`, with `f2802x_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.

GPIOLPMSEL 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_EPWMSYNCI Denotes a EPWMSYNCI 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_SDAA Denotes a SDDA function.
GPIO_32_Mode_EPWMSYNCI Denotes a EPWMSYNCI function.
GPIO_32_Mode_ADCSOCAO_NOT Denotes a ADCSOCAO_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 boolen 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*)

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)

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 `f2802x_common/source/osc.c`, with `f2802x_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)

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 `f2802x_common/source/pie.c`, with `f2802x_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_illegalIsr (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 PIECTRL;
    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)

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 `f2802x_common/source/pll.c`, with `f2802x_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)

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 `f2802x_common/source/pwm.c`, with `f2802x_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_DCTRIPSEL_DCALCOMPSEL_BITS
- PWM_DCTRIPSEL_DCBHCOMPSEL_BITS
- PWM_DCTRIPSEL_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_PHSEN_BITS
- PWM_TBCTL_PRDLD_BITS
- PWM_TBCTL_SWFSYNC_BITS
- PWM_TBCTL_SYNCOSEL_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 tripZone-Flag)
- 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_SYNCOSEL_BITS

Definition:

```
#define PWM_TBCTL_SYNCOSEL_BITS
```

Description:

Defines the location of the SYNCOSEL 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 (DCAEVT1).

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 (DCBEVT1).

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)

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 `f2802x_common/source/pwr.c`, with `f2802x_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.

PWR_NumStandByClocks_38 Denotes 38 standby clock cycles.
PWR_NumStandByClocks_39 Denotes 39 standby clock cycles.
PWR_NumStandByClocks_40 Denotes 40 standby clock cycles.
PWR_NumStandByClocks_41 Denotes 41 standby clock cycles.
PWR_NumStandByClocks_42 Denotes 42 standby clock cycles.
PWR_NumStandByClocks_43 Denotes 43 standby clock cycles.
PWR_NumStandByClocks_44 Denotes 44 standby clock cycles.
PWR_NumStandByClocks_45 Denotes 45 standby clock cycles.
PWR_NumStandByClocks_46 Denotes 46 standby clock cycles.
PWR_NumStandByClocks_47 Denotes 47 standby clock cycles.
PWR_NumStandByClocks_48 Denotes 48 standby clock cycles.
PWR_NumStandByClocks_49 Denotes 49 standby clock cycles.
PWR_NumStandByClocks_50 Denotes 50 standby clock cycles.
PWR_NumStandByClocks_51 Denotes 51 standby clock cycles.
PWR_NumStandByClocks_52 Denotes 52 standby clock cycles.
PWR_NumStandByClocks_53 Denotes 53 standby clock cycles.
PWR_NumStandByClocks_54 Denotes 54 standby clock cycles.
PWR_NumStandByClocks_55 Denotes 55 standby clock cycles.
PWR_NumStandByClocks_56 Denotes 56 standby clock cycles.
PWR_NumStandByClocks_57 Denotes 57 standby clock cycles.
PWR_NumStandByClocks_58 Denotes 58 standby clock cycles.
PWR_NumStandByClocks_59 Denotes 59 standby clock cycles.
PWR_NumStandByClocks_60 Denotes 60 standby clock cycles.
PWR_NumStandByClocks_61 Denotes 61 standby clock cycles.
PWR_NumStandByClocks_62 Denotes 62 standby clock cycles.
PWR_NumStandByClocks_63 Denotes 63 standby clock cycles.
PWR_NumStandByClocks_64 Denotes 64 standby clock cycles.
PWR_NumStandByClocks_65 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)

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 `f2802x_common/source/sci.c`, with `f2802x_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 SCICHAR2-0 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) masater/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) miltprocessor 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)

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 `f2802x_common/source/spi.c`, with `f2802x_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_SteInv_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_setSteInv (SPI_Handle spiHandle, const SPI_SteInv_e steinv)
- void SPI_setTriWire (SPI_Handle spiHandle, const SPI_TriWire_e triwire)
- void SPI_setTxDelay (SPI_Handle spiHandle, const uint8_t delay)
- void SPI_setTxFifoIntLevel (SPI_Handle spiHandle, const SPI_FifoLevel_e fifoLevel)
- void SPI_write (SPI_Handle spiHandle, const uint16_t data)
- void SPI_write8 (SPI_Handle spiHandle, const uint16_t data)

16.2.1 Define Documentation

16.2.1.1 SPI_SPICCR_CHAR_LENGTH_BITS

Definition:

```
#define SPI_SPICCR_CHAR_LENGTH_BITS
```

Description:

Defines the location of the SPICHAR3-0 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_SteInv_e

Description:

Enumerators:

SPI_SteInv_ActiveLow Denotes active low STE pin.

SPI_SteInv_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) masater/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 recieve 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_setSteInv (SPI_Handle *spiHandle*, const SPI_SteInv_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

16.2.4.46 void SPI_write8 (SPI_Handle *spiHandle*, const uint16_t *data*) `[inline]`

Writes a byte of data to the serial peripheral interface (SPI).

Parameters:

← ***spiHandle*** The serial peripheral interface (SPI) object handle

← ***data*** The data value

17 Timer

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 `f2802x_common/source/timer.c`, with `f2802x_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

18.1 Introduction

The Watchdog Timer API provides a set of functions for using the watchdog module.

This driver is contained in `f2802x_common/source/watchdog.c`, with `f2802x_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_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_enable (WDOG_Handle *wdogHandle*)

Enables the watchdog (WDOG) timer.

Parameters:
← ***wdogHandle*** The watchdog (WDOG) timer object handle

18.2.5.5 void WDOG_enableInt (WDOG_Handle *wdogHandle*)

Enables the watchdog (WDOG) timer interrupt.

Parameters:
← ***wdogHandle*** The watchdog (WDOG) timer object handle

18.2.5.6 void WDOG_enableOverRide (WDOG_Handle *wdogHandle*)

Enables the watchdog (WDOG) timer override.

Parameters:
← ***wdogHandle*** The watchdog (WDOG) timer object handle

18.2.5.7 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

18.2.5.8 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

18.2.5.9 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

18.2.5.10 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		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Broadband	www.ti.com/broadband
DSP	dsp.ti.com	Digital Control	www.ti.com/digitalcontrol
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Military	www.ti.com/military
Logic	logic.ti.com	Optical Networking	www.ti.com/opticalnetwork
Power Mgmt	power.ti.com	Security	www.ti.com/security
Microcontrollers	microcontroller.ti.com	Telephony	www.ti.com/telephony
RFID	www.ti-rfid.com	Video & Imaging	www.ti.com/video
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265