HY16F18 系列

C 函数库手册

目录

1 导读

1.1 C 函数库简介

本文件用于描述HYCON HY16F18系列C 函数库使用的参考手册。系统端软件开发人员可以通过使用C 函数库直接调用开发替换寄存器操作来有效的提高整个产品的开发效率。

1.2 相关文档

用户可以在我们公司网站上下载以下所有文档，获取其他相关的数据。
下载文档的网址：
http://www.hycontek.com/

2 系统控制

2.1 函数简介

该部分函数描述芯片工作系统控制，包含：

--工作模式（休眠模式（sleep）、待机模式（Idle）、等待模式（Waite mode））的控制

--芯片工作状态标志位控制

序号	函数名称	功能描述
01	SYS_SleepFlagRead	读取休眠标志位
02	SYS_SleepFlagClear	清除休眠标志位
03	SYS_WdogFlagRead	读取看门狗标志位
04	SYS_WdogFlagClear	清除看门狗标志位
05	SYS_ResetFlagRead	读取复位标志位
06	SYS_ResetFlagClear	清除复位标志位
07	SYS_BOR_FlagRead	读取低电压复位(BOR) 标志位
08	SYS_BOR_FlagClear	清除低电压复位(BOR) 标志位
09	SYS_EnableGIE	使能全局中断GIE且开启对应的中断向量
10	SYS_DisableGIE	关闭全局中断GIE
11	SYS_LowPower	设置IC低功耗工作模式
12	SYS_INTPriority	设置对应中断向量的中断优先权级别

2.2 函数说明

2.2.1 SYS_SleepFlagRead

- **函数**

  unsigned int SYS_SleepFlagRead (void);

- **函数功能**

  读取休眠标志位（Sleep flag）的值;

  读取寄存器0x40104[3]的值。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/System.h

- **函数返回值**

  0：正常工作模式

  1：芯片进入休眠模式

- **函数用法**

```
/* 读取休眠标志位 */
unsigned char temp_flag;     temp_flag=SYS_SleepFlagRead();
```

2.2.2 SYS_SleepFlagClear

- **函数**

  void SYS_SleepFlagClear(void);

- **函数功能**

  清零休眠标志位；

  清零寄存器0x40104[3]的值。.

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/System.h

- **函数返回值**

  无

- **函数用法**

```
/* 清零sleep flag. */
SYS_SleepFlagClear();    //set 0x40104[3]=0
```

2.2.3 SYS_WdogFlagRead

- **函数**

unsigned int SYS_WdogFlagRead (void);

- **函数功能**

读取看门狗(WDT)标志位的值；

读取寄存器0x40104[2]的值。

- **输入参数**

无

- **包含头文件**

Peripheral_lib/System.h

- **函数返回值**

0： 正常

1： 看门狗(WDT)已经触发

- **函数用法**

```
/* 读取看门狗(WDT)标志位. */
unsigned char flag;   flag=SYS_WdogFlagRead();
```

2.2.4 SYS_WdogFlagClear

- **函数**

void SYS_WdogFlagClear(void);

- **函数功能**

清零看门狗(WDT)标志位；

清零寄存器0x40104[2]的值。

- **输入参数**

无

- **包含头文件**

Peripheral_lib/System.h

- **函数返回值**

无

- **函数用法**

```
/* 清零看门狗(WDT)的标志位 */
SYS_WdogFlagClear();   //0x40104[2]=0
```

2.2.5 SYS_ResetFlagRead

- **函数**

unsigned int SYS_ResetFlagRead (void);

- **函数功能**

读取复位标志位的值；

读取寄存器0x40104[1]的值。

- **输入参数**

无

- **包含头文件**

  Peripheral_lib/System.h

- **函数返回值**

  0：正常

  1 : Reset PIN 外部复位已经触发

- **函数用法**

```
/* 读取重置标志位. */
unsigned char flag;   flag=SYS_ResetFlagRead();
```

2.2.6 SYS_ResetFlagClear

- **函数**

  void SYS_ResetFlagClear(void);

- **函数功能**

  清零复位标志位的值；

  清零寄存器0x40104[1]的值。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/System.h

- **函数返回值**

  无

- **函数用法**

```
/* 清零复位标志位 */
SYS_ResetFlagClear();   //0x40104[1]=0
```

2.2.7 SYS_BOR_FlagRead

- **函数**

  unsigned int SYS_BOR_FlagRead (void);

- **函数功能**

  读取低电压复位(BOR)标志位的值；

  读取寄存器0x40104[0]的值。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/System.h

- **函数返回值**

  0：正常

  1：低电压复位(BOR) 功能已触发

- **函数用法**

/* 读取低电压复位(BOR)标志位 . */

unsigned char flag; flag=SYS_BOR_FlagRead();

2.2.8 SYS_BOR_FlagClear

- **函数**

void SYS_BOR_FlagClear(void);

- **函数功能**

清零低电压复位(BOR)标志位；

清零寄存器0x40104[0]的值.

- **输入参数**

无

- **包含头文件**

Peripheral_lib/System.h

- **函数返回值**

无

- **函数用法**

/* 清零低电压复位(BOR) 标志位 */

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

2.2.9 SYS_EnableGIE

- **函数**

unsigned int SYS_EnableGIE (unsigned int uPriority,unsigned short intvector);

- **函数功能**

使能全局中断(GIE),使能对应中断向量并设置相应优先权级别的中断可以进行中断嵌套响应，优先级别高的先响应，中断向量优先权级别可以通过函数SYS_INTPriority()设置.

- **输入参数**

uPriority [in]： 设定允许开启中断向量的优先权级别，设置范围是0~4

0: 不允许任何优先权级别的中断向量回应

1: 只允许优先权级别被SYS_INTPriority()函数设定为最高级别的中断向量响应.

2: 只允许优先权级别被SYS_INTPriority()函数设定为最高级别、次高级别的中断向量响应 .

3: 只允许优先权级别被SYS_INTPriority()函数设定为最高级别、次高级别、低级别的中断向量响应

4: 只允许先权级别被SYS_INTPriority()函数设定为最高级别、次高级别、低级别、最低级别的中断向量回应

intvector[in]选择中断向量[HW5:HW4:HW3:HW2:HW1:HW0]；输入范围为0~0x3F，每一位值对应一个中断向量使能位HW5~HW0，只有对应位为1，才能使能对应的中断向量；

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

使能HW0/HW3/HW5，则invectior=0x29（101001B）；

使能所有中断向量，则invector=0x3F（111111B）；

不开启任何中断向量，则invector=0x00（000000B）

- **包含头文件**

Peripheral_lib/System.h

- **函数返回值**

0：设置成功

1：设置失败

- **函数用法**

/* 使能全局中断GIE，并允许优先权级别为0, 1, 2,3的中断向量回应,使能中断向量HW0~HW5 */

SYS_EnableGIE(4,0x3F);

2.2.10 SYS_DisableGIE

- **函数**

void SYS_DisableGIE (void);

- **函数功能**

关闭全局中断使能（GIE）。

- **输入参数**

无

- **包含头文件**

Peripheral_lib/System.h

- **函数返回值**

无

- **函数用法**

/* 关闭全局中断使能(GIE). */

SYS_DisableGIE();

2.2.11 SYS_LowPower

- **函数**

unsigned char SYS_LowPower(unsigned char umode)

- **函数功能**

设置并启动低功耗工作模式；开启低功耗模式前需要开启任何一个中断向量。且需要切换为低频晶震源。

设置寄存器0x40104[4]。

- **输入参数**

umode[in]：输入范围0~2

0：休眠模式（Sleep mode）

1：待机模式（Idle mode）

2：等待模式（Waite mode）

- **包含头文件**

Peripheral_lib/System.h

- **函数返回值**

0：设置成功

1：设置失败

- **函数用法**

```
/* 启动休眠工作模式*/
DrvGPIO_Open(E_PT1,0xFF,E_IO_IntEnable);   // 开启PT1 外部中断向量
SYS_EnableGIE(4, 0x3F);                    //开启全局中断控制
DrvCLOCK_SelectMCUClock(1,0);              //切换频率源为低频
SYS_LowPower(0);                           //进入休眠工作模式
```

2.2.12 SYS_INTPriority

- **函数**

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

- **函数功能**

设置对应中断向量的优先权级别，优先权级别为0~3,且0为最高级别。

注意：使用前，必须关闭所有的中断使能，才能修改中断优先权级别。

- **输入参数**

intvector[in] 中断向量选择，输入范围为0~5，分别对应HW0~HW5;

upriority[in]：设置开启中断向量的优先权级别，设置范围是0~3

0: 优先权级别为最高级别

1: 优先权级别为次高级别

2: 优先权级别为低级级别.

3: 优先权级别为最低级别

在设置中断优先权级别都为同一级别时，中断响应的的先后顺序为：

HW0 > HW1 > HW2 > …> HW5

- **包含头文件**

Peripheral_lib/System.h

- **函数返回值**

0：设置成功

1：设置失败

- **函数用法**

```
/* 设置中断向量0优先权级别为 1 * /
SYS_INTPriority(0,1);
```

3 芯片时钟源 CLOCK

3.1 函数简介

函数描述 CPU 及其他功能模块的时钟源操作，包含：

--内部高速及低速晶振的控制

--外部高速及低速晶振的控制

--CPU 时钟源的切换

序号	函数名称	功能描述
01	DrvCLOCK_EnableHighOSC	开启高频晶震
02	DrvCLOCK_CloseEHOSC	关闭外部高频晶震
03	DrvCLOCK_CloseIHOSC	关闭内部高频晶震
04	DrvCLOCK_SelectIHOSC	设置内部高频晶震HAO的频率
05	DrvCLOCK_EnableLowOSC	开启低频晶震
06	DrvCLOCK_CloseELOSC	关闭外部低频晶震
07	DrvCLOCK_SelectMCUClock	设置MCU时钟
08	DrvCLOCK_TrimHAO	内部高频晶震校正
09	DrvCLOCK_CalibrateHAO	按照出厂时HAO校正参数进行HAO频率校正
10	DrvCLOCK_SelectOHS_HS	外部高频晶振模式(HSXT)选择
11	DrvCLOCK_EnableENHAO	开启内部高频震荡器
12	DrvCLOCK_SelectIHOSC_CalHAO	设置内部高频晶震HAO的频率并且按照芯片出厂校正值进行HAO频率校正

3.2 内部定义常量

E_CLOCK_SOURCE

标识符	数值	功能意义
E_INTERNAL	0x0	内部
E_EXTERNAL	0x1	外部

E_TRIM_FREQUEN

标识符	数值	功能意义
TRIM_HAO2MHZ	0x0	校正HAO 2MHZ频率
TRIM_HAO4MHZ	0x1	校正HAO 4MHZ频率
TRIM_HAO10MHZ	0x2	校正HAO 10MHZ频率
TRIM_HAO16MHZ	0x3	校正HAO 16MHZ频率

3.3 函数说明

3.3.1 DrvCLOCK_EnableHighOSC

- 函数

  unsigned int DrvCLOCK_EnableHighOSC(E_CLOCK_SOURCE uSource, unsigned int delay)

- 函数功能

  开启高速晶震，并选择CPU时钟来源为外部晶震(HSXT)或者内部晶震(HSRC);

  设定等待晶震达到稳定所需时间;

  若CPU时钟源选择外部晶震，则寄存器0x40300[5]=1 , 0x40300[1]=1;

  若CPU时钟源选择为内部晶震，则寄存器0x40300[5]=0, 0x40300[0]=1;

- 输入参数

  uSource[in] : CPU时钟源选择. 设定范围：0~1

  0: 内部晶震

  1: 外部晶震

delay[in]： 设置等待晶震达到稳定所需时间. 设定范围：1~0xFFFFFFFF

需要注意当前CPU的指令周期CPU_CLK；晶震达到稳定时间t=(1/CPU_CLK)*4000*delay；

函数内部已经有4000次指令周期的循环，参数delay是倍数设置，设置参数时需要参考当前指令周期及需要启动的晶震频率。

外部晶震(EXT OSC)达到稳定需要的时间t :

4MHZ/8MHZ 约30ms

内部晶震（HAO）达到稳定需要的时间t :

2MHZ 约1.0ms

4MHZ 约0.5ms

10MHZ 约0.2ms

- 包含头文件

  Peripheral_lib/DrvCLOCK.h

- 函数返回值

  0：设置成功

  其他：设置失败

- 函数用法

  // 开启外部高速晶震,当前CPU_CK=10MHZ/2，开启外部4MHZ，设定延时t=40ms=(1/(10MHZ/2))*4000*50

  DrvCLOCK_EnableHighOSC(E_EXTERNAL,50);

3.3.2 DrvCLOCK_CloseEHOSC

- 函数

  void DrvCLOCK_CloseEHOSC()

- **函数功能**

关闭外部高速晶震；注意：若被关闭的晶震当前是作为CPU时钟源，必须先切换为有效时钟源才能关闭该晶震。

寄存器0x40300[1]=0;

- **函数输入参数**

无

- **包含头文件**

Peripheral_lib/DrvCLOCK.h

- **函数返回值**

无

- **函数用法**

```
/* 开启设定内部晶震作为CPU时钟源，关闭外部高速晶震 */
DrvCLOCK_EnableHighOSC(E_INTERNAL,50); //开启内部高速晶震
DrvCLOCK_CloseEHOSC();                 //关闭外部高速晶震
```

3.3.3 DrvCLOCK_CloseIHOSC

- **函数**

void DrvCLOCK_CloseIHOSC()

- **函数功能**

关闭内部高速晶震(HAO)；但是前提必须先将CPU时钟源切换到有效的时钟源。

寄存器0x40300[0]=0;

- **函数输入参数**

无

- **包含头文件**

Peripheral_lib/DrvCLOCK.h

- **函数返回值**

无

- **函数用法**

```
/* 开启外部高速晶震作为CPU时钟源，关闭内部高速晶震*/
DrvCLOCK_EnableHighOSC(E_EXTERNAL,50);  //开启外部高速时钟源
DrvCLOCK_CloseIHOSC(); //关闭内部高速晶震
```

3.3.4 DrvCLOCK_SelectIHOSC

- **函数**

unsigned int DrvCLOCK_SelectIHOSC(uMode)

- **函数功能**

设置内部晶震HAO的频率模式;

设置寄存器0x40300[4:3]。

- **输入参数**

uMode [in]：HAO频率值，有效输入范围 ：0~2

0 ：2MHz, 0x40300[4:3]=00b

1 ：4MHz, 0x40300[4:3]=01b

2 ：10MHz, 0x40300[4:3]=10b

3 ：Rsv

- **包含头文件**

Peripheral_lib/DrvCLOCK.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/* 设置内部高速晶震(HAO)=4MHz*/

DrvCLOCK_SelectIHOSC(1);

3.3.5 DrvCLOCK_EnableLowOSC

- **函数**

unsigned int DrvCLOCK_EnableLowOSC(E_CLOCK_SOURCE uSource，unint delay)

- **函数功能**

开启低速晶震频率 ,并设置CPU时钟源是内部晶震(LSRC)或者外部晶震(LSXT)及设置等待晶震达到稳定所需时间；

寄存器0x40300[6]=1. 0x40300[2]=1

- **输入参数**

uSource [in]： 输入范围 0~1

0: 内部晶震LSRC

1: 外部晶震LSXT

Delay[in]： 等待晶震的时间设置，需要参考当前的指令周期来设置

设定范围：0x0~0xffffffff

外部低速晶振LSXT稳定时间：32768HZ 约1.3s

内部低速晶振LSRC稳定时间: 约510us

- **包含头文件**

Peripheral_lib/DrvCLOCK.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/* 开启外部低速晶振LSXT并设置稳定时间为130000*/

DrvCLOCK_EnableLowOSC(E_EXTERNAL,130000);

3.3.6 DrvCLOCK_CloseELOSC

- **函数**

  void DrvCLOCK_CloseELOSC()

- **函数功能**

  关闭外部低速晶振;但是需要先唤醒内部晶振(LPO)并切换至内部晶振(LPO)。

  寄存器0x40300[2]=0。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvCLOCK.h

- **函数返回值**

  无

- **函数用法**

```
/* 先开启内部低速晶振，再关闭外部低速晶振 */
DrvCLOCK_EnableLowOSC(E_INTERNAL,130000);  //开启内部低速晶振
DrvCLOCK_CloseELOSC();  //关闭外部低速晶震
```

3.3.7 DrvCLOCK_SelectMCUClock

- **函数**

  unsigned int DrvCLOCK_SelectMCUClock(uSource,uDiv)

- **函数功能**

  设置CPU的时钟源为高速频率(HS_CK)或低速频率(LS_CK)，设置时钟分频数值；

  设置寄存器0x40308[0]与0x40308[1]。

- **输入参数**

  uSource [in] : CPU时钟源选择

  0： 高速频率(HS_CK)

  1： 低速频率(LS_CK)

  uDiv [in] : CPU时钟源分频设置

  0 : ÷1

  1 : ÷2

- **包含头文件**

  Peripheral_lib/DrvCLOCK.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

```
/* 设置CPU时钟源为高速频率(HS_CK)并且2分频 */
DrvCLOCK_SelectMCUClock(0,1); //设置MCU的高速频率源为HS_CK，且设置2分频
```

3.3.8 DrvCLOCK_TrimHAO

- **函数**

  unsigned int DrvCLOCK_TrimHAO(uTrim)

- **函数功能**

  校正内部晶震(HAO);

  设置寄存器0x40304[7:0]的值。

- **输入参数**

  uTrim[in]:频率校正值，输入范围 ：0~0xff

- **包含头文件**

  Peripheral_lib/DrvCLOCK.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

```
/*写入0x80在效正内部晶振控制缓存器 */
DrvCLOCK_TrimHAO(0x80);  // 设置 0x40304[7:0]=0x80
```

3.3.9 DrvCLOCK_CalibrateHAO

- **函数**

  void DrvCLOCK_CalibrateHAO(short int uMHZ)

- **函数功能**

  按照芯片出厂时HAO校正值来校正内部晶震(HAO);使用时注意要与选定的HAO频率对应；

  设置寄存器0x40304[7:0]的值。

- **输入参数**

  uMHZ [in]： 待校正值的HAO频率模式选择

  0：校正 2MHZ；1：校正 4MHZ； 2：校正 10MHZ；3：Rsv；

- **包含头文件**

  Peripheral_lib/DrvCLOCK.h

- **函数返回值**

  无

- **函数用法**

```
/*校正内部晶震(HAO)4MHZ */
DrvCLOCK_SelectIHOSC(1);  //setting HAO=4MHZ;
DrvCLOCK_CalibrateHAO(1);  //calibrate 4MHZ；
```

3.3.10 DrvCLOCK_SelectOHS_HS

- **函数**

  unsigned int DrvCLOCK_SelectOHS_HS(unsigned int uMode)

- **函数功能**

外部高频晶振模式(HSXT)选择, HSXT可选择是大于4MHZ，或是小于4MHZ；

设置寄存器0x40300[7]的值。

- **输入参数**

uMode [in]： 外部高频晶振(HSXT)模式选择. 输入范围 ：0~1

0：HSXT<4MHz；1：HSXT>4MHz；

- **包含头文件**

Peripheral_lib/DrvCLOCK.h

- **函数返回值**

0： 设置成功

1： 设置失败

- **函数用法**

```
/*选择外部晶震(HSXT)>4MHZ */
DrvCLOCK_SelectOHS_HS(1);   //Select HSXT > 4MHZ;
```

3.3.11 DrvCLOCK_EnableENHAO

- **函数**

void DrvCLOCK_EnableENHAO (void)

- **函数功能**

开启内部高速震荡器

设置寄存器0x40300[0]=1

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvCLOCK.h

- **函数返回值**

无

- **函数用法**

```
/*开启内部高速震荡器 */
DrvCLOCK_EnableENHAO();   //ENHAO=1b
```

3.3.12 DrvCLOCK_SelectIHOSC_CalHAO

- **函数**

unsigned int DrvCLOCK_SelectIHOSC_CalHAO(unsigned int uMode)

- **函数功能**

设置内部晶震HAO的频率模式，并且按照芯片出厂时HAO校正值来校正内部晶震(HAO)

设置寄存器0x40300[4:3]、0x40304[7:0]的值。

- **输入参数**

uMode [in] : HAO频率值，输入范围 ：0~2

0 ：2MHz, 0x40300[4:3]=00b

1 ：4MHz, 0x40300[4:3]=01b

2 ：10MHz, 0x40300[4:3]=10b

- **包含头文件**

Peripheral_lib/DrvCLOCK.h

- **函数返回值**

0：设置成功 1：设置失败

- **函数用法**

```
/*选择HAO=4MHz，并且校正内部晶震(HAO)4MHZ=4MHz<2%误差 */
DrvCLOCK_SelectIHOSC_CalHAO(1);
```

4 定时计数器 TIMER/WDT

4.1 函数简介

该部分函数描述看门狗(WDT)/定时计数器 A(Timer A)/ 定时计数器 B(Timer B) /定时计数器 C(Timer C)的功能控制，包含：

--看门狗(WDT)的配置控制、启动控制、中断控制

--定时计数器 A(Timer A)的配置控制、启动控制、定时中断控制

--定时计数器 B(Timer B)的配置控制、启动控制、定时控制及 PWM 模式控制

--定时计数器 C(Timer C)的配置控制及 Capture 的控制

序号	函数名称	功能描述
01	DrvWDT_Open	开启看门狗(WDT)
02	DrvWDT_CounterRead	读取看门狗计数值
03	DrvWDT_ClearWDT	清零看门狗计数值
04	DrvWDT_ResetEnable	WDT 中断工作模式选择为 Reset Mode
05	DrvTMA_Open	开启定时计数器(Timer A)
06	DrvTMA_Close	关闭定时计数器(Timer A)
07	DrvTMA_CounterRead	读取定时计数器(Timer A)计数值
08	DrvTMA_ClearTMA	清零定时计数器(Timer A)计数值
09	DrvTIMER_EnableInt	开启定时器中断 TA/TB/TC/WDT.
10	DrvTIMER_DisableInt	关闭定时器中断 TA/TB/TC/WDT
11	DrvTIMER_GetIntFlag	读取中断请求标志位
12	DrvTIMER_ClearIntFlag	清除中断请求标志位
13	DrvTMB_Open	开启定时计数器(Timer B)
14	DrvTMB_Clk_Source	设置定时计数器(Timer B/C)时钟源
15	DrvTMB_Clk_Disable	关闭定时计数器(Timer B/C)时钟源
16	DrvTMB_ClearTMB	清零定时计数器(Timer B)计数值
17	DrvTMB_CounterRead	读取定时计数器(Timer B)计数值
18	DrvTMB_Close	关闭定时计数器(Timer B)
19	DrvPWM0_Open	开启 PWM 功能及 PWM0 模式
20	DrvPWM1_Open	开启 PWM 功能及 PWM1 模式
21	DrvPWM_CountCondition	设置 PWM 占空比设置
22	DrvPWM0_Close	关闭 PWM0 模式
23	DrvPWM1_Close	关闭 PWM1 模式
24	DrvCAPTURE1_Open	开启捕捉比较器 1
25	DrvCAPTURE2_Open	开启捕捉比较器 2
26	DrvCAPTURE1_Read	读取捕捉比较器 1 计数值

27	DrvCAPTURE2_Read	读取捕捉比较器 2 计数值
28	DrvCAPTURE_IPort	设置捕捉比较器信号输入 IO 口

4.2 内部定义常量

E_WDT_MODE

标识符	设定值	功能意义
E_IRQ	0x0	IRQ mode
E_RST	0x1	RESET mode

E_WDT_PRE_SCALER

标识符	设定值	功能意义
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

标识符	设定值	功能意义
E_TMA	0x0	定时器 A
E_TMB	0x1	定时器 B
E_TMC	0x2	定时器 C
E_WDT	0x3	看门狗 WDT

E_TMB_MODE

标识符	设定值	功能意义
E_TMB_MODE0	0x0	16-bit 递增计数器TBR[15:0]，步长为1；
E_TMB_MODE1	0x1	16-bit 递增/递减计数器TBR[15:0]，步长为1；
E_TMB_MODE2	0x2	两个8-bit的递增计数器TBR[15:8]/TBR[7:0]，两个计数器独立同时计数，步长为1。
E_TMB_MODE3	0x3	两个8-bit递增计数器TBR[15:8]/TBR[7:0], ，计数器步长为1，计数器TBR[7:0]计数溢出后计数器TBR[15:0]才会自动加1。

E_TRIGGER_SOURCE

标识符	设定值	功能意义
E_TMB_NORMAL	0x0	总是启用（Always Enable）连续计数方式
E_TMB_CMP_HIGH	0x1	比较器(CMP)高电位触发

E_TMB_OP_HIGH	0x2	运放(OP)高电位触发
E_TMB_GPIO_HIGH	0x3	Timer C的输出CPI1 高电位触发

E_DRVTIMER_CLOCK_SOURCE

标识符	数值	函数功能
E_HS_CK	0	TMA 时钟源为HS_CK
E_LS_CK	1	TMA 时钟源为 LS_CK

E_CAPTURE_SOURCE

标识符	数值	函数功能
E_TMC_CMPO	0x0	比较器输出
E_TMC_OPOD	0x1	运算放大器数字输出
E_TMC_LSCK	0x2	低频时钟源
E_TMC_TCI0	0x3	捕捉比较器1 I/O输入
E_TMC_TCI1	0x0	捕捉比较器2 I/O输入
E_TMC_ASTC0	0X1	捕捉比较器2 的输入源与捕捉比较器1一致

4.3 函数说明

4.3.1 DrvWDT_Open

- **函数**

  uint32_t DrvWDT_Open (E_WDT_MODE eMode , E_WDT_PRE_SCALER eWDTpreScaler)

- **函数功能**

  使能看门狗(WDT)，设置看门狗(WDT)工作模式，设置时钟分频来设定计数溢出值；

  设置寄存器0x40108[2:0] / 0x40108[4]=1。

- **输入参数**

  eMode[in]： 看门狗工作模式选择

  0： 定时中断模式

  1： 复位模式

  eWDTpreScaler[in]： 看门狗时钟源分频设置

  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

- **包含头文件**

  Peripheral_lib/DrvTIMER.h

- **函数返回值**

  0：设置成功

  1：设置失败

- **函数用法**

  /* 设置看门狗(WDT)为 IRQ mode 及 CLK / 32 */

  DrvWDT_Open(E_IRQ , E_PRE_SCALER_D32);

4.3.2 DrvWDT_CounterRead

- **函数**

  uint32_t DrvWDT_CounterRead (void)

- **函数功能**

  读取看门狗(WDT)计数寄存器的值；读取寄存器 0x40108 [30:16] 。

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvTIMER.h

- **函数返回值**

看门狗计数值.

- **函数用法**

/* 读取看门狗(WDT)计数寄存器的值 */

unsigned int data；data=DrvWDT_CounterRead();

4.3.3 DrvWDT_ClearWDT

- **函数**

void DrvWDT_ClearWDT (void)

- **函数功能**

清零看门狗(WDT)计数寄存器值，设置寄存器0X40108[5]=1，且清零后该位自动变为0 。

寄存器0x40108[30:16]清除为0。

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvTIMER.h

- **函数返回值**

无

- **函数用法**

/* 清零看门狗 */

DrvWDT_ClearWDT();

4.3.4 DrvWDT_ResetEnable

- **函数**

void DrvWDT_ResetEnable(void)

- **函数功能**

WDT中断工作模式选择为Reset Mode，设置寄存器0x40108[6]=1b。

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvTIMER.h

- **函数返回值**

无

- **函数用法**

/* WDT中断工作模式选择为Reset Mode */

DrvWDT_ResetEnable();

4.3.5 DrvTMA_Open

- **函数**

unsigned int DrvTMA_Open (eTMAOV, E_DRVTIMER_CLOCK_SOURCE uclk)

- **函数功能**

使能定时计数器A(Timer A)，设置定时计数器A(Timer A)时钟源，设定计数溢出值；

设置寄存器0x40C00[5]=1, 0x40C00[3:0], 0x40308[3], 0x40308[2]。

- **输入参数**

eTMAOV [in]： 定时计数器A(Timer A) 计数溢出值设置：.

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]： 定时计数器A(Timer A)时钟源设置.

0: HS_CK

1: LS_CK

- **包含头文件**

Peripheral_lib/DrvTIMER.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/* 使能定时计数器A(Timer A)，且计数溢出值为taclk/8. */

DrvTMA_Open(2, 0);

4.3.6 DrvTMA_Close

- **函数**

  void DrvTMA_Close (void)

- **函数功能**

  关闭定时计数器A(Timer A);

  设置寄存器0x40C00[5]=0。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvTIMER.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭定时计数器A(Timer A) */

  DrvTMA_Close();

4.3.7 DrvTMA_CounterRead

- **函数**

  unsigned int DrvTMA_CounterRead (void)

- **函数功能**

  读取定时计数器A(Timer A)计数寄存器的值TAR;

  读取寄存器0x40C00[15:0] 。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvTIMER.h

- **函数返回值**

  定时计数器A(Timer A)计数值.

- **函数用法**

  /*读取定时计数器A(Timer A)计数值 */

  unsigned short tcounter; tcounter=DrvTMA_CounterRead();

4.3.8 DrvTMA_ClearTMA

- **函数**

  void DrvTMA_ClearTMA (void)

- **函数功能**

  清零定时计数器A(Timer A)计数寄存器TAR;

  设置寄存器0x40C00[4]=1，清零完成后自动置0。寄存器0x40C00[15:0]清除为0

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvTIMER.h

- **函数返回值**

  无

- **函数用法**

  /* 清零定时计数器A(Timer A)计数寄存器 */

  DrvTMA_ClearTMA();

4.3.9 DrvTIMER_EnableInt

- **函数**

  unsigned int DrvTIMER_EnableInt (E_TIMER_CHANNEL ch)

- **函数功能**

  使能WDT/Timer A/Timer B/Timer C 中断功能；

  设置寄存器0x40004[20:16]的对应中断使能位=1。

- **输入参数**

  ch [in] ：中断源设置. 输入范围 ：0~4

  0：定时计数器 A　　1：定时计数器B　2：定时计数器C的C0中断

  3：定时计数器C的C1中断　　4：看门狗

- **包含头文件**

  Peripheral_lib/DrvTIMER.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /*使能定时计数器A(Timer A) 中断 */

  DrvTIMER_EnableInt(E_TMA);

4.3.10 DrvTIMER_DisableInt

- **函数**

  unsigned int DrvTIMER_DisableInt (E_TIMER_CHANNEL ch)

- **函数功能**

  关闭WDT/Timer A/Timer B/Timer C 中断功能；

  设置寄存器0x40004[20:16]的对应模块中断使能位=0。

- **输入参数**

  ch [in]：中断源选择. 输入范围 ：0~4

  0：定时计数器 A　　1：定时计数器B　2：定时计数器C的C0中断

  3：定时计数器C的C1中断　　4：看门狗

- **包含头文件**

  Peripheral_lib/DrvTIMER.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /* 关闭定时计数器A(Timer A) 中断向量*/

  DrvTIMER_DisableInt(E_TMA);

4.3.11 DrvTIMER_GetIntFlag

- **函数**

  unsigned int DrvTIMER_GetIntFlag (E_TIMER_CHANNEL ch)

- **函数功能**

  读取WDT/Timer A/Timer B/Timer C中断请求标志位;

  读取寄存器0x40004[4:0]对应模块中断请求标志位。

- **输入参数**

  ch [in]：中断源选择. 输入范围 ：0~4

  0：定时计数器 A　　1：定时计数器B　2：定时计数器C的C0中断

  3：定时计数器C的C1中断　　4：看门狗

- **包含头文件**

  Peripheral_lib/DrvTIMER.h

- **函数返回值**

  0: 无中断请求

  1: 有中断请求

- **函数用法**

  /*读取定时计数器A(Timer A) 中断请求标志位*/

  unsigned char flag ; flag=DrvTIMER_GetIntFlag(E_TMA);

4.3.12 DrvTIMER_ClearIntFlag

- **函数**

  unsigned int DrvTIMER_ClearIntFlag (E_TIMER_CHANNEL ch)

- **函数功能**

  清除WDT/Timer A/Timer B/Timer C 中断请求标志位；

  设置寄存器0x40004[4:0]对应模块功能中断标志位=0。

- **输入参数**

ch [in]：中断源设置. 输入范围 ：0~4

0：定时计数器 A　　1：定时计数器B　2：定时计数器C的C0中断

3：定时计数器C的C1中断　　4：看门狗

- **包含头文件**

Peripheral_lib/DrvTIMER.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/* 清除定时计数器A(Timer A) 中断要求标志位*/

DrvTIMER_ClearIntFlag(E_TMA);

4.3.13 DrvTMB_Open

- **函数**

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

- **函数功能**

使能定时计数器B(Timer B)，设置定时计数器B(Timer B)寄存器计数模式，设置定时计数器B(Timer B)计数触发源，设置定时计数器B(Timer B)计数溢出值；支持比较器、捕捉、计数和定时功能；

设置寄存器0x40C0C[15:0], 0x40C04[3:0], 0x40C04[5]=1b。

- **输入参数**

eTMBmode [in]： 代表定时计数器B(Timer B)计数模式.

0: 计数寄存器(TBR)是递增计数模式，在每一个TBCLK的上升沿加1.当TBR >TBC0时，在下一个TBCLK的上升沿TBR变为0且TMBIF被置1，然后TBR又重新递增计数。

1: 计数寄存器(TBR)是递增递减计数模式 ;作为递增模式 每一个TBCLK上升沿TBR自动加1 当TBR=TBC0时，TBR 变为递减模式，且 TBR 在每一个 TBCLK 上升沿自动减 1，当 TBR 递减为 0 时，中断标志位(TMBIF)被置 1，然后 TBR 又重新开始递增计数.

2: 计数寄存器(TBR)分为两个 8-bitPWM 模式，两个独立的递增计数器，两个计数器都是在 TBCLK 的上升沿自动加 1；当 TBR[15:8]=TBC0[15:8]时 TBR[15:0]=0，TBR[7:0]=TBC0[7:0]时，TBR[7:0]=0；当 TBR[15:8]=TBC0[15:8]时，在 TBCLK 的下一个上升沿时 TBR[15:8]=0，但 TMBIF 保持为 0,；在 TBR[7:0]=TBC0[7:0]时，在 TBCLK 的下一个上升沿 TBR[7：0]=0，且中断标志位(TMBIF)被置 1。

3: 计数寄存器(TBR)作为步进模式。TBR分为两个8-bit递增计数器，在TBCLK的每个上升沿自动加1，TBR[15:8]计数上限受控于TBC0[15:8]，TBR[7:0]计数上限受控于TBC0[7:0]；当TBR[7:0]=TBC0[7:0]时，在TBCLK的下一个上升沿时TBR[7:0]=0，且TBR[15:8]自动加1，中断标志位TMBIF被置1。

eTriSource [in]： 表示Timer B 计数触发源选择.

0: 总是启用（Always Enable ）连续计数方式

1: 比较器(CMP)高电位触发

2: 运算放大器(OP)数字输出高电位触发

3: 定时计数器 C(Timer C) 输出高电位触发（CPI1)

eTMAOV [in]：计数溢出值设置，设定范围是0~0xffff

- **包含头文件**

Peripheral_lib/DrvTIMER.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/* 开启定时计数器B(TMB)，设置模式1，计数值为Taclk/32，OP 高电位触发 */

DrvTMB_Open(E_TMB_MODE0, E_TMB_OP_HIGH,4);

4.3.14 DrvTMB_Clk_Source

- **函数**

unsigned int DrvTMBC_Clk_Source (E_DRVTIMER_CLOCK_SOURCE uclk, uPerScale)

- **函数功能**

定时计数器B/C 时钟源设置，及时钟源分频设置；

设置寄存器 0x40308[7:6], 0x40308[5:4]

- **输入参数**

uclk[in]：定时计数器 B/C 时钟源

0: HS_CK

1: LS_CK

uPerScale[in]：定时计数器B/C 时钟分频设置

0: ÷1

1: ÷2

2: ÷4

3: ÷8

- **包含头文件**

Peripheral_lib/DrvTIMER.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/* 设置定时计数器B 时钟源为HS_CK, 分频为 2. */

DrvTMBC_Clk_Source(0,1);

4.3.15 DrvTMB_Clk_Disable

- **函数**

viod DrvTMBC_Clk_Disable (viod)

- **函数功能**

  关闭定时计数器B/C 时钟源;

  设置寄存器0x40308[6]=0。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvTIMER.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭定时计数器B/C 时钟源*/

  DrvTMBC_Clk_Disable();

4.3.16 DrvTMB_ClearTMB

- **函数**

  void DrvTMB_ClearTMB (void)

- **函数功能**

  清除定时计数器B(Timer B)的计数寄存器;

  设置寄存器0x40C04[4]=1,清零后该位自动置0, 寄存器0x40C08[15:0]清除为0。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvTIMER.h

- **函数返回值**

  无

- **函数用法**

  /* 清除定时计数器B(Timer B)的计数寄存器. */

  DrvTMB_ClearTMB();

4.3.17 DrvTMB_CounterRead

- **函数**

  unsigned int DrvTMB_CounterRead (void)

- **函数功能**

  读取定时计数器B(Timer B)的计数寄存器的值；

  读取寄存器0x40C08[15:0]。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvTIMER.h

- **函数返回值**

  Timer B 计数值

- **函数用法**

  /* 读取定时计数器B(Timer B)的计数值 */

  unsigned short Tcounter; Tcounter=DrvTMB_CounterRead();

4.3.18 DrvTMB_Close

- **函数**

  void DrvTMB_Close (void)

- **函数功能**

  关闭定时计数器B(Timer B)的功能；设置寄存器0x40C04[5]=0。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvTIMER.h

- **函数返回值**

  无

- **函数用法**

  /*关闭定时计数器B(Timer B)*/

  DrvTMB_Close();

4.3.19 DrvPWM0_Open

- **函数**

  unsigned int DrvPWM0_Open (uPWM_Mode , uInv, uOuputPin)

- **函数功能**

  使能PWM0功能，及设置PWM0的工作模式、输出波形反相设置与输出IO设置；

  设置寄存器0x40C04[18:16] / 0x40C04[19] / 0x40840[4:2] / 0x40840[0]=1b 。

- **输入参数**

  uPWM_Mode [in] : PWM 工作模式设置

  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输出PWM波形相位控制

  0 : 输出波形反相

  1 : 输出波形正常

  uOuputPin[in] : PWM输出IO 设置

  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

- **包含头文件**

Peripheral_lib/DrvTIMER.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

```
/*使能PWM0且工作模式是PWMA ，反相输出，PT1.0输出 */
DrvPWM0_Open(0, 0, 0);
```

4.3.20 DrvPWM1_Open

- **函数**

unsigned int DrvPWM1_Open (uPWM_Mode , uInv, uOuputPin)

- **函数功能**

使能PWM1功能，及设置PWM1的工作模式、输出波形反相设置及输出IO设置；

设置寄存器0x40C04[23:20]/ 0x40840[4:2] / 0x40840[1]=1b 。

- **输入参数**

uPWM_Mode [in] : PWM 工作模式设置

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 输出波形相位控制

0： 输出波形反相

1： 输出波形正常

uOuputPin[in]　PWM输出IO口控制

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

- **包含头文件**

Peripheral_lib/DrvTIMER.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

```
/*使能PWM1，且设置工作模式是PWMA ，反相输出，PT1.1输出 */
DrvPWM1_Open(0, 0, 0);
```

4.3.21 DrvPWM_CountCondition

- **函数**

void DrvPWM_CountCondition (uTBC2 , uTBC1)

- **函数功能**

PWM0/PWM1占空比设置，写入计数寄存器(TBC2, TBC1);

设置寄存器0x40C10[15:0](TBC1) / 0x40C10[31:16](TBC2)

- **输入参数**

uTBC1 [in] ：PWM0占空比设置

TBC1 设定范围0~0xFFFF

uTBC2 [in] ：PWM1占空比设置

TBC2 设定范围0~0xFFFF

- **包含头文件**

Peripheral_lib/DrvTIMER.h

- **函数返回值**

无

- **函数用法**

```
/* 设置TBC1, TBC2 值为0x4000 */
DrvPWM_CountCondition(0x4000,0x4000);
```

4.3.22 DrvPWM0_Close

- **函数**

void DrvPWM0_Close (void)

- **函数功能**

关闭PWM0输出;

设置寄存器0x40840[0]=0.

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvTIMER.h

- **函数返回值**

  无

- **函数用法**

  /*PWM0输出关闭 */

  DrvPWM0_Close();

4.3.23 DrvPWM1_Close

- **函数**

  void DrvPWM1_Close (void)

- **函数功能**

  关闭PWM1输出；

  设置寄存器0x40840[1]=0

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvTIMER.h

- **函数返回值**

  无

- **函数用法**

  /*PWM1输出关闭*/

  DrvPWM1_Close();

4.3.24 DrvCAPTURE1_Open

- **函数**

  unsigned int DrvCapture1_Open (CAPTURE_SOURCE uChannel , uDivider, uEdge)

- **函数功能**

  开启信号捕捉比较器Capture1，捕捉比较器输入信号源设置、信号除频器设置及捕捉信号触发边沿设置.

  设置寄存器0x40C14[21:20] / 0x40C14[19:16] / 0x40C14[1] / 0x40C14[0]=1。

- **输入参数**

  uChannel [in] ：捕捉器Capture1输入信号源设置. 输入范围 :0~3

  0：比较器输出(CMPO)

  1：运算放大器输出(OPOD)

  2：低速时钟源(LS_CK)

  3： IO口输入(TCI1)

  uDivider [in] ：输入信号除频设置. 输入范围 :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]：捕捉信号触发边沿设置

0： 上升沿触发

1： 下降沿触发

- **包含头文件**

Peripheral_lib/DrvTIMER.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/* 使能捕捉器capture1，输入信号选择TCI1，信号除频为2048，上升沿触发模式 */

DrvCapture1_Open(3, 11, 0);

4.3.25 DrvCAPTURE2_Open

- **函数**

unsigned int DrvCapture2_Open (CAPTURE_SOURCE uChannel, uEdge)

- **函数功能**

使能信号捕捉比较器Capture2，设置捕捉信号输入源及捕捉信号触发边沿.

设置寄存器0x40C14[22] / 0x40C14[2] / 0x40C14[0]=1。

- **输入参数**

uChannel [in] : Capture 2 捕捉信号输入源设置

0: 信号输入源 TCI2 来自 GPIO

1: 与 Capture1 一样的信号输入源

uEdge [in]： 捕捉信号触发边沿设置

0: 上升沿触发

1: 下降沿触发

- **包含头文件**

Peripheral_lib/DrvTIMER.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/* 使能捕捉器capture2，输入信号选择与Capture1 一样的信号输入源，上升沿触发模式 */

DrvCapture2_Open(1, 0);

4.3.26 DrvCAPTURE1_Read

- 函数

  unsigned int DrvCapture1_Read (void)

- 函数功能

  读取捕捉比较器Capture1的计数值;

  读取寄存器0x40C18[15:0]值

- 输入参数

  无

- 包含头文件

  Peripheral_lib/DrvTIMER.h

- 函数返回值

  Capture1 计数值TCR0(0~0xffff)

- 函数用法

  /* 读取捕捉比较器Capture1 计数值*/

  unsigned short tcounter; tcounter=DrvCapture1_Read();

4.3.27 DrvCAPTURE2_Read

- 函数

  unsigned int DrvCapture2_Read (void)

- 函数功能

  读取捕捉比较器Capture2 计数值;

  读取寄存器0x40C18[31:16]值

- 输入参数

  无

- 包含头文件

  Peripheral_lib/DrvTIMER.h

- 函数返回值

  Capture2 计数值TCR1(0~0xffff)

- 函数用法

  /* 读取捕捉比较器Capture2 计数值 */

  unsigned short tcounter; tcounter=DrvCapture2_Read();

4.3.28 DrvCAPTURE_IPort

- 函数

  unsigned int DrvCapture_Iport (uInputPin)

- 函数功能

  设置信号捕捉比较器的信号输入IO口;

设置寄存器0x40840[7:5]。

- **输入参数**

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

- **包含头文件**

Peripheral_lib/DrvTIMER.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

```
/*  捕捉比较器Capture输入IO设置Port 1.6=TCI1, Port1.7=TCI2 */
DrvCapture_Iport(3);
```

5 芯片 IO 口 GPIO

5.1 函数简介

该部分函数描述 GPIO 的工作模式控制，包含：

--GPIO 口的工作模式控制

--GPIO 口的上拉控制

--GPIO 口的外部中断功能控制

--GPIO 口状态及采样频率控制

序号	函数名称	功能描述
01	DrvGPIO_Open	设置GPIO port 操作模式
02	DrvGPIO_SetBit	设置GPIO pin 为1
03	DrvGPIO_ClrBit	设置GPIO pin 为0.
04	DrvGPIO_GetBit	获取GPIO pin的值
05	DrvGPIO_SetPortBits	设置GPIO port 值
06	DrvGPIO_ClrPortBits	清除输出GPIO port 的值
07	DrvGPIO_GetPortBits	获取输入GPIO port 的值
08	DrvGPIO_IntTrigger	设置GPIO 口中断触发源模式
09	DrvGPIO_ClkGenerator	设置GPIO 采样时钟源
10	DrvGPIO_ClearIntFlag	清除外部中断标志位
11	DrvGPIO_GetIntFlag	读取外部中断标志位
12	DrvGPIO_Close	关闭GPIO 任何引脚的工作模式
13	DrvGPIO_EnableAnalogPin	关闭IO数字功能，开启模拟功能
14	DrvGPIO_PT1_EnableINPUT	使能PT1口对应引脚输入模式功能
15	DrvGPIO_PT1_DisableINPUT	关闭PT1口对应引脚输入模式功能
16	DrvGPIO_PT1_EnablePullHigh	使能PT1口对应引脚上拉电阻功能
17	DrvGPIO_PT1_DisablePullHigh	关闭PT1口对应引脚上拉电阻功能
18	DrvGPIO_PT1_EnableOUTPUT	使能PT1口对应引脚输出模式功能
19	DrvGPIO_PT1_DisableOUTPUT	关闭PT1口对应引脚输出模式功能
20	DrvGPIO_PT1_EnableINT	使能PT1口对应引脚外部中断功能
21	DrvGPIO_PT1_DisableINT	关闭PT1口对应引脚外部中断功能
22	DrvGPIO_PT1_IntTriggerPorts	设置PT1外部中断触发边沿
23	DrvGPIO_PT1_IntTriggerBit	设置PT1口的某一位IO pin的外部中断触发沿
24	DrvGPIO_PT1_GetIntFlag	清除PT1外部中断请求标志位
25	DrvGPIO_PT1_ClearIntFlag	读取PT1外部中断请求标志位
26	DrvGPIO_PT1_GetPortBits	读取PT1口输入状态值
27	DrvGPIO_PT1_SetPortBits	设置PT1口对应引脚输出1

28	DrvGPIO_PT1_ClrPortBits	设置PT1口对应引脚输出0
29	DrvGPIO_PT2_EnableINPUT	使能PT2口对应引脚输入模式功能
30	DrvGPIO_PT2_DisableINPUT	关闭PT2口对应引脚输入模式功能
31	DrvGPIO_PT2_EnablePullHigh	使能PT2口对应引脚上拉电阻功能
32	DrvGPIO_PT2_DisablePullHigh	关闭PT2口对应引脚上拉电阻功能
33	DrvGPIO_PT2_EnableOUTPUT	使能PT2口对应引脚输出模式功能
34	DrvGPIO_PT2_DisableOUTPUT	关闭PT2口对应引脚输出模式功能
35	DrvGPIO_PT2_EnableINT	使能PT2口对应引脚外部中断功能
36	DrvGPIO_PT2_DisableINT	关闭PT2口对应引脚外部中断功能
37	DrvGPIO_PT2_IntTriggerPorts	设置PT2外部中断触发边沿
38	DrvGPIO_PT2_IntTriggerBit	设置PT2口的某一位IO pin的外部中断触发沿
39	DrvGPIO_PT2_GetIntFlag	清除PT2外部中断请求标志位
40	DrvGPIO_PT2_ClearIntFlag	读取PT2外部中断请求标志位
41	DrvGPIO_PT2_GetPortBits	读取PT2口输入状态值
42	DrvGPIO_PT2_SetPortBits	设置PT2口对应引脚输出1
43	DrvGPIO_PT2_ClrPortBits	设置PT2口对应引脚输出0
44	DrvGPIO_PT3_EnableINPUT	使能PT3口对应引脚输入模式功能
45	DrvGPIO_PT3_DisableINPUT	关闭PT3口对应引脚输入模式功能
46	DrvGPIO_PT3_EnablePullHigh	使能PT3口对应引脚上拉电阻功能
47	DrvGPIO_PT3_DisablePullHigh	关闭PT3口对应引脚上拉电阻功能
48	DrvGPIO_PT3_EnableOUTPUT	使能PT3口对应引脚输出模式功能
49	DrvGPIO_PT3_DisableOUTPUT	关闭PT3口对应引脚输出模式功能
50	DrvGPIO_PT3_GetPortBits	读取PT3口输入状态值
51	DrvGPIO_PT3_SetPortBits	设置PT3口对应引脚输出1
52	DrvGPIO_PT3_ClrPortBits	设置PT3口对应引脚输出0

5.2 内部定义常量

E_DRVGPIO_PORT

标识符	数值	功能意义
E_PT0	0	定义PT0
E_PT1	1	定义PT1
E_PT2	2	定义PT2
E_PT3	3	定义PT3
E_PT4	4	定义PT4

E_DRVGPIO_IO

标识符	数值	功能意义
E_IO_INPIT	0	设置GPIO作为输入模式
E_IO_OUTPUT	1	设置GPIO作为输出模式
E_IO_PullHigh	2	使能上拉电阻功能
E_IO_ IntEnable	3	使能外部中断功能

E_DRVGPIO_IntTriMethod

标识符	数值	功能意义
E_DisableGPIOInt	0	关闭GPIO中断功能
E_P_Edge	1	上升沿触发
E_N_Edge	2	下降沿触发
E_Chang_Level	3	电平变化触发
E_LLTri	4	低电平触发
E_LHTri	5	高电平触发
E_LLTri	6	低低电平触发
E_LHTri	7	高电平触发

E_DRVGPIO_CLOCK_SOURCE

标识符	数值	功能意义
E_HS_CK	0	设置IO 采样时钟源为HS_CK
E_LS_CK	1	设置IO 采样时钟源为LS_CK

5.3 函数说明

5.3.1 DrvGPIO_Open

- **函数**

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

- **函数功能**

  设置GPIO PT1~PT3任何一位IO引脚的工作模式，可选工作模式有输入/输出/外部中断/电阻上拉。

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

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

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

- **输入参数**

  port [in]：代表GPIO port. 它的值可以是1~4.

  1：PT1 2：PT2

  3：PT3 4：Reserved.

  i32Bit [in] ：代表 GPIO 任何一位IO 口引脚，对应位的值为1表示打开对应IO引脚工作模式，为0时不做设置；

  设置值范围是 0~255.

  mode [in] ：代表GPIO 每一位IO 口的工作模式,. 设置值范围 ：0~3

  0：输入模式 1：输出模式

  2：内部上拉 3：使能外部中断

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /* 设置PT1.0 作为输出模式及 PT1.1 作为输入模式*/

  DrvGPIO_Open(E_PT1, 0x01, E_IO_OUTPUT); //PT1.0打开输出模式

  DrvGPIO_Open(E_PT1, 0x02, E_IO_INPUT);//PT1.1打开输入模式

5.3.2 DrvGPIO_SetBit

- 函数

  unsigned int DrvGPIO_SetBit (E_DRVGPIO_PORT uport, unsigned int i32Bit)

- 函数功能

  设置PT1~PT3对应的IO 口输出1.

  设置GPIO的输出状态寄存器0x40804[7:0]/0x40814[7:0]/0x40824[7:0]

- 输入参数

  uport [in]： 代表GPIO port. 设定值范围是1~4

  1：PT1 2：PT2

  3：PT3 4：Rsv.

  i32Bit [in]： 代表GPIO的每一位IO 口，设定值范围是0~7.

- 包含头文件

  Peripheral_lib/DrvGPIO.h

- 函数返回值

  0：设置成功

  其他：设置失败

- 函数用法

```
/* 设置PT1.0 作为输出模式*/
DrvGPIO_Open(E_PT1, 1, E_IO_OUTPUT);
/* 设定PT1.0输出1 */
DrvGPIO_SetBit(E_PT1, 0);
```

5.3.3 DrvGPIO_ClrBit

- 函数

  unsigned int DrvGPIO_ClrBit (E_DRVGPIO_PORT uport, unsigned int i32Bit)

- 函数功能

  设置PT1~PT3对应任何一位的IO口输出状态为 0.

  清零GPIO的输出状态寄存器0x40804[7:0] / 0x40814[7:0] / 0x40824[7:0]

- 输入参数

  uport [in]： 代表GPIO port. 它的设置值范围是1~4.

  1：PT1 2：PT2

  3：PT3 4：Rsv.

  i32Bit [in]： 代表GPIO的每一位IO 口，设定值范围是0~7.

- 包含头文件

  Peripheral_lib/DrvGPIO.h

- 函数返回值

  0：设置成功

  0xff000000：设置失败

- 函数用法

```
/* 设定PT1.0输出0 */
DrvGPIO_ClrBit(E_PT1, 0);
```

5.3.4 DrvGPIO_GetBit

- 函数

uint8_t DrvGPIO_GetBit (E_DRVGPIO_PORT port, uint8_t u32Bit)

- 函数功能

读取GPIO PT1~PT3任何一位IO 口的输入状态值.

读取GPIO输入状态寄存器0x40808[7:0]/0x40818[7:0]/0x40828[7:0]

- 输入参数

port [in]： 代表GPIO port. 它的设定值范围是1~4.

1：PT1　2：PT2

3：PT3　4：Rsv.

u32Bit [in]： 代表GPIO port任何一位IO 口，它的设定值范围是 0、1、2、3、4、5、6、7.

- 包含头文件

Peripheral_lib/DrvGPIO.h

- 函数返回值

0/1：IO pin的输入状态

0xff000000：读取失败

- 函数用法

```
uint32_t  i32Bit数值;
/* 设置PT1.1 作为输入模式，并读取PT1.1的输入状态值*/
DrvGPIO_Open(E_PT1, 1, E_IO_INPUT);
i32Bit数值 = DrvGPIO_GetBit(E_PT1, 1);
if (u32Bit数值 == 1)
{
   printf(“PT1-1 pin status is high.\n”); }
else
{
   printf(“PT1-1 pin status is low.\n”); }
```

5.3.5 DrvGPIO_SetPortBits

- 函数

unsigned int DrvGPIO_SetPortBits (E_DRVGPIO_PORT uport, unsigned int ui32Data)

- 函数功能

设置GPIO PT1~PT3 对应IO口的输出状态.

设置GPIO的输出状态寄存器0x40804[7:0]/0x40814[7:0]/0x40824[7:0]

- **输入参数**

uport [in]：代表GPIO port. 设定值范围是1~4.

1：PT1　2：PT2

3：PT3　4：Rsv.

i32Data [in]：设置对应位IO口，对应位为1则会被置1，对应位为0则会被置0，设定值范围0~0xFF.

- **包含头文件**

Peripheral_lib/DrvGPIO.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/* 设定PT1.1、PT1.4为1，所以设定参数0x12 */

DrvGPIO_SetPortBits(E_PT1, 0x12);

5.3.6 DrvGPIO_ClrPortBits

- **函数**

unsigned int DrvGPIO_ClrPortBits (E_DRVGPIO_PORT uport, unsigned int ui32Data)

- **函数功能**

清除GPIO PT1~PT3 对应位IO口输出状态值.

清零GPIO的输出状态寄存器0x40804[7:0] / 0x40814[7:0] / 0x40824[7:0]

- **输入参数**

uport [in]：代表GPIO port，设定值范围是1~4.

1：PT1　2：PT2

3：PT3　4：Rsv.

i32Data [in]：代表对应位的IO口，对应位为1才会被置0，设定范围是0~0xFF.

- **包含头文件**

Peripheral_lib/DrvGPIO.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/* 清除PT1.1/PT1.4的输出为0，设定输入参数 0x12 */

DrvGPIO_ClrPortBits(E_PT1, 0x12);

5.3.7 DrvGPIO_GetPortBits

- **函数**

uint32_t DrvGPIO_GetPortBits (E_DRVGPIO_PORT port)

- **函数功能**

  读取GPIO PT1~PT3输入状态值. 读取GPIO的输出状态寄存器0x40808[7 :0] / 0x40818[7 :0] / 0x40828[7 :0]

- **输入参数**

  port [in]： 代表GPIO port，设定值范围是1~4.

  1：PT1　2：PT2

  3：PT3　4：Rsv.

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  0 ~ 0xFF：读取成功，待读取GPIO PORT的输入状态值:

  0xff000000：读取失败

- **函数用法**

  /*读取PT1的输入状态值*/

  uint32_t　i32Port; i32Port = DrvGPIO_GetPortBits(E_PT1);

5.3.8 DrvGPIO_IntTrigger

- **函数**

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

- **函数功能**

  使能GPIO PT1~PT2的外部中断触发沿并设置外部中断的触发沿模式.

  设置GPIO寄存器0x4080C[31:0]/0x4081C[31:0]

- **输入参数**

  port [in]： 代表GPIO port.设定值范围是1~2.

  1：PT1　2：PT2

  u32Bit [in]： 代表GPIO port的每一位IO口，对应位为1表示该位IO被设置，输入0x0时，触发功能无效

  设置值范围 ：0x00~0xFF

  mode [in] : IO口的中断触发模式选择，设定值范围是0~7

  0：关闭IO 外部中断触发　　1：上升沿触发

  2：下降沿触发　　3：电平变化触发

  4：低电平触发　　5：高电平触发

  6：低电平触发　　7：高电平触发.

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /* 设置PT1.0中断触发模式为下降沿触发*/

  DrvGPIO_ClkGenerator(0,1);　　//设置IO 采样频率

DrvGPIO_Open(E_PT1, 0x01, E_IO_ IntEnable); //使能IO外部中断

DrvGPIO_IntTrigger(E_PT1, 0x01, E_N_Edge); //设置中断触发模式

5.3.9 DrvGPIO_ClkGenerator

- **函数**

uint32_t DrvGPIO_ClkGenerator (E_DRVGPIO_CLK_SOURCE uClk, uint32_t uDivider)

- **函数功能**

设置IO采样频率时钟源及时钟分频值.

设置寄存器0x4030C[20:16]

- **输入参数**

uClk [in]： 代表GPIO采样频率时钟源. 设置值范围 : 0~1.

0：高速时钟源（HS_CK）

1：低速时钟源（LS_CK）

uDivider [in] : IO 口的时钟源分频值. 设置值范围 : 0~15.

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

- **包含头文件**

Peripheral_lib/DrvGPIO.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/* 设置IO 采样频率时钟源为高速时钟(HS_CK)及clk/2 .*/

DrvGPIO_ClkGenerator(E_HS_CK, 2); //0x4030C[20]=0b,[19:16]=0010

5.3.10 DrvGPIO_ClearIntFlag

- **函数**

unsigned int DrvGPIO_ClearIntFlag (E_DRVGPIO_PORT port, uint32_t u32Bit)

- **函数功能**

清除GPIO PT1~PT2外部中断标志位；

清零中断寄存器0x40010[7 :0] / 0x40014[7 :0]。

- **输入参数**

port [in] ：代表GPIO，设定范围是1~2

1：PT1　2：PT2

u32Bit [in] ：

代表GPIO port的每一位IO，对应位为1的才会被清零，设定值范围是0x00~0xFF;

设定值的每一位对应一位IO pin，对应位为1的IO 口的标志位就被清零。

- **包含头文件**

Peripheral_lib/DrvGPIO.h

- **函数返回值**

GPIO 外部中断标志位的当前状态值

- **函数用法**

```
/* 清零 PT1.2 interrupt flag */
DrvGPIO_ClearIntFlag(E_PT1, 0x04);
/*清零 PT1.3 interrupt flag*/
DrvGPIO_ClearIntFlag(E_PT1,0x08);
```

5.3.11 DrvGPIO_GetIntFlag

- **函数**

unsigned int DrvGPIO_GetIntFlag(E_DRVGPIO_PORT port)

- **函数功能**

读取对应GPIO PT1~PT2的中断标志位，返回寄存器的值，返回值的对应位为1表示该位的IO 口发生中断，若为0,则表示没有中断产生.

读取中断寄存器0x40010[7 :0] / 0x40014[7 :0]的值。

- **输入参数**

port [in]：代表GPIO port.设定值范围值是1~2

1：PT1　2：PT2

- **包含头文件**

Peripheral_lib/DrvGPIO.h

- **函数返回值**

返回值是 GPIO 的中断标志位值: 0 ~ 0Xff

- **函数用法**

```
/* 读取 PT1 外部中断标志位 */
unsigned char  flag；flag=DrvGPIO_GetIntFlag(E_PT1);
```

5.3.12 DrvGPIO_Close

- **函数**

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

- **函数功能**

关闭GPIO PT1~PT3任何一位IO引脚的工作模式，可选工作模式有输入/输出/外部中断/电阻上拉。

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

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

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

- **输入参数**

port [in]：代表GPIO port. 它的值可以是1~4.

1：PT1　2：PT2

3：PT3　4：Rsv.

i32Bit [in] ：代表 GPIO 任何一位IO 口引脚，对应位的值为1表示关闭对应IO引脚工作模式，为0时不做设置；

设置值范围是 0~255.

mode [in] ：代表GPIO 每一位IO 口的工作模式，设置值范围 ：0~3.

0：输入模式　1：输出模式

2：内部上拉　3：使能外部中断

- **包含头文件**

Peripheral_lib/DrvGPIO.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

```
/* 关闭PT1.0 作为输出模式及 PT1.1 作为输入模式*/
DrvGPIO_Close(E_PT1, 0x01, E_IO_OUTPUT); //关闭PT1.0打开输出模式
DrvGPIO_Close(E_PT1, 0x02, E_IO_INPUT);  // 关闭PT1.1打开输入模式
```

5.3.13 DrvGPIO_EnableAnalogPin

- **函数**

unsigned char DrvGPIO_EnableAnalogPin(short port,unsigned int i32Bit)

- **函数功能**

关闭GPIO任何一位IO引脚的数字工作模式，如输入/输出/外部中断/电阻上拉/中断触发沿，开启IO模拟工作模式。

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

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

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

- **输入参数**

port [in]：代表GPIO port. 设定值范围是1~3.

1：PT1　2：PT2

3：PT3

u32Bit [in]：代表 GPIO 任何一位IO口引脚，对应位的值为1表示关闭对应IO引脚工作模式，为0时不做设置；

设置值范围是 0~0xFF.

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  0：设置成功

  1：设置失败

- **函数用法**

```
/* 关闭PT3.1/PT3.3/PT3.5/PT3.7数字功能*/
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

- **函数**

  void DrvGPIO_PT1_EnableINPUT(short int ubit)

- **函数功能**

  使能GPIO PT1任何一位IO引脚的输入模式

  设置PT1寄存器0x40804[23:16]

- **输入参数**

  ubit [in]：代表 GPIO 任何一位IO 口引脚，对应位的值为1表示打开对应IO引脚输入模式，为0时不做设置；

  设置值范围是 0~0xff ;

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

```
/* 设置PT1.0/PT1.1 作为输入模式*/
DrvGPIO_PT1_EnableINPUT(0x01| 0x02); //PT1.0/PT1.1打开输入模式
```

5.3.15 DrvGPIO_PT1_DisableINPUT

- **函数**

  void DrvGPIO_PT1_DisableINPUT(short int ubit)

- **函数功能**

  关闭GPIO PT1任何一位IO引脚的输入模式

  设置PT1寄存器0x40804[23:16]

- **输入参数**

  ubit [in] :代表 GPIO 任何一位IO 口引脚，对应位的值为1表示关闭对应IO引脚输入模式，为0时不做设置；设置值范围是 0~0xff ;

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭PT1.0/PT1.1 作为输入模式*/

  DrvGPIO_PT1_DisableINPUT(0x01|0x02); //PT1.0/PT1.1关闭输入模式

5.3.16 DrvGPIO_PT1_EnablePullHigh

- **函数**

  void DrvGPIO_PT1_EnablePullHigh(short int ubit)

- **函数功能**

  使能GPIO PT1任何一位IO引脚的上拉电阻

  设置PT1寄存器0x40800[23:16]

- **输入参数**

  ubit [in] ：代表 GPIO 任何一位IO 口引脚，对应位的值为1表示打开对应IO引脚上拉电阻，为0时不做设置；设置值范围是 0~0xff ;

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 使能PT1.0/PT1.1 上拉电阻*/

  DrvGPIO_PT1_EnablePullHigh(0x01|0x02); //PT1.0/PT1.1打开上拉电阻

5.3.17 DrvGPIO_PT1_DisablePullHigh

- **函数**

  void DrvGPIO_PT1_DisablePullHigh(short int ubit)

- **函数功能**

  关闭GPIO PT1任何一位IO引脚的上拉电阻

  设置PT1寄存器0x40800[23:16]

- **输入参数**

  ubit [in]： 代表 GPIO 任何一位IO 口引脚，对应位的值为1表示关闭对应IO引脚上拉电阻，为0时不做设置；设置值范围是 0~0xff ;

- **包含头文件**

Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭PT1.0/PT1.1 上拉电阻*/

  DrvGPIO_PT1_ DisablePullHigh (0x01|0x02); //PT1.0/PT1.1关闭上拉电阻

5.3.18 DrvGPIO_PT1_EnableOUTPUT

- **函数**

  void DrvGPIO_PT1_EnableOUTPUT(short int ubit)

- **函数功能**

  使能GPIO PT1任何一位IO引脚的输出模式

  设置PT1寄存器0x40800[7:0]

- **输入参数**

  ubit [in]：代表 GPIO 任何一位IO 口引脚，对应位的值为1表示打开对应IO引脚输出模式，为0时不做设置；

  设置值范围是 0~0xff ;

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 使能PT1.0/PT1.1 输出模式*/

  DrvGPIO_PT1_EnableOUTPUT(0x01|0x02); //PT1.0/PT1.1打开输出模式

5.3.19 DrvGPIO_PT1_DisableOUTPUT

- **函数**

  void DrvGPIO_PT1_DisableOUTPUT(short int ubit)

- **函数功能**

  关闭GPIO PT1任何一位IO引脚的输出模式

  设置PT1寄存器0x40800[7:0]

- **输入参数**

  ubit [in]：代表 GPIO 任何一位IO 口引脚，对应位的值为1表示关闭对应IO引脚输出模式，为0时不做设置；

  设置值范围是 0~0xff ;

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

```
/* 关闭PT1.0/PT1.1 输出模式*/
DrvGPIO_PT1_DisableOUTPUT(0x01|0x02); //PT1.0/PT1.1关闭输出模式
```

5.3.20 DrvGPIO_PT1_EnableINT

- **函数**

void DrvGPIO_PT1_EnableINT(short int ubit)

- **函数功能**

使能GPIO PT1任何一位IO引脚的外部中断功能。

设置PT1寄存器0x40010[23:16]

- **输入参数**

ubit [in]：代表 GPIO 任何一位IO 口引脚，对应位的值为1表示打开对应IO引脚外部中断功能，为0时不做设置；设置值范围是 0~0xFF；

- **包含头文件**

Peripheral_lib/DrvGPIO.h

- **函数返回值**

无

- **函数用法**

```
/* 使能PT1.0/PT1.1 外部中断功能*/
DrvGPIO_PT1_EnableINT(0x01|0x02); //PT1.0/PT1.1打开外部中断功能
```

5.3.21 DrvGPIO_PT1_DisableINT

- **函数**

void DrvGPIO_PT1_DisableINT(short int ubit)

- **函数功能**

关闭GPIO PT1任何一位IO引脚的外部中断功能。

设置PT1寄存器0x40010[23:16]

- **输入参数**

ubit [in]：代表 GPIO 任何一位IO 口引脚，对应位的值为1表示关闭对应IO引脚外部中断功能，为0时不做设置；设置值范围是 0~0xFF；

- **包含头文件**

Peripheral_lib/DrvGPIO.h

- **函数返回值**

无

- **函数用法**

```
/* 关闭PT1.0/PT1.1 外部中断功能*/
DrvGPIO_PT1_DisableINT(0x01|0x02); //PT1.0/PT1.1关闭外部中断功能
```

5.3.22 DrvGPIO_PT1_IntTriggerPorts

- **函数**

  void DrvGPIO_PT1_IntTriggerPorts(uint32_t i32Bit, uint32_t mode)

- **函数功能**

  使能GPIO PT1的外部中断触发沿并设置外部中断的触发沿模式.

  设置GPIO寄存器0x4080C[31:0]

- **输入参数**

  u32Bit [in] :

  代表GPIO port的每一位IO 口，对应位为1表示该位IO被设置，输入0x0时，触发功能无效，设定值是 0~0xFF.

  mode [in] : IO口的中断触发模式选择，设定值范围是0~7

  0：关闭IO 外部中断触发　　1：上升沿触发　　2：下降沿触发　　3：电平变化触发

  4：低电平触发　　5：高电平触发　　6：低电平触发　　7：高电平触发.

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 设置PT1.0中断触发模式为下降沿触发*/

  DrvGPIO_ClkGenerator(0,1); //设置IO 采样频率

  DrvGPIO_PT1_EnableINT(0x1); //使能IO外部中断

  DrvGPIO_PT1_IntTriggerPorts(0x1, E_N_Edge); //设置中断触发模式

5.3.23 DrvGPIO_PT1_IntTriggerBit

- **函数**

  void DrvGPIO_PT1_IntTriggerBit(uint32_t i32Bit, uint32_t mode)

- **函数功能**

  使能GPIO PT1被选中引脚的外部中断触发沿并设置外部中断的触发沿模式.

  设置GPIO寄存器0x4080C[31:0]

- **输入参数**

  u32Bit [in]： 输入范围为0~7，代表GPIO port的bit7~bit0，选中的引脚才被设置.

  mode [in] : IO口的中断触发模式选择，设定值范围是0~7

  0：关闭IO 外部中断触发　　1：上升沿触发　　2：下降沿触发　　3：电平变化触发

  4：低电平触发　　5：高电平触发　　6：低电平触发　　7：高电平触发.

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

```
/* 设置PT1.0中断触发模式为下降沿触发*/
DrvGPIO_ClkGenerator(0,1); //设置IO 采样频率
DrvGPIO_PT1_EnableINT(0x1);  //使能IO外部中断
DrvGPIO_PT1_IntTriggerPorts(0x1, E_N_Edge);    //设置中断触发模式
```

5.3.24 DrvGPIO_PT1_GetIntFlag

- **函数**

unsigned char DrvGPIO_PT1_GetIntFlag(void)

- **函数功能**

读取对应GPIO PT1的中断标志位，返回寄存器的值，返回值的对应位为1表示该位的IO 口发生中断，若为0,则表示没有中断产生.

读取中断寄存器0x40010[7 :0]的值。

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvGPIO.h

- **函数返回值**

返回值是 PT1 的中断标志位值: 0 ~ 0xff

- **函数用法**

```
 /* 读取 PT1 外部中断标志位 */
unsigned char   flag；flag=DrvGPIO_PT1_GetIntFlag();
```

5.3.25 DrvGPIO_PT1_ClearIntFlag

- **函数**

void DrvGPIO_PT1_ClearIntFlag(short int uint32)

- **函数功能**

清除GPIO PT1外部中断标志位；

清零中断寄存器0x40010[7 :0] 。

- **输入参数**

u32Bit [in] :

代表GPIO port的每一位IO，对应位为1的才会被清零，设定值范围是0x00~0xFF;

设定值的每一位对应一位IO pin，对应位为1的IO 口的标志位就被清零。

- **包含头文件**

Peripheral_lib/DrvGPIO.h

- **函数返回值**

无

- **函数用法**

```
/* 清零 PT1.2 interrupt flag */
DrvGPIO_PT1_ClearIntFlag(0x04);
/*清零 PT1.3 interrupt flag*/
DrvGPIO_PT1_ClearIntFlag(0x08);
```

5.3.26 DrvGPIO_PT1_GetPortBits

- **函数**

  unsigned char DrvGPIO_PT1_GetPortBits (void)

- **函数功能**

  读取GPIO PT1输入状态值. 读取GPIO的输入状态寄存器0x40808[7 :0]

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  0 ~ 0xFF：待读取GPIO PORT的输入状态值:

- **函数用法**

```
/*读取PT1的输入状态值*/
uint32_t    i32Port; i32Port = DrvGPIO_PT1_GetPortBits();
```

5.3.27 DrvGPIO_PT1_SetPortBits

- **函数**

  void DrvGPIO_PT1_SetPortBits (unsigned char ui32Data)

- **函数功能**

  设置GPIO PT1对应IO口的输出状态.

  设置GPIO的输出状态寄存器0x40804[7:0]

- **输入参数**

  i32Data [in]： 设定值范围0~0xFF.bit7~bit0对应每一位IO PIN，对应位为1则会被置1，对应位为0则会被置0

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

```
/* 设定PT1.2、PT1.4为1，所以设定参数0x14 */
DrvGPIO_PT1_SetPortBits(0x14);
```

5.3.28 DrvGPIO_PT1_ClrPortBits

- **函数**

  void DrvGPIO_PT1_ClrPortBits (unsigned int ui32Data)

- **函数功能**

  清除GPIO PT1 对应位IO口输出状态值.

  清零GPIO的输出状态寄存器0x40804[7:0]

- **输入参数**

  i32Data [in]： 设定范围是0~0xFF. bit7~bit0对应每一位IO PIN，对应位为1输出才被置0，

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 清除PT1.1/PT1.4的输出为0，设定输入参数 0x12 */

  DrvGPIO_PT1_ClrPortBits(0x12);

5.3.29 DrvGPIO_PT2_EnableINPUT

- **函数**

  void DrvGPIO_PT2_EnableINPUT(short int ubit)

- **函数功能**

  使能GPIO PT2任何一位IO引脚的输入模式

  设置PT2寄存器0x40814[23:16]

- **输入参数**

  ubit [in]： 代表 GPIO 任何一位IO 口引脚，对应位的值为1表示打开对应IO引脚输入模式，为0时不做设置；

  设置值范围是 0~0xff ;

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 设置PT2.0/PT2.1 作为输入模式*/

  DrvGPIO_PT2_EnableINPUT(0x01|0x02); //PT2.0/PT2.1打开输入模式

5.3.30 DrvGPIO_PT2_DisableINPUT

- **函数**

  void DrvGPIO_PT2_DisableINPUT(short int ubit)

- **函数功能**

  关闭GPIO PT2任何一位IO引脚的输入模式

设置PT2寄存器0x40814[23:16]

- **输入参数**

ubit [in]：代表 GPIO 任何一位IO 口引脚，对应位的值为1表示关闭对应IO引脚输入模式，为0时不做设置；

设置值范围是 0~0xff ;

- **包含头文件**

Peripheral_lib/DrvGPIO.h

- **函数返回值**

无

- **函数用法**

```
/* 关闭PT2.0/PT2.1 作为输入模式*/
DrvGPIO_PT2_DisableINPUT(0x01|0x02); //PT2.0/PT2.1关闭输入模式
```

5.3.31 DrvGPIO_PT2_EnablePullHigh

- **函数**

void DrvGPIO_PT2_EnablePullHigh(short int ubit)

- **函数功能**

使能GPIO PT2任何一位IO引脚的上拉电阻

设置PT2寄存器0x40810[23:16]

- **输入参数**

ubit [in]：代表 GPIO 任何一位IO 口引脚，对应位的值为1表示打开对应IO引脚上拉电阻，为0时不做设置；

设置值范围是 0~0xff ;

- **包含头文件**

Peripheral_lib/DrvGPIO.h

- **函数返回值**

无

- **函数用法**

```
/* 使能PT2.0/PT2.1 上拉电阻*/
DrvGPIO_PT2_EnablePullHigh(0x01| 0x02); //PT2.0/PT2.1打开上拉电阻
```

5.3.32 DrvGPIO_PT2_DisablePullHigh

- **函数**

void DrvGPIO_PT2_DisablePullHigh(short int ubit)

- **函数功能**

关闭GPIO PT2任何一位IO引脚的上拉电阻

设置PT2寄存器0x40810[23:16]

- **输入参数**

ubit [in]：代表 GPIO 任何一位IO 口引脚，对应位的值为1表示关闭对应IO引脚上拉电阻，为0时不做设置；

设置值范围是 0~0xff ;

- **包含头文件**

Peripheral_lib/DrvGPIO.h

- **函数返回值**

无

- **函数用法**

```
/* 关闭PT2.0/PT2.1 上拉电阻*/
DrvGPIO_PT2_DisablePullHigh(0x01|0x02); //PT2.0/PT2.1关闭上拉电阻
```

5.3.33 DrvGPIO_PT2_EnableOUTPUT

- **函数**

void DrvGPIO_PT2_EnableOUTPUT(short int ubit)

- **函数功能**

使能GPIO PT2任何一位IO引脚的输出模式

设置PT2寄存器0x40810[7:0]

- **输入参数**

ubit [in]：代表 GPIO 任何一位IO 口引脚，对应位的值为1表示打开对应IO引脚输出模式，为0时不做设置；

设置值范围是 0~0xff ;

- **包含头文件**

Peripheral_lib/DrvGPIO.h

- **函数返回值**

无

- **函数用法**

```
/* 使能PT2.0/PT2.1 输出模式*/
DrvGPIO_PT2_EnableOUTPUT(0x01|0x02); //PT2.0/PT2.1打开输出模式
```

5.3.34 DrvGPIO_PT2_DisableOUTPUT

- **函数**

void DrvGPIO_PT2_DisableOUTPUT(short int ubit)

- **函数功能**

关闭GPIO PT2任何一位IO引脚的输出模式

设置PT2寄存器0x40810[7:0]

- **输入参数**

ubit [in]：代表 GPIO 任何一位IO 口引脚，对应位的值为1表示关闭对应IO引脚输出模式，为0时不做设置；

设置值范围是 0~0xff ;

- **包含头文件**

Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭PT2.0/PT2.1 输出模式*/

  DrvGPIO_PT2_DisableOUTPUT(0x01|0x02); //PT2.0/PT2.1关闭输出模式

5.3.35 DrvGPIO_PT2_EnableINT

- **函数**

  void DrvGPIO_PT2_EnableINT(short int ubit)

- **函数功能**

  使能GPIO PT2任何一位IO引脚的外部中断功能。

  设置PT2寄存器0x40014[23:16]

- **输入参数**

  ubit [in] :代表 GPIO 任何一位IO 口引脚，对应位的值为1表示打开对应IO引脚外部中断功能，为0时不做设置；设置值范围是 0~0xFF；

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 使能PT2.0/PT2.1 外部中断功能*/

  DrvGPIO_PT2_EnableINT(0x01|0x02); //PT2.0/PT2.1打开外部中断功能

5.3.36 DrvGPIO_PT2_DisableINT

- **函数**

  void DrvGPIO_PT2_DisableINT(short int ubit)

- **函数功能**

  关闭GPIO PT2任何一位IO引脚的外部中断功能。

  设置PT2寄存器0x40014[23:16]

- **输入参数**

  ubit [in]： 代表 GPIO 任何一位IO 口引脚，对应位的值为1表示关闭对应IO引脚外部中断功能，为0时不做设置；设置值范围是 0~0xFF；

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

```
/* 关闭PT2.0/PT2.1 外部中断功能*/
DrvGPIO_PT2_DisableINT(0x01|0x02); //PT2.0/PT2.1关闭外部中断功能
```

5.3.37 DrvGPIO_PT2_IntTriggerPorts

- **函数**

  void DrvGPIO_PT2_IntTriggerPorts(uint32_t i32Bit, uint32_t mode)

- **函数功能**

  使能GPIO PT2的外部中断触发沿并设置外部中断的触发沿模式.

  设置GPIO寄存器0x4081C[31:0]

- **输入参数**

  u32Bit [in] :

  代表GPIO port的每一位IO 口，对应位为1表示该位IO被设置，输入0x0时，触发功能无效，设定值是 0~0xFF.

  mode [in] : IO口的中断触发模式选择，设定值范围是0~7

  0：关闭IO 外部中断触发　　1：上升沿触发　　2：下降沿触发　　3：电平变化触发

  4：低电平触发　　5：高电平触发　　6：低电平触发　　7：高电平触发.

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

```
/* 设置PT2.0中断触发模式为下降沿触发*/
DrvGPIO_ClkGenerator(0,1); //设置IO 采样频率
DrvGPIO_PT2_EnableINT(0x1);  //使能IO外部中断
DrvGPIO_PT2_IntTriggerPorts(0x1, E_N_Edge);    //设置中断触发模式
```

5.3.38 DrvGPIO_PT2_IntTriggerBit

- **函数**

  void DrvGPIO_PT2_IntTriggerBit(uint32_t i32Bit, uint32_t mode)

- **函数功能**

  使能GPIO PT2被选中引脚的外部中断触发沿并设置外部中断的触发沿模式.

  设置GPIO寄存器0x4081C[31:0]

- **输入参数**

  u32Bit [in]： 输入范围为0~7，代表GPIO port的bit7~bit0，选中的引脚才被设置.

  mode [in] : IO口的中断触发模式选择，设定值范围是0~7

  0：关闭IO 外部中断触发　　1：上升沿触发　　2：下降沿触发　　3：电平变化触发

  4：低电平触发　　5：高电平触发　　6：低电平触发　　7：高电平触发.

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 设置PT2.0中断触发模式为下降沿触发*/

  DrvGPIO_ClkGenerator(0,1); //设置IO 采样频率

  DrvGPIO_PT2_EnableINT(0x1); //使能IO外部中断

  DrvGPIO_PT2_IntTriggerPorts(0x1, E_N_Edge); //设置中断触发模式

5.3.39 DrvGPIO_PT2_GetIntFlag

- **函数**

  unsigned char DrvGPIO_PT2_GetIntFlag(void)

- **函数功能**

  读取对应GPIO PT2的中断标志位，返回寄存器的值，返回值的对应位为1表示该位的IO 口发生中断，若为0,则表示没有中断产生.

  读取中断寄存器0x40014[7 :0]的值。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  返回值是 PT2 的中断标志位值: 0 ~ 0Xff

- **函数用法**

  /* 读取 PT2 外部中断标志位 */

  unsigned char flag；flag=DrvGPIO_PT2_GetIntFlag();

5.3.40 DrvGPIO_PT2_ClearIntFlag

- **函数**

  void DrvGPIO_PT2_ClearIntFlag(short int uint32)

- **函数功能**

  清除GPIO PT2外部中断标志位；

  清零中断寄存器0X40014[7 :0] 。

- **输入参数**

  u32Bit [in] :

  代表GPIO port的每一位IO，对应位为1的才会被清零，设定值范围是0x00~0xFF;

  设定值的每一位对应一位IO pin，对应位为1的IO 口的标志位就被清零。

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 清零 PT2.2 interrupt flag */
  DrvGPIO_PT2_ClearIntFlag(0x04);
  /*清零 PT2.3 interrupt flag*/
  DrvGPIO_PT2_ClearIntFlag(0x08);

5.3.41 DrvGPIO_PT2_GetPortBits

- **函数**

  unsigned char DrvGPIO_PT2_GetPortBits (void)

- **函数功能**

  读取GPIO PT2输入状态值. 读取GPIO的输入状态寄存器0x40818[7 :0]

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  0 ~ 0xFF：待读取GPIO PORT的输入状态值:

- **函数用法**

  /*读取PT2的输入状态值*/

  uint32_t i32Port; i32Port = DrvGPIO_PT2_GetPortBits();

5.3.42 DrvGPIO_PT2_SetPortBits

- **函数**

  void DrvGPIO_PT2_SetPortBits (unsigned char ui32Data)

- **函数功能**

  设置GPIO PT2对应IO口的输出状态.

  设置GPIO的输出状态寄存器0x40814[7:0]

- **输入参数**

  i32Data [in]： 设定值范围0~0xFF.bit7~bit0对应每一位IO PIN，对应位为1则会被置1，对应位为0则会被置0

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 设定PT2.2、PT2.4为1，所以设定参数0x14 */

  DrvGPIO_PT2_SetPortBits(0x14);

5.3.43 DrvGPIO_PT2_ClrPortBits

- **函数**

  void DrvGPIO_PT2_ClrPortBits (unsigned int ui32Data)

- **函数功能**

  清除GPIO PT2 对应位IO口输出状态值.

  清零GPIO的输出状态寄存器0x40814[7:0]

- **输入参数**

  i32Data [in]： 设定范围是0~0xFF. bit7~bit0对应每一位IO PIN，对应位为1输出才被置0，

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 清除PT2.1/PT2.4的输出为0，设定输入参数 0x12 */

  DrvGPIO_PT2_ClrPortBits(0x12);

5.3.44 DrvGPIO_PT3_EnableINPUT

- **函数**

  void DrvGPIO_PT3_EnableINPUT(short int ubit)

- **函数功能**

  使能GPIO PT3任何一位IO引脚的输入模式

  设置PT3寄存器0x40824[23:16]

- **输入参数**

  ubit [in]： 代表 GPIO 任何一位IO 口引脚，对应位的值为1表示打开对应IO引脚输入模式，为0时不做设置；

  设置值范围是 0~0xff ;

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 设置PT3.0/PT3.1 作为输入模式*/

  DrvGPIO_PT3_EnableINPUT (0x01|0x02); //PT3.0/PT3.1打开输入模式

5.3.45 DrvGPIO_PT3_DisableINPUT

- **函数**

  void DrvGPIO_PT3_DisableINPUT(short int ubit)

- **函数功能**

  关闭GPIO PT3任何一位IO引脚的输入模式

  设置PT3寄存器0x40824[23:16]

- **输入参数**

  ubit [in]：代表 GPIO 任何一位IO 口引脚，对应位的值为1表示关闭对应IO引脚输入模式，为0时不做设置；

  设置值范围是 0~0xff ;

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭PT3.0/PT3.1 作为输入模式*/

  DrvGPIO_PT3_DisableINPUT(0x01|0x02); //PT3.0/PT3.1关闭输入模式

5.3.46 DrvGPIO_PT3_EnablePullHigh

- **函数**

  void DrvGPIO_PT3_EnablePullHigh(short int ubit)

- **函数功能**

  使能GPIO PT3任何一位IO引脚的上拉电阻

  设置PT3寄存器0x40820[23:16]

- **输入参数**

  ubit [in]：代表 GPIO 任何一位IO 口引脚，对应位的值为1表示打开对应IO引脚上拉电阻，为0时不做设置；

  设置值范围是 0~0xff ;

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 使能PT3.0/PT3.1 上拉电阻*/

  DrvGPIO_PT3_EnablePullHigh(0x01|0x02); //PT3.0/PT3.1打开上拉电阻

5.3.47 DrvGPIO_PT3_DisablePullHigh

- **函数**

  void DrvGPIO_PT3_DisablePullHigh(short int ubit)

- **函数功能**

  关闭GPIO PT3任何一位IO引脚的上拉电阻

  设置PT3寄存器0x40820[23:16]

- **输入参数**

  ubit [in]：代表 GPIO 任何一位IO 口引脚，对应位的值为1表示关闭对应IO引脚上拉电阻，为0时不做设置；设置值范围是 0~0xff ;

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭PT3.0/PT3.1 上拉电阻*/

  DrvGPIO_PT3_DisablePullHigh(0x01|0x02); //PT3.0/PT3.1关闭上拉电阻

5.3.48 DrvGPIO_PT3_EnableOUTPUT

- **函数**

  void DrvGPIO_PT3_EnableOUTPUT(short int ubit)

- **函数功能**

  使能GPIO PT3任何一位IO引脚的输出模式

  设置PT3寄存器0x40820[7:0]

- **输入参数**

  ubit [in]：代表 GPIO 任何一位IO 口引脚，对应位的值为1表示打开对应IO引脚输出模式，为0时不做设置；设置值范围是 0~0xff ;

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 使能PT3.0/PT3.1 输出模式*/

  DrvGPIO_PT3_EnableOUTPUT(0x01|0x02); //PT3.0/PT3.1打开输出模式

5.3.49 DrvGPIO_PT3_DisableOUTPUT

- **函数**

  void DrvGPIO_PT3_DisableOUTPUT(short int ubit)

- **函数功能**

  关闭GPIO PT3任何一位IO引脚的输出模式

  设置PT3寄存器0x40820[7:0]

- **输入参数**

  ubit [in]：代表 GPIO 任何一位IO 口引脚，对应位的值为1表示关闭对应IO引脚输出模式，为0时不做设置；设置值范围是 0~0xff ;

- 包含头文件

  Peripheral_lib/DrvGPIO.h

- 函数返回值

  无

- 函数用法

  /* 关闭PT3.0/PT3.1 输出模式*/

  DrvGPIO_PT3_DisableOUTPUT(0x01|0x02); //PT3.0/PT3.1关闭输出模式

5.3.50 DrvGPIO_PT3_GetPortBits

- 函数

  unsigned char DrvGPIO_PT3_GetPortBits (void)

- 函数功能

  读取GPIO PT3输入状态值. 读取GPIO的输入状态寄存器0x40828[7 :0]

- 输入参数

  无

- 包含头文件

  Peripheral_lib/DrvGPIO.h

- 函数返回值

  0 ~ 0xFF：待读取GPIO PORT的输入状态值:

- 函数用法

  /*读取PT3的输入状态值*/

  uint32_t i32Port; i32Port = DrvGPIO_PT3_GetPortBits();

5.3.51 DrvGPIO_PT3_SetPortBits

- 函数

  void DrvGPIO_PT3_SetPortBits (unsigned char ui32Data)

- 函数功能

  设置GPIO PT3对应IO口的输出状态.

  设置GPIO的输出状态寄存器0x40824[7:0]

- 输入参数

  i32Data [in]： 设定值范围0~0xFF.bit7~bit0对应每一位IO PIN，对应位为1则会被置1，对应位为0则会被置0

- 包含头文件

  Peripheral_lib/DrvGPIO.h

- 函数返回值

  无

- 函数用法

  /* 设定PT3.2、PT3.4为1，所以设定参数0x14 */

  DrvGPIO_PT3_SetPortBits(0x14);

5.3.52 DrvGPIO_PT3_ClrPortBits

- **函数**

  void DrvGPIO_PT3_ClrPortBits (unsigned int ui32Data)

- **函数功能**

  清除GPIO PT3 对应位IO口输出状态值.

  清零GPIO的输出状态寄存器0x40824[7:0]

- **输入参数**

  i32Data [in]：设定范围是0~0xFF. bit7~bit0对应每一位IO PIN，对应位为1输出才被置0，

- **包含头文件**

  Peripheral_lib/DrvGPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 清除PT3.1/PT3.4的输出为0，设定输入参数 0x12 */

  DrvGPIO_PT3_ClrPortBits(0x12);

6 模数转换器 ADC

6.1 函数简介

该部分函数描述ADC 系统的控制，包含：

--ADC的信号输入端口与参考输入端口的配置与切换

--ADC放大倍数的设置

--ADC中断配置

--ADC转换值的读取

序号	函数名称	功能描述
01	DrvADC_PInputChannel	ADC正端信号输入源设置
02	DrvADC_NInputChannel	ADC负端信号输入源设置
03	DrvADC_SetADCInputChannel	ADC正负端信号输入源设置
04	DrvADC_InputSwitch	ADC输入端短路开关控制
05	DrvADC_RefInputShort	ADC参考电压输入端短路开关控制
06	DrvADC_SetPGA	ADC输入信号放大倍数PGA设置
07	DrvADC_ADGain	ADC输入信号放大倍数ADGain设置
08	DrvADC_Gain	ADC输入信号整体放大倍数设置
09	DrvADC_DCoffset	ADC输入零点平移(DC offset)设置
10	DrvADC_RefVoltage	ADC参考电压输入设置

11	DrvADC_FullRefRange	ADC参考电压放大倍数设置
12	DrvADC_OSR	ADC转换输出率OSR设置
13	DrvADC_ClkEnable	开启ADC时钟源
14	DrvADC_ClkDisable	关闭ADC时钟源
15	DrvADC_FastChopper	ADC Fast-chopper模式设置
16	DrvADC_CombFilter	梳状滤波器开启控制
17	DrvADC_EnableInt	ADC中断开启
18	DrvADC_DisableInt	ADC中断关闭
19	DrvADC_ReadIntFlag	读取ADC中断标志位
20	DrvADC_ClearIntFlag	清除ADC中断标志位
21	DrvADC_Enable	开启ADC
22	DrvADC_Disable	关闭ADC
23	DrvADC_GetConversionData	读取ADC的A/D转换值

6.2 内部定义常量

E_ADC_INPUT_CHANNEL

标识符	数值	函数功能
ADC_Input_AIO0	0	信号输入端
ADC_Input_AIO1	1	信号输入端
ADC_Input_AIO2	2	信号输入端
ADC_Input_AIO3	3	信号输入端
REFO_I	4	信号输入端
OPOI	5	信号输入端
TSP0	6	信号输入端
TSP1	7	信号输入端
DAOI	8	信号输入端
VDDA/VSSA	9	信号输入端

E_ADC_REFV

标识符	数值	函数功能
External	0	外部输入源
Internal	1	使能缓冲器并使用内部源

E_ADC_PGA & E_ADC_ADGN

标识符	数值	函数功能	标识符	数值	函数功能
ADC_PGA_Disable	0	Disable PGA	ADC_ADGN_1	0	ADGN=1
ADC_PGA_8	1	PGA=8	ADC_ADGN_2	1	ADGN=2
ADC_PGA_16	3	PGA=16	ADC_ADGN_RESER	2	Reserve
ADC_PGA_32	7	PGA=32	ADC_ADGN_4	3	ADGN=4

E_ADC_SIGNAL_SHORT

标识符	数值	函数功能
OPEN	0	ADC信号输入短路开关断开
SHORT	1	ADC信号输入短路开关闭合

E_ADC_VRPS_REF_VOLTAGE

标识符	数值	函数功能
VDDA	0	参考电压正端输入为 VDDA
AIO2	1	参考电压正端输入为 AIO2
AIO4	2	参考电压正端输入为 AIO4
REF_BUFFER_OUT	3	参考电压正端输入为 REFO_I

E_ADC_VRNS_REF_VOLTAGE

标识符	数值	函数功能
VSSA	0	参考电压负端输入为VSSA
AIO3	1	参考电压负端输入为 AIO3
AIO5	2	参考电压负端输入为 AIO5
REF_BUFFER_OUT	3	参考电压负端输入为 REFO_I

6.3 函数说明

6.3.1 DrvADC_PInputChannel

- **函数**

  unsigned int DrvADC_PInputChannel (E_ADC_INPUT_Channel uINP);

- **函数功能**

  设置ADC 输入信号正向输入端；

  设置寄存器0x41104[7:4].

- **输入参数**

  uINP [in]：代表ADC的正向输入埠选择，设定值范围0~9

0：AIO0, 1：AIO1,

2：AIO2, 3：AIO3,

4：REFO_I, 5：OPOI,

6：TPSP0, 7：TPSP1,

8： DAOI, 9：VDDA;

此处TPS0= TPSP0, TPS1=TPSP1;

- **包含头文件**

  Peripheral_lib/DrvADC.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /* 设定ADC正向输入端为AIO0*/

  DrvADC_PInputChannel(ADC_Input_AIO0);

6.3.2 DrvADC_NInputChannel

- **函数**

  unsigned int DrvADC_NInputChannel (E_ADC_INPUT_Channel uINN);

- **函数功能**

  设置ADC 输入信号负向输入端，设置寄存器0x41104[3:0].

- **输入参数**

  uINN [in]：代表ADC负端输入选择埠，设定范围值0~9.

  0：AIO0, 1：AIO1,

  2：AIO2, 3：AIO3,

  4：REFO_I, 5：OPOI,

  6：TPSN0, 7：TPSN1,

8：DAOI,　　9：VSSA

此处TPS0=TPSN0,TPS1=TPSN1；

- **包含头文件**

Peripheral_lib/DrvADC.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

```
/* 设定ADC负向输入端为AIO1*/
DrvADC_NInputChannel(ADC_Input_AIO1);
```

6.3.3 DrvADC_SetADCInputChannel

- **函数**

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

- **函数功能**

设置ADC输入信号的正向、负向输入埠，设置寄存器0x41104[7:4] / 0x41104[3:0].

- **输入参数**

uINP [in]：代表ADC的正向输入选择埠，设定值范围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]：代表ADC负端输入选择埠，设定范围值0~9.

0：AIO0,　　1：AIO1,

2：AIO2,　　3：AIO3,

4：REFO_I,　　5：OPOI,

6：TPSN0,　7：TPSN1,

8：DAOI,　　9：VSS。

在C函数库中正向与负向输入使用同一组代表符：

{ADC_Input_AIO0, ADC_Input_AIO1, ADC_Input_AIO2,ADC_Input_AIO3,

REFO_I,OPOI,TPS0,TPS1,DAOI,VDDA_VSS}.

- **包含头文件**

Peripheral_lib/DrvADC.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/* 设定ADC正向输入端为AIO0，负向输入端为AIO1*/

DrvADC_SetADCInputChannel(ADC_Input_AIO0, ADC_Input_AIO1);

6.3.4 DrvADC_InputSwitch

- **函数**

unsigned int DrvADC_InputSwitch (uVISHR)

- **函数功能**

ADC信号输入端短路开关控制.

设置寄存器0x41100[21]

- **输入参数**

uVISHR[in] : ADC信号输入端短路开关控制. 设定值范围 ：0~1

0: 短路开关断开

1: 短路开关闭合

- **包含头文件**

Peripheral_lib/DrvADC.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/* ADC 输入端短路开关闭合*/

DrvADC_InputSwitch(1);

6.3.5 DrvADC_RefInputShort

- **函数**

unsigned int DrvADC_RefInputShort (E_ADC_SIGNAL_SHORT uVrshr);

- **函数功能**

ADC参考电压输入端短路开关控制.

设置寄存器0x41100[20].

- **输入参数**

uVrshr [in] : ADC参考电压输入端短路开关控制. 设定值范围 ：0~1

0 : ADC 参考电压输入端短路开关断开

1 : ADC 参考电压输入端短路开关闭合

- **包含头文件**

Peripheral_lib/DrvADC.h

- **函数返回值**

0：设置失败

其他：设置失败

- **函数用法**

/* 设置ADC 参考电压输入端短路开关闭合 */

DrvADC_RefInputShort(SHORT);

6.3.6 DrvADC_SetPGA

- **函数**

unsigned int DrvADC_SetPGA (E_ADC_PGA uPGA);

- **函数功能**

配置ADC 输入信号内部放大倍数控制器PGA;

设置寄存器0x41104[18:16].

- **输入参数**

uPGA [in]： 代表ADC内部放大倍数控制器PGA. 设定值范围 ：0~7

0: 放大倍数为 1

1: 放大倍数为 8

2: 不使用

3: 放大倍数为 16

4: 不使用

5: 不使用

6: 不使用

7: 放大倍数为 32

- **包含头文件**

Peripheral_lib/DrvADC.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/* 设置PGA=8 */

DrvADC_SetPGA(ADC_Gain_8);

6.3.7 DrvADC_ADGain

- **函数**

unsigned int DrvADC_ADGain (uADgain);

- **函数功能**

配置ADC 输入信号内部放大倍数控制器ADGIN；

设置寄存器0x41104[21:20].

- **输入参数**

  uADgain[in]：代表ADC 内部放大倍数控制器ADGN. 有效设定值为 ：0, 1, 3

  0: 放大倍数为 1

  1: 放大倍数为 2

  3: 放大倍数为 4

- **包含头文件**

  Peripheral_lib/DrvADC.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /*设置ADGN=2 */

  DrvADC_ADGain(1);

6.3.8 DrvADC_Gain

- **函数**

  unsigned int DrvADC_Gain (E_ADC_PGA uPGA ,uADgain);

- **函数功能**

  配置ADC 输入信号内部放大倍数控制器PGA及ADGN

  设置寄存器0x41104[18:16]/ 0x41104[21:20].

- **输入参数**

  uPGA [in]：代表ADC 放大倍数控制器PGA. 设定值范围 ：0~7

  0: 放大倍数为 1

  1: 放大倍数为 8

  2: 不使用

  3: 放大倍数为 16

  4: 不使用

  5: 不使用

  6: 不使用

  7: 放大倍数为 32

  uADgain [in]：代表ADC 放大倍数器 ADGN. 有效设定值为 ：0, 1, 3

  0: 放大倍数为 1

  1: 放大倍数为 2

  3: 放大倍数为 4

- **包含头文件**

  Peripheral_lib/DrvADC.h

- **函数返回值**

  0：设置成功

其他：设置失败

- **函数用法**

/* 设置ADC放大倍数PGA*ADGN=32*4=128 */

DrvADC_Gain(7,3);

6.3.9 DrvADC_DCoffset

- **函数**

unsigned int DrvADC_DCoffset (uDCoffset);

- **函数功能**

设置ADC 输入信号的零点平移(DC offset);设置寄存器0x41104[27:24]。

- **输入参数**

uDCoffice [in]：代表ADC 零点平移DCSET，VREF=REFP-REFN。设定值范围 ：0~15

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

- **包含头文件**

Peripheral_lib/DrvADC.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/* 设置零点平移(DCSET)为+1/8 VREF. */

DrvADC_DCoffset(1);

6.3.10 DrvADC_RefVoltage

- **函数**

```
unsigned int DrvADC_RefVoltage (
E_ADC_VRPS_REF_VOLTAGE uVrps,
E_ADC_VRNS_REF_VOLTAGE uVrns
 );
```

- **函数功能**

设置ADC 参考电压输入端口,参考电压(VREF)=VRPS-VRNS；

设置寄存器0x41100[19:18] 及 0x41100[17:16].

- **输入参数**

uVrps [in]：代表ADC 参考电压正向输入端VRPS. 设定值范围 ：0~3

0： 参考电压正向输入来自 VDDA

1： 参考电压正向输入来自 AIO2

2： 参考电压正向输入来自 AIO4

3： 参考电压正向输入来自 REFO_I

uVrns [in]：代表ADC 参考电压的负向输入端VRNS. 设定值范围 ：0~3

0： 参考电压负向输入来自 VSSA

1： 参考电压负向输入来自 AIO3

2： 参考电压负向输入来自 AIO5

3： 参考电压负向输入来自 REFO_I

- **包含头文件**

Peripheral_lib/DrvADC.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

```
/* 设置ADC参考输入电压(VRPS=AIO2, VRNS=AIO3) */
DrvADC_RefVoltage(AIO2, AIO3);
```

6.3.11 DrvADC_FullRefRange

- **函数**

```
unsigned int DrvADC_FullRefRange(uFullRange);
```

- **函数功能**

设置ADC 输入参考电压(VREF)的放大倍数;设置寄存器0x41104[19]。

- **输入参数**

uFullRange[in]：设置 ADC 输入参考电压(VREF)的放大数，VREF=VRPS-VRNS. 设定值范围 ：0~1

0: 1　　输入参考电压VREF*1

1: 1/2　输入参考电压VREF*1/2

- **包含头文件**

  Peripheral_lib/DrvADC.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /*设置ADC输入参考电压VREF*1 */

  DrvADC_FullRefRange(0);

6.3.12 DrvADC_OSR

- **函数**

  unsigned int DrvADC_OSR (uADCOSR);

- **函数功能**

  配置ADC 转换值的输出频率(OSR)，设置寄存器0x41100[5:2]。

- **输入参数**

  uADCOSR[in]： 表示ADC 转换值输出率(OSR)除频器设置(以下输出率是以时钟源为327680HZ计算). 设定值范围 ：0~10

  0 ： ÷32768，数据输出率是10sps

  1 ： ÷16384，数据输出率是20sps

  2 ： ÷8192，数据输出率是40sps

  3 ： ÷4096，数据输出率是80sps

  4 ： ÷2048，数据输出率是160sps

  5 ： ÷1024，数据输出率是320sps

  6 ： ÷512，数据输出率是640sps

  7 ： ÷256，数据输出率是1280sps

  8 ： ÷128 ，数据输出率是2560sps

  9 ： ÷64，数据输出率是5120sps

  10 ：÷32，数据输出率是10240sps

- **包含头文件**

  Peripheral_lib/DrvADC.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /* 设置输出率(OSR)为8192/40sps */

  DrvADC_OSR(2);

6.3.13 DrvADC_ClkEnable

- **函数**

  unsigned int DrvADC_ClkEnable(uADCD, uClkPH);

- **函数功能**

  使能ADC 时钟源，并设置时钟源分频值和ADC 时钟源相位调整；

  设置寄存器0x4030C[7:4].

- **输入参数**

  uADCD[in]： 表示ADC 时钟源分频器. 设定值范围 ：0~3

  0 ： ÷6

  1 ： ÷12

  2 ： ÷30

  3 ： ÷60

  uClkPH[in] : ADC 时钟源相位调整设置. 设定值范围 ：0~1

  0 ： ADC clock 上升沿在CPU Clock的低电平.

  1 ： ADC clock 上升沿在CPU Clock的高电平

- **包含头文件**

  Peripheral_lib/DrvADC.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /*设置ADC 频率分频÷12，且ADC时钟的上升沿为CPU时钟的高电平 */

  DrvADC_ClkEnable(1,1);

6.3.14 DrvADC_ClkDisable

- **函数**

  void DrvADC_ClkDisable(void);

- **函数功能**

  关闭ADC 时钟源；设置寄存器0x4030C[6]=0.

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvADC.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /* 关闭ADC时钟源 */

DrvADC_ClkDisable();

6.3.15 DrvADC_FastChopper

- **函数**

  unsigned int DrvADC_FastChopper(uADFDR);

- **函数功能**

  快速chopper 模式控制.,设置寄存器0x41100[6]. 设定值范围 ：0~1

- **输入参数**

  uADFDR[in]： 快速chopper模式控制.

  0： 正常chopper模式，频率等于 ADCLK/128

  1： 快速chopper模式，频率等于 ADCLK/32

- **包含头文件**

  Peripheral_lib/DrvADC.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /* 设置 ADC正常chopper模式. */

  DrvADC_FastChopper(0); //设置正常chopper模式，频率等于ADCLK/128

6.3.16 DrvADC_CombFilter

- **函数**

  unsigned int DrvADC_CombFilter(uCFRST);

- **函数功能**

  梳状滤波器使能控制，设置该位可以自动丢弃前3笔无效ADC资料；设置寄存器0x41100[1]。

- **输入参数**

  uCFRST[in]： 梳状滤波器使能控制. 设定值范围 ：0~1

  0: 复位(RESET)

  1: 开启(ON)

- **包含头文件**

  Peripheral_lib/DrvADC.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /* 使能梳状滤波器并配置自动丢弃前3笔无效数据. */

  DrvADC_CombFilter(0); //滤波器复位

DrvADC_CombFilter(1); //使能滤波器

6.3.17 DrvADC_EnableInt

- **函数**

  void DrvADC_EnableInt (void)

- **函数功能**

  开启ADC中断向量，ADC为中断向量HW2；设置寄存器0x40008[16]=1.

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvADC.h

- **函数返回值**

  无

- **函数用法**

  /* 使能ADC 中断 */

  DrvADC_EnableInt();

6.3.18 DrvADC_DisableInt

- **函数**

  void DrvADC_DisableInt (void)

- **函数功能**

  关闭ADC 中断向量；设置寄存器0x40008[16]=0.

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvADC.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭ADC中断向量 */

  DrvADC_DisableInt();

6.3.19 DrvADC_ReadIntFlag

- **函数**

  unsigned int DrvADC_ReadIntFlag (void)

- **函数功能**

读取ADC中断标志位(ADCIF).读取寄存器0x40008[0]值

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvADC.h

- **函数返回值**

  0：中断标志位值是0，表示无中断产生

  1：中断标志位值是1，表示有中断产生

  >1: 无效返回值

- **函数用法**

```
/* 读取ADC 中断标志位*/
flag=DrvADC_ReadIntFlag();  //读取ADC中断请求标志位
```

6.3.20 DrvADC_ClearIntFlag

- **函数**

  void DrvADC_ClearIntFlag (void)

- **函数功能**

  清除ADC中断标志位(ADCIF).清零寄存器0x40008[0]=0.

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvADC.h

- **函数返回值**

  无

- **函数用法**

```
/* 清除ADC中断标志位*/
DrvADC_ClearIntFlag();    //清除ADC中断标志位
```

6.3.21 DrvADC_Enable

- **函数**

  void DrvADC_Enable(void)

- **函数功能**

  开启ADC功能；设置寄存器0x41100[0]=1.

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvADC.h

- **函数返回值**

  无

- **函数用法**

  /* 使能ADC */

  DrvADC_Enable();

6.3.22 DrvADC_Disable

- **函数**

  void DrvADC_Disable(void)

- **函数功能**

  关闭ADC功能；设置寄存器0x41100[0]=0

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvADC.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭ADC功能 */

  DrvADC_Disable();

6.3.23 DrvADC_GetConversionData

- **函数**

  int DrvADC_GetConversionData (void);

- **函数功能**

  读取A/D 转换值，数据是带符号的.读取寄存器0x41108[31 :0].

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvADC.h

- **函数返回值**

  返回A/D转换值。.

- **函数用法**

  /*读取ADC 转换值*/

  int adc_data ;

  adc_data=DrvADC_GetConversionDate();

7 SPI32 串行通讯

7.1 函数简介

该部分函数描述 SPI 功能的控制，包括：

--SPI 功能的开启控制

--SPI 的工作模式及参数的配置

--SPI 的中断向量的控制

--SPI 状态控制

--SPI 的数据的收发

序号	函数名称	功能描述
01	DrvSPI32_Open	开启SPI 功能
02	DrvSPI32_Close	关闭SPI 功能
03	DrvSPI32_IsBusy	查询SPI 总线繁忙状态
04	DrvSPI32_SetClockFreq	配置SPI 时钟频率
05	DrvSPI32_IsRxBufferFull	查询接收缓存器状态位
06	DrvSPI32_IsTxBufferFull	查询发送缓存器状态位
07	DrvSPI32_EnableRxInt	开启SPI 接收中断功能
08	DrvSPI32_EnableTxInt	开启SPI 发送中断功能
09	DrvSPI32_DisableRxInt	关闭SPI 接收中断功能
10	DrvSPI32_DisableTxInt	关闭SPI 发送中断功能
11	DrvSPI32_GetRxIntFlag	读取SPI 接收中断标志位
12	DrvSPI32_GetTxIntFlag	读取SPI 发送中断标志位
13	DrvSPI32_ClrIntRxFlag	清除SPI 接收中断标志位
14	DrvSPI32_ClrIntTxFlag	清除SPI 发送中断标志位
15	DrvSPI32_Read	读取SPI 接收缓存器的数据
16	DrvSPI32_Write	写入数据至发送缓存器
17	DrvSPI32_Enable	开启SPI 功能
18	DrvSPI32_BitLength	设置数据的长度
19	DrvSPI32_GetDCFlag	读取SPI 数据丢失状态位
20	DrvSPI32_IsABFlag	读取SPI 接收到的数据长度小的状态
21	DrvSPI32_IsOVFlag	检查SPI 接收到资料是否过长
22	DrvSPI32_IsRxFlag	检查接收缓存器数据是否更新
23	DrvSPI32_SetEndian	设定数据发送是从MSB或LSB开始发送

24	DrvSPI32_SetCSO	SPI 时序源极性选择位设置
25	DrvSPI32_DisableIO	关闭IO口复用为SPI通讯口的功能
26	DrvSPI32_EnableIO	开启及选择IO口复用为SPI通讯口功能

7.2 内部定义常量

E_DRVSPI_MODE

标识符	数值	函数功能
E_DRVSPI_MASTER1	0	4-wire 主动模式
E_DRVSPI_MASTER2	1	3-wire 主动模式
E_DRVSPI_MASTER3	2	TI 方式主动模式
E_DRVSPI_SLAVE1	3	4-wire 被动模式
E_DRVSPI_SLAVE2	4	3-wire 被动模式
E_DRVSPI_SLAVE3	5	TI 方式被动模式

E_DRVSPI_TRANS_TYPE

标识符	数值	函数功能
E_DRVSPI_TYPE0	0	SPI 发送模式0
E_DRVSPI_TYPE1	1	SPI 发送模式1
E_DRVSPI_TYPE2	2	SPI 发送模式2
E_DRVSPI_TYPE3	3	SPI 发送模式3

E_DRVSPI_ENDIAN

标识符	数值	函数功能
E_DRVSPI_LSB_FIRST	1	从低8bit(LSB)开始发送
E_DRVSPI_MSB_FIRST	0	从高8bit(MSB)开始发送

E_DRVSPI_CS

标识符	数值	函数功能
E_DRVSPI_CSLow	0	CSO low
E_DRVSPI_CSHigh	1	CSO high

7.3 函数说明

7.3.1 DrvSPI32_Open

- 函数

```
unsigned int DrvSPI32_Open(
E_DRVSPI_MODE uMode,
E_DRVSPI_TRANS_TYPE uType,
uOuputPin,
uClkDiv
  );
```

- 函数功能

函数开启SPI功能，设置SPI工作是主动模式或者被动模式，设置SPI总线时序及通讯IO;

设置寄存器

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

- 输入参数

uMode [in] ：工作模式设置，设置范围0~5

0： 4-wire通讯接口的主动模式.

1： 3-wire通讯接口的主动模式.

2： TI 模式接口的主动模式.

3： 4-wire通讯接口的被动模式.

4： 3-wire通讯接口的被动模式.

5： TI 模式接口的被动模式.

uType [in]：传输类型，如通讯总线时序，设置范围是0~3.

0: 抓取数据在第一个时钟沿，时钟源低电平为空闲状态.(CPHA=0 CPOL=0)

1: 抓取数据在第一个时钟沿，时钟源高电平为空闲状态.(CPHA=0 CPOL=1)

2: 抓取数据在第二个时钟沿，时钟源低电平为空闲状态.(CPHA=1 CPOL=0)

3: 抓取数据在第二个时钟沿，时钟源高电平为空闲状态.(CPHA=1 CPOL=1)

uOuputPin[in]：SPI通讯IO 口设置，设置范围是 ：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

uClkDiv[in]：SPI时钟源分频器设置，设置范围是 ：0~7

0 : ÷1

1 : ÷2

2 : ÷4

3 : ÷8

4 : ÷32

5 : ÷128

6 : ÷512

7 : ÷2048

- **包含头文件**

Peripheral_lib/DrvSPI32.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/*使能SPI主动模式，时钟源分频CLOCK/512，设置传送类型1，通讯IO 口设置: Port1.4 =CS, Port1.5 =CK, Port1.6 = DI, Port1.7 =DO*/

DrvSPI32_Open(E_DRVSPI_MASTER1, E_DRVSPI_TYPE1, 1,6);

7.3.2 DrvSPI32_Close

- **函数**

void DrvSPI32_Close (void);

- **函数功能**

关闭SPI功能，关闭SPI的时钟、IO等功能；

设置寄存器0x40F00[0]=0, 0x4030C[3]=0,0x40844[4]=0 .

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvSPI32.h

- **函数返回值**

无

- **函数用法**

/* 关闭SPI */

DrvSPI32_Close();

7.3.3 DrvSPI32_IsBusy

- **函数**

unsigned int DrvSPI32_IsBusy(void);

- **函数功能**

  查询SPI总线上是否繁忙状态.

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  1：SPI总线繁忙

  0：SPI总线空闲.

- **函数用法**

  /* 检查总线繁忙状态*/

  unsigned char flag;

  flag=DrvSPI32_IsBusy (); //read 0x40f00[19]

7.3.4 DrvSPI32_SetClockFreq

- **函数**

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

- **函数功能**

  配置MCU时钟分频器及SPI时钟分频器且使能SPI时钟源，主动模式下输出时钟频率可程序设计设置;

  设置寄存器0x40308[1], 0x4030C[2:0].

- **输入参数**

  eCPUDV[in] : MCU 时钟分频器设置. 设定值范围 ：0~1

  0 : ÷1

  1 : ÷2

  eTMRDV[in] : SPI 时钟分频器设置. 设定值范围 ：0~7

  0 : ÷1

  1 : ÷2

  2 : ÷4

  3 : ÷8

  4 : ÷32

  5 : ÷128

  6 : ÷512

  7 : ÷2048

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  无

- **函数用法**

```
/* SPI  频率是APCK/512 */
DrvSPI32_SetClockFreq(1, 6);
```

7.3.5 DrvSPI32_IsRxBufferFull

- **函数**

  unsigned int DrvSPI32_IsRxBufferFull(void);

- **函数功能**

  查询接收缓存器满状态位(RXBF)（只用于数据接收）；设置寄存器0x40F00[16]。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  1: 接收已完成，接收缓存器已满.

  0: 接收未完成，接收缓存器为空.

- **函数用法**

```
/*  查询接收缓存器状态*/
unsigned char    flag;
flag = DrvSPI32_IsRxBufferFull() ;
```

7.3.6 DrvSPI32_IsTxBufferFull

- **函数**

  unsigned int DrvSPI32_IsTxBufferFull(void);

- **函数功能**

  查询发送缓存器满的状态(TXBF)（只用于资料发送）设置寄存器0x40F00[17]。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  1: 发送未完成，发送缓存器还有数据.

  0: 发送已完成，发送缓存器为空.

- **函数用法**

```
/* 查询发送缓存器的状态 */
unsigned char   flag;   flag =DrvSPI32_IsTxBufferFull();
```

7.3.7 DrvSPI32_EnableRxInt

- **函数**

  void DrvSPI32_EnableRxInt(void);

- **函数功能**

  使能SPI接收中断，属于中断向量HW0；设置寄存器0x40000[16]=1.

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  无

- **函数用法**

```
/* SPI 接收中断使能*/
DrvSPI32_EnableRxInt();
```

7.3.8 DrvSPI32_EnableTxInt

- **函数**

  void DrvSPI32_EnableTxInt(void);

- **函数功能**

  使能SPI 发送中断，属于中断向量HW0；设置寄存器0x40000[17]=1。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  无

- **函数用法**

  /*SPI 发送中断使能*/

  DrvSPI32_EnableTxInt();

7.3.9 DrvSPI32_DisableRxInt

- **函数**

  void DrvSPI32_DisableRxInt(void);

- **函数功能**

  关闭SPI 接收中断功能，设置寄存器0x40000[16]=0 ..

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  无

- **函数用法**

  /*关闭SPI 接收中断 */

  DrvSPI32_DisableRxInt();

7.3.10 DrvSPI32_DisableTxInt

- **函数**

  void DrvSPI32_DisableTxInt(void);

- **函数功能**

  关闭SPI 发送中断，设置寄存器0x40000[17]=0 .

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭SPI 发送中断 */

  DrvSPI32_DisableTxInt();

7.3.11 DrvSPI32_GetRxIntFlag

- **函数**

  unsigned int DrvSPI32_GetRxIntFlag ();

- **函数功能**

  读取SPI 接收中断请求标志位(SRXIF) ；读取寄存器0x40000[0]。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  1: 中断标志位为1，有中断请求

  0: 中断标志位为0，无中断请求

- **函数用法**

  /*读取SPI接收中断请求标志位*/

  unsigned char flag；flag=DrvSPI32_GetRxIntFlag();

7.3.12 DrvSPI32_GetTxIntFlag

- **函数**

  unsigned int DrvSPI32_GetTxIntFlag ();

- **函数功能**

  读取SPI 发送中断请求标志位(STXIF)；读取寄存器0x40000[1] 。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  1: 中断标志位为1，有中断请求

  0: 中断标志位为0，无中断请求

- **函数用法**

  /* 读取SPI 发送中断请求标志位.*/

  unsigned char flag；flag=DrvSPI32_GetTxIntFlag();

7.3.13 DrvSPI32_ClrIntRxFlag

- **函数**

  void DrvSPI32_ClrIntRxFlag ();

- **函数功能**

  清除SPI 接收中断请求标志位(SRXIF)；设置寄存器0x40000[0]=0.

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  无

- **函数用法**

  /*清除SPI 接收中断请求标志位*/

  DrvSPI32_ClrIntRxFlag();

7.3.14 DrvSPI32_ClrIntTxFlag

- **函数**

  void DrvSPI32_ClrIntTxFlag ();

- **函数功能**

  清除SPI 发送中断请求标志位（STXIF）,设置寄存器0x40000[1]=0 .

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  无

- **函数用法**

  /* 清除SPI发送中断请求标志位*/

  DrvSPI32_ClrIntTxFlag();

7.3.15 DrvSPI32_Read

- **函数**

  unsigned int DrvSPI32_Read();

- **函数功能**

  读取SPI 数据接收缓存器；读取寄存器0x40F08[31:0] 。 .

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  返回值是SPI 接收缓存器的值

- **函数用法**

  /* 读取接收缓存器的值 */

  /*数据接收方式LSB First 8bit 数据*/

  unsigned int data；data=DrvSPI32_Read()>>24;

  /*数据接收方式MSB 8bit 数据*/

  unsigned int data；data=DrvSPI32_Read();

7.3.16 DrvSPI32_Write

- **函数**

  void DrvSPI32_Write (unsigned int uData);

- **函数功能**

  写入待发送数据至发送缓存器并发送；写入寄存器0x40FC[31:0].

- **输入参数**

  uData [in]： 待发送资料：0~0xFFFFFFFF。

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  无

- **函数用法**

```
/*数据传送方式MSB First 8bit  发送0x55*/
DrvSPI32_Write(0x55<<24);
/*数据传送方式LSB First 8bit  发送0x55*/
DrvSPI32_Write(0x55);
```

7.3.17 DrvSPI32_Enable

- **函数**

  void DrvSPI32_Enable (void);

- **函数功能**

  使能SPI 功能；设置寄存器0x40F00[0]=1 .

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  无

- **函数用法**

```
/*  开启SPI */
DrvSPI32_Enable();
```

7.3.18 DrvSPI32_BitLength

- **函数**

  void DrvSPI32_BitLength (unsigned int uData);

- **函数功能**

  设置SPI 发送数据的长度；设置寄存器0x40F04[4:0] .

- **输入参数**

uData[in]：设置SPI 发送数据的长度，设定值范围是 ：0x04~0x20

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  无

- **函数用法**

  /* 设定SPI发送数据的长度为8bit*/

  DrvSPI32_BitLength(8);

7.3.19 DrvSPI32_GetDCFlag

- **函数**

  unsigned int DrvSPI32_GetDCFlag(void);

- **函数功能**

  读取SPI 数据丢失状态位(DCF) ，读取寄存器0x40F00[18]。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  0: 正常.

  1: 接收缓存器已满，读取接收缓存器可以清零该位.

- **函数用法**

  /* 读取数据丢失状态位 DCF */

  unsigned char flag；flag=DrvSPI32_GetDCFlag();

7.3.20 DrvSPI32_IsABFlag

- **函数**

  unsigned int DrvSPI32_IsABFlag(void);

- **函数功能**

  读取SPI 接收到的数据长度是否缺少的状态位(ABF)；读取寄存器0x40F00[20]的值 .

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  0: 正常.

  1: SPI接收到的数据长度比设置的数据长度少

- **函数用法**

  /* 读取数据长度标志位ABF */

  unsigned char flag；flag=DrvSPI32_IsABFlag();

7.3.21 DrvSPI32_IsOVFlag

- **函数**

  unsigned int DrvSPI32_IsOVFlag(void);

- **函数功能**

  读取接收到的数据长度是否比设定值长的状态位(VOF)，读取寄存器0x40F00[21]的值 .

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  0: 正常

  1: SPI接收到的数据长度比设定的数据长度大

- **函数用法**

  /*读取接收到数据长度过长标志位OVF*/

  unsigned char flag；flag=DrvSPI32_IsOVFlag();

7.3.22 DrvSPI32_IsRxFlag

- **函数**

  unsigned int DrvSPI32_IsRxFlag(void);

- **函数功能**

  读取SPI 数据接收缓存器的数据更新标志位(RXF)，确定是否读取接收缓存器；读取寄存器0x40F00[22] ..

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  0: 正常.

  1: SPI 接收缓存器有数据在更新，不能读取接收缓存器。

- **函数用法**

  /*读取SPI 接收缓存器数据更新标志位RxF */

  unsigned char flag；flag=DrvSPI32_IsRxFlag();

7.3.23 DrvSPI32_SetEndian

- **函数**

  void DrvSPI32_SetEndian(E_DRVSPI_ENDIAN eEndian);

- **函数功能**

  设置SPI 是从高8位还是低8位数据开始发送；设置寄存器0x40F04[18] .

- **输入参数**

  eEndian [in]： 输入范围 ：0~1

  1：低8位(LSB) 开始发送

  0：高8位(MSB) 开始发送

- **包含头文件**

  Peripheral_lib/DrvSPI32.h

- **函数返回值**

  无

- **函数用法**

  /*设置SPI 从低 8位数据开始发送 */

  DrvSPI32_ SetEndian(E_DRVSPI_LSB_FIRST);

7.3.24 DrvSPI32_SetCSO

- 函数

  void DrvSPI32_SetCSO(E_DRVSPI_CS eCS);

- 函数功能

  SPI 时序源极性选择位设置，设置寄存器0x40F04[20] .

  注意：该函数是将旧的函数DrvSPI32_SetCS(E_DRVSPI_CS eCS);的名称修改，但是功能新旧函数是一致的；新函数名称明确指出函数是操作CSO位。

  旧函数DrvSPI32_SetCS(E_DRVSPI_CS eCS);依然运行有效。

- 输入参数

  eCS [in]： 输入范围 ：0~1

  0：时序源低电平有效（CSO low）

  1：时序源高电平有效（CSO high）

- 包含头文件

  Peripheral_lib/DrvSPI32.h

- 函数返回值

  无

- 函数用法

```
/* 设置低电平有效 */
DrvSPI32_SetCSO(E_DRVSPI_CSLow);
```

7.3.25 DrvSPI32_DisableIO

- 函数

  void DrvSPI32_DisableIO(void);

- 函数功能

  关闭 SPI 通讯口，设置寄存器0x40844[4]=0; .

- 输入参数

  无

- 包含头文件

  Peripheral_lib/DrvSPI32.h

- 函数返回值

  无

- 函数用法

```
/*关闭SPI 通讯口 */
DrvSPI32_DisableIO();
```

7.3.26 DrvSPI32_EnableIO

- 函数

  unsigned char DrvSPI32_EnableIO(uint32_t uOuputPin);

- 函数功能

  开启SPI 通讯口，设置寄存器0x40844[7:5] / 0x40844[4]=1; .

- 输入参数

  uOuputPin[in]：SPI通讯IO 口设置. 输入范围：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

- 包含头文件

  Peripheral_lib/DrvSPI32.h

- 函数返回值

  0：设置成功

  1：设置失败

- 函数用法

```
/*开启SPI 通讯口并选择PT1.0~PT1.3*/
DrvSPI32_EnableIO(0);
```

8 异步串行通讯 UART

8.1 函数简介

该部分函数描述对 UART 功能的控制，包含：

--UART 功能的启动与关闭

--UART 功能的配置包括发送速率、时钟源、数据格式等

--UART 数据的发送与接收

--UART 中断向量控制

--UART 收发错误控制

序号	函数名称	功能描述
01	DrvUART_Open	开启UART 功能并配置相关参数
02	DrvUART_Close	关闭UART 功能
03	DrvUART_EnableInt	使能UART 中断向量
04	DrvUART_GetTxFlag	读取TX中断标志位
05	DrvUART_GetRxFlag	读取RX中断标志位
06	DrvUART_ClrTxFlag	清除TX中断标志位
07	DrvUART_ClrRxFlag	清除RX中断标志位
08	DrvUART_Read	接收到8位数据
09	DrvUART_Read9Bit	接收到9位数据
10	DrvUART_Write	写入资料并发送
11	DrvUART_EnableWakeUp	开启唤醒功能
12	DrvUART_GetPERR	读取UART校验错误状态位
13	DrvUART_GetFERR	读取UART帧错误状态为
14	DrvUART_GetOERR	读取UART溢出错误状态位
15	DrvUART_GetABDOVF	读取UART自动波特率翻转状态检测位
16	DrvUART_Enable_AutoBaudrate	开启UART自动波特率功能
17	DrvUART_Disable_AutoBaudrate	关闭UART自动波特率功能
18	DrvUART_CheckTRMT	读取UART发送寄存器状态位
19	DrvUART_ClkEnable	开启UART时钟源并设置时钟源
20	DrvUART_ClkDisable	关闭UART时钟源
21	DrvUART_Enable	开启UART
22	DrvUART_ConfigIO	设置IO复用为UART通讯口并选择IO口

8.2 内部定义常量

E_DATABITS_SETTINGS

标识符	数值	函数功能
DRVUART_DATABITS_8	0x0	数据长度为8 bits
DRVUART_DATABITS_9	0x1	数据长度为9 bits.

E_PARITY_SETTINGS

标识符	数值	函数功能
DRVUART_PARITY_NONE	0x0	无同位校验
DRVUART_PARITY_ODD	0x1	使能奇同位校验
DRVUART_PARITY_EVEN	0x2	使能偶同位校验

E_BAUD_RATE_SETTINGS

标识符	数值	函数功能
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

标识符	数值	函数功能
E_UART_ERR_CLOCK	0x2	时钟源输入错误
E_UART_ERR_BAUDRATE	0x3	波特率输入错误
E_UART_ERR_PARITY	0x4	校验方式输入错误
E_UART_ERR_DATABIT	0x5	数据长度输入错误
E_UART_ERR_OUTPIN	0x6	输出 IO 设置输入错误

8.3 函数说明

8.3.1 DrvUART_Open

- 函数

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

- 函数功能

设置UART的工作频率源 (除了晶震源为外部晶震(HSXT)或内部晶震(HSRC), UART除频设置也会影响到实际UART的工作频率源)并根据写入的波特率值自动计算出波特率寄存器0x40E0C[4:0] / 0x40E10[7:0] 的值；

设置UART的数据校验模式、数据的位数及TX/RX的通讯用IO口。

设置寄存器0x40E00[7]=1, 0x40E00[6]=1, 0x40E00[5] / 0x40E00[3]；

寄存器0x40E0C[3:0]/0x40E10[7:0]；设置IO口寄存器0x40844[3:0].

- 输入参数

uClock[in]： 设置UART工作频率源, 输入值为URCK 的频率大小, URCK是由高速晶振频率(外部高速HSXT或者内部高速频率HSRC) 经过UACD[3:0]分频得到, 若UACD=1，则URCK=HSXT(或HSRC), 若UACD=2，则URCK=HSXT/2(或HSRC/2) 依此类推, 以kHZ作为单位计算; 输入范围 ：1000~20000

uBaudRate[in] : UART通讯数据波特率

uParity [in]： 校验模式，分别为无校验/奇校验/偶校验，设定值范围：

0： 无校验

1： 偶校验

2： 奇校验

uDataBits[in]： 数据位数设置，设定范围是：

0 : 8 bit 数据.

1 : 9 bit 数据.

uOuputPin[in]：通讯线TX/RX IO口设置

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

- **包含头文件**

Peripheral_lib/DrvUART.h

- **函数返回值**

0：设置成功.

2：时钟设置错误

3：波特率设置错误

4：校验位设置错误

5：数据长度设置错误

6：通讯IO设置错误

- **函数用法**

/* 设置UART baud rate 115200bps, 8 位数据 ，且无校验，通讯口为PT1.4/PT1.5*/

DrvUART_Open(4000,115200, DRVUART_PARITY_NONE ,DRVUART_DATABITS_8,2);

Note： 如果 UART 工作频率源为 4MHz，所以输入频率为 4000，单位为 kHz.

8.3.2 DrvUART_Close

- **函数**

void DrvUART_Close (void);

- **函数功能**

关闭UART 功能；清零寄存器0x40E00[7]=0.;

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvUART.h

- **函数返回值**

无

- **函数用法**

/* 关闭UART */

DrvUART_Close();

8.3.3 DrvUART_EnableInt

- **函数**

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

- **函数功能**

UART的发送(TX)或接收(RX)中断向量控制. UART属于中断向量HW0;设置寄存器0x40000[19:18]。

- **输入参数**

uTXIE [in] : UART 发送(TX) 中断控制

0：关闭中断

1：使能中断

uRXIE [in] : UART 接收(RX) 中断控制

0：关闭中断

1：使能中断

- **包含头文件**

Peripheral_lib/DrvUART.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

```
/* 使能发送及接收中断 */
DrvUART_EnableInt(1,1);
```

8.3.4 DrvUART_GetTxFlag

- **函数**

```
unsigned int DrvUART_GetTxFlag (void);
```

- **函数功能**

读取发送中断标志位(UTXIF)值，读取寄存器0x40000[3]的值。

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvUART.h

- **函数返回值**

1: 有中断产生

0: 无中断产生

- **函数用法**

```
/* 读取发送中断标志位. */
DrvUART_GetTxFlag();
```

8.3.5 DrvUART_GetRxFlag

- **函数**

```
unsigned int DrvUART_GetRxFlag (void);
```

- **函数功能**

读取接收中断标志位URXIF值，读取寄存器0x40000[2]的值。

- **输入参数**

无

- **包含头文件**

  Peripheral_lib/DrvUART.h

- **函数返回值**

  1：有中断请求

  0：无中断请求

- **函数用法**

  /* 读取接收中断标志位. */

  unsigned char flag；flag=DrvUART_GetRxFlag();

8.3.6 DrvUART_ClrTxFlag

- **函数**

  void DrvUART_ClrTxFlag (void);

- **函数功能**

  清除发送中断标志位UTXIF值，清零寄存器0x40000[3]

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvUART.h

- **函数返回值**

  无

- **函数用法**

  /* 清除发送中断标志位. */

  DrvUART_ClrTxFlag();

8.3.7 DrvUART_ClrRxFlag

- **函数**

  void DrvUART_ClrRxFlag (void);

- **函数功能**

  清除接收中断标志位URXIF值，清零寄存器0x40000[2]

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvUART.h

- **函数返回值**

  无

- **函数用法**

  /* 清除接收中断标志位. */

  DrvUART_ClrRxFlag();

8.3.8 DrvUART_Read

- **函数**

  unsigned int DrvUART_Read(void);

- **函数功能**

  UART接收8位数据，读取寄存器0x40E18[7:0]的值

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvUART.h

- **函数返回值**

  返回接收缓存器的值.

- **函数用法**

  /* UART接收8位数据. */

  unsined int rx_data；rx_data=DrvUART_Read();

8.3.9 DrvUART_Read9Bit

- **函数**

  unsigned int DrvUART_Read9Bit(void);

- **函数功能**

  接收9位数据

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvUART.h

- **函数返回值**

  接收到的9Bits资料

- **函数用法**

  /* 接收9位数据*/

  unsigned int data; data=DrvUART_Read9Bit();

8.3.10 DrvUART_Write

- **函数**

  void DrvUART_Write(unsigned int uData);

- **函数功能**

  写入数值至发送缓存器(TXREG)并等待发送，写入待发送的值至寄存器0x40E14[7:0]。

- **输入参数**

  uData [in]

  待发送的8bit资料

- **包含头文件**

  Peripheral_lib/DrvUART.h

- **函数返回值**

  无

- **函数用法**

  /*发送0x55 */

  DrvUART_Write(0x55);

8.3.11 DrvUART_EnableWakeUp

- **函数**

  void DrvUART_EnableWakeUp(void);

- **函数功能**

  使能UART的唤醒功能，同样启动接收唤醒功能只要接收中断打开；

  设置寄存器0x40E00[0]=1。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvUART.h

- **函数返回值**

  无

- **函数用法**

  /* 使能UART唤醒功能 */

  DrvUART_EnableWakeUp();

8.3.12 DrvUART_GetPERR

- **函数**

  unsigned int DrvUART_GetPERR(void);

- **函数功能**

  读取校验错误标志位(PERR)，读取寄存器0x40E04[5]的值。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvUART.h

- **函数返回值**

1：有校验错误

0：无校验错误

- **函数用法**

/* 读取校验错误标志位. */

unsigned char flag; flag=DrvUART_GetPERR();

8.3.13 DrvUART_GetFERR

- **函数**

unsigned int DrvUART_GetFERR(void);

- **函数功能**

读取帧错误标志位(FERR)，读取寄存器0x40E04[4]的值。

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvUART.h

- **函数返回值**

1：有帧错误

0：无帧错误

- **函数用法**

/* 读取帧错误标志位. */

unsigned char flag ; flag=DrvUART_GetFERR();

8.3.14 DrvUART_GetOERR

- **函数**

unsigned int DrvUART_GetOERR(void);

- **函数功能**

读取溢出错误标志位(OERR)，读取寄存器0x40E04[3]的值。

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvUART.h

- **函数返回值**

1：有溢出错误

0：无溢出错误

- **函数用法**

/* 读取溢出错误标志位. */

unsigned char flag ; flag=DrvUART_GetOERR();

8.3.15 DrvUART_GetABDOVF

- **函数**

  unsigned int DrvUART_GetABDOVF(void);

- **函数功能**

  读取自动波特率发生器翻转状态检测标志位(ABDOVF)，读取寄存器0x40E04[0]值。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvUART.h

- **函数返回值**

  1： 在自动波特率检测模式下波特率发生器发生翻转

  0： 没有波特率发生器发生翻转

- **函数用法**

  /* 读取波特率发生器翻转标志位ABDOVF. */

  unsigned char flag ; flag=DrvUART_GetABDOVF();

8.3.16 DrvUART_Enable_AutoBaudrate

- **函数**

  void DrvUART_Enable_AutoBaudrate ();

- **函数功能**

  使能UART 自动波特率功能，设置寄存器0x40E08[0]=1.

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvUART.h

- **函数返回值**

  无

- **函数用法**

  /* 使能UART自动波特率功能 */

  DrvUART_Enable_AutoBaudrate();

8.3.17 DrvUART_Disable_AutoBaudrate

- **函数**

  void DrvUART_Disable_AutoBaudrate ();

- **函数功能**

  关闭UART 自动波特率功能，设置寄存器0x40E08[0]=0。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvUART.h

- **函数返回值**

  无

- **函数用法**

```
/* 关闭UART自动波特率功能 */
DrvUART_Disable_AutoBaudrate();
```

8.3.18 DrvUART_CheckTRMT

- **函数**

  Unsigned int DrvUART_CheckTRMT(void)

- **函数功能**

  读取UART发送状态位(TRMT)，读取寄存器0x40E04[1]值

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvUART.h

- **函数返回值**

  返回发送状态位TRMT的值;

- **函数用法**

```
/* 读取发送状态位值并实现查询方式发送数据*/
DrvUART_Write(data) ;
While(DrvUART_CheckTRMT()) ;//等待TRMT=0
While( !DrvUART_CheckTRMT() ) ;//等待TRMT=1
```

8.3.19 DrvUART_ClkEnable

- 函数

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

- 函数功能

使能UART的时钟源并选择时钟源及设置时钟源的分频值

设置寄存器0x40308[21 :16] 。

- 输入参数

uclk[in] : EUART 时钟源设置

0： 外部晶震高速时钟

1： 内部晶震高速时钟

uprescale[in]：时钟源分频器

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

- 包含头文件

Peripheral_lib/DrvUART.h

- 函数返回值

0 ：设置成功

1 ：设置失败

- 函数用法

```
/* 设置UART时钟源为外部时钟且分频clk/1 */
DrvUART_ClkEnable(0,0);
```

8.3.20 DrvUART_ClkDisable

- 函数

Void DrvUART_ClkDisable(void) ;

- 函数功能

关闭UART时钟源,设置寄存器0x40308[20]=0。

- 输入参数

无

- 包含头文件

Peripheral_lib/DrvUART.h

- 函数返回值

无

- 函数用法

/*关闭UART 时钟源*/

DrvUART_ClkDisable();

8.3.21 DrvUART_Enable

- 函数

Void DrvUART_Enable(void) ;

- 函数功能

使能UART功能 ,设置寄存器0x40E00[7:6]=11b

- 输入参数

无

- 包含头文件

Peripheral_lib/DrvUART.h

- 函数返回值

无

- 函数用法

/*使能UART功能*/

DrvUART_Enable();

8.3.22 DrvUART_ConfigIO

- 函数

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

- 函数功能

设置IO口复用为UART通讯口，及选择IO口 ,设置寄存器0x40844[3 :0]。

- 输入参数

ioen[in] :IO 口复用功能使能控制

0：关闭IO 复用功能

1：开启IO 复用功能

uoutputPin[in] :选择通讯IO 口

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

- **包含头文件**

  Peripheral_lib/DrvUART.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /*开启IO复用为UART通讯口，并选择PT2.0/PT2.1*/

  DrvUART_ConfigIO(1,4);

9 多功能比较器 CMP

9.1 函数简介

该部分函数描述CMP 功能的控制，包含：

--CMP 模块功能的启动与关闭

--CMP 配置控制

--CMP 中断向量控制

--CMP 的 NON-OVER LAP 功能控制

序号	函数名称	功能描述
01	DrvCMP_Open	开启比较器CMP
02	DrvCMP_Close	关闭比较器CMP
03	DrvCMP_Enable	开启比较器CMP
04	DrvCMP_PInput	设置CMP的正端输入源
05	DrvCMP_NInput	设置CMP的负端输入源
06	DrvCMP_InputSwitch	CMP的输入短路开关控制
07	DrvCMP_OutputFilter	CMP数字滤波控制
08	DrvCMP_OutputPinEnable	开启CMP数字输出功能
09	DrvCMP_OutputPinDisable	关闭CMP数字输出功能
10	DrvCMP_OutputInverse	设置CMP数字输出反相控制
11	DrvCMP_EnableInt	开启CMP中断
12	DrvCMP_DisableInt	关闭CMP中断
13	DrvCMP_ReadIntFlag	读取CMP中断标志位
14	DrvCMP_ClearIntFlag	清除CMP中断标志位
15	DrvCMP_Input	比较器CMP信号输入正负端设置
16	DrvCMP_RLO_Ctrl	CMP内部电阻分压(RLO)网络设置
17	DrvCMP_RLO_refv	CMP内部电阻分压参考电压设置
18	DrvCMP_EnableNonOverlap	开启CMP的non-overlap功能
19	DrvCMP_DisableNonOverlap	关闭CMP的non-overlap功能
20	DrvCMP_ReadData	读取CMP数字输出状态

9.2 内部定义常量

E_NON_OVERLAP_PIN

标识符	数值	函数功能
E_CL1	0x0	设置PT1.2 作为正端输入源
E_CL2	0x1	设置PT1.3 作为正端输入源
E_CL3	0x2	设置PT3.1 作为正端输入源
E_CL4	0x3	设置PT3.2 作为正端输入源

9.3 函数说明

9.3.1 DrvCMP_Open

- **函数**

  unsigned int DrvCMP_Open (uPowerMode , uCPDF, uCPOR)

- **函数功能**

  使能比较器功能，设置低功耗模式，设置比较器2us 延时滤波功能控制及比较器数字输出相位控制。

  设置寄存器0x41804[7:6] / 0x41804[1:0]

- **输入参数**

  uPowerMode[in]：比较器功耗控制：.

  0：低功耗模式

  1：正常模式

  uCPDF [in]：比较器输出滤波控制

  0：不经过delitch滤波

  1：经过2us延时滤波

  uCPOR [in]：比较器数字输出相位控制

  0：正常输出

  1：反相输出

- **包含头文件**

  Peripheral_lib/DrvCMP.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /*使能比较器，设置低功耗，2us滤波，正常输出 */

  DrvCMP_Open(0,1,0);

9.3.2 DrvCMP_Close

- **函数**

  void DrvCMP_Close (void)

- **函数功能**

  关闭比较器功能，设置寄存器0x41804[0]=0.

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvCMP.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭比较器功能*/

  DrvCMP_Close();

9.3.3 DrvCMP_Enable

- **函数**

  void DrvCMP_Enable (void)

- **函数功能**

  使能比较器功能,设置寄存器0x41804[0]=1.

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvCMP.h

- **函数返回值**

  无

- **函数用法**

  /* 使能比较器功能 */

  DrvCMP_Enable();

9.3.4 DrvCMP_PInput

- **函数**

  unsigned int DrvCMP_PInput (uCPPS)

- **函数功能**

  设置比较器正向输入端. 设置寄存器0x41800[7:6].

- **输入参数**

  uCPPS[in]：比较器正向输入端选择：

  0 : CH1

  1 : CH2

  2 : CH3

  3 : V12(V12=1.2V)

- **包含头文件**

  Peripheral_lib/DrvCMP.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /* 设置比较器正向输入端为CH1. */

  DrvCMP_PInput(0);

9.3.5 DrvCMP_NInput

- **函数**

  unsigned int DrvCMP_NInput (uCPNS)

- **函数功能**

  设置比较器负向输入端，设置寄存器0x41800[5:4]

- **输入参数**

  uOPNS[in]：比较器负向输入端选择：

  0 : CH1

  1 : CH2

  2 : CH3

  3 : RLO

- **包含头文件**

  Peripheral_lib/DrvCMP.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

```
/* 比较器负向输入端为CH1. */
DrvCMP_NInput(0);
```

9.3.6 DrvCMP_InputSwitch

- **函数**

  unsigned int DrvCMP_InputSwitch (uInputSwitch)

- **函数功能**

  比较器输入端短路开关控制，设置寄存器0x41804[5]

- **输入参数**

  uInputSwitch[in]：输入端短路开关控制.

  0：短路开关断开

  1：短路开关闭合

- **包含头文件**

  Peripheral_lib/DrvCMP.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

```
/* 比较器输入端短路开关闭合*/
```

DrvCMP_InputSwitch(1);

9.3.7 DrvCMP_OutputFilter

- **函数**

  unsigned int DrvCMP_OutputFilter(uFilter)

- **函数功能**

  比较器数字输出滤波控制，设置寄存器0x41804[1]

- **输入参数**

  uFilter[in] : 2us延时滤波设置

  0： 不经过滤波

  1： 经过2us滤波

- **包含头文件**

  Peripheral_lib/DrvCMP.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /* 比较器输出经过2us延时滤波. */

  DrvCMP_OutputFilter(1);

9.3.8 DrvCMP_OutputPinEnable

- **函数**

  unsigned int DrvCMP_OutputPinEnable (E_OUTPUT_PIN uPin)

- **函数功能**

  使能比较器数字输出IO 口，设置寄存器0x40840[16]=1。

- **输入参数**

  uPin [in] : 0 : PT1.7

- **包含头文件**

  Peripheral_lib/DrvCMP.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /* 使能CMP 数字输出IO 口*/

  DrvCMP_OutputPinEnable(0);

9.3.9 DrvCMP_OutputPinDisable

- **函数**

  void DrvCMP_OutputPinDisable (void)

- **函数功能**

  关闭比较器数字输出IO 口，设置寄存器0x40840[16]=0。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvCMP.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭比较器数字输出IO 口*/

  DrvCMP_OutputPinDisable();

9.3.10 DrvCMP_OutputInverse

- **函数**

  unsigned int DrvCMP_OutputInverse(uInv)

- **函数功能**

  比较器数字输出相位控制，设置寄存器0x41804[7] .

- **输入参数**

  uInv [in] :

  0： 正常输出

  1： 反相输出

- **包含头文件**

  Peripheral_lib/DrvCMP.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /* 比较器输出反相*/

  DrvCMP_OutputInverse(1);

9.3.11 DrvCMP_EnableInt

- **函数**

  void DrvCMP_EnableInt (void)

- **函数功能**

  使能比较器中断，比较器中断属于中断向量HW3，设置寄存器0x40008[17]=1。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvCMP.h

- **函数返回值**

  无

- **函数用法**

```
/* 使能比较器中断*/
DrvCMP_EnableInt();
```

9.3.12 DrvCMP_DisableInt

- **函数**

  void DrvCMP_DisableInt (void)

- **函数功能**

  关闭比较器中断功能，设置寄存器0x40008[17]=0 .

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvCMP.h

- **函数返回值**

  无

- **函数用法**

```
/* 关闭比较器中断功能*/
DrvCMP_DisableInt();
```

9.3.13 DrvCMP_ReadIntFlag

- **函数**

  unsigned int DrvCMP_ReadIntFlag (void)

- **函数功能**

  读取比较器中断请求标志位CPOIF，读取寄存器0x40008[1]的值。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvCMP.h

- **函数返回值**

  0： 中断标志位为0，标明无CMP中断请求

  1： 中断标志位为1，标明有CMP中断请求

  >1: 无效函数返回值

- **函数用法**

  /* 读取比较器中断标志位 */

  unsigned char flag；flag=DrvCMP_ReadIntFlag();

9.3.14 DrvCMP_ClearIntFlag

- **函数**

  void DrvCMP_ClearIntFlag (void)

- **函数功能**

  清除比较器中断标志位CPOIF ，清零寄存器0x40008[1] =0。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvCMP.h

- **函数返回值**

  无

- **函数用法**

  /* 清除比较器中断标志位*/

  DrvCMP_ClearIntFlag();

9.3.15 DrvCMP_Input

- 函数

  unsigned int DrvCMP_Input (uPositiveInput, uNegativeInput,uInputSwitch)

- 函数功能

  设置比较器正向、负向输入端并设置输入端短路开关，

  设置寄存器0x41800[7:6] / 0x41800[5:4] / 0x41804[5] .

- 输入参数

  uPositiveInput[in] : CMP 正向输入端选择：.

  0 : CH1

  1 : CH2

  2 : CH3

  3 : V12

  uNegativeInput[in] : CMP 负向输入端选择：

  0 : CH1

  1 : CH2

  2 : CH3

  3 : RLO

  uInputSwitch[in]： 输入短路开关控制：.

  0： 短路开关断开

  1： 短路开关闭合

- 包含头文件

  Peripheral_lib/DrvCMP.h

- 函数返回值

  0：设置成功

  其他：设置失败

- 函数用法

```
/* 设置CPPS=CH1,CPNS=CH3, 及输入端短路开关断开 */
DrvCMP_Input(0,2,0);
```

9.3.16 DrvCMP_RLO_Ctrl

- **函数**

  unsigned int DrvCMP_RLO_Ctrl (uCPDA ,uCPDM)

- **函数功能**

  使能比较器内部自带电阻分压功能RLO，并设置分压值，设置寄存器0x41804[23:20] / 0x41804[19:16]值。

- **输入参数**

  uCPDA[in]：比较器内部分压阶梯电阻比例控制.

  0 : 0　　8 : 8/16 (CPRLH – CPRLL)

  1 : 1/16 (CPRLH – CPRLL)　　9 : 9/16 (CPRLH – CPRLL)

  2 : 2/16 (CPRLH – CPRLL)　　10 :10/16 (CPRLH – CPRLL)

  3 : 3/16 (CPRLH – CPRLL)　　11 :11/16 (CPRLH – CPRLL)

  4 : 4/16 (CPRLH – CPRLL)　　12 :12/16 (CPRLH – CPRLL)

  5 : 5/16 (CPRLH – CPRLL)　　13 :13/16 (CPRLH – CPRLL)

  6 : 6/16 (CPRLH – CPRLL)　　14 :14/16 (CPRLH – CPRLL)

  7 : 7/16 (CPRLH – CPRLL)　　15 : 15/16 (CPRLH – CPRLL)

  uCPDM [3:0]　比较器输出迟滞控制器，位控制操作模式；在迟滞模式下，uCPDA [3:0]对应位的值与 CMPO 的值一样;

  uCPDM [3]　0：关闭

  1：开启

  uCPDM [2]　0：关闭

  1：开启

  uCPDM [1]　0：关闭

  1：开启

  uCPDM [0]　0：关闭

  1：开启

- **包含头文件**

  Peripheral_lib/DrvCMP.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

```
/* 设置CPDM=0101,CPDA=0011 */
DrvCMP_RLO_Ctrl(0x03,0x05);
```

9.3.17 DrvCMP_RLO_refv

- **函数**

  unsigned int DrvCMP_RLO_refV (uCPRH, uCPRLS)

- 函数功能

  比较器内部自带电阻分压功能的输入参考电压源的设置及电阻低阶短路开关控制；

  设置寄存器0x41800[2:1] / 0x41800[3]

- **输入参数**

  uCPRH[in]：电阻分压功能的参考电压源输入选择：

  0：关闭（高阻态）

  1 : ChargePump 输入（CP_I）

  2 : VDD3V(系统电压)

  3 : VDD18 (1.8V数字电压)

  uCPRLS[in]：电阻低阶短路开关控制：.

  0：短路开关断开

  1：短路开关闭合

- **包含头文件**

  Peripheral_lib/DrvCMP.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /* 电阻参考电压源为VDD18,短路开关闭合 */

  DrvCMP_RLO_refV(3,1);

9.3.18 DrvCMP_EnableNonOverlap

- **函数**

  unsigned int DrvCMP_EnableNonOverlap (E_NON_OVERLAP_PIN uInput)

- **函数功能**

  使能比较器的non-overlap自动切换功能，并设置参考输入通道CLx.

  设置寄存器0x41804[4:2]

- **输入参数**

  uInput[in]：比较器自动切换功能正向参考输入端选择：

  0 : CL1

  1 : CL2

  2 : CL3

  3 : CL4

- **包含头文件**

  Peripheral_lib/DrvCMP.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/* 使能比较器non-overlap自动切换功能并设置CL1 作为正向参考输入端*/

DrvCMP_EnableNonOverlap(0)

9.3.19 DrvCMP_DisableNonOverlap

- **函数**

void DrvCMP_DisableNonOverlap (void)

- **函数功能**

关闭比较器non-overlap 自动切换功能，设置寄存器0x41804[4] =0 。

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvCMP.h

- **函数返回值**

无

- **函数用法**

/* 关闭比较器non-overlap自动切换功能 */

DrvCMP_DisableNonOverlap();

9.3.20 DrvCMP_ReadData

- **函数**

unsigned int DrvCMP_ReadData (void)

- **函数功能**

读取比较器数字输出状态值CMPO，读取寄存器0x41800[16]值

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvCMP.h

- **函数返回值**

0: 负端输入 > 正端输入

1: 正端输入 > 负端输入

- **函数用法**

/* 读取CMP 输出状态值*/

unsigned char flag；flag=DrvCMP_ReadData();

10 低噪声运算放大器 OPAMP

10.1 功能简介

该函数部分描述低噪声运算放大器 OPAMP 功能的操作

--OPAMP 功能的开关

--OPAMP 输入埠及输出埠的控制

--OPAMP 中断的设置及开关

--OPAMP 输出信号的反相输出及滤波控制

序号	函数名称	函数功能
01	DrvOP_Open	开启OPAMP功能
02	DrvOP_Close	关闭OPAMP功能
03	DrvOP_PInput	OPAMP正端输入设置
04	DrvOP_NInput	OPAMP负端输入设置
05	DrvOP_OPOoutEnable	开启OPAMP输出
06	DrvOP_OPOoutDisable	关闭OPAMP输出
07	DrvOP_OuputFilter	OPAMP数字滤波输出设置
08	DrvOP_OutputPinEnable	开启OPAMP 数字输出IO pin.
09	DrvOP_OutputPinDisable	关闭OPAMP 数字输出IO pin
10	DrvOP_OutputInverse	OPAMP数字输出反相设置
11	DrvOP_OutputWithCHPCK	CHPCK多功能器设置
12	DrvOP_EnableInt	开启OPAMP中断
13	DrvOP_DisableInt	关闭OPAMP中断
14	DrvOP_ReadIntFlag	读取OPAMP中断标志位
15	DrvOP_ClearIntFlag	清除OPAMP中断标志位
16	DrvOP_Feedback	OPAMP回馈电路设置
17	DrvOP_OPDEN	OPAMP数字输出功能控制

10.2 内部定义常量

E_OUTPUT_PIN

标识符	数值	函数功能
E_OPO1	0x0	PT3.0作为 OPAMP 数字输出IO口
E_OPO2	0x1	PT3.1作为 OPAMP 数字输出IO口

E_OPN_PPIN

标识符	数值	功能定义
E_OPP_AIO2	0x1	AIO2 作为OPAMP输入口
E_OPP_AIO4	0x2	AIO4 作为OPAMP输入口
E_OPP_DAOI	0x4	DAOI作为OPAMP输入口
E_OPP_REFO_I	0x8	REFO_I作为OPAMP输入口
E_OPN_AIO3	0x1	AIO3 作为OPAMP输入口
E_OPN_AIO5	0x2	AIO5 作为OPAMP输入口
E_OPN_DAOI	0x4	DAOI 作为OPAMP输入口
E_OPN_OPOI	0x8	OPOI 作为OPAMP输入口
E_OPN_OPO	0x10	OPO 作为OPAMP输入口
E_OPN_OPC	0x20	OPC 作为OPAMP输入口

10.3 函数说明

10.3.1 DrvOP_Open

- 函数

  void DrvOP_Open (void)

- 函数功能

  开启运算放大器(OPAMP)功能；设置寄存器0x41900[0]=1.

- 输入参数

  无

- 包含头文件

  Peripheral_lib/DrvOP.h

- 函数返回值

  无

- 函数用法

```
/*  使能运算放大器(OPAMP) */
DrvOP_Open();
```

10.3.2 DrvOP_Close

- 函数

  void DrvOP_Close (void)

- 函数功能

  关闭运算放大器(OPAMP)功能 ,设置寄存器0x41900[0]=0。

- 输入参数

  无

- 包含头文件

  Peripheral_lib/DrvOP.h

- 函数返回值

  无

- 函数用法

```
/*  关闭运算放大器(OPAMP) */
DrvOP_Close();
```

10.3.3 DrvOP_PInput

- 函数

  unsigned int DrvOP_PInput (uOPPS)

- 函数功能

运算放大器(OPAMP)的正向输入端设置，

设置寄存器0x41904[19:16]即OPPS[3:0]，每一位对应一路通道，且可以同时设置多路通道有效。

- **输入参数**

uOPPS [3:0] 运算放大器OPAMP 正向输入端选择，输入范围0x00~0x0f，为0时关闭所有通道。

uOPPS[3]：运算放大器(OPAMP)正向输入通道3

0：关闭通道，高阻抗状态

1：开启通道 REFO_I

uOPPS[2]：运算放大器(OPAMP)正向输入通道2

0：关闭通道，高阻抗状态

1：开启通道 DAOI

uOPPS[1]：运算放大器(OPAMP)正向输入通道1

0：关闭通道，高阻抗状态

1：开启通道 AIO4

uOPPS[0]：运算放大器(OPAMP)正向输入通道0

0：关闭通道，高阻抗状态

1：开启通道 AIO2

- **包含头文件**

Peripheral_lib/DrvOP.h

- **函数返回值**

0：设置成功; 其他：设置失败

- **函数用法**

```
/* 设置运算放大器(OPAMP)正向输入端AIO2与AIO4. */
DrvOP_PInput(0x1|0x2);
```

10.3.4 DrvOP_NInput

- **函数**

unsigned int DrvOP_NInput (uOPNS)

- **函数功能**

运算放大器(OPAMP)负向端输入埠选择;

设置寄存器0x41904[5:0]即OPNS[5:0]，且每一位对应一路输入通道，且可以同时设置多路通道有效。

- **输入参数**

uOPNS[5:0]：运算放大器(OPAMP)负向输入端选择，输入范围0x00~0x3f,为0时关闭所有通道。

uOPNS[5]：运算放大器(OPAMP)负向输入通道 5

0：关闭通道，高阻抗状态

1：开启通道 OPC: 内部连接10pF 电容

uOPNS[4]：运算放大器(OPAMP)负向输入通道 4

0：关闭通道，高阻抗状态

1：开启通道 OPO: 运算放大器OPAMP 内部输出

uOPNS[3]：运算放大器(OPAMP)负向输入通道 3

0：关闭通道，高阻抗状态

1：开启通道 OPOI: 运算放大器OPAMP 外部输出

uOPNS[2]：运算放大器(OPAMP)负向输入通道 2

0：关闭通道，高阻抗状态

1：开启通道 DAOI DAC的输出

uOPNS[1]：运算放大器(OPAMP)负向输入通道 1

0：关闭通道，高阻抗状态

1：开启通道 AI5

uOPNS[0]：运算放大器(OPAMP)负向输入通道 0

0：关闭通道，高阻抗状态

1：开启通道 AI3

- **包含头文件**

  Peripheral_lib/DrvOP.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

```
/*开启运算放大器(OPAMP)负向输入端通道AIO3与AIO5. */
DrvOP_NInput(0x1|0x2);
```

10.3.5 DrvOP_OPOoutEnable

- **函数**

```
void DrvOP_OPOoutEnable(void)
```

- **函数功能**

  打开运算放大器(OPAMP)仿真输出功能，寄存器0x41900[1]=1。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvOP.h

- **函数返回值**

  无

- **函数用法**

```
/* 打开运算放大器(OPAMP)模式输出*/
DrvOP_OPOoutEnable();
```

10.3.6 DrvOP_OPOoutDisable

- **函数**

void DrvOP_OPOoutDisable(void)

- **函数功能**

运算放大器(OPAMP) 仿真输出功能关闭，寄存器0x41900[1]=0.

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvOP.h

- **函数返回值**

无

- **函数用法**

/* 关闭运算放大器(OPAMP)模拟输出 */

DrvOP_OPOoutDisable();

10.3.7 DrvOP_OuputFilter

- **函数**

unsigned int DrvOP_OuputFilter(uFilter)

- **函数功能**

运算放大器(OPAMP)数字输出滤波控制，控制数字输出是否经过2s延时滤波;

设置寄存器0x41900[3]。

- **输入参数**

uFilter[in]

0： 无滤波

1： 经过2us delay 滤波

- **包含头文件**

Peripheral_lib/DrvOP.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/*设置运算放大器(OPAMP) 2us 延时滤波. */

DrvOP_OuputFilter(1);

10.3.8 DrvOP_OutputPinEnable

- **函数**

unsigned int DrvOP_OutputPinEnable (E_OUTPUT_PIN uPin)

- **函数功能**

使能运算放大器(OPAMP)数字输出端口，并选择OPAMP输出IO口；

寄存器0x41900[2]=1,且设置寄存器0x40840[19:18]。

- **输入参数**

uPin [in]

0 : PT3.0

1 : PT3.1

- **包含头文件**

Peripheral_lib/DrvOP.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/* 使能运算放大器(OPAMP)数字输出，IO= PT3.0*/

DrvOP_OutputPinEnable(0);

10.3.9 DrvOP_OutputPinDisable

- **函数**

void DrvOP_OutputPinDisable (void)

- **函数功能**

关闭运算放大器(OPAMP)数字输出功能及OPAMP输出IO口功能；

寄存器0x41900[2]=0, 寄存器0x40840[18]=0。.

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvOP.h

- **函数返回值**

无

- **函数用法**

/* 关闭运算放大器(OPAMP)数字输出*/

DrvOP_OutputPinDisable();

10.3.10 DrvOP_OutputInverse

- **函数**

unsigned int DrvOP_OutputInverse(uInv)

- **函数功能**

运算放大器(OPAMP)数字输出信号输出反相控制，设置寄存器0x41900[5].

- **输入参数**

  uInv [in]

  0： 正常输出

  1： 输出反相

- **包含头文件**

  Peripheral_lib/DrvOP.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

```
/* 使能运算放大器(OPAMP)数字输出反相*/
DrvOP_OutputInverse(1);
```

10.3.11 DrvOP_OutputWithCHPCK

- **函数**

  unsigned int DrvOP_OutputWithCHPCK (uCHPCK)

- **函数功能**

  运算放大器(OPAMP)数字输出信号OPO1/OPO2 是否经过CHPCK多功能器设置.

  设置寄存器0x41900[6],且在使能该功能前需要打开ADC clock，才能正确启动该功能。

  该函数的旧名称是：DrvOP_OutputWithCPCLK()，其功能一样。

- **输入参数**

  uCHPCK [in]

  0： 不带CHPCK多功能器, OPO1/OPO2 输出等于 OPOD

  1： 带CHPCK多功能器r, OPO1/OPO2 输出一个基于CHPCK的高频信号

- **包含头文件**

  Peripheral_lib/DrvOP.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

```
/* 设置运算放大器(OPAMP)数字输出带CHPCK */
DrvADC_ClkEnable(0,0);     //开启ADC 时钟源
DrvOP_OutputWithCHPCK(1);  //使能CHPCK 多功能器
```

10.3.12 DrvOP_EnableInt

- **函数**

  void DrvOP_EnableInt (void)

- **函数功能**

  使能运算放大器(OPAMP)中断向量，运算放大器(OPAMP)处于中断向量HW3；

  设置寄存器0x4000c[16]=1.

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvOP.h

- **函数返回值**

  无

- **函数用法**

  /* 使能运算放大器(OPAMP)中断 */

DrvOP_EnableInt();

10.3.13 DrvOP_DisableInt

- **函数**

  void DrvOP_DisableInt (void)

- **函数功能**

  关闭运算放大器(OPAMP)中断向量 ,清零寄存器0x4000c[16]=0;

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvOP.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭运算放大器(OPAMP)中断 */

  DrvOP_DisableInt();

10.3.14 DrvOP_ReadIntFlag

- **函数**

  unsigned int DrvOP_ReadIntFlag (void)

- **函数功能**

  读取运算放大器(OPAMP)中断标志位OPOIF ；读取寄存器0x4000c[0]的值。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvOP.h

- **函数返回值**

  0：运算放大器(OPAMP)中断标志位为0，无中断请求

  1：运算放大器(OPAMP)中断标志位为1，有中断请求

  >1: 无效返回值

- **函数用法**

  /* 读取运算放大器(OPAMP)中断标志位 */

  unsigned char flag；flag=DrvOP_ReadIntFlag();

10.3.15 DrvOP_ClearIntFlag

- **函数**

  void DrvOP_ClearIntFlag (void)

- **函数功能**

  清除运算放大器(OPAMP)中断请求标志位OPOIF . 清零寄存器0x4000c[0] =0.

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvOP.h

- **函数返回值**

  无

- **函数用法**

  /* 清除运算放大器(OPAMP)中断请求标志位 */

  DrvOP_ClearIntFlag();

10.3.16 DrvOP_Feedback

- **函数**

  unsigned int DrvOP_Feedback(uFeedback)

- **函数功能**

  运算放大器(OPAMP)回馈电路或采样电容连接设置，设置寄存器0x41900[4]。

- **输入参数**

  uFeedback [in]

  0：电容作为集成电容器，下端连接至 OPOI

  1：电容作为采样电容，下端连接至 VSSA

- **包含头文件**

  Peripheral_lib/DrvOP.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/*运算放大器(OPAMP)回馈电容连接到 OPOI */

DrvOP_Feedback(0);

10.3.17 DrvOP_OPDEN

- **函数**

unsigned char DrvOP_OPDEN(uOPDEN)

- **函数功能**

OPAMP数字输出功能控制，写入寄存器0x41900[2]。

- **输入参数**

uOPDEN [in]：OPAMP数字输出功能控制，输入范围: 0~1

0：关闭

1：开启

- **包含头文件**

Peripheral_lib/ DrvOP.h

- **函数返回值**

0：设置成功

1：设置失败

- **函数用法**

/* 开启OPAMP数字输出功能控制*/

DrvOP_OPDEN(1);

11 电源管理 PMU

11.1 函数简介

该部分函数描述电源管理系统的控制，包含：

--VDDA 电压的控制

--带隙(BANDGAP)参考电压的控制

--Charge Pump升压电路控制

--REFO 电压的控制

--Low Power模式及ADC analog ground的控制

序号	函数名称	功能描述
01	DrvPMU_VDDA_Voltage	VDDA电压值设置
02	DrvPMU_VDDA_LDO_Ctrl	VDDA LDO 使能控制
03	DrvPMU_BandgapEnable	带隙参考电压开启控制
04	DrvPMU_BandgapDisable	带隙参考电压关闭控制
05	DrvPMU_ChargePumpEnable	Charge Pump升压电路开启控制
06	DrvPMU_ChargePumpDisable	Charge Pump升压电路关闭控制
07	DrvPMU_REFO_Enablenable	模拟参考电压REFO开启控制
08	DrvPMU_REFO_Disableisable	模拟参考电压REFO关闭控制
09	DrvPMU_AnalogGround	ADC模拟共地端控制
10	DrvPMU_LDO_LowPower	VDD LDO 低功耗模式控制

11.2 内部定义常量

E_VDDA_OUTPUT_VOLTAGE

标识符	数值	函数功能
E_VDDA2_4	0x0	设置VDDA=2.4V
E_VDDA2_7	0x1	设置VDDA=2.7V
E_VDDA3_0	0x2	设置VDDA=3.0V
E_VDDA3_3	0x3	设置VDDA=3.3V

E_VDDA_LDO_ENABLE_CONTROL

标识符	数值	函数功能
E_HighZ	0x0	设置VDDA=0v
E_ VDD3V	0x1	设置VDDA=VDD3V
E_ PullDown	0x2	设置VDDA=0v
E_LDO	0x3	设置VDDA=2.4~3.3V可调

11.3 函数说明

11.3.1 DrvPMU_VDDA_Voltage

- **函数**

  unsigned int DrvPMU_VDDA_Voltage(E_VDDA_OUTPUT_VOLTAGE uVoltage)

- **函数功能**

  设置VDDA输出电压值，设置寄存器0x40400[19:18].

- **输入参数**

  uVoltage [in] :VDDA电压选择. 输入范围 ：0~3

  0 : 2.4V

  1 : 2.7V

  2 : 3.0V

  3 : 3.3V

- **包含头文件**

  Peripheral_lib/DrvPMU.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /* 设置VDDA =2.7V. */

  DrvPMU_VDDA_Voltage(E_VDDA2_7);

11.3.2 DrvPMU_VDDA_LDO_Ctrl

- **函数**

  unsigned int DrvPMU_VDDA_LDO_Ctrl(E_VDDA_LDO_ENABLE_CONTROL uCtrl)

- **函数功能**

  设置VDDA稳压电压输入源；该功能影响到VDDA输出电压，所以配合VDDA设置一起使用；

  设置寄存器0x40400[17:16] .

- **输入参数**

  uCtrl [in] : VDDA稳压电压输入源选择. 输入范围 ：0~3

  0： 高阻态（High Z）,VDDA=0

  1 : VDD3V，VDDA=VDD3V

  2： 弱下拉(Weak pull down)，VDDA=0

  3： 可调稳压(LDO),此模式VDDA才能可调。

- **包含头文件**

Peripheral_lib/DrvPMU.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

```
/* 设置VDDA LDO 使能.且设置VDDA=2.7V*/
DrvPMU_VDDA_LDO_Ctrl(E_LDO);
DrvPMU_VDDA_Voltage(E_VDDA2_7);
```

11.3.3 DrvPMU_BandgapEnable

- **函数**

void DrvPMU_BandgapEnable(void)

- **函数功能**

使能带隙(Bandgap)参考电压，设置寄存器0x40400[4] =1 .

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvPMU.h

- **函数返回值**

无

- **函数用法**

```
/* 使能带隙参考电压*/
DrvPMU_BandgapEnable();
```

11.3.4 DrvPMU_BandgapDisable

- **函数**

void DrvPMU_BandgapDisable(void)

- **函数功能**

关闭带隙(Bandgap)参考电压功能，设置寄存器0x40400[4]=0 .

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvPMU.h

- **函数返回值**

无

- **函数用法**

```
/* 关闭带隙参考电压. */
DrvPMU_BandgapDisable();
```

11.3.5 DrvPMU_ChargePumpEnable

- **函数**

  void DrvPMU_ChargePumpEnable(void)

- **函数功能**

  使能ChargePump升压电路，设置寄存器0x40400[2] =1 .

  注意：Charge Pump升压电路工作需要ADC clock 作为时钟源，因此使用时要打开ADC 时钟源。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvPMU.h

- **函数返回值**

  无

- **函数用法**

```
/* 开启charge pump升压功能*/
DrvADC_ClkEnable(0,0) ;                //开启ADC 时钟源
DrvPMU_ChargePumpEnable();     //开启ChargePump升压功能
```

11.3.6 DrvPMU_ChargePumpDisable

- **函数**

  void DrvPMU_ChargePumpDisable (void)

- **函数功能**

  关闭Charge pump升压电路，设置寄存器0x40400[2]= 0 .

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvPMU.h

- **函数返回值**

  无

- **函数用法**

```
/*关闭charge pump升压功能. */
DrvPMU_ChargePumpDisable();
```

11.3.7 DrvPMU_REFO_Enablenable

- 函数

  void DrvPMU_REFO_Enable(void)

- 函数功能

  模拟参考电压REFO使能控制，输出1.2v电压,但是需要先开启带隙参考电压；设置寄存器0x40400[1] =1 .

- 输入参数

  无

- 包含头文件

  Peripheral_lib/DrvPMU.h

- 函数返回值

  无

- 函数用法

```
/*  使能模拟参考电压REFO. */
DrvPMU_BandgapEnable() ;   //开启带隙参考电压
DrvPMU_REFO_Enable();      //开启仿真参考电压REFO
```

11.3.8 DrvPMU_REFO_Disableisable

- 函数

  void DrvPMU_REFO_Disable(void)

- 函数功能

  仿真参考电压REFO关闭控制，关闭1.2v电压输出；设置寄存器0x40400[1] =0 .

- 输入参数

  无

- 包含头文件

  Peripheral_lib/DrvPMU.h

- 函数返回值

  无

- 函数用法

```
/*关闭模拟参考电压REFO. */
DrvPMU_REFO_Disable();
```

11.3.9 DrvPMU_AnalogGround

- 函数

unsigned int DrvPMU_AnalogGround(uAG)

- **函数功能**

ADC模拟地输入源选择，设置寄存器0x40400[3] 。

- **输入参数**

uAG [in]

0： 外部

1： 使能buffer及使用内部（配合ADC一起使用）

- **包含头文件**

Peripheral_lib/DrvPMU.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

```
/* 设置ADC 模拟地输入来源外部 */
DrvPMU_AnalogGround(0);
```

11.3.10 DrvPMU_LDO_LowPower

- **函数**

unsigned int DrvPMU_LDO_LowPower(uLP)

- **函数功能**

VDD LDO 低功耗控制，设置寄存器0x40400[0]

- **输入参数**

uLP [in]

0： 正常功耗（从sleep模式唤醒后需要设置0）

1： 低功耗

- **包含头文件**

Peripheral_lib/DrvPMU.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

```
/* 使能 LDO 低功耗模式 */
DrvPMU_LDO_LowPower(1);
```

12 数模转换器 DAC

12.1 函数功能简介

该部分函数介绍 DAC 功能的设置

--DAC 功能的启动与关闭

--DAC 输入埠的设置

--DAC 的输出口设置

--DAC 输出电压的设置

序号	函数名称	功能描述
01	DrvDAC_Open	开启DAC并配置相关参数
02	DrvDAC_Close	关闭DAC及DAC IO口输出功能
03	DrvDAC_Enable	开启DAC
04	DrvDAC_Disable	关闭DAC
05	DrvDAC_EnableOutput	开启DAC输出
06	DrvDAC_DisableOutput	关闭DAC输出
07	DrvDAC_PInput	DAC正端参考输入设置
08	DrvDAC_NInput	DAC负端参考输入设置
09	DrvDAC_DABIT	D/A转换输出电压比例设置
10	DrvDAC_SetoutputIO	DAC输出IO 口设置

12.2 内部定义常量

E_DAC_INPUT

标识符	数值	函数功能
E_DAC_PVDD3V	0x0	正端信号输入端
E_ DAC_PVDDA	0x1	正端信号输入端
E_ DAC_PREFO_I	0x2	正端信号输入端
E_DAC_POPO	0x3	正端信号输入端
E_ DAC_PAIO6	0x4	正端信号输入端
E_ DAC_NVSSA	0x0	负端信号输入端
E_ DAC_NREFO_I	0x1	负端信号输入端
E_ DAC_NOPO	0x2	负端信号输入端
E_ DAC_NAIO7	0x3	负端信号输入端

12.3 函数说明

12.3.1 DrvDAC_Open

- 函数

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

- 函数功能

  使能DAC及设置DAC正向、负向参考电压输入端口，并且设置DAC输出分压的初始比例值；

  设置寄存器0x41700[5:0] 及寄存器0x41704[7:0].

  注意：AIO6的设置需要通过设置ADC 寄存器0x41104[28]BIT DA 来启动与关闭！

- 输入参数

  uPinput[in] : DAC 正向参考输入端选择：

  0 : VDD3V

  1 : VDDA

  2 : REFO_I

  3 : OPO

  4：AIO6 （需要通过设置ADC寄存器0x41104[28]BIT DA来选择）

  uNinput[in] : DAC 负向参考输入端选择：.

  0 : VSSA

  1 : REFO_I

  2 : OPO

  3 : AIO7

  uDAO [in] : DAO[7:0] 输出电压值的分压比例设置DAO/255. 输入范围 ：0~255

- 包含头文件

  Peripheral_lib/DrvDAC.h

- 函数返回值

  0：设置成功

  其他：设置失败

- 函数用法

  /* 使能DAC, 设定正向输入端为AIO6, 负向输入端为VSSA ,DAO=5 */

  DrvDAC_Open(E_DAC_AIO6, E_DAC_VSSA ,5);

12.3.2 DrvDAC_Close

- 函数

  void DrvDAC_Close(void)

- 函数功能

关闭DAC及关闭DAC电压输出，设置寄存器0x41700[1:0]=00b;

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvDAC.h

- **函数返回值**

  无

- **函数用法**

```
/* 关闭 DAC及输出功能*/
DrvDAC_Close();
```

12.3.3 DrvDAC_Enable

- **函数**

  void DrvDAC_Enable(void)

- **函数功能**

  开启DAC ，设置寄存器0x41700[0]=1.

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvDAC.h

- **函数返回值**

  无

- **函数用法**

```
/* 开启 DAC */
DrvDAC_Enable();
```

12.3.4 DrvDAC_Disable

- **函数**

  void DrvDAC_Disable(void)

- **函数功能**

  关闭DAC，设置寄存器0x41700[0]=0;

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvDAC.h

- **函数返回值**

  无

- **函数用法**

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

12.3.5 DrvDAC_EnableOutput

- **函数**

  void DrvDAC_EnableOutput(void)
- **函数功能**

  使能DAC 输出，设置寄存器0x41700[1]=1;
- **输入参数**

  无
- **包含头文件**

  Peripheral_lib/DrvDAC.h
- **函数返回值**

  无
- **函数用法**

```
/* 使能DAC输出 */
DrvDAC_EnableOutput();
```

12.3.6 DrvDAC_DisableOutput

- **函数**

  void DrvDAC_DisableOutput (void)
- **函数功能**

  关闭DAC 输出，设置寄存器0x41700[1] =0.
- **输入参数**

  无
- **包含头文件**

  Peripheral_lib/DrvDAC.h
- **函数返回值**

  无
- **函数用法**

```
/*关闭 DAC 输出 */
DrvDAC_DisableOutput();
```

12.3.7 DrvDAC_PInput

- 函数

  unsigned int DrvDAC_Pinput(E_DAC_INPUT uPinput)

- 函数功能

  DAC正向输入端选择设置，设置寄存器0x41700[5:4]，及ADC寄存器0x41104[28]

- 输入参数

  uPinput [in] : DAC 正向输入端选择. 输入范围 : 0~4

  0 : VDD3V

  1 : VDDA

  2 : REFO_I

  3 : OPO

  4 : AIO6

- 包含头文件

  Peripheral_lib/DrvDAC.h

- 函数返回值

  0：设置成功

  其他：设置失败

- 函数用法

```
/* 设置DAC正向输入端为VDD3V. */
DrvDAC_Pinput(E_DAC_VDD3V);
```

12.3.8 DrvDAC_NInput

- 函数

  unsigned int DrvDAC_Ninput(E_DAC_INPUT uNinput)

- 函数功能

  DAC 负向输入端选择设置，设置寄存器0x41700[3:2].

- 输入参数

  uNinput [in] : DAC 负向输入端选择. 输入范围 : 0~3

  0 : VSSA

  1 : REFO_I

  2 : OPO

  3 : AIO7

- 包含头文件

  Peripheral_lib/DrvDAC.h

- 函数返回值

  0：设置成功

  其他：设置失败

- **函数用法**

  /* 设定 DAC 负向输入端为VSSA. */

  DrvDAC_Ninput(E_DAC_VSSA);

12.3.9 DrvDAC_DABIT

- **函数**

  unsigned int DrvDAC_DABIT(uDABIT)

- **函数功能**

  DAO[7:0] 输出电压值的分压比例设置即DAO/255，值写入寄存器0x41704[7:0]。

- **输入参数**

  uDABIT [in]：写入DAO[7:0]，D/A转换输出电压比例值设置uDABIT/255. 输入范围: 0~255

- **包含头文件**

  Peripheral_lib/DrvDAC.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

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

  DrvDAC_DABIT(5);

12.3.10 DrvDAC_SetoutputIO

- **函数**

  unsigned int DrvDAC_SetoutputIO(unsigned int uio)

- **函数功能**

  设置DAC输出IO口,设置PT3寄存器0x40828[17]

- **输入参数**

  uio [in] :

  0： 关闭PT3.1作为DAC 输出口的功能

  1： 开启PT3.1作为DAC 输出口的功能

- **包含头文件**

  Peripheral_lib/DrvDAC.h

- **函数返回值**

  0：设置成功

  1：设置失败

- **函数用法**

  /* 使能PT3.1作为DAC输出 IO*/

DrvDAC_SetoutputIO(1);

13 实时时钟 RTC

13.1 函数简介

函数描述对 RTC 系统的控制包含：

--RTC 时钟源的设置

--RTC 的启动与关闭

--RTC 的时间格式设置及当前时间的写入与读取操作

--RTC 的闹钟功能的设置

序号	函数名称	功能描述
01	DrvRTC_SetFrequencyCompensation	设置 RTC 频率补偿值
02	DrvRTC_WriteEnable	写入寄存器解锁码，解锁寄存器写入操作
03	DrvRTC_WriteDisable	清除寄存器解锁码，寄存器无法写入
04	DrvRTC_ClockSource	设置RTC的时钟源为内部或外部
05	DrvRTC_AlarmEnable	开启RTC闹钟功能
06	DrvRTC_AlarmDisable	关闭RTC闹钟功能
07	DrvRTC_PeriodicTimeEnable	开启RTC定时唤醒功能及设置定时唤醒时间
08	DrvRTC_PeriodicTimeDisable	关闭RTC定时唤醒功能
09	DrvRTC_Enable	开启RTC功能
10	DrvRTC_Disable	关闭RTC功能
11	DrvRTC_HourFormat	设置时间格式为12小时制或24小时制
12	DrvRTC_ReadState	读取RTC的状态位
13	DrvRTC_ClearState	清除RTC的状态位
14	DrvRTC_EnableInt	开启RTC中断功能
15	DrvRTC_DisableInt	关闭RTC中断功能
16	DrvRTC_ReadIntFlag	读取RTC中断标志位
17	DrvRTC_ClearIntFlag	清除RTC中断标志位
18	DrvRTC_Write	设定’当前/闹钟’的时间和日期
19	DrvRTC_Read	读取’当前/闹钟’的时间和日期
20	DrvRTC_ClkConfig	RTC时钟源的设置控制
21	DrvRTC_EnableWUEn	开启RTC唤醒功能
22	DrvRTC_DisableWUEn	关闭RTC唤醒功能

13.2 内部定义常量

E_DRVRTC_CLOCK_SOURCE

标识符	数值	功能意义
E_EXT_CK	0	RTC时钟源由外部低频时钟提供
E_INT_CK	1	RTC时钟源有内部低频时钟提供

E_DRVRTC_ TICK

标识符	数值	功能意义
E_DRVRTC_1_128_SEC	0	定时唤醒除频 1/128
E_DRVRTC_1_64_SEC	1	定时唤醒除频 1/64
E_DRVRTC_1_32_SEC	2	定时唤醒除频 1/32
E_DRVRTC_1_16_SEC	3	定时唤醒除频 1/16
E_DRVRTC_1_8_SEC	4	定时唤醒除频 1/8
E_DRVRTC_1_4_SEC	5	定时唤醒除频 1/4
E_DRVRTC_1_2_SEC	6	定时唤醒除频 1/2
E_DRVRTC_1_SEC	7	定时唤醒除频 1

E_DRVRTC_HOUR_FORMAT

标识符	数值	功能意义
E_DRVRTC_HOUR_12	1	12小时制
E_DRVRTC_HOUR_24	0	24小时制

E_DRVRTC_TIME_SELECT

标识符	数值	功能意义
DRVRTC_CURRENT_TIME	0	选择’当前时间’选项
DRVRTC_ALARM_TIME	1	选择’闹钟时间’选项

E_DRVRTC_FLAG

标识符	数值	功能意义
E_DRVRTC_ALARM_FLAG	0	闹钟标志位
E_DRVRTC_PERIODIC_FLAG	1	定时时间标志位
E_DRVRTC_CLEAR_ALL	2	闹钟标志位和定时时间标志位

13.3 函数说明

注意：需要先使能 RTC clock，再写入解锁码(对寄存器 0X41A00[23:20]写 0110b)，然后才能正确写入寄存器

13.3.1 DrvRTC_SetFrequencyCompensation

- **函数**

  unsigned int DrvRTC_SetFrequencyCompensation(

  unsigned int uFrequencyCom);

- **函数功能**

  设置RTC时钟频率补偿值，设置寄存器0x41a04[22:16]。

- **输入参数**

  uFrequencyCom [in]： 设置RTC时钟频率补偿值，设定范围是 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

- **包含头文件**

  Peripheral_lib/DrvRTC.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /*设置频率补偿为 -2 PPM */

  DrvRTC_ SetFrequencyCompensation(0x41);

13.3.2 DrvRTC_WriteEnable

- **函数**

  void DrvRTC_WriteEnable(void);

- **函数功能**

  写入解锁码恢复RTC寄存器写入操作.，对寄存器0x41A00[23 :20]写入0110b

注意必须要写入解锁码才能对寄存器进行写入！

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvRTC.h

- **函数返回值**

  无

- **函数用法**

```
/* RTC寄存器解锁，解锁后才能写入操作 */
DrvRTC_WriteEnable();
```

13.3.3 DrvRTC_WriteDisable

- **函数**

```
void DrvRTC_WriteDisable(void);
```

- **函数功能**

  清除RTC解锁码，重新锁住寄存器不允许写入，对寄存器0x41A00[23:20]写入0000b

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvRTC.h

- **函数返回值**

  无

- **函数用法**

```
/* 锁住RTC寄存器，不允许写入操作*/
DrvRTC_WriteDisable();
```

13.3.4 DrvRTC_ClockSource

- **函数**

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

- **函数功能**

  设定RTC时钟源为内部或外部低速时钟. 设置寄存器0x41A00[1]。

- **输入参数**

  uClockSource [in]

0：外部低频时钟源

1：内部低频时钟源

- **包含头文件**

Peripheral_lib/DrvRTC.h

- **函数返回值**

CKS ：拥有防误操作保护

0：外部低频时钟

1：内部低频时钟

- **函数用法**

```
/* 设置RTC时钟源来自外部低频时钟 */
DrvRTC_ClockSource(E_EXT_CK);
```

13.3.5 DrvRTC_AlarmEnable

- **函数**

void DrvRTC_AlarmEnable (void);

- **函数功能**

使能闹钟(Alarm)功能；设置寄存器0x41A00[3]=1.

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvRTC.h

- **函数返回值**

无

- **函数用法**

```
/* 使能 RTC 闹钟(Alarm)功能*/
DrvRTC_AlarmEnable();
```

13.3.6 DrvRTC_AlarmDisable

- **函数**

void DrvRTC_AlarmDisable (void);

- **函数功能**

关闭闹钟(Alarm)功能；设置寄存器0x41A00[3]=0.

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvRTC.h

- **函数返回值**

无

- **函数用法**

/*关闭闹钟(Alarm)功能*/

DrvRTC_AlarmDisable();

13.3.7 DrvRTC_PeriodicTimeEnable

- **函数**

unsigned int DrvRTC_PeriodicTimeEnable (E_DRVRTC_TICK uPeriodicTimer);

- **函数功能**

使能定时唤醒(Periodic Time)功能并设置定时唤醒的时间；

设置寄存器0x41A04[2:0] 及 寄存器0x41A00[5]=1,0x41A00[4]=1.

- **输入参数**

uPeriodicTimer[in]：定时唤醒除频器设置

0: 1/128

1: 1/64

2: 1/32

3: 1/16

4: 1/8

5: 1/4

6: 1/2

7: 1

- **包含头文件**

Peripheral_lib/DrvRTC.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

/* 使能RTC定时唤醒功能，设置定时唤醒时间为1/16second*/

DrvRTC_PeriodicTimeEnable(3);

13.3.8 DrvRTC_PeriodicTimeDisable

- **函数**

void DrvRTC_PeriodicTimeDisable (void);

- **函数功能**

关闭定时唤醒(Periodic Time)功能，设置寄存器0x41A00[5]=0 / 0x41A00[4]=0。

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvRTC.h

- **函数返回值**

无

- **函数用法**

/* 关闭定时唤醒(Periodic time)功能*/

DrvRTC_PeriodicTimeDisable();

13.3.9 DrvRTC_Enable

- **函数**

void DrvRTC_Enable (void);

- **函数功能**

使能RTC 功能，设置寄存器0x41A00[0]=1。

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvRTC.h

- **函数返回值**

无

- **函数用法**

/* 使能 RTC 功能*/

DrvRTC_Enable();

13.3.10 DrvRTC_Disable

- **函数**

void DrvRTC_Disable (void);

- **函数功能**

关闭RTC功能，设置寄存器0x41A00[0]=0.

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvRTC.h

- **函数返回值**

无

- **函数用法**

```
/* 关闭RTC功能 */
DrvRTC_Disable();
```

13.3.11 DrvRTC_HourFormat

- **函数**

  unsigned int DrvRTC_HourFormat(E_DRVRTC_HOUR_FORMAT uHourFormat);

- **函数功能**

  设置小时格式为12小时制或24小时制，设置寄存器0x41A00[2]。

- **输入参数**

  uHourFormat[in]： 小时格式设置

  0 : 24小时制

  1 : 12小时制

- **包含头文件**

  Peripheral_lib/DrvRTC.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

```
/* 设置12小时制 */
DrvRTC_DrvRTC_HourFormat(1);
```

13.3.12 DrvRTC_ReadState

- **函数**

  unsigned int DrvRTC_ReadState(void);

- **函数功能**

  读取RTC状态标志位，读取寄存器0x41A00[19:16]值

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvRTC.h

- **函数返回值**

  返回值有效范围是0x0~0xf，每一位值对应一个标志位状态值

  Bit 0 : RTC 闹钟标志位TAF

  Bit 1 : RTC 唤醒功能标志位WUF

  Bit 2 : RTC 定时唤醒标志位PTF

Bit 3 : RTC 闰年标志位LPYF

- **函数用法**

/* 查询RTC闹钟状态位 */

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

- **函数**

unsigned int DrvRTC_ClearState(E_DRVRTC_FLAG uFlag);

- **函数功能**

清除RTC状态标志位，清零寄存器0x41A00[19:16]值

- **输入参数**

uFlag[in] 待清除状态位选择

0： 清除闹钟标志位TAF

1： 清除定时唤醒标志位PTF

2: 清除闹钟（TAF）和定时（PTF）标志位

- **包含头文件**

Peripheral_lib/DrvRTC.h

- **函数返回值**

0：设置成功

其他：设置失败

- **函数用法**

```
/* 清除TAF/ PTF标志位 */
DrvRTC_ClearState(2);
```

13.3.14 DrvRTC_EnableInt

- **函数**

void DrvRTC_EnableInt(void)

- **函数功能**

使能RTC中断。每次进入中断都需要清除定时唤醒标志位（PTF）下次才能正常进入中断；

设置寄存器0x40004[21]=1.

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvRTC.h

- **函数返回值**

  无

- **函数用法**

  /* 使能RTC中断 */

  DrvRTC_EnableInt();

13.3.15 DrvRTC_DisableInt

- **函数**

  void DrvRTC_DisableInt(void)

- **函数功能**

  关闭RTC 中断，设置寄存器0x40004[21]=0.

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvRTC.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭RTC中断 */

  DrvRTC_DisableInt();

13.3.16 DrvRTC_ReadIntFlag

- **函数**

  unsigned int DrvRTC_ReadIntFlag(void)

- **函数功能**

  读取RTC中断请求标志位RTCIF，读取寄存器0x40004[5]的值。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvRTC.h

- **函数返回值**

  0：无中断请求

  1：有中断请求

- **函数用法**

/* 读取RTC中断标志位*/

unsigned char flag；flag=DrvRTC_ReadIntFlag();

13.3.17 DrvRTC_ClearIntFlag

- **函数**

void DrvRTC_ClearIntFlag(void)

- **函数功能**

清除RTC中断请求标志位RTCIF； 寄存器0x40004[5]=0

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvRTC.h

- **函数返回值**

无

- **函数用法**

/* 清除RTC中断请求标志位 */

DrvRTC_ClearIntFlag();

13.3.18 DrvRTC_Write

- **函数**

unsigned int DrvRTC_Write (

E_DRVRTC_TIME_SELECT eTime, S_DRVRTC_TIME_DATA_T *sPt);

- **函数功能**

设置RTC的当前（或闹钟）的时间和日期;

设置寄存器0x41A08/0x41A0C/0x41A10/0x41A14/0x41A18/0x41A1C

- **输入参数**

eTime [in]： 指向‘当前时间/日期’或‘闹钟时间/日期’.

0: 当前时间/日期

1: 闹钟时间/日期

*sPt [in]： 表示要设置的时间/日期，该变量为一个结构体，设置内容为：

u8cClockDisplay	DRVRTC_CLOCK_12 / DRVRTC_CLOCK_24
u8cAmPm	DRVRTC_AM / DRVRTC_PM
u32cSecond	Second 数值
u32cMinute	Minute 数值
u32cHour	Hour 数值(12小时制的输入范围：0~11；24小时制输入范围：0~23)
u32cDayOfWeek	Day of week
u32cDay	Day 数值

u32cMonth Month 数值

u32Year Year 数值

- 包含头文件

Peripheral_lib/DrvRTC.h

- 函数返回值

0：设置成功

其他：设置失败.

- 函数用法

```
/* 更新当前时间‘秒数’为0 */
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

- 函数

```
unsigned int DrvRTC_Read (
E_DRVRTC_TIME_SELECT eTime,
S_DRVRTC_TIME_DATA_T *sPt );
```

- 函数功能

读取RTC的‘当前时间/日期’或‘闹钟的时间/日期’的数据

读取寄存器0x41A08/0x41A0C/0x41A10/0x41A14/0x41A18/0x41A1C

- 输入参数

eTime [in]：指向‘当前时间/日期’或‘闹钟的时间/日期’选项：

0：当前时间/日期(Current time)

1：闹钟时间/日期(Alarm time)

*sPt [in]：表示要设置的时间/日期，该变量为一个结构体，设置内容为：

u8cClockDisplay DRVRTC_CLOCK_12 / DRVRTC_CLOCK_24

u8cAmPm DRVRTC_AM / DRVRTC_PM

u32cSecond Second 数值

u32cMinute Minute 数值

u32cHour Hour 数值

u32cDayOfWeek Day of week

u32cDay Day 数值

u32cMonth Month 数值

u32Year Year 数值

- 包含头文件

Peripheral_lib/DrvRTC.h

- **函数返回值**

  0：设置成功

  其他：设置失败.

- **函数用法**

  /* 获取当前的时间和日期 */

  S_DRVRTC_TIME_DATA_T sCurTime;

  DrvRTC_Read(DRVRTC_CURRENT_TIME, &sCurTime);

13.3.20 DrvRTC_ClkConfig

- **函数**

  unsigned char DrvRTC_ClkConfig(unsigned char uclken);

- **函数功能**

  RTC 时钟源使能控制 设置寄存器0x40308[23] 。

- **输入参数**

  uClockSource [in]

  0： 关闭RTC的时钟源

  1： 开启RTC的时钟源

- **包含头文件**

  Peripheral_lib/DrvRTC.h

- **函数返回值**

  1： 设置失败

  0： 设置成功

- **函数用法**

  /*使能RTC 时钟源*/

  DrvRTC_ClkConfig(1);

13.3.21 DrvRTC_EnableWUEn

- **函数**

  void DrvRTC_EnableWUEn (void)

- **函数功能**

  开启RTC唤醒功能, WUEn=1b

  设置寄存器0x41A00[4]=1

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvRTC.h

- **函数返回值**

无

- **函数用法**

/* 开启RTC唤醒功能 */

```
DrvRTC_EnableWUEn();
```

13.3.22 DrvRTC_DisableWUEn

- **函数**

void DrvRTC_DisableWUEn (void)

- **函数功能**

关闭RTC唤醒功能, WUEn=0b

设置寄存器0x41A00[4]=0

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvRTC.h

- **函数返回值**

无

- **函数用法**

/* 关闭RTC唤醒功能 */

```
DrvRTC_DisableWUEn();
```

14 IIC 串行通讯 I2C

14.1 函数简介

该部分函数描述 IIC 通讯模块的控制，包含：

--IIC 启动控制

--IIC 工作模式控制

--IIC 通讯及收发数据配置控制

--IIC 中断向量控制

序号	函数名称	功能描述
01	DrvI2C_Open	开启 I2C 及配置 I2C 总线时钟
02	DrvI2C_Close	关闭I2C
03	DrvI2C_SlaveSetSlaveSet	开启I2C从机模式，设置从机地址及GC使能控制
04	DrvI2C_SetIOPin	开启并设置I2C通讯IO 口
05	DrvI2C_WriteData	发送1byte资料
06	DrvI2C_Write3ByteData	发送3byte资料
07	DrvI2C_ReadData	读取接收缓存器的数据
08	DrvI2C_Ctrl	设置I2C控制位：STA、STO、AA、SI，控制起始信号、停止信号及应答信号
09	DrvI2C_EnableInt	开启I2C中断功能
10	DrvI2C_DisableInt	关闭I2C中断功能
11	DrvI2C_ReadIntFlag	读取I2C中断请求标志位
12	DrvI2C_ClearIntFlag	清除I2C中断请求标志位
13	DrvI2C_ClearEIRQ	清除错误中断标志位
14	DrvI2C_ClearIRQ	清除I2C器件准备好标志位
15	DrvI2C_GetStatusFlag	读取I2C状态位
16	DrvI2C_TimeOutEnable	开启超时复位功能，设置时钟分频及超时控制值
17	DrvI2C_TimeOutDisable	关闭超时复位功能
18	DrvI2C_STSP	设置I2C发送起始信号或停止信号
19	DrvI2C_MGetACK	查询从机回馈的应答信号ACK/NACK
20	DrvI2C_DisableIOPin	关闭IO口复用作为I2C通讯口功能
21	DrvI2C_EnableSEn	开启I2C Slave模式功能
22	DrvI2C_DisableSEn	关闭I2C Slave模式功能
23	DrvI2C_EnableI2CEn	开启I2C

14.2 内部定义常量

E_DRVI2C_Status

标识符	数值	功能意义
E_DRVI2C_ARBITRATION_FLAG	0	仲裁漏失标志位
E_DRVI2C_GENERAL_CALL_FLAG	1	全呼标志位
E_DRVI2C_ACKNOWLEDGE_FLAG	2	应答信号状态标志位
E_DRVI2C_DATA_FIELD_FLAG	3	数据标志位
E_DRVI2C_RW_STATE_FLAG	4	读/写状态标志位
E_DRVI2C_RS_FLAG	5	接收停止或重新开始标志位
E_DRVI2C_SLAVE_ACTIVE_FLAG	6	从机模式有效标志位
E_DRVI2C_MASTER_ACTIVE_FLAG	7	主机模式有效标志位

E_DRVI2C_TIMEOUT_PRESCALE

标识符	数值	功能意义
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

标识符	数值	功能意义
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

标识符	数值	功能意义
E_DRVI2C_INT	1	开启I2C 中断功能

E_DRVI2C_ERROR_INT	2	开启I2C 错误中断功能
E_DRVI2C_INT_ALL	3	开启I2C 中断及错误中断功能

E_DRVI2C_SLAVE_BIT

标识符	数值	功能意义
E_DRVI2C_SLAVE_7BIT	0	从机7bit 地址码模式
E_DRVI2C_SLAVE_10BIT	1	从机10bit 地址码模式

14.3 函数说明

注意：只有使能 I2C 后才能对 I2C 其他寄存器设置。

14.3.1 DrvI2C_Open

- **函数**

  unsigned int DrvI2C_Open (uint32_t u32CRG);

- **函数功能**

  使能I2C功能，并设置I2C总线波特率；

  设置寄存器0x41000[0]=1,波特率写入寄存器0x41008[23:16] .

  注意：只有使能I2C后才能对I2C其他寄存器设置。

- **输入参数**

  u32CRG [in]

  总线波特率设置值CRG，设置范围 0~0xff.

  数据总线波特率 = (I2CLK/(4*(CRG+1)))

- **包含头文件**

  Peripheral_lib/DrvI2C.h

- **函数返回值**

  0：设置成功

  其他：设置失败

- **函数用法**

  /* 使能I2C，设置CRG =100 */

  DrvI2C_Open(100);

14.3.2 DrvI2C_Close

- **函数**

  void DrvI2C_Close (void);

- **函数功能**

  关闭I2C功能.，设置寄存器0x41000[0]=0。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvI2C.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭I2C */

DrvI2C_Close();

14.3.3 DrvI2C_SlaveSetSlaveSet

- 函数

unsigned int DrvI2C_SlaveSet(
uint32_t uSlaveAddr,
E_DRVI2C_SLAVE_BIT uAddrBit,
uint8_t uSlave3Byte,
uint8_t GC_Flag);

- 函数功能

使能I2C从机模式，并设置从机地址码及地址码模式，从机3byte数据发送模式设置，全呼模式GC设置。;
设置寄存器0x41004[7] /0x41004[5] /0x41000[2] /0x4100C[7:0]

- 输入参数

uSlaveAddr[in]： 从机地址码
7bit :0~0x7f，主要输入值为偶数，如0x00/0x02/0x0C等。
10bit: 0~0x3ff，主要输入值为偶数，即保持bit[0]为0，如0x30C/0x3CC等。
uAddrBit[in]： 从机地址码模式
0: 从机地址码为7bit
1: 从机地址码为10bit
uSlave3Byte[in]：从机3byte数据发送模式
0: 正常数据发送模式
1: 从机发送3byte数据模式
GC_Flag[in] 全呼模式设置
0: 正常呼叫模式
1: 使能全线广播模式

- 包含头文件

Peripheral_lib/DrvI2C.h

- 函数返回值

0：设置成功
其他：设置失败

- 函数用法

/* 使能从机模式，从机地址码为0x30，正常数据模式，正常呼叫模式*/
DrvI2C_SlaveSet(0x30,0,0,0);

14.3.4 DrvI2C_SetIOPin

- 函数

unsigned char DrvI2C_SetIOPin(unsigned int upin);

- **函数功能**

  设置I2C通讯IO口，设置寄存器0x40844[19 :16] 。

- **输入参数**

  upin[in]： 通讯IO选择

  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

- **包含头文件**

  Peripheral_lib/DrvI2C.h

- **函数返回值**

  无

- **函数用法**

  /* 使能通讯口 PT1.0 and PT1.1 */

  DrvI2C_SetIOPin(0);

14.3.5 DrvI2C_WriteData

- **函数**

  void DrvI2C_WriteData(uint8_t uData);

- **函数功能**

  写入待发送的资料，写入寄存器0x41014[7:0].

- **输入参数**

  uData [IN]： 待发送的数据, 1Byte 数据，输入范围 : 0~0xFF

- **包含头文件**

  Peripheral_lib/DrvI2C.h

- **函数返回值**

  无

- **函数用法**

  /*写入资料0x55至发送缓存器 */

  DrvI2C_WriteData(0x55);

14.3.6 DrvI2C_Write3ByteData

- **函数**

  void DrvI2C_Write3ByteData(uint8_t uData1,uData2,uData3);

- **函数功能**

  写入3byte待发送资料，连续发送3byte.

- **输入参数**

  uData1, uData2, uData3 [IN] 待发送的资料, 1Byte 数据, 输入范围 : 0~0xFF

- **包含头文件**

  Peripheral_lib/DrvI2C.h

- **函数返回值**

  无

- **函数用法**

  /* 写入3byte 数据0x11 0x22 0x33 准备发送 */

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

14.3.7 DrvI2C_ReadData

- **函数**

  unsigned char DrvI2C_ReadData(void);

- **函数功能**

  读取接收缓存器接收到的数据并控制主机返回应答或非应答信号.

  读取寄存器0x41010[7:0]的值。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvI2C.h

- **函数返回值**

  1Byte 缓存器接收到的数据

- **函数用法**

  /* 读取数据*/

  unsigned int data; data=DrvI2C_ReadData();

14.3.8 DrvI2C_Ctrl

- **函数**

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

- **函数功能**

  设置I2C控制位包括STA, STO, AA, SI，控制start信号、stop信号、I2C器件准备状态标志位及应答信号。

  设置寄存器0x41004[3:0]

- **输入参数**

start [in]：起始信号控制位

1：I2C产生start信号

0：I2C正常空闲。

stop [in]：停止信号控制位

1：I2C生产停止信号

0：I2C正常空闲。

intFlag [in]　I2C器件状态控制标志位

1：产生中断，接收到到9个clock后器件回应，此时SCL会被器件强行拉低，直到IRQFlag清零后释放SCL

0：正常，写入0，将会清除I2C器件状态控制旗标，使I2C往一个状态执行

ack [in]

1：ACK应答回复

0：未回复ACK或回复NACK信号

- **包含头文件**

Peripheral_lib/DrvI2C.h

- **函数返回值**

无

- **函数用法**

DrvI2C_Ctrl(0, 0, 1, 0); /* 清除I2C器件状态控制标志位IRQFlag */

DrvI2C_Ctrl(1, 0, 0, 0); /* 设置I2C 发送起始信号 */

14.3.9 DrvI2C_EnableInt

- **函数**

void DrvI2C_EnableInt(E_DRVI2C_INTERRUPT uINT)

- **函数功能**

使能I2C中断，包括收发中断及错误中断，属于中断向量HW0.

设置寄存器0x40000[21:20]

- **输入参数**

uINT[IN] :中断项选择

0 : I2C 收发中断使能

1 : I2C 错误中断使能

2 : I2C 收发中断及错误中断使能

- **包含头文件**

Peripheral_lib/DrvI2C.h

- **函数返回值**

无

- **函数用法**

/*I2C 收发中断使能*/

DrvI2C_EnableInt(1);

14.3.10 DrvI2C_DisableInt

- 函数

  void DrvI2C_DisableInt(E_DRVI2C_INTERRUPT uINT)

- 函数功能

  关闭I2C中断控制，包括收发中断和错误中断；

  清零寄存器0x40000[21:20]

- 输入参数

  uINT[IN]：中断项选择

  0：关闭I2C收发中断

  1：关闭I2C错误中断

  2：关闭I2C收发中断及错误中断

- 包含头文件

  Peripheral_lib/DrvI2C.h

- 函数返回值

  无

- 函数用法

  ```
  /* 关闭I2C收发中断 */
  DrvI2C_DisableInt(1);
  ```

14.3.11 DrvI2C_ReadIntFlag

- 函数

  E_DRVI2C_INTERRUPT DrvI2C_ReadIntFlag(void)

- 函数功能

  读取I2C中断标志位I2CEIF/I2CIF. 读取寄存器0x40000[5:4]的值

- 输入参数

  None

- 包含头文件

  Peripheral_lib/DrvI2C.h

- 函数返回值

  0：I2C 无中断请求

  1：I2C 收发中断请求标志位I2CIF

  2：I2C 错误中断请求标志位I2CEIF

  3：I2C 有收发中断请求标志位I2CIF和错误中断请求标志位I2CEIF

- 函数用法

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

14.3.12 DrvI2C_ClearIntFlag

- **函数**

  void DrvI2C_ClearIntFlag(E_DRVI2C_INTERRUPT uINT)

- **函数功能**

  清除I2C中断标志位I2CEIF/I2CIF. 清除寄存器0x40000[5:4]的值

- **输入参数**

  uINT[IN]

  0： 清除I2C中断标志位I2CIF

  1：清除I2C中断标志位I2CEIF

  2： 清除I2C中断标志位I2CEIF/I2CIF

- **包含头文件**

  Peripheral_lib/DrvI2C.h

- **函数返回值**

  无

- **函数用法**

  /*清除I2C中断标志位I2CIF */

  DrvI2C_ClearIntFlag(0);

14.3.13 DrvI2C_ClearEIRQ

- **函数**

  void DrvI2C_ClearEIRQ(void)

- **函数功能**

  清除错误中断旗标EIRQFlag，只有先清零超时标志位(TOFLAG)才能清零EIRQFlag，写入0就可清零；只有清除EIRQFlag才能清除中断请求标志位(I2CEIF)。

  设置寄存器0x41004[4]=0。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvI2C.h

- **函数返回值**

  无

- **函数用法**

  /*清除错误中断旗标EIRQ */

  DrvI2C_ClearEIRQ();

14.3.14 DrvI2C_ClearIRQ

- **函数**

  void DrvI2C_ClearIRQ(void)

- **函数功能**

  清除I2C器件状态控制标志位IRQFlag，IRQFlag从1变为0，使I2C执行下一个状态。

  清零寄存器0x41004[1]=0 。

  无论作为主机模式还是从机模式，只要接收到9个clock后，IRQFlag就被置1，此时SCL就会被拉低直到IRQFlag被清零，SCL得到释放，器件才能执行下一个状态动作。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvI2C.h

- **函数返回值**

  无

- **函数用法**

  /*清除I2C器件准备状态标志位IRQFlag */

  DrvI2C_ClearIRQ();

14.3.15 DrvI2C_GetStatusFlag

- **函数**

  unsigned char DrvI2C_GetStatusFlag(void)

- **函数功能**

  获取I2C状态标志位，返回一个8位的数据，读取寄存器0x41004[23:16]的值 。

- **输入参数**

- **包含头文件**

  Peripheral_lib/DrvI2C.h

- **函数返回值**

  返回值对应位表示的项目设置

  Bit 0： 仲裁漏失标志而为 ARB

  Bit 1: General Call Flag(GC)

  Bit 2: ACK应答标志位（A/NA）

  Bit 3: 数据标志位（DF）

  Bit 4: 读写状态标志位（R/W）

  Bit 5: 接收停止或重新开始标志（RX P/SR）

  Bit 6: 从机模式有效标志位（SAct）

  Bit 7: 主机模式有效标志位(MAct)

  ARB: uStatus=0

0：正常

1：仲裁漏失

GC: uStatus=1

0：正常I

1：当前全呼模式

A/NA: uStatus=2

0 : NACK已经发送或接收.

1 : ACK已经接收或发送.

DF: uStatus=3

0：正常

1：数据被发送或接收.

R/W: uStatus=4

0：写命令已被发送或接收

1：读命令已被发送或接收.

RX P/Sr: uStatus=5

0：正常

1：接收停止或重新开始信号已被发送或接收.

SAct：uStatus=6

0：从机模式无效

1 :从机模式有效

MAct：uStatus=7

0：主机模式无效

1：主机模式有效

- **函数用法**

/* 读取应答信号标志位 /

DrvRTC_GetStatusFlag(2); //读取应答信号ACK的状态标志位

14.3.16 DrvI2C_TimeOutEnable

- **函数**

unsigned char DrvI2C_TimeOutEnable(

E_DRVI2C_TIMEOUT_PRESCALE uPreScale,

E_DRVI2C_TIMEOUT_LIMIT uTimeOutLimit

);

- **函数功能**

使能超时复位功能，设置超时功能时钟分频及超时时间，

设置寄存器0x41000[1]=1，及寄存器0x41008[6:0] 。

- **输入参数**

uPreScale[in]: 时钟分频

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]： 超时时间设置

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

- **包含头文件**

  Peripheral_lib/DrvI2C.h

- **函数返回值**

  0：设置成功

  0xff：设置失败

- **函数用法**

  /*开启超时复位功能，设置频率分频器/ 32 ，及超时限制15 * CLKps Cycle */

  DrvI2C_TimeOutEnable(5,14);

14.3.17 DrvI2C_TimeOutDisable

- **函数**

void DrvI2C_TimeOutDisable(void)

- **函数功能**

关闭超时复位功能，设置寄存器0x41000[1] =0 。

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvI2C.h

- **函数返回值**

无

- **函数用法**

/* 关闭超时复位功能 */

DrvI2C_TimeOutDisable();

14.3.18 DrvI2C_STSP

- **函数**

  void DrvI2C_STSP(unsigned char usignal);

- **函数功能**

  启动I2C的起始信号或停止信号，设置寄存器0x41004[3:2] 。

- **输入参数**

  usignal[in]： 信号模式设置

  0 启动起始信号

  1 启动停止信号

- **包含头文件**

  Peripheral_lib/DrvI2C.h

- **函数返回值**

  无

- **函数用法**

  /* 启动起始信号*/

  DrvI2C_STSP(0);

14.3.19 DrvI2C_MGetACK

- **函数**

  unsigned char DrvI2C_MGetACK(unsigned int utime);

- **函数功能**

  在设定的时间内查询从机回馈的应答信号，设置寄存器0x41004[1]

- **输入参数**

  utimel[in] :设定的查询时间0~0xffff

- **包含头文件**

  Peripheral_lib/DrvI2C.h

- **函数返回值**

  0 查询到应答信号ACK标明发送成功

  1 查询到非应答信号NACK，标明发送失败

- **函数用法**

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

  Err_flag=DrvI2C_MGetACK(0xffff);

14.3.20 DrvI2C_DisableIOPin

- **函数**

  void DrvI2C_DisableIOPin(void)

- **函数功能**

  关闭I2C通讯口功能，设置寄存器0x40844[16]=0

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvI2C.h

- **函数返回值**

  无

- **函数用法**

  /*关闭I2C的通讯IO口*/

  DrvI2C_DisableIOPin();

14.3.21 DrvI2C_EnableSEn

- **函数**

  void DrvI2C_EnableSEn (void);

- **函数功能**

  开启I2C Slave模式功能.，设置寄存器0x41004[7]=1。

- **输入参数**

  无

- **包含头文件**

  Peripheral_lib/DrvI2C.h

- **函数返回值**

  无

- **函数用法**

  /* 开启I2C Slave模式功能 */

  DrvI2C_EnableSEn();

14.3.22 DrvI2C_DisableSEn

- **函数**

  void DrvI2C_DisableSEn (void);

- **函数功能**

  关闭I2C Slave模式功能.，设置寄存器0x41004[7]=0。

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvI2C.h

- **函数返回值**

无

- **函数用法**

/* 关闭I2C Slave模式功能 */

DrvI2C_DisableSEn();

14.3.23 DrvI2C_EnableI2CEn

- **函数**

void DrvI2C_EnableI2CEn (void);

- **函数功能**

开启I2C功能.，设置寄存器0x41000[0]=1。

- **输入参数**

无

- **包含头文件**

Peripheral_lib/DrvI2C.h

- **函数返回值**

无

- **函数用法**

/* 开启I2C功能 */

DrvI2C_EnableI2CEn();

15 Flash 读写

15.1 函数简介

该部分函数描述芯片 Flash 区域的读写操作，包含：

--Flash 的数据写入与读取

--Flash 的字/页写入、字/页的读取；

--Flash 的数据擦除

序号	函数名称	功能描述
01	DrvFlash_Burn_Word	写入一个字资料到flash
02	ROM_BurnPage	写入连续一页的32个字资料到flash
03	ReadWord	读取flash的一个字的数据
04	ReadPage	读取flash连续一页32个字的数据
05	PageErase	擦除flash一页连续的资料
06	SectorErase	擦除一个sector的资料

15.2 函数说明

注意1：在执行Flash刻录与读取程序指令之前，必须先执行SYS_DisableGIE()；关闭全局使能中断，这可以避免程序运行异常的行为发生.

注意2：执行Flash刻录指令，必须确保芯片工作电压VDD3V高于2.7V，如果芯片电压VDD3V低于2.7V，则可能会发生刻录错误行为.

15.2.1 DrvFlash_Burn_Word

- **函数**

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

- **函数功能**

写入一个word的数据至Flash的对应地址。

注意 ：使用该函数做地址数据写入功能之后，请再使用Read功能回读数据做验证动作确保数据以正确被写入.

- **输入参数**

addr[in]：待写入数据的地址

16位的地址，Flash空间是从0x90000开始，所以该地址值只需写16位值，输入范围0~0xffff，且地址的间隔步长为4；如要想0x9A880写入一个word数据，函数参数只需填写0xA880值就可以。

Delay time[in]：刻录延时

data [in]：带写入的数据，输入范围0~0xffffffff

- **包含头文件**

Peripheral_lib/Drvflash.h

- **函数返回值**

0 设置成功

- **函数用法**

/* 写入0xFF05到flash的0x90880地址 */

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

注意 ：执行Flash刻录指令，必须确保芯片工作电压VDD3V高于2.7V，如果芯片电压VDD3V低于2.7V，则可能会发生刻录错误行为.

15.2.2 ROM_BurnPage

- **函数**

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

- **函数功能**

一次性写入一页32个word的数据到Flash对应的连续地址。

注意：使用该函数做地址数据写入功能之后，请再使用Read功能回读数据做验证动作确保数据以正确被写入.

- **输入参数**

addr[in]：待写入数据的首地址，16位的地址，Flash空间是从0x90000开始，所以该地址值只需写16位值，输入范围0~0xffff，且地址的间隔步长为128（32*4），且不能进行跨页写入，因为每个page只有128byte，所以地址只能为0xuu00或0xuu80；如要想从0x9A880开始，函数参数只需填写0xA880值就可以。

Delay time[in]：刻录延时

data [in]：待写入的数据，属于指针类型，数据的长度为32word，每个数据的输入范围0~0xffffffff

- **包含头文件**

Peripheral_lib/Drvflash.h

- **函数返回值**

0xFF 设置成功

- **函数用法**

/* 从地址0x90880开始连续一次写入32word的数据 */

int *A[32]={0};

ROM_BurnPage(0x0880, 0x2000, A);

注意：执行Flash刻录指令，必须确保芯片工作电压VDD3V高于2.7V，如果芯片电压VDD3V低于2.7V，则可能会发生刻录错误行为.

15.2.3 ReadWord

- **函数**

int ReadWord(unsigned int addr);

- **函数功能**

读取Flash对应地址的一个word的数据。

- **输入参数**

addr[in]：待读取数据的地址

16位的地址，Flash空间是从0x90000开始，所以该地址值只需写16位值，输入范围0~0xffff，且地址的间隔步长为4；如要想0x9A880读取一个word数据，函数参数只需填写0xA880值就可以。

- **包含头文件**

Peripheral_lib/Drvflash.h

- **函数返回值**

返回一个word的数据

- **函数用法**

/* 读取Flash的0x90880地址的数据 */

Int flag; flag= ReadWord(0x0880);

15.2.4 ReadPage

- **函数**

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

- **函数功能**

一次性连续读取flash对应连续地址的32个word的数据

- **输入参数**

addr[in]：待读取数据的首地址，16位的地址，Flash空间是从0x90000开始，所以该地址值只需写16位值，输入范围0~0xffff，且地址的间隔步长为128（32*4），且不能进行跨页读取，因为每个page只有128byte，所以地址只能为0xuu00或0xuu80；如要想从0x9A880开始，函数参数只需填写0xA880值就可以。

data [in]：用于保存读取到的数据，属于指针类型，数据的长度为32word，每个数据的输入范围0~0xffffffff

- **包含头文件**

Peripheral_lib/DrvFlash.h

- **函数返回值**

0xFF 设置成功

- **函数用法**

```
/* 读取flash以0x90880地址开始的连续32个word的资料 */
int *A[32]={0};
ReadPage(0x0880, A);
```

15.2.5 PageErase

- **函数**

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

- **函数功能**

一次性清除flash中对应连续地址的32个word的数据

- **输入参数**

addr[in]：待清除数据的首地址，16位的地址，Flash空间是从0x90000开始，所以该地址值只需写16位值，输入范围0~0xffff，且地址的间隔步长为128（32*4），且不能进行跨页写入，因为每个page只有128byte，所以地址只能为0xuu00或0xuu80；如要想从0x9A880开始，函数参数只需填写0xA880值就可以。

Delay time[in]：延时时间

- **包含头文件**

Peripheral_lib/DrvFlash.h

- **函数返回值**

0xFF 设置成功

- **函数用法**

```
/*清除从0x90880地址开始连续32个word数据 */
PageErase(0x0880, 0x2000);
```

15.2.6 SectorErase

- **函数**

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

- **函数功能**

清除flash对应地址的一个sector的数据

- **输入参数**

addr[in]：待清除数据的首地址，16位的地址，Flash空间是从0x90000开始，所以该地址值只需写16位值，输入范围0~0xffff，且一个sector包含32page，所以地址的间隔步长为32*128；所以首地址是以页来计算的，且需要对齐，地址为0xu000（u为可变的值）。如要想从0x91000开始，函数参数只需填写0x1000值就可以。

Delay time[in]：延时时间

- **包含头文件**

Peripheral_lib/DrvFlash.h

- **函数返回值**

0xFF 设置成功

- **函数用法**

/* 从0x91000地址开始连续清除一个sector的数据*/

SectorErase(0x1000, 0x2000);

15.3 Flash 存储空间结构

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();//修改返回值输出
		2、 DrvPWM_Condition();//将函数的输入参数的位置进行调换
		UART
		3、 DrvUART_Clk();//添加该函数，用于选择 UART 的时钟源
		4、 DrvUART_GetRxFlag();/DrvUART_GetRxFlag（）; //修改函数无法读取 RXIF/TXIF 的问题
		5、 DrvUART_CheckTRMT();// 添加该函数对 TRMT bit 进行操作。
		6、 DrvUART_Open();//函数中对 CLK 的设置删除。
		7、 DrvUART_EnableInt();// 修改头文件 DrvUART.H 的 TX_IE/RX_IE 的地址为 19/18
		ADC
		8、 DrvADC_GetConversionData(void)；
		操作：修改函数的返回值类型为 int 类型
		9、 DrvADC_ClkEnable(CLK,DIV);//修改函数由于设置为 CPU LOW EDGE 时无法打开 ADC clock；
		10、 DrvADC_ReadIntFlag(void)；//修改函数无法读取到 Interrupt 的标志位问题。
		CMP
		11、 DrvCMP_ReadIntFlag();//优化函数无法读取中断标志位问题；
		12、 DrvCMP_OuputFilter();// 修函函数名字为 DrvCMP_OutputFilter();使用上才方便
		13、 DrvCMP_RLO_Refv();//优化函数设置参考电压位置的问题
		SYSTEM
		14、 SYS_LowPower(umode) //添加函数直接实现
			2013/12/31

		sleep、idle、wait mode 功能。
		15、 SYS_EnableGIE();//优化函数底层没有打开所有的 system interrupt，导致多个中断在一起无法响应。
		I2C
		16、 DrvI2C_Open ( ) ;//优化函数不能正确的设置 CRG 问题；
		17、 DrvI2C_SetIOPin( ) ;//添加 IO 选择设置函数。
		18、 DrvI2C_ReadData ( ) ; 优化函数不能读取到 SID 上的数据问题；
		19、 DrvI2C_WriteData ( ) //函数无法正确地将数据发送出去问题；
		20、 DrvI2C_MGetACK();//添加函数来检查 slaver 发送回来的 ACK。
		21、 DrvI2C_EnableInt ( ) ; //优化函数不能正确设置 IIC 中断使能位问题；
		22、 DrvI2C_DisableInt();//修改函数不能正确设置 IIC 中断无效位问题；
		23、 DrvI2C_ClearIntFlag();//优化函数不能正确清零中断标志位问题
		24、 DrvI2C_SlaveSet();//优化函数对于 slaver 的 address 设置有误问题。
		CLOCK
		25、 DrvCLOCK_EnableLowOSC(unsigned int uSource,unsigned int udelay);//优化函数，增加输入参数，调整 lowOSC 的启动时间为外部参数控制。
		GPIO
		26、 DrvGPIO_IntTrigger(uint32_t port, uint32_t i32Bit, uint32_t mode)；//优化函数不能同时设置不同 IO 口为不同的触发沿的问题。	

V05	ALL	1、修改 IIC 的收发函数为初始的方式； 2、修改 GPIO 的工作模式开启函数： DrvGPIO_Open ( ) ; 3、添加工作模式关闭函数：DrvGPIO_Close ( ) ;	2014/6/24
V06	ALL	添加 flash 函数说明； 添加函数 SYS_INTPriority()说明 添加 flash 存储空间结构说明	2017/8/21

		删除 flash 函数 MassErase()说明；	
V07	ALL	重新编译 C LIB 函数库：编译方式添加优化设置参数添加‘-ffunction-section -fdata-section -OS’	2014/9/30
V08	ALL	添加一个函数：DrvGPIO_EnableAnalogPin（ ）； 修改 RTC 函数：DrvRTC_HourFormat（ ）的参数定义说明；	2014/11/17

V09	ALL	1.添加函数 HAO 频率校正
		void DrvCLOCK_CalibrateHAO(short int uMHZ)；
		2.修改函数：DrvGPIO_Open（ ），屏蔽在设置输入/输出模式时，会将对应引脚的输出/输入模式关闭的功能；
		3.屏蔽 IP 模块函数对底层中断向量的设置，对 ir14 的设置功能；
		4.修改函数 DrvGPIO_GetBit ()；改正不能读取 bit0 值的错误。
		5.修改 SYS_EnableGIE()函数的说明；
		6.删除函数：DrvSPI32_SetCS()的函数说明；
		7.新曾函数 :DrvSPI32_SetCSO()并添加函数说明；
		8.添加 PT1/PT2/PT3 精简代码操作函数：
		以 DrvGPIO_PT1_为开头，如：
		void DrvGPIO_PT1_EnableINPUT(short int ubit);
		void DrvGPIO_PT1_DisableINPUT(short int ubit);
		void DrvGPIO_PT1_EnablePullHigh(short int ubit);
		以 DrvGPIO_PT2_为开头，如：
		void DrvGPIO_PT2_EnableINPUT(short int ubit);
		void DrvGPIO_PT2_DisableINPUT(short int ubit);
		void DrvGPIO_PT2_EnablePullHigh(short int ubit);
		以 DrvGPIO_PT3_为开头，如：void DrvGPIO_PT3_EnableINPUT(short int ubit);
		void DrvGPIO_PT3_DisableINPUT(short int ubit);
		void DrvGPIO_PT3_EnablePullHigh(short int ubit);
			2015/03/25

V10	ALL	1.更新各个 IP 模块图 ·包括部分符号的更新如 :ADC 的输入端 OPO 更新为 OPOI, REFO 更新为 REFO_I, OPAMP 的 OPNS[3]更正为 OPOI, OPNS[4]更正为 OPO, DAO 更新为 DAOI 2.函数 DrvOP_OutputWithCPCLK()更名为 DrvOP_OutputWithCHPCK() 3.函数 DrvRTC_Write()可以正确设置 12 或 24 小时制；12 小时制的输入范围：00:00~11:59;	2015/06/16
V11	ALL	1、更新排版	2015/09/23
V12	P198	修改函数 DrvUART_Open():的输入参数‘uclock’的数据类型为‘float’; 修改函数 DrvUART_Open():函数库 BaudRate 的计算算法； 更新 C LIB；	2016-1-15
V13	P198	更新 C LIB	2016-04-15
V14	All	1. 移除各章节内的应用方块图 2. Flash 读写章节范例程序描述的 Delay time 时间从 10 修改为 0x2000 3. .DrvUART_Open 输入数据参数从 float 修正为 unsigned int 4. 修正各章节内容缓存器描述错误部分 5. 重新排板.	2017-02-20
V15	All	1. 新增 Flash 自我刻录注意事项, 工作电压 VDD3V 必须高于 2.7V.	2018-02-18
V16	章节 3 章节 13 章节 14 章节 16	新增函式 DrvCLOCK_EnableENHAO 與 DrvCLOCK_SelectIHOSC_CalHAO 新增函式 DrvRTC_EnableWUEn 与 DrvRTC_DisableWUEn 新增函式 DrvI2C_EnableSEn 与 DrvI2C_DisableSEn 与 DrvI2C_EnableI2Cen 修正 Flash Function 的函数返回值描述	2022/12/15

17 C Library Change List

Date	旧版本 Queries List		新版本改善	
	版本	Bug List	版本	改善
2015-2-28	V0.8	函数 DrvGPIO_GetBit()无法读取 BIT0 的输入状态值	V0.9	修改函数位操作算法，能正常读取所有 bit 的输入状态值；
		每个功能模块的中断向量使能设置函数，在开启或关闭中断使能时，出现同时关闭其他功能模块的中断使能，导致其他已开启的中断功能无法正常使用		删除每个功能模块函数中断设置函数里屏蔽关闭其他模块中断使能的程序。
		DrvGPIO_Open（ ）函数： 在使能输入（输出）模式时，会同时关闭输出（输入）模式；		删除关闭输出（输入）模式程序；
		缺少 HAO 频率校正用户功能函数		添加函数： DrvCLOCK_CalibrateHAO(); 用户可调用此函数做 HAO 频率校正；
		单独操作一个 IO 口，原有函数库编译后的代码比较大		添加 PT1/PT2/PT3 的对应单独操作的精简代码函数：
				新增函数 DrvSPI32_SetCSO();

Date	旧版本 Queries List		新版本改善	
	版本	Bug List	版本	改善
2015-6-16	V0.9	函数 DrvRTC_Write()错误设置 HRF 小时格式设置	V1.0	修改函数位操作算法，能正常 HRF 的值；12 小时制的输入范围：00:00~11:59
		IP 模块图更新		头文件 enum 符号定义的更新:修正 ADC 的输入端 OPO 更新为 OPOI, REFO 更新为 REFO_I, OPAMP 的 OPNS[3]更正为 OPOI, OPNS[4]更正为 OPO, DAO 更新为 DAOI
				新增函数 DrvOP_OutputWithCHPCK (uCHPCK)

Date	旧版本 Queries List	新版本改善

	版本	Bug List	版本	改善
2015-09-23	V1.0	函数 int DrvGPIO_IntTrigger（ ）有开启 IO 口外部中断功能	V1.1	删除开启 IO 口外部中断功能的程序
		修改函数： DrvUART_GetPERR(); DrvUART_GetFERR(); DrvUART_GetOERR(); DrvUART_GetABDOVF();		修改函数的位操作算法，使能正确读取到对应位的值；

Date	旧版本 Queries List		新版本改善	
2016-01-15	版本	Bug List	版本	改善
	V1.1	函数 DrvUART_Open(): 频率为小数时不能正确计算；	V1.2	修改输入参数‘uclock’的数据类型为‘float’；
		函数 DrvUART_Open(): 函数库 Baud Rate 的计算公式有错误		修改函数库的计算公式算法；

Date	旧版本 Queries List		新版本改善	
2016-04-15	版本	Bug List	版本	改善
	V1.2	函数 DrvUART_Open(): 函数库 Baud Rate 的计算由于数据类型转换导致四舍五入误差大	V1.3	修改函数库的计算公式算法；

Date	旧版本 Queries List		新版本改善	
2017-02-20	版本	Bug List	版本	改善
	V1.3	函式无法正确执行，部分函式撰写方式无统一规格 SYS_SleepFlagRead SYS_WdogFlagRead SYS_ResetFlagRead SYS_BOR_FlagRead SYS_SleepFlagClear SYS_WdogFlagClear SYS_ResetFlagClear SYS_BOR_FlagClear DrvWDT_CounterRead DrvTIMER_GetIntFlag DrvGPIO_SetPortBits DrvADC_ReadIntFlag DrvSPI32_GetDCFlag DrvSPI32_IsBusy DrvSPI32_IsABFlag DrvSPI32_IsOVFlag DrvSPI32_IsRxFlag DrvSPI32_IsRxBufferFull DrvSPI32_IsTxBufferFull	V1.4	1. 统一部分函式撰写风格 2. 缩减 DrvUART_Open 函式占 code size 资源问题 3. 新增函式 DrvWDT_ResetEnable DrvOP_OPDEN

		DrvSPI32_EnableIO DrvUART_Open DrvUART_GetTxFlag DrvUART_GetRxFlag DrvOP_ReadIntFlag DrvRTC_ReadIntFlag 新增函式 DrvWDT_ResetEnable DrvOP_OPDEN 修改文件内容 SpecialMacro.h DrvOP.h DrvUART.h DrvSPI32.h DrvTimer.h		
2018-02-18	V1.4	1. 使用函式 DrvI2C_SlaveSet 没有功能 2. 使用以下函式 uint32_t DrvSPI32_IsRxBufferFull(void) uint32_t DrvSPI32_IsTxBufferFull(void) 没有功能	V1.5	1. 修正 DrvI2C_SlaveSet 函式使用问题. 2. 修正 uint32_t DrvSPI32_IsRxBufferFull(void) uint32_t DrvSPI32_IsTxBufferFull(void) 函式用问题
2022-12-15	V1.5	1. DrvPWM_Open 函式设置问题，从 111 开始往 110 设置会失效. 2. DrvUART_Open/ DrvUART2_Open 除频设置问题，在某些数值运算会有 N+1 的误差	V1.7	1. 修正函式 DrvPWM_Open, DrvUART_Open/ DrvUART2_Open 2. 新增函式 DrvRTC_EnableWUEn, DrvRTC_DisableWUEn, DrvCLOCK_EnableENHAO, DrvI2C_EnableSEn, DrvI2C_DisableSEn, DrvI2C_EnableI2Cen 3. 新增函式 DrvCLOCK_SelectIHOSC_CalHAO