HY16F18 Series
Peripheral Driver
C Library

Table of Contents

Attention :

1、HYCON Technology Corp. reserves the right to change the content of this datasheet without further notice. For most up-to-date information, please constantly visit our website: http://www.hycontek.com .

2、HYCON Technology Corp. is not responsible for problems caused by figures or application circuits narrated herein whose related industrial properties belong to third parties.

3、Specifications of any HYCON Technology Corp. products detailed or contained herein stipulate the performance, characteristics, and functions of the specified products in the independent state. We does not guarantee of the performance, characteristics, and functions of the specified products as placed in the customer's products or equipment. Constant and sufficient verification and evaluation is highly advised.

4、Please note the operating conditions of input voltage, output voltage and load current and ensure the IC internal power consumption does not exceed that of package tolerance. HYCON Technology Corp. assumes no responsibility for equipment failures that resulted from using products at values that exceed, even momentarily, rated values listed in products specifications of HYCON products specified herein.

5、Notwithstanding this product has built-in ESD protection circuit, please do not exert excessive static electricity to protection circuit.

6、Products specified or contained herein cannot be employed in applications which require extremely high levels of reliability, such as device or equipment affecting the human body, health/medical equipments, security systems, or any apparatus installed in aircrafts and other vehicles.

7、Despite the fact that HYCON Technology Corp. endeavors to enhance product quality as well as reliability in every possible way, failure or malfunction of semiconductor products may happen. Hence, users are strongly recommended to comply with safety design including redundancy and fire-precaution equipments to prevent any accidents and fires that may follow.

8、Use of the information described herein for other purposes and/or reproduction or copying without the permission of HYCON Technology Corp. is strictly prohibited.

1. Overview

1.1. C Library Introduction

This document describes the HYCON ™ HY16F18 series driver reference manual. System-level software developers can use the HYCON ™ HY16F18 series driver to do the fast application software development, instead of using the register level programming, which can reduce the total development time significantly.

1.2. Relative Document

User can find the following documents in our website for other relative information.

http://www.hycontek.com/

2. SYS Driver

2.1. Introduction

The following functions are included in System Manager Section.

Item	Functions	Description
01	SYS_SleepFlagRead	Read Sleep Flag
02	SYS_SleepFlagClear	Clear Sleep Flag
03	SYS_WdogFlagRead	Read watch dog flag
04	SYS_WdogFlagClear	Clear watch dog flag
05	SYS_ResetFlagRead	Read reset flag of data
06	SYS_ResetFlagClear	Clear reset flag
07	SYS_BOR_FlagRead	Read BOR flag of data
08	SYS_BOR_FlagClear	Clear BOR flag
09	SYS_EnableGIE	Enable GIE and set priority level of interrupt
10	SYS_DisableGIE	Disable GIE
11	SYS_LowPower	Set the low power mode
12	SYS_ INTPriority	Set the interrupt priority level of interrupt vector

2.2. Functions

2.2.1. SYS_SleepFlagRead

- **Prototype**

  unsigned int SYS_SleepFlagRead (void);

- **Description**

  Read Sleep Flag of data from registers 0x40104[3].

  Read the value of register 0x40104[3].

- **Parameters**

  None

- **Include**

  Peripheral_lib/System.h

- **Return Value**

  0 : Normal

  1 : Chip has entered Sleep Mode

- **Example**

  /* Read Sleep Flag of data from registers 0x40104[3]. */

  unsigned char temp_flag; temp_flag=SYS_SleepFlagRead();

2.2.2. SYS_SleepFlagClear

- **Prototype**

  void SYS_SleepFlagClear(void);

- **Description**

  Clear Sleep Flag.

  Clear the register 0x40104[3]

- **Parameters**

  None

- **Include**

  Peripheral_lib/System.h

- **Return Value**

  None

- **Example**

  /* Clear Sleep Flag. */

  SYS_SleepFlagClear();

2.2.3. SYS_WdogFlagRead

- **Prototype**

  unsigned int SYS_WdogFlagRead (void);

- **Description**

  Read watch dog flag of data from registers 0x40104[2].

  Read the value of register 0x40104[2]

- **Parameters**

  None

- **Include**

  Peripheral_lib/System.h

- **Return Value**

  0 : Normal

  1 : Watch dog has triggered

- **Example**

```
/* Read watch dog flag of data from registers 0x40104[2]. */
unsigned char flag; flag=SYS_WdogFlagRead();
```

2.2.4. SYS_WdogFlagClear

- **Prototype**

```
void SYS_WdogFlagClear(void);
```

- **Description**

  Clear watch dog flag

  Clear the register 0x40104[2]

- **Parameters**

  None

- **Include**

  Peripheral_lib/System.h

- **Return Value**

  None

- **Example**

```
/* Clear watch dog flag */
SYS_WdogFlagClear();
```

2.2.5. SYS_ResetFlagRead

- **Prototype**

```
unsigned int SYS_ResetFlagRead (void);
```

- **Description**

  Read reset flag of data from registers 0x40104[1].

  Read the value of register 0x40104[1]

- **Parameters**

  None

- **Include**

  Peripheral_lib/System.h

- **Return Value**

  0 : Normal

  1 : The Reset Pin has reset

- **Example**

  /* Read reset flag of data from registers 0x40104[1]. */

  unsigned char flag; flag=SYS_ResetFlagRead();

2.2.6. SYS_ResetFlagClear

- **Prototype**

  void SYS_ResetFlagClear(void);

- **Description**

  Clear reset flag

  Clear the value of register 0x40104[1]

- **Parameters**

  None

- **Include**

  Peripheral_lib/System.h

- **Return Value**

  None

- **Example**

  /* Clear reset flag */

  SYS_ ResetFlagClear(); //0x40104[1]=0

2.2.7. SYS_BOR_FlagRead

- **Prototype**

  unsigned int SYS_BOR_FlagRead (void);

- **Description**

  Read BOR flag of data from registers 0x40104[0].

  Read the value of register 0x40104[0]

- **Parameters**

  None

- **Include**

  Peripheral_lib/System.h

- **Return Value**

  0 : Normal

  1 : BOR has triggered

- **Example**

  /* Read BOR flag of data from registers 0x40104[0]. */

  unsigned char flag; flag=SYS_BOR_FlagRead();

2.2.8. SYS_BOR_FlagClear

- **Prototype**

  void SYS_BOR_FlagClear(void);

- **Description**

  Clear BOR flag

  Clear the value of register 0x40104[0]

- **Parameters**

  None

- **Include**

  Peripheral_lib/System.h

- **Return Value**

  None

- **Example**

  /* Clear BOR flag */

  SYS_BOR_FlagClear(); //0x40104[0]=0

2.2.9. SYS_EnableGIE

- **Prototype**

  unsigned int SYS_EnableGIE (unsigned int uPriority unsigned short invector);

- **Description**

  Enable GIE and the corresponding interrupt vector, set the priority level of interrupt. The high level of priority will be responsed first. The priority of interrupt vector is set by SYS_INTPriority().

- **Parameters**

  uPriority [in] :

  Specify which priority level of interrupt can be responsed. It could be 0~4

  The priority level of the corresponding interrupt can be specified by SYS_INTPriority().

  0: No interrupts are allowed

  1: Only allows interrupts with the highest priority level.

  2: Only allows interrupts with the highest and second highest priority level.

  3: Only allows interrupts with the highest,second highest and lower priority level.

  4: Only allows interrupts with the highest,second highest,lower and lowest priority level

  intvector[in] :Select the interrupt vector[HW5:HW4:HW3:HW2:HW1:HW0]；It could be 0~0x3F. Each bit corresponding to an interrupt vector:HW5~HW0

For exemple： invector=[HW5:HW4:HW3:HW2:HW1:HW0]

Only enable ：HW0/HW3/HW5, invectior=0x29（101001B）;

Enable all interrupt vector: invector=0x3F（111111B）;

Disable all interrupt vector: invector=0x00（000000B）

- **Include**

Peripheral_lib/System.h

- **Return Value**

0: Operation successful

1: Incorrect argument

- **Example**

```
/* Enable GIE and allows interrupts with priority 0, 1, 2,3, enableHW0~HW5. */
SYS_EnableGIE(4, 0x3F);
```

2.2.10. SYS_DisableGIE

- **Prototype**

void SYS_DisableGIE (void);

- **Description**

Disable GIE

- **Parameters**

None

- **Include**

Peripheral_lib/System.h

- **Return Value**

None

- **Example**

```
/* Disable GIE. */
SYS_DisableGIE();
```

2.2.11. SYS_LowPower

- **Prototype**

unsigned char SYS_LowPower(unsigned char umode)

- **Description**

Set up and enable the low power mode. Need to open any an interrupt vector before open low power mode and switch to a low frequency source

Set up the register 0x40104[4]

- **Parameters**

umode[in] : the input range is 0~2

0: sleep mode

1: idle mode

2: waite mode

- **Include**

Peripheral_lib/System.h

- **Return Value**

0: Operation successful

1: Incorrect argument

- **Example**

```
/* Enable the sleep mode */
DrvGPIO_Open(E_PT1,0xFF,E_IO_IntEnable); // enable PT1 external interrupt vector
SYS_EnableGIE(4,0x3F);    // Enable GIE(Global Interrupt)
DrvCLOCK_SelectMCUClock(1,0); // switch to a low frequency source
SYS_LowPower(0); // Enable the sleep mode
```

2.2.12. SYS_ INTPriority

- **Prototype**

unsigned char SYS_INTPriority(unsigned short intvector,unsigned short upriority);

- **Description**

Specify priority level of the corresponding interrupt. Priority level is 0~3. 0 is the highest level

Note : Before use, must disable all interrupt to modify the interrupt priority level.

- **Parameters**

intvector[in] : the interrupt vector selection, input range of 0 to 5, respectively HW0 ~ HW5;;

upriority [in] : set up and enable the priority level of interrupt vector · setting range is from 0 to 3

0: the highest level of priority level

1: the second highest level of priority level

2: the lower level of priority level.

3: the lowest level of priority level

When set the interrupt priority level for the same level, the order for the interrupt response：

HW0 > HW1 > HW2 > …> HW5

- **Include**

Peripheral_lib/System.h

- **Return Value**

0: Operation successful

1: Incorrect argument

- **Example**

```
/* set the priority level of interrupt vector 0 as 1 * /
SYS_ INTPriority(0,1);
```

3. CLOCK Driver

3.1. Introduction

The following functions are included in Clock Manager Section.

Item	Functions	Description
01	DrvCLOCK_EnableHighOSC	Open high-speed oscillator
02	DrvCLOCK_CloseEHOSC	Turn off the external HSXT
03	DrvCLOCK_CloseIHOSC	Turn off the internal HSRC
04	DrvCLOCK_SelectIHOSC	Select HSRC mode
05	DrvCLOCK_EnableLowOSC	Open low-speed oscillator
06	DrvCLOCK_CloseELOSC	Turn off the external LSXT
07	DrvCLOCK_SelectMCUClock	Select the MCU Clock
08	DrvCLOCK_TrimHAO	Write to the HSRC Trim value
09	DrvCLOCK_CalibrateHAO	According to the factory calibration parameters of HAO to calibrate HAO
10	DrvCLOCK_SelectOHS_HS	Selecting External high-speed oscillator HSXT mode
11	DrvCLOCK_EnableENHAO	Enable Internal HAO
12	DrvCLOCK_SelectIHOSC_CalHAO	Select high-speed internal oscillator mode, and according to the factory calibration parameters of HAO to calibrate HAO

3.2. Type Definition

E_CLOCK_SOURCE

Enumeration Identifier	Value	Description
E_INTERNAL	0x0	Internal oscillator
E_EXTERNAL	0x1	External oscillator

E_TRIM_FREQUEN

Enumeration Identifier	Value	Description
TRIM_HAO2MHZ	0x0	Calibrate the frequency of HAO 2MHZ
TRIM_HAO4MHZ	0x1	Calibrate the frequency of HAO 4MHZ
TRIM_HAO10MHZ	0x2	Calibrate the frequency of HAO 10MHZ
TRIM_HAO16MHZ	0x3	Calibrate the frequency of HAO 16MHZ

3.3. Functions

3.3.1. DrvCLOCK_EnableHighOSC

● **Prototype**

unsigned int DrvCLOCK_EnableHighOSC(E_CLOCK_SOURCE uSource, unsigned int delay)

● **Description**

Open a high-speed oscillator, select from the external or internalSet the waiting time needed to stabilize the crystal

Configure the register 0x40300[5]=1 , 0x40300[1]=1 if the external OSC is selected as CPU clock source.

Configure the register 0x40300[5]=0，0x40300[0]=1 if the internal OSC is selected as CPU clock source.

● **Parameter**

uSource [in] :

0: Internal

1: External

delay[in] : Set the waiting time needed to stabilize the crystal. Input range：1~0xFFFFFFFF

Note that the current CPU cycles CPU_CLK, stabilization time of oscillator: t=(1/CPU_CLK)*4000*delay；

Refer to the current instruction cycle and the crystal frequency to start before set the parameter

The time needed to stabilize the EXT OSC :

4MHZ/8MHZ about 30ms

The time needed to stabilize the HAO :

2MHZ about 1.0ms

4MHZ about 0.5ms

10MHZ about 0.2ms

● **Include**

Peripheral_lib/DrvCLOCK.h

● **Return Value**

0: Operation successful

1: Incorrect argument

● **Example**

```
/* Open external high-speed oscillator, current CPU_CK=10MHZ/2, Open external 4MHZ, Delay 40ms =(1/10MHZ/2)*4000*50*/
DrvCLOCK_EnableHighOSC(E_EXTERNAL,50);
```

3.3.2. DrvCLOCK_CloseEHOSC

● **Prototype**

void DrvCLOCK_CloseEHOSC()

- **Description**

  Turn off the external high-speed oscillator . Need to turn on a available clock source before turn off the external high-speed oscillator if the external high-speed oscillator is the CPU clock source.

  Configure the register 0x40300[1]=0

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvCLOCK.h

- **Return Value**

  None

- **Example**

```
/*Turn off the external high-speed oscillator*/
DrvCLOCK_EnableHighOSC(E_INTERNAL,50); //Open internal high-speed oscillator
DrvCLOCK_CloseEHOSC(); / /Turn off the external high-speed oscillator
```

3.3.3. DrvCLOCK_CloseIHOSC

- **Prototype**

  void DrvCLOCK_CloseIHOSC()

- **Description**

  Turn off the internal high-speed oscillator before switching the CPU clock source to available source

  Configure the register 0x40300[0]=0

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvCLOCK.h

- **Return Value**

  None

- **Example**

```
/* Turn on the external high-speed oscillator and turn off the internal high-speed oscillator that is the CPU clock source */
DrvCLOCK_EnableHighOSC(E_EXTERNAL,50); // Open external high-speed oscillator
DrvCLOCK_CloseIHOSC();    // Close internal high-speed oscillator
```

3.3.4. DrvCLOCK_SelectIHOSC

- **Prototype**

  unsigned int DrvCLOCK_SelectIHOSC(uMode)

- **Description**

Select high-speed internal oscillator mode

Configure the register 0x40300[4:3]

- **Parameter**

uMode [in]

0 : 2MHz, 0x40300[4:3]=00b

1 : 4MHz, 0x40300[4:3]=01b

2 : 10MHz, 0x40300[4:3]=10b

3 : Rsv

- **Include**

Peripheral_lib/DrvCLOCK.h

- **Return Value**

0: Operation successful

other: Incorrect argument

- **Example**

/* Select high-speed internal oscillator 4MHz mode*/

DrvCLOCK_SelectIHOSC(1);

3.3.5. DrvCLOCK_EnableLowOSC

- **Prototype**

unsigned int DrvCLOCK_EnableLowOSC(E_CLOCK_SOURCE uSource，unint delay)

- **Description**

Open the low-speed oscillator; select the oscillator from an external or internal. Set the waiting time needed to stabilize the crystal

Configure the register 0x40300[6]=1, 0x40300[2]=1

- **Parameter**

uSource [in] :

0: Internal LSRC

1: External LSXT

delay[in] : Set the waiting time needed to stabilize the crystal. Need to refer to the current instruction cycle. Input range：0~0xFFFFFFFF

The time needed to stabilize the EXT OSC : 32768HZ about 1.3s

The time needed to stabilize the internal OSC : about 510us

- **Include**

Peripheral_lib/DrvCLOCK.h

- **Return Value**

0: Operation successful

1: Incorrect argument

- **Example**

```
/* Open the external low-speed oscillator, set the delay time 1.3s */
DrvCLOCK_EnableLowOSC(E_EXTERNAL,130000);
```

3.3.6. DrvCLOCK_CloseELOSC

- **Prototype**

  void DrvCLOCK_CloseELOSC()

- **Description**

  Turn off the external low-speed oscillator

  Configure the register 0x40300[2]=0

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvCLOCK.h

- **Return Value**

  None

- **Example**

```
/* Turn on the internal low-speed oscillator and then turn off the external low-speed oscillator*/
DrvCLOCK_EnableLowOSC(E_INTERNAL,130000); //turn on internal low-speed oscillator
DrvCLOCK_CloseELOSC();    //close external low-speed oscillator
```

3.3.7. DrvCLOCK_SelectMCUClock

- **Prototype**

  unsigned int DrvCLOCK_SelectMCUClock(uSource,uDiv)

- **Description**

  Select the MCU Clock from HS_CK, or LS_CK and Pre-scale. Configure register 0x40308[0] / 0x40308[1]

- **Parameter**

  uSource [in] :

  0 : HS_CK

  1 : LS_CK

  uDiv [in] :

  0 : ÷1

  1 : ÷2

- **Include**

  Peripheral_lib/DrvCLOCK.h

- **Return Value**

  0: Operation successful

  other: Incorrect argument

- **Example**

  /* Select the MCU Clock from HS_CK and Pre-scale 2 */

  DrvCLOCK_SelectMCUClock(0, 1);

3.3.8. DrvCLOCK_TrimHAO

- **Prototype**

  unsigned int DrvCLOCK_TrimHAO(uTrim)

- **Description**

  Write to the internal oscillator HAO Trim value

  Configure the register 0x40304[7:0]

- **Parameter**

  uTrim [in] : the internal oscillator Trim value, the input range is : 0~0xFF

- **Include**

  Peripheral_lib/DrvCLOCK.h

- **Return Value**

  0: Operation successful

  other: Incorrect argument

- **Example**

  /*Write 0x80 to the internal oscillator Trim value*/

  DrvCLOCK_TrimHAO(0x80);

3.3.9. DrvCLOCK_CalibrateHAO

- **Prototype**

  void DrvCLOCK_CalibrateHAO(short int uMHZ)

- **Description**

  According to the factory calibration parameters of HAO to calibrate HAO, and need to corresponding to the selected HAO frequency. Configure the register 0x40304[7:0]

- **Parameter**

  uMHZ [in] : the HAO frequency to calibrate

  0: calibrate 2MHZ

  1: calibrate 4MHZ

  2: calibrate 10MHZ

  3: Rsv

- **Include**

  Peripheral_lib/DrvCLOCK.h

- **Return Value**

None

- **Example**

/* Calibrate internal OSC 4MHZ*/

DrvCLOCK_SelectIHOSC(1); //setting HAO=4MHZ;

DrvCLOCK_CalibrateHAO(1); //calibrate 4MHZ ;

3.3.10. DrvCLOCK_SelectOHS_HS

- **Prototype**

unsigned int DrvCLOCK_SelectOHS_HS(unsigned int uMode)

- **Description**

Selecting External high-speed oscillator HSXT mode, the mode of HSXT can be more than 4MHz or less than 4MHz.

Configure the register 0x40300[7]

- **Parameter**

uMode [in] : Selecting External high-speed oscillator HSXT mode. The input range is : 0~1

0 : HSXT<4MHz ; 1 : HSXT>4MHz ;

- **Include**

Peripheral_lib/DrvCLOCK.h

- **Return Value**

0 : Operation successful

1 : Incorrect argument

- **Example**

/*Select external high-speed oscillator (HSXT)>4MHZ */

DrvCLOCK_SelectOHS_HS(1); //Select HSXT > 4MHZ;

3.3.11. DrvCLOCK_EnableENHAO

- **Prototype**

void DrvCLOCK_EnableENHAO (void)

- **Description**

Enable Internal HAO

Configure the register 0x40300[0]=1

- **Parameter**

None

- **Include**

Peripheral_lib/DrvCLOCK.h

- **Return Value**

None

- **Example**

```
/*Enable Internal HAO */
DrvCLOCK_EnableENHAO();    //ENHAO=1b
```

3.3.12. DrvCLOCK_SelectIHOSC_CalHAO

- **Prototype**

unsigned int DrvCLOCK_SelectIHOSC_CalHAO(unsigned int uMode)

- **Description**

Select high-speed internal oscillator mode, and according to the factory calibration parameters of HAO to calibrate HAO. Configure the register 0x40300[4:3]、0x40304[7:0]

- **Parameter**

uMode [in] : HAO frequency value, input range: 0~2

0 : 2MHz, 0x40300[4:3]=00b

1 : 4MHz, 0x40300[4:3]=01b

2 : 10MHz, 0x40300[4:3]=10b

- **Include**

Peripheral_lib/DrvCLOCK.h

- **Return Value**

0 : Operation successful

1 : Incorrect argument

- **Example**

```
/*Select HAO=4MHz, and calibrate (HAO)4MHZ=4MHz<2% frequency error */
DrvCLOCK_SelectIHOSC_CalHAO(1);
```

4. TIMER/WDT Driver

4.1. Introduction

The following functions are included in Timer Manager Section.

Item	Functions	Description
01	DrvWDT_Open	Open WDT
02	DrvWDT_CounterRead	Read the current WDT counter
03	DrvWDT_ClearWDT	Watch dog timer clear
04	DrvWDT_ResetEnable	Enable Watch dog(WDT) Reset mode
05	DrvTMA_Open	Enable timer A
06	DrvTMA_Close	Close timer A
07	DrvTMA_CounterRead	Read the current TMA counter
08	DrvTMA_ClearTMA	Clear Timer A
09	DrvTIMER_EnableInt	Enable the specified timer interrupt.
10	DrvTIMER_DisableInt	Disable the specified timer interrupt
11	DrvTIMER_GetIntFlag	Get the interrupt flag status
12	DrvTIMER_ClearIntFlag	Clear the interrupt flag
13	DrvTMB_Open	Enable timer B
14	DrvTMBC_Clk_Source	Timer B,C clock source selection
15	DrvTMBC_Clk_Disable	Disable timer B,C clock
16	DrvTMB_ ClearTMB	Clear Timer B
17	DrvTMB_CounterRead	Read the current TMB counter
18	DrvTMB_Close	Close timer B
19	DrvPWM0_Open	Enable PWM and PWM0 mode
20	DrvPWM1_Open	Enable PWM and PWM1 mode
21	DrvPWM_CountCondition	PWM count condition parameter
22	DrvPWM0_Close	PWM0 off
23	DrvPWM1_Close	PWM1 off
24	DrvCAPTURE1_Open	Enable Capture1
25	DrvCAPTURE2_Open	Enable Capture2
26	DrvCAPTURE1_Read	Read Capture1 counter
27	DrvCAPTURE2_Read	Read Capture2 counter
28	DrvCAPTURE_IPort	Select the capture input pin

4.2. Type Definition

E_WDT_MODE

Enumeration Identifier	Value	Description
E_IRQ	0x0	IRQ mode
E_RST	0x1	RESET mode

E_WDT_PRE_SCALER

Enumeration Identifier	Value	Description
E_PRE_SCALER_D2	0x0	WDT_CK / 2
E_PRE_SCALER_D8	0x1	WDT_CK / 8
E_PRE_SCALER_D32	0x2	WDT_CK / 32
E_PRE_SCALER_D128	0x3	WDT_CK / 128
E_PRE_SCALER_D512	0x4	WDT_CK / 512
E_PRE_SCALER_D2048	0x5	WDT_CK / 2048
E_PRE_SCALER_D8192	0x6	WDT_CK / 8192
E_PRE_SCALER_D32768	0x7	WDT_CK / 32768

E_TIMER_CHANNEL

Enumeration Identifier	Value	Description
E_TMA	0x0	Specify the timer channel – A
E_TMB	0x1	Specify the timer channel – B
E_TMC	0x2	Specify the timer channel - C
E_WDT	0x3	Specify the WDT

E_TMB_MODE

Enumeration Identifier	Value	Description
E_TMB_MODE0	0x0	16-bit saw tooth waveform count up TBC0 for the maximum limit
E_TMB_MODE1	0x1	16-bit triangular waveform up and down the count range of 0 to TBC0
E_TMB_MODE2	0x2	The two independent 8-bit saw tooth type count, up to TBC0 bit 15-8 and bit 7-0 for the maximum limit
E_TMB_MODE3	0x3	The two 8-bit saw tooth type count, TBR[15:0] will be automatically added by 1, only after TBR[7:0] overflow

E_TRIGGER_SOURCE

Enumeration Identifier	Value	Description
E_TMB_NORMAL	0x0	Always Enable
E_TMB_CMP_HIGH	0x1	CMP high trigger
E_TMB_OP_HIGH	0x2	OP high trigger
E_TMB_GPIO_HIGH	0x3	GPIO high trigger

E_DRVTIMER_CLOCK_SOURCE

Enumeration Identifier	Value	Description
E_HS_CK	0	TMA clock source from HS_CK
E_LS_CK	1	TMA clock source from LS_CK

E_CAPTURE_SOURCE

Enumeration Identifier	Value	Description
E_TMC_CMPO	0x0	Comparator output
E_TMC_OPOD	0x1	Rail-to-rail OP amp digital output
E_TMC_LSCK	0x2	Low speed clock source
E_TMC_TCI0	0x3	TC1 form I/O port
E_TMC_TCI1	0x0	TC2 form I/O port
E_TMC_ASTC0	0X1	Inupt source of TC2 is the same as TC1

4.3. Functions

4.3.1. DrvWDT_Open

- **Prototype**

  uint32_t DrvWDT_Open (E_WDT_MODE eMode , E_WDT_PRE_SCALER eWDTpreScaler)

- **Description**

  Enable WDT engine clock and set WDT time-out interval and set WDT mode.

  Configure the register 0x40108[2:0] / 0x40108[4]=1

- **Parameter**

  eMode [in] : the operating mode of WDT

  0 : Timer mode

  1 : Reset mode

  eWDTpreScaler [in] : the prescaler of WDT clock source

  0 : WDT_CK / 2

  1 : WDT_CK / 8

  2 : WDT_CK / 32

  3 : WDT_CK / 128

  4 : WDT_CK / 512

  5 : WDT_CK / 2048

  6: WDT_CK / 8192

  7: WDT_CK / 32768

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Value**

  0: Operation successful

  1: WDT open fail

  - **Example**

  /* Set the WDT in IRQ mode and CLK / 32 */

  DrvWDT_Open(E_IRQ , E_PRE_SCALER_D32);

4.3.2. DrvWDT_CounterRead

- **Prototype**

  uint32_t DrvWDT_CounterRead (void)

- **Description**

  Read the current WDT counter.

  Read the register 0x40108 [30:16]

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Value**

  The return values of WDT counter.

- **Example**

/* Read the return values of WDT counter */

unsigned int data；data=DrvWDT_CounterRead();

4.3.3. DrvWDT_ClearWDT

- **Prototype**

  void DrvWDT_ClearWDT (void)

- **Description**

  Watch dog timer clear.

  Configure the register 0x40108[5]=1, and 0x40108 [30:16] automatically becomes 0 after clear.

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Value**

  None

- **Example**

  /* Watch dog timer clear. */

  DrvWDT_ClearWDT();

4.3.4. DrvWDT_ResetEnable

- **Prototype**

  void DrvWDT_ ResetEnable(void)

- **Description**

  Enable Watch dog(WDT) Reset mode .

  Configure the register 0x40108[6]=1b

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Value**

None

- **Example**

/* Enable Watch dog(WDT) Reset mode. */

DrvWDT_ResetEnable();

4.3.5. DrvTMA_Open

- **Prototype**

unsigned int DrvTMA_Open (eTMAOV, E_DRVTIMER_CLOCK_SOURCE uclk)

- **Description**

Enable timerA ,set counter value and clock source of TMA.

Configure the register 0x40C00[5]=1b, 0x40C00[3:0], 0x40308[3], 0x40308[2]

- **Parameter**

eTMAOV [in] : Specify timer A overflow condition.

0 : taclk/2

1 : taclk/4

2 : taclk/8

3 : taclk/16

4 : taclk/32

5 : taclk/64

6 : taclk/128

7 : taclk/256

8 : taclk/512

9 : taclk/1024

10 : taclk/2048

11 : taclk/4096

12 : taclk/8192

13 : taclk/16384

14 : taclk/32768

15: taclk/65536

uclk[in] : Specify timer A clock source.

0: HS_CK

1: LS_CK

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Enable timerA and set counter value is taclk/8. */
DrvTMA_Open(2, 0);
```

4.3.6. DrvTMA_Close

- **Prototype**

  void DrvTMA_Close (void)

- **Description**

  Close timerA

  Configure the register 0x40C00[5]=0b

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Value**

  None

- **Example**

```
/* Disable timerA */
DrvTMA_Close();
```

4.3.7. DrvTMA_CounterRead

- **Prototype**

  unsigned int DrvTMA_CounterRead (void)

- **Description**

  Read the current TMA counter.

  Read the register 0x40C00[15:0]

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Value**

  The return values of TMA counter.

- **Example**

```
/* Read the current TMA counter */
unsigned short TMA_counter; TMA_counter =DrvTMA_CounterRead();
```

4.3.8. DrvTMA_ClearTMA

- **Prototype**

  void DrvTMA_ClearTMA (void)

- **Description**

  Clear Timer A counter.

  Configure the register 0x40C00[4]=1, and 0x40C00[15:0] automatically becomes 0 after clear.

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Value**

  None

- **Example**

```
/* Clear Timer A. */
DrvTMA_ClearTMA();
```

4.3.9. DrvTIMER_EnableInt

- **Prototype**

  unsigned int DrvTIMER_EnableInt (E_TIMER_CHANNEL ch)

- **Description**

  This function is used to enable the specified timer interrupt WDT/Timer A/Timer B /Timer C.

  Configure the corresponding bit of register 0x40004[20:16] interrupt function=1

- **Parameter**

  ch [in] : timer interrupt source, the input range is 0~4

  0：TMA　　1：TMB　2：TMC C0

  3：TMC C1　4：WDT

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Value**

  0: Operation successful

  Other : Invalid

- **Example**

```
/* Enable Timer-A interrupt function */
DrvTIMER_EnableInt(E_TMA);
```

4.3.10. DrvTIMER_DisableInt

- **Prototype**

  unsigned int DrvTIMER_DisableInt (E_TIMER_CHANNEL ch)

- **Description**

  This function is used to disable the specified timer interrupt WDT/Timer A/Timer B /Timer C.

  Configure a corresponding bit of register 0x40004[20:16] interrupt function=0

- **Parameter**

  ch [in] : timer interrupt source, the input range is 0~4

  0：TMA　　1：TMB　2：TMC C0

  3：TMC C1　4：WDT

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Value**

  0: Operation successful

  Other : Invalid

- **Example**

```
/* Disable Timer-A interrupt function */
DrvTIMER_DisableInt(E_TMA);
```

4.3.11. DrvTIMER_GetIntFlag

- **Prototype**

  unsigned int DrvTIMER_GetIntFlag (E_TIMER_CHANNEL ch)

- **Description**

  Get the interrupt flag status from the specified timer channe WDT/Timer A/Timer B /Timer C.

  Read a corresponding bit of register 0x40004[4:0]

- **Parameter**

  ch [in] : timer interrupt source, the input range is 0~4

  0：TMA　　1：TMB　2：TMC C0

  3：TMC C1　4：WDT

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Value**

  0: No interrupt

  1: Interrupt occurred

- **Example**

```
/* Get the interrupt flag status from Timer-A */
DrvTIMER_GetIntFlag(E_TMA);
```

4.3.12. DrvTIMER_ClearIntFlag

- **Prototype**

  unsigned int DrvTIMER_ClearIntFlag (E_TIMER_CHANNEL ch)

- **Description**

  Clear the interrupt flag of the specified timer channel WDT/Timer A/Timer B /Timer C.

  Clear a corresponding bit of register 0x40004[4:0]

- **Parameter**

  ch [in] : timer interrupt source, the input range is 0~4

  0：TMA　　1：TMB　2：TMC C0

  3：TMC C1　4：WDT

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Value**

  0: Operation successful

  Other : Invalid

- **Example**

```
/* Clear Timer-A interrupt flag */
DrvTIMER_ClearIntFlag(E_TMA);
```

4.3.13. DrvTMB_Open

- **Prototype**

  unsigned int DrvTMB_Open (E_TMB_MODE eTMBmode, E_TRIGGER_SOURCE eTriSource, eTMBOV)

- **Description**

  Enable TMB and set TMB counter value and TMB mode and trigger source.

  Support compare and capture and counting and timing functions.

  Configure the register 0x40C0C[15:0], 0x40C04[3:0] / 0x40C04[5]=1b。

- **Parameter**

  eTMBmode [in] : Specify timer B counting mode.

  0: TBR is in UP mode. In the UP mode, the TBR is increase by 1 for every positive edge of TBCLK. If it is larger than TBC0, TBR changes to 0 for next positive edge of TBCK and TMBIF is change to 1. Then, TBR starts to up count again.

  1: TBR is in UP/Down mode. In the UP mode, the TBR is increase by 1 for every positive edge of TBCLK. If it is equal to TBC0, TBR changes to down mode and TMR become to decrease by 1 for every positive edge of TBCK. Until TBR down count to 0, TMBIF changes to 1 and TBR starts to up count again.

  2: TBR is in two 8-bit PWM mode. The TBR is broke to two independent 8-bit UP counters: TBR[15:8]

and TBR[7:0]. The TBR[15:8] up limit is controlled by TBC[15:8] and TBR[7:0] up limit is controlled by TBC[7:0]. Both of the TBRs are increase by 1 for every positive edge of TBCLK. If TBR[15:8] is equal to TBC0[15:8], then the next positive edge of TBCLK would make TBR[15:8] to be 0. TMBIF still remains 0. If TBR[7:0] is equal to TBC0[7:0], then the next positive edge of TBCLK would make TBR[7:0] to be 0. TMBIF changes to 1.

3: TBR is in step increment mode. TBR is break into two counters TBR[15:8] and TBR[7:0]. Both of them are in Up mode. However, the limit of TBR[7:0] is controlled by TBC0[7:0]. The TBR[7:0] is increase by 1 for every positive edge of TBCLK. If TBR[7:0] is equal to TBC0[7:0], then it would change to 0 at next positive edge of TBCLK. Moreover, the TMBIF changes to 1 and TBR[15:8] increases by 1.

eTriSource [in] : Specify TMB trigger source.

0: Always Enable

1: CMP high trigger

2: OP high trigger

3:TMC output high trigger (CPI1)

eTMAOV [in] : Specify overflow condition. (0~0xffff)

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Enable timerB mode1 and set counter value is taclk/32 and trigger form OP. */
DrvTMB_Open(E_TMB_MODE0, E_TMB_OP_HIGH,4);
```

4.3.14. DrvTMBC_Clk_Source

- **Prototype**

unsigned int DrvTMBC_Clk_Source (E_DRVTIMER_CLOCK_SOURCE uclk, uPerScale)

- **Description**

Timer B,C clock source selection and clock divider selection.

Configure the register 0x40308[7:6], 0x40308[5:4]

- **Parameter**

uclk[in] : Specify timer B,C clock source, the input range is 0~1

0: HS_CK

1: LS_CK

uPerScale [in] : Specify timer B,C clock divider, , the input range is 0~3

0: ÷1

1: ÷2

2: ÷4

3: ÷8

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

/* Select the timer B clock from HS_CK, divider of 2. */

DrvTMB_Clk_Source(0,1);

4.3.15. DrvTMBC_Clk_Disable

- **Prototype**

viod DrvTMBC_Clk_Disable (viod)

- **Description**

Disable timer B,C clock.

Configure the register 0x40308[6]=0。

- **Parameter**

None

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Value**

None

- **Example**

/* Disable timer B,C clock.*/

DrvTMBC_Clk_Disable();

4.3.16. DrvTMB_ ClearTMB

- **Prototype**

void DrvTMA_ClearTMB (void)

- **Description**

Clear Timer B.

Configure the register 0x40C04[4]=1, and 0x40C08[15:0] automatically change to 0 after clear.

- **Parameter**

None

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Value**

None

- **Example**

/* Clear Timer B counter */

DrvTMB_ClearTMB();

4.3.17. DrvTMB_CounterRead

- **Prototype**

unsigned int DrvTMB_CounterRead (void)

- **Description**

Read the current TMB counter. Read the register 0x40C08[15:0]

- **Parameter**

None

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Value**

The return values of TMB counter.

- **Example**

/* Read the current TMB counter */

unsigned short TMB_counter; TMB_counter =DrvTMB_CounterRead();

4.3.18. DrvTMB_Close

- **Prototype**

void DrvTMB_Close (void)

- **Description**

Close timer B

Configure the register 0x40C04[5]=0

- **Parameter**

None

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Value**

None

- **Example**

/* Disable timerB */

DrvTMB_Close();

4.3.19. DrvPWM0_Open

- **Prototype**

unsigned int DrvPWM0_Open (uPWM_Mode , uInv, uOuputPin)

- **Description**

Enable PWM and PWM0 operation mode selection. Select IO port to output. PWM output inverse control

Configure the register 0x40C04[18:16] / 0x40C04[19]、0x40840[4:2] / 0x40840[0]=1b

- **Parameter**

uPWM_Mode [in] : PWM Operation mode selection

0: PWM A 1: PWM B

2: PWM C 3: PWM D

4 : RSV 5 : PWM F

6 : PWM G 7 : PWM G

uInv[in] : PWM output inverse control.

0 : inverse

1 : Normal

uOuputPin[in] :PWM IO port selection

0 : Port 1.0 =PWMO1, Port 1.1 =PWMO2

1 : Port 1.2 =PWMO1, Port 1.3 =PWMO2

2 : Port 1.4 =PWMO1, Port 1.5 =PWMO2

3 : Port 1.6 =PWMO1, Port 1.7 =PWMO2

4 : Port 2.0 =PWMO1, Port 2.1 =PWMO2

5 : Port 2.2 =PWMO1, Port 2.3 =PWMO2

6 : Port 2.4 =PWMO1, Port 2.5 =PWMO2

7 : Port 2.6 =PWMO1, Port 2.7 =PWMO2

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Enable PWM, PWM0 working in PWMA mode, the reverse signal, PT1.0 Output. */
DrvPWM0_Open(0, 0, 0);
```

4.3.20. DrvPWM1_Open

● **Prototype**

unsigned int DrvPWM1_Open (uPWM_Mode , uInv, uOuputPin)

● **Description**

Enable PWM and PWM1 operation mode selection. Select IO port to output. PWM output inverse control

Set 0x40C04[23:20] / 0x40840[4:2] / 0x40840[1]=1b

● **Parameter**

uPWM_Mode [in] : PWM operation mode selection

0: PWM A 1: PWM B

2: PWM C 3: PWM D

4 : RSV 5 : PWM F

6 : PWM G 7 : PWM G

uInv[in] : PWM output inverse control..

0 : inverse

1 : Normal

uOuputPin[in] :PWM IO port selection

0 : Port 1.0 =PWMO1, Port 1.1 =PWMO2

1 : Port 1.2 =PWMO1, Port 1.3 =PWMO2

2 : Port 1.4 =PWMO1, Port 1.5 =PWMO2

3 : Port 1.6 =PWMO1, Port 1.7 =PWMO2

4 : Port 2.0 =PWMO1, Port 2.1 =PWMO2

5 : Port 2.2 =PWMO1, Port 2.3 =PWMO2

6 : Port 2.4 =PWMO1, Port 2.5 =PWMO2

7 : Port 2.6 =PWMO1, Port 2.7 =PWMO2

● **Include**

Peripheral_lib/DrvTIMER.h

● **Return Value**

0: Operation successful

Other : Incorrect argument

● **Example**

```
/* Enable PWM, PWM1 working in PWMA mode, the reverse signal., PT1.1 output */
DrvPWM1_Open(0, 0, 0);
```

4.3.21. DrvPWM_CountCondition

● **Prototype**

void DrvPWM_CountCondition (uTBC2, uTBC1)

● **Description**

PWM0/PWM1 count condition parameter (TBC2, TBC1) setting.

Configure the register 0x40C10[15:0](TBC1) / 0x40C10[31:16](TBC2)

- **Parameter**

  uTBC1 [in] : PWM0 count condition, specify the TBC1 condition. (The range is 0~0xFFFF)

  uTBC2 [in] : PWM1 count condition, specify the TBC2 condition. (The range is 0~0xFFFF)

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Value**

  None

- **Example**

```
/* Set TBC1, TBC2 value of 0x4000 */
DrvPWM_CountCondition(0x4000, 0x4000);
```

4.3.22. DrvPWM0_Close

- **Prototype**

  void DrvPWM0_Close (void)

- **Description**

  PWM0 off

  Configure the register 0x40840[0]=0b

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Value**

  None

- **Example**

```
/*PWM0 off */
DrvPWM0_Close();
```

4.3.23. DrvPWM1_Close

- **Prototype**

  void DrvPWM1_Close (void)

- **Description**

  PWM1 off

  Configure the register 0x40840[1]=0b

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Value**

  None

- **Example**

  /*PWM1 off */

  DrvPWM1_Close();

4.3.24. DrvCAPTURE1_Open

- **Prototype**

  unsigned int DrvCapture1_Open (CAPTURE_SOURCE uChannel , uDivider, uEdge)

- **Description**

  Enable Capture1, Selected the input sources, pre-scale, the trigger source control.

  Configure the register 0x40C14[21:20] / 0x40C14[19:16] / 0x40C14[1] / 0x40C14[0]=1。

- **Parameter**

  uChannel [in] : Capture 1 input source selection, the input range is 0~3

  0 : CMPO

  1 : OPOD

  2 : LS_CK

  3 : TCI1

  uDivider [in] : Input clock prescale, the input range is 0~15

  0: ÷1 8: ÷256

  1: ÷2 9: ÷512

  2: ÷4 10: ÷1024

  3: ÷8 11: ÷2048

  4: ÷16 12: ÷4096

  5: ÷32 13: ÷8192

  6: ÷64 14: ÷16384

  7: ÷128 15: ÷32768

  uEdge [in] :

  0 : Rising-edge trigger

  1 : Falling-edge trigger

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

  /* Enable capture1, Choose TCI1 input , divided by 2048，rising-edge trigger */

  DrvCapture1_Open(3, 11, 0);

4.3.25. DrvCAPTURE2_Open

- **Prototype**

  unsigned int DrvCapture2_Open (CAPTURE_SOURCE uChannel, uEdge)

- **Description**

  Enable Capture2, Selected the input sources, pre-scale, the trigger source control.

  Configure the register 0x40C14[22] / 0x40C14[2] / 0x40C14[0]=1

- **Parameter**

  uChannel [in] : Capture 2 input source selection.

  0:TCI2 from GPIO

  1: With the Capture1 the same trigger source

  uEdge [in] :

  0: Rising -edge trigger

  1: Falling -edge trigger

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Enable capture2, choose with the Capture1 the same trigger source, rising -edge trigger.  */
DrvCapture2_Open(1, 0);
```

4.3.26. DrvCAPTURE1_Read

- **Prototype**

  unsigned int DrvCapture1_Read (void)

- **Description**

  Read Capture1 counter.

  Configure the register 0x40C18[15:0]

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Value**

  Capture the results TCR0(0~0xffff)

- **Example**

/* Read Capture1 counter */

unsigned short tcounter; tcounter=DrvCapture1_Read();

4.3.27. DrvCAPTURE2_Read

- **Prototype**

  unsigned int DrvCapture2_Read (void)

- **Description**

  Read Capture2 counter.

  Read the register 0x40C18[31:16]

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Value**

  Capture2 the results TCR1(0~0xffff)

- **Example**

  /* Read Capture2 counter */

  unsigned short tcounter; tcounter=DrvCapture2_Read();

4.3.28. DrvCAPTURE_IPort

- **Prototype**

  unsigned int DrvCapture_Iport (uInputPin)

- **Description**

  Select the capture input pin.

  Configure the register 0x40840[7:5]

- **Parameter**

  uInputPin[in]

  0 : Port 1.0 =TCI1, Port 1.1 =TCI2

  1 : Port 1.2 =TCI1, Port 1.3 =TCI2

  2 : Port 1.4 =TCI1, Port 1.5 =TCI2

  3 : Port 1.6 =TCI1, Port 1.7 =TCI2

  4 : Port 2.0 =TCI1, Port 2.1 =TCI2

  5 : Port 2.2 =TCI1, Port 2.3 =TCI2

  6 : Port 2.4 =TCI1, Port 2.5 =TCI2

  7 : Port 2.6 =TCI1, Port 2.7 =TCI2

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Set capture input pin of Port 1.6=TCI1, Port1.7=TCI2 */
DrvCapture_Iport(3);
```

5. GPIO Driver

5.1. Introduction

The following functions are included in GPIO Manager Section.

Item	Functions	Description
01	DrvGPIO_Open	Set the GPIO operation mode
02	DrvGPIO_SetBit	Set the specified GPIO pin to 1
03	DrvGPIO_ClrBit	Set the specified GPIO pin to 0.
04	DrvGPIO_GetBit	Get the pin value
05	DrvGPIO_SetPortBits	Set the output port value
06	DrvGPIO_ClrPortBits	Clear the output port value
07	DrvGPIO_GetPortBits	Get the input port value
08	DrvGPIO_IntTrigger	Set the specified interrupt pin mode
09	DrvGPIO_ClkGenerator	Set IO sampling clock input source
10	DrvGPIO_ClearIntFlag	Clear external interrupt flag
11	DrvGPIO_GetIntFlag	Get the interrupt flag
12	DrvGPIO_Close	Close the GPIO operation mode
13	DrvGPIO_EnableAnalogPin	Disabled the GPIO digital mode.Enable the GPIO analog mode
14	DrvGPIO_PT1_EnableINPUT	Enable the input mode for specified pin of PT1
15	DrvGPIO_PT1_DisableINPUT	Disable the input mode for specified pin of PT1
16	DrvGPIO_PT1_EnablePullHigh	Enable the pull up mode for specified pin of PT1
17	DrvGPIO_PT1_DisablePullHigh	Disable the pull up mode for specified pin of PT1
18	DrvGPIO_PT1_EnableOUTPUT	Enable the output mode for specified pin of PT1
19	DrvGPIO_PT1_DisableOUTPUT	Disable the output mode for specified pin of PT1
20	DrvGPIO_PT1_EnableINT	Enable the external interrupt for the specified pin of PT1
21	DrvGPIO_PT1_DisableINT	Disable the external interrupt for the specified pin of PT1
22	DrvGPIO_PT1_IntTriggerPorts	Configure the external interrupt trigger method for PT1
23	DrvGPIO_PT1_IntTriggerBit	Configure the external interrupt trigger method for the specified pin of PT1
24	DrvGPIO_PT1_GetIntFlag	Clear the interrupt flag of the specified pin
25	DrvGPIO_PT1_ClearIntFlag	Get the interrupt flag of the specified pin
26	DrvGPIO_PT1_GetPortBits	Get the input port value from the specified pin
27	DrvGPIO_PT1_SetPortBits	Set the output port value of the specified pin
28	DrvGPIO_PT1_ClrPortBits	Clear the output port value of the specified pin
29	DrvGPIO_PT2_EnableINPUT	Enable the input mode of the specified pin
30	DrvGPIO_PT2_DisableINPUT	Disable the input mode of the specified pin
31	DrvGPIO_PT2_EnablePullHigh	Enable the pull up of the specified pin
32	DrvGPIO_PT2_DisablePullHigh	Disable the pull up of the specified pin
33	DrvGPIO_PT2_EnableOUTPUT	Enable the output mode of the specified pin
34	DrvGPIO_PT2_DisableOUTPUT	Disable the output mode of the specified pin
35	DrvGPIO_PT2_EnableINT	Enable the external interrupt function of the specified pin
36	DrvGPIO_PT2_DisableINT	Disable the external interrupt function of the

		specified pin
37	DrvGPIO_PT2_IntTriggerPorts	Configure the external interrupt trigger method for PT2
38	DrvGPIO_PT2_IntTriggerBit	Configure the external interrupt trigger method for the specified pin of PT2
39	DrvGPIO_PT2_GetIntFlag	Clear the interrupt flag of the specified pin
40	DrvGPIO_PT2_ClearIntFlag	Get the interrupt flag of the specified pin
41	DrvGPIO_PT2_GetPortBits	Get the input port value from the specified pin
42	DrvGPIO_PT2_SetPortBits	Set the output port value of the specified pin
43	DrvGPIO_PT2_ClrPortBits	Clear the output port value of the specified pin
44	DrvGPIO_PT3_EnableINPUT	Enable the input mode of the specified pin
45	DrvGPIO_PT3_DisableINPUT	Disable the input mode of the specified pin
46	DrvGPIO_PT3_EnablePullHigh	Enable the pull up of the specified pin
47	DrvGPIO_PT3_DisablePullHigh	Disable the pull up of the specified pin
48	DrvGPIO_PT3_EnableOUTPUT	Enable the output mode of the specified pin
49	DrvGPIO_PT3_DisableOUTPUT	Disable the output mode of the specified pin
50	DrvGPIO_PT3_GetPortBits	Get the input port value from the specified pin
51	DrvGPIO_PT3_SetPortBits	Set the output port value of the specified pin
52	DrvGPIO_PT3_ClrPortBits	Clear the output port value of the specified pin

5.2. Type Definition

E_DRVGPIO_PORT

Enumeration Identifier	Value	Description
E_PT0	0	Define GPIO Port 0
E_PT1	1	Define GPIO Port 1
E_PT2	2	Define GPIO Port 2
E_PT3	3	Define GPIO Port 3
E_PT4	4	Define GPIO Port 4

E_DRVGPIO_IO

Enumeration Identifier	Value	Description
E_IO_INPIT	0	Set GPIO as Input mode
E_IO_OUTPUT	1	Set GPIO as Output mode
E_IO_PullHigh	2	Pull High Enable
E_IO_ IntEnable	3	Interrupt Enable

E_DRVGPIO_IntTriMethod

Enumeration Identifier	Value	Description
E_DisableGPIOInt	0	Disable GPIO Interrupt
E_P_Edge	1	Positive Edge
E_N_Edge	2	Negative Edge
E_Chang_Level	3	Chang Level
E_LLTri	4	Level Low Trigger
E_LHTri	5	Level High Trigger
E_LLTri	6	Level Low Trigger
E_LHTri	7	Level High Trigger

E_DRVGPIO_CLOCK_SOURCE

Enumeration Identifier	Value	Description
E_HS_CK	0	Set IO sampling clock input source is HS_CK
E_LS_CK	1	Set IO sampling clock input source is LS_CK

5.3. Functions

5.3.1. DrvGPIO_Open

- **Prototype**

  int32_t DrvGPIO_Open (E_DRVGPIO_PORT port, int32_t i32Bit, E_DRVGPIO_IO mode)

- **Description**

  Set the specified GPIO(PT1~PT3) pin to the specified GPIO operation mode.

  Configure the register

  PT1 : 0x40800[23:16] / 0x40800[7:0] / 0x40804[23:16] / 0x40010[23:16]

  PT2 : 0x40810[23:16] / 0x40810[7:0] / 0x40814[23:16] / 0x40014[23:16]

  PT3 : 0x40820[23:16] / 0x40820[7:0] / 0x40824[23:16] / 0x40010[23:16]

- **Parameter**

  port [in] : specify GPIO port, the effectively input range is 1~3

  1 : PT1 2 : PT2

  3 : PT3 4 : Rsv

  i32Bit [in] : Specify pin of the GPIO port. It could be 0~0xFF.

  The operation mode of the pin will be set if the bit of i32Bit is equal to 1

  The operation mode of the pin will not be change if the bit of i32Bit is equal to 0

  mode [in] : set the operation mode of the specified GPIO pin

  0: Enable input mode 1: Enable output mode

  2: Enable pull up internally 3: Enable external interrupt

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* configure PT2.0 to GPIO output mode and PT2.1 to GPIO input mode*/
DrvGPIO_Open(E_PT2, 0x01, E_IO_OUTPUT);    //set the operation mode of PT2.0
DrvGPIO_Open(E_PT2, 0x02, E_IO_INPUT);    //set the operation mode of PT2.1
```

5.3.2. DrvGPIO_SetBit

- **Prototype**

  unsigned int DrvGPIO_SetBit (E_DRVGPIO_PORT uport, unsigned int i32Bit)

- **Description**

  Set the output status value of the specified GPIO(PT1~PT3) pad to 1.

Configure the register 0x40804[7:0]/0x40814[7:0]/0x40824[7:0]

- **Parameter**

uport [in] : specify GPIO port, the effectively input range is 1~3.

1 : PT1 2 : PT2

3 : PT3 4 : Rsv

i32Bit [in] : Specify pin of the GPIO port. It could be 0~7.

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* configure PT2.0 as GPIO output mode*/
DrvGPIO_Open(E_PT2, 1, E_IO_OUTPUT);
/* Set PT2.0 to 1(high) */
DrvGPIO_SetBit(E_PT2,0);
```

5.3.3. DrvGPIO_ClrBit

- **Prototype**

unsigned int DrvGPIO_ClrBit (E_DRVGPIO_PORT uport, unsigned int i32Bit)

- **Description**

Clear the output status value of the specified GPIO(PT1~PT3) port.

Clear the register 0x40804[7:0] / 0x40814[7:0] / 0x40824[7:0]

- **Parameter**

uport [in] : specify GPIO port, the effectively input range is 1~3.

1 : PT1 2 : PT2

3 : PT3 4 : Rsv

i32Bit [in] : Specify pin of the GPIO port. It could be 0~7.

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

0: Operation successful

0xff000000 : Incorrect argument

- **Example**

```
/* Set PT2.0 output 0(low) */
DrvGPIO_ClrBit(E_PT2, 0);
```

5.3.4. DrvGPIO_GetBit

- **Prototype**

  uint8_t DrvGPIO_GetBit (E_DRVGPIO_PORT port, uint8_t u32Bit)

- **Description**

  Get the pin value from the specified input GPIO(PT1~PT3) port.

  Read the register 0x40808[7:0]/0x40818[7:0]/0x40828[7:0]

- **Parameter**

  uport [in] : specify GPIO port, the effectively input range is 1~3.

  1：PT1　　2：PT2

  3：PT3　　4：Rsv

  i32Bit [in] : Specify pin of the GPIO port. The input range is 0~7.

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  0 / 1:The specified input pin value:

  0xff000000 : Incorrect argument,

- **Example**

```
uint32_t i32BitValue;
/* Configure PT2.1 as GPIO input mode, and read the input value of status*/
DrvGPIO_Open(E_PT2, 0x02, E_IO_INPUT);
DrvGPIO_Open(E_PT2, 0x02, E_IO_PullHigh);
i32Bit = DrvGPIO_GetBit(E_PT2,1);    // Read 0x40818[1]
```

5.3.5. DrvGPIO_SetPortBits

- **Prototype**

  unsigned int DrvGPIO_SetPortBits (E_DRVGPIO_PORT uport, unsigned int ui32Data)

- **Description**

  Set the output port value to the specified GPIO(PT1~PT3) port.

  Configure the register 0x40804[7:0]/0x40814[7:0]/0x40824[7:0]

- **Parameter**

  uport [in] : specify GPIO port, the effectively input range is 1~3.

  1：PT1　　2：PT2

  3：PT3　　4：Rsv

  i32Data [in] : specify which bit to be set, the input range is 0x00~0xFF

  The bit will be set 1 if the bit of the i32Data is equal to 1, the bit will be set 0 if the bit of the i32Data is equal to 0.

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Set PT2.1 and PT2.4 to 1*/
DrvGPIO_SetPortBits(E_PT2, 0x12);    //set 0x40814[1][4]
```

5.3.6. DrvGPIO_ClrPortBits

- **Prototype**

  unsigned int DrvGPIO_ClrPortBits (E_DRVGPIO_PORT uport, unsigned int ui32Data)

- **Description**

  Clear the output port value to the specified GPIO(PT1~PT3) port

  Clear the register 0x40804[7:0] / 0x40814[7:0] / 0x40824[7:0]

- **Parameter**

  uport [in] : specify GPIO port, the effectively input range is 1~3

  1 : PT1 2 : PT2

  3 : PT3 4 : Rsv

  i32Data [in] : specify which bit to be set, the input range is 0x00~0xFF

  The bit will change to 0 if the bit of i32Data is equal to 1

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Clear the output value of PT2.1/PT2.4 to 0*/
DrvGPIO_ClrPortBits(E_PT2, 0x12);    //Clear 0x40814[1][4]
```

5.3.7. DrvGPIO_GetPortBits

- **Prototype**

  uint32_t DrvGPIO_GetPortBits (E_DRVGPIO_PORT port)

- **Description**

  Get the input port value from the specified GPIO(PT1~PT3) port.

  Read the register 0x40808[7 :0] / 0x40818[7 :0] / 0x40828[7 :0]

- **Parameter**

port [in] : specify GPIO port, the effectively input range is 1~3

1 : PT1 2 : PT2

3 : PT3 4 : Rsv

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

0 ~ 0xFF :The specified input port value

0xff000000: Incorrect argument ,

- **Example**

```
uint32_t    i32Port;
i32Port = DrvGPIO_GetPortBits(E_PT2);    //Read 0x40818[7:0]
i32Port = DrvGPIO_GetPortBits(E_PT3);    // Read 0x40828[7:0]
```

5.3.8. DrvGPIO_IntTrigger

- **Prototype**

int32_t DrvGPIO_IntTrigger (E_DRVGPIO_PORT port, uint32_t u32Bit, E_DRVGPIO_TriMethod mode)

- **Description**

Set the specified interrupt pin to the specified interrupt trigger method operation mode.

Configure the register 0x4080C[31:0]/0x4081C[31:0]

- **Parameter**

port [in] : Specify GPIO port, the effectively input range is 1~2

1 : PT1 2 : PT2

u32Bit [in] : Specify pin of the GPIO port.

The bit will be set if the bit of the u32Bit is equal to 1. It could be 0~255.

mode [in] : set the specified interrupt method

0: disable the IO external interrupt trigger 1:rising-edge trigger

2: falling-edge trigger 3: level change trigger

4: level low trigger 5:level high trigger

6: level low trigger 7:level high trigger

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Configure PT2.0 to GPIO Interrupt mode ,trigger method is negative edge*/
DrvGPIO_ClkGenerator(0,1); //set IO sample frequency
DrvGPIO_Open(E_PT2, 0x01, E_IO_ IntEnable); //enable PT2 external interrupt. PT2.0.
```

DrvGPIO_IntTrigger(E_PT2, 0x01, E_N_Edge); //set PT2.0 interrupt trigger method. PT2.0.

5.3.9. DrvGPIO_ClkGenerator

- **Prototype**

uint32_t DrvGPIO_ClkGenerator (E_DRVGPIO_CLK_SOURCE uClk, uint32_t uDivider)

- **Description**

Set IO sampling clock input source and divider.

Configure the register 0x4030C[20:16]

- **Parameter**

uClk [in] : specify GPIO sampling clock source, , the effectively input range is 0~1

0 : （HS_CK）

1 : （LS_CK）

uDivider [in] : Specify I/O Port sampling clock source divider. It could be 0~15.

0: off	1: ÷1
2: ÷2	3: ÷4
4: ÷8	5: ÷16
6: ÷32	7: ÷64
8: ÷128	9: ÷256
10: ÷512	11: ÷1024
12: ÷2048	13: ÷4096
14: ÷8192	15: ÷16384

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Specify I/O Port sampling clock source is HS_CK and clk/2 .*/
DrvGPIO_ClkGenerator (E_HS_CK, 2); //0x4030C[20]=0b,[19:16]=0010
```

5.3.10. DrvGPIO_ClearIntFlag

- **Prototype**

unsigned int DrvGPIO_ClearIntFlag (E_DRVGPIO_PORT port, uint32_t u32Bit)

- **Description**

Clear external interrupt flag.

Clear the register 0x40010[7 :0] / 0x40014[7 :0]

- **Parameter**

port [in] : Specify GPIO port, , the effectively input range is 1~2

1：PT1　　2：PT2

u32Bit [in] : Specify pin of the GPIO port. It could be 0~0xFF

The corresponding bit of register will be clear if the bit of the u32Bit is equal to 1.

- **include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

The current state of GPIO external interrupt 4flags

- **Example**

```
/* Clear PT2.2 interrupt flag */
DrvGPIO_ClearIntFlag(E_PT2, 0x04);    //0x40014[3]=0b
/* Clear PT2.3 interrupt flag*/
DrvGPIO_ClearIntFlag(E_PT2, 0x08);    //0x40014[7]=0b
```

5.3.11. DrvGPIO_GetIntFlag

- **Prototype**

unsigned int DrvGPIO_GetIntFlag(E_DRVGPIO_PORT port)

- **Description**

Get the port value from the specified Interrupt Trigger Source Indicator Register. If the corresponding bit of the return port value is 1, it is meaning the interrupt occurred at the corresponding bit. Otherwise, no interrupt occurred at that bit.

Read the register 0x40010[7:0] / 0x40014[7:0]

- **Parameter**

port [in] : Specify GPIO port, , the effectively input range is 1~2

1：PT1　　2：PT2

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

The port value of the specified register: 0 ~ 0Xff

- **Example**

```
/* Get PT1 interrupt status. */
unsigned char    flag ; flag=DrvGPIO_GetIntFlag(E_PT1);
```

5.3.12. DrvGPIO_Close

- **Prototype**

int32_t DrvGPIO_Close (E_DRVGPIO_PORT port, int32_t i32Bit, E_DRVGPIO_IO mode)

- **Description**

Close the specified operation mode of the specified GPIO pin

Configure the register

PT1 0x40800[23:16] / 0x40800[7:0] / 0x40804[23:16] / 0x40010[23:16]

PT2 0x40810[23:16] / 0x40810[7:0] / 0x40814[23:16] / 0x40014[23:16]

PT3 0x40820[23:16] / 0x40820[7:0] / 0x40824[23:16]

- **Parameter**

port [in] : Specify GPIO port, , the effectively input range is 1~3

1：PT1 2：PT2

3：PT3 4：Rsv

i32Bit [in] : Specify pin of the GPIO port. It could be 0~0xFF.

The operation mode of the pin will be close if the bit of i32Bit is equal to 1

The operation mode of the pin will not be change if the bit of i32Bit is equal to 0

mode [in] : set the operation mode of the specified GPIO pin

0: input mode 1: output mode

2: pull up internally 3: external interrupt

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

```
DrvGPIO_Close(E_PT2, 0x01, E_IO_OUTPUT); //close the specified operation mode of PT2.0
DrvGPIO_Close(E_PT2, 0x02, E_IO_INPUT); //close the specified operation mode of PT2.1
```

5.3.13. DrvGPIO_EnableAnalogPin

- **Prototype**

unsigned char DrvGPIO_EnableAnalogPin(short port,unsigned int i32Bit)

- **Description**

Close the digital operation mode of the specified GPIO pin, it could be input/output/external interrupt/pull-up/interrupt trigger edge and open analog operation mode

Configure the register

PT1 0x40800[23:16] / 0x40800[7:0] / 0x40804[23:16] /0x4080C[23:0]/ 0x40010[23:16]

PT2 0x40810[23:16] / 0x40810[7:0] / 0x40814[23:16] /0x4081C[23:0]/ 0x40014[23:16]

PT3 0x40820[23:16] / 0x40820[7:0] / 0x40824[23:16]

- **Parameter**

port [in] : specify GPIO port, the effectively input range is 1~3

1：PT1 2：PT2

3：PT3

i32Bit [in] : Specify pin of the GPIO port. It could be 0~0xFF.

The operation mode of the pin will be change if the bit of i32Bit is equal to 1

The operation mode of the pin will not be change if the bit of i32Bit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

0: Operation successful

1 : Incorrect argument

- **Example**

```
/* Close PT3.1/PT3.3/PT3.5/PT3.7 digital operation mode*/
DrvGPIO_Open(E_PT3,0xAA,E_IO_INPUT);
DrvGPIO_Open(E_PT3,0x55,E_IO_OUTPUT);
DrvGPIO_Open(E_PT3,0xAA,E_IO_PullHigh);
DrvGPIO_IntTrigger(E_PT3,0xAA,E_N_Edge);
DrvGPIO_EnableAnalogPin(E_PT3,0xAA);
```

5.3.14. DrvGPIO_PT1_EnableINPUT

- **Prototype**

void DrvGPIO_PT1_EnableINPUT(short int ubit)

- **Description**

Enable the input mode of the specified GPIO pin .

Configure the register 0x40804[23:16]

- **Parameter**

ubit[in] : specified PT1 pin. It could be 0~0xff

Set the specified GPIO pin to the input operation mode if the bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* set PT1.0/PT1.1 as input mode*/
DrvGPIO_PT1_EnableINPUT(0x01|0x02);
```

5.3.15. DrvGPIO_PT1_DisableINPUT

- **Prototype**

void DrvGPIO_PT1_DisableINPUT(short int ubit)

- **Description**

Disable the input mode of the specified GPIO pin .

Configure the register 0x40804[23:16]

- **Parameter**

ubit[in] : specified PT1 pin. It could be 0~0xff

Disable the input mode of the specified GPIO pin if the specified bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* disable the input mode of PT1.0/PT1.1*/
DrvGPIO_PT1_DisableINPUT(0x01|0x02);
```

5.3.16. DrvGPIO_PT1_EnablePullHigh

- **Prototype**

void DrvGPIO_PT1_EnablePullHigh (short int ubit)

- **Description**

Enable the pull up of the specified GPIO pin .

Configure the register 0x40800[23:16]

- **Parameter**

ubit[in] : specified PT1 pin. It could be 0~0xff

Set the specified GPIO pin to enable pull-up if the bit of ubit is equal to 1

The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* enable the pull up of PT1.0/PT1.1 */
DrvGPIO_PT1_EnablePullHigh(0x01|0x02);
```

5.3.17. DrvGPIO_PT1_DisablePullHigh

- **Prototype**

void DrvGPIO_PT1_DisablePullHigh (short int ubit)

- **Description**

Disable the pull up of the specified GPIO pin .

Configure the register 0x40800[23:16]

- **Parameter**

  ubit[in] specified PT1 pin. It could be 0~0xff

  Set the specified GPIO pin to disable pull-up if the bit of ubit is equal to 1

  The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  None

- **Example**

```
/* disable the pull up of PT1.0/PT1.1 */
DrvGPIO_PT1_DisablePullHigh(0x01|0x02);
```

5.3.18. DrvGPIO_PT1_EnableOUTPUT

- **Prototype**

  void DrvGPIO_PT1_EnableOUTPUT(short int ubit)

- **Description**

  Enable the output mode of the specified GPIO pin .

  Configure the register 0x40800[7:0]

- **Parameter**

  ubit[in] : specified PT1 pin. It could be 0~0xff

  Set the specified GPIO pin to the output operation mode if the specified bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  None

- **Example**

```
/* set PT1.0/PT1.1 as output mode*/
DrvGPIO_PT1_EnableOUTPUT(0x01|0x02);
```

5.3.19. DrvGPIO_PT1_DisableOUTPUT

- **Prototype**

  void DrvGPIO_PT1_DisableOUTPUT(short int ubit)

- **Description**

  Disable the output mode of the specified GPIO pin .

  Configure the register 0x40800[7:0]

- **Parameter**

ubit[in] : specified PT1 pin. It could be 0~0xff

Disable the output mode of the specified GPIO pin if the specified bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* disable the output mode of PT1.0/PT1.1*/
DrvGPIO_PT1_DisableOUTPUT(0x01|0x02);
```

5.3.20. DrvGPIO_PT1_EnableINT

- **Prototype**

void DrvGPIO_PT1_EnableINT (short int ubit)

- **Description**

Enable the external interrupt of the specified GPIO pin .

Configure the register 0x40010[23:16]

- **Parameter**

ubit[in] : specified PT1 pin. It could be 0~0xff

Set the specified GPIO pin to enable the external interrupt if the bit of ubit is equal to 1

The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* enable the external interrupt of PT1.0/PT1.1 */
DrvGPIO_PT1_EnableINT(0x01|0x02);
```

5.3.21. DrvGPIO_PT1_DisableINT

- **Prototype**

void DrvGPIO_PT1_DisableINT (short int ubit)

- **Description**

Disable the external interrupt of the specified GPIO pin .

Configure the register 0x40010[23:16]

- **Parameter**

ubit[in] : specified PT1 pin. It could be 0~0xff

Set the specified GPIO pin to disable the external interrupt if the bit of ubit is equal to 1

The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* disable the external interrupt of PT1.0/PT1.1 */
DrvGPIO_PT1_DisableINT(0x01|0x02);
```

5.3.22. DrvGPIO_PT1_IntTriggerPorts

- **Prototype**

void DrvGPIO_PT1_IntTriggerPorts(uint32_t i32Bit, uint32_t mode)

- **Description**

Enable the external interrupt trigger of the specified GPIO pin . Select the method of interrupt trigger.

Configure the register 0x4080c[31:0]

- **Parameter**

u32Bit [in] : specified PT1 pin. It could be 0~0xff

Set the specified GPIO if the bit of u32Bit is equal to 1

Disable the interrupt trigger of specified GPIO pin if the bit of u32Bit is equal to 0

mode [in] : interrupt trigger method . it could be 0~7

0：disable GPIO interrupt trigger	1：positive edge	2：negative edge	3：level change
4：low level	5：high level	6：low level	7：high level

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* set the negative edge as the interrupt method of PT1.0*/
DrvGPIO_ClkGenerator(0,1); // set the sampling frequency
DrvGPIO_PT1_EnableINT(0x1); // enable GPIO external interrupt
DrvGPIO_PT1_IntTriggerPorts(0x1, E_N_Edge);    //configure the interrupt trigger method
```

5.3.23. DrvGPIO_PT1_IntTriggerBit

- **Prototype**

void DrvGPIO_PT1_IntTriggerBit(uint32_t i32Bit, uint32_t mode)

- **Description**

Enable the external interrupt trigger of the specified GPIO pin . select the method of interrupt trigger.

Configure the register 0x4080c[31:0]

- **Parameter**

u32Bit [in] : specified PT1 pin. It could be 0~7 stand for bit7~bit0 of GPIO port

The specified GPIO pin will be set.

mode [in] : interrupt trigger method . it could be 0~7

0：disable GPIO interrupt trigger	1：positive edge	2：negative edge	3：level change
4：low level	5：high level	6：low level	7：high level

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* set the negative edge as the interrupt method of PT1.0*/
DrvGPIO_ClkGenerator(0,1); // set the sampling frequency
DrvGPIO_PT1_EnableINT(0x1);// enable GPIO external interrupt
DrvGPIO_PT1_IntTriggerPorts(0x1, E_N_Edge); //configure the interrupt trigger method
```

5.3.24. DrvGPIO_PT1_GetIntFlag

- **Prototype**

unsigned char DrvGPIO_PT1_GetIntFlag(void)

- **Description**

Get the port value from the PT1 Interrupt Trigger Source Indicator Register. If the corresponding bit of the return port value is 1, it is meaning the interrupt occurred at the corresponding bit. Otherwise, no interrupt occurred at that bit.

Read the register 0x40010[7 :0]

- **Parameter**

None

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

The interrupt flag value of the specified register: 0 ~ 0xff

- **Example**

```
/* Get PT1 interrupt flag. */
unsigned char   flag ; flag=DrvGPIO_PT1_GetIntFlag();
```

5.3.25. DrvGPIO_PT1_ClearIntFlag

- **Prototype**

  unsigned char DrvGPIO_PT1_ClearIntFlag(short int uint32)

- **Description**

  Clear the external interrupt flag of PT1

  Configure the register 0X40010[7 :0]

- **Parameter**

  u32Bit [in] : specified PT1 pin. It could be 0~0xff

  The each bit of u32Bit is corresponding to one pin .

  Clear the corresponding interrupt flag when the specified bit of u32Bit is equal to 1

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  None

- **Example**

  /* Clear PT1.2 interrupt flag */

  DrvGPIO_PT1_ClearIntFlag(0x04);

  /* Clear PT1.3 interrupt flag*/

  DrvGPIO_PT1_ClearIntFlag(0x08);

5.3.26. DrvGPIO_PT1_GetPortBits

- **Prototype**

  uint32_t DrvGPIO_PT1_GetPortBits (void)

- **Description**

  Get the input port data from the specified GPIO port.

  Read the register 0x40808[7:0]

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  0 ~ 0xFF :The input value of specified port

- **Example**

  /* Get the PT1 port input data value */

  uint32_t i32Port; i32Port = DrvGPIO_PT1_GetPortBits();

5.3.27. DrvGPIO_PT1_SetPortBits

- **Prototype**

void DrvGPIO_PT1_SetPortBits (unsigned char ui32Data)

- **Description**

Set the output port value to the specified pin.

Configure the register 0x40804[7:0]

- **Parameter**

i32Data [in] : specify which bit to be set. It could be 0~0xFF

The each bit of i32Data corresponding to one pin .

Output data of the specified GPIO pin will be set 1 when the bit of u32Bit is equal to 1, the bit will be set 0 if the bit of the i32Data is equal to 0.

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* Set PT1.2, PT1.4 */
DrvGPIO_PT1_SetPortBits(0x14);
```

5.3.28. DrvGPIO_PT1_ClrPortBits

- **Prototype**

void DrvGPIO_PT1_ClrPortBits (unsigned int ui32Data)

- **Description**

Clear the output data of the specified pin.

Configure the register 0x40804[7:0]

- **Parameter**

i32Data [in] : specify which bit to be clear. It could be 0~0xFF

The each bit of i32Data corresponding to one pin .

Output data of the specified GPIO pin will be clear when the corresponding bit of u32Bit is equal to 1

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* clear PT1.1    PT1.4 */
DrvGPIO_PT1_ClrPortBits(0x12);
```

5.3.29. DrvGPIO_PT2_EnableINPUT

- **Prototype**

void DrvGPIO_PT2_EnableINPUT(short int ubit)

- **Description**

Enable the input mode of the specified GPIO pin .

Configure the register 0x40814[23:16]

- **Parameter**

ubit[in] : specified PT2 pin. It could be 0~0xff

Set the specified GPIO pin to the input operation mode if the bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* set PT2.0/PT2.1 as input mode*/
DrvGPIO_PT2_EnableINPUT(0x01|0x02);
```

5.3.30. DrvGPIO_PT2_DisableINPUT

- **Prototype**

void DrvGPIO_PT2_DisableINPUT(short int ubit)

- **Description**

Disable the input mode of the specified GPIO pin .

Configure the register 0x40814[23:16]

- **Parameter**

ubit[in] : specified PT2 pin. It could be 0~0xff

Disable the input mode of the specified GPIO pin if the bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* disable the input mode of PT2.0/PT2.1*/
DrvGPIO_PT2_DisableINPUT(0x01|0x02);
```

5.3.31. DrvGPIO_PT2_EnablePullHigh

- **Prototype**

void DrvGPIO_PT2_EnablePullHigh (short int ubit)

- **Description**

Enable the pull up of the specified GPIO pin .

Configure the register 0x40810[23:16]

- **Parameter**

ubit[in] : specified PT2 pin. It could be 0~0xff

Set the specified GPIO pin to enable pull-up if the bit of ubit is equal to 1

The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* enable the pull up of PT2.0/PT2.1 */
DrvGPIO_PT2_EnablePullHigh(0x01|0x02);
```

5.3.32. DrvGPIO_PT2_DisablePullHigh

- **Prototype**

void DrvGPIO_PT2_DisablePullHigh (short int ubit)

- **Description**

Disable the pull up of the specified GPIO pin .

Configure the register 0x40810[23:16]

- **Parameter**

ubit[in] : specified PT2 pin. It could be 0~0xff

Set the specified GPIO pin to disable pull-up if the bit of ubit is equal to 1

The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* disable the pull up of PT2.0/PT2.1 */
DrvGPIO_PT2_DisablePullHigh(0x01|0x02);
```

5.3.33. DrvGPIO_PT2_EnableOUTPUT

- **Prototype**

void DrvGPIO_PT2_EnableOUTPUT(short int ubit)

- **Description**

Enable the output mode of the specified GPIO pin .

Configure the register 0x40810[7:0]

- **Parameter**

ubit[in] : specified PT2 pin. It could be 0~0xff

Set the specified GPIO pin to the output operation mode if the bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* set PT2.0/PT2.1 as output mode*/
DrvGPIO_PT2_EnableOUTPUT(0x01|0x02);
```

5.3.34. DrvGPIO_PT2_DisableOUTPUT

- **Prototype**

void DrvGPIO_PT2_DisableOUTPUT(short int ubit)

- **Description**

Disable the output mode of the specified GPIO pin .

Configure the register 0x40810[7:0]

- **Parameter**

ubit[in] : specified PT2 pin. It could be 0~0xff

Disable the output mode of the specified GPIO pin if the bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* disable the output mode of PT2.0/PT2.1*/
DrvGPIO_PT2_DisableOUTPUT(0x01|0x02);
```

5.3.35. DrvGPIO_PT2_EnableINT

- **Prototype**

void DrvGPIO_PT2_EnableINT (short int ubit)

- **Description**

Enable the external interrupt of the specified GPIO pin .

Configure the register 0x40014[23:16]

- **Parameter**

ubit[in] : specified PT2 pin. It could be 0~0xff

Set the specified GPIO pin to enable the external interrupt if the bit of ubit is equal to 1

The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* enable the external interrupt of PT2.0/PT2.1 */
DrvGPIO_PT2_EnableINT(0x01|0x02);
```

5.3.36. DrvGPIO_PT2_DisableINT

- **Prototype**

void DrvGPIO_PT2_DisableINT (short int ubit)

- **Description**

Disable the external interrupt of the specified GPIO pin .

Configure the register 0x40014[23:16]

- **Parameter**

ubit[in] : specified PT2 pin. It could be 0~0xff

Set the specified GPIO pin to disable the external interrupt if the bit of ubit is equal to 1

The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* disable the external interrupt of PT2.0/PT2.1 */
DrvGPIO_PT2_DisableINT(0x01|0x02);
```

5.3.37. DrvGPIO_PT2_IntTriggerPorts

- **Prototype**

  void DrvGPIO_PT2_IntTriggerPorts(uint32_t i32Bit, uint32_t mode)

- **Description**

  Enable the external interrupt trigger of the specified GPIO pin . select the method of interrupt trigger.

  Configure the register 0x4081C[31:0]

- **Parameter**

  u32Bit [in] : specified PT2 pin. It could be 0~0xff

  Set the specified GPIO if the bit of u32Bit is equal to 1

  Disable the interrupt trigger of specified GPIO pin if the bit of u32Bit is equal to 0

  mode [in] : interrupt trigger method . it could be 0~7

  0：disable GPIO interrupt trigger　1：rising-edge　2：falling-edge　3：level change

  4：low level　5：high level　6：low level　7：high level

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  None

- **Example**

```
/* set the falling edge as the interrupt method of PT2.0*/
DrvGPIO_ClkGenerator(0,1); // set the sampling frequency
DrvGPIO_PT2_EnableINT(0x1); // enable GPIO external interrupt
DrvGPIO_PT2_IntTriggerPorts(0x1, E_N_Edge); //configure the interrupt trigger method
```

5.3.38. DrvGPIO_PT2_IntTriggerBit

- **Prototype**

  void DrvGPIO_PT2_IntTriggerBit(uint32_t i32Bit, uint32_t mode)

- **Description**

  Enable the external interrupt trigger of the specified GPIO pin . select the method of interrupt trigger.

  Configure the register 0x4081C[31:0]

- **Parameter**

  u32Bit [in] : specified PT2 pin. It could be 0~7 stand for bit7~bit0 of GPIO port

  The specified GPIO pin will be set.

  mode [in] : interrupt trigger method . it could be 0~7

  0：disable GPIO interrupt trigger　1：rising-edge　2：falling-edge　3：level change

  4：low level　5：high level　6：low level　7：high level

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  None

- **Example**

```
/* set the falling-edge as the interrupt method of PT2.0*/
DrvGPIO_ClkGenerator(0,1); // set the sampling frequency
DrvGPIO_PT2_EnableINT(0x1); // enable GPIO external interrupt
DrvGPIO_PT2_IntTriggerBit(0x1, E_N_Edge); //configure the interrupt trigger method
```

5.3.39. DrvGPIO_PT2_GetIntFlag

- **Prototype**

unsigned char DrvGPIO_PT2_GetIntFlag(void)

- **Description**

Get the port value from the PT2 Interrupt Trigger Source Indicator Register. If the corresponding bit of the return port value is 1, it is meaning the interrupt occurred at the corresponding bit. Otherwise, no interrupt occurred at that bit.

Read the register 0x40014[7:0]

- **Parameter**

None

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

The interrupt flag value of the specified register: 0 ~ 0xff

- **Example**

```
/* Get PT2 interrupt flag. */
unsigned char flag ; flag=DrvGPIO_PT2_GetIntFlag();
```

5.3.40. DrvGPIO_PT2_ClearIntFlag

- **Prototype**

unsigned char DrvGPIO_PT2_ClearIntFlag(short int uint32)

- **Description**

Clear the external interrupt flag of PT2

Configure the register 0x40014[7 :0]

- **Parameter**

u32Bit [in] : specified PT2 pin. It could be 0~0xff

The each bit of u32Bit corresponding to one pin .

Clear the corresponding interrupt flag when the specified bit of u32Bit is equal to 1

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Value**

None

- **Example**

```
/* Clear PT2.2 interrupt flag */
DrvGPIO_PT2_ClearIntFlag(0x04);
/* Clear PT2.3 interrupt flag*/
DrvGPIO_PT2_ClearIntFlag(0x08);
```

5.3.41. DrvGPIO_PT2_GetPortBits

- **Prototype**

  unsigned char DrvGPIO_PT2_GetPortBits (void)

- **Description**

  Get the input port value from the specified GPIO port.

  Read the register 0x40818[7:0]

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  0 ~ 0xFF :The input value of specified port

- **Example**

```
/* Get the PT2 port input data value */
uint32_t i32Port; i32Port = DrvGPIO_PT2_GetPortBits();
```

5.3.42. DrvGPIO_PT2_SetPortBits

- **Prototype**

  void DrvGPIO_PT2_SetPortBits (unsigned char ui32Data)

- **Description**

  Set the output port value to the specified pin.

  Read the register 0x40814[7:0]

- **Parameter**

  i32Data [in] : specify which bit to be set. It could be 0~0xFF

  The each bit of i32Data corresponding to one pin .

  Output data of the specified GPIO pin will be set 1 when the bit of u32Bit is equal to 1, the bit will be set 0 if the bit of the i32Data is equal to 0.

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  None

- **Example**

```
/* Set PT2.2, PT2.4 to 1(high) */
DrvGPIO_PT2_SetPortBits(0x14);
```

5.3.43. DrvGPIO_PT2_ClrPortBits

- **Prototype**

  void DrvGPIO_PT2_ClrPortBits (unsigned int ui32Data)

- **Description**

  Clear the output data of the specified pin.

  Read the register 0x40814[7:0]

- **Parameter**

  i32Data [in] : specify which bit to be clear. It could be 0~0xFF

  The each bit of i32Data corresponding to one pin .

  Output data of the specified GPIO pin will be clear when the corresponding bit of u32Bit is equal to 1

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  None

- **Example**

```
/* clear PT2.1, PT2.4 */
DrvGPIO_PT2_ClrPortBits(0x12);
```

5.3.44. DrvGPIO_PT3_EnableINPUT

- **Prototype**

  void DrvGPIO_PT3_EnableINPUT(short int ubit)

- **Description**

  Enable the input mode of the specified GPIO pin .

  Configure the register 0x40824[23:16]

- **Parameter**

  ubit[in] : specified PT3 pin. It could be 0~0xff

  Set the specified GPIO pin to the input operation mode if the bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  None

- **Example**

```
/* set PT3.0/PT3.1 as input mode*/
DrvGPIO_PT3_EnableINPUT(0x01|0x02);
```

5.3.45. DrvGPIO_PT3_DisableINPUT

- **Prototype**

  void DrvGPIO_PT3_DisableINPUT(short int ubit)

- **Description**

  Disable the input mode of the specified GPIO pin .

  Configure the register 0x40824[23:16]

- **Parameter**

  ubit[in] : specified PT3 pin. It could be 0~0xff

  Disable the input mode of the specified GPIO pin if the specified bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  None

- **Example**

```
/* disable the input mode of PT3.0/PT3.1*/
DrvGPIO_PT3_DisableINPUT(0x01|0x02);
```

5.3.46. DrvGPIO_PT3_EnablePullHigh

- **Prototype**

  void DrvGPIO_PT3_EnablePullHigh (short int ubit)

- **Description**

  Enable the pull up of the specified GPIO pin .

  Configure the register 0x40820[23:16]

- **Parameter**

  ubit[in] : specified PT3 pin. It could be 0~0xff

  Set the specified GPIO pin to enable pull-up if the bit of ubit is equal to 1

  The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  None

- **Example**

```
/* enable the pull up of PT3.0/PT3.1 */
```

DrvGPIO_PT3_EnablePullHigh(0x01|0x02);

5.3.47. DrvGPIO_PT3_DisablePullHigh

- **Prototype**

  void DrvGPIO_PT3_DisablePullHigh (short int ubit)

- **Description**

  Disable the pull up of the specified GPIO pin .

  Configure the register 0x40820[23:16]

- **Parameter**

  ubit[in] : specified PT3 pin. It could be 0~0xff

  Set the specified GPIO pin to disable pull-up if the bit of ubit is equal to 1

  The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  None

- **Example**

  /* disable the pull up of PT3.0/PT3.1 */

  DrvGPIO_PT3_DisablePullHigh(0x01|0x02);

5.3.48. DrvGPIO_PT3_EnableOUTPUT

- **Prototype**

  void DrvGPIO_PT3_EnableOUTPUT(short int ubit)

- **Description**

  Enable the output mode of the specified GPIO pin .

  Configure the register 0x40820[7:0]

- **Parameter**

  ubit[in] : specified PT3 pin. It could be 0~0xff

  Set the specified GPIO pin to the output operation mode if the specified bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  None

- **Example**

  /* set PT3.0/PT3.1 as output mode*/

  DrvGPIO_PT3_EnableOUTPUT(0x01|0x02);

5.3.49. DrvGPIO_PT3_DisableOUTPUT

- **Prototype**

  void DrvGPIO_PT3_DisableOUTPUT(short int ubit)

- **Description**

  Disable the output mode of the specified GPIO pin .

  Configure the register 0x40820[7:0]

- **Parameter**

  ubit[in] : specified PT3 pin. It could be 0~0xff

  Disable the output mode of the specified GPIO pin if the specified bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  None

- **Example**

  /* disable the output mode of PT3.0/PT3.1*/

  DrvGPIO_PT3_DisableOUTPUT(0x01|0x02);

5.3.50. DrvGPIO_PT3_GetPortBits

- **Prototype**

  uint32_t DrvGPIO_PT3_GetPortBits (void)

- **Description**

  Get the input port data from the specified GPIO port.

  Read the register 0x40828[7:0]

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  0 ~ 0xFF :The input value of specified port

- **Example**

  /* Get the PT3 port input data value */

  uint32_t i32Port; i32Port = DrvGPIO_PT3_GetPortBits();

5.3.51. DrvGPIO_PT3_SetPortBits

- **Prototype**

  void DrvGPIO_PT3_SetPortBits (unsigned char ui32Data)

- **Description**

  Set the output port value to the specified pin.

  Read the register 0x40824[7:0]

- **Parameter**

  i32Data [in] specify which bit to be set. It could be 0~0xFF

  The each bit of i32Data corresponding to one pin .

  Output data of the specified GPIO pin will be set 1 when the bit of u32Bit is equal to 1, the bit will be set 0 if the bit of the i32Data is equal to 0.

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  None

- **Example**

```
/* Set PT3.2, PT3.4 as 1*/
DrvGPIO_PT3_SetPortBits(0x14);
```

5.3.52. DrvGPIO_PT3_ClrPortBits

- **Prototype**

  void DrvGPIO_PT3_ClrPortBits (unsigned int ui32Data)

- **Description**

  Clear the output data of the specified pin.

  Configure the register 0x40824[7:0]

- **Parameter**

  i32Data [in] : specify which bit to be clear. It could be 0~0xFF

  The each bit of i32Data corresponding to one pin .

  Output data of the specified GPIO pin will be clear when the corresponding bit of u32Bit is equal to 1

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Value**

  None

- **Example**

```
/* clear PT3.1 , PT3.4 as 0 */
DrvGPIO_PT3_ClrPortBits(0x12);
```

6. ADC Driver

6.1. Introduction

The following functions are included in ADC Manager Section.

Item	Functions	Description
01	DrvADC_PInputChannel	Positive input source
02	DrvADC_NInputChannel	Negative input source
03	DrvADC_SetADCInputChannel	Set the ADC input mode
04	DrvADC_InputSwitch	ADC input short control
05	DrvADC_RefInputShort	ADC reference short control
06	DrvADC_SetPGA	Input signal gain
07	DrvADC_ADGain	Input signal gain
08	DrvADC_Gain	Input signal gain
09	DrvADC_DCoffset	DC offset input selection
10	DrvADC_RefVoltage	Set the ADC reference
11	DrvADC_FullRefRange	Set the ADC reference
12	DrvADC_OSR	Set the ADC OSR
13	DrvADC_ClkEnable	Enable ADC clock
14	DrvADC_ClkDisable	Disable ADC clock
15	DrvADC_FastChopper	Fast chopper stable mode
16	DrvADC_CombFilter	Comb filter enable control
17	DrvADC_EnableInt	ADC Interrupt Enable
18	DrvADC_DisableInt	ADC Interrupt Disable
19	DrvADC_ReadIntFlag	Read ADC interrupt flag
20	DrvADC_ClearIntFlag	Clear ADC interrupt flag
21	DrvADC_Enable	Enable ADC control
22	DrvADC_Disable	Disable ADC control
23	DrvADC_GetConversionData	Get the A/D conversion data

6.2. Type Definition

E_ADC_INPUT_CHANNEL

Enumeration Identifier	Value	Description
ADC_Input_AIO0	0	Signal input
ADC_Input_AIO1	1	Signal input
ADC_Input_AIO2	2	Signal input
ADC_Input_AIO3	3	Signal input
REFO_I	4	Signal input
OPOI	5	Signal input
TSP0	6	Signal input
TSP1	7	Signal input
DAO	8	Signal input
VDDA_VSSA	9	Signal input

E_ADC_REFV

Enumeration Identifier	Value	Description
External	0	External
Internal	1	Enable buffer and use internal source

E_ADC_PGA & E_ADC_ADGN

Enumeration Identifier	Value	Description
ADC_PGA_Disable	0	Disable PGA
ADC_Gain_8	1	Gain=8
ADC_Gain_16	3	Gain=16
ADC_Gain_32	7	Gain=32
ADC_ADGN_1	0	ADGN=1
ADC_ADGN_2	1	ADGN=2
ADC_ADGN_RESER	2	Reserve
ADC_ADGN_4	3	ADGN=4

E_ADC_SIGNAL_SHORT

Enumeration Identifier	Value	Description
OPEN	0	ADC signal input (positive and negative)open control
SHORT	1	ADC signal input(positive and negative)short control

E_ADC_VRPS_REF_VOLTAGE

Enumeration Identifier	Value	Description
VDDA	0	Reference voltage VDDA
AIO2	1	Reference voltage form AIO2
AIO4	2	Reference voltage form AIO4
REF_BUFFER_OUT	3	Reference voltage form REFO_I

E_ADC_VRNS_REF_VOLTAGE

Enumeration Identifier	Value	Description
VSSA	0	Reference voltage VSSA
AIO3	1	Reference voltage form AIO3
AIO5	2	Reference voltage form AIO5
REF_BUFFER_OUT	3	Reference voltage form REFO_I

6.3. Functions

6.3.1. DrvADC_PInputChannel

- **Prototype**

  unsigned int DrvADC_PInputChannel (E_ADC_INPUT_Channel uINP);

- **Description**

  Set the ADC positive input voltage source.

  Configure the register 0x41104[7:4]

- **Parameters**

  uINP [in] : Specify the input channel. It could be 0~9

0 : AIO0, 1 : AIO1,

2 : AIO2, 3 : AIO3,

4 : REFO_I, 5 : OPOI,

6 : TPSP0, 7 : TPSP1,

8 : DAOI, 9 : VDDA;

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

  /* Select the positive input voltage source form AIO0*/

  DrvADC_PInputChannel(ADC_Input_AIO0);

6.3.2. DrvADC_NInputChannel

- **Prototype**

  unsigned int DrvADC_NInputChannel (E_ADC_INPUT_Channel uINN);

- **Description**

  Set the ADC negative input voltage source.

  Configure the register 0x41104[3:0]

- **Parameters**

  uINN [in] : Specify the input channel. It could be 0~9

  0 : AIO0, 1 : AIO1,

  2 : AIO2, 3 : AIO3,

4：REFO_I,　　5：OPOI,

6：TPSN0,　7：TPSN1,

8：DAOI,　　9：VSSA

- **Include**

Peripheral_lib/DrvADC.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Select the negative input voltage source form AIO1*/
DrvADC_NInputChannel(ADC_Input_AIO1);
```

6.3.3. DrvADC_SetADCInputChannel

- **Prototype**

```
unsigned int DrvADC_SetADCInputChannel (
                    E_ADC_INPUT_Channel uINP,
                    E_ADC_INPUT_Channel uINN
);
```

- **Description**

Set the ADC input source.

Configure the register 0x41104[7:4] / 0X41104[3:0].

- **Parameters**

uINP [in] : Specify the positive input channel. It could be 0~9

0：AIO0,　　1：AIO1,

2：AIO2,　　3：AIO3,

4：REFO_I,　　5：OPOI,

6：TPSP0,　7：TPSP1,

8：　DAOI,　　9：VDDA;

uINN [in] : Specify the negative input channel. It could be 0~9

0：AIO0,　　1：AIO1,

2：AIO2,　　3：AIO3,

4：REFO_I,　　5：OPOI,

6：TPSN0,　　7：TPSN1,

8：DAOI,　　9：VSS。

- **Include**

Peripheral_lib/DrvADC.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

/* the following statement indicates that the external analog input is AIO0 and AIO1 input */

DrvADC_SetADCInputChannel(ADC_Input_AIO0, ADC_Input_AIO1);

6.3.4. DrvADC_InputSwitch

- **Prototype**

unsigned int DrvADC_InputSwitch (uVISHR)

- **Description**

ADC signal input (positive and negative) short control.

Configure the register 0x41100[21]

- **Parameter**

uVISHR[in] : ADC input short switch.

0: OPEN

1: SHORT

- **Include**

Peripheral_lib/DrvADC.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

/* ADC input short */

DrvADC_InputSwitch(1);

6.3.5. DrvADC_RefInputShort

- **Prototype**

unsigned int DrvADC_RefInputShort (E_ADC_SIGNAL_SHORT uVrshr);

- **Description**

Set the ADC reference input (positive and negative) short control.

Configure the register 0x41100[20]

- **Parameters**

uVrshr [in] : ADC reference input short control.

0 : ADC reference input (positive and negative)open control

1 : ADC reference input(positive and negative)short control

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Set the ADC reference input short */
DrvADC_RefInputShort(SHORT);
```

6.3.6. DrvADC_SetPGA

- **Prototype**

  unsigned int DrvADC_SetPGA (E_ADC_PGA uPGA);

- **Description**

  Input signal gain for ADC modulator.

  Configure the register 0x41104[18:16]

- **Parameters**

  uPGA [in] : Specify the ADC PGA.

  0: Gain=1

  1: Gain=8

  2: Reserved

  3: Gain=16

  4: Reserved

  5: Reserved

  6: Reserved

  7: Gain=32

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Set the gain of 8 */
DrvADC_SetPGA(ADC_Gain_8);
```

6.3.7. DrvADC_ADGain

- **Prototype**

unsigned int DrvADC_ADGain (uADgain);

- **Description**

Input signal gain for ADC modulator.

Configure the register 0x41104[21:20]

- **Parameters**

uADgain [in] : Specify the ADC ADGN.

0: Gain=1

1: Gain=2

3: Gain=4

- **Include**

Peripheral_lib/DrvADC.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Set the gain of 2 */
DrvADC_ADGain(1);
```

6.3.8. DrvADC_Gain

- **Prototype**

unsigned int DrvADC_Gain (E_ADC_PGA uPGA ,uADgain);

- **Description**

Input signal gain for modulator.

Configure the register 0x41104[18:16]/ 0x41104[21:20]

- **Parameters**

uPGA [in] : Specify the ADC PGA.

0: Gain=1

1: Gain=8

2: Reserved

3: Gain=16

4: Reserved

5: Reserved

6: Reserved

7: Gain=32

uADgain [in] : Specify the ADC ADGN.

0: Gain=1

1: Gain=2

3: Gain=4

- **Include**

Peripheral_lib/DrvADC.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Set the total gain of 128 */
DrvADC_Gain(7,3);
```

6.3.9. DrvADC_DCoffset

- **Prototype**

unsigned int DrvADC_DCoffset (uDCoffset);

- **Description**

DC offset input voltage selection (VREF=REFP-REFN)

Configure the register 0x41104[27:24]

- **Parameters**

uDCoffice [in] : Specify the ADC DCSET.

0 : 0 VREF

1 : +1/8 VREF

2 : +1/4 VREF

3 : +3/8 VREF

4 : +1/2 VREF

5 : +5/8 VREF

6 : +3/4 VREF

7 : +7/8 VREF

8 : 0 VREF

9 : -1/8 VREF

10 : -1/4 VREF

11 : -3/8 VREF

12 : -1/2 VREF

13 : -5/8 VREF

14 : -3/4 VREF

15 : -7/8 VREF

- **Include**

Peripheral_lib/DrvADC.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

/* Set the DC offset of +1/8 voltage. */

DrvADC_DCoffset(1);

6.3.10. DrvADC_RefVoltage

- **Prototype**

unsigned int DrvADC_RefVoltage (

E_ADC_VRPS_REF_VOLTAGE uVrps,

E_ADC_VRNS_REF_VOLTAGE uVrns

);

- **Description**

Set the ADC reference voltage.

Configure the register 0x41100[19:18] and 0x41100[17:16].

- **Parameters**

uVrps [in] : Specify the ADC VRPS, the input range is 0~3

0 : Reference voltage VDDA

1 : Reference voltage form AIO2

2 : Reference voltage form AIO4

3 : Reference voltage form REFO_I

uVrns [in] : Specify the ADC VRNS, the input range is 0~3

0 : Reference voltage VSSA

1 : Reference voltage form AIO3

2 : Reference voltage form AIO5

3 : Reference voltage form REFO_I

- **Include**

Peripheral_lib/DrvADC.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

/* Set the ADC reference voltage.(VRPS=AIO2, VRNS=AIO3) */

DrvADC_RefVoltage(AIO2, AIO3);

6.3.11. DrvADC_FullRefRange

- **Prototype**

  unsigned int DrvADC_FullRefRange(uFullRange);

- **Description**

  Set the ADC full reference range select.

  Configure the register 0x41104[19]

- **Parameters**

  uFullRange [in] : Specify the VREF gain. VREF= VRPS-VRNS

  0: Full reference range input=VREF*1

  1: 1/2 reference range input=VREF*1/2

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Set the ADC full reference range input. */
DrvADC_FullRefRange(0);
```

6.3.12. DrvADC_OSR

- **Prototype**

  unsigned int DrvADC_OSR (uADCOSR);

- **Description**

  Set the ADC OSR. Configure the register 0x41100[5:2]

- **Parameters**

  uADCOSR [in] : Specify the ADC OSR. (The following output rate is calculated when clock is 327680HZ)

  0 : ÷32768 · Data Output Rate is10sps

  1 : ÷16384 · Data Output Rate is 20sps

  2 : ÷8192 · Data Output Rate is 40sps

  3 : ÷4096 · Data Output Rate is 80sps

  4 : ÷2048 · Data Output Rate is 160sps

  5 : ÷1024 · Data Output Rate is 320sps

  6 : ÷512 · Data Output Rate is 640sps

  7 : ÷256 · Data Output Rate is 1280sps

  8 : ÷128 · Data Output Rate is 2560sps

  9 : ÷64 · Data Output Rate is 5120sps

  10 : ÷32 · Data Output Rate is 10240sps

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Set the OSR of 8192 data rate 40sps. */
DrvADC_OSR(2);
```

6.3.13. DrvADC_ClkEnable

- **Prototype**

  unsigned int DrvADC_ClkEnable(uADCD, uClkPH);

- **Description**

  Enable ADC clock, set the clock divider, the ADC clock phase adjustment

  Configure the register 0x4030C[7:4]

- **Parameters**

  uADCD [in] : Specify the ADC clock divider.

  0 : ÷6

  1 : ÷12

  2 : ÷30

  3 : ÷60

  uClkPH [in] :

  0 : ADC clock rising edge of CPU clock low.

  1 : ADC clock rising edge of CPU clock high.

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Set the ADCD of ÷12 , ADC clock rising edge of CPU clock high. */
DrvADC_ClkEnable(1,1);
```

6.3.14. DrvADC_ClkDisable

- **Prototype**

void DrvADC_ClkDisable(void);

- **Description**

Disable ADC clock.

Configure the register 0x4030C[6]=0

- **Parameters**

None

- **Include**

Peripheral_lib/DrvADC.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Disable ADC clock. */
DrvADC_ClkDisable();
```

6.3.15. DrvADC_FastChopper

- **Prototype**

unsigned int DrvADC_FastChopper(uADFDR);

- **Description**

Fast chopper stable mode control.

Configure the register 0x41100[6]

- **Parameters**

uADFDR [in]: Fast chopper stable mode control.

0 : Normal mode chopper frequency = ADCLK/128

1 : Fast mode chopper frequency = ADCLK/32

- **Include**

Peripheral_lib/DrvADC.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Select the normal mode. */
DrvADC_FastChopper(0);    // frequency = ADCLK/128
```

6.3.16. DrvADC_CombFilter

- **Prototype**

  unsigned int DrvADC_CombFilter(uCFRST);

- **Description**

  Comb filter enable control

  Configure the register 0x41100[1]

- **Parameters**

  uCFRST [in] : Comb filter enable control

  0: Reset

  1: On

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Open the comb filter. */
DrvADC_CombFilter(0);// reset the comb filter
DrvADC_CombFilter(1);//enable the comb filter
```

6.3.17. DrvADC_EnableInt

- **Prototype**

  void DrvADC_EnableInt (void)

- **Description**

  ADC Interrupt Enable

  Configure the register 0x40008[16]=1b

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Value**

  None

- **Example**

```
/* Enable ADC interrupt */
DrvADC_EnableInt();
```

6.3.18. DrvADC_DisableInt

- **Prototype**

  void DrvADC_DisableInt (void)

- **Description**

  ADC Interrupt Disable

  Configure the register 0x40008[16]=0b

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Value**

  None

- **Example**

```
/* Disable ADC interrupt */
DrvADC_DisableInt();
```

6.3.19. DrvADC_ReadIntFlag

- **Prototype**

  unsigned int DrvADC_ReadIntFlag (void)

- **Description**

  Read ADC interrupt flag.

  Read the register 0x40008[0]

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Value**

  0 : Interrupt flag is 0, No interrupt occurred.

  1 : Interrupt flag is 1, interrupt occurred.

  >1: Invalid return value

- **Example**

```
/* Read ADC interrupt flag */
flag=DrvADC_ReadIntFlag();
```

6.3.20. DrvADC_ClearIntFlag

- **Prototype**

  void DrvADC_ClearIntFlag (void)

- **Description**

  Clear ADC interrupt flag.

  Configure the register 0x40008[0]=0

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Value**

  None

- **Example**

```
/* Clear ADC interrupt flag */
DrvADC_ClearIntFlag();
```

6.3.21. DrvADC_Enable

- **Prototype**

  void DrvADC_Enable(void)

- **Description**

  Enable ADC control

  Configure the register 0x41100[0]=1b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Value**

  None.

- **Example**

```
/* Enable ADC */
DrvADC_Enable();
```

6.3.22. DrvADC_Disable

- **Prototype**

  void DrvADC_Disable(void)

- **Description**

  Disable ADC control. Configure the register 0x41100[0]=0b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Value**

  None.

- **Example**

```
/* Disable ADC */
DrvADC_Disable();
```

6.3.23. DrvADC_GetConversionData

- **Prototype**

  unsigned int DrvADC_GetConversionData (void);

- **Description**

  Get the A/D conversion data with signed.

  Configure the register 0x41108[31:0]

- **Parameters**

  None.

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Value**

  Return the conversion data.

- **Example**

```
/* Get the ADC conversion data */
int adc_data ;
adc_data=DrvADC_GetConversionDate();
```

7. SPI32 Driver

7.1. Introduction

The following functions are included in SPI Manager Section.

Item	Functions	Description
01	DrvSPI32_Open	Open SPI module
02	DrvSPI32_Close	Close SPI module
03	DrvSPI32_IsBusy	Check busy status
04	DrvSPI32_SetClockFreq	Configure the frequency of SPI clock
05	DrvSPI32_IsRxBufferFull	Check Rx buffer status
06	DrvSPI32_IsTxBufferFull	Check Tx buffer status
07	DrvSPI32_EnableRxInt	Enable the SPI Rx interrupt
08	DrvSPI32_EnableTxInt	Enable the SPI Tx interrupt
09	DrvSPI32_DisableRxInt	Disable the SPI Rx interrupt
10	DrvSPI32_DisableTxInt	Disable the SPI Tx interrupt
11	DrvSPI32_GetRxIntFlag	Get the SPI32 Rx interrupt flag
12	DrvSPI32_GetTxIntFlag	Get the SPI32 Tx interrupt flag
13	DrvSPI32_ClrIntRxFlag	Clear the SPI Rx interrupt flag
14	DrvSPI32_ClrIntTxFlag	Clear the SPI Tx interrupt flag
15	DrvSPI32_Read	Read data from SPIBUF registers
16	DrvSPI32_Write	Write data to SPIBUF register
17	DrvSPI32_Enable	Enable the SPI
18	DrvSPI32_BitLength	Set the SPI transfer Bit length
19	DrvSPI32_GetDCFlag	Get data loss flag of state
20	DrvSPI32_IsABFlag	Read the flag of received data deficient
21	DrvSPI32_IsOVFlag	Read the flag of SPI Bus Data too long
22	DrvSPI32_IsRxFlag	Read the flag of Rx Buffer Updata
23	DrvSPI32_SetEndian	Set the data transmitted from the MSB or LSB start
24	DrvSPI32_SetCSO	Configure CS Polarity
25	DrvSPI32_DisableIO	Disable the SPI port to transmit
26	DrvSPI32_EnableIO	Enable and specify the SPI port to transmit

7.2. Type Definition

E_DRVSPI_MODE

Enumeration Identifier	Value	Description
E_DRVSPI_MASTER1	0	Master,4wire mode
E_DRVSPI_MASTER2	1	Master,3wire mode
E_DRVSPI_MASTER3	2	Master,TI mode
E_DRVSPI_SLAVE1	3	Slave,4wire mode
E_DRVSPI_SLAVE2	4	Slave,3wire mode
E_DRVSPI_SLAVE3	5	Slave,TI mode

E_DRVSPI_TRANS_TYPE

Enumeration Identifier	Value	Description
E_DRVSPI_TYPE0	0	SPI transfer type 0
E_DRVSPI_TYPE1	1	SPI transfer type 1
E_DRVSPI_TYPE2	2	SPI transfer type 2
E_DRVSPI_TYPE3	3	SPI transfer type 3

E_DRVSPI_ENDIAN

Enumeration Identifier	Value	Description
E_DRVSPI_LSB_FIRST	0	Send LSB first
E_DRVSPI_MSB_FIRST	1	Send MSB first

E_DRVSPI_CS

Enumeration Identifier	Value	Description
E_DRVSPI_CSLow	0	CSO low
E_DRVSPI_CSHigh	1	CSO high

7.3. Functions

7.3.1. DrvSPI32_Open

● Prototype

unsigned int DrvSPI_Open(

E_DRVSPI_MODE uMode,

E_DRVSPI_TRANS_TYPE uType,

uOuputPin,

uClkDiv

);

● Description

This function is used to open SPI module. It decides the SPI to work in master or slave mode, SPI bus timing, specified I / O port. Configure the register

0x4030C[2:0],0x4030C[3]=1b, 0x40844[4]=1b, 0x40844[7:5],0x40F00[3:0],0x40f04[16:17]

uMode : 0x40f00[0]=1b, 0x40f00[1]=xb, 0x40f04[16:17]=0xb. uMode : 0~5

uType : 0x40f00[3:2]=xxb. uType : 0~3

uOuputPin : 0x40844[4]=1b, 0x40844[7:5]=xxxb. uOuputPin : 0~7

uClkDiv : 0x4030C[2:0]=xxxb, 0x4030C[3]=1b. uClkDiv : 0~7

● Parameters

uMode [in] : Specify the operation mode

0 : Work in master mode interface 4-wire.

1 : Work in master mode interface 3-wire.

2 : Work in master mode interface TI mode.

3 : Work in slave mode interface 4-wire.

4 : Work in slave mode interface 3-wire.

5 : Work in slave mode interface TI mode.

uType [in] : Transfer types, i.e. the bus timing. It could be 0~ 3.

0: Latch data on first edge of serial clock, clock idle state is low.(CPHA=0 CPOL=0)

1: Latch data on first edge of serial clock, clock idle state is high.(CPHA=0 CPOL=1)

2: Latch data on second edge of serial clock, clock idle state is low.(CPHA=1 CPOL=0)

3: Latch data on second edge of serial clock, clock idle state is high.(CPHA=1 CPOL=1)

uOuputPin [in] : Specify the trasmission port

0 : Port1.0 =CS, Port1.1 =CK, Port1.2 = DI, Port1.3 =DO

1 : Port1.4 =CS, Port1.5 =CK, Port1.6 = DI, Port1.7 =DO

2 : Port2.0 =CS, Port2.1 =CK, Port2.2 = DI, Port2.3 =DO

3 : Port2.4 =CS, Port2.5 =CK, Port2.6 = DI, Port2.7 =DO

uClkDiv [in] : Specify the clock divider

0 : ÷1

1 : ÷2

2 : ÷4

3 : ÷8

4 : ÷32

5 : ÷128

6 : ÷512

7 : ÷2048

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Configure SPI as a master, SPI transfer type 1, Output Pin to select 1 : Port1.4 =CS, Port1.5 =CK,
Port1.6 = DI, Port1.7 =DO, Set SPI clock/512 */
DrvSPI32_Open(E_DRVSPI_MASTER1, E_DRVSPI_TYPE1,1,6);
```

7.3.2. DrvSPI32_Close

- **Prototype**

  void DrvSPI32_Close (void);

- **Description**

  Disable the SPI clock source divider and SPI function and SPI IO port.

  Configure the register 0x40F00[0]=0, 0x4030C[3]=0,0x40844[4]=0

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  None

- **Example**

```
/* Close the (SPI) */
DrvSPI32_Close();
```

7.3.3. DrvSPI32_IsBusy

- **Prototype**

unsigned int DrvSPI32_IsBusy(void);

- **Description**

Check the busy status of the SPI port.

- **Parameters**

None

- **Include**

Peripheral_lib/DrvSPI32.h

- **Return Value**

1: The SPI port is in busy.

0: The SPI port is not in busy.

- **Example**

```
/* Check the busy status */
unsigned char    flag;
flag=DrvSPI32_IsBusy (); //read 0x40f00[19]
```

7.3.4. DrvSPI32_SetClockFreq

- **Prototype**

unsigned int DrvSPI32_SetClockFreq(unsigned int uCPUDV, unsigned int uTMRDV);

- **Description**

Configure the frequency of SPI clock. In master mode, the output frequency of serial clock is programmable.

Configure the register 0x40308[1], 0x4030C[2:0]

- **Parameters**

eCPUDV [in] : Specify the MCU clock input divider.

0 : ÷1

1 : ÷2

eTMRDV [in] : Specify the SPI clock source divider.

0 : ÷1

1 : ÷2

2 : ÷4

3 : ÷8

4 : ÷32

5 : ÷128

6 : ÷512

7 : ÷2048

- **Include**

Peripheral_lib/DrvSPI32.h

- **Return Value**

None

- **Example**

/* MCU clock is /2, SPI clock rate is APCK/512 */

DrvSPI32_SetClockFreq(1, 6);

7.3.5. DrvSPI32_IsRxBufferFull

- **Prototype**

unsigned int DrvSPI32_IsRxBufferFull(void);

- **Description**

Check Rx buffer status (only for data reception), read the register 0x40F00[16]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvSPI32.h

- **Return Value**

1: Rx buffer is full.

0: Rx buffer is not full.

- **Example**

```
/* Check the status of Rx buffer */
 unsigned char    flag;
 flag = DrvSPI32_IsRxBufferFull() ;
```

7.3.6. DrvSPI32_IsTxBufferFull

- **Prototype**

unsigned int DrvSPI32_IsTxBufferFull(void);

- **Description**

Check Tx buffer status

Configure the register 0x40F00[17]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvSPI32.h

- **Return Value**

1: Tx buffer is full.

0: Tx buffer is not full, Tx buffer is empty.

- **Example**

```
/* Check the status of Tx buffer */
unsigned char    flag;    flag =DrvSPI32_IsTxBufferFull();
```

7.3.7. DrvSPI32_EnableRxInt

- **Prototype**

  void DrvSPI32_EnableRxInt(void);

- **Description**

  Enable the SPI Rx interrupt.

  Configure the register 0x40000[16]=1b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  None

- **Example**

```
/* Enable the SPI Rx interrupt */
DrvSPI32_EnableRxInt();
```

7.3.8. DrvSPI32_EnableTxInt

- **Prototype**

  void DrvSPI32_EnableTxInt(void);

- **Description**

  Enable the SPI Tx interrupt.

  Configure the register 0x40000[17]=1b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  None

- **Example**

```
/* Enable the SPI Tx interrupt */
DrvSPI32_EnableTxInt();
```

7.3.9. DrvSPI32_DisableRxInt

- **Prototype**

  void DrvSPI32_DisableRxInt(void);

- **Description**

  Disable the SPI Rx interrupt.

  Configure the register 0x40000[16]=0b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  None

- **Example**

```
/* Disable the SPI Rx interrupt */
DrvSPI32_DisableRxInt();
```

7.3.10. DrvSPI32_DisableTxInt

- **Prototype**

  void DrvSPI32_DisableTxInt(void);

- **Description**

  Disable the SPI Tx interrupt.

  Configure the register 0x40000[17]=0b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  None

- **Example**

```
/* Disable the SPI Tx interrupt */
DrvSPI32_DisableTxInt();
```

7.3.11. DrvSPI32_GetRxIntFlag

- **Prototype**

  unsigned int DrvSPI32_GetRxIntFlag ();

- **Description**

  Get the SPI RX interrupt flag.

  Read the register 0x40000[0].

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  1: Interrupted

  0: Normal

- **Example**

```
/* Get the SPI RX interrupt flag. */
unsigned char flag; flag=DrvSPI_GetRxIntFlag();
```

7.3.12. DrvSPI32_GetTxIntFlag

- **Prototype**

  unsigned int DrvSPI32_GetTxIntFlag ();

- **Description**

  Get the SPI TX interrupt flag.

  Read the register 0x40000[1]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  0: Interrupted

  1: Normal

- **Example**

```
/* Get the SPI Tx interrupt flag. */
unsigned char flag ;
flag=DrvSPI32_GetTxIntFlag();
```

7.3.13. DrvSPI32_ClrIntRxFlag

- **Prototype**

  void DrvSPI32_ClrIntRxFlag ();

- **Description**

  Clear the SPI Rx interrupt flag.

  Configure the register 0x40000[0]=0b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  None

- **Example**

```
/* Clear the SPI Rx interrupt flag. */
DrvSPI32_ClrIntRxFlag();
```

7.3.14. DrvSPI32_ClrIntTxFlag

- **Prototype**

  void DrvSPI32_ClrIntTxFlag ();

- **Description**

  Clear the SPI Tx interrupt flag.

  Configure the register 0x40000[1]=0b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  None

- **Example**

```
/* Clear the SPI Tx interrupt flag. */
DrvSPI32_ClrIntTxFlag();
```

7.3.15. DrvSPI32_Read

- **Prototype**

  unsigned int DrvSPI32_Read();

- **Description**

  Read data from SPI Rx buffer registers.

  Read the register 0x40F08[31:0]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  The return value is SPI Rx buffer register data.

- **Example**

```
/*Data transmission：LSB First，8bit*/
unsigned int data；data=DrvSPI32_Read()>>24;
/*Data transmission：MSB First，8bit*/
unsigned int data；data=DrvSPI32_Read();
```

7.3.16. DrvSPI32_Write

- **Prototype**

  void DrvSPI32_Write (unsigned int uData);

- **Description**

  Write data to SPI Tx buffer register. Configure the register 0x40F0C[31:0]

- **Parameters**

  uData [in] : Pre-sent data:0~0xFFFFFFFF

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  None

- **Example**

```
/*Data transmission：MSB First, 8bit Send 0x55*/
DrvSPI32_Write(0x55<<24);
/*Data transmission：LSB First, 8bit Send 0x55*/
DrvSPI32_Write(0x55);
```

7.3.17. DrvSPI32_Enable

- **Prototype**

  void DrvSPI32_Enable (void);

- **Description**

  Enable the SPI function.

  Configure the register 0x40F00[0]=1b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  None.

- **Example**

```
/* Enable the SPI */
DrvSPI32_Enable();
```

7.3.18. DrvSPI32_BitLength

- **Prototype**

  void DrvSPI32_BitLength (unsigned int uData);

- **Description**

  Set the SPI transfer Bit length.

  Configure the register 0x40F04[4:0]

- **Parameters**

  uData[in] : Specify SPI data length. It could be 0x04~0x20

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  None.

- **Example**

```
/* Set SPI transfer Bit length 8*/
DrvSPI32_BitLength(8);
```

7.3.19. DrvSPI32_GetDCFlag

- **Prototype**

  unsigned int DrvSPI32_GetDCFlag(void);

- **Description**

  Get data loss flag of state.

  Read the register 0x40F00[18]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  0: Normal.

  1: Rx buffer data is overwritten.

- **Example**

```
/* Check the status of DCF */
unsigned char flag ;
flag=DrvSPI32_GetDCFlag();
```

7.3.20. DrvSPI32_IsABFlag

- **Prototype**

  unsigned int DrvSPI32_IsABFlag(void);

- **Description**

  Check whether the data deficient

  Read the register 0x40F00[20]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  0: Normal.

  1: SPI Bus receive data length is less than BL.

- **Example**

```
/* Check the status of ABF */
unsigned char flag；flag=DrvSPI32_IsABFlag();
```

7.3.21. DrvSPI32_IsOVFlag

- **Prototype**

  unsigned int DrvSPI32_IsOVFlag(void);

- **Description**

  Check whether the received data is too long

  Read the register 0x40F00[21]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  0: Normal.

  1: SPI Bus receive data length is greater than BL

- **Example**

```
/* Check the status of OVF */
unsigned char flag；flag=DrvSPI32_IsOVFlag();
```

7.3.22. DrvSPI32_IsRxFlag

- **Prototype**

  unsigned int DrvSPI32_IsRxFlag(void);

- **Description**

  Check whether the Rx buffer data in the update.

  Read the register 0x40F00[22]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  0: Normal.

  1: SPI buffer data update

- **Example**

```
/* Check the status of RxF */
unsigned char flag；flag=DrvSPI32_IsRxFlag();
```

7.3.23. DrvSPI32_SetEndian

- **Prototype**

  void DrvSPI32_SetEndian(E_DRVSPI_ENDIAN eEndian);

- **Description**

  Set the data transfer from the MSB or LSB start

  Configure the register 0x40F04[18]

- **Parameters**

  eEndian [in] : the input range is 0~1

  1：Send LSB first

  0：Send MSB first

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  None

- **Example**

```
/* The transfer order is LSB first */
DrvSPI32_SetEndian(E_DRVSPI_LSB_FIRST);
```

7.3.24. DrvSPI32_SetCSO

- **Prototype**

  void DrvSPI32_SetCSO(E_DRVSPI_CS eCS);

- **Description**

  Set the CS signal simulator control bit, Configure the register 0x40F04[20]

  Note: The old function DrvSPI32_SetCS(E_DRVSPI_CS eCS) is the same operation as DrvSPI32_SetCSO(E_DRVSPI_CS eCS)

- **Parameters**

  eCS[in]:

  0: CS signal active low

  1: CS signal active high

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  None

- **Example**

```
/* Set low level is effective */
DrvSPI32_SetCSO(E_DRVSPI_CSLow);
```

7.3.25. DrvSPI32_DisableIO

- **Prototype**

  void DrvSPI32_DisableIO(void);

- **Description**

  Disable the SPI port to transmit

  Configure the register 0x40844[4]=0

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  None

- **Example**

```
/* Disable the SPI port to transmit */
DrvSPI32_DisableIO();
```

7.3.26. DrvSPI32_EnableIO

- **Prototype**

  unsigned char DrvSPI32_EnableIO(uint32_t uOuputPin);

- **Description**

  Enable and specify the SPI port to transmit

  Configure the register 0x40844[7:5] / 0x40844[4]=1;

- **Parameters**

  uOuputPin [in]：specify the port as SPI, the effectively input range is 0~3.

  0 : Port1.0 =CS, Port1.1 =CK, Port1.2 = DI, Port1.3 =DO

  1 : Port1.4 =CS, Port1.5 =CK, Port1.6 = DI, Port1.7 =DO

  2 : Port2.0 =CS, Port2.1 =CK, Port2.2 = DI, Port2.3 =DO

  3 : Port2.4 =CS, Port2.5 =CK, Port2.6 = DI, Port2.7 =DO

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Value**

  None

- **Example**

```
/* Enable the SPI port to transmit , select PT2.0~PT2.3*/
DrvSPI32_EnableIO(2);
```

8. UART Driver

8.1. Introduction

The Universal Asynchronous Receiver/Transmitter (UART) performs a serial-to-parallel conversion on data characters received from the peripheral such as MODEM, and a parallel-to-serial conversion on data characters received from the CPU. Details please refer to the section in the target chip specification titled UART.

Item	Functions	Description
01	DrvUART_Open	Set UART module
02	DrvUART_Close	Close UART module
03	DrvUART_EnableInt	Enable the UART interrupt
04	DrvUART_GetTxFlag	Get the TX interrupt flag of UART
05	DrvUART_GetRxFlag	Get the RX interrupt flag of UART
06	DrvUART_ClrTxFlag	Clear the TX interrupt flag of UART
07	DrvUART_ClrRxFlag	Clear the RX interrupt flag of UART
08	DrvUART_Read	Read data from RCREG of UART
09	DrvUART_Read9Bit	Receive 9Bits data
10	DrvUART_Write	Write data to TXREG of UART
11	DrvUART_EnableWakeUp	Enable wake-up mode of UART
12	DrvUART_GetPERR	Get the PERR flag of UART
13	DrvUART_GetFERR	Get the FERR flag of UART
14	DrvUART_GetOERR	Get the OERR flag of UART
15	DrvUART_GetABDOVF	Get the ABDOVF flag of UART
16	DrvUART_Enable_AutoBaudrate	Enable Auto Baudrate of UART
17	DrvUART_Disable_AutoBaudrate	Disable Auto Baudrate of UART
18	DrvUART_CheckTRMT	Read the flag of Transmit Shift Register Status
19	DrvUART_ClkEnable	Enable and select the UART clock source
20	DrvUART_ClkDisable	Disable the UART `clock source
21	DrvUART_Enable	Enable the UART function
22	DrvUART_ConfigIO	Enable and select the IO port as UART communication port

8.2. Type Definition

E_DATABITS_SETTINGS

Enumeration identifier	Value	Description
DRVUART_DATABITS_8	0x0	Word length select: Character length is 8 bits.
DRVUART_DATABITS_9	0x1	Word length select: Character length is 9 bits.

E_PARITY_SETTINGS

Enumeration identifier	Value	Description
DRVUART_PARITY_NONE	0x0	None parity
DRVUART_PARITY_ODD	0x1	Odd parity enable
DRVUART_PARITY_EVEN	0x2	Even parity enable

E_BAUD_RATE_SETTINGS

Enumeration identifier	Value	Description
B1200	0x0	Baud rate=1200
B2400	0x1	Baud rate=2400
B4800	0x2	Baud rate=4800
B9600	0x3	Baud rate=9600
B14400	0x4	Baud rate=14400
B19200	0x5	Baud rate=19200
B38400	0x6	Baud rate=38400

E_UART_ERROR_MESSAGE

Enumeration identifier	Value	Description
E_UART_ERR_CLOCK	0x2	CLOCK Parameter input error
E_UART_ERR_BAUDRATE	0x3	Baud rate Parameter input error
E_UART_ERR_PARITY	0x4	Parity Parameter input error
E_UART_ERR_DATABIT	0x5	Data bit Parameter input error
E_UART_ERR_OUTPIN	0x6	Output pin Parameter input error

8.3. Functions

8.3.1. DrvUART_Open

● **Prototype**

```
unsigned int DrvUART_Open (
                        unsigned int    uClock
E_RAUD_RATE_SETTINGS    uBaudRate ,
E_PARITY_SETTINGS       uParity,
E_DATABITS_SETTINGS     uDataBits,
unsigned int            uStopBits,
unsigned int            uOuputPin
);
```

● **Description**

Select the UART frequency value to used (Should be noted, oscillator clock soure HSXT or HSRC effects UART frequency value, UART divider also effects UART frequency value), to automatically calculate the value to register 0x40E0C[4:0] / 0x40E10[7:0], according to input the required baud rate value,UART with bit set, set the UART data-bit, Stop-bit,set the UART output pin.

Configure the register 0x40E00[7]=1, 0x40E00[6]=1, 0x40E00[5] / 0x40E00[3] ;
0x40E0C[3:0]/0x40E10[7:0], 0x40844[3:0].

● **Parameter**

uClock : Type UART frequency value in kHz Unit. The input value of UART is URCK frequency. URCK frequency is selected from external HSXT or internal HSRC clock source, and it goes throught UACD[3:0] divider. If UACD=1, URCK=HSXT(or HSRC). If UACD=2, URCK=HSXT/2(or HSRC/2) and so on.
The input range is 1000~20000

uBaudRate [in] : Type baud rate

uParity [in] : NONE/EVEN/ODD parity, It could be

0 : None parity

1 : Even parity

2 : Odd parity.

uDataBits[in] : data bit setting, It could be

0 : 8 data bits.

1 : 9 data bits.

uStopBits[in] : stop bit setting

0 : 0.5 Bit 1: 1 Bit

2 : 1.5 Bit 3 : 2 Bit

uOuputPin [in] :

0 : Port 1.0 =TX, Port 1.1 =RX

1 : Port 1.2 =TX, Port 1.3 =RX

2 : Port 1.4 =TX, Port 1.5 =RX

3 : Port 1.6 =TX, Port 1.7 =RX

4 : Port 2.0 =TX, Port 2.1 =RX

5 : Port 2.2 =TX, Port 2.3 =RX

6 : Port 2.4 =TX, Port 2.5 =RX

7 : Port 2.6 =TX, Port 2.7 =RX

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Value**

  0: Success.

  2 : Wrong clock setting

  3 : Wrong baud rate setting

  4: Wrong party setting

  5: Wrong Data bit setting

  6:Wrong Stop bit setting

  7: Wrong output pin setting

- **Example**

```
/* Set UART baud rate 115200bps, 8 data bits ,1 stop bit, and none parity. PT1.4/PT1.5 used as interface*/
DrvUART_Open(4000,115200, DRVUART_PARITY_NONE ,DRVUART_DATABITS_8,2);
```

  Note : Because UART frequency value is 4MHz, so input value is 4000. The unit is kHz

8.3.2. DrvUART_Close

- **Prototype**

```
void DrvUART_Close (void );
```

- **Description**

  Disable uart

  Clear the register 0x40E00[7]=0.

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Value**

  None

- **Example**

```
/* Close UART */
DrvUART_Close();
```

8.3.3. DrvUART_EnableInt

- **Prototype**

  unsigned int DrvUART_EnableInt(unsigned int uTXIE, unsigned int uRXIE);

- **Description**

  Enable the UART TX or RX interrupt.

  Configure the register 0x40000[19:18]

- **Parameters**

  uTXIE [in] : UART Tx Interrupt

  0 : Disable

  1 : Enable

  uRXIE [in] : UART Rx Interrupt

  0 : Disable

  1 : Enable

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Enable the UART TX and RX interrupt */
DrvUART_EnableInt(1,1);
```

8.3.4. DrvUART_GetTxFlag

- **Prototype**

  unsigned int DrvUART_GetTxFlag ();

- **Description**

  Get the Tx interrupt flag of UART.

  Read the register 0x40000[3]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Value**

  1: Interrupted

  0: Normal

- **Example**

```
/* Get the Tx interrupt flag. */
DrvUART_GetTxFlag();
```

8.3.5. DrvUART_GetRxFlag

- **Prototype**

  unsigned int DrvUART_GetRxFlag ();

- **Description**

  Get the Rx interrupt flag of UART.

  Read the register 0x40000[2]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Get the Rx interrupt flag. */
  unsigned char flag ; flag=DrvUART_GetRxFlag();
```

8.3.6. DrvUART_ClrTxFlag

- **Prototype**

  void DrvUART_ClrTxFlag ();

- **Description**

  Clear the Tx interrupt flag of UART.

  Configure the register 0x40000[3]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Value**

None

- **Example**

/* Clear the Tx interrupt flag. */

DrvUART_ClrTxFlag();

8.3.7. DrvUART_ClrRxFlag

- **Prototype**

void DrvUART_ClrRxFlag ();

- **Description**

Clear the Rx interrupt flag of UART.

Configure the register 0x40000[2]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Value**

None

- **Example**

/* Clear the Rx interrupt flag. */

DrvUART_ClrRxFlag();

8.3.8. DrvUART_Read

- **Prototype**

unsigned int DrvUART_Read();

- **Description**

Read data received from UART.

Read the register 0x40E18[7:0]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Value**

The return value is RX Data buffer register data.

- **Example**

/* Read the RX Data buffer register data. */

unsined int rx_data；rx_data=DrvUART_Read();

8.3.9. DrvUART_Read9Bit

- **Prototype**

  unsigned int DrvUART_Read9Bit(void);

- **Description**

  Receive 9Bits data

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Value**

  9Bits data

- **Example**

  /* Receive 9Bits data */

  unsigned int data; data=DrvUART_Read9Bit();

8.3.10. DrvUART_Write

- **Prototype**

  void DrvUART_Write(unsigned int uData);

- **Description**

  Write data to TX data register of UART.

  Configure the register 0x40E14[7:0]

- **Parameters**

  uData [in]

  data to be sent

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Value**

  None

- **Example**

  /* Using UART to send one byte 0x55 */

  DrvUART_Write(0x55);

8.3.11. DrvUART_EnableWakeUp

- **Prototype**

  void DrvUART_EnableWakeUp();

- **Description**

  Enable wake-up mode of UART

  Configure the register 0x40E00[0]=1

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Value**

  None

- **Example**

```
/* Enable wake up. */
DrvUART_EnableWakeUp();
```

8.3.12. DrvUART_GetPERR

- **Prototype**

  unsigned int DrvUART_GetPERR();

- **Description**

  Get the Parity Error flag of UART.

  Read the register 0x40E04[5]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Value**

  1 : Parity error

  0 : No parity error

- **Example**

```
/* Get the PERR flag. */
unsigned char flag; flag=DrvUART_GetPERR();
```

8.3.13. DrvUART_GetFERR

- **Prototype**

  unsigned int DrvUART_GetFERR();

- **Description**

  Get the FERR flag of UART.

  Read the register 0x40E04[4]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Value**

  1 : Framing error

  0 : No framing error

- **Example**

```
/* Get the FERR flag. */
unsigned char flag ; flag=DrvUART_GetFERR();
```

8.3.14. DrvUART_GetOERR

- **Prototype**

  unsigned int DrvUART_GetOERR();

- **Description**

  Get the OERR flag of UART.

  Read the register 0x40E04[3]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Value**

  1 : Overrun error

  0 : No overrun error

- **Example**

```
/* Get the OERR flag. */
unsigned char flag ; flag=DrvUART_GetOERR();
```

8.3.15. DrvUART_GetABDOVF

- **Prototype**

  unsigned int DrvUART_GetABDOVF();

- **Description**

  Get the RxABDF flag of UART.

  Read the register 0x40E04[0]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Value**

  1 : A BRG rollover has occurred during Auto-Baud Rate Detect mode

  0 : No BRG rollover has occurred

- **Example**

```
/* Get the RxABDF flag. */
unsigned char flag ; flag=DrvUART_GetABDOVF();
```

8.3.16. DrvUART_Enable_AutoBaudrate

- **Prototype**

  void DrvUART_Enable_AutoBaudrate ();

- **Description**

  Enable Auto Baudrate of UART

  Configure the register 0x40E08[0]=1.

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Value**

  None

- **Example**

```
/* Enable Auto Baudrate */
DrvUART_Enable_AutoBaudrate();
```

8.3.17. DrvUART_Disable_AutoBaudrate

- **Prototype**

  void DrvUART_Disable_AutoBaudrate ();

- **Description**

  Disable Auto Baudrate of UART

  Configure the register 0x40E08[0]=0

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Value**

  None

- **Example**

```
/* Disable Auto Baudrate */
DrvUART_Disable_AutoBaudrate();
```

8.3.18. DrvUART_CheckTRMT

- **Prototype**

  unsigned int DrvUART_CheckTRMT(void)

- **Description**

  Read the UART flag of Transmit Shift Register Status(TXBF).

  Read the register 0x40E04[1]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Value**

  0: Transmit Shift Register empty

  1: Transmit Shift Register full

- **Example**

```
/* Read the flag of Transmit Shift Register Status */
DrvUART_Write(data) ;
While(DrvUART_CheckTRMT()) ;//wait TRMT=0
While( !DrvUART_CheckTRMT() ) ;//wait TRMT=1
```

8.3.19. DrvUART_ClkEnable

- **Prototype**

  unsigned int DrvUART_ClkEnable(unsigned int uclk,unsigned int uprescale)

● **Description**

Enable and select the UART clock source . Specify the clock source divider

Configure the register 0x40308[21:16]

● **Parameters**

uclk[in] : EUART clock source

0 : External high speed oscillator

1 : Internal high speed oscillator

uprescale[in] : the clock source divider

0000	EUART CLOCK SOURCE/1	1000	EUART CLOCK SOURCE/256
0001	EUART CLOCK SOURCE/2	1001	EUART CLOCK SOURCE/512
0010	EUART CLOCK SOURCE/4	1010	EUART CLOCK SOURCE/1024
0011	EUART CLOCK SOURCE/8	1011	EUART CLOCK SOURCE/2048
0100	EUART CLOCK SOURCE/16	1100	EUART CLOCK SOURCE/4096
0101	EUART CLOCK SOURCE/32	1101	EUART CLOCK SOURCE/8192
0110	EUART CLOCK SOURCE/64	1110	EUART CLOCK SOURCE/16384
0111	EUART CLOCK SOURCE/128	1111	EUART CLOCK SOURCE/32768

● **Include**

Peripheral_lib/DrvUART.h

● **Return Value**

0: Operation successful

Other : Incorrect argument

● **Example**

```
/* select the external clock source, divider:clk/1 */
DrvUART_ClkEnable(0,0);
```

8.3.20. DrvUART_ClkDisable

● **Prototype**

void DrvUART_ClkDisable(void) ;

● **Description**

Disable the UART clock source.

Configure the register 0x40308[20]=0

● **Parameters**

None

● **Include**

Peripheral_lib/DrvUART.h

● **Return Value**

None

- **Example**

```
/* Disable the UART clock source */
DrvUART_ClkDisable();
```

8.3.21. DrvUART_Enable

- **Prototype**

```
void DrvUART_Enable(void) ;
```

- **Description**

Enable the UART function

Configure the register 0x40E00[7:6]=11b

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Value**

None

- **Example**

```
/* Enable the UART clock source */
DrvUART_Enable();
```

8.3.22. DrvUART_ConfigIO

- **Prototype**

```
unsigned char DrvUART_ConfigIO(unsigned char ioen,unsigned int uOuputPin) ;
```

- **Description**

Enable and select the IO port as UART communication port

Configure the register 0x40844[3:0]

- **Parameters**

ioen[in] : EURAT1 input/output to port enable control

0 : disable

1 : enable

uoutputPin[in] : select the UART communication port

0 : Port 1.0 =TX, Port 1.1 =RX

1 : Port 1.2 =TX, Port 1.3 =RX

2 : Port 1.4 =TX, Port 1.5 =RX

3 : Port 1.6 =TX, Port 1.7 =RX

4 : Port 2.0 =TX, Port 2.1 =RX

5 : Port 2.2 =TX, Port 2.3 =RX

6 : Port 2.4 =TX, Port 2.5 =RX

7 : Port 2.6 =TX, Port 2.7 =RX

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* enable and select PT2.0/PT2.1 as UART port*/
DrvUART_ConfigIO(1,4);
```

9. CMP Driver

9.1. Introduction

The following functions are included in CMP Manager Section.

Item	Functions	Description
01	**錯誤! 找不到參照來源。**	Open Comparator
02	DrvCMP_Close	Close Comparator
03	DrvCMP_Enable	Enable CMP
04	DrvCMP_PInput	Positive reference input
05	DrvCMP_NInput	Negative reference input
06	DrvCMP_InputSwitch	Comparator input short switch
07	DrvCMP_OutputFilter	Digital filter options.
08	DrvCMP_OutputPinEnable	Enable the CMP digital output
09	DrvCMP_OutputPinDisable	Disable the CMP digital output
10	DrvCMP_OutputInverse	The digital output CMP inverse
11	DrvCMP_EnableInt	CMP Interrupt Enable
12	DrvCMP_DisableInt	CMP Interrupt Disable
13	DrvCMP_ReadIntFlag	Read CMP Interrupt flag
14	DrvCMP_ClearIntFlag	Clear CMP Interrupt flag
15	DrvCMP_Input	Comparator reference input
16	DrvCMP_RLO_Ctrl	Internal network configuration
17	DrvCMP_RLO_refv	Comparator reference voltage
18	DrvCMP_EnableNonOverlap	Enable the non-overlap
19	DrvCMP_DisableNonOverlap	Disable the non-overlap
20	DrvCMP_ReadData	Read CMP Digital output status

9.2. Type Definition

E_NON_OVERLAP_PIN

Enumeration identifier	Value	Description
E_CL1	0x0	Comparator positive reference input from the PT1.2
E_CL2	0x1	Comparator positive reference input from the PT1.3
E_CL3	0x2	Comparator positive reference input from the PT3.1
E_CL4	0x3	Comparator positive reference input from the PT3.2

9.3. Functions

9.3.1. DrvCMP_Open

- **Prototype**

  unsigned int DrvCMP_Open (uPowerMode , uCPDF, uCPOR)

- **Description**

  Enable Comparator, Power modes can choose low power or normal · whether the comparator output after deglitch filter,set output signal whether the reverse or not

  Configure the register 0x41804[7:6] / 0x41804[1:0]

- **Parameter**

  uPowerMode [in] : Comparator low power mode enable.

  0 : Low power mode

  1 : Normal mode

  uCPDF [in] : Comparator output deglitch filter

  0 : No deglitch filter

  1 : With 2us delay deglitch filter

  uCPOR [in] : Output inverse control

  0 : Normal

  1 : Inverse

- **Include**

  Peripheral_lib/DrvCMP.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Enable comparator, Choose low power mode and 2us deglitch filter. */
DrvCMP_Open(0,1,0);
```

9.3.2. DrvCMP_Close

- **Prototype**

  void DrvCMP_Close (void)

- **Description**

  Close comparator. Configure the register 0x41804[0]=0

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvCMP.h

- **Return Value**

  None

- **Example**

  /* Close comparator */

  DrvCMP_Close();

9.3.3. DrvCMP_Enable

- **Prototype**

  void DrvCMP_Enable (void)

- **Description**

  Enable CMP. Configure the register 0x41804[0]=1.

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvCMP.h

- **Return Value**

  None

- **Example**

  /* Enable CMP */

  DrvCMP_Enable();

9.3.4. DrvCMP_PInput

- **Prototype**

  unsigned int DrvCMP_PInput (uCPPS)

- **Description**

  Comparator positive reference input selection. Configure the register 0x41800[7:6].

- **Parameter**

  uCPPS [in] : Comparator positive reference input selection.

  0 : CH1

  1 : CH2

  2 : CH3

  3 : V12 (V12=1.2V)

- **Include**

Peripheral_lib/DrvCMP.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

/* Select the CMP positive input of CH1. */

DrvCMP_PInput(0);

9.3.5. DrvCMP_NInput

- **Prototype**

unsigned int DrvCMP_NInput (uCPNS)

- **Description**

Comparator negative reference input selection. Configure the register 0x41800[5:4]

- **Parameter**

uOPNS [in] : Comparator negative reference input selection.

0 : CH1

1 : CH2

2 : CH3

3 : RLO

- **Include**

Peripheral_lib/DrvCMP.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

/* Select the CMP negative input of CH1. */

DrvCMP_NInput(0);

9.3.6. DrvCMP_InputSwitch

- **Prototype**

unsigned int DrvCMP_InputSwitch (uInputSwitch)

- **Description**

Comparator input short switch.

Configure the register 0x41804[5]

- **Parameter**

uInputSwitch[in] : Comparator input short switch.

0 : Open

1 : Short

- **Include**

Peripheral_lib/DrvCMP.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Comparator input short */
DrvCMP_InputSwitch(1);
```

9.3.7. DrvCMP_OutputFilter

- **Prototype**

unsigned int DrvCMP_OuputFilter(uFilter)

- **Description**

The output of CMP connected with the digital filter options. Configure the register 0x41804[1]

- **Parameter**

uFilter[in] :

0 : Disable

1 : Enable(pass a 2us deglitch)

- **Include**

Peripheral_lib/DrvCMP.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* CMP output delay 2us. */
DrvCMP_OuputFilter(1);
```

9.3.8. DrvCMP_OutputPinEnable

- **Prototype**

unsigned int DrvCMP_OutputPinEnable (E_OUTPUT_PIN uPin)

- **Description**

Enable the CMP digital output to port.

Configure the register 0x40840[16]=1

- **Parameter**

uPin [in]

0 : PT1.7

- **Include**

Peripheral_lib/DrvCMP.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Enable the CMP digital output */
DrvCMP_OutputPinEnable(0);
```

9.3.9. DrvCMP_OutputPinDisable

- **Prototype**

void DrvCMP_OutputPinDisable (void)

- **Description**

Disable the CMP digital output to port.

Configure the register 0x40840[16]=0

- **Parameter**

None

- **Include**

Peripheral_lib/DrvCMP.h

- **Return Value**

None

- **Example**

```
/* Disable the CMP digital output */
DrvCMP_OutputPinDisable();
```

9.3.10. DrvCMP_OutputInverse

- **Prototype**

unsigned int DrvCMP_OutputInverse(uInv)

- **Description**

The digital output CMP inverse control.

Configure the register 0x41804[7]

- **Parameter**

uInv [in]

0 : Normal

1 : Output Reverse

- **Include**

Peripheral_lib/DrvCMP.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Set CMP digital output inverse control */
DrvCMP_OutputInverse(1);
```

9.3.11. DrvCMP_EnableInt

- **Prototype**

void DrvCMP_EnableInt (void)

- **Description**

CMP Interrupt Enable

Configure the register 0x40008[17]=1

- **Parameter**

None

- **Include**

Peripheral_lib/DrvCMP.h

- **Return Value**

None

- **Example**

```
/* Enable CMP interrupt */
DrvCMP_EnableInt();
```

9.3.12. DrvCMP_DisableInt

- **Prototype**

void DrvCMP_DisableInt (void)

- **Description**

CMP Interrupt Disable

Configure the register 0x40008[17]=0

- **Parameter**

None

- **Include**

Peripheral_lib/DrvCMP.h

- **Return Value**

None

- **Example**

/* Disable CMP interrupt */

DrvCMP_DisableInt();

9.3.13. DrvCMP_ReadIntFlag

- **Prototype**

unsigned int DrvCMP_ReadIntFlag (void)

- **Description**

Read CMP interrupt flag.

Configure the register 0x40008[1]

- **Parameter**

None

- **Include**

Peripheral_lib/DrvCMP.h

- **Return Value**

0 : Normal. Interrupt flag is0

1 : Interrupted. Interrupt flag is1

>1: Invalid return value

- **Example**

/* Read CMP interrupt flag */

unsigned char flag；flag=DrvCMP_ReadIntFlag();

9.3.14. DrvCMP_ClearIntFlag

- **Prototype**

void DrvCMP_ClearIntFlag (void)

- **Description**

Clear CMP interrupt flag.

Configure the register 0x40008[1] =0

- **Parameter**

None

- **Include**

Peripheral_lib/DrvCMP.h

- **Return Value**

  None

- **Example**

  /* Clear CMP interrupt flag */

  DrvCMP_ClearIntFlag();

9.3.15. DrvCMP_Input

- **Prototype**

  unsigned int DrvCMP_Input (uPositiveInput, uNegativeInput,uInputSwitch)

- **Description**

  Comparator positive and negative reference input selection. Set the Input short switch.

  Configure the register 0x41800[7:6] / 0x41800[5:4] / 0x41804[5] . .

- **Parameter**

  uPositiveInput[in] : Comparator positive reference input selection.

  0 : CH1

  1 : CH2

  2 : CH3

  3 : V12

  uNegativeInput[in] : Comparator negative reference input selection.

  0 : CH1

  1 : CH2

  2 : CH3

  3 : RLO

  uInputSwitch[in] : Comparator input short switch.

  0 : Open

  1 : Short

- **Include**

  Peripheral_lib/DrvCMP.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

  /* Select CPPS=CH1,CPNS=CH3, the input switch is open */

  DrvCMP_Input(0,2,0);

9.3.16. DrvCMP_RLO_Ctrl

- **Prototype**

  unsigned int DrvCMP_RLO_Ctrl (uCPDA ,uCPDM)

- **Description**

  Comparator network internal network configuration

  Configure the register 0x41804[23:20] / 0x41804[19:16]

- **Parameter**

  uCPDA[in] : Comparator internal resistor ladder control.

  0 : 0

  1 : 1/16 (CPRLH – CPRLL)

  2 : 2/16 (CPRLH – CPRLL)

  3 : 3/16 (CPRLH – CPRLL)

  4 : 4/16 (CPRLH – CPRLL)

  5 : 5/16 (CPRLH – CPRLL)

  6 : 6/16 (CPRLH – CPRLL)

  7 : 7/16 (CPRLH – CPRLL)

  8 : 8/16 (CPRLH – CPRLL)

  9 : 9/16 (CPRLH – CPRLL)

  10 :10/16 (CPRLH – CPRLL)

  11 :11/16 (CPRLH – CPRLL)

  12 :12/16 (CPRLH – CPRLL)

  13 :13/16 (CPRLH – CPRLL)

  14 :14/16 (CPRLH – CPRLL)

  15 : 15/16 (CPRLH – CPRLL)

  uCPDM [3:0] : Comparator output with hysteresis controller. In the Hysteresis mode, the uCPDA[3:0] corresponding bit is the same with CMPO.

  uCPDM [3] 0 : disable

  1 : enable

  uCPDM [2] 0 : disable

  1 : enable

  uCPDM [1] 0 : disable

  1 : enable

  uCPDM [0] 0 : disable

  1 : enable

- **Include**

  Peripheral_lib/DrvCMP.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

/* Set CPDM=0101,CPDA=0011 */

DrvCMP_RLO_Ctrl(0x03,0x05);

9.3.17. DrvCMP_RLO_refv

- **Prototype**

unsigned int DrvCMP_RLO_refV (uCPRH, uCPRLS)

- **Description**

Comparator reference voltage control

Configure the register 0x41800[2:1] / 0x41800[3]

- **Parameter**

uCPRH [in] : Comparator resistor ladder high voltage selection.

0 : off(high impendent)

1 : CP_I charge pump input

2 : VDD3V (system power voltage)

3 : VDD18 (digital power voltage)

uCPRLS [in] : Comparator resistor ladder short switch.

0 : Off

1 : On

- **Include**

Peripheral_lib/DrvCMP.h

- **Return Value**

0: Operation successful

1: Incorrect argument

- **Example**

/* Set comparator resistor ladder high voltage from VDD18, resistor ladder short */

DrvCMP_RLO_refV(3,1);

9.3.18. DrvCMP_EnableNonOverlap

- **Prototype**

unsigned int DrvCMP_EnableNonOverlap (E_NON_OVERLAP_PIN uInput)

- **Description**

Enable the non-overlap and select reference input.

Configure the register 0x41804[4:2]

- **Parameter**

  uInput [in] : Comparator positive reference input selection.

  0 : CL1

  1 : CL2

  2 : CL3

  3 : CL4

- **Include**

  Peripheral_lib/DrvCMP.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Enable the non-overlap and select CL1 reference input.   */
DrvCMP_EnableNonOverlap(0)
```

9.3.19. DrvCMP_DisableNonOverlap

- **Prototype**

  void DrvCMP_DisableNonOverlap (void)

- **Description**

  Disable the non-overlap

  Configure the register 0x41804[4] =0

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvCMP.h

- **Return Value**

  None

- **Example**

```
/* Disable the non-overlap. */
DrvCMP_DisableNonOverlap();
```

9.3.20. DrvCMP_ReadData

- **Prototype**

  unsigned int DrvCMP_ReadData (void)

- **Description**

  Read CMP Digital output status.

  Configure the register 0x41800[16]

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvCMP.h

- **Return Value**

  0: Negative input > positive input

  1: Positive input > negative

- **Example**

```
/* Read CMP Digital output status */
unsigned char flag；flag=DrvCMP_ReadData();
```

10. OPAMP Driver

10.1. Introduction

The following functions are included in OPA Manager Section.

Item	Functions	Description
01	DrvOP_Open	Enable OPAMP
02	DrvOP_Close	Close OPAMP
03	DrvOP_PInput	Positive input selection
04	DrvOP_NInput	Negative input selection.
05	DrvOP_OPOoutEnable	OPAMP output enable
06	DrvOP_OPOoutDisable	OPAMP output disable
07	DrvOP_OuputFilter	OPAMP digital filter options
08	DrvOP_OutputPinEnable	Enable digital output to port.
09	DrvOP_OutputPinDisable	Disable digital output to port.
10	DrvOP_OutputInverse	The digital output inverse
11	DrvOP_OutputWithCHPCK	CPCLK multiplier selection.
12	DrvOP_EnableInt	OPA Interrupt Enable
13	DrvOP_DisableInt	OPA Interrupt Disable
14	DrvOP_ReadIntFlag	Read OPAMP interrupt flag
15	DrvOP_ClearIntFlag	Clear OPAMP interrupt flag.
16	DrvOP_Feedback	OPAMP feedback settings
17	DrvOP_OPDEN	OPAMP digital output function control

10.2. Type Definition

E_OUTPUT_PIN

Enumeration identifier	Value	Description
E_OPO1	0x0	OPAMP output digital output from the PT3.0
E_OPO2	0x1	OPAMP output digital output from the PT3.1

E_OPN_PPIN

Enumeration identifier	Value	Description
E_OPP_AIO2	0x1	OPAMP input from AIO2
E_OPP_AIO4	0x2	OPAMP input from AIO4
E_OPP_DAOI	0x4	OPAMP input from DAOI
E_OPP_REFO_I	0x8	OPAMP input from REFO_I
E_OPN_AIO3	0x1	OPAMP input from AIO3
E_OPN_AIO5	0x2	OPAMP input from AIO5
E_OPN_DAOI	0x4	OPAMP input from DAOI
E_OPN_OPOI	0x8	OPAMP input from OPOI
E_OPN_OPO	0x10	OPAMP input from OPO
E_OPN_OPC	0x20	OPAMP input from OPC

10.3. Functions

10.3.1. DrvOP_Open

- **Prototype**

  void DrvOP_Open (void)

- **Description**

  Enable OPAMP

  Configure the register 0x41900[0]=1b

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvOP.h

- **Return Value**

  None

- **Example**

```
/* Enable OP */
DrvOP_Open();
```

10.3.2. DrvOP_Close

- **Prototype**

  void DrvOP_Close (void)

- **Description**

  Close OPAMP

  Configure the register 0x41900[0]=0b

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvOP.h

- **Return Value**

  None

- **Example**

```
/* Close OP */
DrvOP_Close();
```

10.3.3. DrvOP_PInput

- **Prototype**

  unsigned int DrvOP_PInput (uOPPS)

- **Description**

  Rail-to-rail OPAMP positive input selection.

  Configure the register 0x41904[19:16]

- **Parameter**

  uOPPS [in] : Rail-to-rail OPAMP positive input selection. It could be 0~0xf

  uOPPS[3] : OPAMP positive input channel 3

  0 : Turn-off: High impendent

  1 : Turn-on and connect to REFO_I

  uOPPS[2] : OPAMP positive input channel 2

  0 : Turn-off: High impendent

  1 : Turn-on and connect to DAOI

  uOPPS[1] : OPAMP positive input channel 1

  0 : Turn-off: High impendent

  1 : Turn-on and connect to AIO4

  uOPPS[0] : OPAMP positive input channel 0

  0 : Turn-off: High impendent

  1 : Turn-on and connect to AIO2

- **Include**

  Peripheral_lib/DrvOP.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Selection the Rail-to-rail OPAMP positive input of AIO2 and AIO4. */
DrvOP_PInput(0x1| 0x2);
```

10.3.4. DrvOP_NInput

- **Prototype**

  unsigned int DrvOP_NInput (uOPNS)

- **Description**

  Rail-to-rail OPAMP negative input selection.

  Configure the register 0x41904[5:0]

- **Parameter**

uOPPS [in] : Rail-to-rail OPAMP negative input selection register. It could be 0~0x3f

uOPNS[5] : OPAMP negative input channel 5

0 : Turn-off: High impendent

1 : Turn-on and connect to OPC: Internal 10pF capacitor

uOPNS[4] : OPAMP negative input channel 4

0 : Turn-off: High impendent

1 : Turn-on and connect to OPO: Internal OPAMP output

uOPNS[3] : OPAMP negative input channel 3

0 : Turn-off: High impendent

1 : Turn-on and connect to OPOI: External OPAMP output

uOPNS[2] : OPAMP negative input channel 2

0 : Turn-off: High impendent

1 : Turn-on and connect to DAOI

uOPNS[1] : OPAMP negative input channel 1

0 : Turn-off: High impendent

1 : Turn-on and connect to AI5

uOPNS[0] : OPAMP negative input channel 0

0 : Turn-off: High impendent

1 : Turn-on and connect to AI3

- **Include**

  Peripheral_lib/DrvOP.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Selection the Rail-to-rail OPAMP negative input of AIO3 and AIO5. */
DrvOP_NInput (0x1|0x2);
```

10.3.5. DrvOP_OPOoutEnable

- **Prototype**

  void DrvOP_OPOoutEnable(void)

- **Description**

  OPAMP output enable

  Configure the register 0x41900[1] =1b

- **Parameter**

  None

- **Include**

Peripheral_lib/DrvOP.h

- **Return Value**

None

- **Example**

/* OPAMP output enable */

DrvOP_OPOoutEnable();

10.3.6. DrvOP_OPOoutDisable

- **Prototype**

void DrvOP_OPOoutDisable(void)

- **Description**

OPAMP output disable

Configure the register 0x41900[1]=0b

- **Parameter**

None

- **Include**

Peripheral_lib/DrvOP.h

- **Return Value**

None

- **Example**

/* OPAMP output disable */

DrvOP_OPOoutDisable();

10.3.7. DrvOP_OuputFilter

- **Prototype**

unsigned int DrvOP_OuputFilter(uFilter)

- **Description**

The output of OPAMP connected with the digital filter options.

Configure the register 0x41900[3]

- **Parameter**

uFilter[in]

0 : Disable

1 : Enable(pass a 2us deglitch)

- **Include**

Peripheral_lib/DrvOP.h

- **Return Value**

  0: Operation successful

  Other: Incorrect argument

- **Example**

```
/* OPAMP output delay 2us. */
DrvOP_OuputFilter(1);
```

10.3.8. DrvOP_OutputPinEnable

- **Prototype**

  unsigned int DrvOP_OutputPinEnable (E_OUTPUT_PIN uPin)

- **Description**

  Enable and select the OPAMP digital output to port.

  Configure the register 0x41900[2]=1b, 0x40840[19:18]

- **Parameter**

  uPin [in]

  0 : PT3.0

  1 : PT3.1

- **Include**

  Peripheral_lib/DrvOP.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Enable the OPAMP digital output    IO=PT3.0*/
DrvOP_OutputPinEnable(0);
```

10.3.9. DrvOP_OutputPinDisable

- **Prototype**

  void DrvOP_OutputPinDisable (void)

- **Description**

  Disable the OPAMP digital output to port.

  Configure the register 0x41900[2]=0, 0x40840[18]=0

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvOP.h

- **Return Value**

  None

- **Example**

  /* Disable the OPAMP digital output */

  DrvOP_OutputPinDisable();

10.3.10. DrvOP_OutputInverse

- **Prototype**

  unsigned int DrvOP_OutputInverse(uInv)

- **Description**

  The digital output op amp inverse control.

  Configure the register 0x41900[5]

- **Parameter**

  uInv [in]

  0 : Normal

  1 : Output Reverse

- **Include**

  Peripheral_lib/DrvOP.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

  /* Set OPAMP digital output inverse control */

  DrvOP_OutputInverse(1);

10.3.11. DrvOP_OutputWithCHPCK

- **Prototype**

  unsigned int DrvOP_OutputWithCHPCK(uCHPCK)

- **Description**

  OPO1/OPO2 with/without CHPCK multiplier selection.

  Configure the register 0x41900[6]

- **Parameter**

  uCHPCK [in]

  0 : No CHPCK multiplier, OPO1/OPO2 is equal to OPOD

  1 : With CHPCK multiplier, OPO1/OPO2 is OPOD multiply by CHPCK

- **Include**

  Peripheral_lib/DrvOP.h

- **Return Value**

  0: Operation successful

  Other: Incorrect argument

- **Example**

  /* Set the output with CHPCK */

  DrvADC_ClkEnable(0,0); // must enable ADC clock source first

  DrvOP_OutputWithCHPCK(1); //enable CHPCK multiplier

10.3.12. DrvOP_EnableInt

- **Prototype**

  void DrvOP_EnableInt (void)

- **Description**

  OPAMP Interrupt Enable

  Configure the register 0x4000C[16]=1b

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvOP.h

- **Return Value**

  None

- **Example**

  /* Enable OPAMP interrupt */

  DrvOP_EnableInt();

10.3.13. DrvOP_DisableInt

- **Prototype**

  void DrvOP_DisableInt (void)

- **Description**

  OPAMP Interrupt Disable

  Configure the register 0x4000C[16]=0b

- **Parameter**

  None

- **Include**

Peripheral_lib/DrvOP.h

- **Return Value**

None

- **Example**

/* Disable OPAMP interrupt */

DrvOP_DisableInt();

10.3.14. DrvOP_ReadIntFlag

- **Prototype**

unsigned int DrvOP_ReadIntFlag (void)

- **Description**

Read OPAMP interrupt flag.

Read the register 0x4000C[0]

- **Parameter**

None

- **Include**

Peripheral_lib/DrvOP.h

- **Return Value**

0 : Interrupt flag is0

1 : Interrupt flag is1

>1: Invalid return value

- **Example**

/* Read OPAMP interrupt flag */

unsigned char flag；flag=DrvOP_ReadIntFlag();

10.3.15. DrvOP_ClearIntFlag

- **Prototype**

void DrvOP_ClearIntFlag (void)

- **Description**

Clear OPAMP interrupt flag.

Clear the register 0x4000C[0] =0b

- **Parameter**

None

- **Include**

Peripheral_lib/DrvOP.h

- **Return Value**

  None

- **Example**

  /* Clear OPAMP interrupt flag */

  DrvOP_ClearIntFlag();

10.3.16. DrvOP_Feedback

- **Prototype**

  unsigned int DrvOP_Feedback(uFeedback)

- **Description**

  OPAMP feedback or sample capacitor connection settings

  Configure the register 0x41900[4]

- **Parameter**

  uFeedback [in]

  0 : The capacitor is used as integrated capacitor. The bottom plate connects to OPOI

  1 : The capacitor is used as sample capacitor. The bottom plate connects to VSSA

- **Include**

  Peripheral_lib/DrvOP.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

  /* OPAMP feedback capacitor connected to OPOI */

  DrvOP_Feedback(0);

10.3.17. DrvOP_OPDEN

- **Prototype**

  unsigned char DrvOP_OPDEN(uOPDEN)

- **Description**

  OPAMP digital output function control

  Configure the register 0x41900[2]

- **Parameter**

  uOPDEN [in]：OPAMP digital output function selection. Input range: 0~1

  0 : Disable

  1 : Enable

- **Include**

  Peripheral_lib/DrvOP.h

- **Return Value**

  0: Operation successful

  1 : Incorrect argument

- **Example**

```
/* Enable OPAMP digital output function, set OPDEN=1b */
DrvOP_OPDEN(1);
```

11. PMU Driver

11.1. Introduction

The following functions are included in Power Manager Section.

Item	Functions	Description
01	DrvPMU_VDDA_Voltage	VDDA voltage selection
02	DrvPMU_VDDA_LDO_Ctrl	VDDA LDO enable control
03	DrvPMU_BandgapEnable	Band gap enable control
04	DrvPMU_BandgapDisable	Band gap disable control
05	DrvPMU_ChargePumpEnableDrvPMU_ChargePumpEnable	Charge Pump enable
06	DrvPMU_ChargePumpDisable	Charge Pump disable
07	DrvPMU_REFO_Enable	Reference buffer enable
08	DrvPMU_REFO_Disable	Reference buffer disable
09	DrvPMU_AnalogGround	ADC analog ground source
10	DrvPMU_LDO_LowPower	VDD LDO low power

11.2. Type Definition

E_VDDA_OUTPUT_VOLTAGE

Enumeration identifier	Value	Description
E_VDDA2_4	0x0	Select the VDDA voltage of 2.4V
E_VDDA2_7	0x1	Select the VDDA voltage of 2.7V
E_VDDA3_0	0x2	Select the VDDA voltage of 3.0V
E_VDDA3_3	0x3	Select the VDDA voltage of 3.3V

E_VDDA_LDO_ENABLE_CONTROL

Enumeration identifier	Value	Description
E_HighZ	0x0	Select the VDDA voltage of 0V
E_ VDD3V	0x1	Select the VDDA voltage of 3V
E_ PullDown	0x2	Select the VDDA voltage of 0V
E_LDO	0x3	Select the VDDA voltage of 2.4~3.3V

11.3. Functions

11.3.1. DrvPMU_VDDA_Voltage

- **Prototype**

  unsigned int DrvPMU_VDDA_Voltage(E_VDDA_OUTPUT_VOLTAGE uVoltage)

- **Description**

  VDDA output voltage selection

  Configure the register 0x40400[19:18]

- **Parameter**

  uVoltage [in] : VDDA voltage selection, the input range is 0~3

  0 : 2.4V

  1 : 2.7V

  2 : 3.0V

  3 : 3.3V

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Select the VDDA voltage of 2.7V. */
DrvPMU_VDDA_Voltage(E_VDDA2_7);
```

11.3.2. DrvPMU_VDDA_LDO_Ctrl

- **Prototype**

  unsigned int DrvPMU_VDDA_LDO_Ctrl(E_VDDA_LDO_ENABLE_CONTROL uCtrl)

- **Description**

  VDDA LDO enable control.

  Configure the register 0x40400[17:16]

- **Parameter**

  uCtrl [in] :

  0 : High Z, VDDA=0

  1 : VDD3V,VDDA=VDD3V

  2 : Weak pull down,VDDA=0

  3 : LDO, VDDA output voltage regulation

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Select the VDDA LDO enable. VDDA=2.7V*/
DrvPMU_VDDA_LDO_Ctrl(E_LDO);
DrvPMU_VDDA_Voltage(E_VDDA2_7);
```

11.3.3. DrvPMU_BandgapEnable

- **Prototype**

  void DrvPMU_BandgapEnable(void)

- **Description**

  Bandgap enable control.

  Configure the register 0x40400[4]=1b

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Value**

  None

- **Example**

```
/* Enable bandgap. */
DrvPMU_BandgapEnable();
```

11.3.4. DrvPMU_BandgapDisable

- **Prototype**

  void DrvPMU_BandgapDisable(void)

- **Description**

  Bandgap disable control.

  Configure the register 0x40400[4]=0b

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Value**

  None

- **Example**

```
/* Disable bandgap. */
DrvPMU_BandgapDisable();
```

11.3.5. DrvPMU_ChargePumpEnable

- **Prototype**

  void DrvPMU_ChargePumpEnable(void)

- **Description**

  ChargePump enable control.

  Note : Enable the ADC clock source before use it. Becauce ChargePump boost-up circuit need to use ADC clock as clock source.

  Configure the register 0x40400[2]=1

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Value**

  None

- **Example**

```
/* Enable charge pump */
DrvADC_ClkEnable(0,0) ; //enable ADC clock source
DrvPMU_ChargePumpEnable(); // Enable charge pump
```

11.3.6. DrvPMU_ChargePumpDisable

- **Prototype**

  void DrvPMU_ ChargePumpDisable (void)

- **Description**

  Charge pump disable control.

  Configure the register 0x40400[2]=0

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Value**

  None

- **Example**

```
/* Disable charge pump. */
DrvPMU_ChargePumpDisable();
```

11.3.7. DrvPMU_REFO_Enable

- **Prototype**

  void DrvPMU_REFO_Enable(void)

- **Description**

  Reference buffer enable control.

  The output voltage is 1.2V. Need to enable the Bandgap.

  Configure the register 0x40400[1] =1b

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Value**

  None

- **Example**

```
/* Enable REFO. */
DrvPMU_BandgapEnable() ; // enable the Bandgap
DrvPMU_REFO_Enable(); //Enable REFO
```

11.3.8. DrvPMU_REFO_Disable

- **Prototype**

  void DrvPMU_REFO_Disable(void)

- **Description**

  Reference buffer disable control.

  Configure the register 0x40400[1] =0b

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Value**

  None

- **Example**

```
/* Disable REFO. */
DrvPMU_REFO_Disable();
```

11.3.9. DrvPMU_AnalogGround

- **Prototype**

  unsigned int DrvPMU_AnalogGround(uAG)

- **Description**

  ADC analog ground source selection.

  Configure the register 0x40400[3]

- **Parameter**

  uAG [in] :

  0 : External

  1 : Enable buffer and use internal source(need to work with ADC)

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Select the analog ground of external. */
DrvPMU_AnalogGround(0);
```

11.3.10. DrvPMU_LDO_LowPower

- **Prototype**

  unsigned int DrvPMU_LDO_LowPower(uLP)

- **Description**

  VDD LDO with low power control.

  Configure the register 0x40400[0]

- **Parameter**

  uLP [in]

  0 : Normal(form sleep mode make up needs to set 0)

  1 : Low power

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Enable the LDO of low power. */
DrvPMU_LDO_LowPower(1);
```

12. 8-bit Resistance Ladder Driver(DAC)

12.1. Introduction

The following functions are included in 8-bit resistance ladder (the file:DrvDAC.h)Manager Section.

Item	Functions	Description
01	DrvDAC_Open	Open the DAC
02	DrvDAC_Close	Close the DAC
03	DrvDAC_Enable	Enable the DAC
04	DrvDAC_Disable	Disable the DAC
05	DrvDAC_EnableOutput	DAC output enable
06	DrvDAC_DisableOutput	DAC output disable
07	DrvDAC_Pinput	DAC positive reference input selection.
08	DrvDAC_NInput	DAC negative reference input selection
09	DrvDAC_DABIT	DAO[7:0] buffer from MSB to LSB
10	DrvDAC_SetoutputIO	Set output pin of DAC

12.2. Type Definition

E_DAC_INPUT

Enumeration identifier	Value	Description
E_DAC_PVDD3V	0x0	Signal input for positive
E_ DAC_PVDDA	0x1	Signal input for positive
E_ DAC_PREFO_I	0x2	Signal input for positive
E_DAC_POPO	0x3	Signal input for positive
E_ DAC_PAIO6	0x4	Signal input for positive
E_ DAC_NVSSA	0x0	Signal input for negative
E_ DAC_NREFO_I	0x1	Signal input for negative
E_ DAC_NOPO	0x2	Signal input for negative
E_ DAC_NAIO7	0x3	Signal input for negative

12.3. Functions

12.3.1. DrvDAC_Open

- **Prototype**

  unsigned int DrvDAC_Open(E_DAC_INPUT uPinput ,E_DAC_INPUT uNinput, uDAO)

- **Description**

  Enable the DAC，select positive and negative reference input, set the initial scale value of DAC output voltage. Configure the register 0x41700[5:0],0x41704[7:0].

- **Parameter**

  uPinput [in] : DAC positive reference input selection.

  0 : VDD3V

  1 : VDDA

  2 : REFO_I

  3 : OPO

  4：AIO6 （selected through register 0x41104[28]）

  uNinput [in] : DAC negative reference input selection.

  0 : VSSA

  1 : REFO_I

  2 : OPO

  3 : AIO7

  uDAO [in] : The scale value of output voltage DAO/255

  DAO[7:0]buffer from MSB to LSB, the input range is 0~255

- **Include**

  Peripheral_lib/DrvDAC.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Enable DAC, Select positive reference input=AIO6, negative reference input=VSSA ,DAO=5*/
DrvDAC_Open(E_DAC_AIO6, E_DAC_VSSA ,5 );
```

12.3.2. DrvDAC_Close

- **Prototype**

  void DrvDAC_Close(void)

- **Description**

  Close the DAC(ENDA and DAOE off)

  Configure the register 0x41700[1:0]=00b

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvDAC.h

- **Return Value**

  None

- **Example**

```
/* Close DAC */
DrvDAC_Close();
```

12.3.3. DrvDAC_Enable

- **Prototype**

  void DrvDAC_Enable(void)

- **Description**

  Enable DAC (ENDA enable)

  Configure the register 0x41700[0]=1b

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvDAC.h

- **Return Value**

  None

- **Example**

```
/* Close DAC */
DrvDAC_Enable();
```

12.3.4. DrvDAC_Disable

- **Prototype**

void DrvDAC_Disable(void)

- **Description**

Close the DAC (ENDA disable)

Configure the register 0x41700[0]=0b

- **Parameter**

None

- **Include**

Peripheral_lib/DrvDAC.h

- **Return Value**

None

- **Example**

```
/* Close DAC */
DrvDAC_Disable();
```

12.3.5. DrvDAC_EnableOutput

- **Prototype**

void DrvDAC_EnableOutput(void)

- **Description**

DAC output enable (DAOE enable)

Configure the register 0x41700[1]=1b

- **Parameter**

None

- **Include**

Peripheral_lib/DrvDAC.h

- **Return Value**

None

- **Example**

```
/* DAC output enable */
DrvDAC_EnableOutput();
```

12.3.6. DrvDAC_DisableOutput

- **Prototype**

void DrvDAC_DisableOutput (void)

- **Description**

DAC output disable(DAOE disable)

Configure the register 0x41700[1]=0b

- **Parameter**

None

- **Include**

Peripheral_lib/DrvDAC.h

- **Return Value**

None

- **Example**

```
/* DAC output disable */
DrvDAC_DisableOutput();
```

12.3.7. DrvDAC_Pinput

- **Prototype**

unsigned int DrvDAC_Pinput(E_DAC_INPUT uPinput)

- **Description**

DAC positive reference input selection.

Configure the register 0x41700[5:4] and ADC register 0x41104[28]

- **Parameter**

uPinput [in] : DAC positive reference input selection.

0 : VDD3V

1 : VDDA

2 : REFO_I

3 : OPO

4 : AIO6

- **Include**

Peripheral_lib/DrvDAC.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Select DAC positive reference input from VDD3V. */
DrvDAC_Pinput(0);
```

12.3.8. DrvDAC_NInput

- **Prototype**

  unsigned int DrvDAC_Ninput(E_DAC_INPUT uNinput)

- **Description**

  DAC negative reference input selection.

  Configure the register 0x41700[3:2]

- **Parameter**

  uNinput [in] : DAC negative reference input selection.

  0 : VSSA

  1 : REFO_I

  2 : OPO

  3 : AIO7

- **Include**

  Peripheral_lib/DrvDAC.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

  /* Select DAC negative reference input from VSSA. */

  DrvDAC_Ninput(E_DAC_VSSA);

12.3.9. DrvDAC_DABIT

- **Prototype**

  unsigned int DrvDAC_DABIT(uDABIT)

- **Description**

  DAO[7:0] : the scale of output voltage :DAO/255

  Configure the register 0x41704[7:0]

- **Parameter**

  uDABIT [in]

  the scale of output voltage :uDABIT/255, the input range is 0~255

- **Include**

  Peripheral_lib/DrvDAC.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

  /* DAO [7:0] =5 */

  DrvDAC_DABIT(5);

12.3.10. DrvDAC_SetoutputIO

- **Prototype**

  unsigned int DrvDAC_SetoutputIO(unsigned int uio)

- **Description**

  DAO Output to PT3.1 enable

  Configure the register 0x40828[17]

- **Parameter**

  uio [in] :

  0 : disable DAO Output to PT3.1

  1 : enable DAO Output to PT3.1

- **Include**

  Peripheral_lib/DrvDAC.h

- **Return Value**

  0: Operation successful

  1 : Incorrect argument

- **Example**

```
/* enable DAO Output to PT3.1*/
DrvDAC_SetoutputIO(1);
```

13. RTC Driver

13.1. Introduction

The following functions are included in RTC Manager Section.

Item	Functions	Description
01	DrvRTC_SetFrequencyCompensation	Set Frequency Compensation Data
02	DrvRTC_WriteEnable	Access Password to KEY to make access other register enable
03	DrvRTC_WriteDisable	Clear the RTC KEY to make the RTC register can not write
04	DrvRTC_ClockSourceDrvRTC_Clock Source	Select clock source from the external or internal.
05	DrvRTC_AlarmEnable	Enable the TAEn.
06	DrvRTC_AlarmDisable	Disable the TAEn.
07	DrvRTC_PeriodicTimeEnable	Enable PTEn and set periodic timer frequency of RTC.
08	DrvRTC_PeriodicTimeDisable	Disable the PTEn.
09	DrvRTC_Enable	Enable the RTCEn.
10	DrvRTC_Disable	Disable the RTCEn.
11	DrvRTC_HourFormat	Set the clock for 12 or 24hour
12	DrvRTC_ReadState	Read the RTC state
13	DrvRTC_ClearState	Clear the RTC state
14	DrvRTC_EnableInt	RTC Interrupt Enable
15	DrvRTC_DisableInt	RTC Interrupt disable
16	DrvRTC_ReadIntFlag	Read the RTC Interrupt flag.
17	DrvRTC_ClearIntFlag	Clear the RTC Interrupt flag.
18	DrvRTC_Write	Set current date/time or alarm date/time to RTC
19	DrvRTC_Read	Read current date/time or alarm date/time from RTC setting
20	DrvRTC_ClkConfig	Set RTC clock source
21	DrvRTC_EnableWUEn	Enable RTC WUEn
22	DrvRTC_DisableWUEn	Disable RTC WUEn

13.2. Type Definition

E_DRVRTC_CLOCK_SOURCE

Enumeration Identifier	Value	Description
E_EXT_CK	0	RTC clock source from external low speed crystal source
E_INT_CK	1	RTC clock source from internal low speed crystal source

E_DRVRTC_ TICK

Enumeration Identifier	Value	Description
E_DRVRTC_1_128_SEC	0	Set tick period 1/128 tick per second
E_DRVRTC_1_64_SEC	1	Set tick period 1/64 tick per second
E_DRVRTC_1_32_SEC	2	Set tick period 1/32 tick per second
E_DRVRTC_1_16_SEC	3	Set tick period 1/16 tick per second
E_DRVRTC_1_8_SEC	4	Set tick period 1/8 tick per second
E_DRVRTC_1_4_SEC	5	Set tick period 1/4 tick per second
E_DRVRTC_1_2_SEC	6	Set tick period 1/2 tick per second
E_DRVRTC_1_SEC	7	Set tick period 1 tick per second

E_DRVRTC_HOUR_FORMAT

Enumeration Identifier	Value	Description
E_DRVRTC_HOUR_12	1	The hour format by 12
E_DRVRTC_HOUR_24	0	The hour format by 24

E_DRVRTC_TIME_SELECT

Enumeration Identifier	Value	Description
DRVRTC_CURRENT_TIME	0	Select current time option
DRVRTC_ALARM_TIME	1	Select alarm time option

E_DRVRTC_FLAG

Enumeration Identifier	Value	Description
E_DRVRTC_ALARM_FLAG	0	alarm flag
E_DRVRTC_PERIODIC_FLAG	1	periodic timer flag
E_DRVRTC_CLEAR_ALL	2	clear alarm flag and periodic timer flag

13.3. Functions

Note : It is necessary to enable RTC clock and write <0110> in the RTKEY (register 0x41A00[23:20]) before writing data into the RTC register

13.3.1. DrvRTC_SetFrequencyCompensation

- **Prototype**

  unsigned int DrvRTC_SetFrequencyCompensation(

  unsigned int uFrequencyCom);

- **Description**

  Set Frequency Compensation Data

  Configure the register 0x41A04[22:16]

- **Parameters**

  uFrequencyCom [in] : specified RTC clock frequency compensation, It could be 0~0x7f

  0111111 : +126 ppm

  0111110 : +124 ppm

  |:

  0000001 : +2 ppm

  0000000 : +0 ppm

  1000000 : - 0 ppm

  1000001 : - 2 ppm

  | :

  1111110 : -124 ppm

  1111111 : -126 ppm

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Compensation deviation -2 PPM */
DrvRTC_SetFrequencyCompensation(0x41);
```

13.3.2. DrvRTC_WriteEnable

- **Prototype**

void DrvRTC_WriteEnable(void);

- **Description**

Access Password to KEY to make access other register enable.

Configure the register 0x41a00[23:20] =0110b

- **Parameters**

None

- **Include**

Peripheral_lib/DrvRTC.h

- **Return Value**

None.

- **Example**

```
/* Unlock RTC register, the RTC registers can be written */
DrvRTC_WriteEnable();
```

13.3.3. DrvRTC_WriteDisable

- **Prototype**

void DrvRTC_WriteDisable(void);

- **Description**

Clear the RTC KEY to make the RTC register can not write

Configure the register 0x41A00[23:20]=0000b

- **Parameters**

None

- **Include**

Peripheral_lib/DrvRTC.h

- **Return Value**

None.

- **Example**

```
/* Lock RTC register, the RTC registers can not write */
DrvRTC_WriteDisable();
```

13.3.4. DrvRTC_ClockSource

- **Prototype**

```
unsigned int DrvRTC_ClockSource(
E_DRVRTC_CLOCK_SOURCE uClockSource
);
```

- **Description**

  Select clock source from the external or internal.

  Configure the register 0x41A00[1]

- **Parameters**

  uClockSource [in]

  0 : E_EXT_CK

  1 : E_INT_CK

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Value**

  CKS has a foolproof protection

  0 : E_EXT_CK

  1 : E_INT_CK

- **Example**

```
/* Select the RTC Clock from external. */
DrvRTC_ClockSource(E_EXT_CK);
```

13.3.5. DrvRTC_AlarmEnable

- **Prototype**

  void DrvRTC_AlarmEnable (void);

- **Description**

  Enable the RTC Alarm function.

  Configure the register 0x41A00[3]=1b

- **Parameters**

  none

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Value**

  none

- **Example**

```
/* Enable RTC alarm */
DrvRTC_AlarmEnable();
```

13.3.6. DrvRTC_AlarmDisable

- **Prototype**

void DrvRTC_AlarmDisable (void);

- **Description**

Disable the RTC Alarm function.

Configure the register 0x41A00[3]=0b

- **Parameters**

none

- **Include**

Peripheral_lib/DrvRTC.h

- **Return Value**

none

- **Example**

```
/* Disable Alarm*/
DrvRTC_AlarmDisable();
```

13.3.7. DrvRTC_PeriodicTimeEnable

- **Prototype**

unsigned int DrvRTC_PeriodicTimeEnable (E_DRVRTC_TICK uPeriodicTimer);

- **Description**

Set periodic timer frequency of RTC.

Configure the register 0x41A04[2:0], 0x41A00[5]=1, 0x41A00[4]=1.

- **Parameters**

uPeriodicTimer[in]

0: 1/128Second

1: 1/64Second

2: 1/32Second

3: 1/16Second

4: 1/8Second

5: 1/4Second

6: 1/2Second

7: 1Second

- **Include**

Peripheral_lib/DrvRTC.h

- **Return Value**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Enable RTC alarm and select 1/16 second */
```

```
DrvRTC_PeriodicTimeEnable(3);
```

13.3.8. DrvRTC_PeriodicTimeDisable

- **Prototype**

  void DrvRTC_PeriodicTimeDisable (void);

- **Description**

  Disable the periodic time function.

  Configure the register 0x41A00[5]=0 / 0x41A00[4]=0

- **Parameters**

  none

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Value**

  none

- **Example**

```
/* Disable the PTEn*/
DrvRTC_PeriodicTimeDisable();
```

13.3.9. DrvRTC_Enable

- **Prototype**

  void DrvRTC_Enable (void);

- **Description**

  Enable the RTC function.

  Configure the register 0x41A00[0]=1b

- **Parameters**

  none

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Value**

  none

- **Example**

```
/* Enable the RTC */
DrvRTC_Enable();
```

13.3.10. DrvRTC_Disable

- **Prototype**

  void DrvRTC_Disable (void);

- **Description**

  Disable the RTC function.

  Configure the register 0x41A00[0]=0b

- **Parameters**

  none

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Value**

  none

- **Example**

```
/* Disable the RTC */
DrvRTC_Disable();
```

13.3.11. DrvRTC_HourFormat

- **Prototype**

  unsigned int DrvRTC_HourFormat(E_DRVRTC_HOUR_FORMAT uHourFormat);

- **Description**

  Set the clock for 12 or 24hour

  Configure the register 0x41A00[2]

- **Parameters**

  0 : The hour format by 24

  1 : The hour format by 12

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Set 12-hour */
DrvRTC_HourFormat(1);
```

13.3.12. DrvRTC_ReadState

- **Prototype**

  unsigned int DrvRTC_ReadState(void);

- **Description**

  Read the RTC state

  Configure the register 0x41A00[19:16]

- **Parameters**

  none

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Value**

  Return 0x0~0xf

  Bit 0 : RTC Alarm Flag

  Bit 1 : RTC Wakeup Flag

  Bit 2 : RTC Aeriodic Timer Flag

  Bit 3 : RTC Leap Year Flag

- **Example**

  /* Check the RTC of alarm flag */

 Sample code 1 :

```
 If (DrvRTC_ReadState()&0x1)
            //RTC alarm triggered
   else
            // RTC Wakeup triggered
```

Sample code 2 :

```
flag = DrvRTC_ReadState();
```

13.3.13. DrvRTC_ClearState

- **Prototype**

  unsigned int DrvRTC_ClearState(E_DRVRTC_FLAG uFlag);

- **Description**

  Clear the RTC state

  Clear the register 0x41A00[19:16]

- **Parameters**

  0 : clear alarm flag

  1 : clear periodic timer flag

  2: clear alarm flag and periodic timer flag

- **Include**

Peripheral_lib/DrvRTC.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Clear TAF/ PTF flag */
DrvRTC_ClearState(2);
```

13.3.14. DrvRTC_EnableInt

- **Prototype**

  void DrvRTC_EnableInt(void)

- **Description**

  RTC Interrupt Enable. Need to clear the PTF after response to interrupt,then it can response to interrupt normally next time.

  Configure the register 0x40004[21]=1b

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Value**

  None

- **Example**

```
/* Enable RTC interrupt */
DrvRTC_EnableInt();
```

13.3.15. DrvRTC_DisableInt

- **Prototype**

  void DrvRTC_DisableInt(void)

- **Description**

  RTC Interrupt disable

  Configure the register 0x40004[21]=0b

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Value**

  None

- **Example**

  /* Disable RTC interrupt */

  DrvRTC_DisableInt();

13.3.16. DrvRTC_ReadIntFlag

- **Prototype**

  unsigned int DrvRTC_ReadIntFlag(void)

- **Description**

  Read the RTC Interrupt flag.

  Read the register 0x40004[5]

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Value**

  0 : Normal

  1 :Interrupted

- **Example**

  /* Read the RTC Interrupt flag */

  unsigned char flag；flag=DrvRTC_ReadIntFlag();

13.3.17. DrvRTC_ClearIntFlag

- **Prototype**

  void DrvRTC_ClearIntFlag(void)

- **Description**

  Clear the RTC Interrupt flag.

  Configure the register 0x40004[5]=0b

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Value**

  None

● **Example**

/* Clear the RTC Interrupt flag */

DrvRTC_ClearIntFlag();

13.3.18. DrvRTC_Write

● **Prototype**

Unsigned int DrvRTC_Write (

E_DRVRTC_TIME_SELECT eTime, S_DRVRTC_TIME_DATA_T *sPt);

● **Description**

Set current date/time or alarm date/time to RTC

Configure the register 0x41A08/0x41A0C/0x41A10/0x41A14/0x41A18/0x41A1C

● **Parameter**

eTime [in] : Specify the current/alarm time to be written.

0 : Current time

1 : Alarm time

*sPt [in] : Specify the data to write to RTC. It includes:

u8cClockDisplay	DRVRTC_CLOCK_12(00:00~11:59) / DRVRTC_CLOCK_24(00:00~23:59)
u8cAmPm	DRVRTC_AM / DRVRTC_PM
u32cSecond	Second value
u32cMinute	Minute value
u32cHour	Hour value
u32cDayOfWeek	Day of week
u32cDay	Day value
u32cMonth	Month value
u32Year	Year value

● **Include**

Peripheral_lib/DrvRTC.h

● **Return Value**

0: Operation successful

Other : Incorrect argument

● **Example**

/* Condition: Update current the second of time to zero */

S_DRVRTC_TIME_DATA_T sCurTime;

DrvRTC_Read(DRVRTC_ALARM_TIME, &sCurTime);

sCurTime.u32cSecond = 0;

DrvRTC_Write(DRVRTC_ALARM_TIME, &sCurTime);

13.3.19. DrvRTC_Read

- **Prototype**

  unsigned int DrvRTC_Read (

  E_DRVRTC_TIME_SELECT eTime,

  S_DRVRTC_TIME_DATA_T *sPt);

- **Description**

  Read current date/time or alarm date/time from RTC setting

  Configure the register 0x41A08/0x41A0C/0x41A10/0x41A14/0x41A18/0x41A1C

- **Parameter**

  eTime [in] : Specify the current/alarm time to be written.

  0 : Current time

  1 : Alarm time

  *sPt [in] : Specify the data to write to RTC. It includes:

  u8cClockDisplay DRVRTC_CLOCK_12 / DRVRTC_CLOCK_24

  u8cAmPm DRVRTC_AM / DRVRTC_PM

  u32cSecond Second value

  u32cMinute Minute value

  u32cHour Hour value

  u32cDayOfWeek Day of week

  u32cDay Day value

  u32cMonth Month value

  u32Year Year value

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Condition: You want to get current RTC calendar and time */
S_DRVRTC_TIME_DATA_T sCurTime;
DrvRTC_Read(DRVRTC_CURRENT_TIME, &sCurTime);
```

13.3.20. DrvRTC_ClkConfig

- **Prototype**

  unsigned char DrvRTC_ClkConfig(unsigned char uclken)

- **Description**

  Configure RTC clock source

  Configure the register 0x40308[23]

- **Parameter**

  uClockSource [in]

  0 : disable the RTC clock source

  1 : Enable the RTC clock source

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Value**

  1: Operation successful

  0 : Incorrect argument

- **Example**

```
/* Enable the RTC clock source */
DrvRTC_ClkConfig(1);
```

13.3.21. DrvRTC_EnableWUEn

- **Prototype**

  void DrvRTC_EnableWUEn (void)

- **Description**

  Enable RTC WUEn, WUEn=1b

  Configure the register 0x41A00[4]=1

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Value**

  None

- **Example**

```
/* Enable RTC WUEn */
DrvRTC_EnableWUEn();
```

13.3.22. DrvRTC_DisableWUEn

- **Prototype**

  void DrvRTC_DisableWUEn (void)

- **Description**

Disable RTC WUEn, WUEn=0b

Configure the register 0x41A00[4]=0

- **Parameter**

None

- **Include**

Peripheral_lib/DrvRTC.h

- **Return Value**

None

- **Example**

```
/* Disable RTC WUEn */
DrvRTC_DisableWUEn();
```

14. I2C Driver

14.1. Introduction

The following functions are included in I2C Manager Section.

Item	Functions	Description
01	DrvI2C_Open	Enable the I2C and configure the I2C bus clock
02	DrvI2C_Close	Disable the I2C
03	DrvI2C_SlaveSet	Enable slave mode · set address · choose whether to enable GC
04	DrvI2C_SetIOPin	Select IO port as IIC port
05	DrvI2C_WriteData	To set a byte of data to be sent.
06	DrvI2C_Write3ByteData	To set a 3byte of data to be sent.
07	DrvI2C_ReadData	Read the data form receiver data buffer.
08	DrvI2C_Ctrl	To set I2C control bit include STA, STO, AA, SI in control register.
09	DrvI2C_EnableInt	Enable the I2C Interrupt
10	DrvI2C_DisableInt	Disable the I2C Interrupt
11	DrvI2C_ReadIntFlag	Read the I2C Interrupt flag.
12	DrvI2C_ClearIntFlag	Clear the I2C Interrupt flag.
13	DrvI2C_ClearEIRQ	Clear the EIRQ
14	DrvI2C_ClearIRQ	Clear the IRQ.
15	DrvI2C_GetStatusFlag	Take status flags
16	DrvI2C_TimeOutEnable	Enable TimeOut · set clock pre scale and time out limit
17	DrvI2C_TimeOutDisable	Disable the Timeout
18	DrvI2C_STSP	Generate the START or STOP singal from IIC bus
19	DrvI2C_MGetACK	Check the ACK from slaver during the set time
20	DrvI2C_DisableIOPin	Disable IIC communication function of the IO port
21	DrvI2C_EnableSEn	Enable I2C Slave mode function
22	DrvI2C_DisableSEn	Disable I2C Slave mode function
23	DrvI2C_EnableI2CEn	Enable I2C function

14.2. Type Definition

E_DRVI2C_Status

Enumeration Identifier	Value	Description
E_DRVI2C_ARBITRATION_FLAG	0	Arbitration Lost Flag
E_DRVI2C_GENERAL_CALL_FLAG	1	General Call Flag
E_DRVI2C_ACKNOWLEDGE_FLAG	2	Acknowledge Flag
E_DRVI2C_DATA_FIELD_FLAG	3	Data Field Flag
E_DRVI2C_RW_STATE_FLAG	4	Read/Write State Flag
E_DRVI2C_RS_FLAG	5	Received Stop/Repeat-Start Flag
E_DRVI2C_SLAVE_ACTIVE_FLAG	6	Slave Mode Active Flag
E_DRVI2C_MASTER_ACTIVE_FLAG	7	Master Mode Active Flag

E_DRVI2C_TIMEOUT_PRESCALE

Enumeration Identifier	Value	Description
E_DRVI2C_I2CLK_DIV_1	0	I2C CLK/1
E_DRVI2C_I2CLK_DIV_2	1	I2C CLK/2
E_DRVI2C_I2CLK_DIV_4	2	I2C CLK/4
E_DRVI2C_I2CLK_DIV_8	3	I2C CLK/8
E_DRVI2C_I2CLK_DIV_16	4	I2C CLK/16
E_DRVI2C_I2CLK_DIV_32	5	I2C CLK/32
E_DRVI2C_I2CLK_DIV_64	6	I2C CLK/64
E_DRVI2C_I2CLK_DIV_128	7	I2C CLK/128

E_DRVI2C_TIMEOUT_LIMIT

Enumeration Identifier	Value	Description
E_DRVI2C_CLKPSX1	0	1 * CLKps Cycle
E_DRVI2C_CLKPSX2	1	2 * CLKps Cycle
E_DRVI2C_CLKPSX3	2	3 * CLKps Cycle
E_DRVI2C_CLKPSX4	3	4 * CLKps Cycle
E_DRVI2C_CLKPSX5	4	5 * CLKps Cycle
E_DRVI2C_CLKPSX6	5	6 * CLKps Cycle
E_DRVI2C_CLKPSX7	6	7 * CLKps Cycle
E_DRVI2C_CLKPSX8	7	8 * CLKps Cycle
E_DRVI2C_CLKPSX9	8	9 * CLKps Cycle
E_DRVI2C_CLKPSX10	9	10 * CLKps Cycle
E_DRVI2C_CLKPSX11	10	11 * CLKps Cycle
E_DRVI2C_CLKPSX12	11	12 * CLKps Cycle
E_DRVI2C_CLKPSX13	12	13 * CLKps Cycle
E_DRVI2C_CLKPSX14	13	14 * CLKps Cycle
E_DRVI2C_CLKPSX15	14	15 * CLKps Cycle
E_DRVI2C_CLKPSX16	15	16 * CLKps Cycle

E_DRVI2C_INTERRUPT

Enumeration Identifier	Value	Description
E_DRVI2C_INT	1	I2C Interrupt enable
E_DRVI2C_ERROR_INT	2	I2C error Interrupt enable
E_DRVI2C_INT_ALL	3	enable I2C interrupt and error interrupt

E_DRVI2C_SLAVE_BIT

Enumeration Identifier	Value	Description
E_DRVI2C_SLAVE_7BIT	0	Slave 7bit address mode
E_DRVI2C_SLAVE_10BIT	1	Slave 10bit address mode

14.3. Functions

Note：Configure the IIC register after enable IIC

14.3.1. DrvI2C_Open

- **Prototype**

  unsigned int DrvI2C_Open (uint32_t u32CRG);

- **Description**

  Enable the I2C and configure the I2C bus clock

  Configure the register 0x41000[0]=1, 0x41008[23:16]

- **Parameters**

  u32CRG [in] : Set CRG value. It could be 0~0xff.

  Data Baud Rate = (I2CLK/(4*(CRG+1)))

- **Include**

  Peripheral_lib/DrvI2C.h

- **Return Value**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Enable I2C and set CRG value 100 */
DrvI2C_Open(100);
```

14.3.2. DrvI2C_Close

- **Prototype**

  void DrvI2C_Close (void);

- **Description**

  Disable the I2C.

  Configure the register 0x41000[0]=0

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvI2C.h

- **Return Value**

  None.

● **Example**

/* Close the I2C */

DrvI2C_Close();

14.3.3. DrvI2C_SlaveSet

● **Prototype**

unsigned int DrvI2C_SlaveSet(

uint32_t uSlaveAddr,

E_DRVI2C_SLAVE_BIT uAddrBit,

uint8_t uSlave3Byte,

uint8_t GC_Flag);

● **Description**

Enable slave mode, and set the location address, and choose whether to enable GC

Configure the register 0x41004[7] / 0x41004[5] / 0x41000[2] / 0x4100C[7:0]

● **Parameters**

uSlaveAddr : slaver address

7bit : 0~0x7f

10bit: 0~0x3ff

uAddrBit : slaver address mode

0: Slave 7bit address mode

1: Slave 10bit address mode

uSlave3Byte: Slave 3 Byte Data Mode Enable control

0: Normal

1:Slave 3byte Data transfer

GC_Flag : general call flag

0: Normal

1: Enable general call

● **Include**

Peripheral_lib/DrvI2C.h

● **Return Value**

1: Operation successful

Other : Incorrect argument

● **Example**

/* Enable Slave mode, position setting 0x30*/

DrvI2C_SlaveSet(0x30,0,0,0);

14.3.4. DrvI2C_SetIOPin

- **Prototype**

  unsigned char DrvI2C_SetIOPin(unsigned int upin)

- **Description**

  Set IO port of I2C

  Configure the register 0x40844[19:16]

- **Parameters**

  upin[in] : select IO port as I2C port

  0 SCL=PT1.0;SDA=PT1.1

  1 SCL=PT1.2;SDA=PT1.3

  2 SCL=PT1.4;SDA=PT1.5

  3 SCL=PT1.6;SDA=PT1.7

  4 SCL=PT2.0;SDA=PT2.1

  5 SCL=PT2.2;SDA=PT2.3

  6 SCL=PT2.4;SDA=PT2.5

  7 SCL=PT2.6;SDA=PT2.7

- **Include**

  Peripheral_lib/DrvI2C.h

- **Return Value**

  None

- **Example**

```
/* specify PT2.0 and PT2.1 */
DrvI2C_SetIOPin(4);
```

14.3.5. DrvI2C_WriteData

- **Prototype**

  void DrvI2C_WriteData(uint8_t uData);

- **Description**

  To set a byte of data to be sent.

  Configure the register 0x41014[7:0]

- **Parameters**

  uData [IN] : the data to be sent

  1 Byte data

- **Include**

  Peripheral_lib/DrvI2C.h

- **Return Value**

None

- **Example**

/* Set byte data 0x55 into transmitter data buffer register */

DrvI2C_WriteData(0x55);

14.3.6. DrvI2C_Write3ByteData

- **Prototype**

void DrvI2C_Write3ByteData(uint8_t uData1,uData2,uData3);

- **Description**

To set a 3byte of data to be sent.

Configure the register 0x41014

- **Parameters**

uData1, uData2, uData3 [IN] : Byte data. Input range is : 0~0xFF

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Value**

None

- **Example**

/* Set 3byte data 0x11 0x22 0x33 into transmitter dada buffer register */

DrvI2C_Write3ByteData(0x11,0x22,0x33);

14.3.7. DrvI2C_ReadData

- **Prototype**

unsigned char DrvI2C_ReadData(void);

- **Description**

Read the data form receiver data buffer.

Configure the register 0x41010[7:0]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Value**

1 Byte received

- **Example**

```
/* Read byte data from receiver data buffer */
unsigned int data; data=DrvI2C_ReadData();
```

14.3.8. DrvI2C_Ctrl

- **Prototype**

  void DrvI2C_Ctrl(uint8_t start, uint8_t stop, uint8_t intFlag, uint8_t ack);

- **Description**

  To set I2C control bit include STA, STO, AA, SI in control register.

  Configure the register 0x41004[3:0]

- **Parameters**

  start [in]:

  To set STA bit or not. (1: set, 0: don"t set). If the STA bit is set, a START or repeat START signal will be generated when I2C bus is free.

  stop [in]:

  To set STO bit or not. (1: set, 0: don"t set). If the STO bit is set, a STOP signal will be generated. When a STOP condition is detected, this bit will be cleared by hardward automatically.

  intFlag [in]:

  To clear SI flag (I2C interrupt flag). (1: clear, 0: don"t work)

  ack [in]:

  To enable AA bit (Assert Acknowledge control bit) or not. (1: enable, 0: disable)

- **Include**

  Peripheral_lib/DrvI2C.h

- **Return Value**

  Byte data

- **Example**

```
DrvI2C_Ctrl(0, 0, 1, 0); /* Set I2C SI bit to clear SI flag */
DrvI2C_Ctrl(1, 0, 0, 0); /* Set I2C STA bit to send START signal */
```

14.3.9. DrvI2C_EnableInt

- **Prototype**

  void DrvI2C_EnableInt(E_DRVI2C_INTERRUPT uINT)

- **Description**

  Enable the I2C Interrupt

  Configure the register 0x40000[21:20]

- **Parameter**

uINT[IN]

0 : I2C Interrupt enable

1 : I2C error Interrupt enable

2 : enable I2C interrupt and error interrupt

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Value**

None

- **Example**

```
/* Enable I2C interrupt */
DrvI2C_EnableInt(1);
```

14.3.10. DrvI2C_DisableInt

- **Prototype**

void DrvI2C_DisableInt(E_DRVI2C_INTERRUPT uINT)

- **Description**

Disable the I2C Interrupt

Configure the register 0x40000[21:20]

- **Parameter**

uINT[IN] :

0: I2C Interrupt disable

1 : I2C error Interrupt disable

2 : disable I2C interrupt and error interrupt

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Value**

None

- **Example**

```
/* Disable I2C interrupt */
DrvI2C_DisableInt(1);
```

14.3.11. DrvI2C_ReadIntFlag

- **Prototype**

E_DRVI2C_INTERRUPT DrvI2C_ReadIntFlag(void)

- **Description**

Read the I2C Interrupt flag.

Read the register 0x40000[5:4]

- **Parameter**

None

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Value**

0: no I2C IRQ

1 : I2C Interrupt flag is true

2 : I2C error Interrupt flag is true

3 : enable I2C interrupt and error interrupt of flag is true

- **Example**

```
/* Read the I2C Interrupt flag */
uint32_t temp;
temp=DrvI2C_ReadIntFlag();
```

14.3.12. DrvI2C_ClearIntFlag

- **Prototype**

void DrvRTC_ClearIntFlag(E_DRVI2C_INTERRUPT uINT)

- **Description**

Clear the RTC Interrupt flag.

Clear the register 0x40000[5:4]

- **Parameter**

uINT[IN] :

0 : Clear the I2C Interrupt flag

1 : Clear the I2C error Interrupt flag

2 : Clear the I2C interrupt and error flag

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Value**

None

- **Example**

```
/* Clear the I2C Interrupt flag */
DrvI2C_ClearIntFlag(0);
```

14.3.13. DrvI2C_ClearEIRQ

- **Prototype**

  void DrvI2C_ClearEIRQ(void)

- **Description**

  Clear the EIRQ.

  Note :The EIRQ flag can be clear after clear the TOPFLAG. The I2CEIF can be clear after clear the EIRQ flag. Write 0 to this bit to clear EIRQ.

  Configure the register 0x41004[4]=0

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvI2C.h

- **Return Value**

  None

- **Example**

  /* Clear the EIRQ */

  DrvI2C_ClearEIRQ();

14.3.14. DrvI2C_ClearIRQ

- **Prototype**

  void DrvI2C_ClearIRQ(void)

- **Description**

  Clear the IRQ.

  The IRQFlag will be set 1 when receive 9th clock, and SCL will be pull down until clear IRQFlag

  Configure the register 0x41004[1]=0

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvI2C.h

- **Return Value**

  None

- **Example**

  /* Clear the IRQ */

  DrvI2C_ClearIRQ();

14.3.15. DrvI2C_GetStatusFlag

- **Prototype**

  unsigned char DrvI2C_GetStatusFlag(void)

- **Description**

  Take status flags

  Read the register 0x41004[23:16]

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvI2C.h

- **Return Value**

  The meaning of each bit for the return value:

  Bit 0 : Arbitration Lost Flag

  Bit 1: General Call Flag

  Bit 2: Acknowledge Flag

  Bit 3: Data Field Flag

  Bit 4: Read/Write State Flag

  Bit 5: Received Stop/Repeat-Start Flag

  Bit 6: Slave Mode Active Flag

  Bit 7: Master Mode Active Flag

  ARB:uStatus=0

  0 : Normal

  1 : Arbitration Lost

  GC:uStatus=1

  0 : Normal

  1 : Currently General Call Operation

  A/NA:uStatus=2

  0 : No Ack has been transmitted or received.

  1 : Ack has been transmitted or received.

  DF:uStatus=3

  0 : Normal

  1 : I2C Data Byte has been transmitted or received.

  R/W:uStatus=4

  0 : Write Command has been transmitted or received.

  1 : Read Command has been transmitted or received.

  RX P/Sr: uStatus=5

  0 : Normal

1 : Stop/Repeat-Start has been transmitted or received.

SAct:uStatus=6

0 : Inactive

1 : Active

MAct: uStatus=7

0 : Inactive

1 : Active

- **Example**

/* Read Flags /

char flag; flag=DrvI2C_GetStatusFlag(2);

14.3.16. DrvI2C_TimeOutEnable

- **Prototype**

unsigned char DrvI2C_TimeOutEnable(

E_DRVI2C_TIMEOUT_PRESCALE uPreScale,

E_DRVI2C_TIMEOUT_LIMIT uTimeOutLimit

);

- **Description**

Enable TimeOut, and set the clock pre scale and time out limit

Configure the register 0x41000[1] , 0x41008[6:0]

- **Parameters**

uPreScale[in]:

0 I2C CLK/1

1 I2C CLK/2

2 I2C CLK/4

3 I2C CLK/8

4 I2C CLK/16

5 I2C CLK/32

6 I2C CLK/64

7 I2C CLK/128

uTimeOutLimit [in] :

0 1 * CLKps Cycle

1 2 * CLKps Cycle

2 3 * CLKps Cycle

3 4 * CLKps Cycle

4 5 * CLKps Cycle

5 6 * CLKps Cycle

6 7 * CLKps Cycle

7 8 * CLKps Cycle

8 9 * CLKps Cycle

9 10 * CLKps Cycle

10 11 * CLKps Cycle

11 12 * CLKps Cycle

12 13 * CLKps Cycle

13 14 * CLKps Cycle

14 15 * CLKps Cycle

15 16 * CLKps Cycle

- **Include**

  Peripheral_lib/DrvI2C.h

- **Return Value**

  0: Operation successful

  0xff: Incorrect argument

- **Example**

```
/*Enable TimeOut · set clock pre scale / 32 and time out limit=15 * CLKps Cycle */
DrvI2C_TimeOutEnable(5,14);
```

14.3.17. DrvI2C_TimeOutDisable

- **Prototype**

  void DrvI2C_TimeOutDisable(void)

- **Description**

  Disable the Timeout

  Configure the register 0x41000[1]=0

- **Parameter**

  none

- **Include**

  Peripheral_lib/DrvI2C.h

- **Return Value**

  None

- **Example**

```
/* Disable time out */
DrvI2C_TimeOutDisable();
```

14.3.18. DrvI2C_STSP

- **Prototype**

  void DrvI2C_STSP(unsigned char usignal);

- **Description**

  Generate the START or STOP singal from IIC bus.

  Configure the register 0x41004[3:2]

- **Parameter**

  usignal[in] : singal control

  0 Generate START singal

  1 Generate STOP singal

- **Include**

  Peripheral_lib/DrvI2C.h

- **Return Value**

  None

- **Example**

  /* Generate the START or STOP singal */

  DrvI2C_STSP(0);

14.3.19. DrvI2C_MGetACK

- **Prototype**

  unsigned char DrvI2C_MGetACK(unsigned int utime);

- **Description**

  Check the ACK from slaver during the set time

  Configure the register 0x41004[1]

- **Parameter**

  utimel[in]

  the set time :0~0xffff

- **Include**

  Peripheral_lib/DrvI2C.h

- **Return Value**

  0: ACK was returned

  1: No ACK was returned

- **Example**

  /* check the ACK during the 0xffff time*/

  Err_flag=DrvI2C_MGetACK(0xffff);

14.3.20. DrvI2C_DisableIOPin

- **Prototype**

  void DrvI2C_DisableIOPin(void)

- **Description**

  Disable IIC communication function of the IO port

  Configure the register 0x40844[16]=0

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvI2C.h

- **Return Value**

  None

- **Example**

  /* Disable IIC communication function of the IO port */

  DrvI2C_DisableIOPin();

14.3.21. DrvI2C_EnableSEn

- **Prototype**

  void DrvI2C_EnableSEn (void)

- **Description**

  Enable I2C Slave mode function. · Configure the register 0x41004[7]=1

- **Parameter**

  None

- **Include**

  Peripheral_lib/DrvI2C.h

- **Return Value**

  None

- **Example**

  /* Enable I2C Slave mode function */

  DrvI2C_EnableSEn();

14.3.22. DrvI2C_DisableSEn

- **Prototype**

void DrvI2C_DisableSEn (void)

- **Description**

Disable I2C Slave mode function. · Configure the register 0x41004[7]=0

- **Parameter**

None

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Value**

None

- **Example**

```
/* Disable I2C Slave mode function */
DrvI2C_DisableSEn();
```

14.3.23. DrvI2C_EnableI2CEn

- **Prototype**

void DrvI2C_EnableI2CEn (void)

- **Description**

Enable I2C function. · Configure the register 0x41000[0]=1

- **Parameter**

None

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Value**

None

- **Example**

```
/* Enable I2C function */
DrvI2C_ EnableI2CEn();
```

15. FLASH Read/Write Driver

15.1. Introduction

The following functions are included in FLASH Manager Section.

Item	Functions	Description
01	DrvFlash_Burn_Word	Write a data of word to the specified address
02	ROM_BurnPage	Write 32 data of word to the specified address in a row
03	ReadWord	Read a data of word from the specified address
04	ReadPage	Read 32 data of word from the specified address in a row
05	PageErase	Erase32 data of word to the specified address in a row
06	SectorErase	Erase one sector to the specified address

15.2. Functions

Note：User has to do SYS_DisableGIE, before execute Flash burn/read function. Disable system global GIE function, it can prevent program exception when executing Flash burn/read function.

15.2.1. DrvFlash_Burn_Word

- **Prototype**

  int DrvFlash_Burn_Word(unsigned int addr,unsigned int DelayTime,unsigned int data);

- **Description**

  Write a data of word to the specified address.

  Note : after executing the function “DrvFlash_Burn_Word” to do flash burn, please use “ReadPage” or “ReadWord” function to read out and verify the flash data correctness.

- **Parameters**

  addr[in] : the address to be written

  The input range is 0~0xffff, and the start address of flash is 0x90000. The interval of address is 4 bytes.

  For exemple：The address of 0x9a880 will written a word if addr[in]=0xa880.

  Delay time[in] : delay time of burning

  data [in] : the data to be written, it could be 0~0xffffffff

- **Include**

  Peripheral_lib/Drvflash.h

- **Return Value**

  0: Operation successful

- **Example**

  /* write the data of 0xFF05 to address of 0x90880 */

  DrvFlash_Burn_Word(0x0880,0x2000,0xFF05);

15.2.2. ROM_BurnPage

- **Prototype**

  int ROM_BurnPage(unsigned int addr,unsigned int DelayTime,int* data);

- **Description**

  Write 32 data of word to the specified address in a row one time.

  Note : after executing the function “ROM_BurnPage” to do flash burn, please use “ReadPage” or “ReadWord” function to read out and verify the flash data correctness.

- **Parameters**

  addr[in] : the initial address to be written

  The input range is 0~0xffff, and the start address of flash is0x90000. The interval of address is 128(32*4)

bytes, and the page only can be written one by one, for only 128byte in each page . and the address only could be 0xuu00 or 0xuu80. (u is defined by user)

For exemple：The address of 0x9a880 will written a word if addr[in]=0xa880.

Delay time[in] : delay time of burning

data [in] : the data to be written

it could be 0~0xffffffff . The length of data[in] is 32 word

- **Include**

Peripheral_lib/Drvflash.h

- **Return Value**

0xFF: Operation successful e

- **Example**

```
/* write 32 data of word to address of 0x90880 in a row one time */
int *A[32]={0};
ROM_BurnPage(0x0880, 0x2000, A);
```

15.2.3. ReadWord

- **Prototype**

int ReadWord(unsigned int addr);

- **Description**

Read a data of word from the specified address .

- **Parameters**

addr[in] : the address to be read

The input range is 0~0xffff, and the start address of flash is0x90000. The interval of address is 4 bytes.

For exemple： A word will be read from the address of 0x9a880 if addr[in]=0xa880.

- **Include**

Peripheral_lib/Drvflash.h

- **Return Value**

The value of the word

- **Example**

```
/* read the data from the address of 0x90880 */
Int flag;   flag= ReadWord(0x0880);
```

15.2.4. ReadPage

- **Prototype**

int ReadPage(unsigned int addr,int* data);

- **Description**

Read 32 data of word from the specified address in a row one time.

- **Parameters**

addr[in] : the initial address to be read

The input range is 0~0xffff, and the start address of flash is0x90000. The interval of address is 128(32*4) bytes, the page only could be read one by one, for only 128byte in each page . and the address only could be 0xuu00 or 0xuu80.

For exemple：The address of 0x9a880 will be read if addr[in]=0xa880.

data [in] : storage the data to be read

it could be 0~0xffffffff . The length of data[in] is 32 word

- **Include**

Peripheral_lib/Drvflash.h

- **Return Value**

0xFF: Operation successful

- **Example**

/* read 32 data of word from the address of 0x90880 in a row one time */

int *A[32]={0};

ReadPage(0x0880, A);

15.2.5. PageErase

- **Prototype**

int PageErase(unsigned int addr,unsigned int DelayTime);

- **Description**

Erase32 data of word to the specified address in a row one time.

- **Parameters**

addr[in] : the initial address to Erase

The input range is 0~0xffff, and the start address of flash is0x90000. The interval of address is 128(32*4) bytes, and the page only can be written one by one, for only 128byte in each page . and the address only could be 0xuu00 or 0xuu80.

For exemple：The address of 0x9a880 will written a word if addr[in]=0xa880.

Delay time[in]　: delay time of burning

- **Include**

Peripheral_lib/Drvflash.h

- **Return Value**

0xFF: Operation successful

- **Example**

```
/* Erase 32 data of word to address of 0x90880 in a row one time */
PageErase(0x0880, 0x2000);
```

15.2.6. SectorErase

- **Prototype**

```
int SectorErase(unsigned int addr,unsigned int DelayTime);
```

- **Description**

Erase one sector to the specified address.

- **Parameters**

addr[in] : the initial address to Erase

The input range is 0~0xffff, and the start address of flash is0x90000. Each sector include 32page.The interval of address is 128*32 bytes, and the first address is calculated by page,the address only could be 0xu000(u is defined by user)

For exemple：The address of 0x91000 will Erase first if addr[in]=0x1000.

Delay time[in] : delay time of burning

- **Include**

Peripheral_lib/Drvflash.h

- **Return Value**

0xFF: Operation successful

- **Example**

```
/* Erase one sector from the inital address of 0x91000 */
SectorErase(0x1000, 0x2000);
```

15.3. The storage structure of Flash

1 page=128 Bytes

1 sector=32 pages=4096 Bytes

Sector & Page			
Sector	Page	Address Range (Byte)	
0	0	000000H	00007FH
	1	000080H	0000FFH
	…	…	…
	30	000F00H	000F7FH
	31	000F80H	000FFFH
1	32	001000H	00107FH
	33	001080H	0010FFH
	…	…	…
	63	001F80H	001FFFH
…	…	…	…
15	480	00F000H	00F07FH
	…	…	…
	511	00FF80H	00FFFFH

16. Revision History

Version	Page	Revision Summary	The Date Of Revision
V01	ALL	First edition	2012/12/25
V02	ALL	2 edition	2013/06/21
V03	ALL	3 edition	2013/09/09

V04	ALL	Timer
		1、DrvTMA_CounterRead();//modify the return value
		2、DrvPWM_Condition();//Changing the location of the input parameters
		UART
		3、DrvUART_Clk();//Add new function
		4、DrvUART_GetRxFlag();/DrvUART_GetTxFlag(); // RXIF/TXIFcan be read correctly
		5、DrvUART_CheckTRMT();//new function
		6、DrvUART_Open();//delect the code about enbable the UART clock source
		7、DrvUART_EnableInt();//change the DrvUART.H file: the value of TX_IE/RX_IE: 19/18
		ADC
		8、DrvADC_GetConversionData(void);//modify the type of return value for int type
		9、DrvADC_ClkEnable(CLK,DIV);//modify the operation of enable the ADC Clock when uclkPH=0
		10、DrvADC_ReadIntFlag(void); // modify the operation about get ADC flag correctly
		CMP
		11、DrvCMP_ReadIntFlag();//modify the operation about get the CMP interrupt flag
		12、DrvCMP_OuputFilter();//change the function name :DrvCMP_OutputFilter();
		13、DrvCMP_RLO_Refv();// correct the setting of reference voltage
		SYSTEM
		14、SYS_LowPower(umode) //add the code to the mode of sleep/idle/wait
		15、SYS_EnableGIE();//correct the operation of open system interrupt
			2013/12/31

		I2C
		16、DrvI2C_Open();// correct the operation of CRG setting
		17、DrvI2C_SetIOPin() ; // new function
		18、DrvI2C_ReadData();// the data of SID can be read correctly
		19、DrvI2C_WriteData();//modify the operation of data sent
		20、DrvI2C_MGetACK();// new function
		21、DrvI2C_EnableInt();//the IIC interrupt flag can be setting correctly
		22、DrvI2C_DisableInt();//the IIC interrupt flag can be setting correctly
		23、DrvI2C_ClearIntFlag();//the IIC interrupt flag can be clear correctly
		24、DrvI2C_SlaveSet();//slaver address can be set correctly
		CLOCK
		25、DrvCLOCK_EnableLowOSC(unsigned int uSource, unsigned int udelay);//add input parameter,the lowOSC start time is control by external paramter
		GPIO
		26、DrvGPIO_IntTrigger(uint32_t port, uint32_t i32Bit, uint32_t mode) ; //optimize the operation that the trigger edge of IO port can be set more freely	

V05	ALL	1. change the setting of send and receive function for IIC
		2. DrvGPIO_Open();//modify the code of operation mode
		3. DrvGPIO_Close();// new function
			2014/6/24

V06	ALL	Add the description of Flash function Add the description of SYS_INTPriority() Add the description for flash storage struction Delete the description of MassErase()	2017/8/21
V07	ALL	Rebuild the C LIB: add the optimization property selection : '-ffunction-section -fdata-section -OS'	2014/9/30

V08	ALL	DrvGPIO_EnableAnalogPin();//Add new function
		Add the description of DrvRTC_HourFormat()
			2014/11/17

V09	ALL	1. DrvCLOCK_CalibrateHAO(short int uMHZ); // new function : calibrate the frequency of HAO	2015/03/25

		2. DrvGPIO_Open();//modify the performance and description
		3. Shielding the underlying interrupt vector set: the ir14 setting
		4. DrvGPIO_GetBit ();//the bit0 can be read.
		5. modify the description of SYS_EnableGIE()
		6. modify the description of DrvSPI32_SetCS()
		7. DrvSPI32_SetCSO() // new function
		8. add small size code function for openration mode setting of PT1/PT2/PT3
		Begin with DrvGPIO_PT1 (or DrvGPIO_PT2) (or DrvGPIO_PT3), for example：
		void DrvGPIO_PT1_EnableINPUT(short int ubit);
		void DrvGPIO_PT2_EnablePullHigh(short int ubit);
		void DrvGPIO_PT3_EnablePullHigh(short int ubit);	

V10	ALL	1.update each IP block diagram,and the Type Definition：ADC(OPO is update as OPOI, REFO is update as REFO_I), OPAMP(OPNS[3] is update as OPOI, OPNS[4] is update as OPO, DAO is update as DAOI) 2. DrvOP_OutputWithCPCLK() is update as DrvOP_OutputWithCHPCK() 3.DrvRTC_Write() // the HourFormat of 12HR or 24HR can be set correctly , hour value input range for 12HR Format :00~11	2015/06/16
V11	ALL	Update c library	2015/09/23
V12	P198	DrvUART_Open（ ）: change the the data type of input parameter ’uclock’ for ‘float ‘ ; Update the C LIB	2016/1/15
V13	P198	Update the C LIB	2016/04/15
V14	All	1. Remove all function diagram block of each chapter 2. In the flash read/write function in the example code, change the delay time from 10 to 0x2000. 3. Change input parameter ‘uclcok’ which data type for ‘unsigned int’ of ‘DrvUART_Open();‘ Modify the BaudRate calculation Update the C LIB 4. Re-typesetting document. EX : unified word size.	2017/02/20
V15	Chapter 16	1. Add note on Flash burn function. VDD3V have to work more than 2.7V	2018/02/18
V16	Chapter 3 Chapter 13	Add C Lib, DrvCLOCK_EnableENHAO, DrvCLOCK_SelectIHOSC_CalHAO Add C Lib DrvRTC_EnableWUEn and DrvRTC_DisableWUEn	2022/12/15

	Chapter 14 Chapter 16	Add C Lib DrvI2C_EnableSEn and DrvI2C_DisableSEn and DrvI2C_EnableI2Cen Modify Flash Function descriotion in the return value	

17. C Library Change List

Date	Previous Version List		New version List	
	Version	Bug List	Version	Improvement
2015-2-28	V0.8	DrvGPIO_GetBit() :the value of BIT0 cannot be read correctly	V0.9	Improve the code, each bit can be read correctly
		Other interrupt will be disable and that cannot be responsible when enable or disable a interrupt by the Interrupt Vector Enable Setting function		Delete the code of Interrupt Setting Function : disable the interrupt for other module
		DrvGPIO_Open() : The output(input) mode will be disable when the input(output) mode is enable		Delete the code: disable output/input mode
		The lack of HAO frequency correction function		Add new function : DrvCLOCK_CalibrateHAO(); the HAO frequency can be calibrate by this function
		The original function library compiled code is larger when setting the mode of one IO port		Add new function : Add PT1 PT2 / PT3 corresponding streamline function
				Add new function : DrvSPI32_SetCSO();

Date	Previous Version List		New version List	
	Version	Bug List	Version	Improvement
2015-6-16	V0.9	DrvRTC_Write(): the hour formate is set incorrectly	V1.0	the HourFormat of 12HR or 24HR can be set correctly , hour value input range for 12HR Format :00~11
		Update the block diagram of each IP		Update the Type Definition :ADC(OPO is update as OPOI, REFO is update as REFO_I), OPAMP(OPNS[3] is update as OPOI, OPNS[4] is update as OPO, DAO is update as DAOI)

				Add new function： DrvOP_OutputWithCHPCK (uCHPCK)

Date	Previous Version List		New version List	
	Version	Bug List	Version	Improvement
2015-9-23	V1.0	DrvGPIO_IntTrigger (): Enable the IO port external interrupt when setting trig edge;	V1.1	Delect the program for enable the external interrupt with IO Port
		There is wrong operation by bit for follow function: DrvUART_GetPERR(); DrvUART_GetFERR(); DrvUART_GetOERR(); DrvUART_GetABDOVF();		modify the algorithm of bit operation

Date	Previous Version List		New version List	
	Version	Bug List	Version	Improvement
2016-01-15	V1.1	DrvUART_Open(): The input parameter clock source number can not be the decimals	V1.2	Change the input parameter –‘uclock’ for ‘float ‘ data type;
		There is wrong calculation with ‘Baud Rate’ in the C lib;		modify the algorithm of Baud Rate calculation

Date	Previous Version List		New version List	
	Version	Bug List	Version	Improvement
2016-04-15	V1.2	DrvUART_Open(): There is calculation error with ‘Baud Rate’ in the C lib;	V1.3	modify the algorithm of Baud Rate calculation

Date	Previous Version List		New version List	
	Version	Bug List	Version	Improvement
2017-02-20	V1.3	Correct and modify the parts of the function : SYS_SleepFlagRead SYS_WdogFlagRead SYS_ResetFlagRead SYS_BOR_FlagRead SYS_SleepFlagClear SYS_WdogFlagClear SYS_ResetFlagClear SYS_BOR_FlagClear DrvWDT_CounterRead DrvTIMER_GetIntFlag	V1.4	1. Unified the parts of the function style 2. Reduce the DrvUART_Open code size issue 3. Creates new functions : DrvWDT_ResetEnable DrvOP_OPDEN

		DrvGPIO_SetPortBits DrvADC_ReadIntFlag DrvSPI32_GetDCFlag DrvSPI32_IsBusy DrvSPI32_IsABFlag DrvSPI32_IsOVFlag DrvSPI32_IsRxFlag DrvSPI32_IsRxBufferFull DrvSPI32_IsTxBufferFull DrvSPI32_EnableIO DrvUART_Open DrvUART_GetTxFlag DrvUART_GetRxFlag DrvOP_ReadIntFlag DrvRTC_ReadIntFlag Creates new functions : DrvWDT_ResetEnable DrvOP_OPDEN Modify and correct the header files : SpecialMacro.h DrvOP.h DrvUART.h DrvSPI32.h DrvTimer.h		
2018-02-18	V1.4	1. Set DrvI2C_SlaveSet but no any function issue. 2. . Set DrvSPI32_IsRxBufferFull(void) And DrvSPI32_IsTxBufferFull(void) but no any function issue.	V1.5	1. Correct DrvI2C_SlaveSet 2. Correct DrvSPI32_IsRxBufferFull(void) And DrvSPI32_IsTxBufferFull(void)
2022-12-15	V1.5	1. Correct DrvPWM_Open issue. 2. Correct DrvUART_Open/ DrvUART2_Open baud rate issue..	V1.7	1. Correct C Lib functions, DrvPWM_Open, DrvUART_Open/ DrvUART2_Open 2. Add new C Lib functions, DrvRTC_EnableWUEn, DrvRTC_DisableWUEn, DrvCLOCK_EnableENHAO, DrvI2C_EnableSEn, DrvI2C_DisableSEn, DrvI2C_EnableI2Cen 3. Add new C Lib functions, DrvCLOCK_SelectIHOSC_CalHAO