HYCON TECHNOLOGY

HY11P 系列

C 函数库用户手册

目录

1. 导读

1.1 C 函数库简介

本文件用于描述HYCON HY11P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开发来有效的提高整个产品的开发效率。

文件中C 函数库的每一个函数都带有说明、用法及使用例程，所有的函数都存在我们HYCON提供的C IDE安装目录下的Driver文件夹里。

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，获取其他相关的资料。

下载文档的网址：

http://www.hycontek.com/cn/products-cn/6083

(1)HYCON HY11P Series Data Sheet

(2)HYCON HY11P Series User's Guide

(3)HYCON HY11P Series Hardware TOOL User Manual

(4)HYCON HY11P Series Software TOOL User Manual

2. MCU 系统控制

2.1 函数简介

该部分函数描述晶片中断系统控制及 MCU 状态读取，包含：

--工作模式（休眠模式（sleep）、待机模式（Idle））的控制

--全局中断的控制

--MCU 程序状态读取

--包含 RST.h/INT.h 头文件

序号	函数名称	功能描述
01	Sleep	启动低功耗睡眠模式；
02	Idle	启动低功耗待机模式；
03	INT_GIE_Enable	使能全局中断；
04	INT_GIE_Disable	关闭全局中断；
05	SYS_ReadSKERR	读取堆栈错误复位标志位(SKERR)的值
06	SYS_ReadBOR	读取电源干扰复位标志位(BOR)的值
07	SYS_ReadTO	读取看门狗计数复位标志位(TO)的值

2.2 函数说明

2.2.1 Sleep

- **函数**

  Sleep();

- **函数功能**

  启动低功耗睡眠模式。

- **输入参数**

  无

- **包含头文件**

  Driver/RST.h

- **函数返回值**

  无

- **函数用法**

```
/* 休眠之前需要将所有不用功能关闭，然后调用睡眠函数，使IC进入睡眠模式 */
Sleep();
```

2.2.2 Idle

- 函数

  Idle();

- 函数功能

  启动低功耗待机模式。

- 输入参数

  无

- 包含头文件

  Driver/RST.h

- 函数返回值

  无

- 函数用法

  /* 休眠之前需要将所有不用功能关闭，然后调用待机函数，使IC进入待机模式 */

  Idle( );

2.2.3 INT_GIE_Enable

- 函数

  INT_GIE_Enable();

- 函数功能

  使能全局中断，设置寄存器INTE1[7]的值。

- 输入参数

  无

- 包含头文件

  Driver/INT.h

- 函数返回值

  无

- 函数用法

  /* 使能全局中断 */

  INT_GIE_Enable();

2.2.4 INT_GIE_Disable

- 函数

  INT_GIE_Disable();

- 函数功能

  关闭全局中断，清零寄存器INTE1[7]的值。

- 输入参数

  无

- 包含头文件

Driver/INT.h

- **函数返回值**

无

- **函数用法**

/* 关闭全局中断 */

INT_GIE_Disable();

2.2.5 SYS_ReadSKERR

- **函数**

SYS_ReadSKERR();

- **函数功能**

读取堆栈错误复位标志位(SKERR)的值，读取寄存器PSTATU[2]的值。

- **输入参数**

无

- **包含头文件**

Driver/RST.h

- **函数返回值**

0x00：清除需透过BOR、RESET或指令

0x04：堆栈错误

- **函数用法**

/* 读取堆栈错误复位标志位 */

unsigned char flag;

flag = SYS_ReadSKERR();

2.2.6 SYS_ReadBOR

- **函数**

SYS_ReadBOR();

- **函数功能**

读取电源干扰复位标志位(BOR)的值，清零寄存器PSTATU[4]的值。

- **输入参数**

无

- **包含头文件**

Driver/RST.h

- **函数返回值**

0x00：未发生复位

0x10：发生电源干扰复位

- **函数用法**

/* 读取电源干扰复位标志位(BOR)的值 */

```
unsigned char flag;
flag = SYS_ReadBOR();
```

2.2.7 SYS_ReadTO

- **函数**

  SYS_ReadTO();

- **函数功能**

  读取看门狗计数复位标志位(TO)的值，读取寄存器PSTATU[6]的值。

- **输入参数**

  无

- **包含头文件**

  Driver/RST.h

- **函数返回值**

  0x00: 正常，清除需透过BOR、RST或指令

  0x40: 看门狗计数溢出复位

- **函数用法**

```
/* 读取看门狗计数复位标志位(TO)的值 */
unsigned char flag;
flag = SYS_ReadTO();
```

3. 晶片时钟源 CLOCK

3.1 函数简介

函数描述 MCU 及其他功能模块的时钟源操作，包含：

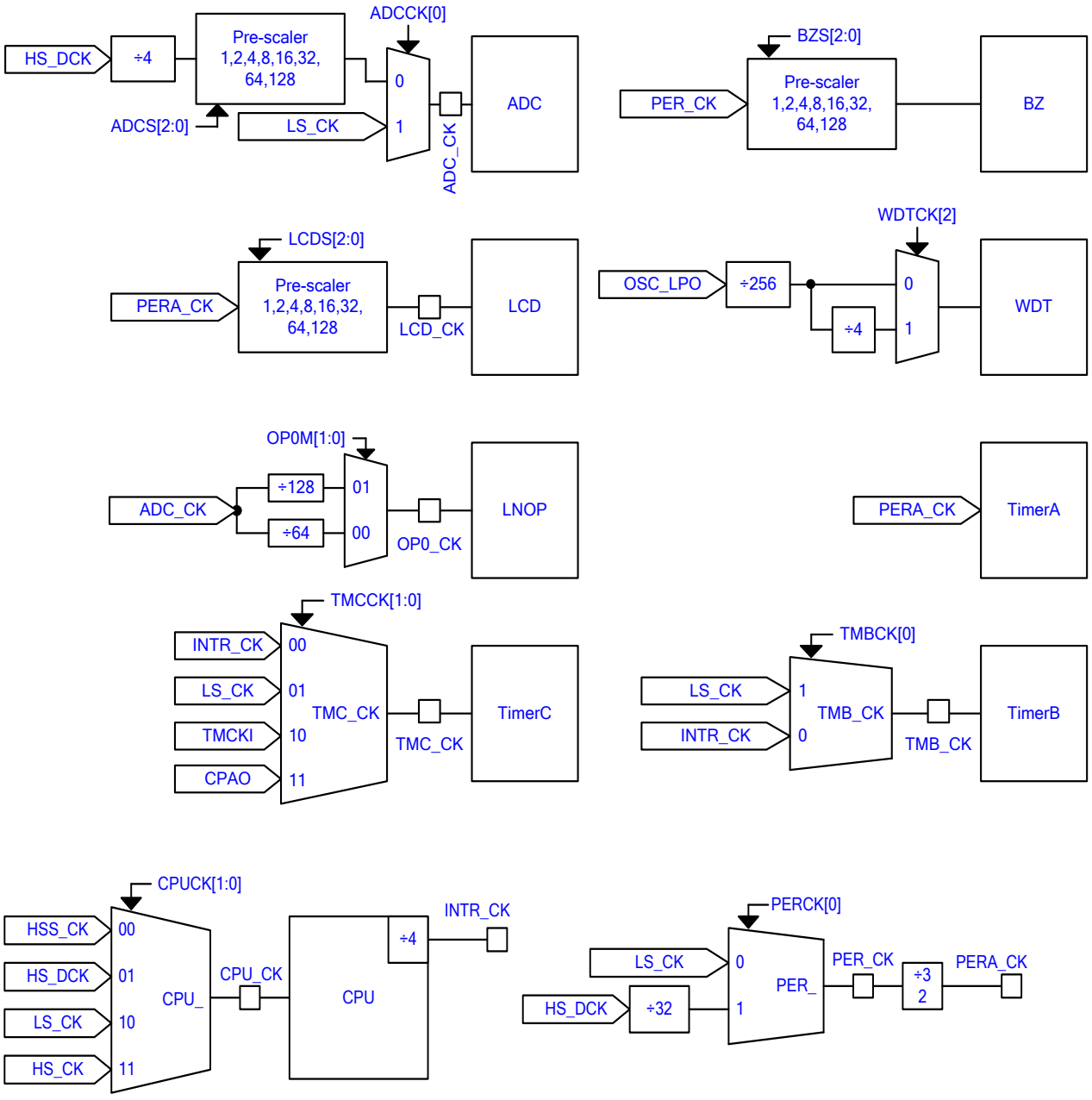
--内部高速及低速频率的控制

--外部高速及低速晶振的控制

--MCU 周边功能模块时钟源控制

序号	函数名称	功能描述
01	CLK_GPIO_OpenXTInput	设置PT2.0/PT2.1作为外部晶振输入引脚
02	CLK_CPUCKOpen	设置晶片工作频率及外部晶振
03	CLK_ExtHSCKEnable	开启外部高速晶振
04	CLK_ExtLSCKEnable	开启外部低速晶振
05	CLK_ExtCKDisable	关闭外部晶振
06	CLK_ExtHSCkSelect	选择外部高速晶振作为晶片高速频率来源
07	CLK_ExtLSCkSelect	选择外部低速晶振作为晶片低速频率来源
08	CLK_HAOEnable	开启内部高速频率HAO
09	CLK_HAODisable	关闭内部高速HAO
10	CLK_HAOHSCkSel	选择内部HAO作为晶片高速频率来源
11	CLK_LPOHSCkSel	选择内部LPO作为晶片高速频率来源
12	CLK_CPUCKSel	选择晶片CPU频率来源，HSS_CK频率分频
13	CLK_PERCKHSDCKSel	选择HSD_CK作为周边频率PERCK的频率来源
14	CLK_PERCKLSCKSel	选择LS_CK作为周边频率PERCK的频率来源
15	CLK_LPOHSCkSel	选择内部LPO作为晶片高速频率来源
16	CLK_LCDCKSel	设置LCD工作频率预分频
17	CLK_BZCKSel	设置蜂鸣器驱动频率预分频
18	CLK_ADCCKSel	设置ADC工作频率来源
19	CLK_ADCCKDivSel	设置ADC工作频率预分频

3.2 CLOCK 模块方框图



3.3 函数说明

3.3.1 CLK_GPIO_OpenXTInput

- **函数**

  void CLK_GPIO_OpenXTInput(void);

- **函数功能**

  设置PT2.0/PT2.1作为外部晶振输入引脚。

- **输入参数**

  无

- **包含头文件**

  Driver/CLK.h

- **函数返回值**

  无

- **函数用法**

```
/* 设置PT2.0/PT2.1作为外部晶振输入引脚 */
CLK_GPIO_OpenXTInput();
```

3.3.2 CLK_CPUCKOpen

- **函数**

```
void CLK_CPUCKOpen(
        unsigned char CPUCKSel,
        unsigned char ExtCKEn,
        unsigned char HSSCKDiv,
        unsigned char VXTSP,
        unsigned char oscSOURCE)；
```

- **函数功能**

  设置晶片的CPU工作频率来源及高速频率HSS的分频设置，设置晶片外部晶振开关机高低频率选择，设置寄存器MCKCN1/MCKCN2。

- **函数输入参数**

  CPUCKSel [in] : 选择CPU_CK来源

  MCKCN2_CPUCK_HSCK：CPU_CK频率来源于HS_CK

  MCKCN2_CPUCK_LSCK：CPU_CK频率来源于LS_CK

  MCKCN2_CPUCK_HSDCK：CPU_CK频率来源于HSD_CK

  MCKCN2_CPUCK_HSSCK：CPU_CK频率来源于HSS_CK

  ExtCKEn [in]：开关外部晶振

  MCKCN1_ENXT_ENABLE：开启外部晶振

  MCKCN1_ENXT_DISABLE：关闭外部晶振

HSSCKDiv [in]：设置高速频率HSS分频

MCKCN2_HSS_HSCKDIV8：高速频率HSS进行8分频

MCKCN2_HSS_HSCKDIV4：高速频率HSS进行4分频

MCKCN2_HSS_HSCKDIV2：高速频率HSS进行2分频

MCKCN2_HSS_HSCKDIV1：高速频率HSS进行1分频

VXTSP [in]：选择外部晶振高频或低频

MCKCN1_XTSP_XTS：外部高速频率2~8MHZ

MCKCN1_XTSP_XTL：外部低速频率32768HZ

OscSOURCE [in]：选择晶片工作频率来源

当外部晶振开启时：

MCKCN2_LSCK_CY ：晶片低速频率来源为外部晶振

MCKCN2_LSCK_LPO：晶片低速频率拉远为内部LPO

当外部晶振关闭时：

MCKCN2_LSCK_CY ：不能设置

MCKCN2_LSCK_LPO：晶片低速频率拉远为内部LPO

当外部晶振开启时：

MCKCN2_HSCK_CY：晶片高速频率来源为外部晶振

MCKCN2_HSCK_HAO：晶片高速频率来源为内部HAO

当外部晶振关闭时：

MCKCN2_HSCK_CY：不能设置

MCKCN2_HSCK_HAO：晶片高速频率来源为内部HAO

- **包含头文件**

Driver/ CLK.h

- **函数返回值**

无

- **函数用法**

```
/* 关闭外部高速晶震，选择HAO作为MCU频率源，HSS进行1分频，且高速频率HSD_CK作为CPU时钟源 */
CLK_CPUCKOpen(
MCKCN2_CPUCK_HSDCK,
MCKCN1_ENXT_DISABLE,
MCKCN2_HSS_HSCKDIV1,
MCKCN1_XTSP_XTL,
MCKCN2_HSCK_HAO);
```

3.3.3 CLK_ExtHSCKEnable

- **函数**

CLK_ExtHSCKEnable();

- **函数功能**

开启外部高速晶振，操作寄存器MCKCN1[2:1]=11B。

- **函数输入参数**

无

- **包含头文件**

Driver/CKL.h

- **函数返回值**

无

- **函数用法**

```
/* 开启外部晶振 */
CLK_GPIO_OpenXTInput();          //开启外部高速晶振引脚
CLK_ExtHSCKEnable();             //开启外部高速晶振
```

3.3.4 CLK_ExtLSCKEnable

- **函数**

CLK_ExtLSCKEnable();

- **函数功能**

开启外部低速晶振，操作寄存器MCKCN1[2:1]=10B。

- **输入参数**

无

- **包含头文件**

Driver/CLK.h

- **函数返回值**

无

- **函数用法**

```
/* 开启外部低速晶振 */
CLK_GPIO_OpenXTInput();          //开启外部高速晶振引脚
CLK_ExtLSCKEnable();             //开启外部低速晶振
```

3.3.5 CLK_ExtCKDisable

- **函数**

CLK_ExtCKDisable();

- **函数功能**

关闭外部晶振，操作寄存器MCKCN1[1]=0B。

- **输入参数**

无

- **包含头文件**

Driver/CLK.h

- **函数返回值**

无

- 函数用法

```
/* 关闭外部晶振 */
CLK_ExtCKDisable();                    //关闭外部晶振
```

3.3.6 CLK_ExtHSCKSelect

- 函数

CLK_ExtHSCkSelect();

- 函数功能

选择外部高速晶振作为晶片高速频率来源，在设置前需要先开启外部高速晶振并等待稳定，寄存器MCKCN2[4]=1。

- 输入参数

无

- 包含头文件

Driver/CLK.h

- 函数返回值

无

- 函数用法

```
/* 选择外部高速晶震作为晶片高速频率来源 */
CLK_GPIO_OpenXTInput();                //开启外部高速晶振引脚
CLK_ExtHSCKEnable();                   //开启外部高速晶振
CLK_ExtHSCkSelect();                   //选择晶片高速频率源为外部晶振
```

3.3.7 CLK_ExtLSCkSelect

- 函数

CLK_ExtLSCkSelect();

- 函数功能

选择外部低速晶振作为晶片低速频率来源，在设置前需要先开启外部低速晶振并等待稳定，寄存器MCKCN2[5]=1。

- 输入参数

无

- 包含头文件

Driver/CLK.h

- 函数返回值

无

- 函数用法

```
/* 设置外部低速晶振作为晶片低速频率来源 */
CLK_GPIO_OpenXTInput();                //开启外部高速晶振引脚
```

```
CLK_ExtLSCKEnable();          //开启外部低速晶振；
CLK_ExtLSCkSelect() ;         //设置外部低速晶振作为晶片低速频率来源
```

3.3.8 CLK_HAOEnable

- 函数

CLK_HAOEnable();

- 函数功能

开启内部高速频率HAO，设置MCKCN1[0]=1。

- 输入参数

无

- 包含头文件

Driver/CLK.h

- 函数返回值

无

- 函数用法

```
/*  使能内部HAO */
CLK_HAOEnable();              //使能内部高速 HAO
```

3.3.9 CLK_HAODisable

- 函数

CLK_HAODisable();

- 函数功能

关闭内部高速HAO，设置MCKCN1[0]=0。

- 输入参数

无

- 包含头文件

Driver/CLK.h

- 函数返回值

无

- 函数用法

```
/*  关闭内部高速晶振HAO */
CLK_HAODisable();             //关闭内部高速频率 HAO
```

3.3.10 CLK_HAOHSCkSel

- 函数

CLK_HAOHSCkSel();

- 函数功能

选择内部HAO作为晶片高速频率来源，在设置前需要先开启HAO并等待稳定，寄存器MCKCN2[4]=0。

- **输入参数**

无

- **包含头文件**

Driver/CLK.h

- **函数返回值**

无

- **函数用法**

```
/* 开启内部高速晶振HAO并设置为晶片高速频率来源 */
CLK_HAOEnable();                    //开启内部高速频率 HAO
CLK_HAOHSCkSel();                   //设置 HAO 作为晶片高速频率来源
```

3.3.11 CLK_LPOHSCkSel

- **函数**

CLK_LPOHSCkSel();

- **函数功能**

选择内部LPO作为晶片高速频率来源，寄存器MCKCN2[5]=0。

- **输入参数**

无

- **包含头文件**

Driver/CLK.h

- **函数返回值**

无

- **函数用法**

```
/* 设置LPO作为晶片低速频率来源 */
CLK_LPOHSCkSel();                   //设置 LPO 作为晶片低速频率来源
```

3.3.12 CLK_CPUCKSel

- **函数**

CLK_CPUCKSel(hssdiv, cpucksel);

- **函数功能**

选择晶片CPU频率来源，并设置高速频率HSS_CK频率分频，寄存器MCKCN2[1:0]及MCKCN2[3:2]。

- **输入参数**

hssdiv [in]：设置高速频率HSS_CK频率分频

MCKCN2_HSS_HSCKDIV8：高速频率HSS进行8分频

MCKCN2_HSS_HSCKDIV4：高速频率HSS进行4分频

MCKCN2_HSS_HSCKDIV2：高速频率HSS进行2分频

MCKCN2_HSS_HSCKDIV1：高速频率HSS进行1分频

cpucksel [in]：选择晶片CPU频率来源

MCKCN2_CPUCK_HSCK：CPU_CK频率来源于HS_CK

MCKCN2_CPUCK_LSCK：CPU_CK频率来源于LS_CK

MCKCN2_CPUCK_HSDCK：CPU_CK频率来源于HSD_CK

MCKCN2_CPUCK_HSSCK：CPU_CK频率来源于HSS_CK

- **包含头文件**

Driver/CLK.h

- **函数返回值**

无

- **函数用法**

```
/* 设置CPU_CK来源HSD_CK，并设置HSS_CK为1分频 */
CLK_CPUCKSel( MCKCN2_HSS_HSCKDIV1, MCKCN2_CPUCK_HSDCK );
```

3.3.13 CLK_PERCKHSDCKSel

- **函数**

CLK_PERCKHSDCKSel();

- **函数功能**

选择HSD_CK作为周边频率PERCK的频率来源，寄存器MCKCN3[3]=1。

- **输入参数**

无

- **包含头文件**

Driver/CLK.h

- **函数返回值**

无

- **函数用法**

```
/* 选择HSD_CK作为周边频率PERCK的频率来源 */
CLK_PERCKHSDCKSel();
```

3.3.14 CLK_PERCKLSCKSel

- **函数**

CLK_PERCKLSCKSel();

- **函数功能**

选择LS_CK作为周边频率PERCK的频率来源，寄存器MCKCN3[3]=0。

- **输入参数**

无

- **包含头文件**

Driver/CLK.h

- **函数返回值**

无

- 函数用法

/* 选择LS_CK作为周边频率PERCK的频率来源 */

CLK_PERCKLSCKSel();

3.3.15 CLK_LPOHSCkSel

- 函数

CLK_LPOHSCkSel();

- 函数功能

选择内部LPO作为晶片高速频率来源，寄存器MCKCN2[5]=0。

- 输入参数

无

- 包含头文件

Driver/CLK.h

- 函数返回值

无

- 函数用法

/* 设置LPO作为晶片低速频率来源 */

CLK_LPOHSCkSel(); //设置 LPO 作为晶片低速频率来源

3.3.16 CLK_LCDCKSel

- 函数

CLK_LCDCKSel(LCDCKSel);

- 函数功能

设置LCD工作频率预分频，设置寄存器MCKCN3[7:5]。

- 输入参数

LCDCKSel [in]：LCD工作频率LCD_CK的预分频设置

MCKCN3_LCDS_PERACKDIV128 ： LCD_CK=PERA_CK/128

MCKCN3_LCDS_PERACKDIV64 ： LCD_CK=PERA_CK/64

MCKCN3_LCDS_PERACKDIV32 ： LCD_CK=PERA_CK/32

MCKCN3_LCDS_PERACKDIV16 ： LCD_CK=PERA_CK/16

MCKCN3_LCDS_PERACKDIV8 ： LCD_CK=PERA_CK/8

MCKCN3_LCDS_PERACKDIV4 ： LCD_CK=PERA_CK/4

MCKCN3_LCDS_PERACKDIV2 ： LCD_CK=PERA_CK/2

MCKCN3_LCDS_PERACKDIV1 ： LCD_CK=PERA_CK/1

- 包含头文件

Driver/CLK.h

- 函数返回值

无

- 函数用法

/* 设置LCD工作频率为PERA_CK/16预分频 */

CLK_LCDCKSel(MCKCN3_LCDS_PERACKDIV16); //设置 LCD_CK= PERA_CK/16

3.3.17 CLK_BZCKSel

- 函数

CLK_BZCKSel(BZCKSel);

- 函数功能

设置蜂鸣器驱动频率预分频，设置寄存器MCKCN3[2:0]。

- 输入参数

BZCKSel [in]：设置蜂鸣器驱动频率预分频

MCKCN3_BZS_PERCKDIV128 ：蜂鸣器驱动频率为PER_CK/128

MCKCN3_BZS_PERCKDIV64 ：蜂鸣器驱动频率为PER_CK/64

MCKCN3_BZS_PERCKDIV32 ：蜂鸣器驱动频率为PER_CK/32

MCKCN3_BZS_PERCKDIV16 ：蜂鸣器驱动频率为PER_CK/16

MCKCN3_BZS_PERCKDIV8 ：蜂鸣器驱动频率为PER_CK/8

MCKCN3_BZS_PERCKDIV4 ：蜂鸣器驱动频率为PER_CK/4

MCKCN3_BZS_PERCKDIV3 ：蜂鸣器驱动频率为PER_CK/3

MCKCN3_BZS_PERCKDIV1 ：蜂鸣器驱动频率为PER_CK/1

- 包含头文件

Driver/CLK.h

- 函数返回值

无

- 函数用法

/* 设置蜂鸣器频率为PER_CK/32 */

CLK_BZCKSel(MCKCN3_BZS_PERCKDIV32);

3.3.18 CLK_ADCCKSel

- 函数

CLK_ADCCKSel(ADCCKSel);

- 函数功能

设置ADC工作频率来源，设置寄存器MCKCN1[4]。

- 输入参数

ADCKCKSel [in]：设置ADC工作频率来源

MCKCN1_ADCCK_LSCK ：ADC工作频率来源于LS_CK

MCKCN1_ADCCK_HSDCK ：ADC工作频率来源于HSD_CK

- 包含头文件

Driver/CLK.h

- **函数返回值**

无

- **函数用法**

/* 设置ADC工作频率来源于HSD_CK */

CLK_ADCCKSel(MCKCN1_ADCCK_HSDCK); //设置 ADC 工作频率来源于 HSD_CK

3.3.19 CLK_ADCCKDivSel

- **函数**

CLK_ADCCKDivSel(ADCDivSel);

- **函数功能**

设置ADC工作频率预分频，设置寄存器MCKCN1[7:5]。

- **输入参数**

ADCDivSel [in]：设置ADC工作频率预分频

MCKCN1_ADCS_DIV128 ：ADC_CK/128

MCKCN1_ADCS_DIV64 ：ADC_CK/64

MCKCN1_ADCS_DIV32 ：ADC_CK/32

MCKCN1_ADCS_DIV16 ：ADC_CK/16

MCKCN1_ADCS_DIV8 ：ADC_CK/8

MCKCN1_ADCS_DIV4 ：ADC_CK/4

MCKCN1_ADCS_DIV2 ：ADC_CK/2

MCKCN1_ADCS_DIV1 ：ADC_CK/1

- **包含头文件**

Driver/CLK.h

- **函数返回值**

无

- **函数用法**

/* 设置ADC工作频率进行32预分频 */

CLK_ADCCKDivSel(MCKCN1_ADCS_DIV32); //设置 ADC_CK/32

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

--定时计数器 C(Timer C)的配置控制及 PWM 模式控制

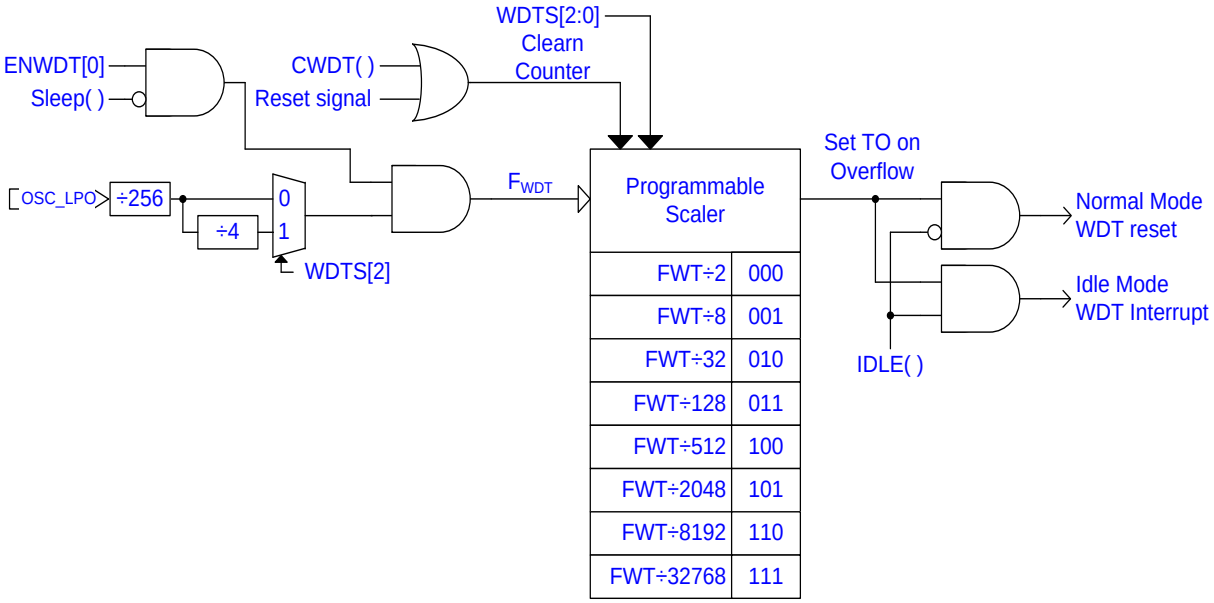
--包含 TMA.h/ TMB.h/ TMC.h/ PWM.h

序号	函数名称	功能描述
01	WDT_Open	使能看门狗(WDT)，设定计数溢出值
02	WDT_INT_Enable	使能看门狗定时计数中断功能
03	WDT_INT_Disable	关闭看门狗定时计数中断功能
04	WDT_INT_IsFlag	读取看门狗中断请求标志位
05	WDT_INT_ClearFlag	清除看门狗中断请求标志位
06	WDT_Enable	启动看门狗功能
07	WDT_OFTimeSel	设置看门狗定时计数溢出时间
08	WDT_Clear	清零看门狗计数值
09	TMA_Open	设置 TMA 频率源及计数溢出值启动 TMA
10	TMA_INT_Enable	使能 TMA 定时中断功能
11	TMA_INT_Disable	关闭 TMA 定时中断功能
12	TMA_INT_IsFlag	读取 TMA 定时中断请求标志位
13	TMA_INT_ClearFlag	清零 TMA 中断请求标志位
14	TMA_Enable	启动 TMA 定时计数功能
15	TMA_Disable	关闭 TMA 定时计数功能
16	TMA_CLKSel	设置TMA的时钟频率源
17	TMA_OFControlSel	设置 TMA 定时计数溢出值
18	TMB_Open	使能TMB，设置TMB频率源和频率分频器，及TMB计数模式
19	TMB_INT_Enable	使能TMB定时中断功能
20	TMB_INT_Disable	关闭TMB定时中断功能
21	TMB_INT_IsFlag	读取TMB中断请求标志位
22	TMB_INT_ClearFlag	清零TMB定时中断标志位
23	TMB_Enable	使能TMB定时计数功能
24	TMB_Disable	关闭TMB定时计数功能
25	TMB_CLKSel	设置 TMB 工作频率源
26	TMB_CLKPrescalerSel	设置 TMB 工作频率源分频器
27	TMB_SynchConfig	TMB工作频率与CPU同步处理设置
28	TMB_TMBROperateMode	设置TMB计数方式
29	TMC_Open	启动TMC定时计数功能，设置TMC的时钟源及时钟源预分频值，设置TMC定时计数溢出值
30	TMC_INT_Enable	使能 TMC 定时计数中断功能
31	TMC_INT_Disable	关闭 TMC 定时计数中断功能
32	TMC_INT_IsFlag	读取 TMC 定时计数中断请求标志位
33	TMC_INT_ClearFlag	清除 TMC 中断请求标志位

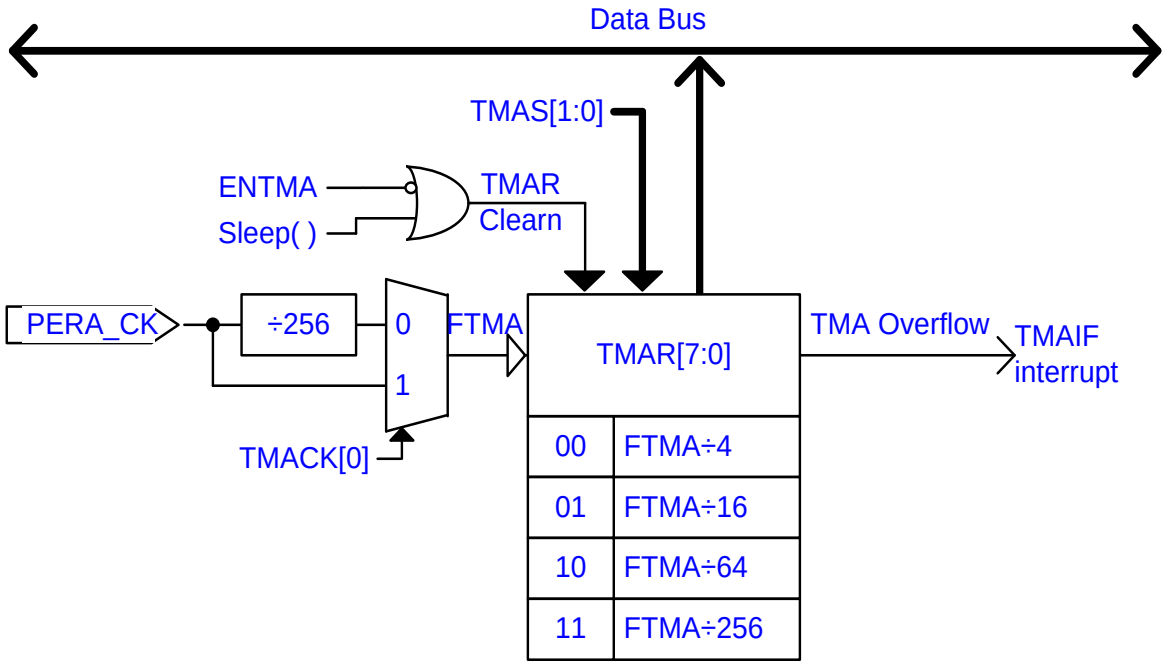
34	TMC_Enable	启动 TMC 定时计数功能
35	TMC_Disable	关闭TMC定时计数功能
36	TMC_CLKSel	设置TMC工作频率源
37	TMC_OFControlSel	设置TMC计数溢出值
38	TMC_CLKPrescalerSel	TMC工作频率预分频设置
39	PWM_Open	使能PWM功能 ;设置PWM输出模式及对应引脚复用功能，输出引脚有效准位，PWM周期及占空比，死区延迟时间
40	PWM_Enable	使能PWM功能
41	PWM_Disable	关闭PWM功能
42	PWM_ShutoffEnable	使能PWM自动关闭功能
43	PWM_ShutoffDisable	关闭PWM自动关闭功能
44	PWM_ShutoffConfig	设置PWM自动关闭功能触发事件
45	PWM_ShutoffDefine0	设置PWM自动关闭时PWM0/PWM2 输出引脚状态定义
46	PWM_ShutoffDefine1	设置PWM自动关闭时PWM1/PWM3 输出引脚状态定义
47	PWM_OutMode	设置PWM输出模式
48	PWM_OutStateSelect	设置PWM输出引脚有效准位
49	PWM_StartConfig	设置PWM自动开启控制条件
50	PWM_DBDTime	设置PWM死区延迟时间
51	PWM_PWMRL	写入PWM周期控制值PWMR[9:0]低 2bit
52	PWM_PWMRH	写入PWM周期控制值PWMR[9:0]高 8bit
53	PFD_Enable	使能PFD功能
54	PFD_Disable	关闭PFD功能
55	PFD_Open	使能PFD功能并设置PFD输出波形周期

4.2 功能方框图

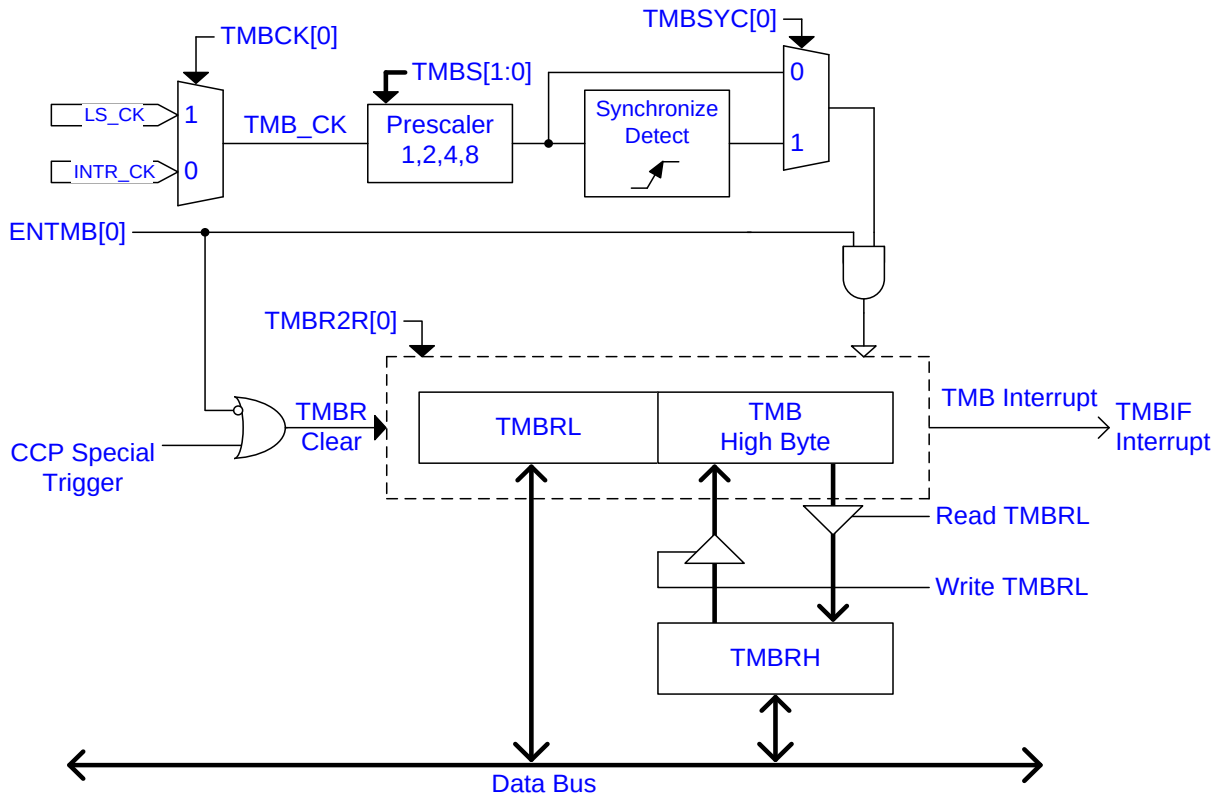
4.2.1 看门狗(WDT) 模块方框图



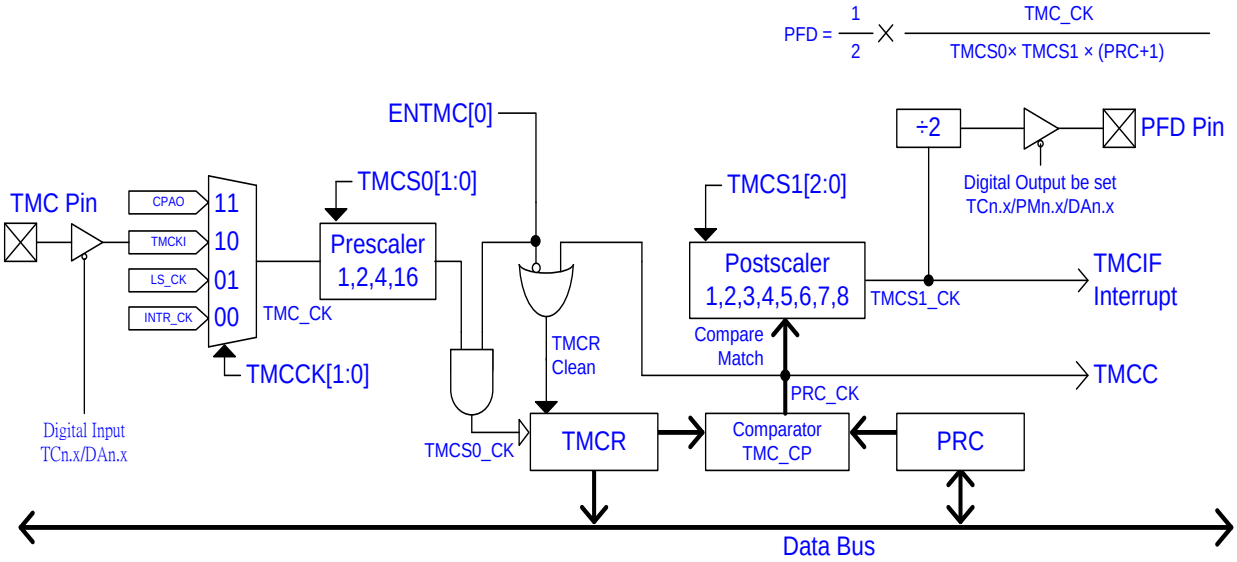
4.2.2 定时计数器 A(Timer A) 模块方框图



4.2.3 定时计数器 B(Timer B) 模块方框图



4.2.4 定时计数器 C(Timer C) 模块方框图



Note: PWM just work at TMCCK=00 mode (clock source = INTR_CK)

4.3 函数说明

4.3.1 WDT_Open

- **函数**

  void WDT_Open(unsigned char wdts);

- **函数功能**

  使能看门狗(WDT)，设定计数溢出值，设置寄存器TMACN[3:0]。

- **输入参数**

  wdts [in]　看门狗时钟源分频设置

  TMACN_WDTS_32768：Fwdt / 32768

  TMACN_WDTS_8192 ： Fwdt / 8192

  TMACN_WDTS_2048 ： Fwdt / 2048

  TMACN_WDTS_512 ： Fwdt / 512

  TMACN_WDTS_128 ： Fwdt / 128

  TMACN_WDTS_32 ： Fwdt / 32

  TMACN_WDTS_8 ： Fwdt / 8

  TMACN_WDTS_2 ： Fwdt / 2

- **包含头文件**

  Driver/WDT.h

- **函数返回值**

无

- 函数用法

/* 设置看门狗溢出时间Fwdt/32 */

WDT_Open (TMACN_WDTS_32);

4.3.2 WDT_INT_Enable

- 函数

WDT_INT_Enable();

- 函数功能

使能看门狗定时计数中断功能，设置寄存器INTE1[2]=1 。

- 输入参数

无

- 包含头文件

Driver/WDT.h

- 函数返回值

无

- 函数用法

/* 使能看门狗定时计数中断 */

unsigned int data；

data = DrvWDT_CounterRead();

4.3.3 WDT_INT_Disable

- 函数

WDT_INT_Disable();

- 函数功能

关闭看门狗定时计数中断功能，设置寄存器INTE1[2]=0 。

- 输入参数

无

- 包含头文件

Driver/WDT.h

- 函数返回值

无

- 函数用法

/* 关闭看门狗定时中断功能 */

WDT_INT_Disable();

4.3.4 WDT_INT_IsFlag

- 函数

WDT_INT_IsFlag();

- 函数功能

读取看门狗中断请求标志位，读取寄存器INTF1[2]。

- 输入参数

无

- 包含头文件

Driver/WDT.h

- 函数返回值

0x00：看门狗没产生中断请求标志位

0x04：看门狗产生中断请求标志位

- 函数用法

```
/* 读取看门中断请求标志位 */
unsigned char flag;
Flag = WDT_INT_IsFlag();
```

4.3.5 WDT_INT_ClearFlag

- 函数

WDT_INT_ClearFlag()

- 函数功能

清除看门狗中断请求标志位，设置寄存器INTF1[2]=0 。

- 输入参数

无

- 包含头文件

Driver/WDT.h

- 函数返回值

无

- 函数用法

```
/* 清除看门狗中断请求标志位 */
WDT_INT_ClearFlag();
```

4.3.6 WDT_Enable

- 函数

WDT_Enable();

- 函数功能

启动看门狗功能，设置寄存器TMACN[3]=1 。

- 输入参数

无

- 包含头文件

  Driver/WDT.h

- 函数返回值

  无

- 函数用法

```
/* 启动看门狗定时计数功能 */
WDT_Enable();
```

4.3.7 WDT_OFTimeSel

- 函数

  WDT_OFTimeSel(WDTSel);

- 函数功能

  设置看门狗定时计数溢出时间，设置寄存器TMACN[2:0]。

- 输入参数

  WDTSel [in]　看门狗时钟源分频设置

  TMACN_WDTS_32768：Fwdt / 32768

  TMACN_WDTS_8192 ： Fwdt / 8192

  TMACN_WDTS_2048 ： Fwdt / 2048

  TMACN_WDTS_512　： Fwdt / 512

  TMACN_WDTS_128　： Fwdt / 128

  TMACN_WDTS_32　　： Fwdt / 32

  TMACN_WDTS_8　　： Fwdt / 8

  TMACN_WDTS_2　　： Fwdt / 2

- 包含头文件

  Driver/WDT.h

- 函数返回值

  无

- 函数用法

```
/*  设置看门狗定时溢出为Fwdt/32 */
WDT_OFTimeSel( TMACN_WDTS_32 );
```

4.3.8 WDT_Clear

- 函数

  WDT_Clear();

- 函数功能

  清零看门狗计数值。

- 输入参数

  无

- 包含头文件

  Driver/WDT.h

- 函数返回值

  无

- 函数用法

  /* 清零看门狗定时计数值 */

  WDT_Clear();

4.3.9 TMA_Open

- 函数

  void TMA_Open(unsigned char ck, unsigned char cks);

- 函数功能

  设置TMA时钟频率源及定时计数溢出值，并启动TMA定时功能，设置寄存器TMACN[7:4]。

- 输入参数

  ck [in]：TMA时钟频率源选择

  TMACN_TMACK_DIV1 ：F_{TMA} = PERA_CK

  TMACN_TMACK_DIV256：F_{TMA} = PERA_CK/256

  cks [in]：TMA定时计数溢出值设置

  TMACN_TMAS_DIV256 ：F_{TMA} / 256，每次溢出发生中断，TMAR[7:0]=TMAR[7:0]+256

  TMACN_TMAS_DIV64 ：F_{TMA} / 64，每次溢出发生中断，TMAR[7:0]=TMAR[7:0]+64

  TMACN_TMAS_DIV16 ：F_{TMA} / 16，每次溢出发生中断，TMAR[7:0]=TMAR[7:0]+16

  TMACN_TMAS_DIV4 ：F_{TMA} / 4，每次溢出发生中断，TMAR[7:0]=TMAR[7:0]+4

- 包含头文件

  Driver/TMA.h

- 函数返回值

  无

- 函数用法

  /* 设置TMA时钟源为PERA_CK，及计数溢出值为F_{TMA} / 16 */

  TMA_Open(TMACN_TMACK_DIV1, TMACN_TMAS_DIV16);

4.3.10 TMA_INT_Enable

- 函数

  TMA_INT_Enable();

- 函数功能

  使能TMA定时中断功能，设置寄存器INTE1[3]=1 。

- 输入参数

  无

- 包含头文件

Driver/TMA.h

- 函数返回值

无

- 函数用法

/* 使能TMA定时中断功能 */

TMA_INT_Enable();

4.3.11 TMA_INT_Disable

- 函数

TMA_INT_Disable();

- 函数功能

关闭TMA定时中断功能，设置寄存器INTE1[3]=0 。

- 输入参数

无

- 包含头文件

Driver/TMA.h

- 函数返回值

无

- 函数用法

/* 关闭TMA定时中断功能 */

TMA_INT_Disable();

4.3.12 TMA_INT_IsFlag

- 函数

TMA_INT_IsFlag();

- 函数功能

读取TMA定时中断请求标志位，读取寄存器INTF1[3]。

- 输入参数

无

- 包含头文件

Driver/TMA.h

- 函数返回值

返回TMA中断请求标志位：

0x00：TMA没有中断请求

0x08：TMA产生中断请求

- 函数用法

/* 读取TMA定时中断请求标志位 */

unsigned char flag ;

```
flag = TMA_INT_IsFlag();
```

4.3.13 TMA_INT_ClearFlag

- 函数

TMA_INT_ClearFlag();

- 函数功能

清零TMA中断请求标志位，设置寄存器INTF1[3]=0 。

- 输入参数

无

- 包含头文件

Driver/TMA.h

- 函数返回值

无

- 函数用法

```
/* 清零TMA中断请求标志位 */
TMA_INT_ClearFlag();
```

4.3.14 TMA_Enable

- 函数

TMA_Enable()

- 函数功能

启动TMA定时计数功能，设置寄存器TMACN[7]=1 。

- 输入参数

无

- 包含头文件

Driver/TMA.h

- 函数返回值

无

- 函数用法

```
/*  启动TMA定时计数功能  */
TMA_Enable();
```

4.3.15 TMA_Disable

- 函数

TMA_Disable();

- 函数功能

关闭TMA定时计数功能，设置寄存器TMACN[7]=0 。

- **输入参数**

  无

- **包含头文件**

  Driver/TMA.h

- **函数返回值**

  无

- **函数用法**

```
/* 关闭TMA定时计数功能 */
TMA_Disable();
```

4.3.16 TMA_CLKSel

- **函数**

  TMA_CLKSel(CLKSel);

- **函数功能**

  设置TMA的时钟频率源，设置寄存器TMACN[6]。

- **输入参数**

  CLKSel [in]：设置TMA时钟频率源

  TMACN_TMACK_DIV1 ：F_{TMA} = PERA_CK

  TMACN_TMACK_DIV256：F_{TMA} = PERA_CK/256

- **包含头文件**

  Driver/TMA.h

- **函数返回值**

  无

- **函数用法**

```
/* 设置TMA时钟频率源为PERA_CK */
TMA_CLKSel( TMACN_TMACK_DIV1 );
```

4.3.17 TMA_OFControlSel

- **函数**

  TMA_OFControlSel(TMASel);

- **函数功能**

  设置TMA定时计数溢出值，设置寄存器TMACN[5:4]。

- **输入参数**

  TMASel [in]：设置TMA定时计数溢出值

  TMACN_TMAS_DIV256 ：F_{TMA} / 256，每次溢出发生中断，TMAR[7:0]=TMAR[7:0]+256

  TMACN_TMAS_DIV64 ：F_{TMA} / 64，每次溢出发生中断，TMAR[7:0]=TMAR[7:0]+64

  TMACN_TMAS_DIV16 ：F_{TMA} / 16，每次溢出发生中断，TMAR[7:0]=TMAR[7:0]+16

  TMACN_TMAS_DIV4 ：F_{TMA} / 4，每次溢出发生中断，TMAR[7:0]=TMAR[7:0]+4

- 包含头文件

Driver/TMA.h

- 函数返回值

无

- 函数用法

/* 设定TMA定时计数溢出值为F_{TMA} / 16 */

TMA_OFControlSel(TMACN_TMAS_DIV16);

4.3.18 TMB_Open

- 函数

void TMB_Open(unsigned char ck, unsigned char cks, unsigned char mode);

- 函数功能

使能TMB，设置TMB时钟频率源和时钟频率分频器，及TMB计数模式，设置寄存器TMBCN[7:4]/ TMBCN[2]。

- 输入参数

ck [in]：设置TMB工作频率源

TMBCN_TMBCK_LSCK ：TMB工作频率为LS_CK

TMBCN_TMBCK_INTRCK ：TMB工作频率为INTR_CK

cks [in]：设置TMB工作频率分频器

TMBCN_TMBS_DIV8：TMB_CK /8

TMBCN_TMBS_DIV4：TMB_CK /4

TMBCN_TMBS_DIV2：TMB_CK /2

TMBCN_TMBS_DIV1：TMB_CK /1

mode [in]：设置TMB计数模式

TMBCN_TMBR2R_16BIT：TMB作为16bit定时计数器使用

TMBCN_TMBR2R_8BIT ：TMB作为8bit定时计数器使用

- 包含头文件

Driver/TMB.h

- 函数返回值

无

- 函数用法

/* 使能TMB，设置TMB时钟源为INTR_CK，时钟源进行2分频，作为16bit计数器 */

TMB_Open(TMBCN_TMBCK_INTRCK, TMBCN_TMBS_DIV2, TMBCN_TMBR2R_16BIT);

4.3.19 TMB_INT_Enable

- 函数

TMB_INT_Enable();

- 函数功能

使能TMB定时中断功能，设置寄存器INTE1[4]=1 。

- **输入参数**

  无

- **包含头文件**

  Driver/TMB.h

- **函数返回值**

  无

- **函数用法**

  /* 使能TMB定时中断功能 */

  TMB_INT_Enable();

4.3.20 TMB_INT_Disable

- **函数**

  TMB_INT_Disable();

- **函数功能**

  关闭TMB定时中断功能，设置寄存器INTE1[4]=0 。

- **输入参数**

  无

- **包含头文件**

  Driver/TMB.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭TMB定时中断功能 */

  TMB_INT_Disable();

4.3.21 TMB_INT_IsFlag

- **函数**

  TMB_INT_IsFlag() ;

- **函数功能**

  读取TMB中断请求标志位，读取寄存器INTF1[4]。

- **输入参数**

  无

- **包含头文件**

  Driver/TMB.h

- **函数返回值**

  返回TMB定时中断请求标志位值

  0x00：TMB没有产生中断请求

  0x10：TMB产生中断请求

- 函数用法

/* 读取TMB中断请求标志位 */

unsigned char flag ;

flag = TMB_INT_IsFlag();

4.3.22 TMB_INT_ClearFlag

- 函数

TMB_INT_ClearFlag() ;

- 函数功能

清零TMB定时中断标志位，设置寄存器INTF1[4]=0 。

- 输入参数

无

- 包含头文件

Driver/TMB.h

- 函数返回值

无

- 函数用法

/*清零TMB定时中断标志位*/

TMB_INT_ClearFlag();

4.3.23 TMB_Enable

- 函数

TMB_Enable() ;

- 函数功能

使能TMB定时计数功能，设置寄存器TMBCN[7]=1 。

- 输入参数

无

- 包含头文件

Driver/TMB.h

- 函数返回值

无

- 函数用法

/* 使能TMB定时计数功能 */

TMB_Enable();

4.3.24 TMB_Disable

- 函数

TMB_Disable();

- 函数功能

关闭TMB定时计数功能，设置寄存器TMBCN[7]=0 。

- 输入参数

无

- 包含头文件

Driver/TMB.h

- 函数返回值

无

- 函数用法

/* 关闭TMB定时计数功能 */

TMB_Disable();

4.3.25 TMB_CLKSel

- 函数

TMB_CLKSel(CLKSel);

- 函数功能

设置TMB工作频率源，设置寄存器TMBCN[6]。

- 输入参数

CLKSel [in]：设置TMB工作频率源

TMBCN_TMBCK_LSCK ：TMB工作频率为LS_CK

TMBCN_TMBCK_INTRCK ：TMB工作频率为INTR_CK

- 包含头文件

Driver/TMB.h

- 函数返回值

无

- 函数用法

/* 设置TMB工作频率源为INTR_CK */

TMB_CLKSel(TMBCN_TMBCK_INTRCK);

4.3.26 TMB_CLKPrescalerSel

- 函数

TMB_CLKPrescalerSel(CLKPreSel);

- 函数功能

设置TMB工作频率源分频器，设置寄存器TMBCN[5:4]。

- 输入参数

CLKPreSel [in]：设置TMB工作频率分频器

TMBCN_TMBS_DIV8：TMB_CK /8

TMBCN_TMBS_DIV4：TMB_CK /4

TMBCN_TMBS_DIV2：TMB_CK /2

TMBCN_TMBS_DIV1：TMB_CK /1

- 包含头文件

Driver/TMB.h

- 函数返回值

无

- 函数用法

```
/* 设置TMB工作频率源为INTR_CK，并进行2分频 */
TMB_CLKSel( TMBCN_TMBCK_INTRCK );
TMB_CLKPrescalerSel( TMBCN_TMBS_DIV2 );
```

4.3.27 TMB_SynchConfig

- 函数

TMB_SynchConfig(SynchCon);

- 函数功能

TMB工作频率与CPU同步处理设置，若TMB的工作频率与CPU工作频率不一致时，需要进行同步处理，设置寄存器TMBCN[3]。

- 输入参数

TMBCN_TMBSYC_SYNCH ：与CPU同步处理

TMBCN_TMBSYC_ASYNCH ：与CPU不同步处理

- 包含头文件

Driver/TMB.h

- 函数返回值

无

- 函数用法

```
/* 设置TMB工作频率为LS_CK，同时设置同步处理 */
TMB_CLKSel( TMBCN_TMBCK_LSCK );
TMB_SynchConfig( TMBCN_TMBSYC_SYNCH );
```

4.3.28 TMB_TMBROperateMode

- 函数

TMB_TMBROperateMode(TMBRSel) ;

- 函数功能

设置TMB计数方式，设置寄存器TMBCN[2]。

- 输入参数

TMBRSel [in]：设置TMB计数模式

TMBCN_TMBR2R_16BIT：TMB作为16bit定时计数器使用

TMBCN_TMBR2R_8BIT ： TMB作为8bit定时计数器使用

- 包含头文件

Driver/TMB.h

- 函数返回值

无

- 函数用法

```
/* 设置TMB计数器为16bit */
TMB_TMBROperateMode( TMBCN_TMBR2R_16BIT );
```

4.3.29 TMC_Open

- 函数

void TMC_Open(unsigned char tmcck, unsigned char tmcs0, unsigned char tmcs1) ;

- 函数功能

启动TMC定时计数功能，设置TMC的时钟源及时钟源预分频值，设置TMC定时计数溢出值，设置寄存器TMCCN[7:0] 。

- 输入参数

tmcck [in]：TMC工作频率选择

TMCCN_TMCCK_CPAO ： TMC 工作频率为 CPAO
TMCCN_TMCCK_TMCKI ： TMC 工作频率为 TMCKI
TMCCN_TMCCK_LSCK ： TMC 工作频率为 LS_CK
TMCCN_TMCCK_INTRCK ： TMC 工作频率为 INTR_CK

tmcs0 [in]：TMC 工作频率预分频设置

TMCCN_TMCS0_DIV16 ： TMC_CK/16
TMCCN_TMCS0_DIV4 ： TMC_CK/4
TMCCN_TMCS0_DIV2 ： TMC_CK/2
TMCCN_TMCS0_DIV1 ： TMC_CK/1

tmcs1 [in]：TMC 定时计数溢出值设置

TMCCN_TMCS1_DIV8 ： PRC_CK/8
TMCCN_TMCS1_DIV7 ： PRC_CK/7
TMCCN_TMCS1_DIV6 ： PRC_CK/6
TMCCN_TMCS1_DIV5 ： PRC_CK/5
TMCCN_TMCS1_DIV4 ： PRC_CK/4
TMCCN_TMCS1_DIV3 ： PRC_CK/3
TMCCN_TMCS1_DIV2 ： PRC_CK/2
TMCCN_TMCS1_DIV1 ： PRC_CK/1

- 包含头文件

Driver/TMC.h

- 函数返回值

无

- 函数用法

/* 设置TMC工作频率为LS_CK/2,溢出控制值为PRC_CK/4 */

TMC_Open(TMCCN_TMCCK_LSCK, TMCCN_TMCS0_DIV2, TMCCN_TMCS1_DIV4);

4.3.30 TMC_INT_Enable

- 函数

TMC_INT_Enable();

- 函数功能

使能TMC定时计数中断功能，设置寄存器INTE1[5]=1 。

- 输入参数

无

- 包含头文件

Driver/TMC.h

- 函数返回值

无

- 函数用法

/* 使能TMC定时计数中断功能 */

TMC_INT_Enable();

4.3.31 TMC_INT_Disable

- 函数

TMC_INT_Disable();

- 函数功能

关闭TMC定时计数中断功能，设置寄存器INTE1[5]=0 。

- 输入参数

无

- 包含头文件

Driver/TMC.h

- 函数返回值

无

- 函数用法

/* 关闭TMC定时中断功能 */

TMC_INT_Disable();

4.3.32 TMC_INT_IsFlag

- 函数

TMC_INT_IsFlag();

- 函数功能

读取TMC定时计数中断请求标志位，读取寄存器INTF1[5]。

- 输入参数

无

- 包含头文件

Driver/TMC.h

- 函数返回值

0x00：TMC没中断请求

0x20：TMC产生中断请求

- 函数用法

```
/* 读取TMC中断请求标志位 */
unsigned char flag ;
flag = TMC_INT_IsFlag();
```

4.3.33 TMC_INT_ClearFlag

- 函数

TMC_INT_ClearFlag();

- 函数功能

清除TMC中断请求标志位，设置寄存器INTF1[5]=0 。

- 输入参数

无

- 包含头文件

Driver/TMC.h

- 函数返回值

无

- 函数用法

```
/* 清除TMC中断请求标志位 */
TMC_INT_ClearFlag();
```

4.3.34 TMC_Enable

- 函数

TMC_Enable();

- 函数功能

启动TMC定时计数功能，设置寄存器TMCCN[7]=1 。

- 输入参数

无

- 包含头文件

Driver/TMC.h

- 函数返回值

无

- 函数用法

```
/* 启动TMC定时计数功能 */
TMC_Enable();
```

4.3.35 TMC_Disable

- 函数

TMC_Disable();

- 函数功能

关闭TMC定时计数功能，设置TMCCN[7]=0 。

- 输入参数

无

- 包含头文件

Driver/TMC.h

- 函数返回值

无

- 函数用法

```
/* 关闭TMC定时计数功能 */
TMC_Disable();
```

4.3.36 TMC_CLKSel

- 函数

TMC_CLKSel(CLKSel);

- 函数功能

设置TMC工作频率源，设置寄存器TMCCN[6:5]。

- 输入参数

CLKSel [in]：

TMCCN_TMCCK_CPAO ： TMC 工作频率为 CPAO

TMCCN_TMCCK_TMCKI ： TMC 工作频率为 TMCKI

TMCCN_TMCCK_LSCK ： TMC 工作频率为 LS_CK

TMCCN_TMCCK_INTRCK ： TMC 工作频率为 INTR_CK

- 包含头文件

Driver/TMC.h

- 函数返回值

无

- 函数用法

```
/* 设置TMC工作频率源为INTR_CK */
TMC_CLKSel( TMCCN_TMCCK_INTRCK );
```

4.3.37 TMC_OFControlSel

- 函数

TMC_OFControlSel(TMCSel1);

- 函数功能

设置TMC计数溢出值，设置寄存器TMCCN[4:2]。

- 输入参数

TMCSel1 [in]：TMC计数溢出值设置

TMCCN_TMCS1_DIV8 ： PRC_CK/8
TMCCN_TMCS1_DIV7 ： PRC_CK/7
TMCCN_TMCS1_DIV6 ： PRC_CK/6
TMCCN_TMCS1_DIV5 ： PRC_CK/5
TMCCN_TMCS1_DIV4 ： PRC_CK/4
TMCCN_TMCS1_DIV3 ： PRC_CK/3
TMCCN_TMCS1_DIV2 ： PRC_CK/2
TMCCN_TMCS1_DIV1 ： PRC_CK/1

- 包含头文件

Driver/TMC.h

- 函数返回值

无

- 函数用法

```
/* 设置TMC计数溢出为PRC_CK/5 */
TMC_OFControlSel( TMCCN_TMCS1_DIV5 );
```

4.3.38 TMC_CLKPrescalerSel

- 函数

TMC_CLKPrescalerSel(TMCSel0) ;

- 函数功能

TMC工作频率预分频设置，设置寄存器TMCCN[1 :0]。

- 输入参数

TMCSel0 [in]：TMC工作频率预分频设置

TMCCN_TMCS0_DIV16 ： TMC_CK/16
TMCCN_TMCS0_DIV4 ： TMC_CK/4
TMCCN_TMCS0_DIV2 ： TMC_CK/2
TMCCN_TMCS0_DIV1 ： TMC_CK/1

- 包含头文件

Driver/TMC.h

- 函数返回值

无

- 函数用法

/* 设置TMC工作预分频为TMC_CK/4 */

TMC_CLKPrescalerSel(TMCCN_TMCS0_DIV4);

4.3.39 PWM_Open

- 函数

void PWM_Open(unsigned char mode,
unsigned char state,
unsigned char prc,
unsigned int pwmr,
unsigned char dbd)

- 函数功能

使能PWM功能，设置PWM输出模式及对应引脚复用功能，设置PWM输出引脚有效准位，设置PWM周期及占空比，设置PWM死区延迟时间，设置寄存器TMCCN/PRC/PWMR/PWMCN/PDBD/PT2M1/PT2M2/TRISC2

- 输入参数

mode [in]：设置PWM输出模式

PWMCN_PWMCG_REVERSE：全桥反向输出，PWM1调变输出，PWM2有效准位输出

PWMCN_PWMCG_HALF ：半桥输出，PWM0/PWM1死区延迟调变输出，PWM2/PWM3作为IO引脚

PWMCN_PWMCG_FULL ：全桥正向输出，PWM3调变输出，PWM0有效准位输出

PWMCN_PWMCG_SINGLE ：单输出，PWM0调变输出，PWM1/PWM2/PWM3作为IO引脚

state [in]：设置PWM输出引脚有效准位

PWMCN_PWMM_M3：PWM0/PWM2有效准位为Low，PWM1/PWM3有效准位为Low

PWMCN_PWMM_M2：PWM0/PWM2有效准位为Low，PWM1/PWM3有效准位为High

PWMCN_PWMM_M1：PWM0/PWM2有效准位为High，PWM1/PWM3有效准位为Low

PWMCN_PWMM_M0：PWM0/PWM2有效准位为High，PWM1/PWM3有效准位为High

prc [in]：设置PWM周期，8bit的设置值，设置范围是0x0~0xFF

pwmr [in]：设置PWM占空比，10bit的设置值，设置范围是0x0~0x3FF

dbd [in]：设置PWM死区延迟时间，7bit的设置值，设置范围是0x0~0x7F

- 包含头文件

Driver/PWM.h

- 函数返回值

无

- 函数用法

/* 设置PWM为单端输出模式，PWM0/PWM1/PWM2/PWM3有效准位Low，周围为0xFF，占空比为0x7F，死区延时为0x0A */

```
void PWM_Open( PWMCN_PWMCG_SINGLE, PWMCN_PWMM_M3, 0xFF, 0x7F, 0x0A ) ;
```

4.3.40 PWM_Enable

- **函数**

  PWM_Enable();

- **函数功能**

  使能PWM功能，设置寄存器PWMCN[7]=1 。

- **输入参数**

  无

- **包含头文件**

  Driver/PWM.h

- **函数返回值**

  无

- **函数用法**

```
/* 使能PWM功能 */
PWM_Enable();
```

4.3.41 PWM_Disable

- **函数**

  PWM_Disable();

- **函数功能**

  关闭PWM功能，设置寄存器PWMCN[7]=0 。

- **输入参数**

  无

- **包含头文件**

  Driver/PWM.h

- **函数返回值**

  无

- **函数用法**

```
/* 关闭PWM功能 */
PWM_Disable();
```

4.3.42 PWM_ShutoffEnable

- **函数**

  PWM_ShutoffEnable();

- **函数功能**

  使能PWM自动关闭功能，设置寄存器PASC[7]=1 。

- **输入参数**

  无

- **包含头文件**

  Driver/PWM.h

- **函数返回值**

  无

- **函数用法**

  /* 使能PWM自动关闭功能 */

  PWM_ShutoffEnable();

4.3.43 PWM_ShutoffDisable

- **函数**

  PWM_ShutoffDisable();

- **函数功能**

  关闭PWM自动关闭功能，设置寄存器PASC[7]=0 。

- **输入参数**

  无

- **包含头文件**

  Driver/PWM.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭PWM自动关闭功能 */

  PWM_ShutoffDisable();

4.3.44 PWM_ShutoffConfig

- **函数**

  PWM_ShutoffConfig(SCFCon);

- **函数功能**

  设置PWM自动关闭功能触发事件，设置寄存器PASC[5:4] 。

- **输入参数**

  SCFCon [in]：设置PWM自动关闭功能触发事件

  PASC_PASCF_CPAO ：触发信号为CPAO

  PASC_PASCF_MIX ：触发信号为CPAO或FIL0

  PASC_PASCF_FIL0 ：触发信号为FIL0

  PASC_PASCF_CLOSE ：关闭自动关闭事件

- **包含头文件**

Driver/PWM.h

- 函数返回值

无

- 函数用法

/* 设置PWM自动关闭功能触发事件为CPAO */

PWM_ShutoffConfig(PASC_PASCF_CPAO);

4.3.45 PWM_ShutoffDefine0

- 函数

PWM_ShutoffDefine0(SCN0Sel);

- 函数功能

设置PWM自动关闭时PWM0/PWM2输出引脚状态定义，设置寄存器PASC[3:2]。

- 输入参数

SCN0Sel [in]：设置PWM自动关闭时PWM0/PWM2输出引脚状态定义

PASC_PSSCN0_OTHER ：自动关闭时，PWM0 & PWM2输出引脚的为三态状态

PASC_PSSCN0_HIGH ：自动关闭时，PWM0 & PWM2输出引脚的为高电位

PASC_PSSCN0_LOW ：自动关闭时，PWM0 & PWM2输出引脚的为低电位

- 包含头文件

Driver/PWM.h

- 函数返回值

无

- 函数用法

/* 设置自动关闭时，PWM0 & PWM2输出引脚的为高电位 */

PWM_ShutoffDefine0(PASC_PSSCN0_HIGH);

4.3.46 PWM_ShutoffDefine1

- 函数

PWM_ShutoffDefine1(SCN1Sel);

- 函数功能

设置PWM自动关闭时PWM1/PWM3输出引脚状态定义，设置寄存器PASC[1 :0]。

- 输入参数

SCN1Sel [in]：设置PWM自动关闭时PWM1/PWM3输出引脚状态定义

PASC_PSSCN1_OTHER ：自动关闭时，PWM1 & PWM3输出引脚的为三态状态

PASC_PSSCN1_HIGH ：自动关闭时，PWM1 & PWM3输出引脚的为高电位

PASC_PSSCN1_LOW ：自动关闭时，PWM1 & PWM3输出引脚的为低电位

- 包含头文件

Driver/PWM.h

- 函数返回值

无

- 函数用法

/* 设置自动关闭时，PWM1 & PWM3输出引脚的为高电位 */

PWM_ShutoffDefine1(PASC_PSSCN1_HIGH);

4.3.47 PWM_OutMode

- 函数

PWM_OutMode(CGSel);

- 函数功能

设置PWM输出模式，设置寄存器PWMCN[3:2]。

- 输入参数

CGSel [in]：设置PWM输出模式

PWMCN_PWMCG_REVERSE：全桥反向输出，PWM1调变输出，PWM2有效准位输出

PWMCN_PWMCG_HALF ：半桥输出，PWM0/PWM1死区延迟调变输出，PWM2/PWM3作为IO引脚

PWMCN_PWMCG_FULL ：全桥正向输出，PWM3调变输出，PWM0有效准位输出

PWMCN_PWMCG_SINGLE ：单输出，PWM0调变输出，PWM1/PWM2/PWM3作为IO引脚

- 包含头文件

Driver/PWM.h

- 函数返回值

无

- 函数用法

/* 设置PWM为单端输出 */

PWM_OutMode(PWMCN_PWMCG_SINGLE);

4.3.48 PWM_OutStateSelect

- 函数

PWM_OutStateSelect(MSel);

- 函数功能

设置PWM输出引脚有效准位，设置寄存器PWMCN[1 :0]。

- 输入参数

MSel [in]：设置PWM输出引脚有效准位

PWMCN_PWMM_M3：PWM0/PWM2有效准位为Low，PWM1/PWM3有效准位为Low

PWMCN_PWMM_M2：PWM0/PWM2有效准位为Low，PWM1/PWM3有效准位为High

PWMCN_PWMM_M1：PWM0/PWM2有效准位为High，PWM1/PWM3有效准位为Low

PWMCN_PWMM_M0：PWM0/PWM2有效准位为High，PWM1/PWM3有效准位为High

- 包含头文件

Driver/PWM.h

- 函数返回值

无

- 函数用法

/* 设置PWM0/PWM2有效准位为High，PWM1/PWM3有效准位为Low */

PWM_OutStateSelect(PWMCN_PWMM_M1);

4.3.49 PWM_OutStateSelect

- 函数

PWM_StartConfig(PRSCon);

- 函数功能

设置PWM自动开启控制条件，设置寄存器PDBD[7]。

- 输入参数

PRSCon [in]：设置PWM自动开启控制条件

PDBD_ENPRS_AUTO ：硬件自动将PASF置0，PWMx在下一个周期自动开启

PDBD_ENPRS_USER ：用户程序上将PASF置0，PWMx在下一个周期自动开启

- 包含头文件

Driver/PWM.h

- 函数返回值

无

- 函数用法

/* 设置PWM由硬件控制自动开启 */

PWM_StartConfig(PDBD_ENPRS_AUTO);

4.3.50 PWM_OutStateSelect

- 函数

PWM_DBDTime(DBDCSel) ;

- 函数功能

设置PWM死区延迟时间，设置寄存器PDBD[6:0]。

- 输入参数

DBDCSel [in]：设置PWM死区延迟时间，7bit的设置值，设置范围是0x0~0x7F

- 包含头文件

Driver/PWM.h

- 函数返回值

无

- 函数用法

/* 设置PWM死区延迟时间为0XA */

PWM_DBDTime(0x0A);

4.3.51 PWM_PWMRL

- 函数

PWM_PWMRL(RLSel);

- 函数功能

写入PWM周期控制值PWMR[9:0]的低2位PWMRL[1:0]，PWMR = PWMRH[9:2] + PWMRL[1:0]，设置寄存器PWMCN[5:4]。

- 输入参数

RLSel [in]：设置PWM周期控制值PWMR[9:0]的低2位PWMRL[1:0]，设置范围是0x0~0x03

- 包含头文件

Driver/PWM.h

- 函数返回值

无

- 函数用法

/* 设置PWMR[9:0]为0xAA，则低2位PWMRL[1:0]=0x02 */

PWM_PWMRL(0x02);

4.3.52 PWM_PWMRH

- 函数

PWM_PWMRH(RHSel) ;

- 函数功能

写入PWM周期控制值PWMR[9:0]的高8位PWMRH[9:2]，PWMR = PWMRH[9:2] + PWMRL[1:0]，设置寄存器PWMR[7:0] 。

- 输入参数

RHSel [in]：设置PWM周期控制值PWMR[9:0]的高8位PWMRH[9:2]，设置范围是0x0~0xFF

- 包含头文件

Driver/PWM.h

- 函数返回值

无

- 函数用法

/* 设置PWMR[9:0]为0xAA，则PWMRH[9:2]=0x2A */

PWM_PWMRL(0x2A);

4.3.53 PFD_Enable

- 函数

PFD_Enable() ;

- 函数功能

使能PFD功能 ， 设置寄存器PWMCN[6]=1 。

- 输入参数

无

- 包含头文件

  Driver/PWM.h

- 函数返回值

  无

- 函数用法

  /* 使能PFD功能 */

  PFD_Enable();

4.3.54 PFD_Disable

- 函数

  PFD_Disable();

- 函数功能

  关闭PFD功能，设置寄存器PWMCN[6]=0 。

- 输入参数

  无

- 包含头文件

  Driver/PWM.h

- 函数返回值

  无

- 函数用法

  /* 关闭PFD功能 */

  PFD_Disable();

4.3.55 PFD_Open

- 函数

  void PFD_Open(unsigned char prc) ;

- 函数功能

  使能PFD功能并设置PFD输出波形周期，设置寄存器PWMCN[6]=1，PRC[7:0]，PT2M1[4]。

- 输入参数

  prc [in]：设置PFD输出波形周期控制值，输入范围0x0~0xFF

- 包含头文件

  Driver/PWM.h

- 函数返回值

  无

- 函数用法

  /* 使能PFD功能，设置PFD频率控制值为0x2F */

  PFD_Open(0x2F);

5. 晶片 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

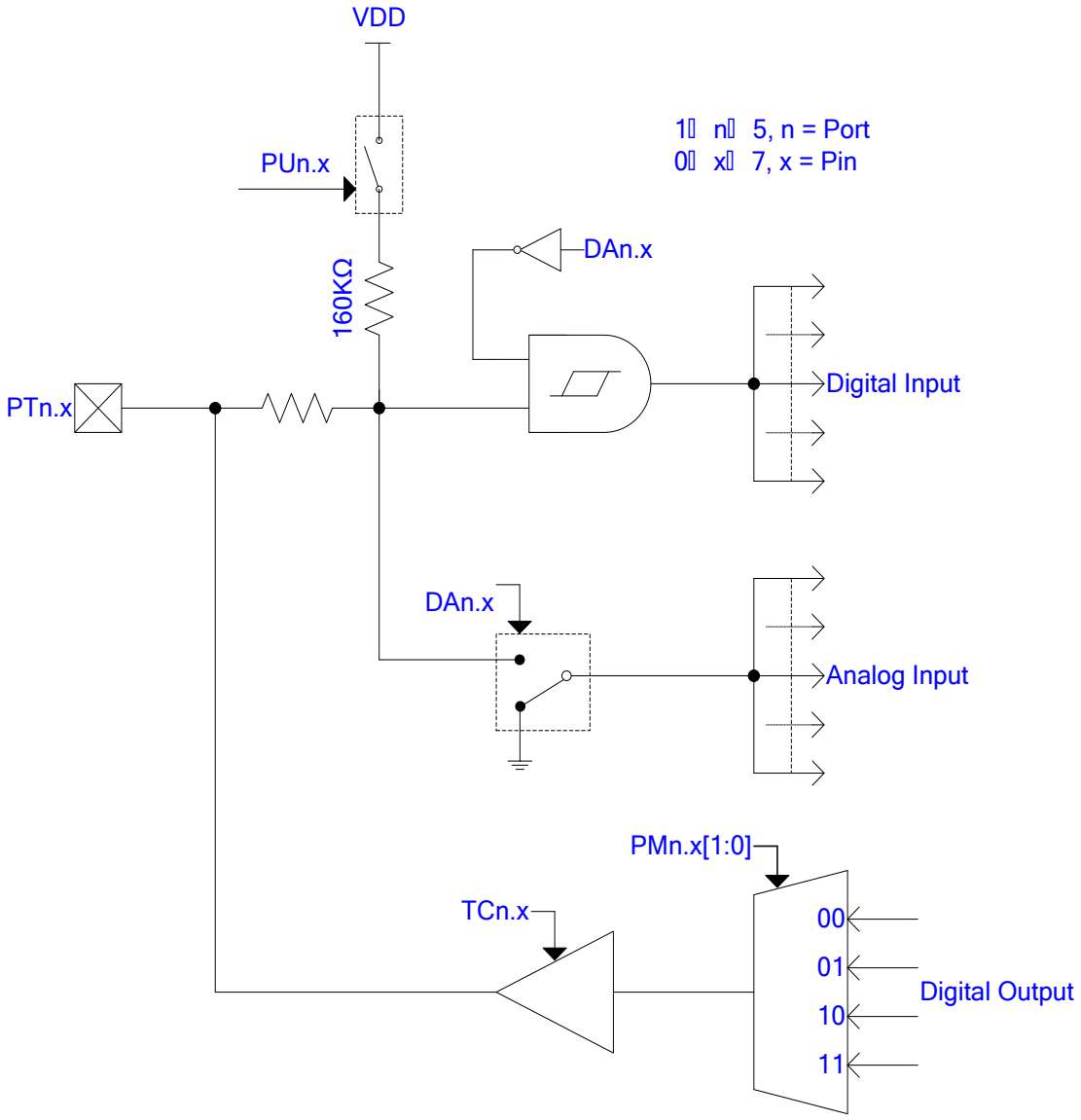
--GPIO 口复用功能控制

序号	函数名称	功能描述
01	GPIO_OpenPT1Input	设置PT1对应引脚输入模式及使能上拉电阻
02	GPIO_OpenPT2Input	设置PT2对应引脚输入模式及使能上拉电阻
03	GPIO_OpenPT3Input	设置PT3对应引脚输入模式及使能上拉电阻
04	GPIO_OpenPT4Input	设置PT4对应引脚输入模式及使能上拉电阻
05	GPIO_OpenPT5Input	设置PT5对应引脚输入模式及使能上拉电阻
06	GPIO_PT1OutputMode	设置PT1作为输出模式
07	GPIO_PT1OutputHigh	设置PT1对应IO输出高电平
08	GPIO_PT1OutputLow	设置PT1对应IO输出低电平
09	GPIO_PT1InputMode	设置PT1对应IO口作为输入模式
10	GPIO_PT1InputPullHight	使能PT1 对应IO口的输入上拉电阻
11	GPIO_PT1InputPullHightClear	关闭PT1对应IO pin的输入上拉电阻
12	GPIO_GetPT1Data	读取PT1输入状态
13	GPIO_PT1AnalogMode	使能PT1.0/PT1.1/PT1.2模拟类功能
14	GPIO_PT1DigitalMode	使能PT1.0/PT1.1/PT1.2数字类功能
15	BZ_Open	设置蜂鸣器驱动频率预分频，及PT1.7作为输出
16	BZ_OutEnable	使能BZ蜂鸣器功能
17	BZ_OutDisable	关闭BZ蜂鸣器输出功能
18	INT0_Enable	使能PT1.0外部中断功能
19	INT0_Disable	关闭PT1.0外部中断功能
20	INT0_IsFlag	读取PT1.0外部中断旗标
21	INT0_ClearFlag	清除PT1.0外部中断请求标志位
22	GPIO_INTEG0Sel	设置PT1.0外部中断触发沿
23	INT1_Enable	使能PT1.1外部中断功能
24	INT1_Disable	关闭PT1.1外部中断功能
25	INT1_IsFlag	读取PT1.1外部中断旗标
26	INT1_ClearFlag	清除PT1.1外部中断请求标志位
27	GPIO_INTEG1Sel	设置PT1.1外部中断触发沿
28	GPIO_PM14Sel	设置PT1.4数字类复用功能
29	GPIO_PM15Sel	设置PT1.5数字类复用功能
30	GPIO_PM16Sel	设置PT1.6数字类复用功能
31	GPIO_PM17Sel	设置PT1.7数字类复用功能
32	GPIO_PT2OutputMode	设置PT2作为输出模式
33	GPIO_PT2OutputHigh	设置PT2对应IO输出高电平
34	GPIO_PT2OutputLow	设置PT2对应IO口输出低电平
35	GPIO_PT2InputMode	设置PT2对应IO口作为输入模式.
36	GPIO_PT2InputPullHight	使能PT2 对应IO口的输入上拉电阻
37	GPIO_PT2InputPullHightClear	关闭PT2对应IO pin的输入上拉电阻

38	GPIO_GetPT2Data	读取PT2输入状态
39	GPIO_PM22Sel	设置PT2.2数字类复用功能
40	GPIO_PM23Sel	设置PT2.3数字类复用功能
41	GPIO_PM24Sel	设置PT2.4数字类复用功能
42	GPIO_PM25Sel	设置PT2.5数字类复用功能
43	GPIO_PM26Sel	设置PT2.6数字类复用功能
44	GPIO_PM27Sel	设置PT2.7数字类复用功能
45	GPIO_PT3OutputMode	设置PT3作为输出模式
46	GPIO_PT3OutputHigh	设置PT3对应IO输出高电平
47	GPIO_PT3OutputLow	设置PT3对应IO口输出低电平
48	GPIO_PT3InputMode	设置PT3对应IO口作为输入模式
49	GPIO_PT3InputPullHight	使能PT3 对应IO口的输入上拉电阻
50	GPIO_PT3InputPullHightClear	关闭PT3对应IO pin的输入上拉电阻
51	GPIO_GetPT3Data	读取PT3输入状态
52	GPIO_PT4InputPullHight	使能PT4 对应IO口的输入上拉电阻
53	GPIO_PT4InputPullHightClear	关闭PT4对应IO pin的输入上拉电阻
54	GPIO_GetPT4Data	读取PT4输入状态
55	GPIO_PT4AnalogMode	使能PT4模拟类功能
56	GPIO_PT4DigitalMode	使能PT4数字类功能
57	GPIO_PT5InputPullHight	使能PT5 对应IO口的输入上拉电阻
58	GPIO_PT5InputPullHightClear	关闭PT5对应IO pin的输入上拉电阻
59	GPIO_GetPT5Data	读取PT5输入状态
60	GPIO_PT5AnalogMode	使能PT5模拟类功能
61	GPIO_PT5DigitalMode	使能PT5数字类功能
62	GPIO_INT0_Enable	使能PT1.0外部中断功能
63	GPIO_INT0_Disable	关闭PT1.0外部中断功能
64	GPIO_INT1_Enable	使能PT1.1外部中断功能
65	GPIO_INT1_Disable	关闭PT1.1外部中断功能
66	GPIO_INT_TYPE_PT10	设置PT1.0外部中断触发沿
67	GPIO_INT_TYPE_PT11	设置PT1.1外部中断触发沿
68	GPIO_INT_Low2BitEnable	使能PT1.1/PT1.0外部中断功能
69	GPIO_AI0Enable	使能PT4.0（AI0）模拟功能
70	GPIO_AI1Enable	使能PT4.1（AI1）模拟功能
71	GPIO_AI2Enable	使能PT4.2（AI2）模拟功能
72	GPIO_AI3Enable	使能PT4.3（AI3）模拟功能
73	GPIO_AI4Enable	使能PT4.4（AI4）模拟功能
74	GPIO_AI5Enable	使能PT4.5（AI5）模拟功能
75	GPIO_AI6Enable	使能PT4.6（AI6）模拟功能
76	GPIO_AI7Enable	使能PT4.7（AI7）模拟功能
77	GPIO_AI8Enable	使能PT5.0（AI8）模拟功能
78	GPIO_AI9Enable	使能PT5.1（AI9）模拟功能
79	GPIO_AI10Enable	使能PT5.2（AI10）模拟功能
80	GPIO_AI11Enable	使能PT5.3（AI11）模拟功能
81	GPIO_CPAI0Enable	使能PT2.2复用为比较器信号输入端CPAI0
82	GPIO_CPAI0Disable	关闭PT2.2复用为比较器信号输入端CPAI0
83	GPIO_CPAI1Enable	使能PT2.3复用为比较器信号输入端CPAI1
84	GPIO_CPAI1Disable	关闭PT2.3复用为比较器信号输入端CPAI1
85	GPIO_CPAI2Enable	使能PT2.4复用为比较器信号输入端CPAI2
86	GPIO_CPAI2Disable	关闭PT2.4复用为比较器信号输入端CPAI2
87	GPIO_CPAI3Enable	使能PT2.5复用为比较器信号输入端CPAI3
88	GPIO_CPAI3Disable	关闭PT2.5复用为比较器信号输入端CPAI3
89	GPIO_CPAI4Enable	使能PT2.6复用为比较器信号输入端CPAI4

90	GPIO_CPAI4Disable	关闭PT2.6复用为比较器信号输入端CPAI4
91	GPIO_CPAI5Enable	使能PT2.7复用为比较器信号输入端CPAI5
92	GPIO_CPAI5Disable	关闭PT2.7复用为比较器信号输入端CPAI5
93	GPIO_CPAI6Enable	使能PT1.0复用为比较器信号输入端CPAI6
94	GPIO_CPAI6Disable	关闭PT1.0复用为比较器信号输入端CPAI6
95	GPIO_CPAI7Enable	使能PT1.1复用为比较器信号输入端CPAI7
96	GPIO_CPAI7Disable	关闭PT1.1复用为比较器信号输入端CPAI7

5.2 GPIO 模块方框图



5.3 函数说明

5.3.1 GPIO_OpenPT1Input

- **函数**

  void GPIO_OpenPT1Input(unsigned char InputBits, unsigned char PullHighBits);

- **函数功能**

  设置PT1对应引脚作为GPIO的输入模式及使能上拉电阻，设置PT1寄存器TRISC1/PT1PU/PT1DA。

- **输入参数**

  InputBitst [in]：代表GPIO port，PT1.0~PT1.3默认为输入模式，输入参数如下

  TRISC1_TC14_INPUT：PT1.4作为输入模式

  TRISC1_TC15_INPUT：PT1.5作为输入模式

  TRISC1_TC16_INPUT：PT1.6作为输入模式

  TRISC1_TC17_INPUT：PT1.7作为输入模式

  注意：PT1.0~PT1.3默认为输入模式，不用设置

  PullHighBits [in]：代表 GPIO port，设置输入上拉电阻，输入参数如下

  PT1PU_PU10_ENABLE：使能PT1.0上拉电阻； PT1PU_PU10_DISABLE：关闭PT1.0上拉电阻

  PT1PU_PU11_ENABLE：使能PT1.1上拉电阻； PT1PU_PU11_DISABLE：关闭PT1.1上拉电阻

  PT1PU_PU12_ENABLE：使能PT1.2上拉电阻； PT1PU_PU12_DISABLE：关闭PT1.2上拉电阻

  PT1PU_PU13_ENABLE：使能PT1.3上拉电阻； PT1PU_PU13_DISABLE：关闭PT1.3上拉电阻

  PT1PU_PU14_ENABLE：使能PT1.4上拉电阻； PT1PU_PU14_DISABLE：关闭PT1.4上拉电阻

  PT1PU_PU15_ENABLE：使能PT1.5上拉电阻； PT1PU_PU15_DISABLE：关闭PT1.5上拉电阻

  PT1PU_PU16_ENABLE：使能PT1.6上拉电阻； PT1PU_PU16_DISABLE：关闭PT1.6上拉电阻

  PT1PU_PU17_ENABLE：使能PT1.7上拉电阻； PT1PU_PU17_DISABLE：关闭PT1.7上拉电阻

- **包含头文件**

  Driver/GPIO.h

- **函数返回值**

  无

- **函数用法**

```
/* 设置PT1.4/PT1.5作为输入模式及使能输入上拉电阻 */
GPIO_OpenPT1Input(TRISC1_TC14_INPUT, PT1PU_PU14_ENABLE); //PT1.4打开输入模式及使能上拉
GPIO_OpenPT1Input(TRISC1_TC15_INPUT, PT1PU_PU15_ENABLE); //PT1.5打开输入模式及使能上拉
```

5.3.2 GPIO_OpenPT2Input

- **函数**

  void GPIO_OpenPT2Input(unsigned char InputBits,unsigned char PullHighBits);

- **函数功能**

  设置PT2对应引脚作为GPIO的输入模式及使能上拉电阻，设置PT2寄存器TRISC2/PT2PU。

- **输入参数**

InputBitst [in]：代表GPIO port PT2，输入参数如下

TRISC2_TC20_INPUT：PT2.0作为输入模式

TRISC2_TC21_INPUT：PT2.1作为输入模式

TRISC2_TC22_INPUT：PT2.2作为输入模式

TRISC2_TC23_INPUT：PT2.3作为输入模式

TRISC2_TC24_INPUT：PT2.4作为输入模式

TRISC2_TC25_INPUT：PT2.5作为输入模式

TRISC2_TC26_INPUT：PT2.6作为输入模式

TRISC2_TC27_INPUT：PT2.7作为输入模式

PullHighBits [in]：代表 GPIO port PT2，设置输入上拉电阻，输入参数如下

PT2PU_PU20_ENABLE：使能PT2.0上拉电阻； PT2PU_PU20_DISABLE：关闭PT2.0上拉电阻

PT2PU_PU21_ENABLE：使能PT2.1上拉电阻； PT2PU_PU21_DISABLE：关闭PT2.1上拉电阻

PT2PU_PU22_ENABLE：使能PT2.2上拉电阻； PT2PU_PU22_DISABLE：关闭PT2.2上拉电阻

PT2PU_PU23_ENABLE：使能PT2.3上拉电阻； PT2PU_PU23_DISABLE：关闭PT2.3上拉电阻

PT2PU_PU24_ENABLE：使能PT2.4上拉电阻； PT2PU_PU24_DISABLE：关闭PT2.4上拉电阻

PT2PU_PU25_ENABLE：使能PT2.5上拉电阻； PT2PU_PU25_DISABLE：关闭PT2.5上拉电阻

PT2PU_PU26_ENABLE：使能PT2.6上拉电阻； PT2PU_PU26_DISABLE：关闭PT2.6上拉电阻

PT2PU_PU27_ENABLE：使能PT2.7上拉电阻； PT2PU_PU27_DISABLE：关闭PT2.7上拉电阻

- **包含头文件**

Driver/GPIO.h

- **函数返回值**

无

- **函数用法**

```
/* 设置PT2.4/PT2.5作为输入模式及使能输入上拉电阻 */
GPIO_OpenPT2Input(TRISC2_TC24_INPUT, PT2PU_PU24_ENABLE); //PT2.4打开输入模式及使能上拉
GPIO_OpenPT2Input(TRISC2_TC25_INPUT, PT2PU_PU25_ENABLE); //PT2.5打开输入模式及使能上拉
```

5.3.3 GPIO_OpenPT3Input

- **函数**

void GPIO_OpenPT3Input(unsigned char InputBits, unsigned char PullHighBits);

- **函数功能**

设置PT3对应引脚作为GPIO的输入模式及使能上拉电阻，设置PT3寄存器TRISC3/PT3PU。

- **输入参数**

InputBitst [in]：代表GPIO port PT3，输入参数如下

TRISC3_TC30_INPUT：PT3.0作为输入模式

TRISC3_TC31_INPUT：PT3.1作为输入模式

TRISC3_TC32_INPUT：PT3.2作为输入模式

TRISC3_TC33_INPUT：PT3.3作为输入模式

TRISC3_TC34_INPUT：PT3.4作为输入模式

TRISC3_TC35_INPUT：PT3.5作为输入模式

TRISC3_TC36_INPUT：PT3.6作为输入模式

TRISC3_TC37_INPUT：PT3.7作为输入模式

PullHighBits [in]：代表 GPIO port PT3，设置输入上拉电阻，输入参数如下

PT3PU_PU30_ENABLE：使能PT3.0上拉电阻； PT3PU_PU30_DISABLE：关闭PT3.0上拉电阻

PT3PU_PU31_ENABLE：使能PT3.1上拉电阻； PT3PU_PU31_DISABLE：关闭PT3.1上拉电阻

PT3PU_PU32_ENABLE：使能PT3.2上拉电阻； PT3PU_PU32_DISABLE：关闭PT3.2上拉电阻

PT3PU_PU33_ENABLE：使能PT3.3上拉电阻； PT3PU_PU33_DISABLE：关闭PT3.3上拉电阻

PT3PU_PU34_ENABLE：使能PT3.4上拉电阻； PT3PU_PU34_DISABLE：关闭PT3.4上拉电阻

PT3PU_PU35_ENABLE：使能PT3.5上拉电阻； PT3PU_PU35_DISABLE：关闭PT3.5上拉电阻

PT3PU_PU36_ENABLE：使能PT3.6上拉电阻； PT3PU_PU36_DISABLE：关闭PT3.6上拉电阻

PT3PU_PU37_ENABLE：使能PT3.7上拉电阻； PT3PU_PU37_DISABLE：关闭PT3.7上拉电阻

- **包含头文件**

Driver/GPIO.h

- **函数返回值**

无

- **函数用法**

```
/* 设置PT3.4/PT3.5作为输入模式及使能输入上拉电阻 */
GPIO_OpenPT3Input(TRISC3_TC34_INPUT, PT3PU_PU34_ENABLE); //PT3.4打开输入模式及使能上拉
GPIO_OpenPT3Input(TRISC3_TC35_INPUT, PT3PU_PU35_ENABLE); //PT3.5打开输入模式及使能上拉
```

5.3.4 GPIO_OpenPT4Input

- **函数**

void GPIO_OpenPT4Input(unsigned char InputBits, unsigned char PullHighBits);

- **函数功能**

设置PT4对应引脚作为GPIO的输入模式及使能上拉电阻，设置PT4寄存器PT4DA/PT4PU。

- **输入参数**

InputBitst [in]：代表GPIO port PT4，PT4默认作为输入口，需要关闭模拟类功能，输入参数如下

PT4DA_DA40_DIG：设置PT4.0作为数字口

PT4DA_DA41_DIG：设置PT4.1作为数字口

PT4DA_DA42_DIG：设置PT4.2作为数字口

PT4DA_DA43_DIG：设置PT4.3作为数字口

PT4DA_DA44_DIG：设置PT4.4作为数字口

PT4DA_DA45_DIG：设置PT4.5作为数字口

PT4DA_DA46_DIG：设置PT4.6作为数字口

PT4DA_DA47_DIG：设置PT4.7作为数字口

PullHighBits [in]：代表 GPIO port PT4，设置输入上拉电阻，输入参数如下

PT4PU_PU40_ENABLE：使能PT4.0上拉电阻；　PT4PU_PU40_DISABLE：关闭PT4.0上拉电阻
PT4PU_PU41_ENABLE：使能PT4.1上拉电阻；　PT4PU_PU41_DISABLE：关闭PT4.1上拉电阻
PT4PU_PU42_ENABLE：使能PT4.2上拉电阻；　PT4PU_PU42_DISABLE：关闭PT4.2上拉电阻
PT4PU_PU43_ENABLE：使能PT4.3上拉电阻；　PT4PU_PU43_DISABLE：关闭PT4.3上拉电阻
PT4PU_PU44_ENABLE：使能PT4.4上拉电阻；　PT4PU_PU44_DISABLE：关闭PT4.4上拉电阻
PT4PU_PU45_ENABLE：使能PT4.5上拉电阻；　PT4PU_PU45_DISABLE：关闭PT4.5上拉电阻
PT4PU_PU46_ENABLE：使能PT4.6上拉电阻；　PT4PU_PU46_DISABLE：关闭PT4.6上拉电阻
PT4PU_PU47_ENABLE：使能PT4.7上拉电阻；　PT4PU_PU47_DISABLE：关闭PT4.7上拉电阻

- **包含头文件**

Driver/GPIO.h

- **函数返回值**

无

- **函数用法**

```
/* 设置PT4.4/PT4.5作为输入模式及使能输入上拉电阻 */
GPIO_OpenPT4Input(PT4DA_DA44_DIG, PT4PU_PU44_ENABLE); //PT4.4打开输入模式及使能上拉
GPIO_OpenPT4Input(PT4DA_DA45_DIG, PT4PU_PU45_ENABLE); //PT4.5打开输入模式及使能上拉
```

5.3.5 GPIO_OpenPT5Input

- **函数**

void GPIO_OpenPT5Input(unsigned char InputBits, unsigned char PullHighBits);

- **函数功能**

设置PT5对应引脚作为GPIO的输入模式及使能上拉电阻，设置PT5寄存器PT5DA/PT5PU。

- **输入参数**

InputBitst [in]：代表GPIO port PT5，PT4默认作为输入口，需要关闭模拟类功能，输入参数如下

PT5DA_DA50_DIG：设置PT5.0作为数字口
PT5DA_DA51_DIG：设置PT5.1作为数字口
PT5DA_DA52_DIG：设置PT5.2作为数字口
PT5DA_DA53_DIG：设置PT5.3作为数字口

PullHighBits [in]：代表 GPIO port PT5，设置输入上拉电阻，输入参数如下

PT5PU_PU50_ENABLE：使能PT5.0上拉电阻；　PT5PU_PU50_DISABLE：关闭PT5.0上拉电阻
PT5PU_PU51_ENABLE：使能PT5.1上拉电阻；　PT5PU_PU51_DISABLE：关闭PT5.1上拉电阻
PT5PU_PU52_ENABLE：使能PT5.2上拉电阻；　PT5PU_PU52_DISABLE：关闭PT5.2上拉电阻
PT5PU_PU53_ENABLE：使能PT5.3上拉电阻；　PT5PU_PU53_DISABLE：关闭PT5.3上拉电阻
PT5PU_PU54_ENABLE：使能PT5.4上拉电阻；　PT5PU_PU54_DISABLE：关闭PT5.4上拉电阻
PT5PU_PU55_ENABLE：使能PT5.5上拉电阻；　PT5PU_PU55_DISABLE：关闭PT5.5上拉电阻
PT5PU_PU56_ENABLE：使能PT5.6上拉电阻；　PT5PU_PU56_DISABLE：关闭PT5.6上拉电阻
PT5PU_PU57_ENABLE：使能PT5.7上拉电阻；　PT5PU_PU57_DISABLE：关闭PT5.7上拉电阻

- 包含头文件

Driver/GPIO.c

- 函数返回值

无

- 函数用法

```
/* 设置PT5.4/PT5.5作为输入模式及使能输入上拉电阻 */
GPIO_OpenPT5Input(PT4DA_DA54_DIG, PT5PU_PU54_ENABLE); //PT5.4打开输入模式及使能上拉
GPIO_OpenPT5Input(PT4DA_DA55_DIG, PT5PU_PU55_ENABLE); //PT5.5打开输入模式及使能上拉
```

5.3.6 GPIO_PT1OutputMode

- 函数

GPIO_PT1OutputMode(BitSet);

- 函数功能

设置PT1作为输出模式，操作寄存器TRISC1。

- 输入参数

BitSet [in]：代表GPIO port PT1，PT1.0~PT1.3默认为输入，设置参数如下

TRISC1_TC14_OUTPUT：PT1.4作为输出模式

TRISC1_TC15_OUTPUT：PT1.5作为输出模式

TRISC1_TC16_OUTPUT：PT1.6作为输出模式

TRISC1_TC17_OUTPUT：PT1.7作为输出模式

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 设置PT1.5/PT1.4的输出模式 */
GPIO_PT1OutputMode( TRISC1_TC14_OUTPUT );
GPIO_PT1OutputMode( TRISC1_TC15_OUTPUT );
//或是如下用法
GPIO_PT1OutputMode( TRISC1_TC15_OUTPUT | TRISC1_TC14_OUTPUT );
```

5.3.7 GPIO_PT1OutputHigh

- 函数

GPIO_PT1OutputHigh(BitSet);

- 函数功能

设置PT1对应IO输出高电平，操作寄存器PT1[7 :4]。

- 输入参数

BitSet [in]：代表GPIO port PT1，8bit数据bit0~bit7分别对应PT1.0~PT1.7，输入参数如下

PT1_PT14_H：PT1.4输出高电平

PT1_PT15_H：PT1.5输出高电平

PT1_PT16_H：PT1.6输出高电平

PT1_PT17_H：PT1.7输出高电平

注意：PT1.0~PT1.3默认是输出模式，所以不能设置输出及输出高电平

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* PT1.4输出高电平 */
GPIO_PT1OutputMode( TRISC1_TC14_OUTPUT );          //设置PT1.4作为输出模式
GPIO_PT1OutputHigh( PT1_PT14_H );                  //设置PT1.4输出高电平
```

5.3.8 GPIO_PT1OutputLow

- 函数

GPIO_PT1OutputLow(BitSet);

- 函数功能

设置PT1对应IO口输出低电平，操作寄存器PT1[7:4]。

- 输入参数

BitSet [in]：代表GPIO port PT1，8bit数据bit0~bit7分别对应PT1.0~PT1.7，输入参数如下

PT1_PT14_L：PT1.4输出低电平

PT1_PT15_L：PT1.5输出低电平

PT1_PT16_L：PT1.6输出低电平

PT1_PT17_L：PT1.7输出低电平

注意：PT1.0~PT1.3默认是输出模式，所以不能设置输出及输出低电平

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* PT1.4输出低电平 */
GPIO_PT1OutputMode( TRISC1_TC14_OUTPUT );          //设置PT1.4作为输出模式
GPIO_PT1OutputHigh( PT1_PT14_L );                  //设置PT1.4输出低电平
```

5.3.9 GPIO_PT1InputMode

- 函数

GPIO_PT1InputMode(BitSet);

- **函数功能**

设置PT1对应IO口作为输入模式，设置寄存器TRISC1。

- **输入参数**

BitSet [in]：代表PT1对应IO pin，PT1.0~PT1.3默认是输入口，只需针对PT1.4~PT1.7设置，输入参数如下

TRISC1_TC14_INPUT：PT1.4作为输入模式

TRISC1_TC15_INPUT：PT1.5作为输入模式

TRISC1_TC16_INPUT：PT1.6作为输入模式

TRISC1_TC17_INPUT：PT1.7作为输入模式

注意：PT1.0~PT1.3默认为输入模式，不用设置

- **包含头文件**

Driver/GPIO.h

- **函数返回值**

无

- **函数用法**

```
/*  设置PT1.4作为输入口  */
GPIO_PT1InputMode( TRISC1_TC14_INPUT );
```

5.3.10 GPIO_PT1InputPullHight

- **函数**

GPIO_PT1InputPullHight(BitSet);

- **函数功能**

使能PT1 对应IO口的输入上拉电阻，设置寄存器PT1PU。

- **输入参数**

BitSet [in]：代表GPIO PT1，输入值如下

PT1PU_PU10_ENABLE：使能PT1.0上拉电阻

PT1PU_PU11_ENABLE：使能PT1.1上拉电阻

PT1PU_PU12_ENABLE：使能PT1.2上拉电阻

PT1PU_PU13_ENABLE：使能PT1.3上拉电阻

PT1PU_PU14_ENABLE：使能PT1.4上拉电阻

PT1PU_PU15_ENABLE：使能PT1.5上拉电阻

PT1PU_PU16_ENABLE：使能PT1.6上拉电阻

PT1PU_PU17_ENABLE：使能PT1.7上拉电阻

- **包含头文件**

Driver/GPIO.h

- **函数返回值**

无

- **函数用法**

```
/*  使能 PT1.4 输入上拉电阻  */
GPIO_PT1InputPullHight( PT1PU_PU14_ENABLE );
```

5.3.11 GPIO_PT1InputPullHightClear

- 函数

GPIO_PT1InputPullHightClear(BitSet) ;

- 函数功能

关闭PT1对应IO pin的输入上拉电阻，设置寄存器PT1PU。

- 输入参数

BitSet [in]：代表GPIO port PT1输入值如下

PT1PU_PU10_DISABLE：关闭PT1.0上拉电阻

PT1PU_PU11_DISABLE：关闭PT1.1上拉电阻

PT1PU_PU12_DISABLE：关闭PT1.2上拉电阻

PT1PU_PU13_DISABLE：关闭PT1.3上拉电阻

PT1PU_PU14_DISABLE：关闭PT1.4上拉电阻

PT1PU_PU15_DISABLE：关闭PT1.5上拉电阻

PT1PU_PU16_DISABLE：关闭PT1.6上拉电阻

PT1PU_PU17_DISABLE：关闭PT1.7上拉电阻

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 关闭 PT1.4 输入上拉电阻 */
GPIO_PT1InputPullHightClear( PT1PU_PU14_DISABLE );
```

5.3.12 GPIO_GetPT1Data

- 函数

GPIO_GetPT1Data(PT1Data);

- 函数功能

读取PT1输入状态，读取寄存器PT1。

- 输入参数

PT1Data [in]：用于存放读取到的PT1输入状态值

- 包含头文件

Driver/GPIO.h

- 函数返回值

PT1Data：返回 PT1 输入状态值

- 函数用法

```
/* 读取 PT1 输入状态值 */
unsigned char PT1_DATA;
GPIO_GetPT1Data( PT1_DATA );
```

5.3.13 GPIO_PT1AnalogMode

- 函数

GPIO_PT1AnalogMode(BitSet);

- 函数功能

使能PT1.0/PT1.1/PT1.2模拟类功能，设置寄存器PT1DA[2:0]。

- 输入参数

BitSet [in]：代表GPIO port PT1输入值如下

PT1DA_DA10_CPAI6：PT1.0作为CPAI6输入口

PT1DA_DA11_CPAI7：PT1.1作为CPAI7输入口

PT1DA_DA12_LVDIN：PT1.2作为LVD输入引脚

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 使能PT1.2的模拟类功能 */
GPIO_PT1AnalogMode( PT1DA_DA12_LVDIN );          //使能PT1.2模拟类功能；
```

5.3.14 GPIO_PT1DigitalMode

- 函数

GPIO_PT1DigitalMode(BitSet);

- 函数功能

使能PT1.0/PT1.1/PT1.2数字类功能，设置寄存器PT1DA[2:0]。

- 输入参数

BitSet [in]：代表GPIO port PT1，输入值如下

PT1DA_DA10_DIG：使能PT1.0数字类功能

PT1DA_DA11_DIG：使能PT1.1数字类功能

PT1DA_DA12_DIG：使能PT1.2数字类功能

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 使能PT1.2数字类功能 */
GPIO_PT1DigitalMode( PT1DA_DA12_DIG );          //使能PT1.2数字类功能
```

5.3.15 BZ_Open

- 函数

void BZ_Open(unsigned char cks);

- **函数功能**

设置蜂鸣器驱动频率预分频及设置PT1.7作为输出模式，PT10寄存器0X40890[19:18][3:2]。

- **输入参数**

cks [in]：设置蜂鸣器驱动频率预分频

MCKCN3_BZS_PERCKDIV128 ：蜂鸣器驱动频率为PER_CK/128
MCKCN3_BZS_PERCKDIV64 ：蜂鸣器驱动频率为PER_CK/64
MCKCN3_BZS_PERCKDIV32 ：蜂鸣器驱动频率为PER_CK/32
MCKCN3_BZS_PERCKDIV16 ：蜂鸣器驱动频率为PER_CK/16
MCKCN3_BZS_PERCKDIV8 ：蜂鸣器驱动频率为PER_CK/8
MCKCN3_BZS_PERCKDIV4 ：蜂鸣器驱动频率为PER_CK/4
MCKCN3_BZS_PERCKDIV3 ：蜂鸣器驱动频率为PER_CK/3
MCKCN3_BZS_PERCKDIV1 ：蜂鸣器驱动频率为PER_CK/1

- **包含头文件**

Driver/GPIO.h

- **函数返回值**

无

- **函数用法**

```
/* 设置蜂鸣器驱动频率为PER_CK/16 */
BZ_Open( MCKCN3_BZS_PERCKDIV16 );                    //设置蜂鸣器驱动频率16分频
```

5.3.16 BZ_OutEnable

- **函数**

BZ_OutEnable();

- **函数功能**

使能BZ蜂鸣器功能，设置寄存器P1M2[6]=1 。

- **输入参数**

无

- **包含头文件**

Driver/GPIO.h

- **函数返回值**

无

- **函数用法**

```
/* 使能BZ输出功能 */
BZ_Open( MCKCN3_BZS_PERCKDIV16 );                    //设置蜂鸣器驱动频率16分频
BZ_OutEnable();
```

5.3.17 BZ_OutDisable

- 函数

BZ_OutDisable();

- 函数功能

. 关闭BZ蜂鸣器输出功能，设置寄存器PT1M2[6]=0 。

- 输入参数

无

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 关闭BZ输出功能 */
BZ_OutDisable();                                    //关闭蜂鸣器输出功能
GPIO_PT1InputMode( TRISC1_TC17_INPUT );             //设置PT1.7作为输入
```

5.3.18 INT0_Enable

- 函数

INT0_Enable();

- 函数功能

使能PT1.0外部中断功能，设置寄存器INTE1[0]=1 。

- 输入参数

无.

- 包含头文件

Driver/INT.h

- 函数返回值

无

- 函数用法

```
/* 使能PT1.0外部中断功能 */
INT0_Enable();
```

5.3.19 INT0_Disable

- 函数

INT0_Disable();

- 函数功能

关闭PT1.0外部中断功能，设置寄存器INTE1[0]=0 。

- 输入参数

无

- 包含头文件

Driver/INT.h

- 函数返回值

无

- 函数用法

```
/* 关闭PT1.0外部中断功能 */
INT0_Disable();
```

5.3.20 INT0_IsFlag

- 函数

INT0_IsFlag();

- 函数功能

读取PT1.0外部中断旗标，读取寄存器INTF0[0]。

- 输入参数

无

- 包含头文件

Driver/INT.h

- 函数返回值

0x00：PT1.0没有中断请求

0x01：PT1.0产生中断请求

- 函数用法

```
/* 读取PT1.0的外部中断旗标 */
unsigned char flag;
flag = INT0_IsFlag();
```

5.3.21 INT0_ClearFlag

- 函数

INT0_ClearFlag();

- 函数功能

清除PT1.0外部中断请求标志位，设置寄存器INTF0[0]=0 。

- 输入参数

无

- 包含头文件

Driver/INT.h

- 函数返回值

无

- 函数用法

```
/* 清除PT1.0外部中断请求标志位 */
INT0_ClearFlag();
```

5.3.22 GPIO_INTEG0Sel

- 函数

  GPIO_INTEG0Sel(EG0Sel);

- 函数功能

  设置PT1.0外部中断触发沿，设置寄存器PT1M1[1:0]。

- 输入参数

  EG0Sel [in]：PT1.0外部中断触发沿选择

  PT1M1_INTEG0_LEV_ ：电平变化触发

  PT1M1_INTEG0_LEV ：电平变化触发

  PT1M1_INTEG0_EDGERISE：上升沿触发

  PT1M1_INTEG0_EDGEFALL：下降沿触发

- 包含头文件

  Driver/GPIO.h

- 函数返回值

  无

- 函数用法

```
/* 设置PT1.0外部中断触发沿为下降沿触发 */
GPIO_INTEG0Sel( PT1M1_INTEG0_EDGEFALL );
```

5.3.23 INT1_Enable

- 函数

  INT1_Enable();

- 函数功能

  使能PT1.1外部中断功能，设置寄存器INTE1[1]=1 。

- 输入参数

  无.

- 包含头文件

  Driver/INT.h

- 函数返回值

  无

- 函数用法

```
/* 使能PT1.1外部中断功能 */
INT1_Enable();
```

5.3.24 INT1_Disable

- 函数

  INT1_Disable();

- 函数功能

关闭PT1.1外部中断功能，设置寄存器INTE1[1]=0 。

- 输入参数

无

- 包含头文件

Driver/INT.h

- 函数返回值

无

- 函数用法

/* 关闭PT1.1外部中断功能 */

INT0_Disable();

5.3.25 INT1_IsFlag

- 函数

INT1_IsFlag();

- 函数功能

读取PT1.1外部中断旗标，读取寄存器INTF0[1]。

- 输入参数

无

- 包含头文件

Driver/INT.h

- 函数返回值

0x00：PT1.1没有中断请求

0x02：PT1.1产生中断请求

- 函数用法

/* 读取PT1.1的外部中断旗标 */

unsigned char flag;

flag = INT1_IsFlag();

5.3.26 INT1_ClearFlag

- 函数

INT1_ClearFlag();

- 函数功能

清除PT1.1外部中断请求标志位，设置寄存器INTF0[1]=0 。

- 输入参数

无

- 包含头文件

Driver/INT.h

- 函数返回值

无

- 函数用法

/* 清除PT1.1外部中断请求标志位 */

INT1_ClearFlag();

5.3.27 GPIO_INTEG1Sel

- 函数

GPIO_INTEG1Sel(EG1Sel);

- 函数功能

设置PT1.1外部中断触发沿，设置寄存器PT1M1[3:2]。

- 输入参数

EG1Sel [in]：PT1.1外部中断触发沿选择

PT1M1_INTEG1_LEV_ ：电平变化触发

PT1M1_INTEG1_LEV ：电平变化触发

PT1M1_INTEG1_EDGERISE：上升沿触发

PT1M1_INTEG1_EDGEFALL：下降沿触发

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

/* 设置PT1.1外部中断触发沿为下降沿触发 */

GPIO_INTEG1Sel(PT1M1_INTEG1_EDGEFALL);

5.3.28 GPIO_PM14Sel

- 函数

GPIO_PM14Sel(PM14Sel);

- 函数功能

设置PT1.4数字类复用功能，设置寄存器PT1M2[0]。

- 输入参数

PM14Sel [in]：PT1.4复用功能选择

PT1M2_PM14_TX ：PT1.4复用为TX 引脚

PT1M2_PM14_GPIO：PT1.4复用为GPIO

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* PT1.4复用TX引脚 */
GPIO_PM14Sel( PT1M2_PM14_TX );
```

5.3.29 GPIO_PM15Sel

- **函数**

GPIO_PM15Sel(PM15Sel);

- **函数功能**

设置PT1.5数字类复用功能，设置寄存器PT1M2[2]。

- **输入参数**

PM15Sel [in] PT1.5数字类复用功能设置

PT1M2_PM15_SDO ：PT1.5复用SPI通讯的SDO引脚

PT1M2_PM15_GPIO：PT1.5复用为GPIO

- **包含头文件**

Driver/GPIO.h

- **函数返回值**

无

- **函数用法**

```
/* PT1.5复用为SPI的SDO引脚 */
GPIO_PM15Sel( PT1M2_PM15_SDO );
```

5.3.30 GPIO_PM16Sel

- **函数**

GPIO_PM16Sel(PM16Sel);

- **函数功能**

设置PT1.6数字类复用功能，设置寄存器PT1M2[4]。

- **输入参数**

PM16Sel [in]：PT1.6复用功能设置

PT1M2_PM16_SCK ：PT1.6复用为SPI通讯引脚SCK

PT1M2_PM16_GPIO：PT1.6复用GPIO

- **包含头文件**

Driver/GPIO.h

- **函数返回值**

无

- **函数用法**

```
/* PT1.6复用为SPI的SCK引脚 */
GPIO_PM16Sel( PT1M2_PM16_SCK );
```

5.3.31 GPIO_PM17Sel

- 函数

GPIO_PM17Sel(PM17Sel);

- 函数功能

设置PT1.7数字类复用功能，设置寄存器PT1M2[6]。

- 输入参数

PM17Sel [in]：PT1.7复用功能设置

PT1M2_PM17_BZ ：PT1.7复用为蜂鸣器驱动引脚

PT1M2_PM17_GPIO：PT1.7复用为GPIO

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 设置PT1.7作为BZ驱动引脚 */
GPIO_PM17Sel( PT1M2_PM17_BZ );
```

5.3.32 GPIO_PT2OutputMode

- 函数

GPIO_PT2OutputMode(BitSet);

- 函数功能

设置PT2作为输出模式，操作寄存器TRISC2。

- 输入参数

BitSet [in]：代表GPIO port PT2，设置参数如下

TRISC2_TC20_OUTPUT：PT2.0作为输出模式

TRISC2_TC21_OUTPUT：PT2.1作为输出模式

TRISC2_TC22_OUTPUT：PT2.2作为输出模式

TRISC2_TC23_OUTPUT：PT2.3作为输出模式

TRISC2_TC24_OUTPUT：PT2.4作为输出模式

TRISC2_TC25_OUTPUT：PT2.5作为输出模式

TRISC2_TC26_OUTPUT：PT2.6作为输出模式

TRISC2_TC27_OUTPUT：PT2.7作为输出模式

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 设置PT2.5/PT2.4的输出模式 */
```

```
GPIO_PT2OutputMode( TRISC2_TC24_OUTPUT );
GPIO_PT2OutputMode( TRISC2_TC25_OUTPUT );
//或改成以下用法
GPIO_PT2OutputMode( TRISC2_TC25_OUTPUT | TRISC2_TC24_OUTPUT );
```

5.3.33 GPIO_PT2OutputHigh

- **函数**

GPIO_PT2OutputHigh(BitSet);

- **函数功能**

设置PT2对应IO输出高电平，操作寄存器PT2[7:0] 。

- **输入参数**

BitSet [in]：代表GPIO port PT2，8bit数据bit0~bit7分别对应PT2.0~PT2.7，输入参数如下

PT2_PT20_H：PT2.0输出高电平

PT2_PT21_H：PT2.1输出高电平

PT2_PT22_H：PT2.2输出高电平

PT2_PT23_H：PT2.3输出高电平

PT2_PT24_H：PT2.4输出高电平

PT2_PT25_H：PT2.5输出高电平

PT2_PT26_H：PT2.6输出高电平

PT2_PT27_H：PT2.7输出高电平

- **包含头文件**

Driver/GPIO.h

- **函数返回值**

无

- **函数用法**

```
/* PT2.4输出高电平 */
GPIO_PT2OutputMode( TRISC2_TC24_OUTPUT );          //设置PT2.4作为输出模式
GPIO_PT2OutputHigh( PT2_PT24_H );                  //设置PT2.4输出高电平
```

5.3.34 GPIO_PT2OutputLow

- **函数**

GPIO_PT2OutputLow(BitSet);

- **函数功能**

设置PT2对应IO口输出低电平，操作寄存器PT2[7:0]。

- **输入参数**

BitSet [in]：代表GPIO port PT2，8bit数据bit0~bit7分别对应PT2.0~PT2.7，输入参数如下

PT2_PT20_L：PT2.0输出低电平

PT2_PT21_L：PT2.1输出低电平

PT2_PT22_L：PT2.2输出低电平

PT2_PT23_L：PT2.3输出低电平

PT2_PT24_L：PT2.4输出低电平

PT2_PT25_L：PT2.5输出低电平

PT2_PT26_L：PT2.6输出低电平

PT2_PT27_L：PT2.7输出低电平

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* PT2.4输出低电平 */
GPIO_PT2OutputMode( TRISC2_TC24_OUTPUT );          //设置PT2.4作为输出模式
GPIO_PT2OutputHigh( PT2_PT24_L );                  //设置PT2.4输出低电平
```

5.3.35 GPIO_PT2InputMode

- 函数

GPIO_PT2InputMode(BitSet);

- 函数功能

设置PT2对应IO口作为输入模式，设置寄存器TRISC2。

- 输入参数

BitSet [in]：代表PT2对应IO pin，针对PT2.0~PT2.7设置，输入参数如下

TRISC2_TC20_INPUT：PT2.0作为输入模式

TRISC2_TC21_INPUT：PT2.1作为输入模式

TRISC2_TC22_INPUT：PT2.2作为输入模式

TRISC2_TC23_INPUT：PT2.3作为输入模式

TRISC2_TC24_INPUT：PT2.4作为输入模式

TRISC2_TC25_INPUT：PT2.5作为输入模式

TRISC2_TC26_INPUT：PT2.6作为输入模式

TRISC2_TC27_INPUT：PT2.7作为输入模式

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/*  设置PT2.4作为输入口 */
GPIO_PT2InputMode( TRISC2_TC24_INPUT );
```

5.3.36 GPIO_PT2InputPullHight

- 函数

GPIO_PT2InputPullHight(BitSet);

- 函数功能

使能PT2 对应IO口的输入上拉电阻，设置寄存器PT2PU。

- 输入参数

BitSet [in]：代表GPIO PT2，输入值如下

PT2PU_PU20_ENABLE：使能PT2.0上拉电阻

PT2PU_PU21_ENABLE：使能PT2.1上拉电阻

PT2PU_PU22_ENABLE：使能PT2.2上拉电阻

PT2PU_PU23_ENABLE：使能PT2.3上拉电阻

PT2PU_PU24_ENABLE：使能PT2.4上拉电阻

PT2PU_PU25_ENABLE：使能PT2.5上拉电阻

PT2PU_PU26_ENABLE：使能PT2.6上拉电阻

PT2PU_PU27_ENABLE：使能PT2.7上拉电阻

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 使能 PT2.4 输入上拉电阻 */
GPIO_PT2InputPullHight( PT2PU_PU24_ENABLE );
```

5.3.37 GPIO_PT2InputPullHightClear

- 函数

GPIO_PT2InputPullHightClear(BitSet);

- 函数功能

关闭PT2对应IO pin的输入上拉电阻，设置寄存器PT2PU。

- 输入参数

BitSet [in]：代表GPIO port PT2输入值如下

PT2PU_PU20_DISABLE：关闭PT2.0上拉电阻

PT2PU_PU21_DISABLE：关闭PT2.1上拉电阻

PT2PU_PU22_DISABLE：关闭PT2.2上拉电阻

PT2PU_PU23_DISABLE：关闭PT2.3上拉电阻

PT2PU_PU24_DISABLE：关闭PT2.4上拉电阻

PT2PU_PU25_DISABLE：关闭PT2.5上拉电阻

PT2PU_PU26_DISABLE：关闭PT2.6上拉电阻

PT2PU_PU27_DISABLE：关闭PT2.7上拉电阻

- 包含头文件

Driver/GPIO.h

- **函数返回值**

无

- **函数用法**

```
/* 关闭 PT2.4 输入上拉电阻 */
GPIO_PT2InputPullHightClear( PT2PU_PU24_DISABLE );
```

5.3.38 GPIO_GetPT2Data

- **函数**

GPIO_GetPT2Data(PT2Data);

- **函数功能**

读取PT2输入状态，读取寄存器PT2。

- **输入参数**

PT2Data [in]：用于存放读取到的PT2输入状态值

- **包含头文件**

Driver/GPIO.h

- **函数返回值**

PT2Data：返回 PT2 输入状态值

- **函数用法**

```
/* 读取 PT2 输入状态值 */
unsigned char PT2_DATA;
GPIO_GetPT2Data( PT2_DATA );
```

5.3.39 GPIO_PM22Sel

- **函数**

GPIO_PM22Sel(PM22Sel);

- **函数功能**

设置PT2.2数字类复用功能，设置寄存器PT2M1[5:4]。

- **输入参数**

PM22Sel [in]：设置PT2.2复用功能

PT2M1_PM22_PFD ：PT2.2复用为PFD输出引脚

PT2M1_PM22_PWM0 ：PT2.2复用为PWM0输出引脚

PT2M1_PM22_GPIO ：PT2.2复用为GPIO

- **包含头文件**

Driver/GPIO.h

- **函数返回值**

无

- **函数用法**

```
/* 设置PT2.2复用为PWM0输出引脚 */
GPIO_PM22Sel( PT2M1_PM22_PWM0 );                    //PT2.2复用为PWM0输出引脚
```

5.3.40 GPIO_PM23Sel

- 函数

GPIO_PM23Sel(PM23Sel);

- 函数功能

设置PT2.3数字类复用功能，设置寄存器PT2M1[6]。

- 输入参数

PM23Sel [in]：设置PT2.3复用功能

PT2M1_PM23_PWM1：PT2.3复用为PWM1输出引脚

PT2M1_PM23_GPIO ：PT2.3复用为GPIO

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 设置PT2.3复用为PWM1输出引脚 */
GPIO_PM23Sel( PT2M1_PM23_PWM1 );                    //PT2.3复用为PWM1输出引脚
```

5.3.41 GPIO_PM24Sel

- 函数

GPIO_PM24Sel(PM24Sel);

- 函数功能

设置PT2.4数字类复用功能，设置寄存器PT2M2[1:0]。

- 输入参数

PM24Sel [in]：设置PT2.4复用功能

PT2M2_PM24_CCP0 ：PT2.4复用为CCP0引脚

PT2M2_PM24_PWM2：PT2.4复用为PWM2输出引脚

PT2M2_PM24_GPIO ：PT2.4复用为GPIO

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 设置PT2.4复用为PWM2输出引脚 */
GPIO_PM24Sel( PT2M2_PM24_PWM2 );                    //PT2.4复用为PWM2输出引脚
```

5.3.42 GPIO_PM25Sel

- 函数

GPIO_PM25Sel(PM25Sel);

- 函数功能

设置PT2.5数字类复用功能，设置寄存器PT2M2[3:2]。

- 输入参数

PM25Sel [in]：设置PT2.5复用功能

PT2M2_PM25_CCP1 ：PT2.5复用为CCP1引脚

PT2M2_PM25_PWM3：PT2.5复用为PWM3输出引脚

PT2M2_PM25_GPIO ：PT2.5复用为GPIO

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 设置PT2.5复用为PWM3输出引脚 */
GPIO_PM25Sel( PT2M2_PM25_PWM3 );                    //PT2.5复用为PWM3输出引脚
```

5.3.43 GPIO_PM26Sel

- 函数

GPIO_PM26Sel(PM26Sel);

- 函数功能

设置PT2.6数字类复用功能，设置寄存器PT2M2[4]。

- 输入参数

PM26Sel [in]：设置PT2.6复用功能

PT2M2_PM26_CPAO ：PT2.6复用为CPAO引脚

PT2M2_PM26_GPIO ：PT2.6复用为GPIO

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 设置PT2.6复用为CPAO输出引脚 */
GPIO_PM26Sel( PT2M2_PM26_CPAO );                    //PT2.6复用为CPAO输出引脚
```

5.3.44 GPIO_PM27Sel

- 函数

GPIO_PM27Sel(PM27Sel);

- 函数功能

设置PT2.7数字类复用功能，设置寄存器PT2M2[6]。

- 输入参数

PM27Sel [in]：设置PT2.7复用功能

PT2M2_PM27_CPAO ：PT2.7复用为CPAO引脚

PT2M2_PM27_GPIO ：PT2.7复用为GPIO

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 设置PT2.7复用为CPAO输出引脚 */
GPIO_PM27Sel( PT2M2_PM27_CPAO );                    //PT2.7复用为CPAO输出引脚
```

5.3.45 GPIO_PT3OutputMode

- 函数

GPIO_PT3OutputMode(BitSet);

- 函数功能

设置PT3作为输出模式，操作寄存器TRISC3。

- 输入参数

BitSet [in]：代表GPIO port PT3，设置参数如下

TRISC3_TC30_OUTPUT：PT3.0作为输出模式

TRISC3_TC31_OUTPUT：PT3.1作为输出模式

TRISC3_TC32_OUTPUT：PT3.2作为输出模式

TRISC3_TC33_OUTPUT：PT3.3作为输出模式

TRISC3_TC34_OUTPUT：PT3.4作为输出模式

TRISC3_TC35_OUTPUT：PT3.5作为输出模式

TRISC3_TC36_OUTPUT：PT3.6作为输出模式

TRISC3_TC37_OUTPUT：PT3.7作为输出模式

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 设置PT3.5/PT3.4的输出模式 */
GPIO_PT3OutputMode( TRISC3_TC34_OUTPUT );
GPIO_PT3OutputMode( TRISC3_TC35_OUTPUT );
//或改成以下用法
GPIO_PT3OutputMode( TRISC3_TC35_OUTPUT | TRISC3_TC34_OUTPUT );
```

5.3.46 GPIO_PT3OutputHigh

- 函数

GPIO_PT3OutputHigh(BitSet);

- 函数功能

设置PT3对应IO输出高电平，操作寄存器PT3[7:0]。

- 输入参数

BitSet [in]：代表GPIO port PT3，8bit数据bit0~bit7分别对应PT3.0~PT3.7，输入参数如下

PT3_PT30_H：PT3.0输出高电平

PT3_PT31_H：PT3.1输出高电平

PT3_PT32_H：PT3.2输出高电平

PT3_PT33_H：PT3.3输出高电平

PT3_PT34_H：PT3.4输出高电平

PT3_PT35_H：PT3.5输出高电平

PT3_PT36_H：PT3.6输出高电平

PT3_PT37_H：PT3.7输出高电平

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* PT3.4输出高电平 */
GPIO_PT3OutputMode( TRISC3_TC34_OUTPUT );          //设置PT3.4作为输出模式
GPIO_PT3OutputHigh( PT3_PT34_H );                  //设置PT3.4输出高电平
```

5.3.47 GPIO_PT3OutputLow

- 函数

GPIO_PT3OutputLow(BitSet);

- 函数功能

设置PT3对应IO口输出低电平，操作寄存器PT3[7:0]。

- 输入参数

BitSet [in]：代表GPIO port PT3，8bit数据bit0~bit7分别对应PT3.0~PT3.7，输入参数如下

PT3_PT30_L：PT3.0输出低电平

PT3_PT31_L：PT3.1输出低电平

PT3_PT32_L：PT3.2输出低电平

PT3_PT33_L：PT3.3输出低电平

PT3_PT34_L：PT3.4输出低电平

PT3_PT35_L：PT3.5输出低电平

PT3_PT36_L：PT3.6输出低电平

PT3_PT37_L：PT3.7输出低电平

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* PT3.4输出低电平 */
GPIO_PT3OutputMode( TRISC3_TC34_OUTPUT );          //设置PT3.4作为输出模式
GPIO_PT3OutputHigh( PT3_PT34_L );                  //设置PT3.4输出低电平
```

5.3.48 GPIO_PT3InputMode

- 函数

GPIO_PT3InputMode(BitSet);

- 函数功能

设置PT3对应IO口作为输入模式，设置寄存器TRISC3。

- 输入参数

BitSet [in]：代表PT3对应IO pin，针对PT3.0~PT3.7设置，输入参数如下

TRISC3_TC30_INPUT：PT3.0作为输入模式
TRISC3_TC31_INPUT：PT3.1作为输入模式
TRISC3_TC32_INPUT：PT3.2作为输入模式
TRISC3_TC33_INPUT：PT3.3作为输入模式
TRISC3_TC34_INPUT：PT3.4作为输入模式
TRISC3_TC35_INPUT：PT3.5作为输入模式
TRISC3_TC36_INPUT：PT3.6作为输入模式
TRISC3_TC37_INPUT：PT3.7作为输入模式

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 设置PT3.4作为输入口 */
GPIO_PT3InputMode( TRISC3_TC34_INPUT );
```

5.3.49 GPIO_PT3InputPullHight

- 函数

GPIO_PT3InputPullHight(BitSet);

- 函数功能

使能PT3 对应IO口的输入上拉电阻，设置寄存器PT3PU。

- 输入参数

BitSet [in]：代表GPIO PT3，输入值如下

PT3PU_PU30_ENABLE：使能PT3.0上拉电阻

PT3PU_PU31_ENABLE：使能PT3.1上拉电阻

PT3PU_PU32_ENABLE：使能PT3.2上拉电阻

PT3PU_PU33_ENABLE：使能PT3.3上拉电阻

PT3PU_PU34_ENABLE：使能PT3.4上拉电阻

PT3PU_PU35_ENABLE：使能PT3.5上拉电阻

PT3PU_PU36_ENABLE：使能PT3.6上拉电阻

PT3PU_PU37_ENABLE：使能PT3.7上拉电阻

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/*  使能 PT3.4 输入上拉电阻* /
GPIO_PT3InputPullHight( PT3PU_PU34_ENABLE );
```

5.3.50 GPIO_PT3InputPullHightClear

- 函数

GPIO_PT3InputPullHightClear(BitSet) ;

- 函数功能

关闭PT3对应IO pin的输入上拉电阻，设置寄存器PT3PU。

- 输入参数

BitSet [in]：代表GPIO port PT3输入值如下

PT3PU_PU30_DISABLE：关闭PT3.0上拉电阻

PT3PU_PU31_DISABLE：关闭PT3.1上拉电阻

PT3PU_PU32_DISABLE：关闭PT3.2上拉电阻

PT3PU_PU33_DISABLE：关闭PT3.3上拉电阻

PT3PU_PU34_DISABLE：关闭PT3.4上拉电阻

PT3PU_PU35_DISABLE：关闭PT3.5上拉电阻

PT3PU_PU36_DISABLE：关闭PT3.6上拉电阻

PT3PU_PU37_DISABLE：关闭PT3.7上拉电阻

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
 /*  关闭 PT3.4 输入上拉电阻  */
GPIO_PT3InputPullHightClear( PT3PU_PU34_DISABLE );
```

5.3.51 GPIO_GetPT3Data

- 函数

  GPIO_GetPT3Data(PT3Data);

- 函数功能

  读取PT3输入状态，读取寄存器PT3。

- 输入参数

  PT3Data [in]：用于存放读取到的PT3输入状态值

- 包含头文件

  Driver/GPIO.h

- 函数返回值

  PT3Data：返回 PT3 输入状态值

- 函数用法

```
/* 读取 PT3 输入状态值 */
unsigned char PT3_DATA;
GPIO_GetPT3Data( PT3_DATA );
```

5.3.52 GPIO_PT4InputPullHight

- 函数

  GPIO_PT4InputPullHight(BitSet);

- 函数功能

  使能PT4 对应IO口的输入上拉电阻，设置寄存器PT4PU。

- 输入参数

  BitSet [in]：代表GPIO PT4，输入值如下

  PT4PU_PU40_ENABLE：使能PT4.0上拉电阻

  PT4PU_PU41_ENABLE：使能PT4.1上拉电阻

  PT4PU _PU42_ENABLE：使能PT4.2上拉电阻

  PT4PU_PU43_ENABLE：使能PT4.3上拉电阻

  PT4PU _PU44_ENABLE：使能PT4.4上拉电阻

  PT4PU_PU45_ENABLE：使能PT4.5上拉电阻

  PT4PU _PU46_ENABLE：使能PT4.6上拉电阻

  PT4PU_PU47_ENABLE：使能PT4.7上拉电阻

- 包含头文件

  Driver/GPIO.h

- 函数返回值

  无

- 函数用法

```
/* 使能 PT4.4 输入上拉电阻 */
GPIO_PT4InputPullHight( PT4PU _PU44_ENABLE );
```

5.3.53 GPIO_PT4InputPullHightClear

- 函数

GPIO_PT4InputPullHightClear(BitSet) ;

- 函数功能

关闭PT4对应IO pin的输入上拉电阻，设置寄存器PT4PU。

- 输入参数

BitSet [in]：代表GPIO port PT4输入值如下

PT4PU_PU40_DISABLE：关闭PT4.0上拉电阻

PT4PU_PU41_DISABLE：关闭PT4.1上拉电阻

PT4PU_PU42_DISABLE：关闭PT4.2上拉电阻

PT4PU_PU43_DISABLE：关闭PT4.3上拉电阻

PT4PU_PU44_DISABLE：关闭PT4.4上拉电阻;

PT4PU_PU45_DISABLE：关闭PT4.5上拉电阻

PT4PU_PU46_DISABLE：关闭PT4.6上拉电阻

PT4PU_PU47_DISABLE：关闭PT4.7上拉电阻

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 关闭 PT4.4 输入上拉电阻 */
GPIO_PT4InputPullHightClear( PT4PU_PU44_DISABLE );
```

5.3.54 GPIO_GetPT4Data

- 函数

GPIO_GetPT4Data(PT4Data);

- 函数功能

读取PT4输入状态，读取寄存器PT4。

- 输入参数

PT4Data [in]：用于存放读取到的PT4输入状态值

- 包含头文件

Driver/GPIO.h

- 函数返回值

PT4Data：返回 PT4 输入状态值

- 函数用法

```
/* 读取 PT4 输入状态值 */
unsigned char PT4_DATA;
GPIO_GetPT4Data( PT4_DATA );
```

5.3.55 GPIO_PT4AnalogMode

- 函数

GPIO_PT4AnalogMode(BitSet);

- **函数功能**

  使能PT4模拟类功能，设置寄存器PT4DA[7:0]。

- **输入参数**

  BitSet [in]：代表GPIO port PT4输入值如下

  PT4DA_DA40_AI0：PT4.0复用为ADC输入口AI0

  PT4DA_DA41_AI1：PT4.1复用为ADC输入口AI1

  PT4DA_DA42_AI2：PT4.2复用为ADC输入口AI2

  PT4DA_DA43_AI3：PT4.3复用为ADC输入口AI3

  PT4DA_DA44_AI4：PT4.4复用为ADC输入口AI4

  PT4DA_DA45_AI5：PT4.5复用为ADC输入口AI5

  PT4DA_DA46_AI6：PT4.6复用为ADC输入口AI6

  PT4DA_DA47_AI7：PT4.7复用为ADC输入口AI7

- **包含头文件**

  Driver/GPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 使能PT4.2的模拟类功能 */

  GPIO_PT4AnalogMode(PT4DA_DA42_AI2);　　//使能PT4.2模拟类功能

5.3.56 GPIO_PT4DigitalMode

- **函数**

  GPIO_PT4DigitalMode(BitSet);

- **函数功能**

  使能PT4数字类功能，设置寄存器PT4DA[7:0]。

- **输入参数**

  BitSet [in]：代表GPIO port PT4，输入值如下

  PT4DA_DA40_DIG：使能PT4.0数字类功能

  PT4DA_DA41_DIG：使能PT4.1数字类功能

  PT4DA_DA42_DIG：使能PT4.2数字类功能

  PT4DA_DA43_DIG：使能PT4.3数字类功能

  PT4DA_DA44_DIG：使能PT4.4数字类功能

  PT4DA_DA45_DIG：使能PT4.5数字类功能

  PT4DA_DA46_DIG：使能PT4.6数字类功能

  PT4DA_DA47_DIG：使能PT4.7数字类功能

- **包含头文件**

  Driver/GPIO.h

- **函数返回值**

无

- 函数用法

```
/* 使能PT4.2数字类功能 */
GPIO_PT4DigitalMode( PT4DA_DA42_DIG );                    //使能PT4.2数字类功能
```

5.3.57 GPIO_PT5InputPullHight

- 函数

GPIO_PT5InputPullHight(BitSet);

- 函数功能

使能PT5 对应IO口的输入上拉电阻，设置寄存器PT5PU。

- 输入参数

BitSet [in]：代表GPIO PT5，输入值如下

PT5PU_PU50_ENABLE：使能PT5.0上拉电阻

PT5PU_PU51_ENABLE：使能PT5.1上拉电阻

PT5PU_PU52_ENABLE：使能PT5.2上拉电阻

PT5PU_PU53_ENABLE：使能PT5.3上拉电阻

PT5PU_PU54_ENABLE：使能PT5.4上拉电阻

PT5PU_PU55_ENABLE：使能PT5.5上拉电阻

PT5PU_PU56_ENABLE：使能PT5.6上拉电阻

PT5PU_PU57_ENABLE：使能PT5.7上拉电阻

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 使能 PT5.4 输入上拉电阻 */
GPIO_PT5InputPullHight( PT5PU _PU54_ENABLE );
```

5.3.58 GPIO_PT5InputPullHightClear

- 函数

GPIO_PT5InputPullHightClear(BitSet) ;

- 函数功能

关闭PT5对应IO pin的输入上拉电阻，设置寄存器PT5PU。

- 输入参数

BitSet [in]：代表GPIO port PT5输入值如下

PT5PU_PU50_DISABLE：关闭PT5.0上拉电阻

PT5PU_PU51_DISABLE：关闭PT5.1上拉电阻

PT5PU_PU52_DISABLE：关闭PT5.2上拉电阻

PT5PU_PU53_DISABLE：关闭PT5.3上拉电阻

PT5PU_PU54_DISABLE：关闭PT5.4上拉电阻

PT5PU_PU55_DISABLE：关闭PT5.5上拉电阻

PT5PU_PU56_DISABLE：关闭PT5.6上拉电阻

PT5PU_PU57_DISABLE：关闭PT5.7上拉电阻

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 关闭 PT5.4 输入上拉电阻 */
GPIO_PT5InputPullHightClear( PT5PU_PU54_DISABLE );
```

5.3.59 GPIO_GetPT5Data

- 函数

GPIO_GetPT5Data(PT5Data);

- 函数功能

读取PT5输入状态，读取寄存器PT5。

- 输入参数

PT5Data [in]：用于存放读取到的PT5输入状态值

- 包含头文件

Driver/GPIO.h

- 函数返回值

PT5Data：返回 PT5 输入状态值

- 函数用法

```
/* 读取 PT5 输入状态值 */
unsigned char PT5_DATA;
GPIO_GetPT5Data( PT5_DATA );
```

5.3.60 GPIO_PT5AnalogMode

- 函数

GPIO_PT5AnalogMode(BitSet);

- 函数功能

使能PT5模拟类功能，设置寄存器PT5DA[3:0]。

- 输入参数

BitSet [in]：代表GPIO port PT5.0~PT5.3输入值如下

PT5DA_DA50_AI8：PT5.0复用为ADC输入口AI8

PT5DA_DA51_AI9：PT5.1复用为ADC输入口AI9

PT5DA_DA52_AI10：PT5.2复用为ADC输入口AI10

PT5DA_DA53_AI11：PT5.3复用为ADC输入口AI11

- 包含头文件

Driver/GPIO.h

- 函数返回值

  无

- 函数用法

```
/* 使能PT5.2的模拟类功能 */
GPIO_PT5AnalogMode( PT5DA_DA52_AI10 );                    //使能PT5.2模拟类功能
```

5.3.61 GPIO_PT5DigitalMode

- 函数

  GPIO_PT5DigitalMode(BitSet);

- 函数功能

  使能PT5数字类功能，设置寄存器PT5DA[3:0]。

- 输入参数

  BitSet [in]：代表GPIO port PT5，输入值如下

  PT5DA_DA50_DIG：使能PT5.0数字类功能

  PT5DA_DA51_DIG：使能PT5.1数字类功能

  PT5DA_DA52_DIG：使能PT5.2数字类功能

  PT5DA_DA53_DIG：使能PT5.3数字类功能

- 包含头文件

  Driver/GPIO.h

- 函数返回值

  无

- 函数用法

```
/* 使能PT5.2数字类功能 */
GPIO_PT5DigitalMode( PT5DA_DA52_DIG );                    //使能PT5.2数字类功能
```

5.3.62 GPIO_INT0_Enable

- 函数

  GPIO_INT0_Enable();

- 函数功能

  使能PT1.0外部中断功能，设置寄存器INTE1[0]=1 。

- 输入参数

  无.

- 包含头文件

  Driver/INT.h

- 函数返回值

  无

- 函数用法

```
/* 使能PT1.0外部中断功能 */
```

GPIO_INT0_Enable();

5.3.63 GPIO_INT0_Disable

- 函数

GPIO_INT0_Disable();

- 函数功能

关闭PT1.0外部中断功能，设置寄存器INTE1[0]=0 。

- 输入参数

无

- 包含头文件

Driver/INT.h

- 函数返回值

无

- 函数用法

/* 关闭PT1.0外部中断功能 */

GPIO_INT0_Disable();

5.3.64 GPIO_INT1_Enable

- 函数

GPIO_INT1_Enable();

- 函数功能

使能PT1.1外部中断功能，设置寄存器INTE1[1]=1 。

- 输入参数

无

- 包含头文件

Driver/INT.h

- 函数返回值

无

- 函数用法

/* 使能PT1.1外部中断功能 */

GPIO_INT1_Enable();

5.3.65 GPIO_INT1_Disable

- 函数

GPIO_INT1_Disable();

- 函数功能

关闭PT1.1外部中断功能，设置寄存器INTE1[1]=0 。

- **输入参数**

  无

- **包含头文件**

  Driver/INT.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭PT1.1外部中断功能 */

  GPIO_INT1_Disable();

5.3.66 GPIO_INT_TYPE_PT10

- **函数**

  GPIO_INT_TYPE_PT10(PT10INTType);

- **函数功能**

  设置PT1.0外部中断触发沿，设置寄存器PT1M1[1:0]。

- **输入参数**

  PT10INTType [in]：PT1.0外部中断触发沿选择

  PT1M1_INTEG0_LEV_ ：电平变化触发

  PT1M1_INTEG0_LEV ：电平变化触发

  PT1M1_INTEG0_EDGERISE ：上升沿触发

  PT1M1_INTEG0_EDGEFALL ：下降沿触发

- **包含头文件**

  Driver/GPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 设置PT1.0外部中断触发沿为下降沿触发 */

  GPIO_INT_TYPE_PT10(PT1M1_INTEG0_EDGEFALL);

5.3.67 GPIO_INT_TYPE_PT11

- **函数**

  GPIO_INT_TYPE_PT11(PT11INTType);

- **函数功能**

  设置PT1.1外部中断触发沿，设置寄存器PT1M1[3:2]。

- **输入参数**

  PT11INTType [in]：PT1.1外部中断触发沿选择

  PT1M1_INTEG1_LEV_ ：电平变化触发

  PT1M1_INTEG1_LEV ：电平变化触发

PT1M1_INTEG1_EDGERISE：上升沿触发

PT1M1_INTEG1_EDGEFALL：下降沿触发

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 设置PT1.1外部中断触发沿为下降沿触发 */
GPIO_INT_TYPE_PT11( PT1M1_INTEG1_EDGEFALL );
```

5.3.68 GPIO_INT_Low2BitEnable

- 函数

GPIO_INT_Low2BitEnable(L2BSet);

- 函数功能

使能PT1.1/PT1.0外部中断功能，设置寄存器INTE1[1:0]。

- 输入参数

L2Bset [in]：使能PT1.0/PT1.1外部中断

INTE1_E0IE_ENABLE：使能PT1.0外部中断

INTE1_E1IE_ENABLE：使能PT1.1外部中断

- 包含头文件

Driver/INT.h

- 函数返回值

无

- 函数用法

```
/* 使能PT1.1/PT1.0外部中断功能 */
GPIO_INT_Low2BitEnable( INTE1_E0IE_ENABLE | INTE1_E1IE_ENABLE );
```

5.3.69 GPIO_AI0Enable

- 函数

GPIO_AI0Enable();

- 函数功能

使能PT4.0（AI0）模拟功能，设置PT4DA[0]。

- 输入参数

无

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 使能PT4.0模拟功能 */
GPIO_AI0Enable();
```

5.3.70 GPIO_AI1Enable

- 函数

GPIO_AI1Enable();

- 函数功能

使能PT4.1（AI1）模拟功能，设置PT4DA[1]。

- 输入参数

无

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 使能PT4.1模拟功能 */
GPIO_AI1Enable();
```

5.3.71 GPIO_AI2Enable

- 函数

GPIO_AI2Enable();

- 函数功能

使能PT4.2（AI2）模拟功能，设置PT4DA[2]。

- 输入参数

无

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 使能PT4.2模拟功能 */
GPIO_AI2Enable();
```

5.3.72 GPIO_AI3Enable

- 函数

GPIO_AI3Enable();

- 函数功能

使能PT4.3（AI3）模拟功能，设置PT4DA[3]。

- 输入参数

无

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 使能PT4.3模拟功能 */
GPIO_AI3Enable();
```

5.3.73 GPIO_AI4Enable

- 函数

GPIO_AI4Enable();

- 函数功能

使能PT4.4（AI4）模拟功能，设置PT4DA[4]。

- 输入参数

无

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 使能PT4.4模拟功能 */
GPIO_AI4Enable();
```

5.3.74 GPIO_AI5Enable

- 函数

GPIO_AI5Enable();

- 函数功能

使能PT4.5（AI5）模拟功能，设置PT4DA[5]。

- 输入参数

无

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 使能PT4.5模拟功能 */
GPIO_AI5Enable();
```

5.3.75 GPIO_AI6Enable

- 函数

GPIO_AI6Enable();

- 函数功能

使能PT4.6（AI6）模拟功能，设置PT4DA[6]。

- 输入参数

无

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 使能PT4.6模拟功能 */
GPIO_AI6Enable();
```

5.3.76 GPIO_AI7Enable

- 函数

GPIO_AI7Enable();

- 函数功能

使能PT4.7（AI7）模拟功能，设置PT4DA[7]。

- 输入参数

无

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 使能PT4.7模拟功能 */
GPIO_AI7Enable();
```

5.3.77 GPIO_AI8Enable

- 函数

GPIO_AI8Enable();

- 函数功能

使能PT5.0（AI8）模拟功能，设置PT5DA[0]。

- 输入参数

无

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 使能PT5.0模拟功能 */
GPIO_AI8Enable();
```

5.3.78 GPIO_AI9Enable

- 函数

GPIO_AI9Enable();

- 函数功能

使能PT5.1（AI9）模拟功能，设置PT5DA[1]。

- 输入参数

无

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 使能PT5.1模拟功能 */
GPIO_AI9Enable();
```

5.3.79 GPIO_AI10Enable

- 函数

GPIO_AI10Enable();

- 函数功能

使能PT5.2（AI10）模拟功能，设置PT5DA[2]。

- 输入参数

无

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- **函数用法**

```
/* 使能PT5.2模拟功能 */
GPIO_AI10Enable();
```

5.3.80 GPIO_AI8Enable

- **函数**

GPIO_AI11Enable();

- **函数功能**

使能PT5.3（AI11）模拟功能，设置PT5DA[3]。

- **输入参数**

无

- **包含头文件**

Driver/GPIO.h

- **函数返回值**

无

- **函数用法**

```
/* 使能PT5.3模拟功能 */
GPIO_AI11Enable();
```

5.3.81 GPIO_CPAI0Enable

- **函数**

GPIO_CPAI0Enable();

- **函数功能**

使能PT2.2复用为增强型比较器信号输入端CPAI0，设置PT2M1[5:4]。

- **输入参数**

无

- **包含头文件**

Driver/GPIO.h

- **函数返回值**

无

- **函数用法**

```
/* 使能PT2.2复用为CPAI0引脚 */
GPIO_CPAI0Enable();
```

5.3.82 GPIO_CPAI0Disable

- **函数**

GPIO_CPAI0Disable();

- **函数功能**

关闭PT2.2复用为增强型比较器信号输入端CPAI0功能，设置PT2M1[5:4]。

- **输入参数**

无

- **包含头文件**

Driver/GPIO.h

- **函数返回值**

无

- **函数用法**

```
/* 关闭PT2.2复用为CPAI0引脚 */
GPIO_CPAI0Disable();
```

5.3.83 GPIO_CPAI1Enable

- **函数**

GPIO_CPAI1Enable();

- **函数功能**

使能PT2.3复用为增强型比较器信号输入端CPAI1，设置PT2M1[6]。

- **输入参数**

无

- **包含头文件**

Driver/GPIO.h

- **函数返回值**

无

- **函数用法**

```
/* 使能PT2.3复用为CPAI1引脚 */
GPIO_CPAI1Enable();
```

5.3.84 GPIO_CPAI1Disable

- **函数**

GPIO_CPAI1Disable();

- **函数功能**

关闭PT2.3复用为增强型比较器信号输入端CPAI1功能，设置PT2M1[6]。

- **输入参数**

无

- **包含头文件**

Driver/GPIO.h

- **函数返回值**

无

- 函数用法

```
/* 关闭PT2.3复用为CPAI1引脚 */
GPIO_CPAI1Disable();
```

5.3.85 GPIO_CPAI2Enable

- 函数

GPIO_CPAI2Enable();

- 函数功能

使能PT2.4复用为增强型比较器信号输入端CPAI2，设置PT2M2[1:0]。

- 输入参数

无

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 使能PT2.4复用为CPAI2引脚 */
GPIO_CPAI2Enable();
```

5.3.86 GPIO_CPAI2Disable

- 函数

GPIO_CPAI2Disable();

- 函数功能

关闭PT2.4复用为增强型比较器信号输入端CPAI2功能，设置PT2M2[1:0]。

- 输入参数

无

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 关闭PT2.4复用为CPAI2引脚 */
GPIO_CPAI2Disable();
```

5.3.87 GPIO_CPAI3Enable

- 函数

GPIO_CPAI3Enable();

- **函数功能**

  使能PT2.5复用为增强型比较器信号输入端CPAI3，设置PT2M2[3:2]。

- **输入参数**

  无

- **包含头文件**

  Driver/GPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 使能PT2.5复用为CPAI3引脚 */

  GPIO_CPAI3Enable();

5.3.88 GPIO_CPAI3Disable

- **函数**

  GPIO_CPAI3Disable();

- **函数功能**

  关闭PT2.5复用为增强型比较器信号输入端CPAI3功能，设置PT2M2[3:2]。

- **输入参数**

  无

- **包含头文件**

  Driver/GPIO.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭PT2.5复用为CPAI3引脚 */

  GPIO_CPAI3Disable();

5.3.89 GPIO_CPAI4Enable

- **函数**

  GPIO_CPAI4Enable();

- **函数功能**

  使能PT2.6复用为增强型比较器信号输入端CPAI4，设置PT2M2[4]。

- **输入参数**

  无

- **包含头文件**

  Driver/GPIO.h

- **函数返回值**

  无

- 函数用法

```
/* 使能PT2.6复用为CPAI4引脚 */
GPIO_CPAI4Enable();
```

5.3.90 GPIO_CPAI4Disable

- 函数

GPIO_CPAI4Disable();

- 函数功能

关闭PT2.6复用为增强型比较器信号输入端CPAI4功能，设置PT2M2[4]。

- 输入参数

无

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 关闭PT2.6复用为CPAI4引脚 */
GPIO_CPAI4Disable();
```

5.3.91 GPIO_CPAI5Enable

- 函数

GPIO_CPAI5Enable();

- 函数功能

使能PT2.7复用为增强型比较器信号输入端CPAI5，设置PT2M2[6]。

- 输入参数

无

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 使能PT2.7复用为CPAI5引脚 */
GPIO_CPAI5Enable();
```

5.3.92 GPIO_CPAI5Disable

- 函数

GPIO_CPAI5Disable();

- **函数功能**

  关闭PT2.7复用为增强型比较器信号输入端CPAI5功能，设置PT2M2[6]。

- **输入参数**

  无

- **包含头文件**

  Driver/GPIO.h

- **函数返回值**

  无

- **函数用法**

```
/* 关闭PT2.7复用为CPAI5引脚 */
GPIO_CPAI5Disable();
```

5.3.93 GPIO_CPAI6Enable

- **函数**

  GPIO_CPAI6Enable();

- **函数功能**

  使能PT1.0复用为增强型比较器信号输入端CPAI6，设置PT1DA[0]。

- **输入参数**

  无

- **包含头文件**

  Driver/GPIO.h

- **函数返回值**

  无

- **函数用法**

```
/* 使能PT1.0复用为CPAI6引脚 */
GPIO_CPAI6Enable();
```

5.3.94 GPIO_CPAI6Disable

- **函数**

  GPIO_CPAI6Disable();

- **函数功能**

  关闭PT1.0复用为增强型比较器信号输入端CPAI6功能，设置PT1DA[0]。

- **输入参数**

  无

- **包含头文件**

  Driver/GPIO.h

- **函数返回值**

  无

- 函数用法

```
/* 关闭PT1.0复用为CPAI6引脚 */
GPIO_CPAI6Disable();
```

5.3.95 GPIO_CPAI7Enable

- 函数

GPIO_CPAI7Enable();

- 函数功能

使能PT1.1复用为增强型比较器信号输入端CPAI7，设置PT1DA[1]。

- 输入参数

无

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 使能PT1.1复用为CPAI7引脚 */
GPIO_CPAI7Enable();
```

5.3.96 GPIO_CPAI7Disable

- 函数

GPIO_CPAI7Disable();

- 函数功能

关闭PT1.1复用为增强型比较器信号输入端CPAI7功能，设置PT1DA[1]。

- 输入参数

无

- 包含头文件

Driver/GPIO.h

- 函数返回值

无

- 函数用法

```
/* 关闭PT1.1复用为CPAI7引脚 */
GPIO_CPAI7Disable();
```

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

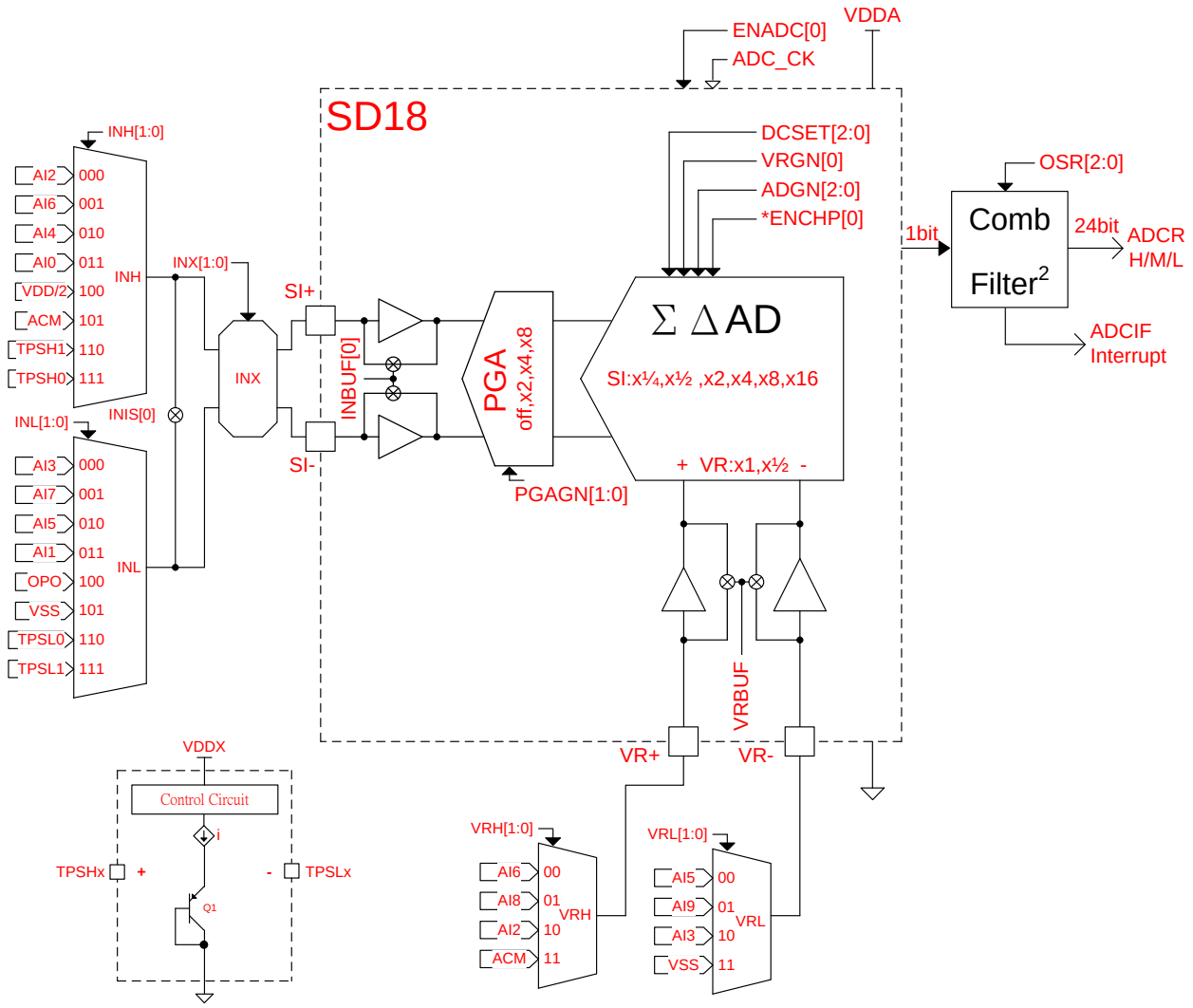
--ADC中断配置

--ADC转换值的读取

--包含ADC.h

序号	函数名称	功能描述
01	ADC_Open	启动ADC，并设置ADC的工作频率、信号输入、参考电压输入端、放大倍数、数据输出速率、直流偏置电压
02	ADC_GetData	读取ADC转换值
03	ADC_CLK_Enable	设置ADC工作频率及预分频值
04	ADC_CLK_Disable	关闭ADC工作频率
05	ADC_INT_Enable	使能ADC中断功能
06	ADC_INT_Disable	关闭ADC中断功能
07	ADC_INT_IsFlag	读取ADC中断请求标志位
08	ADC_INT_ClearFlag	清除ADC中断请求标志位
09	ADC_INSEnable	使能ADC信号输入端内部短路功能
10	ADC_INSDisable	关闭ADC信号输入端内部短路功能
11	ADC_Enable	启动ADC转换功能
12	ADC_Disable	关闭ADC转换功能
13	ADC_VRbufEnable	使能ADC的参考电压输入端的输入缓冲器功能
14	ADC_VRbufDisable	关闭ADC参考电压输入端输入缓冲器功能
15	ADC_INbufEnable	使能ADC信号输入端的输入缓冲器功能
16	ADC_INbufDisable	关闭ADC信号输入端的输入缓冲器功能
17	ADC_AINConfig	设置ADC信号输入端的来源
18	ADC_VRINConfig	设置ADC参考电压输入端来源
19	ADC_VRGainSelect	设置ADC参考电压输入端放大倍数
20	ADC_GainConfig	设置ADC信号输入端放大倍数ADGN * PGA
21	ADC_OSRConfig	设置ADC转换输出速率，输出速率为ADC_CK /OSR
22	ADC_DCSetConfig	设置ADC信号输入端的直流偏置电压
23	ADC_INXConfig	设置ADC信号输入端的输入信号转置器

6.2 ADC 模块方框图



6.3 函数说明

6.3.1 ADC_Open

- **函数**

void ADC_Open(unsigned char ck, unsigned char cks, unsigned char inh,
unsigned char inl, unsigned char vrh, unsigned char vrl,
unsigned char adgn, unsigned char pgagn, unsigned char vrgn,
unsigned char dcset, unsigned char osr);

- **函数功能**

启动ADC，并设置ADC的工作频率、信号输入、参考电压输入端、放大倍数、数据输出速率、直流偏置电压等，设置寄存器MCKCN1[7:4]/ADCCN1/ADCCN2/ADCCN3/AINET1/AINET2。

- **输入参数**

ck [in]：设置ADC工作频率源

MCKCN1_ADCCK_LSCK ：ADC工作频率源为LS_CK

MCKCN1_ADCCK_HSDCK ：ADC工作频率源为HSD_CK

cks [in]：设置ADC工作频率源预分频

MCKCN1_ADCS_DIV128 ：ADC工作频率预分频ADC_CK/128

MCKCN1_ADCS_DIV64 ：ADC工作频率预分频ADC_CK/64

MCKCN1_ADCS_DIV32 ：ADC工作频率预分频ADC_CK/32

MCKCN1_ADCS_DIV16 ：ADC工作频率预分频ADC_CK/16

MCKCN1_ADCS_DIV8 ：ADC工作频率预分频ADC_CK/8

MCKCN1_ADCS_DIV4 ：ADC工作频率预分频ADC_CK/4

MCKCN1_ADCS_DIV2 ：ADC工作频率预分频ADC_CK/2

MCKCN1_ADCS_DIV1 ：ADC工作频率预分频ADC_CK/1

inh [in]：设置ADC信号输入端的正向输入通道

AINET1_INH_AI2 ：AI2

AINET1_INH_AI6 ：AI6

AINET1_INH_AI4 ：AI4

AINET1_INH_AI0 ：AI0

AINET1_INH_HALF_VDDA ：VDDA/2

AINET1_INH_ADM ：ACM

AINET1_INH_TPSH1 ：TPSH1

AINET1_INH_TPSH0 ：TPSH0

inl [in]：设置ADC信号输入端的负向输入通道

AINET1_INL_AI3 ：AI3

AINET1_INL_AI7 ：AI7

AINET1_INL_AI5 ：AI5

AINET1_INL_AI1 ：AI1

AINET1_INL_OPO ：OPO

AINET1_INL_VSS ：VSS

AINET1_INL_TPSL0 ：TPSL0

AINET1_INL_TPSL1 ：TPSL1

vrh [in]：设置ADC参考电压输入端的正向输入通道

AINET2_VRH_AI6 ：AI6

AINET2_VRH_AI8 ：AI8

AINET2_VRH_AI2 ：AI2

AINET2_VRH_ACM ：ACM

vrl [in]：设置ADC参考电压输入端的负向输入通道

AINET2_VRL_AI5 ：AI5

AINET2_VRL_AI9 ：AI9

AINET2_VRL_AI3 ： AI3

AINET2_VRL_VSS：VSS

adgn [in]：设置ADC的ADGN放大倍数

ADCCN1_ADGN_1DIV4 ： x 1/4

ADCCN1_ADGN_1DIV2 ： x 1/2

ADCCN1_ADGN_1 ： x 1

ADCCN1_ADGN_2 ： x 2

ADCCN1_ADGN_4 ： x 4

ADCCN1_ADGN_8 ： x 8

ADCCN1_ADGN_16 ： x 16

pgagn [in]：设置ADC的PGA放大倍数

ADCCN1_PGAGN_1：x 1

ADCCN1_PGAGN_2：x 2

ADCCN1_PGAGN_4：x 4

ADCCN1_PGAGN_8：x 8

vrgn [in]：设置参考电压端放大倍数

ADCCN2_VREGN_DIV2：VREF * 1/2

ADCCN2_VREGN_DIV1：VREF* 1

dcset [in]：设置ADC的信号输入端直流偏置电压

ADCCN2_DCSET_0 ：不偏压

ADCCN2_DCSET_P1DIV4 ： + VREF* 1/4

ADCCN2_DCSET_P1DIV2 ： + VREF* 1/2

ADCCN2_DCSET_P3DIV4 ： + VREF* 3/4

ADCCN2_DCSET_00 ：不偏压

ADCCN2_DCSET_N1DIV4 ： -VREF* 1/4

ADCCN2_DCSET_N1DIV2 ： -VREF* 1/2

ADCCN2_DCSET_N3DIV4 ： -VREF* 3/4

osr [in]：设置ADC转换输出速率控制值

ADCCN3_OSR_256 ： 转换输出速率为ADC_CK/256

ADCCN3_OSR_512 ： 转换输出速率为ADC_CK/512

ADCCN3_OSR_1024 ： 转换输出速率为ADC_CK/1024

ADCCN3_OSR_2048 ： 转换输出速率为ADC_CK/2048

ADCCN3_OSR_4096 ： 转换输出速率为ADC_CK/4096

ADCCN3_OSR_8192 ： 转换输出速率为ADC_CK/8192

ADCCN3_OSR_16384 ： 转换输出速率为ADC_CK/16384

ADCCN3_OSR_32768 ： 转换输出速率为ADC_CK/32768

- **包含头文件**

Drirver/ADC.h

- **函数返回值**

无

- **函数用法**

/* 设置ADC工作频率为HSD_CK/4，信号输入端为AI0---AI1，参考电压输入端为AI2—AI3，参考电压衰减倍数1/2，ADC放大倍数为16*8，偏置电压为0，OSR为32768 */

```
ADC_Open( MCKCN1_ADCCK_HSDCK, MCKCN1_ADCS_DIV4,
          AINET1_INH_AI0, AINET1_INH_AI1, AINET2_VRH_AI2, AINET2_VRH_AI3,
          ADCCN1_ADGN_16, ADCCN1_PGAGN_8,
          ADCCN2_VREGN_DIV2, ADCCN2_DCSET_0,
          ADCCN3_OSR_32768 );
```

6.3.2 ADC_GetData

- **函数**

long ADC_GetData(void);

- **函数功能**

读取ADC转换值，读取寄存器ADCRH:ADCRM:ADCRL

- **输入参数**

无

- **包含头文件**

Drirver/ADC.h

- **函数返回值**

24位的ADC转换值

- **函数用法**

```
/* 读取ADC转换值 */
long ADC_DATA;
ADC_DATA = ADC_GetData();
```

6.3.3 ADC_CLK_Enable

- **函数**

ADC_CLK_Enable(adcck,adcs);

- **函数功能**

设置ADC工作频率及预分频值，设置寄存器MCKCN1[7:4]。

- **输入参数**

adcck [in]：设置ADC工作频率源

MCKCN1_ADCCK_LSCK ：ADC工作频率源为LS_CK

MCKCN1_ADCCK_HSDCK ：ADC工作频率源为HSD_CK

adcs [in]：设置ADC工作频率预分频值

MCKCN1_ADCS_DIV128 ：ADC工作频率预分频ADC_CK/128

MCKCN1_ADCS_DIV64 ：ADC工作频率预分频ADC_CK/64

MCKCN1_ADCS_DIV32 ：ADC工作频率预分频ADC_CK/32
MCKCN1_ADCS_DIV16 ：ADC工作频率预分频ADC_CK/16
MCKCN1_ADCS_DIV8 ：ADC工作频率预分频ADC_CK/8
MCKCN1_ADCS_DIV4 ：ADC工作频率预分频ADC_CK/4
MCKCN1_ADCS_DIV2 ：ADC工作频率预分频ADC_CK/2
MCKCN1_ADCS_DIV1 ：ADC工作频率预分频ADC_CK/1

- **包含头文件**

Drirver/ADC.h

- **函数返回值**

无

- **函数用法**

```
/* 设定ADC工作频率为HSD_CK/4 */
ADC_CLK_Enable( MCKCN1_ADCCK_HSDCK, MCKCN1_ADCS_DIV4 );
```

6.3.4 ADC_CLK_Disable

- **函数**

ADC_CLK_Disable();

- **函数功能**

关闭ADC工作频率，设置寄存器MCKCN1[4]。

- **输入参数**

无

- **包含头文件**

Drirver/ADC.h

- **函数返回值**

无

- **函数用法**

```
/* 关闭ADC 工作频率 */
ADC_CLK_Disable();
```

6.3.5 ADC_INT_Enable

- **函数**

ADC_INT_Enable();

- **函数功能**

使能ADC中断功能，设置寄存器INTE1[6]=1 。

- **输入参数**

无

- **包含头文件**

Drirver/ADC.h

- **函数返回值**

  无

- **函数用法**

  /* 使能ADC中断功能 */

  ADC_INT_Enable();

6.3.6 ADC_INT_Disable

- **函数**

  ADC_INT_Disable();

- **函数功能**

  关闭ADC中断功能，设置寄存器INTE1[6]=0 。

- **输入参数**

  无

- **包含头文件**

  Drirver/ADC.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭ADC中断功能 */

  ADC_INT_Disable();

6.3.7 ADC_INT_IsFlag

- **函数**

  ADC_INT_IsFlag()

- **函数功能**

  读取ADC中断请求标志位，读取寄存器INTF1[6]。

- **输入参数**

  无

- **包含头文件**

  Drirver/ADC.h

- **函数返回值**

  0x00：ADC不产生中断

  0x40：ADC产生中断请求

- **函数用法**

  /* 读取ADC中断请求标志位 */

  unsigned char flag;

  flag = ADC_INT_IsFlag();

6.3.8 ADC_INT_ClearFlag

- **函数**

  ADC_INT_ClearFlag();

- **函数功能**

  清除ADC中断请求标志位，设置寄存器INTF1[6]=0。

- **输入参数**

  无

- **包含头文件**

  Drirver/ADC.h

- **函数返回值**

  无

- **函数用法**

  /* 清零ADC中断请求标志位 */

  ADC_INT_ClearFlag();

6.3.9 ADC_INSEnable

- **函数**

  ADC_INSEnable();

- **函数功能**

  使能ADC信号输入端内部短路功能，设置寄存器AINET1[1]=1 。

- **输入参数**

  无

- **包含头文件**

  Drirver/ADC.h

- **函数返回值**

  无

- **函数用法**

  /* 设置ADC信号输入端内部短路 */

  ADC_INSEnable();

6.3.10 ADC_INSDisable

- **函数**

  ADC_INSDisable();

- **函数功能**

  关闭ADC信号输入端内部短路功能，设置寄存器AINET1[1]=0 。

- **输入参数**

  无

- **包含头文件**

Drirver/ADC.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭ADC信号输入端内部短路功能 */

  ADC_INSDisable();

6.3.11 ADC_Enable

- **函数**

  ADC_Enable();

- **函数功能**

  启动ADC转换功能;

  设置寄存器ADCCN1[7]=1 。

- **输入参数**

  无

- **包含头文件**

  Drirver/ADC.h

- **函数返回值**

  无

- **函数用法**

  /* 启动ADC转换功能 */

  ADC_Enable();

6.3.12 ADC_Disable

- **函数**

  ADC_Disable();

- **函数功能**

  关闭ADC转换功能，设置寄存器ADCCN1[7]=0 。

- **输入参数**

  无

- **包含头文件**

  Drirver/ADC.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭ADC转换功能 */

  ADC_Disable();

6.3.13 ADC_VRbufEnable

- **函数**

  ADC_VRbufEnable();

- **函数功能**

  使能ADC的参考电压输入端的输入缓冲器功能，设置寄存器ADCCN2[4]=1 。

- **输入参数**

  无

- **包含头文件**

  Drirver/ADC.h

- **函数返回值**

  无

- **函数用法**

  /* 使能ADC参考电压输入端输入缓冲器功能 */

  ADC_VRbufEnable();

6.3.14 ADC_VRbufDisable

- **函数**

  ADC_VRbufDisable();

- **函数功能**

  关闭ADC参考电压输入端输入缓冲器功能，设置寄存器ADCCN2[4]=0 。

- **输入参数**

  无

- **包含头文件**

  Drirver/ADC.h

- **函数返回值**

  无

- **函数用法**

  /* 关闭ADC参考电压输入端输入缓冲器功能 */

  ADC_VRbufDisable();

6.3.15 ADC_INbufEnable

- **函数**

  ADC_INbufEnable();

- **函数功能**

  使能ADC信号输入端的输入缓冲器功能，设置寄存器ADCCN2[5]=1 。

- **输入参数**

  无

- **包含头文件**

Drirver/ADC.h

- **函数返回值**

无

- **函数用法**

```
/* 使能ADC信号输入端的输入缓冲器功能 */
ADC_INbufEnable();
```

6.3.16 ADC_INbufDisable

- **函数**

```
ADC_INbufDisable();
```

- **函数功能**

关闭ADC信号输入端的输入缓冲器功能，设置寄存器ADCCN2[5]=0 。

- **输入参数**

无

- **包含头文件**

Drirver/ADC.h

- **函数返回值**

无

- **函数用法**

```
/* 关闭ADC信号输入端的输入缓冲器功能 */
ADC_INbufDisable();
```

6.3.17 ADC_AINConfig

- **函数**

```
ADC_AINConfig(inh,inl);
```

- **函数功能**

设置ADC信号输入端的来源，设置寄存器AINET1[7:5]。

- **输入参数**

inh[in]：设置ADC信号输入端的正向输入通道

AINET1_INH_AI2	：AI2
AINET1_INH_AI6	：AI6
AINET1_INH_AI4	：AI4
AINET1_INH_AI0	：AI0
AINET1_INH_HALF_VDDA	：VDDA/2
AINET1_INH_ADM	：ACM
AINET1_INH_TPSH1	：TPSH1
AINET1_INH_TPSH0	：TPSH0

inl[in]：设置ADC信号输入端的正向输入通道

AINET1_INL_AI3 ： AI3
AINET1_INL_AI7 ： AI7
AINET1_INL_AI5 ： AI5
AINET1_INL_AI1 ： AI1
AINET1_INL_OPO ： OPO
AINET1_INL_VSS ： VSS
AINET1_INL_TPSL0 ： TPSL0
AINET1_INL_TPSL1 ： TPSL1

- **包含头文件**

Drirver/ADC.h

- **函数返回值**

无

- **函数用法**

```
/* 设置ADC信号输入端为AI0---AI1 */
ADC_AINConfig( AINET1_INH_AI0, AINET1_INL_AI1 );
```

6.3.18 ADC_VRINConfig

- **函数**

ADC_VRINConfig(vrh,vrl);

- **函数功能**

设置ADC参考电压输入端来源，设置寄存器AINET2[6:5]/AINET2[2:1]。

- **输入参数**

vrh [in]：设置ADC参考电压输入端的正向输入通道
AINET2_VRH_AI6 ： AI6
AINET2_VRH_AI8 ： AI8
AINET2_VRH_AI2 ： AI2
AINET2_VRH_ACM：ACM
vrl [in]：设置ADC参考电压输入端的负向输入通道
AINET2_VRL_AI5 ： AI5
AINET2_VRL_AI9 ： AI9
AINET2_VRL_AI3 ： AI3
AINET2_VRL_VSS：VSS

- **包含头文件**

Drirver/ADC.h

- **函数返回值**

无

- **函数用法**

```
/* 设置ADC参考电压输入端为AI2—AI3 */
```

ADC_VRINConfig(AINET2_VRH_AI2, AINET2_VRL_AI3);

6.3.19 ADC_VRGainSelect

- **函数**

ADC_VRGainSelect(VRGSel);

- **函数功能**

设置ADC参考电压输入端放大倍数，设置寄存器ADCCN2[3]。

- **输入参数**

VRGSel [in]：设置ADC参考电压输入端放大倍数

ADCCN2_VREGN_DIV2：VREF * 1/2

ADCCN2_VREGN_DIV1：VREF* 1

- **包含头文件**

Drirver/ADC.h

- **函数返回值**

无

- **函数用法**

/* 设置ADC参考电压输入端放大倍数为1/2 */

ADC_VRGainSelect(ADCCN2_VREGN_DIV2);

6.3.20 ADC_GainConfig

- **函数**

ADC_GainConfig(adgn,pgagn) ;

- **函数功能**

设置ADC信号输入端放大倍数ADGN * PGA，设置寄存器ADCCN1[4 :0]。

- **输入参数**

adgn [in]：设置ADC的ADGN放大倍数

ADCCN1_ADGN_1DIV4 ： x 1/4

ADCCN1_ADGN_1DIV2 ： x 1/2

ADCCN1_ADGN_1 ： x 1

ADCCN1_ADGN_2 ： x 2

ADCCN1_ADGN_4 ： x 4

ADCCN1_ADGN_8 ： x 8

ADCCN1_ADGN_16 ： x 16

pgagn [in]：设置ADC的PGA放大倍数

ADCCN1_PGAGN_1：x 1

ADCCN1_PGAGN_2：x 2

ADCCN1_PGAGN_4：x 4

ADCCN1_PGAGN_8：x 8

- **包含头文件**

  Drirver/ADC.h

- **函数返回值**

  无

- **函数用法**

  /* 设置ADC放大倍数为16*8 */

  ADC_GainConfig(ADCCN1_ADGN_16, ADCCN1_PGAGN_8);

6.3.21 ADC_OSRConfig

- **函数**

  ADC_OSRConfig(osr) ;

- **函数功能**

  设置ADC转换输出速率，输出速率为ADC_CK/ OSR，设置寄存器ADCCN3[7 :5]。

- **输入参数**

  osr [in]：设置ADC转换输出速率控制值

  ADCCN3_OSR_256 ：转换输出速率为ADC_CK/256

  ADCCN3_OSR_512 ：转换输出速率为ADC_CK/512

  ADCCN3_OSR_1024 ：转换输出速率为ADC_CK/1024

  ADCCN3_OSR_2048 ：转换输出速率为ADC_CK/2048

  ADCCN3_OSR_4096 ：转换输出速率为ADC_CK/4096

  ADCCN3_OSR_8192 ：转换输出速率为ADC_CK/8192

  ADCCN3_OSR_16384 ：转换输出速率为ADC_CK/16384

  ADCCN3_OSR_32768 ：转换输出速率为ADC_CK/32768

- **包含头文件**

  Drirver/ADC.h

- **函数返回值**

  无

- **函数用法**

  /* 设置ADC转换输出速率为ADC_CK / 32768 */

  ADC_OSRConfig(ADCCN3_OSR_32768);

6.3.22 ADC_DCSetConfig

- **函数**

  ADC_DCSetConfig(dcset) ;

- **函数功能**

  设置ADC信号输入端的直流偏置电压，设置寄存器ADCCN2[2 :0]。

- **输入参数**

  dcset [in]：设置ADC的信号输入端直流偏置电压

ADCCN2_DCSET_0 ：不偏压
ADCCN2_DCSET_P1DIV4 ：+ VREF* 1/4
ADCCN2_DCSET_P1DIV2 ：+ VREF* 1/2
ADCCN2_DCSET_P3DIV4 ：+ VREF* 3/4
ADCCN2_DCSET_00 ：不偏压
ADCCN2_DCSET_N1DIV4 ：-VREF* 1/4
ADCCN2_DCSET_N1DIV2 ：-VREF* 1/2
ADCCN2_DCSET_N3DIV4 ：-VREF* 3/4

- **包含头文件**

Drirver/ADC.h

- **函数返回值**

无

- **函数用法**

```
/* 设置ADC的直流偏置电压为+ VREF* 1/4 */
ADC_DCSetConfig( ADCCN2_DCSET_P1DIV4 );
```

6.3.23 ADC_INXConfig

- **函数**

ADC_INXConfig(inxmode) ;

- **函数功能**

设置ADC信号输入端的输入信号转置器，设置寄存器AINET2[4:3]。

- **输入参数**

inxmode [in]：输入信号转置器工作模式选择
AINET2_INX_MODE0：INH→ADH，INL→ADL
AINET2_INX_MODE1：INL→ADH &ADL，INH浮接
AINET2_INX_MODE2：INH→ADH & ADL，INL浮接
AINET2_INX_MODE3：INH→ADL，INL→ADH

- **包含头文件**

Drirver/ADC.h

- **函数返回值**

无.

- **函数用法**

```
/* 设置ADC输入信号转置器模式为INH→ADH，INL→ADL */
ADC_INXConfig( AINET2_INX_MODE0 );
```

7. SPI 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 功能的开启控制及中断矢量的控制

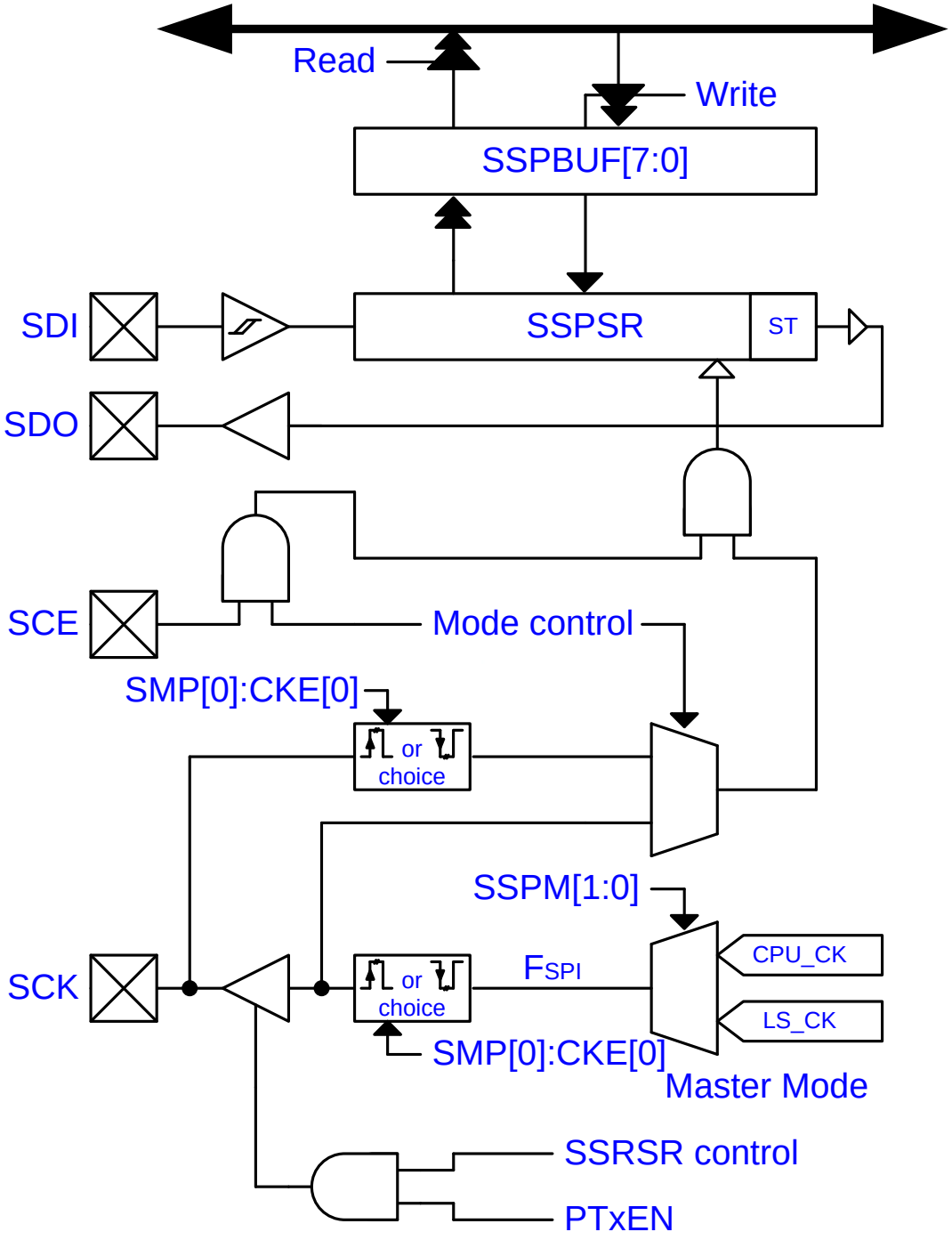
--SPI 的工作模式配置及工作状态控制

--SPI 的数据的收发

--包含 SPI.h

序号	函数名称	功能描述
01	SPI_Open	开启SPI功能，设置SPI工作模式，设置SPI通讯IO，设置SPI在收发数据完成后SCK电平状态及SPI收发数据时SCK电平状态
02	SPI_INT_Enable	使能 SPI 中断功能
03	SPI_INT_Disable	关闭 SPI 中断功能
04	SPI_INT_IsFlag	读取SPI 中断请求标志位
05	SPI_INT_ClearFlag	清除 SPI 中断请求标志位
06	SPI_Enable	开启SPI 功能
07	SPI_Disable	关闭 SPI 功能
08	SPI_CLKIdleConfig	设置数据收发完整后SCK电平状态
09	SPI_CLKConfig	设置数据收发时SCK电平状态
10	SPI_MasterMode	设置SPI 主动模式下输入数据采样时间点
11	SPI_SlaveMode	设置SPI 被动模式下输入数据采样时间点
12	SPI_Mode	设置SPI 工作模式
13	SPI_BUYCheck	读取SPI写入冲突状态标志位
14	SPI_BFCheck	接收缓冲器满标志位
15	SPI_ClearPOV	读取SPI 接收溢出标志位

7.2 SPI 模块方框图


Read
Write
SSPBUF[7:0]
SDI
SSPSR
ST
SDO
SCE
Mode control
SMP[0]:CKE[0]
or
choice
SSPM[1:0]
SCK
or
choice
FSPI
CPU_CK
LS_CK
SMP[0]:CKE[0]
Master Mode
SSRSR control
PTxEN

7.3 函数说明

7.3.1 SPI_Open

- **函数**

void SPI_Open(unsigned char sspm, unsigned char ckp, unsigned char cke, unsigned char smp);

- **函数功能**

开启SPI功能，设置SPI工作模式，设置SPI通讯IO，设置SPI在收发数据完成后SCK电平状态及SPI收发数据时SCK电平状态，设置寄存器SSPCON1/PT1DA/PT1M2/TRISC1。

- **输入参数**

sspm [in]：SPI工作模式设置

SSPCON1_SSPM_LSCK ：SPI主动模式，时钟源为LS_CK

SSPCON1_SSPM_CPUCK ：SPI主动模式，时钟源为CPU_CK

SSPCON1_SSPM_SCK ：SPI 3-线被动模式，时钟源由SCK引脚输入，SCE作为IO引脚使用

SSPCON1_SSPM_4WIRE ：SPI 4-线被动模式，时钟源由SCK引脚输入，SCE作为使能控制引脚

ckp [in]：设置数据收发完整后SCK电平状态

SSPCON1_CKP_HI ：时钟源为高电平时为空闲状态

SSPCON1_CKP_LOW ：时钟源为低电平时为空闲状态

cke [in]：设置数据收发时SCK电平状态

SSPCON1_CKE_IDLE ：当时钟源从有效状态变为空闲状态时发送

SSPCON1_CKE_EFFECTIVE ：当时钟源从空闲状态变为有效状态时发送

smp Div [in]：设置SPI 输入数据采样时间点

SPI 主动模式下：

SSPCON1_SMP_MASTEREND ：在资料输出时间的末端采样输入资料

SSPCON1_SMP_MASTERMID ：在资料输出时间的中间采样输入资料

SPI 被动模式下：

SSPCON1_SMP_SLAVE ：被动模式下，使用者需要将SMP BIT置0

- **包含头文件**

Driver/SPI.h

- **函数返回值**

无

- **函数用法**

/* 设置SPI 为主动模式且时钟源为CPU_CK，有效状态变为空闲状态时发送数据，及发送完成后为低电平，在时间中间采样输入数据 */

```
SPI_Open( SSPCON1_SSPM_CPUCK, SSPCON1_CKP_LOW,
          SSPCON1_CKE_IDLE, SSPCON1_SMP_MASTERMID );
```

7.3.2 SPI_INT_Enable

- 函数

SPI_INT_Enable();

- 函数功能

使能 SPI 中断功能，设置寄存器INTE2[2]=1 。

- 输入参数

无

- 包含头文件

Driver/SPI.h

- 函数返回值

无

- 函数用法

```
/* 使能 SPI 中断功能 */
SPI_INT_Enable();
```

7.3.3 SPI_INT_Disable

- 函数

SPI_INT_Disable() ;

- 函数功能

关闭 SPI 中断功能，设置寄存器INTE2[2]=0 。

- 输入参数

无

- 包含头文件

Driver/SPI.h

- 函数返回值

无

- 函数用法

```
/* 关闭 SPI 中断功能 */
SPI_INT_Disable();
```

7.3.4 SPI_INT_IsFlag

- 函数

SPI_INT_IsFlag() ;

- 函数功能

读取SPI 中断请求标志位，设置寄存器INTF2[2]。

- 输入参数

无

- 包含头文件

Driver/SPI.h

- 函数返回值

  0x00：SPI 没有中断请求

  0x04：SPI产生中断请求

- 函数用法

```
/* 读取 SPI 中断请求标志位 */
unsigned char flag;
flag = SPI_INT_IsFlag()>>2;
```

7.3.5 SPI_INT_ClearFlag

- 函数

  SPI_INT_ClearFlag();

- 函数功能

  清除 SPI 中断请求标志位，设置寄存器INTF2[2]=0 。

- 输入参数

  无

- 包含头文件

  Driver/SPI.h

- 函数返回值

  无

- 函数用法

```
/* 清除 SPI 中断请求标志位 */
SPI_INT_ClearFlag();
```

7.3.6 SPI_Enable

- 函数

  SPI_Enable();

- 函数功能

  开启SPI 功能，设置寄存器SSPCON1[7]=1 。

- 输入参数

  无

- 包含头文件

  Driver/SPI.h

- 函数返回值

  无

- 函数用法

```
/* 开启SPI 功能 */
SPI_Enable();
```

7.3.7 SPI_Disable

- 函数

SPI_Disable();

- 函数功能

关闭 SPI 功能，设置寄存器SSPCON1[7]=0 。

- 输入参数

无

- 包含头文件

Driver/SPI.h

- 函数返回值

无

- 函数用法

/* 关闭 SPI 功能 */

SPI_Disable();

7.3.8 SPI_CLKIdleConfig

- 函数

SPI_CLKIdleConfig(IdleCon);

- 函数功能

设置数据收发完整后SCK电平状态，设置寄存器SSPCON1[6]。

- 输入参数

IdleCon [in]：设置数据收发完整后SCK电平状态

SSPCON1_CKP_HI ：时钟源为高电平时为空闲状态

SSPCON1_CKP_LOW ：时钟源为低电平时为空闲状态

- 包含头文件

Driver/SPI.h

- 函数返回值

无

- 函数用法

/* 设置 SPI 收发数据完整后SCK电平为高电平 */

SPI_CLKIdleConfig(SSPCON1_CKP_HI);

7.3.9 DrvSPI32_DisableRxInt

- 函数

SPI_CLKConfig(CLKCon);

- 函数功能

设置数据收发时SCK电平状态，设置寄存器SSPCON1[5]。

- 输入参数

CLKCon [in]：设置数据收发时SCK电平状态

SSPCON1_CKE_IDLE ：当时钟源从有效状态变为空闲状态时发送

SSPCON1_CKE_EFFECTIVE ：当时钟源从空闲状态变为有效状态时发送

- 包含头文件

Driver/SPI.h

- 函数返回值

无

- 函数用法

```
/* 设置SCK从有效状态变为空闲状态时发送数据 */
SPI_CLKConfig( SSPCON1_CKE_IDLE );
```

7.3.10 SPI_MasterMode

- 函数

SPI_MasterMode(MModeCon);

- 函数功能

设置SPI 主动模式下输入数据采样时间点，设置寄存器SSPCON1[4] 。

- 输入参数

MModeCon Div [in]：设置SPI 输入数据采样时间点

SPI 主动模式下：

SSPCON1_SMP_MASTEREND ：在资料输出时间的末端采样输入资料

SSPCON1_SMP_MASTERMID ：在资料输出时间的中间采样输入资料

SPI 被动模式下：

SSPCON1_SMP_SLAVE ：被动模式下，使用者需要将SMP BIT置0

- 包含头文件

Driver/SPI.h

- 函数返回值

无

- 函数用法

```
/* 设置输入数据采样时间点为资料输出时间的中间 */
SPI_MasterMode( SSPCON1_SMP_MASTERMID );
```

7.3.11 SPI_SlaveMode

- 函数

SPI_SlaveMode();

- 函数功能

设置SPI 被动模式下输入数据采样时间点，设置寄存器SSPCON1[4]=0 。

- 输入参数

无

- 包含头文件

Driver/SPI.h

- 函数返回值

无

- 函数用法

/* 设置SPI 被动模式下输入数据采样时间点 */

SPI_SlaveMode();

7.3.12 SPI_Mode

- 函数

SPI_Mode(SSPMSel);

- 函数功能

设置 SPI 工作模式，读取寄存器SSPCON1[1:0]。

- 输入参数

SSPMSel [in]：SPI工作模式设置

SSPCON1_SSPM_LSCK ：SPI主动模式，时钟源为LS_CK

SSPCON1_SSPM_CPUCK ：SPI主动模式，时钟源为CPU_CK

SSPCON1_SSPM_SCK ：SPI 3-线被动模式，时钟源由SCK引脚输入，SCE作为IO引脚使用

SSPCON1_SSPM_4WIRE ：SPI 4-线被动模式，时钟源由SCK引脚输入，SCE作为使能控制引脚

- 包含头文件

Driver/SPI.h

- 函数返回值

无

- 函数用法

/* 设置SPI为主动模式，时钟源为CPU_CK */

SPI_Mode(SSPCON1_SSPM_CPUCK);

7.3.13 SPI_BUYCheck

- 函数

SPI_BUYCheck();

- 函数功能

读取SPI写入冲突状态标志位，设置寄存器SSPSTA[7]。

- 输入参数

无

- 包含头文件

Driver/SPI.h

- 函数返回值

0x00：未发生冲突

0x80：发生写入冲突，资料仍在发送

- **函数用法**

/* 读取SPI 写入冲突状态标志位 */

unsigned char flag ;

flag = SPI_BUYCheck();

7.3.14 SPI_BFCheck

- **函数**

SPI_BFCheck();

- **函数功能**

接收缓冲器满标志位，读取寄存器SSPSTA[0]。

- **输入参数**

无

- **包含头文件**

Driver/SPI.h

- **函数返回值**

0x00：接收未完整，SSPBUF为空

0x01：接收完成，SSPBUF已满

- **函数用法**

/* 读取 SPI 接收缓冲器满标志位 */

unsigned char flag;

flag = SPI_BFCheck();

7.3.15 SPI_ClearPOV

- **函数**

SPI_ClearPOV();

- **函数功能**

读取SPI 接收溢出标志位，读取寄存器SSPSTA[6] 。

- **输入参数**

无

- **包含头文件**

Driver/SPI.h

- **函数返回值**

0x00：未发生溢出

0x40：发生溢出

- **函数用法**

/* 读取SPI 接收溢出标志位 */

unsigned char flag；flag = SPI_ClearPOV();

8. 非同步串行通讯 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 中断矢量控制

--UART 收发错误控制

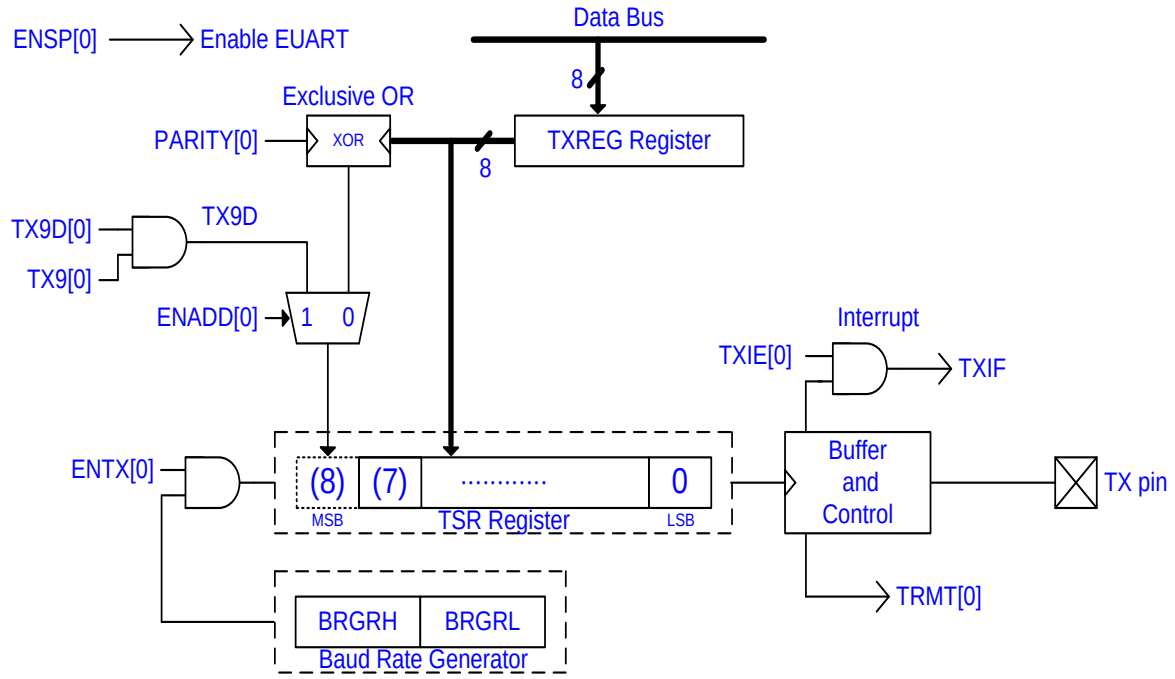
--包含UART.h

序号	函数名称	功能描述
01	UART_Open	启动UART及使能收发功能，设置波特率值，设置数据校验，设置UART通讯引脚，设置发送数据长度
02	UART_INT_TXEnable	使能UART发送中断功能
03	UART_INT_TXIsEnable	读取UART发送中断使能控制位状态
04	UART_INT_TXDisable	关闭UART发送中断功能
05	UART_INT_TXIsFlag	读取UART发送中断标志位状态值
06	UART_INT_TXIsFlag	清除UART发送中断标志位值
07	UART_INT_RCEnable	使能UART接收中断功能
08	UART_INT_RCDisable	关闭UART接收中断功能
09	UART_INT_RCIsFlag	读取UART接收中断请求标志位的值
10	UART_INT_RCClearFlag	清除UART接收中断请求标志位
11	UART_Enable	使能UART功能
12	UART_Disable	关闭UART功能
13	UART_TXEnable	启动UART发送功能
14	UART_TXDisable	关闭UART发送功能
15	UART_TX9Enable	启动UART发送9位数据的功能
16	UART_TX9Disable	关闭UART发送9位数据功能
17	UART_WakeUpEnable	启动接收自动唤醒功能
18	UART_WakeUpDisable	关闭接收自动唤醒功能
19	UART_DataReceiveEnable	使能UART接收功能
20	UART_DataReceiveDisable	关闭UART接收功能
21	UART_RC9Enable	使能UART接收9位数据功能
22	UART_RC9Disable	关闭UART接收9位数据功能
23	UART_AddrDetectionEnable	使能UART位地址检测功能
24	UART_AddrDetectionDisable	关闭UART位地址检测功能
25	UART_AutoBaudEnable	启动UART自动波特率功能
26	UART_AutoBaudDisable	关闭UART自动波特率功能
27	UART_TXData9	设定UART发送9位数据时的第9位的数值
28	UART_ParityCheck	设置UART的校验模式
29	UART_RCData9	读取接收9位数据时，第9位的数值
30	UART_PERRIsFlag	读取UART的资料同位检查结果标志位
31	UART_OERRIsFlag	读取UART已接收到2笔数据未处理状态标志位
32	UART_FERRIsFlag	读取UART接收数据完整状态标志位
33	UART_RCIDLIsFlag	读取UART 的接收状态标志位
34	UART_TRMTIsFlag	读取UART发送移位寄存器(TSR)状态标志位

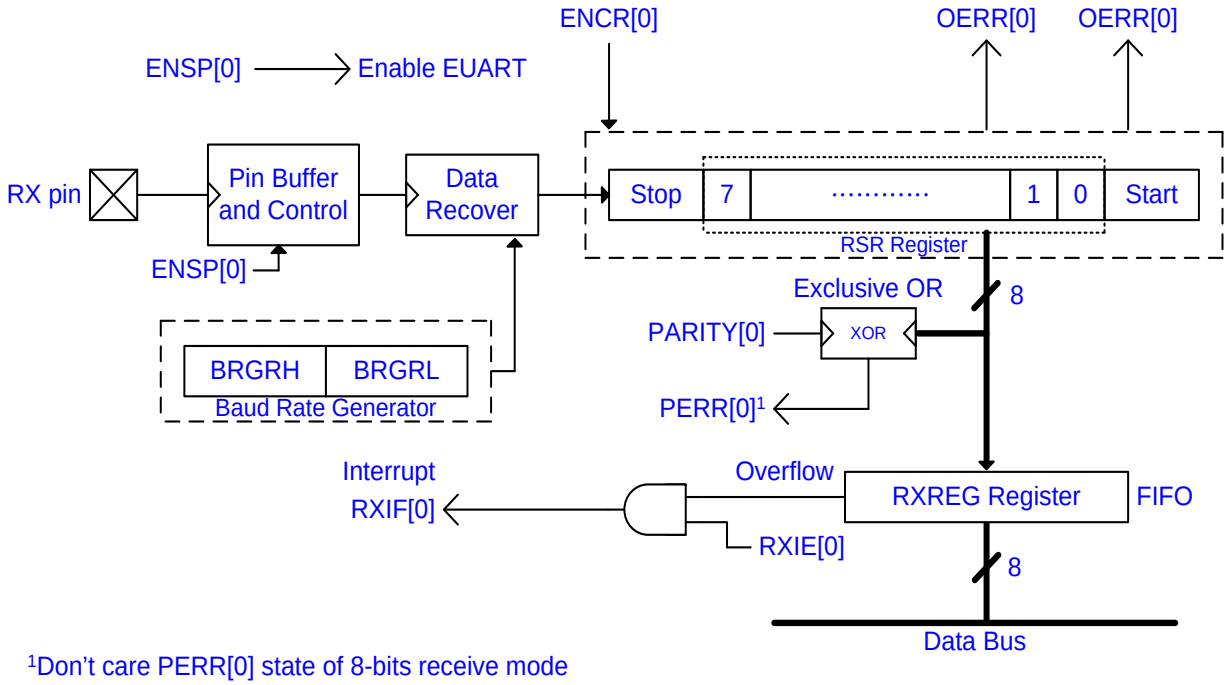
35	UART_ABDOVFIsFlag	读取UART自动波特率溢出标志位

8.2 UART 模块方框图

EUSART TRANSMIT BLOCK DIAGRAM



EUSART 8-BITs RECEIVE BLOCK DIAGRAM



[1]Don't care PERR[0] state of 8-bits receive mode

8.3 函数说明

8.3.1 UART_Open

- 函数

void UART_Open(unsigned int uBRGR,unsigned char uDataBits,unsigned char uParity);

- 函数功能

启动UART及使能收发功能，设置波特率值，设置数据校验，设置UART通讯引脚，设置发送数据长度，设置寄存器URCON/BAUDCON/BRGRH/BRGRL/TRISC1/PT1PU/PT1M1。

- 输入参数

uBRGR [in]：设置UART通讯数据波特率，输入范围0x00~0xFFFF

uDataBits [in]：UART数据长度

8：8位数据

9：9位数据

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

URCON_PARITY_Odd ：奇校验

URCON_PARITY_Even ：偶校验

URCON_PARITY_None ：无校验

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

```
/* HAO=2MHZ，设置UART波特率为9600bp，8 位数据，且无校验，通讯口为PT1.4/PT1.3 */
 UART_Open ( 0x33, 8 , URCON_PARITY_None );
```

8.3.2 UART_INT_TXEnable

- 函数

UART_INT_TXEnable();

- 函数功能

使能UART发送中断功能，清零寄存器INTE2[7]=1 。

- 输入参数

无

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

/* 使能UART发送中断功能 */

UART_INT_TXEnable();

8.3.3 UART_INT_TXIsEnable

- 函数

UART_INT_TXIsEnable();

- 函数功能

读取UART发送中断使能控制位状态，读取寄存器INTE2[7]。

- 输入参数

无

- 包含头文件

Drviver/UART.h

- 函数返回值

0x00：关闭UART发送中断功能

0x80：使能UART发送中断功能

- 函数用法

/* 读取UART发送中断使能控制位状态 */

unsigned char flag;

flag = UART_INT_TXIsEnable();

8.3.4 UART_INT_TXDisable

- 函数

UART_INT_TXDisable();

- 函数功能

关闭UART发送中断功能，设置寄存器INTE2[7]=0 。

- 输入参数

无

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

/* 关闭UART发送中断功能 */

UART_INT_TXDisable();

8.3.5 UART_INT_TXIsFlag

- 函数

  UART_INT_TXIsFlag();

- 函数功能

  读取UART发送中断标志位状态值，读取寄存器INTF2[7]的值。

- 输入参数

  无

- 包含头文件

  Drviver/UART.h

- 函数返回值

  0x80 : UART产生中断请求

  0x00 : UART未产生中断请求

- 函数用法

```
/* 读取UART发送中断标志位 */
 unsigned char flag；
 flag = UART_INT_TXIsFlag();
```

8.3.6 UART_INT_TXClearFlag

- 函数

  UART_INT_TXIsFlag();

- 函数功能

  清除UART发送中断标志位值，设置寄存器INTF2[7]=0 。

- 输入参数

  无

- 包含头文件

  Drviver/UART.h

- 函数返回值

  无

- 函数用法

```
/* 清除UART发送中断标志位 */
UART_INT_TXIsFlag();
```

8.3.7 UART_INT_RCEnable

- 函数

  UART_INT_RCEnable();

- 函数功能

  使能UART接收中断功能，设置寄存器INTE2[6]=1 。

- 输入参数

无

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

/* 使能UART接收中断功能 */

UART_INT_RCEnable();

8.5.8 UART_INT_RCDisable

- 函数

UART_INT_RCDisable();

- 函数功能

关闭UART接收中断功能，设置寄存器INTE2[6]=0 。

- 输入参数

无

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

/* 关闭UART接收中断功能 */

UART_INT_RCDisable();

8.3.9 UART_INT_RCIsFlag

- 函数

UART_INT_RCIsFlag();

- 函数功能

读取UART接收中断请求标志位的值，读取寄存器INTF2[6]。

- 输入参数

无

- 包含头文件

Drviver/UART.h

- 函数返回值

0x00：UART无接收中断请求

0x40：UART有接收中断请求

- 函数用法

/* 读取UART的接收中断标志位 */

```
unsigned char flag;
flag = UART_INT_RCIsFlag();
```

8.3.10 UART_INT_RCClearFlag

- 函数

UART_INT_RCClearFlag();

- 函数功能

清除UART接收中断请求标志位，设置寄存器INTF2[6]=0 。

- 输入参数

uData [in]：待发送的数据

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

```
/* 清除UART接收中断请求标志位 */
UART_INT_RCClearFlag();
```

8.3.11 UART_Enable

- 函数

UART_Enable();

- 函数功能

使能UART功能，设置寄存器URCON[7]=1 。

- 输入参数

无

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

```
/* 使能UART功能 */
UART_Enable();
```

8.3.12 UART_Disable

- 函数

UART_Disable();

- 函数功能

关闭UART功能，设置寄存器URCON[7]=0 。

- 输入参数

无

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

```
/* 关闭UART功能 */
UART_Disable();
```

8.3.13 UART_TXEnable

- 函数

UART_TXEnable();

- 函数功能

启动UART发送功能，设置寄存器URCON[6]=1 。

- 输入参数

无

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

```
/* 启动UART发送功能 */
UART_TXEnable();
```

8.3.14 UART_TXDisable

- 函数

UART_TXDisable();

- 函数功能

关闭UART发送功能，设置寄存器URCON[6]=0 。

- 输入参数

无

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

```
/* 关闭UART发送功能 */
UART_TXDisable();
```

8.3.15 UART_TX9Enable

- 函数

UART_TX9Enable();

- 函数功能

启动UART发送9位数据的功能，设置寄存器URCON[5]=1 。

- 输入参数

无

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

```
/* 启动UART发送9位数据功能 */
UART_TX9Enable();
```

8.3.16 UART_TX9Disable

- 函数

UART_TX9Disable();

- 函数功能

关闭UART发送9位数据功能，设置寄存器URCON[5]=0 。

- 输入参数

无

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

```
/* 关闭UART发送9位数据功能 */
UART_TX9Disable();
```

8.3.17 UART_WakeUpEnable

- 函数

UART_WakeUpEnable();

- 函数功能

启动接收自动唤醒功能，设置寄存器URCON[0]=1 。

- 输入参数

  无

- 包含头文件

  Drviver/UART.h

- 函数返回值

  无

- 函数用法

```
/* 启动接收自动唤醒功能 */
UART_WakeUpEnable();
```

8.3.18 UART_WakeUpDisable

- 函数

  UART_WakeUpDisable() ;

- 函数功能

  关闭接收自动唤醒功能，设置寄存器URCON[0]=0 。

- 输入参数

  无

- 包含头文件

  Drviver/UART.h

- 函数返回值

  无

- 函数用法

```
/* 关闭接收自动唤醒功能 */
UART_WakeUpDisable();
```

8.3.19 UART_DataReceiveEnable

- 函数

  UART_DataReceiveEnable();

- 函数功能

  使能UART接收功能，设置寄存器BAUDCON[3]=1 。

- 输入参数

  无

- 包含头文件

  Drviver/UART.h

- 函数返回值

  无

- 函数用法

```
/* 使能UART接收功能 */
UART_DataReceiveEnable();
```

8.3.20 UART_DataReceiveDisable

- 函数

  UART_DataReceiveDisable() ;

- 函数功能

  关闭UART接收功能，设置寄存器BAUDCON[3]=0 。

- 输入参数

  无

- 包含头文件

  Drviver/UART.h

- 函数返回值

  无

- 函数用法

```
/* 关闭UART接收功能 */
UART_DataReceiveDisable();
```

8.3.21 UART_RC9Enable

- 函数

  UART_RC9Enable();

- 函数功能

  使能UART接收9位数据功能，设置寄存器BAUDCON[2]=1 。

- 输入参数

  无

- 包含头文件

  Drviver/UART.h

- 函数返回值

  无

- 函数用法

```
/* 使能UART接收9位数据功能 */
UART_RC9Enable();
```

8.3.22 UART_RC9Disable

- 函数

  UART_RC9Disable() ;

- 函数功能

  关闭UART接收9位数据功能，设置寄存器BAUDCON[2]=0 。

- 输入参数

  无

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

```
/* 关闭UART接收9位数据功能 */
UART_RC9Disable();
```

8.3.23 UART_AddrDetectionEnable

- 函数

UART_AddrDetectionEnable();

- 函数功能

使能UART位地址检测功能，读取寄存器BAUDCON[1]=1 。

- 输入参数

无

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

```
/* 使能UART位地址检测功能 */
UART_AddrDetectionEnable();
```

8.3.24 UART_AddrDetectionDisable

- 函数

UART_AddrDetectionDisable();

- 函数功能

关闭UART位地址检测功能，读取寄存器BAUDCON[1]=0 。

- 输入参数

无。

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

```
/* 关闭UART位地址检测功能 */
UART_AddrDetectionDisable();
```

8.3.25 UART_AutoBaudEnable

- 函数

UART_AutoBaudEnable();

- 函数功能

启动UART自动波特率功能，设置寄存器BAUDCON[0]=1 。

- 输入参数

无

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

/* 启动UART自动波特率功能 */

UART_AutoBaudEnable();

8.3.26 UART_AutoBaudDisable

- 函数

UART_AutoBaudDisable();

- 函数功能

关闭UART自动波特率功能，设置寄存器BAUDCON[0]=0 。

- 输入参数

无

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

/* 关闭UART自动波特率功能 */

UART_AutoBaudDisable();

8.3.27 UART_TXData9

- 函数

UART_TXData9(TX9DSel);

- 函数功能

设定UART发送9位数据时的第9位的数值，设置寄存器URCON[4]。

- 输入参数

TX9Dsel [in]：设定UART发送9位数据时的第9位的数值

URCON_TX9D_1：第9位数据为1

URCON_TX9D_0：第9位数据为0

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

/* 设置UART发送9位数据时第九位数值为1 */

UART_TX9Enable();

UART_TXData9(URCON_TX9D_1);

8.3.28 UART_ParityCheck

- 函数

UART_ParityCheck(PARSel);

- 函数功能

设置UART的校验模式，寄存器URCON[3]。

- 输入参数

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

URCON_PARITY_Odd ：奇校验

URCON_PARITY_Even ：偶校验

URCON_PARITY_None ：无校验

- 包含头文件

Drviver/UART.h

- 函数返回值

无

- 函数用法

/* 设置UART为偶校验 */

UART_ParityCheck(URCON_PARITY_Even);

8.3.29 UART_RCData9

- 函数

UART_RCData9();

- 函数功能

读取接收9位数据时，第9位的数值，读取寄存器BAUDCON[2]。

- 输入参数

无

- 包含头文件

Drviver/UART.h

- 函数返回值

返回BAUDCON的值：

0x00：RC9D为0

0x04：RC9D为1

- 函数用法

```
/* 读取接收到的第9位数值 */
unsigned char RC9D;
RC9D = UART_RCData9()>>2;
```

8.3.30 UART_PERRIsFlag

- 函数

UART_PERRIsFlag();

- 函数功能

读取UART的资料同位检查结果标志位(PERR)，读取寄存器URSTA[5]。

- 输入参数

无

- 包含头文件

Drviver/UART.h

- 函数返回值

0x00：接收同位检查正确

0x20：接收同位检查错误

- 函数用法

```
/* 取UART的资料同位检查结果标志位(PERR) */
unsigned char PERR_F;
PERR_F = UART_PERRIsFlag();
```

8.3.31 UART_OERRIsFlag

- 函数

UART_OERRIsFlag();

- 函数功能

读取UART已接收到2笔数据未处理状态标志位(OERR)，读取寄存器URSTA[3]。

- 输入参数

无

- 包含头文件

Drviver/UART.h

- 函数返回值

0x00：未发生

0x08：已发生

- 函数用法

```
/* 读取UART已接收到2笔数据未处理状态标志位(OERR) */
unsigned char OERR_F;
```

OERR_F = UART_OERRIsFlag();

8.3.32 UART_FERRIsFlag

- 函数

  UART_FERRIsFlag()

- 函数功能

  读取UART接收数据完整状态标志位(FERR)，读取寄存器URSTA[4]。

- 输入参数

  无

- 包含头文件

  Drviver/UART.h

- 函数返回值

  0x00：表示资料接收完整

  0x10：表示资料接收不完整

- 函数用法

  /* 读取UART接收数据完整状态标志位(FERR) */

  unsigned char FERR_F ;

  FERR_F = UART_FERRIsFlag();

8.3.33 UART_RCIDLIsFlag

- 函数

  UART_RCIDLIsFlag() ;

- 函数功能

  读取UART 的接收状态标志位(RCIDL)，读取寄存器URSTA[2]。

- 输入参数

  无

- 包含头文件

  Drviver/UART.h

- 函数返回值

  0x00：不在接收状态

  0x04：在接收状态

- 函数用法

  /* 读取UART 的接收状态标志位(RCIDL) */

  unsigned char flag ;

  flag = UART_RCIDLIsFlag();

8.3.34 UART_TRMTIsFlag

- **函数**

  UART_TRMTIsFlag();

- **函数功能**

  读取UART发送移位寄存器(TSR)状态标志位(RTMT)，读取寄存器URSTA[1]。

- **输入参数**

  无

- **包含头文件**

  Drviver/UART.h

- **函数返回值**

  0x00：表示TSR寄存器有资料

  0x02：表示TSR寄存器为空

- **函数用法**

```
/* 读取UART发送移位寄存器(TSR)状态标志位(RTMT) */
unsigned char flag ;
flag = UART_TRMTIsFlag();
```

8.3.35 UART_ABDOVFIsFlag

- **函数**

  UART_ABDOVFIsFlag();

- **函数功能**

  读取UART自动波特率溢出标志位(ABDOVF)，读取寄存器URSTA[0]。

- **输入参数**

  无

- **包含头文件**

  Drviver/UART.h

- **函数返回值**

  0x00：波特率没发生溢出

  0x01：波特率发生溢出

- **函数用法**

```
/* 读取UART自动波特率溢出标志位(ABDOVF) */
unsigned char flag ;
flag = UART_ABDOVFIsFlag();
```

9. 增强型多功能比较器 ECPA

9.1 函数简介

该部分函数描述增强型比较器功能的控制，包含：

--ECPA 模块功能的启动与关闭

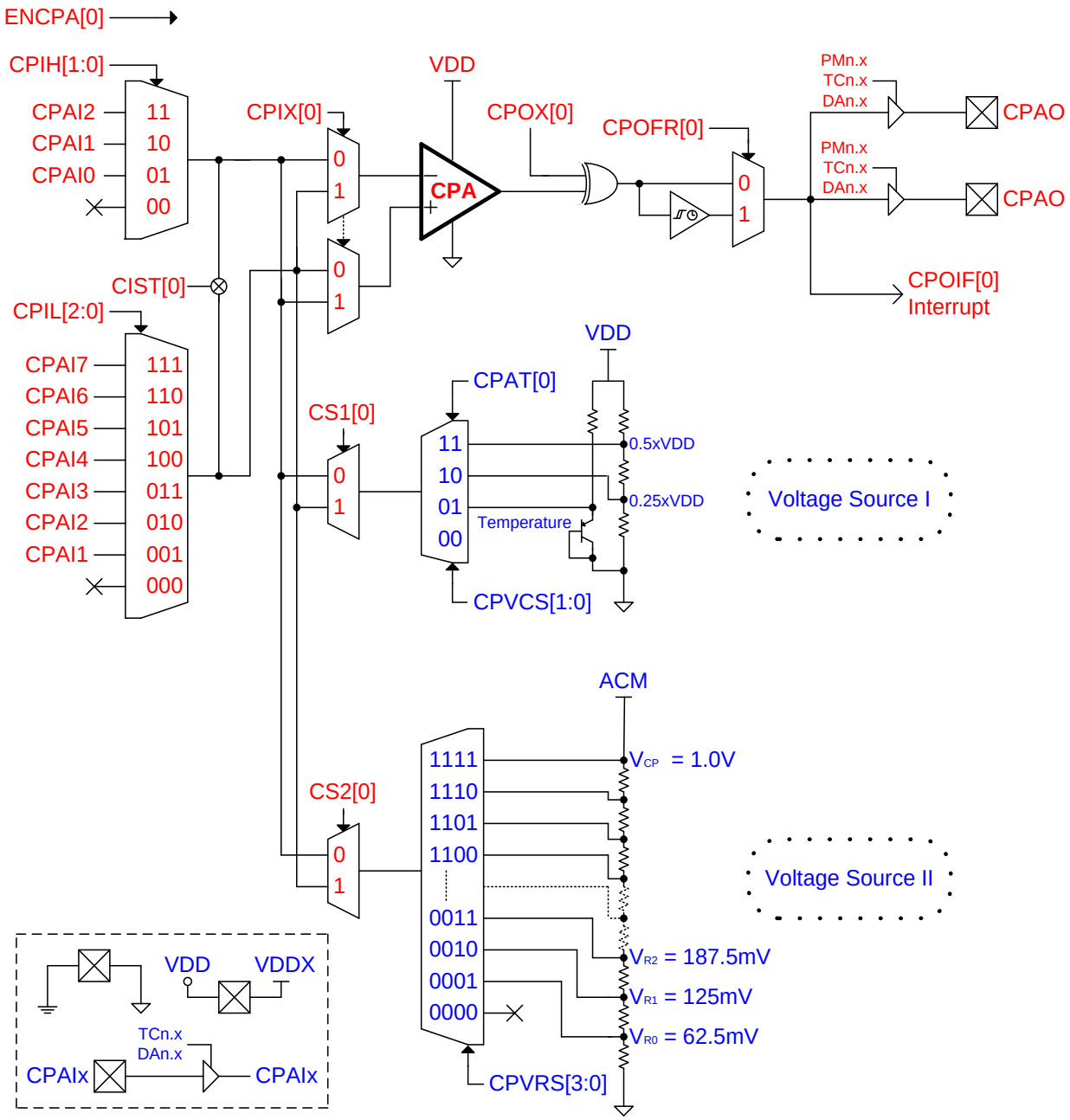
--ECPA 配置控制

--ECPA 中断矢量控制

--包含 ECPA.h

序号	函数名称	功能描述
01	ECPA_Open	使能比较器功能，设置比较器信号输入通道，设置比较器参考电压及设置比较器信号输出引脚
02	ECPA_INT_Enable	使能比较器中断功能
03	ECPA_INT_Disable	关闭比较器中断功能
04	ECPA_INT_IsFlag	读取比较器中断请求标志位
05	ECPA_INT_ClearFlag	清除比较器中断请求标志位
06	ECPA_Enable	使能比较器功能
07	ECPA_Disable	关闭比较器功能
08	ECPA_InShort	设置比较器输入通道短路器功能
9	ECPA_SignConverter	设置输入通道交换器功能
10	ECPA_InChanH	设置比较器正向输入通道
11	ECPA_InChanL	设置比较器负向输入通道
12	ECPA_OutReverse	设置比较器输出信号反相器功能
13	ECPA_SignProcessor	设置比较器输出信号经过滤波器功能
14	ECPA_OutState	设置比较器使用参考电压发生器I时，输出状态自动转态
15	ECPA_Ref2CompIn1	设置参考电压发生器I的输出信号，输出到正向端或负向端
16	ECPA_RefVolChan1	设置比较器参考电压发生器 I的输出电压
17	ECPA_Ref2CompIn2	设置参考电压发生器II的输出信号，输出到正向端或负向端
18	ECPA_RefVolChan2	设置比较器参考电压发生器II的输出电压

9.2 ECPA 模块方框图



9.3 函数说明

9.3.1 ECPA_Open

- **函数**

void ECPA_Open(unsigned char cpil, unsigned char cpih, unsigned char cpvcs,
unsigned char cpvrx, unsigned char output);

- **函数功能**

使能比较器功能，设置比较器信号输入通道，设置比较器参考电压及设置比较器信号输出引脚，设置寄存器CPACN1/CPACN2/CPACN3。

- **输入参数**

cpih [in]：比较器信号正向输入通道设置

CPACN1_CPIH_CAPI2 ：输入通道CAPI2
CPACN1_CPIH_CAPI1 ：输入通道CAPI1
CPACN1_CPIH_CAPI0 ：输入通道CAPI0
CPACN1_CPIH_HIZ ：高阻抗

cpil [in]：比较器信号负向输入通道设置

CPACN1_CPIL_CAPI7：输入通道CAPI7
CPACN1_CPIL_CAPI6：输入通道CAPI6
CPACN1_CPIL_CAPI5：输入通道CAPI5
CPACN1_CPIL_CAPI4：输入通道CAPI4
CPACN1_CPIL_CAPI3：输入通道CAPI3
CPACN1_CPIL_CAPI2：输入通道CAPI2
CPACN1_CPIL_CAPI1：输入通道CAPI1
CPACN1_CPIL_HIZ ：高阻抗

cpvcs [in]：设置比较器参考电压发生器I的输出电压

CPACN2_CPVCS_VDDDIV2 ：0.5 * VDD
CPACN2_CPVCS_VDDDIV4 ：0.25 * VDD
CPACN2_CPVCS_REFTEMP ：参考温度
CPACN2_CPVCS_HIZ ：高阻抗

cpvrx [in]：设置比较器参考电压发生器II的输出电压

CPACN3_CPVRX_1V000 ：输出1000mV
CPACN3_CPVRX_0V9375 ：输出0.9375mV
CPACN3_CPVRX_0V875 ：输出0.875mV
CPACN3_CPVRX_0V8125 ：输出0.8125mV
CPACN3_CPVRX_0V750 ：输出0.750mV
CPACN3_CPVRX_0V6875 ：输出0.6875mV
CPACN3_CPVRX_0V625 ：输出0.625mV
CPACN3_CPVRX_0V5625 ：输出0.5625mV

CPACN3_CPVRX_0V4375 ：输出0.4375mV
CPACN3_CPVRX_0V375 ：输出0.375mV
CPACN3_CPVRX_0V3125 ：输出0.3125mV
CPACN3_CPVRX_0V250 ：输出0.250mV
CPACN3_CPVRX_0V1875 ：输出0.1875mV
CPACN3_CPVRX_0V125 ：输出0.125mV
CPACN3_CPVRX_0V0625 ：输出0.0625mV
CPACN3_CPVRX_HIZ ：高阻抗

output [in]：设置比较器输出信号引脚
0：没有输出
1：PT2.7
2：PT2.7
3：PT2.6 & PT2.7

- **包含头文件**

Driver/ECPA.h

- **函数返回值**

无

- **函数用法**

/* 使能比较器，设置输入通道为CPIH=AI2,CPIL=AI3，参考电压发生器I输出电压为0.5*VDD，参考电压发生器II输出电压为0.375mv，输出引脚为PT2.7 */

```
ECPA_Open( CPACN1_CPIH_CAPI2, CPACN1_CPIL_CAPI3, CPACN2_CPVCS_VDDDIV2,
                CPACN3_CPVRX_0V375, 2 );
```

9.3.2 ECPA_INT_Enable

- **函数**

ECPA_INT_Enable();

- **函数功能**

使能比较器中断功能，设置寄存器INTE2[3]=1 。

- **输入参数**

无

- **包含头文件**

Driver/ECPA.h

- **函数返回值**

无

- **函数用法**

/* 使能比较器中断功能 */

ECPA_INT_Enable();

9.3.3 ECPA_INT_Disable

- 函数

  ECPA_INT_Disable();

- 函数功能

  关闭比较器中断功能，设置寄存器INTE2[3]=0 。

- 输入参数

  无

- 包含头文件

  Driver/ECPA.h

- 函数返回值

  无

- 函数用法

  /* 关闭比较器中断功能 */

  ECPA_INT_Disable();

9.3.4 ECPA_INT_IsFlag

- 函数

  ECPA_INT_IsFlag();

- 函数功能

  读取比较器中断请求标志位，读取寄存器INTF2[3]。

- 输入参数

  无

- 包含头文件

  Driver/ECPA.h

- 函数返回值

  0x00：比较器没有产生中断请求

  0x08：比较器产生中断请求

- 函数用法

  /* 读取比较器中断请求标志位 */

  unsigned char flag;

  flag = ECPA_INT_IsFlag();

9.3.5 ECPA_INT_ClearFlag

- 函数

  ECPA_INT_ClearFlag();

- 函数功能

  清除比较器中断请求标志位，设置寄存器INTF2[3]=0 。

- 输入参数

无

- 包含头文件

Driver/ECPA.h

- 函数返回值

无

- 函数用法

```
/* 清除比较器中断请求标志位 */
ECPA_INT_ClearFlag();
```

9.3.6 ECPA_Enable

- 函数

ECPA_Enable();

- 函数功能

使能比较器功能，设置寄存器CPACN1[7]=1 。

- 输入参数

无

- 包含头文件

Driver/ECPA.h

- 函数返回值

无

- 函数用法

```
/* 开启比较器功能 */
ECPA_Enable();
```

9.3.7 ECPA_Disable

- 函数

ECPA_Disable();

- 函数功能

关闭比较器功能，设置寄存器CPACN1[7]=0 。

- 输入参数

无

- 包含头文件

Driver/ECPA.h

- 函数返回值

无

- 函数用法

```
/* 关闭比较器功能 */
ECPA_Disable();
```

9.3.8 ECPA_InShort

- 函数

ECPA_InShort(ISTSel);

- 函数功能

设置比较器输入通道短路器功能，设置寄存器CPACN1[6]。

- 输入参数

ISTSel [in] 设置比较器输入通道短路器功能

CPACN1_CPIST_SHORT：输入短路

CPACN1_CPIST_OPEN ：输入不短路

- 包含头文件

Driver/ECPA.h

- 函数返回值

无

- 函数用法

/* 使能比较器输入通道短路 */

ECPA_InShort(CPACN1_CPIST_SHORT);

9.3.9 ECPA_SignConverter

- 函数

ECPA_SignConverter(IXSel);

- 函数功能

设置输入通道交换器功能，设置寄存器CPACN1[5]。

- 输入参数

IXSel [in]：设置输入通道交换器功能

CPACN1_CPIX_SWITCH ：输入通道交换

CPACN1_CPIX_NOTSWITCH ：输入通道不交换

- 包含头文件

Driver/ECPA.h

- 函数返回值

无

- 函数用法

/* 设置比较器输入信号通道不交换 */

ECPA_SignConverter(CPACN1_CPIX_NOTSWITCH);

9.3.10 ECPA_InChanH

- 函数

ECPA_InChanH(IHSel);

- 函数功能

设置比较器正向输入通道，设置寄存器CPACN1[4:3]。

- 输入参数

IHSel [in]：比较器信号正向输入通道设置

CPACN1_CPIH_CAPI2 ：输入通道CAPI2

CPACN1_CPIH_CAPI1 ：输入通道CAPI1

CPACN1_CPIH_CAPI0 ：输入通道CAPI0

CPACN1_CPIH_HIZ ：高阻抗

- 包含头文件

Driver/ECPA.h

- 函数返回值

无

- 函数用法

```
/* 设置比较器正向输入通道为AI2 */
ECPA_InChanH( CPACN1_CPIH_CAPI2 );
```

9.3.11 ECPA_InChanL

- 函数

ECPA_InChanL(ILSel);

- 函数功能

设置比较器负向输入通道，设置寄存器CPACN1[2:0]。

- 输入参数

ILSel [in]：比较器信号负向输入通道设置

CPACN1_CPIL_CAPI7：输入通道CAPI7

CPACN1_CPIL_CAPI6：输入通道CAPI6

CPACN1_CPIL_CAPI5：输入通道CAPI5

CPACN1_CPIL_CAPI4：输入通道CAPI4

CPACN1_CPIL_CAPI3：输入通道CAPI3

CPACN1_CPIL_CAPI2：输入通道CAPI2

CPACN1_CPIL_CAPI1：输入通道CAPI1

CPACN1_CPIL_HIZ ：高阻抗

- 包含头文件

Driver/ECPA.h

- 函数返回值

无

- 函数用法

```
/* 设置比较器负向输入通道为AI3 */
ECPA_InChanL( CPACN1_CPIL_CAPI3 );
```

9.3.12 ECPA_OutReverse

- 函数

ECPA_OutReverse(OXSel);

- 函数功能

设置比较器输出信号反相器功能，设置寄存器CPACN2[6]。

- 输入参数

OXSel [in]：设置比较器输出信号反相器功能

CPACN2_CPOX_REVERSE ：输出反向

CPACN2_CPOX_NOTREVERSE ：输出不反向

- 包含头文件

Driver/ECPA.h

- 函数返回值

无

- 函数用法

```
/* 设置比较器输出信号反向 */
ECPA_OutReverse( CPACN2_CPOX_REVERSE );
```

9.3.13 ECPA_SignProcessor

- 函数

ECPA_SignProcessor(OFRSel);

- 函数功能

设置比较器输出信号经过滤波器功能，读取寄存器CPACN2[5]的值。

- 输入参数

OFRSel [in]：设置比较器输出信号经过滤波功能

CPACN2_CPOFR_FILTER ：输出信号经过滤波器

CPACN2_CPOFR_NOTFILTER ：输出信号不经过滤波器

- 包含头文件

Driver/ECPA.h

- 函数返回值

无

- 函数用法

```
/* 设置比较器输出信号不经过滤波器 */
ECPA_SignProcessor( CPACN2_CPOFR_NOTFILTER );
```

9.3.14 ECPA_OutState

- 函数

ECPA_OutState(ATSel);

- 函数功能

设置比较器使用参考电压发生器I时，比较器输出状态是否自动转态，设置寄存器CPACN2[3] 。

- 输入参数

ATSel [in]：设置比较器使用参考电压发生器I时，比较器输出状态是否自动状态

CPACN2_CPAT_AUTO ：输出状态自动转态

CPACN2_CPAT_NORMAL：正常操作

- 包含头文件

Driver/ECPA.h

- 函数返回值

无

- 函数用法

```
/* 设置比较器在使用参考电压发生器I时，输出状态自动转态 */
ECPA_OutState( CPACN2_CPAT_AUTO );
```

9.3.15 ECPA_Ref2CompIn1

- 函数

ECPA_Ref2CompIn1(CS1Sel) ;

- 函数功能

设置参考电压发生器I的输出信号，输出到正向端或负向端，设置寄存器CPACN2[4]。

- 输入参数

CS1Sel [in]：设置参考电压发生器I的输出信号，输出到正向端或负向端

当CPIX = 1时：

CPACN2_CS1_P： 输出至‘+’端

CPACN2_CS1_N： 输出至‘-’端

当CPIX = 0时：

CPACN2_CS1_P： 输出至‘-’端

CPACN2_CS1_N： 输出至‘+’端

- 包含头文件

Driver/ECPA.h

- 函数返回值

无

- 函数用法

```
/* CPIX=1，设置参考电压输出信号至‘+’端 */
ECPA_SignConverter( CPACN1_CPIX_SWITCH );
ECPA_Ref2CompIn1( CPACN2_CS1_P );
```

9.3.16 ECPA_RefVolChan1

- 函数

ECPA_RefVolChan1(VCSSel) ;

- **函数功能**

设置比较器参考电压发生器 I 的输出电压，设置寄存器CPACN2[2:1]。

- **输入参数**

VCSSel [in]：设置比较器参考电压发生器 I 的输出电压

CPACN2_CPVCS_VDDDIV2 ：0.5 * VDD

CPACN2_CPVCS_VDDDIV4 ：0.25 * VDD

CPACN2_CPVCS_REFTEMP ：参考温度

CPACN2_CPVCS_HIZ ：高阻抗

- **包含头文件**

Driver/ECPA.h

- **函数返回值**

无

- **函数用法**

```
/* 设置比较器参考电压发生器I的输出电压为0.5*VDD */
ECPA_RefVolChan1( CPACN2_CPVCS_VDDDIV2 );
```

9.3.17 ECPA_Ref2CompIn2

- **函数**

ECPA_Ref2CompIn2(CS2Sel);

- 函数功能

设置参考电压发生器II的输出信号，输出到正向端或负向端，设置寄存器CPACN3[4]。

- **输入参数**

CS2Sel [in]：设置参考电压发生器I的输出信号，输出到正向端或负向端

当CPIX = 1时：

CPACN3_CS2 _P： 输出至‘+’端

CPACN3_CS2 _N： 输出至‘-’端

当CPIX = 0时：

CPACN3_CS2 _P： 输出至‘-’端

CPACN3_CS2 _N： 输出至‘+’端

- **包含头文件**

Driver/ECPA.h

- **函数返回值**

无

- **函数用法**

```
/* CPIX=1，设置参考电压发生器II输出信号至‘+’端 */
ECPA_SignConverter( CPACN1_CPIX_SWITCH );
ECPA_Ref2CompIn2( CPACN3_CS2_P );
```

9.3.18 ECPA_RefVolChan2

- **函数**

  ECPA_RefVolChan2(VRXSel);

- **函数功能**

  设置比较器参考电压发生器II的输出电压，设置寄存器CPACN3[3:0]。

- **输入参数**

  VRXSel [in]：设置比较器参考电压发生器II的输出电压

  CPACN3_CPVRX_1V000 ： 输出1000mV
  CPACN3_CPVRX_0V9375 ： 输出0.9375mV
  CPACN3_CPVRX_0V875 ： 输出0.875mV
  CPACN3_CPVRX_0V8125 ： 输出0.8125mV
  CPACN3_CPVRX_0V750 ： 输出0.750mV
  CPACN3_CPVRX_0V6875 ： 输出0.6875mV
  CPACN3_CPVRX_0V625 ： 输出0.625mV
  CPACN3_CPVRX_0V5625 ： 输出0.5625mV
  CPACN3_CPVRX_0V4375 ： 输出0.4375mV
  CPACN3_CPVRX_0V375 ： 输出0.375mV
  CPACN3_CPVRX_0V3125 ： 输出0.3125mV
  CPACN3_CPVRX_0V250 ： 输出0.250mV
  CPACN3_CPVRX_0V1875 ： 输出0.1875mV
  CPACN3_CPVRX_0V125 ： 输出0.125mV
  CPACN3_CPVRX_0V0625 ： 输出0.0625mV
  CPACN3_CPVRX_HIZ ： 高阻抗

- **包含头文件**

  Driver/ECPA.h

- **函数返回值**

  无

- **函数用法**

  /* 设置参考电压发生器II的输出电压为0.375mV */

  ECPA_RefVolChan2(CPACN3_CPVRX_0V375);

10. 低杂讯放大器 LNOP1/LNOP2

10.1 功能简介

该函数部分描述低杂讯放大器 LNOP1/LNOP2 功能的操作

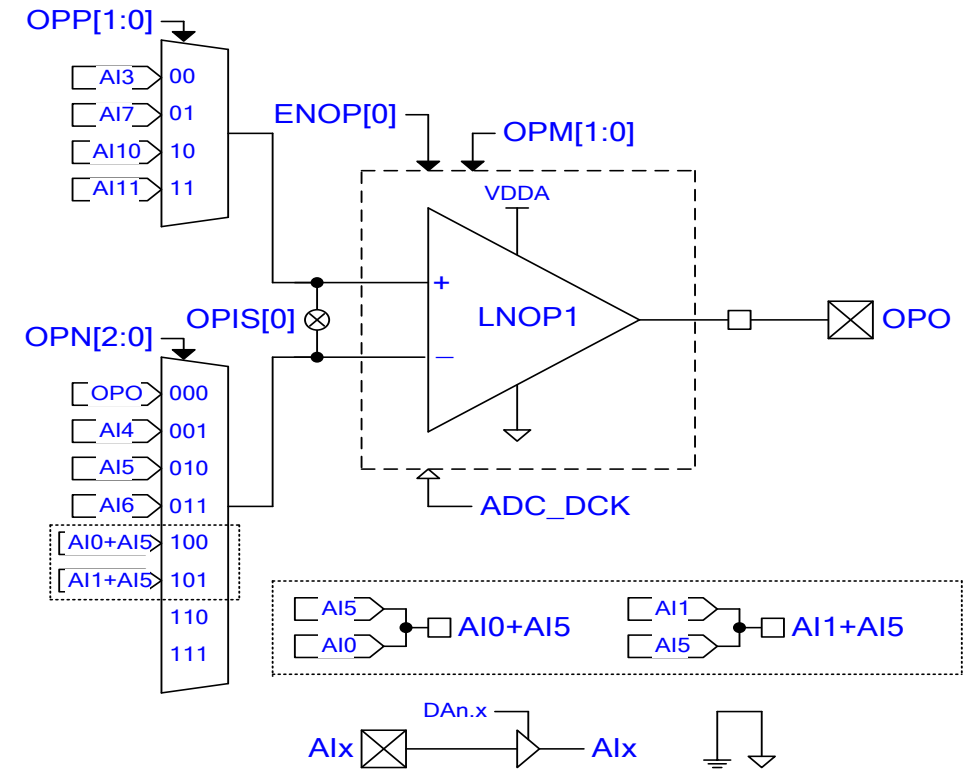
--LNOP1/LNOP1 功能的开关

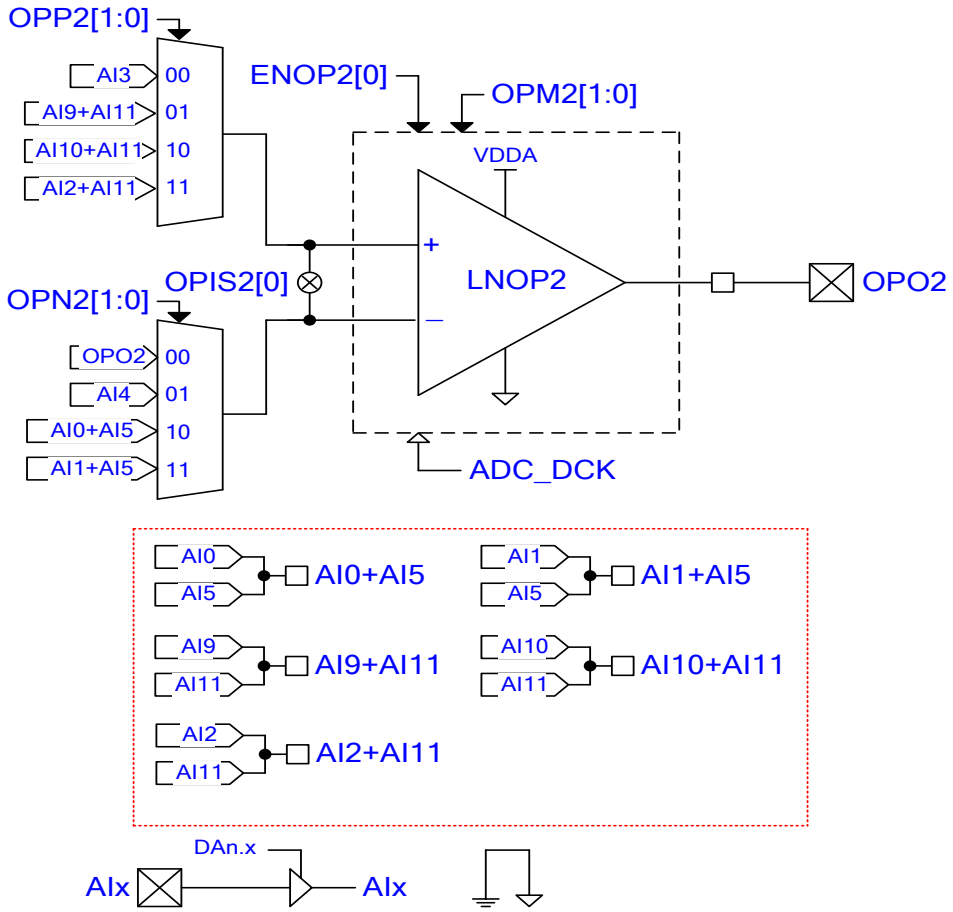
--LNOP1/LNOP2 输入端口及输出的控制

--包含 LNOP1.h/LNOP2.h

序号	函数名称	函数功能
01	LNOP1_Open	开启低杂讯运算放大器(LNOP1)功能，设置LNOP1输入端网络及chooper功能
02	LNOP1_Enable	开启低杂讯运算放大器(LNOP1)功能
03	LNOP1_Disable	关闭低杂讯运算放大器(LNOP1)功能
04	LNOP1_OPMode	设置低杂讯运算放大器（LNOP1）输入处理器模式
05	LNOP1_OPChanIn	设置低杂讯运算放大器（LNOP1）输入端短路功能
06	LNOP1_OPInputP	设置低杂讯运算放大器（LNOP1）正向输入端通道
07	LNOP1_OPInputN	设置低杂讯运算放大器（LNOP1）负向输入端通道
08	LNOP2_Open	开启低杂讯运算放大器(LNOP2)功能，设置LNOP2输入端网络及chooper功能
09	LNOP2_Enable	开启低杂讯运算放大器(LNOP2)功能
10	LNOP2_Disable	关闭低杂讯运算放大器(LNOP2)功能
11	LNOP2_OPMode	设置低杂讯运算放大器（LNOP2）输入处理器模式
12	LNOP2_OPChanIn	设置低杂讯运算放大器（LNOP2）输入端短路功能
13	LNOP2_OPInputP	设置低杂讯运算放大器（LNOP2）正向输入端通道
14	LNOP2_OPInputN	设置低杂讯运算放大器（LNOP2）负向输入端通道

10.2 LNOP1/LN OP2 模块方框图





10.3 函数说明

注意：使用LNOP1需要先开启VDDA/ACM电压！

10.3.1 LNOP1_Open

- 函数

  void LNOP1_Open(unsigned char NInput, unsigned char PInput, unsigned char Chop);

- 函数功能

  开启低杂讯运算放大器(LNOP1)功能，设置LNOP1输入端网络及chooper功能，设置寄存器OPCN1[7:0]。

- 输入参数

  NInput[in]：设置LNOP1负向输入通道

  OPCN1_OPN_AI1 ：AI1—AI5，AI1与AI5短路，但不与AI0短路

  OPCN1_OPN_AI0 ：AI0—AI5，AI0与AI5短路，但不与AI1短路

  OPCN1_OPN_AI6 ：AI6

  OPCN1_OPN_AI5 ：AI5

  OPCN1_OPN_AI4 ：AI4

  OPCN1_OPN_OPO：OPO

  PInput[in]：设置LNOP1正向输入通道

  OPCN1_OPP_AI11：AI11

  OPCN1_OPP_AI10：AI10

  OPCN1_OPP_AI7 ：AI7

  OPCN1_OPP_AI3 ：AI3

  Chop [in]：设置LNOP1输入处理器模式

  OPCN1_OPM_REVERSE ：输入的反向偏移量

  OPCN1_OPM_FORWARD ：输入的正向偏移量

  OPCN1_OPM_CHOPPER128 ：Chooper，Frequency as ADC_CK/128

  OPCN1_OPM_CHOPPER64 ：Chooper，Frequency as ADC_CK/64

- 包含头文件

  Driver/LNOP1.h

- 函数返回值

  无

- 函数用法

```
/* 设置LNOP1的输入端为AI3—AI4，且为输入正向偏移量模式 */
PWR_Open( PWRCN_VDDAX_2V4, PWRCN_ENACM_ENABLE, 0xFF );      //启动VDDA/ACM
LNOP1_Open( OPCN1_OPN_AI4, OPCN1_OPP_AI3, OPCN1_OPM_REVERSE );
```

10.3.2 LNOP1_Enable

- 函数

  LNOP1_Enable();

- 函数功能

  开启低杂讯运算放大器(LNOP1)功能，设置寄存器OPCN1[7] =1 。

- 输入参数

  无

- 包含头文件

  Driver/LNOP1.h

- 函数返回值

  无

- 函数用法

  /* 开启低杂讯运算放大器(LNOP1) */

  LNOP1_Enable();

10.3.3 LNOP1_Disable

- 函数

  LNOP1_Disable();

- 函数功能

  关闭低杂讯运算放大器(LNOP1)功能，设置寄存器OPCN1[7]=0 。

- 输入参数

  无

- 包含头文件

  Driver/LNOP1.h

- 函数返回值

  无

- 函数用法

  /* 关闭低杂讯运算放大器(LNOP1)功能 */

  LNOP1_Disable();

10.3.4 LNOP1_OPMode

- 函数

  LNOP1_OPMode(ModeSel);

- 函数功能

  设置低杂讯运算放大器（LNOP1）输入处理器模式，设置寄存器OPCN1[5:4]。

- 输入参数

  ModeSel [in]：设置LNOP1输入处理器模式

  OPCN1_OPM_REVERSE ：输入的反向偏移量

  OPCN1_OPM_FORWARD ：输入的正向偏移量

OPCN1_OPM_CHOPPER128 ： Chooper，Frequency as ADC_CK/128

OPCN1_OPM_CHOPPER64 ： Chooper，Frequency as ADC_CK/64

- 包含头文件

Driver/LNOP1.h

- 函数返回值

无

- 函数用法

```
/* 设置LNOP1的输入处理为输入正向偏移量 */
LNOP1_OPMode( OPCN1_OPM_FORWARD );
```

10.3.5 LNOP1_OPChanIn

- 函数

LNOP1_OPChanIn(OPISel);

- 函数功能

设置低杂讯运算放大器（LNOP1）输入端短路功能，设置寄存器AINET1[0]。

- 输入参数

OPISel [in]：设置LNOP1的输入端短路功能

AINET1_OPIS_ENABLE ： 输入端短路

AINET1_OPIS_DISABLE ： 输入端不短路

- 包含头文件

Driver/LNOP1.h ， Driver/ADC.h

- 函数返回值

无

- 函数用法

```
/* 设置LNOP1输入端短路 */
LNOP1_OPChanIn( AINET1_OPIS_ENABLE );
```

10.3.6 LNOP1_OPInputP

- 函数

LNOP1_OPInputP(OPPSel);

- 函数功能

设置低杂讯运算放大器（LNOP1）正向输入端通道，设置寄存器OPCN1[4:3]。

- 输入参数

OPPSel [in]：设置LNOP1正向输入端通道

OPCN1_OPP_AI11：AI11

OPCN1_OPP_AI10：AI10

OPCN1_OPP_AI7 ： AI7

OPCN1_OPP_AI3 ： AI3

- 包含头文件

  Driver/LNOP1.h

- 函数返回值

  无

- 函数用法

  /* 设置LNOP1的正向输入通道为AI3 */

  LNOP1_OPInputP(OPCN1_OPP_AI3);

10.3.7 LNOP1_OPInputN

- 函数

  LNOP1_OPInputN(OPNSel);

- 函数功能

  设置低杂讯运算放大器（LNOP1）的负向输入端通道，设置寄存器OPCN1[2:0]。

- 输入参数

  OPNSel [in]：设置LNOP1的负向输入端通道

  OPCN1_OPN_AI1 ： AI1—AI5，AI1与AI5短路，但不与AI0短路

  OPCN1_OPN_AI0 ： AI0—AI5，AI0与AI5短路，但不与AI1短路

  OPCN1_OPN_AI6 ： AI6

  OPCN1_OPN_AI5 ： AI5

  OPCN1_OPN_AI4 ： AI4

  OPCN1_OPN_OPO ： OPO

- 包含头文件

  Driver/LNOP1.h

- 函数返回值

  无

- 函数用法

  /* 设置运算放大器 的负向输入端通道为AI4 */

  LNOP1_OPInputN(OPCN1_OPN_AI4);

10.3.8 LNOP2_Open

- 函数

  void LNOP2_Open(unsigned char Ninput, unsigned char PInput, unsigned char Chop);

- 函数功能

  开启低杂讯运算放大器(LNOP2)功能，设置LNOP2输入端网络及chooper功能，设置寄存器OPCN2[7:0]。

- 输入参数

  Ninput [in]：设置LNOP2负向输入通道

  OPCN2_OPN2_AI1 ： AI1

  OPCN2_OPN2_AI0 ： AI0

OPCN2_OPN2_AI4 ：AI4

OPCN2_OPN2_OPO2 ：OPO2

Pinput [in]：设置LNOP2正向输入通道

OPCN2_OPP2_AI2 ：AI2

OPCN2_OPP2_AI10：AI10

OPCN2_OPP2_AI9 ：AI9

OPCN2_OPP2_AI3 ：AI3

Chop [in]：设置LNOP2输入处理器模式

OPCN2_OPM2_REVERSE ：输入的反向偏移量

OPCN2_OPM2_FORWARD ：输入的正向偏移量

OPCN2_OPM2_CHOPPER128 ：Chooper，Frequency as ADC_CK/128

OPCN2_OPM2_CHOPPER64 ：Chooper，Frequency as ADC_CK/64

- **包含头文件**

  Driver/LNOP2.h

- **函数返回值**

  无

- **函数用法**

```
/* 设置LNOP2的输入端为AI3—AI4，且为输入正向偏移量模式 */
PWR_Open( PWRCN_VDDAX_2V4, PWRCN_ENACM_ENABLE, 0xFF );          //启动VDDA/ACM
LNOP2_Open( OPCN2_OPN2_AI4, OPCN2_OPP2_AI3, OPCN2_OPM2_REVERSE );
```

10.3.9 LNOP2_Enable

- **函数**

  LNOP2_Enable();

- **函数功能**

  开启低杂讯运算放大器(LNOP2)功能，设置寄存器OPCN2[7]=1 。

- **输入参数**

  无

- **包含头文件**

  Driver/LNOP2.h

- **函数返回值**

  无

- **函数用法**

```
/* 开启低杂讯运算放大器(LNOP2) */
LNOP2_Enable();
```

10.3.10 LNOP2_Disable

- **函数**

LNOP2_Disable();

- 函数功能

关闭低杂讯运算放大器(LNOP2)功能，设置寄存器OPCN2[7]=0 。

- 输入参数

无

- 包含头文件

Driver/LNOP2.h

- 函数返回值

无

- 函数用法

```
/* 关闭低杂讯运算放大器(LNOP2)功能 */
LNOP2_Disable();
```

10.3.11 LNOP2_OPMode

- 函数

LNOP2_OPMode(ModeSel);

- 函数功能

设置低杂讯运算放大器（LNOP2）输入处理器模式，设置寄存器OPCN2[5:4]。

- 输入参数

ModeSel [in]：设置LNOP2输入处理器模式

OPCN2_OPM2_REVERSE	：	输入的反向偏移量
OPCN2_OPM2_FORWARD	：	输入的正向偏移量
OPCN2_OPM2_CHOPPER128	：	Chooper，Frequency as ADC_CK/128
OPCN2_OPM2_CHOPPER64	：	Chooper，Frequency as ADC_CK/64

- 包含头文件

Driver/LNOP2.h

- 函数返回值

无

- 函数用法

```
/* 设置LNOP2的输入处理为输入正向偏移量 */
LNOP2_OPMode( OPCN2_OPM2_FORWARD );
```

10.3.12 LNOP2_OPChanIn

- 函数

LNOP2_OPChanIn(OPISel);

- 函数功能

设置低杂讯运算放大器（LNOP2）输入端短路功能，设置寄存器OPCN2[4]。

- 输入参数

OPISel[in]：设置LNOP2的输入端短路功能

OPCN2_OPIS2_ENABLE ：输入端短路

OPCN2_OPIS2_DISABLE ：输入端不短路

- 包含头文件

Driver/LNOP2.h

- 函数返回值

无

- 函数用法

```
/* 设置LNOP2输入端短路 */
LNOP2_OPChanIn( OPCN2_OPIS2_ENABLE );
```

10.3.13 LNOP2_OPInputP

- 函数

LNOP2_OPInputP(OPPSel) ;

- 函数功能

设置低杂讯运算放大器（LNOP2）正向输入端通道，设置寄存器OPCN2[3:2]。

- 输入参数

OPPSel [in]：设置LNOP2正向输入端通道

OPCN2_OPP2_AI2 ：AI2

OPCN2_OPP2_AI10：AI10

OPCN2_OPP2_AI9 ：AI9

OPCN2_OPP2_AI3 ：AI3

- 包含头文件

Driver/LNOP2.h

- 函数返回值

无

- 函数用法

```
/* 设置LNOP2的正向输入通道为AI3 */
LNOP2_OPInputP( OPCN2_OPP2_AI3 );
```

10.3.14 LNOP2_OPInputN

- 函数

LNOP2_OPInputN(OPNSel);

- 函数功能

设置低杂讯运算放大器（LNOP2）的负向输入端通道，设置寄存器OPCN2[1:0]。

- 输入参数

OPNSel [in] 设置LNOP2的负向输入端通道

OPCN2_OPN2_AI1 ：AI1

OPCN2_OPN2_AI0 ： AI0

OPCN2_OPN2_AI4 ： AI4

OPCN2_OPN2_OPO2 ： OPO2

- 包含头文件

Driver/LNOP1.h

- 函数返回值

无

- 函数用法

```
/* 设置运算放大器 的负向输入端通道为AI4 */
LNOP2_OPInputN( OPCN2_OPN2_AI4 );
```

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及低电压检测功能，包含：

--VDDA 电压的控制

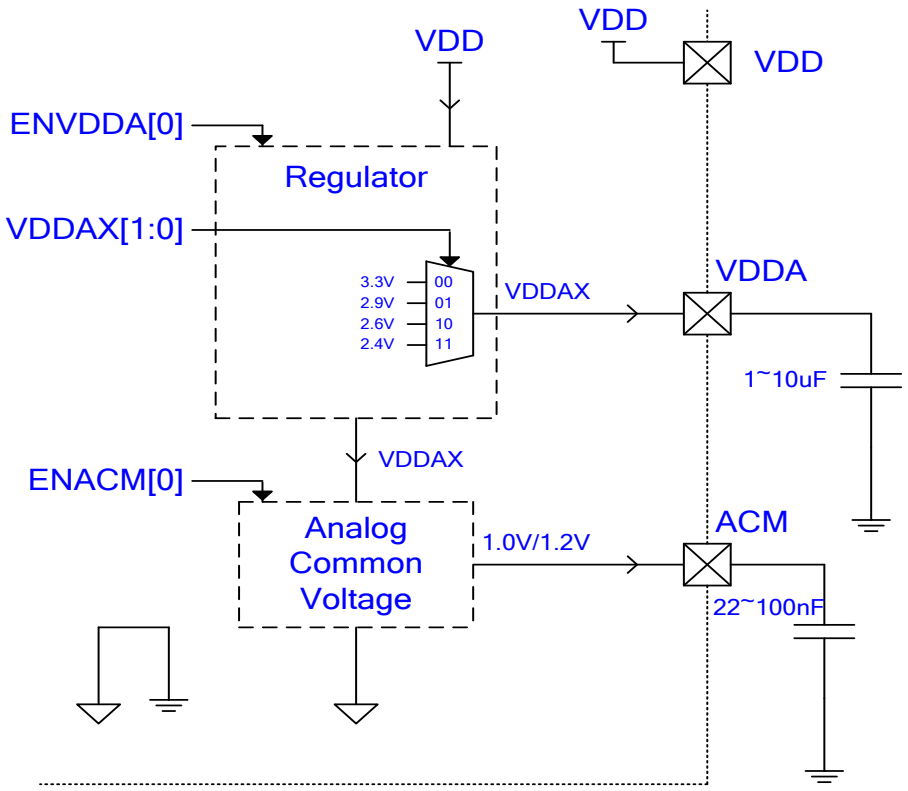
--ACM电压的控制

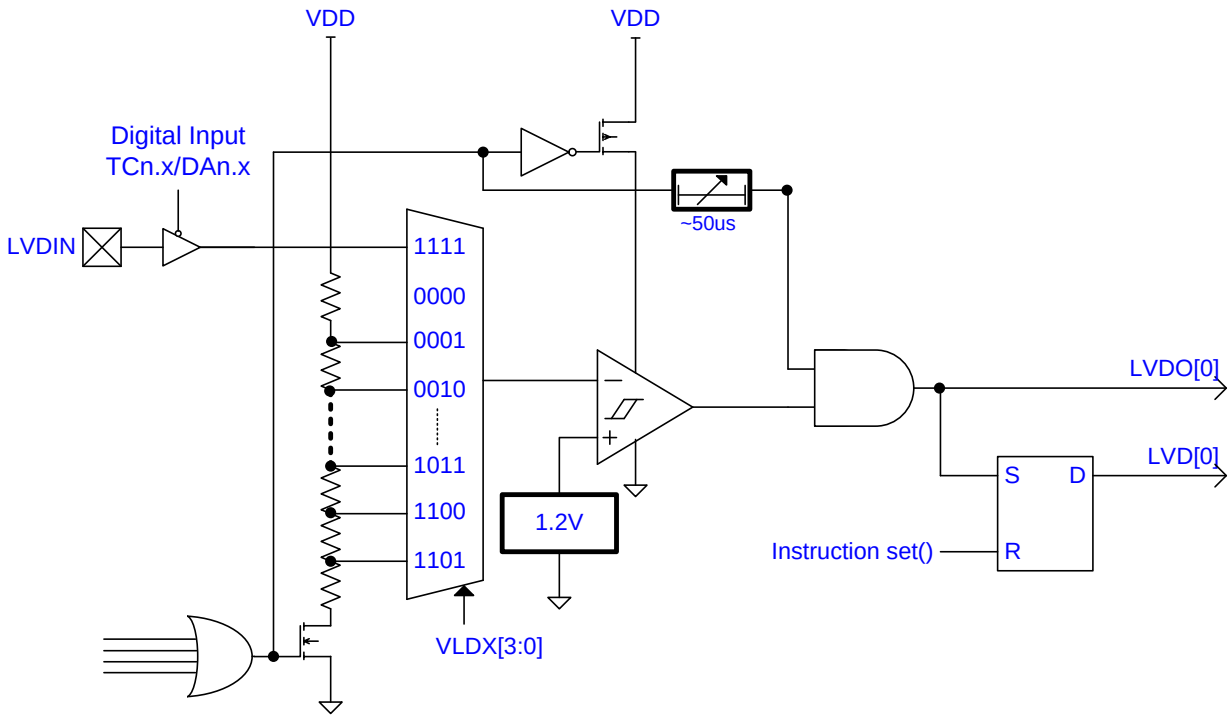
--低电压检测功能控制

--包含头文件PWR.h/LVD.h

序号	函数名称	功能描述
01	PWR_Open	使能内部ACM和VDDA，及设置VDDA输出电压值
02	PWR_ACMEnable	启动内部模拟地ACM电压
03	PWR_ACMDisable	关闭内部模拟地ACM电压
04	PWR_VDDAEnable	启动内部LDO产生VDDA电压输出
05	PWR_VDDADisable	关闭内部LDO产生VDDA电压功能
06	PWR_VDDASel	设置内部VDDA输出电压值
07	LVD_Open	使能低电压检测功能及设置低电压检测电压点
08	LVD_FGClr	清除低电压发生记录旗标LVDFG
09	LVD_IsFlag	读取低电压检测发生记录标志位LVDFG
10	LVD_GetStatus	读取低电压检测反应标志位LVD
11	LVD_GetLVDON	读取低电压检测稳定标志位LVDON

11.2 PowerManage & LVD 模块方框图





11.3 函数说明

11.4.1 PWR_Open

- **函数**

  void PWR_Open(unsigned char VDDASel, unsigned char ACMEn, unsigned char DelayCnt);

- **函数功能**

  使能IC内部模拟地ACM及VDDA，及设置VDDA输出电压值，设置寄存器PWRCN[7:4]。

- **输入参数**

  VDDASel [in]：设置VDDA电压

  PWRCN_VDDAX_2V4 : VDDA输出2.4V

  PWRCN_VDDAX_2V6 : VDDA输出2.6V

  PWRCN_VDDAX_2V9 : VDDA输出2.9V

  PWRCN_VDDAX_3V3 : VDDA输出3.3V

  ACMEn [in]：设置ACM电压

  PWRCN_ENACM_ENABLE ：启动ACM电压

  PWRCN_ENACM_DISABLE ：关闭ACM电压

  DelayCnt [in] 设置VDDA电压稳定时间，输入范围0x00~0xFF

- **包含头文件**

  Driver/PWR.h

- **函数返回值**

无

- 函数用法

/* 设置VDDA =2.6V. 并启动ACM电压，VDDA稳定时间为0xFF */

PWR_Open(PWRCN_VDDAX_2V6, PWRCN_ENACM_ENABLE, 0xFF);

11.3.2 PWR_ACMEnable

- 函数

PWR_ACMEnable();

- 函数功能

启动内部模拟地ACM电压，设置寄存器PWRCN[4]=1 。

- 输入参数

无

- 包含头文件

Driver/PWR.h

- 函数返回值

无

- 函数用法

/* 启动ACM电压 */

PWR_ACMEnable();

11.3.3 PWR_ACMDisable

- 函数

PWR_ACMDisable();

- 函数功能

关闭内部模拟地ACM电压，设置寄存器PWRCN[4]=0 。

- 输入参数

无

- 包含头文件

Driver/PWR.h

- 函数返回值

无

- 函数用法

/* 关闭ACM电压 */

PWR_ACMDisable();

11.3.4 PWR_VDDAEnable

- 函数

PWR_VDDAEnable();

- 函数功能

启动内部VDDA电压输出，设置寄存器PWRCN[7]=1 。

- 输入参数

无

- 包含头文件

Driver/PWR.h

- 函数返回值

无

- 函数用法

```
/* 启动VDDA电压 */
PWR_VDDAEnable();
Delay( 0xFF ) ;
```

11.3.5 PWR_VDDADisable

- 函数

PWR_VDDADisable() ;

- 函数功能

关闭内部VDDA电压功能，设置寄存器PWRCN[7]=0 。

- 输入参数

无

- 包含头文件

Driver/PWR.h

- 函数返回值

无

- 函数用法

```
/* 关闭VDDA电压输出 */
PWR_VDDADisable() ;
```

11.3.6 PWR_VDDASel

- 函数

PWR_VDDASel(VDDAVSel) ;

- 函数功能

设置内部VDDA输出电压值，设置寄存器PWRCN[6 :5]=0 。

- 输入参数

VDDASel [in]：设置VDDA电压

PWRCN_VDDAX_2V4 : VDDA输出2.4V

PWRCN_VDDAX_2V6 : VDDA输出2.6V

PWRCN_VDDAX_2V9 : VDDA输出2.9V

PWRCN_VDDAX_3V3 : VDDA输出3.3V

- 包含头文件

Driver/PWR.h

- 函数返回值

无

- 函数用法

```
/* 启动VDDA输出2.4V */
PWR_VDDASel( VDDAVSel );
PWE_VDDAEnable();
Delay( 0xFF ) ;
```

11.3.7 LVD_Open

- 函数

LVD_Open(LVDSel) ;

- 函数功能

设置低电压检测电压点，设置寄存器LVDCN[3 :0]。

- 输入参数

LVDSel [in]：设置电压检测点

LVDCN_VLDX_LVDIN ：待检测电压由PT1.2输入，检测电压点位1.2V

LVDCN_VLDX_3V31 ：检测电压点为3.31V

LVDCN_VLDX_3V21 ：检测电压点为3.21V

LVDCN_VLDX_3V11 ：检测电压点为3.11V

LVDCN_VLDX_3V01 ：检测电压点为3.01V

LVDCN_VLDX_2V91 ：检测电压点为2.91V

LVDCN_VLDX_2V81 ：检测电压点为2.81V

LVDCN_VLDX_2V71 ：检测电压点为2.71V

LVDCN_VLDX_2V61 ：检测电压点为2.61V

LVDCN_VLDX_2V51 ：检测电压点为2.51V

LVDCN_VLDX_2V41 ：检测电压点为2.41V

LVDCN_VLDX_2V31 ：检测电压点为2.31V

LVDCN_VLDX_2V21 ：检测电压点为2.21V

LVDCN_VLDX_2V11 ：检测电压点为2.11V

LVDCN_VLDX_2V01 ：检测电压点为2.01V

LVDCN_VLDX_LVDOFF ：关闭低电压检测功能

- 包含头文件

Driver/LVD.h

- 函数返回值

  无

- 函数用法

  /* 设置LVD检测电压点为2.61V */

  LVD_Open(LVDCN_VLDX_2V61);

11.3.8 LVD_FGClr

- 函数

  LVD_FGClr() ;

- 函数功能

  清除低电压发生记录旗标LVDFG，设置寄存器LVDCN[6]=0 。

- 输入参数

  无

- 包含头文件

  Driver/LVD.h

- 函数返回值

  无

- 函数用法

  /* 清除低电压发生记录旗标 */

  LVD_FGClr();

11.3.9 LVD_IsFlag

- 函数

  LVD_IsFlag();

- 函数功能

  读取低电压检测发生记录标志位LVDFG，读取寄存器LVDCN[6]。

- 输入参数

  无

- 包含头文件

  Driver/LVD.h

- 函数返回值

  0x00：电压检测未发生过

  0x40：电压检测已发生过

- 函数用法

  /* 读取低电压检测发生记录标志位 */

  unsigned char flag；

  flag = LVD_IsFlag();

11.3.10 LVD_GetStatus

- 函数

  LVD_GetStatus() ;

- 函数功能

  读取低电压检测反应标志位LVD，读取寄存器LVD[5]。

- 输入参数

  无

- 包含头文件

  Driver/LVD.h

- 函数返回值

  0x00：未低电压

  0x20：低电压

- 函数用法

  /* 读取低电压反应标志位 */

  unsigned char flag；

  flag = LVD_GetStatus();

11.3.11 LVD_GetLVDON

- 函数

  LVD_GetLVDON() ;

- 函数功能

  读取低电压检测稳定标志位LVDON，读取寄存器LVD[4]。

- 输入参数

  无

- 包含头文件

  Driver/LVD.h

- 函数返回值

  0x00：未稳定

  0x10：稳定

- 函数用法

  /* 读取低电压检测稳定标志位LVDON */

  unsigned char flag；

  flag = LVD_GetLVDON();

12. 捕捉/比较器(CCP)

12.1 函数功能简介

该部分函数介绍捕捉/比较器（CCP）功能的设置

--CCP 功能的启动与关闭

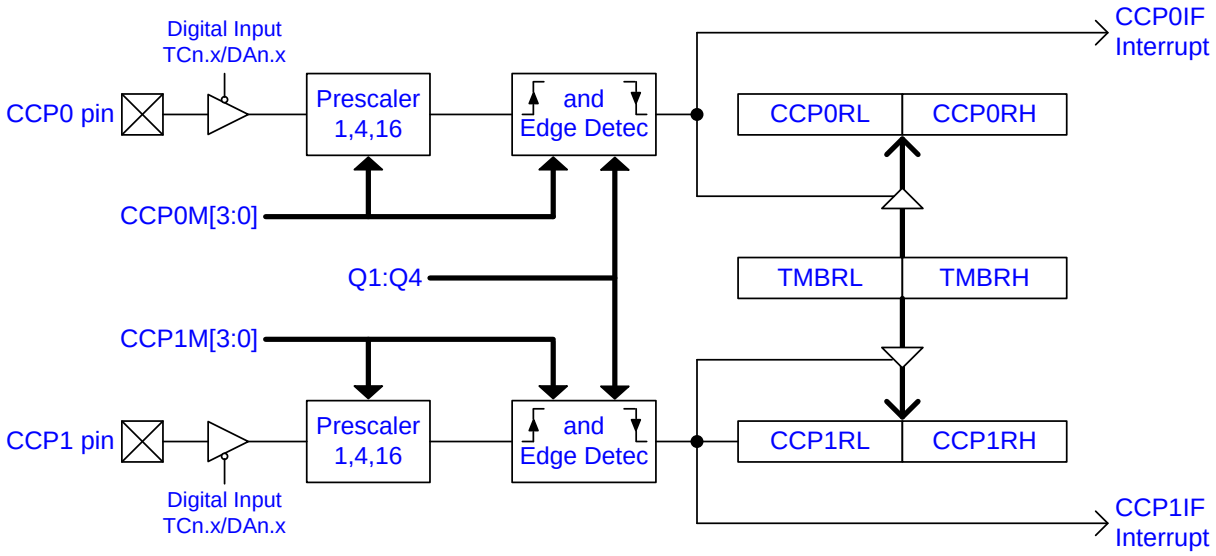
--CCP 工作模式及输入端口的设置

--CCP 中断功能控制

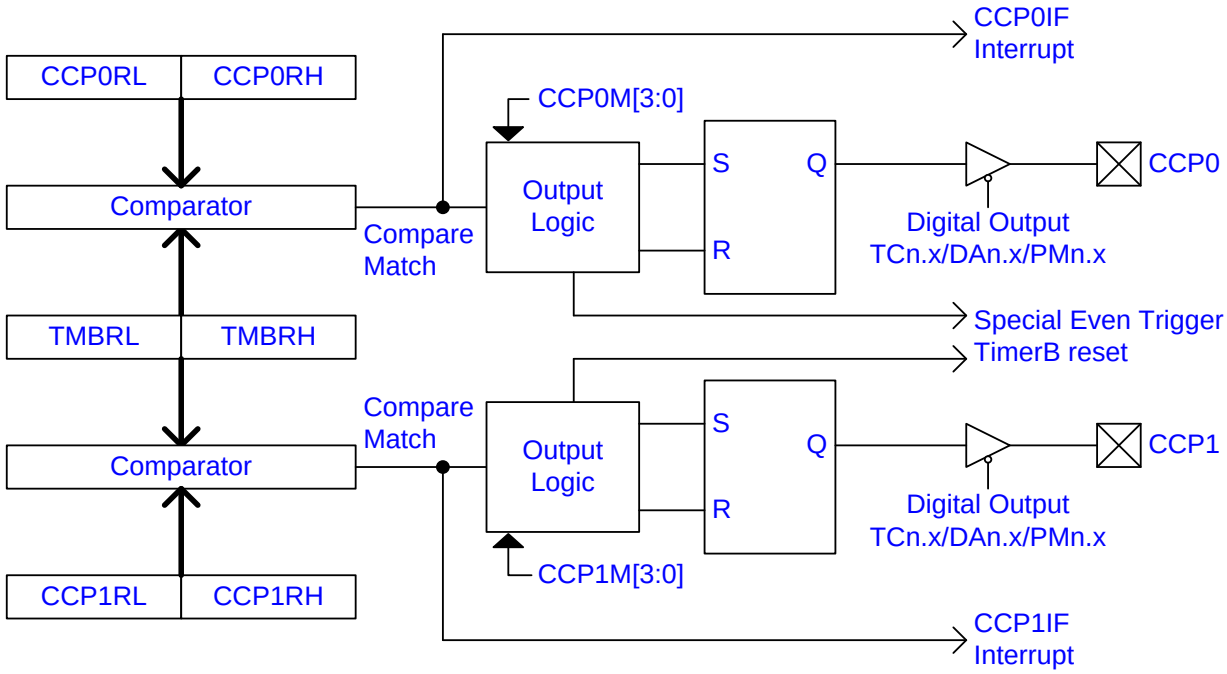
--包含 CCP.h

序号	函数名称	功能描述
01	CCP_Open	设置捕捉/比较器的工作模式
02	CCP_SetData0	设置比较暂存器CCP0R计数值
03	CCP_SetData1	设置比较暂存器CCP1R计数值
04	CCP_GetData0	读取捕捉器的计数器CCP0R的值
05	CCP_GetData1	读取捕捉器的计数器CCP1R的值
06	CCP_INT0_Enable	使能捕捉/比较器CCP0中断功能
07	CCP_INT0_Disable	关闭捕捉/比较器CCP0中断功能
08	CCP_INT0_IsFlag	读取捕捉/比较器CCP0的中断请求标志位
09	CCP_INT0_ClearFlag	清除捕捉/比较器CCP0的中断请求标志位
10	CCP_INT1_Enable	使能捕捉/比较器CCP1中断功能
11	CCP_INT1_Disable	关闭捕捉/比较器CCP1中断功能
12	CCP_INT1_IsFlag	读取捕捉/比较器CCP1的中断请求标志位
13	CCP_INT1_ClearFlag	清除捕捉/比较器CCP1的中断请求标志位
14	CCP_CCP1Mode	设置捕捉/比较器CCP1工作模式
15	CCP_CCP0Mode	设置捕捉/比较器CCP0工作模式

12.2 捕捉/比较器模块方框图



（a）捕捉器功能方框图



（b）比较器功能方框图

12.3 函数说明

12.3.1 CCP_Open

- **函数**

  void CCP_Open(unsigned char ccp0m, unsigned char ccp1m);

- **函数功能**

  设置捕捉/比较器的工作模式，设置寄存器CCPCN。

- **输入参数**

  ccp0m [in]：DAC 正向参考输入端选择

CCPCN_CCP0M_CLRTMB ：比较模式，事件成立CCP0IF置1并清零TMB计数值

CCPCN_CCP0M_NOCCP ：比较模式，事件成立CCP0IF置1，不送信号至CCP0引脚

CCPCN_CCP0M_LOWCCP ：比较模式，CCP0引脚初始为High，事件成立CCP0IF=1，CCP0为Low

CCPCN_CCP0M_HICCP ：比较模式，CCP0引脚初始为Low，事件成立CCP0IF=1，CCP0为High

CCPCN_CCP0M_RISE16 ：捕捉模式，每16个上升沿捕捉一次，事件成立CCP0IF=1

CCPCN_CCP0M_RISE4 ：捕捉模式，每4个上升沿捕捉一次，事件成立CCP0IF=1

CCPCN_CCP0M_RISE1 ：捕捉模式，每1个上升沿捕捉一次，事件成立CCP0IF=1

CCPCN_CCP0M_FALL1 ：捕捉模式，每1个下降沿捕捉一次，事件成立CCP0IF=1

CCPCN_CCP0M_REVCCP ：比较模式，事件成立CCP0IF置1，CCP0引脚输出电平反相

CCPCN_CCP0M_CLOSE ：关闭捕捉/比较器功能

  ccp1m [in]：DAC 负向参考输入端选择

CCPCN_CCP1M_CLRTMB ：比较模式，事件成立CCP1IF置1并清零TMB计数值

CCPCN_CCP1M_NOCCP ：比较模式，事件成立CCP1IF置1，不送信号至CCP1引脚

CCPCN_CCP1M_LOWCCP ：比较模式，CCP1引脚初始为High，事件成立CCP1IF=1，CCP1为Low

CCPCN_CCP1M_HICCP ：比较模式，CCP1引脚初始为Low，事件成立CCP1IF=1，CCP1为High

CCPCN_CCP1M_RISE16 ：捕捉模式，每16个上升沿捕捉一次，事件成立CCP1IF=1

CCPCN_CCP1M_RISE4 ：捕捉模式，每4个上升沿捕捉一次，事件成立CCP1IF=1

CCPCN_CCP1M_RISE1 ：捕捉模式，每1个上升沿捕捉一次，事件成立CCP1IF=1

CCPCN_CCP1M_FALL1 ：捕捉模式，每1个下降沿捕捉一次，事件成立CCP1IF=1

CCPCN_CCP1M_REVCCP ：比较模式，事件成立CCP1IF置1，CCP1引脚输出电平反相

CCPCN_CCP1M_CLOSE ：关闭捕捉/比较器功能

- **包含头文件**

  Driver/CCP.h

- **函数返回值**

  无

- **函数用法**

```
/* 设置CCP0为捕捉模式，且16个上升沿捕捉，CCP1为捕捉模式，且16个上升沿捕捉 */
CCP_Open( CCPCN_CCP0M_RISE16, CCPCN_CCP1M_RISE16 );
```

12.3.2 CCP_SetData0

- 函数

  void CCP_SetData0(unsigned int CCPdata);

- 函数功能

  设置比较暂存器CCP0R计数值，设置寄存器CCP0RH/CCP0RL。

- 输入参数

  CCPdata [in]：设置比较暂存器CCP0R的值，范围0x0~0xFFFF

- 包含头文件

  Driver/CCP.h

- 函数返回值

  无

- 函数用法

  /* 设置比较暂存器值CCP0R为0X7F7F */

  CCP_SetData0(0x7F7F);

12.3.3 CCP_SetData1

- 函数

  void CCP_SetData1(unsigned int CCPdata);

- 函数功能

  设置比较暂存器CCP1R计数值，设置寄存器CCP1RH/CCP1RL。

- 输入参数

  CCPdata [in]：设置比较暂存器CCP1R的值，范围0x0~0xFFFF

- 包含头文件

  Driver/CCP.h

- 函数返回值

  无

- 函数用法

  /* 设置比较暂存器值CCP1R为0X7F7F */

  CCP_SetData1(0x7F7F);

12.3.4 CCP_GetData0

- 函数

  int CCP_GetData0(void);

- 函数功能

  读取捕捉器的计数器CCP0R的值，读取寄存器CCP0RH/CCP0RL。

- 输入参数

  无

- 包含头文件

Driver/CCP.h

- **函数返回值**

反馈CCP0R:CCP0RL的值

- **函数用法**

```
/* 读取捕捉器的计数值CCP0R/CCP0RL */
int data;
data = CCP_GetData0();
```

12.3.5 CCP_GetData1

- **函数**

```
int CCP_GetData1(void);
```

- **函数功能**

读取捕捉器的计数器CCP1R的值，读取寄存器CCP1RH/CCP1RL。

- **输入参数**

无

- **包含头文件**

Driver/CCP.h

- **函数返回值**

反馈CCP1R:CCP1RL的值

- **函数用法**

```
/* 读取捕捉器的计数值CCP1R/CCP1RL */
int data;
data = CCP_GetData1();
```

12.3.6 CCP_INT0_Enable

- **函数**

```
CCP_INT0_Enable();
```

- **函数功能**

使能捕捉/比较器CCP0中断功能，设置寄存器INTE2[0]=1 。

- **输入参数**

无

- **包含头文件**

Driver/CCP.h

- **函数返回值**

无

- **函数用法**

```
/* 使能捕捉/比较器CCP0中断功能 */
CCP_INT0_Enable();
```

12.3.7 CCP_INT0_Disable

- 函数

  CCP_INT0_Disable();

- 函数功能

  关闭捕捉/比较器CCP0中断功能，设置寄存器INTE2[0]=0 。

- 输入参数

  无

- 包含头文件

  Driver/CCP.h

- 函数返回值

  无

- 函数用法

  /* 关闭捕捉/比较器CCP0中断功能 */

  CCP_INT0_Disable();

12.3.8 CCP_INT0_IsFlag

- 函数

  CCP_INT0_IsFlag();

- 函数功能

  读取捕捉/比较器CCP0的中断请求标志位，读取寄存器INTF2[0]。

- 输入参数

  无

- 包含头文件

  Driver/CCP.h

- 函数返回值

  0x00：捕捉/比较器CCP0未产生中断请求

  0x01：捕捉/比较器CCP0产生中断请求

- 函数用法

  /* 读取捕捉/比较器CCP0的中断请求标志位 */

  unsigned char flag;

  flag = CCP_INT0_IsFlag();

12.3.9 CCP_INT0_ClearFlag

- 函数

  CCP_INT0_ClearFlag();

- 函数功能

  清除捕捉/比较器CCP0的中断请求标志位，设置寄存器INTF2[0]=0 。

- 输入参数

无

- 包含头文件

Driver/CCP.h

- 函数返回值

无

- 函数用法

/* 清除捕捉/比较器CCP0的中断请求标志位 */

CCP_INT0_ClearFlag();

12.3.10 CCP_INT1_Enable

- 函数

CCP_INT1_Enable();

- 函数功能

使能捕捉/比较器CCP1中断功能，设置寄存器INTE2[1]=1 。

- 输入参数

无

- 包含头文件

Driver/CCP.h

- 函数返回值

无

- 函数用法

/* 使能捕捉/比较器CCP1中断功能 */

CCP_INT1_Enable();

12.3.11 CCP_INT1_Disable

- 函数

CCP_INT1_Disable();

- 函数功能

关闭捕捉/比较器CCP1中断功能，设置寄存器INTE2[1]=0 。

- 输入参数

无

- 包含头文件

Driver/CCP.h

- 函数返回值

无

- 函数用法

/* 关闭捕捉/比较器CCP1中断功能 */

CCP_INT1_Disable();

12.3.12 CCP_INT1_IsFlag

- 函数

  CCP_INT1_IsFlag();

- 函数功能

  读取捕捉/比较器CCP1的中断请求标志位，读取寄存器INTF2[1]。

- 输入参数

  无

- 包含头文件

  Driver/CCP.h

- 函数返回值

  0x00：捕捉/比较器CCP1未产生中断请求

  0x02：捕捉/比较器CCP1产生中断请求

- 函数用法

```
/* 读取捕捉/比较器CCP1的中断请求标志位 */
unsigned char flag;
flag = CCP_INT1_IsFlag();
```

12.3.13 CCP_INT1_IsFlag

- 函数

  CCP_INT1_ClearFlag();

- 函数功能

  清除捕捉/比较器CCP1的中断请求标志位，设置寄存器INTF2[1]=0 。

- 输入参数

  无

- 包含头文件

  Driver/CCP.h

- 函数返回值

  无

- 函数用法

```
/* 清除捕捉/比较器CCP1的中断请求标志位 */
CCP_INT1_ClearFlag();
```

12.3.14 CCP_CCP1Mode

- 函数

  CCP_CCP1Mode(CCP1Sel);

- 函数功能

  设置捕捉/比较器CCP1工作模式，设置寄存器CCPCN[7:4]。

- 输入参数

CCP1Sel [in]：DAC 负向参考输入端选择.

CCPCN_CCP1M_CLRTMB ：比较模式，事件成立CCP1IF置1并清零TMB计数值

CCPCN_CCP1M_NOCCP ：比较模式，事件成立CCP1IF置1，不送信号至CCP1引脚

CCPCN_CCP1M_LOWCCP ：比较模式，CCP1引脚初始为High，事件成立CCP1IF=1，CCP1为Low

CCPCN_CCP1M_HICCP ：比较模式，CCP1引脚初始为Low，事件成立CCP1IF=1，CCP1为High

CCPCN_CCP1M_RISE16 ：捕捉模式，每16个上升沿捕捉一次，事件成立CCP1IF=1

CCPCN_CCP1M_RISE4 ：捕捉模式，每4个上升沿捕捉一次，事件成立CCP1IF=1

CCPCN_CCP1M_RISE1 ：捕捉模式，每1个上升沿捕捉一次，事件成立CCP1IF=1

CCPCN_CCP1M_FALL1 ：捕捉模式，每1个下降沿捕捉一次，事件成立CCP1IF=1

CCPCN_CCP1M_REVCCP ：比较模式，事件成立CCP1IF置1，CCP1引脚输出电平反相

CCPCN_CCP1M_CLOSE ：关闭捕捉/比较器功能

- **包含头文件**

  Driver/CCP.h

- **函数返回值**

  无

- **函数用法**

```
/* 设置捕捉/比较器CCP1工作模式为16个上升捕捉 */
CCP_CCP1Mode( CCPCN_CCP1M_RISE16 );
```

12.3.15 CCP_CCP0Mode

- **函数**

  CCP_CCP0Mode(CCP0Sel);

- **函数功能**

  设置捕捉/比较器CCP0工作模式，设置寄存器CCPCN[3:0]。

- **输入参数**

  CCP0Sel [in]：DAC 负向参考输入端选择

CCPCN_CCP0M_CLRTMB ：比较模式，事件成立CCP0IF置1并清零TMB计数值

CCPCN_CCP0M_NOCCP ：比较模式，事件成立CCP0IF置1，不送信号至CCP0引脚

CCPCN_CCP0M_LOWCCP ：比较模式，CCP0引脚初始为High，事件成立CCP0IF=1，CCP0为Low

CCPCN_CCP0M_HICCP ：比较模式，CCP0引脚初始为Low，事件成立CCP0IF=1，CCP0为High

CCPCN_CCP0M_RISE16 ：捕捉模式，每16个上升沿捕捉一次，事件成立CCP0IF=1

CCPCN_CCP0M_RISE4 ：捕捉模式，每4个上升沿捕捉一次，事件成立CCP0IF=1

CCPCN_CCP0M_RISE1 ：捕捉模式，每1个上升沿捕捉一次，事件成立CCP0IF=1

CCPCN_CCP0M_FALL1 ：捕捉模式，每1个下降沿捕捉一次，事件成立CCP0IF=1

CCPCN_CCP0M_REVCCP ：比较模式，事件成立CCP0IF置1，CCP0引脚输出电平反相

CCPCN_CCP0M_CLOSE ：关闭捕捉/比较器功能

- **包含头文件**

  Driver/CCP.h

- **函数返回值**

  无

- **函数用法**

  /* 设置捕捉/比较器CCP0工作模式为16个上升捕捉 */

  CCP_CCP0Mode (CCPCN_CCP0M_RISE16);

13. LCD 显示驱动器

13.1 函数简介

该部分函数描述 LCD 驱动器相关设置

--LCD 驱动器的时钟频率设置

--LCD 驱动器的偏置电压的设置及 duty 设置

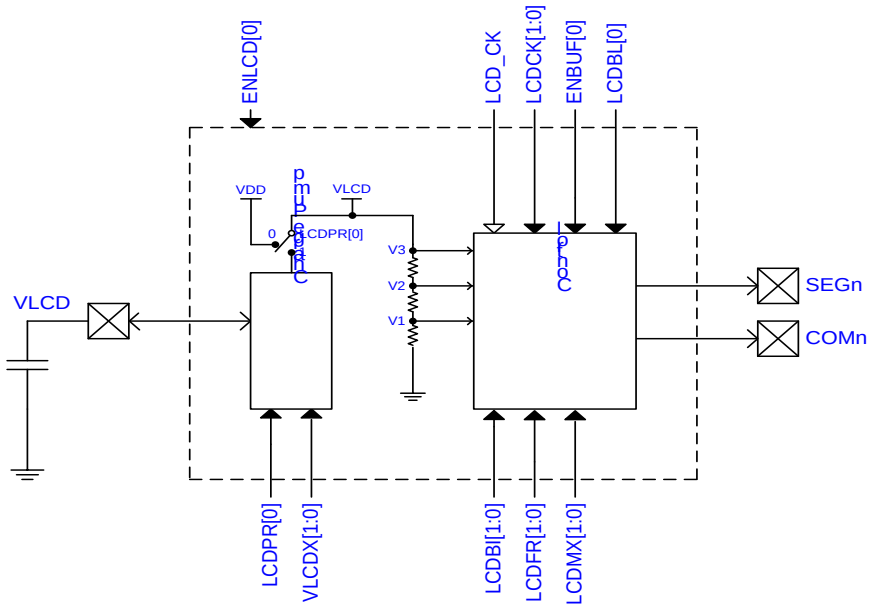
--LCD 驱动器显示功能设置

--LCD 驱动器显示数据的写入及输出缓冲器设置

--包含 LCD.h

序号	函数名称	功能描述
01	LCD_Open	使能LCD功能；设置LCD工作频率，设置LCD偏置电压，设置LCD驱动电压，设置LCD波形驱动模式及使能LCD输出缓冲器
02	LCD_Disable	关闭LCD功能
03	LCD_Enable	使能LCD功能
04	LCD_OutBufferEnable	使能LCD输出缓冲器
05	LCD_OutBufferDisable	关闭LCD输出缓冲器
06	LCD_DisplayOn	设置LCD显示全亮
07	LCD_DisplayOff	设置LCD显示全灭
08	LCD_CLKSel	设置LCD工作频率
09	LCD_ChargePumpConfig	LCD的驱动电压VLCD电压源设置
10	LCD_ChargePumpSelect	VLCD输出电压点设置
11	LCD_BiasInput	LCD波形偏压设置
12	LCD_DutyMode	LCD驱动波形设置
13	LCD_WriteData	向LCD显示寄存器写入显示值
14	LCD_ReadData	读取LCD显示寄存器的显示值

13.2 LCD 驱动器功能方框图



13.3 函数说明

13.3.1 LCD_Open

- 函数

void LCD_Open(unsigned char lcdpr, unsigned char ckdiv, unsigned char vlcdx,
unsigned char lcdmx, unsigned char lcdbi);

- 函数功能

使能LCD功能，设置LCD工作频率，设置LCD偏置电压，设置LCD驱动电压，设置LCD波形驱动模式及使能LCD输出缓冲器，设置寄存器MCKCN3[7:5]/LCDCN1/LCDCN2。

- 输入参数

lcdpr [in]：LCD驱动电压VLCD电压源设置

LCDCN1_LCDPR_INTERNAL ：VLCD电压源由内部产生

LCDCN1_LCDPR_EXTERNAL ：VLCD电压源由外部供应

vlcdx [in]：设置内部产生的VLCD电压值

LCDCN1_VLCDX_2V55 ：VLCD=2.55V

LCDCN1_VLCDX_2V8 ：VLCD=2.80V

LCDCN1_VLCDX_3V05 ：VLCD=3.05V

LCDCN1_VLCDX_3V3 ：VLCD=3.30V

ckdiv [in]：LCD工作频率源设置

MCKCN3_LCDS_PERACKDIV128 ：LCDS_CK=PERA_CK/128

MCKCN3_LCDS_PERACKDIV64 ：LCDS_CK=PERA_CK/64

MCKCN3_LCDS_PERACKDIV32 ：LCDS_CK=PERA_CK/32

MCKCN3_LCDS_PERACKDIV16 ：LCDS_CK=PERA_CK/16

MCKCN3_LCDS_PERACKDIV8 ：LCDS_CK=PERA_CK/8

MCKCN3_LCDS_PERACKDIV4 ：LCDS_CK=PERA_CK/4

MCKCN3_LCDS_PERACKDIV2 ：LCDS_CK=PERA_CK/2

MCKCN3_LCDS_PERACKDIV1 ：LCDS_CK=PERA_CK/1

lcdmx [in]：LCD驱动波形设置

LCDCN2_LCDMX_Static：固定状态（COM0）

LCDCN2_LCDMX_duty2：1/2 duty,(COM0，COM1)，COM3=SEG1，COM2=SEG0

LCDCN2_LCDMX_duty3：1/3 duty,（COM0，COM1，COM2）,COM3=SEG1

LCDCN2_LCDMX_duty4：1/4 duty,（COM0，COM1，COM2，COM3）

lcdbi [in]：LCD波形偏压设置

LCDCN1_LCDBI_Unused ：未使用

LCDCN1_LCDBI_BIAS ：1/3偏压

LCDCN1_LCDBI_Reserved ：保留

LCDCN1_LCDBI_Static ：静态操作

- 包含头文件

Drviver/LCD.h , Drviver/CLK.h

- 函数返回值

无

- 函数用法

```
/* 设置LCD工作频率为PERA_CK/4,VLCD=3.05V, 1/3BIAS,1/4duty */
LCD_Open( LCDCN1_LCDPR_INTERNAL, MCKCN3_LCDS_PERACKDIV4, LCDCN1_VLCDX_3V05,
          LCDCN2_LCDMX_duty4, LCDCN1_LCDBI_BIAS );
```

13.3.2 LCD_Disable

- 函数

LCD_Disable();

- 函数功能

关闭LCD功能，设置寄存器LCDCN1[7]=0 。

- 输入参数

无

- 包含头文件

Drviver/LCD.h

- 函数返回值

无

- 函数用法

```
/* 关闭LCD功能 */
LCD_Disable();
```

13.3.3 LCD_Enable

- 函数

LCD_Enable();

- 函数功能

使能LCD功能，设置寄存器LCDCN1[7]=1 。

- 输入参数

无

- 包含头文件

Drviver/LCD.h

- 函数返回值

无

- 函数用法

```
/* 使能LCD功能 */
LCD_Enable();
```

13.3.4 LCD_OutBufferEnable

- 函数

  LCD_OutBufferEnable();

- 函数功能

  使能LCD输出缓冲器，设置寄存器LCDCN1[3]=1 。

- 输入参数

  无

- 包含头文件

  Drviver/LCD.h

- 函数返回值

  无

- 函数用法

  /* 使能LCD输出缓冲器 */

  LCD_OutBufferEnable();

13.3.5 LCD_OutBufferDisable

- 函数

  LCD_OutBufferDisable();

- 函数功能

  关闭LCD输出缓冲器，设置寄存器LCDCN1[3]=0 。

- 输入参数

  无

- 包含头文件

  Drviver/LCD.h

- 函数返回值

  无

- 函数用法

  /* 关闭LCD输出缓冲器 */

  LCD_OutBufferDisable();

13.3.6 LCD_DisplayOn

- 函数

  LCD_DisplayOn();

- 函数功能

  设置LCD显示全亮，设置寄存器LCDCN2[7]=0 。

- 输入参数

  无

- 包含头文件

Drviver/LCD.h

- 函数返回值

无

- 函数用法

/* 设置LCD显示全亮 */

LCD_DisplayOn();

13.3.7 LCD_DisplayOff

- 函数

LCD_DisplayOff();

- 函数功能

设置LCD显示全灭，设置寄存器LCDCN2[7]=1 。

- 输入参数

无

- 包含头文件

Drviver/LCD.h

- 函数返回值

无

- 函数用法

/* 设置LCD显示全灭 */

LCD_DisplayOff();

13.3.8 LCD_CLKSel

- 函数

LCD_CLKSel(ClkSel);

- 函数功能

设置LCD工作频率，设置寄存器MCKCN3[7:5]。

- 输入参数

ClkSel [in]：LCD工作频率设置

MCKCN3_LCDS_PERACKDIV128 ： LCDS_CK=PERA_CK/128

MCKCN3_LCDS_PERACKDIV64 ： LCDS_CK=PERA_CK/64

MCKCN3_LCDS_PERACKDIV32 ： LCDS_CK=PERA_CK/32

MCKCN3_LCDS_PERACKDIV16 ： LCDS_CK=PERA_CK/16

MCKCN3_LCDS_PERACKDIV8 ： LCDS_CK=PERA_CK/8

MCKCN3_LCDS_PERACKDIV4 ： LCDS_CK=PERA_CK/4

MCKCN3_LCDS_PERACKDIV2 ： LCDS_CK=PERA_CK/2

MCKCN3_LCDS_PERACKDIV1 ： LCDS_CK=PERA_CK/1

- 包含头文件

Drviver/LCD.h

- 函数返回值

无

- 函数用法

/* 设置LCD工作频率为PERA_CK/4 */

LCD_CLKSel(MCKCN3_LCDS_PERACKDIV4);

13.3.9 LCD_ChargePumpConfig

- 函数

LCD_ChargePumpConfig(PRSel);

- 函数功能

LCD的驱动电压VLCD电压源设置，设置寄存器LCDCN1[6]。

- 输入参数

PRSel [in]：LCD驱动电压VLCD电压源设置

LCDCN1_LCDPR_INTERNAL ：VLCD电压源由内部产生

LCDCN1_LCDPR_EXTERNAL ：VLCD电压源由外部供应

- 包含头文件

Drviver/LCD.h

- 函数返回值

无

- 函数用法

/* 设置VLCD电压源为内部产生 */

LCD_ChargePumpConfig(LCDCN1_LCDPR_INTERNAL);

13.3.10 LCD_ChargePumpSelect

- 函数

LCD_ChargePumpSelect(VLCDSel);

- 函数功能

VLCD输出电压点设置，设置寄存器LCDCN1[5:4]。

- 输入参数

PRSel [in]：设置内部产生的VLCD电压值

LCDCN1_VLCDX_2V55 ：VLCD=2.55V

LCDCN1_VLCDX_2V8 ：VLCD=2.80V

LCDCN1_VLCDX_3V05 ：VLCD=3.05V

LCDCN1_VLCDX_3V3 ：VLCD=3.30V

- 包含头文件

Drviver/LCD.h

- 函数返回值

无

- 函数用法

/* 设置VLCD电压值为3.05V */

LCD_ChargePumpSelect(LCDCN1_VLCDX_3V05);

13.3.11 LCD_BiasInput

- 函数

LCD_BiasInput(BISel);

- 函数功能

LCD波形偏压设置，设置寄存器LCDCN1[2:1]。

- 输入参数

BISel [in]：LCD波形偏压设置

LCDCN1_LCDBI_Unused ：未使用

LCDCN1_LCDBI_BIAS ：1/3偏压

LCDCN1_LCDBI_Reserved ：保留

LCDCN1_LCDBI_Static ：静态操作

- 包含头文件

Drviver/LCD.h

- 函数返回值

无

- 函数用法

/* 设置LCD波形偏压为1/3 BIAS */

LCD_BiasInput(LCDCN1_LCDBI_BIAS);

13.3.12 LCD_DutyMode

- 函数

LCD_DutyMode(LCDMX);

- 函数功能

LCD驱动波形设置，设置寄存器LCDCN2[6:5]。

- 输入参数

LCDMX [in]：LCD驱动波形设置

LCDCN2_LCDMX_Static：固定状态（COM0）

LCDCN2_LCDMX_duty2：1/2 duty,(COM0，COM1)，COM3=SEG1，COM2=SEG0

LCDCN2_LCDMX_duty3：1/3 duty,（COM0，COM1，COM2）,COM3=SEG1

LCDCN2_LCDMX_duty4：1/4 duty,（COM0，COM1，COM2，COM3）

- 包含头文件

Drviver/LCD.h

- 函数返回值

无

- 函数用法

/* 设置LCD驱动波形为1/4 duty */

LCD_DutyMode(LCDCN2_LCDMX_duty4);

13.3.13 LCD_WriteData

- 函数

LCD_WriteData(uAddr,uData);

- 函数功能

向LCD显示寄存器写入显示值，设置寄存器

LCD0/LCD1/LCD2/LCD3/LCD4/LCD5/LCD6/LCD7/LCD8/LCD9/LCD10

LCD11/LCD12/LCD13/LCD14/LCD15/LCD16/LCD17/LCD18/LCD19 。

- 输入参数

uAddr [in]：显示寄存器地址

LCD0/LCD1/LCD2/LCD3/LCD4/LCD5/LCD6/LCD7/LCD8/LCD9/LCD10

LCD11/LCD12/LCD13/LCD14/LCD15/LCD16/LCD17/LCD18/LCD19

uData [in] 显示内容，0x00~0xFF

- 包含头文件

Drviver/LCD.h

- 函数返回值

无

- 函数用法

/* 往LCD0写入0XAA */

LCD_WriteData(LCD0，0XAA);

13.3.14 LCD_ReadData

- 函数

LCD_ReadData(uAddr);

- 函数功能

读取LCD显示寄存器的显示值，设置寄存器

LCD0/LCD1/LCD2/LCD3/LCD4/LCD5/LCD6/LCD7/LCD8/LCD9/LCD10

LCD11/LCD12/LCD13/LCD14/LCD15/LCD16/LCD17/LCD18/LCD19 。

- 输入参数

uAddr [in]：显示寄存器地址

LCD0/LCD1/LCD2/LCD3/LCD4/LCD5/LCD6/LCD7/LCD8/LCD9/LCD10

LCD11/LCD12/LCD13/LCD14/LCD15/LCD16/LCD17/LCD18/LCD19

- 包含头文件

Drviver/LCD.h

- **函数返回值**

  无

- **函数用法**

```
/* 读取LCD0显示值 */
unsigned char lcd_buf;
lcd_buf = LCD_ReadData( LCD0 );
```

14. Library

14.1 Library File

HY11P Driver C Library source code 在软体安装目录下的HY11P CIDE\Driver\HY11。

15. Revision History

Version	Page	Revision Summary	The Date Of Revision
V01	ALL	First edition	2017/11/10

16. C Library Change List

Date	旧版本 Queries List		新版本改善	
2017-11-10	版本	Bug List	版本	改善
	V01	无	V01	无