巨晟科技
Jusheng Technology

USER
MANUALS

# 微控制器 JSH3000 系列

## 库使用手册

V1.0


2019 版

www.honor-ic.com

集成电路设计及智能通信服务提供商

Integrated Circuit Design and Intelligent Communication Service Provider

此参考手册为巨晟公司用于 JSH3000 微控制器用户操作所用。版本若有更新，不另行通知。请打电话询问所购买销售人员。

技 术 创 新　　质 量 第 一


珠海巨晟科技股份有限公司

Zhuhai Jusheng Technology CO.,LTD

地址/Add：广东省珠海市香洲区金唐路 1 号港湾 1 号湾 8 栋 4 楼

4th Floor, 8th Building, No. 1 Harbour, No. 1 Jintang Road,
Xiangzhou District, Zhuhai City, Guangdong Province

电话/Tel ： 0756-3335384

传真/Fax ： 0756-3335384

**客户热线**

**0756-3335384**

# 修订历史记录

变更类型：A - 增加　M - 修订　D - 删除

| 变更版本号 | 日期 | 变更类型 | 修改人 | 审核 | 摘要 |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

# 版权声明

本资料是为了让用户根据用途选择合适的产品而提供的参考资料，不转让属于珠海巨晟科技股份有限公司或者第三方所有的知识产权以及其他权利的许可。在使用本资料所记载的信息并对有关产品是否适用做出最终判断前，请您务必将所有信息作为一个整体系统来评价。对于本资料所记载的信息使用不当而引起的损害、责任问题或者其他损失，珠海巨晟科技股份有限公司将不承担责任。未经珠海巨晟科技股份有限公司的许可，不得翻印或者复制全部或部分本资料的内容。

今后日常产品的更新会在适当的时候发布，恕不另行通知。在购买本资料所记载的产品时，请预先向珠海巨晟科技股份有限公司确认最新信息，并请您通过各种方式关注珠海巨晟科技股份有限公司公布的信息。

如果您需要了解有关本资料所记载的信息或产品的详情，请与珠海巨晟科技股份有限公司的技术服务部门联系，我们会为您提供全方位的技术支持。

# 商标声明

巨晟科技 Jusheng Technology是珠海巨晟科技股份有限公司注册商标， 未经事先书面许可， 任何人不得以任何方式用巨晟名称及巨晟的商标标记。

# 目录

# 1. 文档和库规范

本用户手册和固态函数库按照以下章节所描述的规范编写。

## 1.1 缩写

Table1

| 缩写 | 外设 |
| --- | --- |
| | |
| | |

## 1.2 命名规则

函数库遵从以下命名规则：

系统、源程序文件和头文件命名都以“jsh300x_ll_”作为开头，例如：jsh300x_ll_adkey.c。

常量仅被应用于一个文件的，定义于该文件中；被应用于多个文件的，在对应头文件中定义。寄存器的命名都由英文字母大写书写。外设函数的命名以该外设的缩写加下划线为开头。在函数名中，允许存在一个下划线用以分隔外设缩写和函数名的其它部分。

## 1.3 编码规则

### 1.3.1 变量

固态函数库定义了以下几个变量类型，他们的类型和大小是固定的。在文件 typedef.h 中我们定义了这些变量：

```
typedef int64_t          s64;
typedef int32_t          s32;
typedef int16_t          s16;
typedef int8_t           s8;
typedef const int32_t    sc32;          /*!< Read Only */
typedef const int16_t    sc16;          /*!< Read Only */
typedef const int8_t     sc8;           /*!< Read Only */
typedef __IO int32_t     vs32;
typedef __IO int16_t     vs16;
typedef __IO int8_t      vs8;
typedef __I int32_t      vsc32;         /*!< Read Only */
typedef __I int16_t      vsc16;         /*!< Read Only */
typedef __I int8_t       vsc8;          /*!< Read Only */
typedef uint64_t         u64;
```

```
typedef uint32_t            u32;
typedef uint16_t            u16;
typedef uint8_t             u8;
typedef const uint32_t      uc32;           /*!< Read Only */
typedef const uint16_t      uc16;           /*!< Read Only */
typedef const uint8_t       uc8;            /*!< Read Only */
typedef __IO uint32_t       vu32;
typedef __IO uint16_t       vu16;
typedef __IO uint8_t        vu8;
typedef __I uint32_t        vuc32;          /*!< Read Only */
typedef __I uint16_t        vuc16;          /*!< Read Only */
typedef __I uint8_t         vuc8;           /*!< Read Only */
```

**1.3.2 布尔类型**

在文件 typedef.h 中，布尔形变量被定义如下：

```
typedef enum {false = 0, true = 1, FALSE = 0, TRUE = 1} BOOL, bool;
```

**1.3.3 标志位状态类型**

在文件 typedef.h 中，标志位状态形变量被定义如下：

```
typedef enum {RESET = 0, SET = !RESET} FlagStatus, ITStatus;
```

**1.3.4 功能状态型类型**

在文件 typedef.h 中，功能状态形变量被定义如下：

```
typedef enum {DISABLE = 0, ENABLE = !DISABLE} FunctionalState;
```

**1.3.5 错误状态类型**

在文件 typedef.h 中，错误状态形变量被定义如下：

```
typedef enum {ERROR = 0, SUCCESS = !ERROR} ErrorStatus;
```

**1.3.6 外设**

用户可以通过指向各个外设的指针访问各外设的控制寄存器。这些指针所指向的数据结构与各个外设的控制寄存器布局一一对应。

文件 jsh300x.h 包含了所有外设控制寄存器的结构，下例为 SPI 寄存器结构的声明：

```
typedef struct
{
    __IO uint32_t CON0;
    __IO uint32_t CON1;
    __IO uint32_t CMD_DATA;
    __IO uint32_t BAUD;
```

```
    __IO uint32_t DMA_LEN;
    __IO uint32_t DMA_CNT;
    __IO uint32_t DMA_STADR;
    __IO uint32_t STA;
} SPI_I2C_TypeDef;
```

寄存器命名遵循上节的寄存器缩写命名规则。

# 2. 固件函数库

## 2.1 压缩包描述

Jsh300x 固件函数库被压缩在一个 zip 文件中。解压该文件会产生一个文件夹：jsh300x_sdk_V2.00-20190903。

### 2.1.1 文件夹 Example

文件夹 Examples，对应每一个 jsh300x 外设，都包含一个子文件夹。这些子文件夹包含了整套文件，组成典型的例子，来示范如何使用对应外设。

Readme.txt：每个例子的简单描述和使用说明。

user.c：该文件包含了所使用到的外设代码。

user.h：函数的声明。

jsh300x_irq.c：该源文件包含了所有的中断处理程序（如果未使用中断，则所有的函数体都为空）。

jsh300x_irq.h：该头文件包含了所有的中断处理程序的原形。

main.c：例程代码

Include.h：主头文件，包含了其他头文件

注：所有的例程的使用，都不受不同软件开发环境的影响。

### 2.1.2 文件夹 HAL

jsh300x_hal_tk_bsp.c：该文件包含了 touchkey 使用的函数体。

jsh300x_hal_tk_bsp.h：该文件包含了 touchkey 使用的函数的声明

jsh300x_tk_cfg.h：该文件包含了 touchkey 功能配置的宏。

### 2.1.3 文件夹 Libraries

文件夹 CMSIS：包含了内核特殊指令的指令包装。

文件夹 Device\JSH300x\JSH300x_LL_Driver：包含的是所有外设驱动源程序文件和头文件。

文件夹 Device\JSH300x\startup\arm：包含是的芯片启动代码。

固件库函数文件描述

| 文件名 | 描述 |
| --- | --- |

| | |
|---|---|
| jsh300x.h | 定义了外设寄存器的地址 |
| jsh300x_assert.c | 报告发生参数错误的源文件的名称和源行号 |
| jsh300x_assert.h | 声明函数 |
| jsh300x_debug.c | Debug 相关功能的配置 |
| jsh300x_debug.h | Degug 相关函数的声明 |
| jsh300x_misc.c | NVIC 初始化及系统滴答计时器的配置函数 |
| jsh300x_misc.h | NVIC 及系统滴答计时器相关函数的声明 |
| jsh300x_system.c | 包含了系统及系统时钟的相关函数 |
| jsh300x_system.h | 包含了系统及系统时钟的相关函数的声明 |
| Typedef.h | 通用声明文件。<br>包含所有外设驱动使用的通用类型和常数。 |

### 2.2 外设初始化

本节按步骤描述了如何初始化和设置任意外设。这里 PPP 代表任意外设。

1、在主应用文件中，声明一个结构 TYPE_LL_PPP_INIT，例如：

TYPE_LL_PPP_INIT ppp_struct;

这里 ppp_struct 是一个位于内存中的工作变量，用来初始化一个或者多个外设 PPP。

2、为变量 ppp_struct 的各个结构成员填入允许的值。

ppp_struct.member1 = val1;

ppp_struct.member2 = val2;

.....

ppp_struct.membern = valn;

3. 调用函数 ll_ppp_init(..)来初始化外设 PPP。

4. 在这一步，外设 PPP 已被初始化。可以调用函数 PPP_Cmd(..)来使能之。

ll_ppp_enable(PPP);

可以通过调用一系列函数来使用外设。每个外设都拥有各自的功能函数。

## 3. 模拟数字转换器

该模块是一个12bit的逐次逼近式的ADC控制器。ADC支持多种工作模式：单次转换和连续转换，并且可选择通道自动扫描。ADC启动包括软件启动、外部引脚触发以及其他片内外设启动（timer触发启动）。

### 3.1 函数 ll_adc_once_sample_init

| | |
|---|---|
| 函数名 | Void ll_adc_once_sample_init(ADKEY_TypeDef *p_adc，<br>TYPE_LL_ADC_ONCE_SAMPLE_INIT *p_init) |
| 功能 | 配置 adc 单次采样 |
| 参数 1 | p_adc 只有 ADKEY |
| 参数 2 | p_init：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_adc_once_sample_init {
    FunctionalState                 adc_en;
    TYPE_ENUM_LL_ADC_DATA_ALIGN     data_align;
    TYPE_ENUM_LL_ADC_ONCE_MD_CH_SEL adc_channel;
    u8                              adc_psc;
} TYPE_LL_ADC_ONCE_SAMPLE_INIT;
```

**adc_en** ：adc 使能开启位。

| 定义 | 描述 |
|---|---|
| ENABLE | 使能开 |
| DISABLE | 使能关 |

**data_align**：数据对齐

| 定义 | 描述 |
|---|---|
| LL_ADC_DATA_LEFT_ALIGN | 数据左对齐 |
| LL_ADC_DATA_RIGHT_ALIGN | 数据右对齐 |

**adc_channel**：采样通道

详细参考 TYPE_ENUM_LL_ADC_ONCE_MD_CH_SEL 成员

**adc_psc**：adc 时钟预分频系数

| 定义 | 描述 |
|---|---|
| 0 | 2 分频 |
| n | N+1 分频 |

### 3.2 函数 ll_adc_single_cycle_init

| | |
|---|---|
| 函数名 | Void ll_adc_once_sample_init(ADKEY_TypeDef *p_adc，<br>TYPE_LL_ADC_ONCE_SAMPLE_INIT *p_init) |
| 功能 | 配置 adc 单周期扫描采样 |
| 参数 1 | p_adc 只有 ADKEY |
| 参数 2 | p_init：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_adc_single_cycle_init {
FunctionalState                 adc_en;
    u8                          adc_psc;
u8                              d2dcyc;
    TYPE_ENUM_LL_ADC_DATA_ALIGN     data_align;
    u32                         adc_channel_map;
    TYPE_ENUM_LL_ADC_SCAN_DIR       adc_scan_direction;
    FunctionalState             adc_dma_en;
    u32                         adc_dma_addr;

} TYPE_LL_ADC_SINGLE_CYCLE_INIT;
```

**adc_en** ：adc 使能开启位。

| 定义 | 描述 |
|---|---|
| ENABLE | 使能开 |
| DISABLE | 使能关 |

**data_align**：数据对齐

| 定义 | 描述 |
|---|---|
| LL_ADC_DATA_LEFT_ALIGN | 数据左对齐 |
| LL_ADC_DATA_RIGHT_ALIGN | 数据右对齐 |

**adc_channel_map**：采样通道

详细参考 TYPE_ENUM_LL_ADC_ONCE_MD_CH_SEL 成员

**adc_psc**：adc 时钟预分频系数

| 定义 | 描述 |
|---|---|
| 0 | 2 分频 |
| n | n+1 分频 |

**d2dcyc**:两次采样的时间间隔

| 定义 | 描述 |
|---|---|
| n | n*adcclk |

**adc_scan_direction**: adc 采样顺序

| 定义 | 描述 |
|---|---|
| LL_ADC_SCAN_DIR_L2H | 从低序列通道到高序列通道 |
| LL_ADC_SCAN_DIR_H2L | 从高序列通道到低序列通道 |

**adc_dma_en**：dma 传输使能

| 定义 | 描述 |
|---|---|
| ENABLE | 使能开 |
| DISABLE | 使能关 |

**adc_dma_addr**：dma 传输地址

### 3.3 函数 ll_adc_continuous_cycle_init

| 函数名 | Void ll_adc_once_sample_init(ADKEY_TypeDef *p_adc，<br>TYPE_LL_ADC_CONTINUOUS_CYCLE_INIT *p_init) |
|---|---|
| 功能 | 配置 adc 连续扫描采样 |
| 参数 1 | p_adc 只有 ADKEY |
| 参数 2 | p_init：指向包含了指定外设的配置信息的结构体 |

| 返回值 | 无 |
|---|---|

```
typedef struct __ll_adc_continus_init {
    FunctionalState                 adc_en;
    u8                              adc_psc;
    u8                              d2dcyc;
    TYPE_ENUM_LL_ADC_DATA_ALIGN     data_align;
    u32                             adc_channel_map;
    TYPE_ENUM_LL_ADC_SCAN_DIR       adc_scan_direction;
    FunctionalState                 adc_dma_en;
    u32                             adc_dma_addr;

} TYPE_LL_ADC_CONTINUOUS_CYCLE_INIT;
```

**adc_en** ：adc 使能开启位。

| 定义 | 描述 |
|---|---|
| ENABLE | 使能开 |
| DISABLE | 使能关 |

**data_align**：数据对齐

| 定义 | 描述 |
|---|---|
| LL_ADC_DATA_LEFT_ALIGN | 数据左对齐 |
| LL_ADC_DATA_RIGHT_ALIGN | 数据右对齐 |

**adc_channel_map**：采样通道

详细参考 TYPE_ENUM_LL_ADC_ONCE_MD_CH_SEL 成员

**adc_psc**：adc 时钟预分频系数

| 定义 | 描述 |
|---|---|
| 0 | 2 分频 |
| n | n+1 分频 |

**d2dcyc**:两次采样的时间间隔

| 定义 | 描述 |
| --- | --- |
| n | n*adcclk |

**adc_scan_direction**: adc 采样顺序

| 定义 | 描述 |
| --- | --- |
| LL_ADC_SCAN_DIR_L2H | 从低序列通道到高序列通道 |
| LL_ADC_SCAN_DIR_H2L | 从高序列通道到低序列通道 |

**adc_dma_en**：dma 传输使能

| 定义 | 描述 |
| --- | --- |
| ENABLE | 使能开 |
| DISABLE | 使能关 |

**adc_dma_addr**：dma 传输地址

### 3.4 函数 ll_dac0_set_data

| 函数名 | void ll_dac0_set_data(u8 data) |
| --- | --- |
| 功能 | 设置 da0c 待转换数据 |
| 参数 1 | 待转换数据 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 3.5 函数 ll_dac1_set_data

| 函数名 | void ll_dac1_set_data(u16 data) |
| --- | --- |
| 功能 | 设置 dac1 待转换数据 |
| 参数 1 | 待转换数据 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 3.6 函数 ll_adc_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_adc_interrupt_enable(void) |
| 功能 | 使能 adc 中断（如果 ADIE 置位，A/D 转换结束后产生中断请求） |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 3.7 函数 ll_adc_interrupt_disable

| | |
|---|---|
| 函数名 | voidll_adc_interrupt_disable(void) |
| 功能 | 禁用 adc 中断 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 3.8 函数 ll_adc_ovr_interrupt_enable

| | |
|---|---|
| 函数名 | voidll_adc_ovr_interrupt_enable(void) |
| 功能 | 使能数据覆盖中断 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 3.9 函数 ll_adc_ovr_interrupt_disable

| | |
|---|---|
| 函数名 | voidll_adc_ovr_interrupt_disable(void) |
| 功能 | 禁用数据覆盖中断 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 3.10 函数 ll_adc_enable

| | |
|---|---|
| 函数名 | voidll_adc_enable(void) |
| 功能 | 使能 adc |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 3.11 函数 ll_adc_disable

| | |
|---|---|
| 函数名 | voidll_adc_disable(void) |
| 功能 | 禁用 adc |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 3.12 函数 ll_adc_dma_enable

| | |
|---|---|
| 函数名 | voidll_adc_dma_enablee(void) |
| 功能 | 使能 adc dma 传输 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 3.13 函数 ll_adc_dma_disable

| | |
|---|---|
| 函数名 | voidll_adc_dma_disablee(void) |
| 功能 | 禁用 adc dma 传输 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 3.14 函数 ll_adc_ext_trg_src_enable

| | |
|---|---|
| 函数名 | voidll_adc_ext_trg_src_enable(void) |
| 功能 | 使能外部信号触发 ad 转换 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 3.15 函数 ll_adc_ext_trg_src_disable

| | |
|---|---|
| 函数名 | voidll_adc_ext_trg_src_disable(void) |
| 功能 | 禁用外部信号触发 ad 转换 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 3.16 函数 ll_adc_start_enable

| | |
|---|---|
| 函数名 | voidll_adc_start_enable(void) |
| 功能 | 开始启动 ad 转换 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 3.17 函数 ll_adc_start_disable

| | |
|---|---|
| 函数名 | voidll_adc_start_disable(void) |
| 功能 | 关闭 ad 转换 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 3.18 函数 ll_dac0_enable

| | |
|---|---|
| 函数名 | voidll_dac0_enable(void) |
| 功能 | 使能 dac0 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 3.19 函数 ll_dac0_disable

| | |
|---|---|
| 函数名 | voidll_dac0_disable(void) |
| 功能 | 禁用 dac0 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 3.20 函数 ll_dac1_enable

| | |
|---|---|
| 函数名 | voidll_dac1_enable(void) |
| 功能 | 使能 dac1 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 3.21 函数 ll_dac1_disable

| | |
|---|---|
| 函数名 | voidll_dac1_disable(void) |
| 功能 | 禁用 dac1 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

# 4 比较器

芯片内嵌两个通用比较器 COMP0 和 COMP1，可独立使用，也可与定时器结合使用。比较器是将一个模拟电压信号与一个基准电压相比较的电路。比较器的两路输入（正极输入和负极输入）为模拟信号，输出则为二进制信号 0 或 1，当输入的电压的差值增大或减小且正负符号不变时，其输出保持恒定。

## 4.1 函数 ll_comp_init

| | |
|---|---|
| 函数名 | Void ll_comp_init(TYPE_ENUM_LL_COMP_CH channel,TYPE_LL_COMP_INIT *p_init) |
| 功能 | 禁用 dac1 |
| 参数 1 | Channel：比较器 0 LL_COMP_CH0 或比较器 1 LL_COMP_CH1 |
| 参数 2 | p_init：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_comp_init {
    TYPE_ENUM_LL_COMP0_POSITIVE_SEL     comp0_positive_sel;
    TYPE_ENUM_LL_COMP0_NEGATIVE_SEL     comp0_negative_sel;
    TYPE_ENUM_LL_COMP1_POSITIVE_SEL     comp1_positive_sel;
    FunctionalState                     invert_en;
    FunctionalState                     po_en;
    FunctionalState                     lowp_en;
    TYPE_ENUM_LL_COMP0_HY_SEL           hy_mode;
    u8                                  filt_num;
} TYPE_LL_COMP_INIT;
```

**comp0_positive_sel**：比较器 0 正极输入选择

| 定义 | 描述 |
|---|---|
| LL_COMP0_POSITIVE_SEL_DACOUT | Dac 输出 |
| LL_COMP0_POSITIVE_SEL_PC2 | Pc2 |
| LL_COMP0_POSITIVE_SEL_PC3 | Pc3 |
| LL_COMP0_POSITIVE_SEL_PC4 | Pc4 |

**comp0_negative_sel**：比较器 0 负极输入选择

| 定义 | 描述 |
| --- | --- |
| LL_COMP0_NEGATIVE_SEL_PA3 | Pa3 |
| LL_COMP0_NEGATIVE_SEL_PA2 | Pa2 |
| LL_COMP0_NEGATIVE_SEL_PA1 | Pa1 |
| LL_COMP0_NEGATIVE_SEL_PA0 | Pa0 |

**comp1_positive_sel**：比较器 1 正向输入选择

| 定义 | 描述 |
| --- | --- |
| LL_COMP1_POSITIVE_SEL_PC0 | Pa3 |
| LL_COMP1_POSITIVE_SEL_PC1 | Pa2 |
| LL_COMP1_POSITIVE_SEL_PC2 | Pa1 |
| LL_COMP1_POSITIVE_SEL_PC3 | Pa0 |
| LL_COMP1_POSITIVE_SEL_PC6 | Pc6 |
| LL_COMP1_POSITIVE_SEL_PC7 | Pc7 |
| LL_COMP1_POSITIVE_SEL_PB14 | Pb14 |
| LL_COMP1_POSITIVE_SEL_PB12 | Pb12 |
| LL_COMP1_POSITIVE_SEL_PB10 | Pb10 |
| LL_COMP1_POSITIVE_SEL_PB8 | Pb8 |
| LL_COMP1_POSITIVE_SEL_PB6 | Pb6 |
| LL_COMP1_POSITIVE_SEL_PB4 | Pb4 |
| LL_COMP1_POSITIVE_SEL_PB2 | Pb2 |
| LL_COMP1_POSITIVE_SEL_PB0 | Pb0 |
| LL_COMP1_POSITIVE_SEL_PA11 | Pa11 |
| LL_COMP1_POSITIVE_SEL_PA10 | Pa10 |
| LL_COMP1_POSITIVE_SEL_PA9 | Pa9 |
| LL_COMP1_POSITIVE_SEL_PA8 | Pa8 |
| LL_COMP1_POSITIVE_SEL_PA7 | Pa7 |

| | |
|---|---|
| LL_COMP1_POSITIVE_SEL_PA6 | Pa6 |
| LL_COMP1_POSITIVE_SEL_PA5 | Pa5 |
| LL_COMP1_POSITIVE_SEL_PA4 | Pa4 |
| LL_COMP1_POSITIVE_SEL_PA3 | Pa3 |
| LL_COMP1_POSITIVE_SEL_PA2 | Pa2 |
| LL_COMP1_POSITIVE_SEL_PA1 | Pa1 |
| LL_COMP1_POSITIVE_SEL_PA0 | Pa0 |
| LL_COMP1_POSITIVE_SEL_BG | bg |
| LL_COMP1_POSITIVE_SEL_TEMP | temp |
| LL_COMP1_POSITIVE_SEL_DAC_DIV_4 | Dac voltage /4 |
| LL_COMP1_POSITIVE_SEL_DAC | Dac voltage |
| LL_COMP1_POSITIVE_SEL_OPMA | opma 输出 |
| LL_COMP1_POSITIVE_SEL_OPMB | opmb 输出 |

**invert_en**：输出结果取反

| 定义 | 描述 |
|---|---|
| ENABLE | 使能输出取反 |
| DISABLE | 禁用输出取反 |

**po_en**：

| 定义 | 描述 |
|---|---|
| ENABLE | 使能 |
| DISABLE | 禁用 |

**lowp_en**：开启低功耗

| 定义 | 描述 |
|---|---|
| ENABLE | 使能低功耗 |
| DISABLE | 禁用低功耗 |

**hy_mode**：比较器电压迟滞选择

| 定义 | 描述 |
|---|---|
| LL_COMP0_HY_NONE | 没有迟滞 |
| LL_COMP0_HY_44MV | 迟滞 44mv |
| LL_COMP0_HY_68MV | 迟滞 68mv |
| LL_COMP0_HY_84MV | 迟滞 84mv |

**filt_num**：比较器滤波周期

| 定义 | 描述 |
|---|---|
| n | n*cycle |

## 4.2 函数 ll_comp_wakeup_or_irq_config

| 函数名 | Void ll_comp_wakeup_or_irq_config(TYPE_ENUM_LL_COMP_CH channel, TYPE_LL_COMP_WAKEUP_OR_IRQ_CFG *p_cfg) |
|---|---|
| 功能 | 配置中断或唤醒功能 |
| 参数 1 | Channel：比较器 0 LL_COMP_CH0 或比较器 1 LL_COMP_CH1 |
| 参数 2 | p_cfg：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_comp_wakeup_or_irq_cfg {
    FunctionalState                     wakeup_en;
    FunctionalState                     irq_en;
    TYPE_ENUM_LL_COMP_WAKEUP_OR_IRQ_MODE    wk_irq_mode;
} TYPE_LL_COMP_WAKEUP_OR_IRQ_CFG;
```

**wakeup_en**：唤醒使能

| 定义 | 描述 |
|---|---|
| ENABLE | 使能唤醒功能 |

| | |
|---|---|
| DISABLE | 禁用唤醒功能 |

**irq_en**：中断使能

| 定义 | 描述 |
|---|---|
| ENABLE | 使能中断功能 |
| DISABLE | 禁用中断功能 |

### 4.3 函数 ll_comp0_interrupt_enable

| | |
|---|---|
| 函数名 | Void ll_comp0_interrupt_enable(void) |
| 功能 | 使能比较器 0 中断 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 4.4 函数 ll_comp0_interrupt_disable

| | |
|---|---|
| 函数名 | Void ll_comp0_interrupt_disable(void) |
| 功能 | 禁用比较器 0 中断 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 4.5 函数 ll_comp1_interrupt_enable

| | |
|---|---|
| 函数名 | Void ll_comp1_interrupt_enable(void) |
| 功能 | 使能比较器 1 中断 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 4.6 函数 ll_comp1_interrupt_disable

| | |
|---|---|
| 函数名 | Void ll_comp1_interrupt_disable(void) |
| 功能 | 禁用比较器 1 中断 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 4.7 函数 ll_comp0_enable

| | |
|---|---|
| 函数名 | Voidll_comp0_enable(void) |
| 功能 | 使能比较器 0 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 4.8 函数 ll_comp0_disable

| | |
|---|---|
| 函数名 | Voidll_comp0_disable(void) |
| 功能 | 禁用比较器 0 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 4.9 函数 ll_comp1_enable

| | |
|---|---|
| 函数名 | Void ll_comp1_enable(void) |
| 功能 | 使能比较器 1 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 4.10 函数 ll_comp1_disable

| | |
|---|---|
| 函数名 | Void ll_comp1_disable(void) |
| 功能 | 禁用比较器 1 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 4.11 模块相关宏定义

| 定义 | 描述 |
|---|---|
| #define LL_COMP0_EN_GET() | 获取比较器 0 使能信号 |
| #define LL_COMP1_EN_GET() | 获取比较器 1 使能信号 |
| #define ll_COMP0_PENDING_GET() | 获取比较器 0 输出状态 |
| #define ll_COMP1_PENDING_GET() | 获取比较器 1 输出状态 |
| #define ll_COMP0_PENDING_CLR() | 清除比较器 0 输出状态 |
| #define ll_COMP1_PENDING_CLR() | 清除比较器 1 输出状态 |

# 5 crc 校验

循环冗余校验（CRC）计算单元是根据自定义的生成多项式得到任一 32 位全字的 CRC 计算结果。在其他的应用中，CRC 技术主要应用于核实数据传输或者数据存储的正确性和完整性。CRC 计算单元可以在程序运行时计算出软件的标识，之后与在连接时生成的参考标识比较，然后存放在指定的存储器空间。

## 5.1 函数 ll_crc_init

| | |
|---|---|
| 函数名 | void ll_crc_init(CRC_TypeDef *p_crc, TYPE_LL_CRC_INIT *p_init) |
| 功能 | CRC 初始化配置 |
| 参数 1 | p_crc：crc 寄存器基地址 |
| 参数 2 | p_init：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_crc_init {
    u32                         init_val;
    u32                         poly;
    TYPE_ENUM_LL_CRC_POLY_WIDTH       poly_bits;
    TYPE_ENUM_LL_CRC_POLY_SHIFT_DIR poly_shift_dir;
    u32                         out_invert;
    u32                         dma_addr;
    u32                         dma_len;
} TYPE_LL_CRC_INIT;
```

**init_val**：初始值

| 定义 | 描述 |
|---|---|
| n | 初始值 |

**Poly**：多项式配置

| 定义 | 描述 |
|---|---|
| n | CRC 多项式，默认为 32 位多项式 |

**poly_bits**：多项式长度

| 定义 | 描述 |
| --- | --- |
| LL_CRC_POLY_5BIT | 多项式长度为 5 个位 |
| LL_CRC_POLY_7BIT | 多项式长度为 7 个位 |
| LL_CRC_POLY_8BIT | 多项式长度为 8 个位 |
| LL_CRC_POLY_16BIT | 多项式长度为 16 个位 |
| LL_CRC_POLY_32BIT | 多项式长度为 32 个位 |

**poly_shift_dir**：移位方向选择

| 定义 | 描述 |
| --- | --- |
| LL_CRC_POLY_SHIFT_RIGHT | 右移 |
| LL_CRC_POLY_SHIFT_LEFT | 左移 |

**out_invert**：结果取反

**dma_addr**：dma 数据地址：以 4 字节对齐，只能位于 SRAM

**dma_len**：dma 数据长度

## 5.2 函数 ll_crc_deinit

| 函数名 | void ll_crc_deinit(CRC_TypeDef *p_crc) |
| --- | --- |
| 功能 | CRC 释放 |
| 参数 1 | p_crc：crc 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

## 5.3 函数 ll_crc_start

| 函数名 | void ll_crc_start(CRC_TypeDef *p_crc, u32 addr, u32 len) |
| --- | --- |

| | |
|---|---|
| 功能 | CRC 校验开始 |
| 参数 1 | p_crc：crc 寄存器基地址 |
| 参数 2 | 启动地址（32 位对齐） |
| 参数 3 | 长度 |
| 返回值 | 无 |

### 5.4 函数 ll_crc_wait_done_pending

| | |
|---|---|
| 函数名 | void ll_crc_wait_done_pending(CRC_TypeDef *p_crc) |
| 功能 | 等待校验完成 |
| 参数 1 | p_crc：crc 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 5.5 函数 ll_crc_stop

| | |
|---|---|
| 函数名 | void ll_crc_stop(CRC_TypeDef *p_crc) |
| 功能 | 等待校验完成 |
| 参数 1 | p_crc：crc 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 5.6 函数 ll_crc_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_crc_interrupt_enable(CRC_TypeDef *p_crc) |
| 功能 | 使能 crc 校验中断 |
| 参数 1 | p_crc：crc 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 5.7 函数 ll_crc_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_crc_interrupt_disable(CRC_TypeDef *p_crc) |
| 功能 | 使能 crc 校验中断 |
| 参数 1 | p_crc：crc 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 5.8 模块相关宏定义

| 定义 | 描述 |
|---|---|
| #define LL_CRC_INTERRUPT_GET(p_crc) | 获取中断使能状态 |
| #define LL_CRC_DONE_PENDING_GET(p_crc) | 获取 CRC 状态 |
| #define LL_CRC_DONE_PENDING_CLR(p_crc) | 清除 CRC 状态 |

# 6 eflash

- 高达 32K 字节闪存存储器
- 存储器结构：
  - 主闪存空间：32K 字节
  - 副闪存空间（系统存储器）：2K 字节
- 带预取缓冲器的读接口
- 闪存编程和擦除操作
- 访问和写保护
- 低功耗模式

## 6.1 函数 ll_eflash_init

| | |
|---|---|
| 函数名 | void ll_eflash_init(EFLASH_TypeDef *p_eflash, TYPE_LL_EFLASH_INIT *p_init) |
| 功能 | Eflash 初始化配置 |
| 参数 1 | p_eflashl：eflash 寄存器基地址 |
| 参数 2 | p_init：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_eflash_init {
    u8 reserved;
} TYPE_LL_EFLASH_INIT;
```

## 6.2 函数 ll_eflash_deinit

| | |
|---|---|
| 函数名 | void ll_eflash_deinit(EFLASH_TypeDef *p_eflash) |
| 功能 | 释放 Eflash |
| 参数 1 | p_eflashl：eflash 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

## 6.3 函数 ll_eflash_nvr_lock

| | |
|---|---|
| 函数名 | void ll_eflash_nvr_lock(EFLASH_TypeDef *p_eflash, TYPE_ENUM_LL_EF_LOCK isLock) |

| | |
|---|---|
| 功能 | nvr 解锁 |
| 参数 1 | p_eflashl：eflash 寄存器基地址 |
| 参数 2 | isLock：是否上锁 |
| 返回值 | 无 |

**isLock**：是否上锁

| 定义 | 描述 |
|---|---|
| LL_EF_UNLOCK | 未上锁 |
| LL_EF_LOCK | 已锁 |

## 6.4 函数 ll_eflash_main_lock

| | |
|---|---|
| 函数名 | void ll_eflash_main_lock(EFLASH_TypeDef *p_eflash,<br>TYPE_ENUM_LL_EF_LOCK isLock) |
| 功能 | main 解锁 |
| 参数 1 | p_eflashl：eflash 寄存器基地址 |
| 参数 2 | isLock：是否上锁 |
| 返回值 | 无 |

**isLock**：是否上锁

| 定义 | 描述 |
|---|---|
| LL_EF_UNLOCK | 未上锁 |
| LL_EF_LOCK | 已锁 |

## 6.5 函数 ll_eflash_timing_set

| | |
|---|---|
| 函数名 | void ll_eflash_timing_set(EFLASH_TypeDef *p_eflash,<br>u32 time_reg0,<br>u32 time_reg1) |
| 功能 | 时序设置 |
| 参数 1 | p_eflashl：eflash 寄存器基地址 |

| | |
|---|---|
| 参数 2 | time_reg0：时序 0 寄存器 |
| 参数 3 | time_reg1：时序 1 寄存器 |
| 返回值 | 无 |

## 6.6 函数 ll_eflash_clk_sel

| | |
|---|---|
| 函数名 | void ll_eflash_clk_sel(EFLASH_TypeDef *p_eflash,TYPE_ENUM_LL_EF_PROG_CLK_SEL exosc_sel) |
| 功能 | 烧写时钟源设置 |
| 参数 1 | p_eflashl：eflash 寄存器基地址 |
| 参数 2 | iexosc_sel：时钟源 |
| 返回值 | 无 |

**iexosc_sel**：时钟源

| 定义 | 描述 |
|---|---|
| LL_EF_PROG_CLK_RC2DIV | 高速 RC 时钟 2 分频 |
| LL_EF_PROG_CLK_EXOSC | 晶振 |

## 6.7 函数 ll_eflash_prefetch_set

| | |
|---|---|
| 函数名 | void ll_eflash_prefetch_set(EFLASH_TypeDef *p_eflash, BOOL on) |
| 功能 | 预取设置 |
| 参数 1 | p_eflashl：eflash 寄存器基地址 |
| 参数 2 | BOOL：1：预取设置开启 0：禁用预取设置 |
| 返回值 | 无 |

## 6.8 函数 ll_eflash_cache_set

| | |
|---|---|
| 函数名 | void ll_eflash_cache_set(EFLASH_TypeDef *p_eflash, BOOL on) |
| 功能 | 缓存设置 |
| 参数 1 | p_eflashl：eflash 寄存器基地址 |

| | |
|---|---|
| 参数 2 | BOOL：1：缓存设置开启　0：禁用缓存设置 |
| 返回值 | 无 |

### 6.9 函数 ll_eflash_cache_data_clr

| | |
|---|---|
| 函数名 | void ll_eflash_cache_data_clr(EFLASH_TypeDef *p_eflash) |
| 功能 | 清缓存启动 |
| 参数 1 | p_eflashl：eflash 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 6.10 函数 ll_eflash_prog_one_data

| | |
|---|---|
| 函数名 | void ll_eflash_prog_one_data(EFLASH_TypeDef *p_eflash, u32 addr, u32 data) |
| 功能 | 写一个数据 |
| 参数 1 | p_eflashl：eflash 寄存器基地址 |
| 参数 2 | Addr：目标地址 |
| 参数 3 | Data：要写的数据 |
| 返回值 | 无 |

### 6.11 函数 ll_eflash_prog_one_data_nvr

| | |
|---|---|
| 函数名 | void ll_eflash_prog_one_data_nvr(EFLASH_TypeDef *p_eflash, u32 addr, u32 data) |
| 功能 | 写一个 nvr 数据 |
| 参数 1 | p_eflashl：eflash 寄存器基地址 |
| 参数 2 | Addr：目标地址 |
| 参数 3 | Data：要写的数据 |
| 返回值 | 无 |

### 6.12 函数 ll_eflash_erase_one_sector

| | |
|---|---|
| 函数名 | void ll_eflash_erase_one_sector(EFLASH_TypeDef *p_eflash, u16 sect_addr) |
| 功能 | 擦除一个扇区 |
| 参数 1 | p_eflashl：eflash 寄存器基地址 |
| 参数 2 | sect_addr：扇区地址 |
| 返回值 | 无 |

### 6.13 函数 ll_eflash_erase_one_sector_nvr

| | |
|---|---|
| 函数名 | void ll_eflash_erase_one_sector_nvr(EFLASH_TypeDef *p_eflash, u16 sect_addr) |
| 功能 | 擦除一个 nvr 扇区 |
| 参数 1 | p_eflashl：eflash 寄存器基地址 |
| 参数 2 | sect_addr：扇区地址 |
| 返回值 | 无 |

### 6.14 函数 ll_eflash_erase_chip

| | |
|---|---|
| 函数名 | void ll_eflash_erase_chip(EFLASH_TypeDef *p_eflash) |
| 功能 | 擦除整片 |
| 参数 1 | p_eflashl：eflash 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 6.15 函数 ll_eflash_crc32

| | |
|---|---|
| 函数名 | u32 ll_eflash_crc32(EFLASH_TypeDef *p_eflash, u32 st_addr, u32 len) |
| 功能 | 32 位 crc 校验 |
| 参数 1 | p_eflashl：eflash 寄存器基地址 |
| 参数 2 | Addr：crc dma 起始地址 |
| 参数 3 | Len：数据长度 |

| | |
|---|---|
| 返回值 | Crc 校验结果 |

### 6.16 函数 ll_eflash_auto_program

| | |
|---|---|
| 函数名 | void ll_eflash_auto_program(EFLASH_TypeDef *p_eflash, u32 ef_addr, u32 ram_addr, u32 len) |
| 功能 | 自动编程 |
| 参数 1 | p_eflashl：eflash 寄存器基地址 |
| 参数 2 | ef_addr：main 区编程地址，4 字节对齐。 |
| 参数 3 | ram_addr：sram 起始地址 |
| 返回值 | 无 |

### 6.17 函数 ll_eflash_crc_out_result_get

| | |
|---|---|
| 函数名 | u32 ll_eflash_crc_out_result_get(EFLASH_TypeDef *p_eflash) |
| 功能 | 读取 crc 校验结果 |
| 参数 1 | p_eflashl：eflash 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 校验结果 |

### 6.18 函数 ll_eflash_cfg_sector_main_get

| | |
|---|---|
| 函数名 | u32 ll_eflash_cfg_sector_main_get(EFLASH_TypeDef *p_eflash) |
| 功能 | 读取主用户扇区配置 |
| 参数 1 | p_eflashl：eflash 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 主用户扇区配置 |

# 7 gpio 端口

## 7.1 函数 ll_gpio_init

| | |
|---|---|
| 函数名 | void ll_gpio_init(GPIO_TypeDef* p_gpio, TYPE_LL_GPIO_INIT* gpio_initstruct) |
| 功能 | 读取主用户扇区配置 |
| 参数 1 | p_gpio：gpio 寄存器基地址 |
| 参数 2 | gpio_initstruct：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __type_ll_gpio_init {
    u16 gpio_pin;
    TYPE_ENUM_LL_GPIO_MODE    gpio_mode;
    TYPE_ENUM_LL_GPIO_TYPE    gpio_type;
    TYPE_ENUM_LL_GPIO_SPEED_MODE gpio_speed_mode;
    TYPE_ENUM_LL_GPIO_SPEED_LEVEL gpio_speed_level;
    TYPE_ENUM_LL_GPIO_PUPD    gpio_pupd ;
} TYPE_LL_GPIO_INIT;
```

**gpio_pin**：选择配置的引脚

| 定义 | 描述 |
|---|---|
| LL_GPIO_PIN_0 | 选择引脚 0 |
| LL_GPIO_PIN_1 | 选择引脚 1 |
| LL_GPIO_PIN_2 | 选择引脚 2 |
| ..... | ..... |
| LL_GPIO_PIN_15 | 选择引脚 15 |
| LL_GPIO_PIN_ALL | 选择全部引脚 |

**gpio_mode**：设置 gpio 引脚工作模式

| 定义 | 描述 |
|---|---|

| | |
|---|---|
| LL_GPIO_MODE_IN | 输入模式 |
| LL_GPIO_MODE_OUT | 输出模式 |
| LL_GPIO_MODE_AF | 复用功能模式 |
| LL_GPIO_MODE_AN | 模拟输入输出模式 |

**gpio_type**：引脚输出类型

| 定义 | 描述 |
|---|---|
| LL_GPIO_TYPE_OUT_PP | 推挽输出 |
| LL_GPIO_TYPE_OUT_OD | 开漏输出 |

**gpio_speed_mode**：引脚驱动模式选择

| 定义 | 描述 |
|---|---|
| LL_GPIO_SPEED_MODE_V | 电压驱动 |
| LL_GPIO_SPEED_MODE_I | 电流驱动 |
| 电流驱动需配置：PMUCON0[11] == 1　PMUCON0[13] == 1 | |

**gpio_speed_level**：驱动能力选择

| 定义 | 描述 |
|---|---|
| LL_GPIO_SPEED_LEVEL_0 | 端口 a、b、c、低速 |
| LL_GPIO_SPEED_LEVEL_1 | 端口 a:speed 1 端口 b、c：超高速 |
| LL_GPIO_SPEED_LEVEL_2 | 端口 a:speed 2 端口 b、c：超高速 |
| LL_GPIO_SPEED_LEVEL_3 | 端口 a:speed 3 端口 b、c：超高速 |
| LL_GPIO_SPEED_LEVEL_4 | 端口 a:speed 4 端口 b、c：超高速 |
| LL_GPIO_SPEED_LEVEL_5 | 端口 a:speed 5 端口 b、c：超高速 |
| LL_GPIO_SPEED_LEVEL_6 | 端口 a:speed 6 端口 b、c：超高速 |
| LL_GPIO_SPEED_LEVEL_7 | 端口 a:高速 端口 b、c：超高速 |

**gpio_pupd**：端口上下拉选择

| 定义 | 描述 |
| --- | --- |
| LL_GPIO_PUPD_NOPULL | 无上下拉 |
| LL_GPIO_PUPD_UP | 上拉 |
| LL_GPIO_PUPD_DOWN | 下拉 |

### 7.2 函数 ll_gpio_read_input_data

| 函数名 | u16 ll_gpio_read_input_data(GPIO_TypeDef* p_gpio) |
| --- | --- |
| 功能 | 读指定的输入端口 |
| 参数 1 | p_gpio：GPIOA/B/C 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 输入端口值 |

### 7.3 函数 ll_gpio_read_input_data_bit

| 函数名 | u8 ll_gpio_read_input_data_bit(GPIO_TypeDef* p_gpio, u16 gpio_pin) |
| --- | --- |
| 功能 | 读指定的输入端口位 |
| 参数 1 | p_gpio：GPIOA/B/C 寄存器基地址 |
| 参数 2 | gpio_pin：指定端口位 |
| 返回值 | 输入端口位值 |

**gpio_pin**：指定读取的端口位

| 定义 | 描述 |
| --- | --- |
| LL_GPIO_PIN_0 | 选择引脚 0 |
| LL_GPIO_PIN_1 | 选择引脚 1 |
| LL_GPIO_PIN_2 | 选择引脚 2 |
| ..... | ..... |
| LL_GPIO_PIN_15 | 选择引脚 15 |

### 7.4 函数 ll_gpio_read_output_data

| 函数名 | u16 ll_gpio_read_output_data(GPIO_TypeDef* p_gpio) |
| --- | --- |

| | |
|---|---|
| 功能 | 读指定的输出端口 |
| 参数 1 | p_gpio：GPIOA/B/C 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 输入端口值 |

### 7.5 函数 ll_gpio_read_output_data_bit

| | |
|---|---|
| 函数名 | u8 ll_gpio_read_output_data_bit(GPIO_TypeDef* p_gpio, u16 gpio_pin) |
| 功能 | 读指定的输出端口位 |
| 参数 1 | p_gpio：GPIOA/B/C 寄存器基地址 |
| 参数 2 | gpio_pin：指定端口位 |
| 返回值 | 输入端口位值 |

**gpio_pin**：指定读取的端口位

| 定义 | 描述 |
|---|---|
| LL_GPIO_PIN_0 | 选择引脚 0 |
| LL_GPIO_PIN_1 | 选择引脚 1 |
| LL_GPIO_PIN_2 | 选择引脚 2 |
| ..... | ..... |
| LL_GPIO_PIN_15 | 选择引脚 15 |

### 7.6 函数 ll_gpio_write_bit

| | |
|---|---|
| 函数名 | void ll_gpio_write_bit(GPIO_TypeDef* p_gpio, u16 gpio_pin, TPYE_ENUM_LL_GPIO_BIT_ACTION gpio_set_value) |
| 功能 | 对指定的输出端口位置位复位操作 |
| 参数 1 | p_gpio：GPIOA/B/C 寄存器基地址 |
| 参数 2 | gpio_pin：指定端口位（可选择全部引脚） |
| 参数 3 | gpio_set_value：设置的值 |
| 返回值 | 无 |

**gpio_pin**：指定端口位

| 定义 | 描述 |
| --- | --- |
| LL_GPIO_PIN_0 | 选择引脚 0 |
| LL_GPIO_PIN_1 | 选择引脚 1 |
| LL_GPIO_PIN_2 | 选择引脚 2 |
| ..... | ..... |
| LL_GPIO_PIN_15 | 选择引脚 15 |
| LL_GPIO_PIN_ALL | 选择全部引脚 |

**gpio_set_value**：设置的值

| 定义 | 描述 |
| --- | --- |
| LL_GPIO_RESET | 复位 |
| LL_GPIO_SET | 置位 |

## 7.7 函数 ll_gpio_set_bits

| 函数名 | void ll_gpio_set_bits(GPIO_TypeDef* p_gpio, u16 gpio_pin) |
| --- | --- |
| 功能 | 对指定的输出端口位置位 |
| 参数 1 | p_gpio：GPIOA/B/C 寄存器基地址 |
| 参数 2 | gpio_pin：指定端口位（可选择全部引脚） |
| 参数 3 | 无 |
| 返回值 | 无 |

**gpio_pin**：指定端口位

| 定义 | 描述 |
| --- | --- |
| LL_GPIO_PIN_0 | 选择引脚 0 |
| LL_GPIO_PIN_1 | 选择引脚 1 |
| LL_GPIO_PIN_2 | 选择引脚 2 |
| ..... | ..... |

| | |
|---|---|
| LL_GPIO_PIN_15 | 选择引脚 15 |
| LL_GPIO_PIN_ALL | 选择全部引脚 |

### 7.8 函数 ll_gpio_reset_bits

| | |
|---|---|
| 函数名 | void ll_gpio_reset_bits(GPIO_TypeDef* p_gpio, u16 gpio_pin) |
| 功能 | 对指定的输出端口位复位 |
| 参数 1 | p_gpio：GPIOA/B/C 寄存器基地址 |
| 参数 2 | gpio_pin：指定端口位（可选择全部引脚） |
| 参数 3 | 无 |
| 返回值 | 无 |

**gpio_pin**：指定端口位

| 定义 | 描述 |
|---|---|
| LL_GPIO_PIN_0 | 选择引脚 0 |
| LL_GPIO_PIN_1 | 选择引脚 1 |
| LL_GPIO_PIN_2 | 选择引脚 2 |
| ..... | ..... |
| LL_GPIO_PIN_15 | 选择引脚 15 |
| LL_GPIO_PIN_ALL | 选择全部引脚 |

### 7.9 函数 ll_gpio_write_data

| | |
|---|---|
| 函数名 | void ll_gpio_write_data(GPIO_TypeDef* p_gpio, u32 gpio_set_value) |
| 功能 | 对指定的输出端口置位复位操作 |
| 参数 1 | p_gpio：GPIOA/B/C 寄存器基地址 |
| 参数 2 | gpio_set_value：设置的值（低高 16 位保留，低 16 位每个位代表一个引脚） |
| 返回值 | 无 |

## 7.10 函数 ll_gpio_pin_lock_config

| | |
|---|---|
| 函数名 | void ll_gpio_pin_lock_config(GPIO_TypeDef* p_gpio, u16 gpio_pin) |
| 功能 | 锁定 gpio 引脚寄存器 |
| 参数 1 | p_gpio：GPIOA/B/C 寄存器基地址 |
| 参数 2 | gpio_pin：指定端口位（可选择全部引脚） |
| 返回值 | 无 |

**gpio_pin**：指定端口位（可选择全部引脚）

| 定义 | 描述 |
|---|---|
| LL_GPIO_PIN_0 | 选择引脚 0 |
| LL_GPIO_PIN_1 | 选择引脚 1 |
| LL_GPIO_PIN_2 | 选择引脚 2 |
| ..... | ..... |
| LL_GPIO_PIN_15 | 选择引脚 15 |
| LL_GPIO_PIN_ALL | 选择全部引脚 |

## 7.11 函数 ll_gpio_group_lock_config

| | |
|---|---|
| 函数名 | void ll_gpio_group_lock_config(GPIO_TypeDef* p_gpio) |
| 功能 | 锁定 gpio 组寄存器 |
| 参数 1 | p_gpio：GPIOA/B/C 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

## 7.12 函数 ll_gpio_pin_af_config

| | |
|---|---|
| 函数名 | void ll_gpio_pin_af_config(GPIO_TypeDef* p_gpio, u16 gpio_pin_src, u16 gpio_af) |
| 功能 | Gpio 复用功能寄存器 |
| 参数 1 | p_gpio：GPIOA/B/C 寄存器基地址 |

| | |
|---|---|
| 参数 2 | gpio_pin_src：选择的引脚（注意与之前的 LL_GPIO_PIN_X 区别） |
| 参数 3 | gpio_af：复用的功能（需选择 GPIO_MODE = GPIO_MODE_AF） |
| 返回值 | 无 |

**gpio_pin_src**：选择的引脚

| 定义 | 描述 |
|---|---|
| LL_GPIO_PIN_SOURCE_0 | 选择引脚 0 |
| LL_GPIO_PIN_SOURCE_1 | 选择引脚 1 |
| LL_GPIO_PIN_SOURCE_2 | 选择引脚 2 |
| ..... | ..... |
| LL_GPIO_PIN_SOURCE_14 | 选择引脚 14 |
| LL_GPIO_PIN_SOURCE_15 | 选择引脚 15 |

**gpio_af**：复用的功能

| 定义 | 描述 |
|---|---|
| LL_GPIO_AF_0 | 复用功能 0 |
| LL_GPIO_AF_1 | 复用功能 1 |
| LL_GPIO_AF_2 | 复用功能 2 |
| LL_GPIO_AF_3 | 复用功能 3 |

## 7.13 函数 ll_gpio_irq_config

| | |
|---|---|
| 函数名 | void ll_gpio_irq_config(GPIO_TypeDef* p_gpio, u16 gpio_pin, FunctionalState newstate) |
| 功能 | Gpio 复用功能寄存器 |
| 参数 1 | p_gpio：GPIOA/B/C 寄存器基地址 |
| 参数 2 | gpio_pin：选择的引脚(可选择全部引脚) |
| 参数 3 | newstate：使能位 |
| 返回值 | 无 |

**gpio_pin**：选择的引脚

| 定义 | 描述 |
| --- | --- |
| LL_GPIO_PIN_0 | 选择引脚 0 |
| LL_GPIO_PIN_1 | 选择引脚 1 |
| LL_GPIO_PIN_2 | 选择引脚 2 |
| ..... | ..... |
| LL_GPIO_PIN_15 | 选择引脚 15 |
| LL_GPIO_PIN_ALL | 选择全部引脚 |

**newstate**：使能位

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

## 7.14 函数 ll_gpio_port_toggle

| 函数名 | void ll_gpio_port_toggle(GPIO_TypeDef* p_gpio, u16 gpio_pin) |
| --- | --- |
| 功能 | Gpio 端口位切换寄存器（即引脚取反） |
| 参数 1 | p_gpio：GPIOA/B/C 寄存器基地址 |
| 参数 2 | gpio_pin：选择的引脚(可选择全部引脚) |
| 参数 3 | 无 |
| 返回值 | 无 |

**gpio_pin**：选择的引脚

| 定义 | 描述 |
| --- | --- |
| LL_GPIO_PIN_0 | 选择引脚 0 |
| LL_GPIO_PIN_1 | 选择引脚 1 |
| LL_GPIO_PIN_2 | 选择引脚 2 |

| | |
|---|---|
| ..... | ..... |
| LL_GPIO_PIN_15 | 选择引脚 15 |
| LL_GPIO_PIN_ALL | 选择全部引脚 |

# 8 led

以动态扫描的方式驱动数码管、LED，支持自动定时扫描或者与 TK 复用的扫描方式，区别是前者按照定时器定时启动扫描，后者是等待 TK 模块给出启动信号再启动扫描。通过控制 COM 和 SEG 口的高低电平来控制是否点亮。

## 8.1 函数 ll_led_init

| | |
|---|---|
| 函数名 | void ll_led_init(LED_TypeDef *p_led, TYPE_LL_LED_INIT *p_init) |
| 功能 | LED 初始化 |
| 参数 1 | p_led：led 寄存器基地址 |
| 参数 2 | p_init：指向包含了指定外设的配置信息的结构体 |
| 参数 3 | 无 |
| 返回值 | 无 |

```
typedef struct __ll_led_init {

    u32                         led_drv;
    u32                         scan_polling_time;
    TYPE_ENUM_LL_LED_SCAN_WAY       scan_way;
    TYPE_ENUM_LL_LED_PLY        led_polarity;
    u32                         sweep_hold_time;
    u32                         led_com_map;
    u32                         led_seg_map;
    u32                         led_addr;
    FunctionalState             com_divt_en;
    FunctionalState             led_drv_en;
    FunctionalState             auto_scan_en;
    FunctionalState             led_en;
    u8                          support_num;
} TYPE_LL_LED_INIT;
```

**led_drv**：LED SEG 端口驱动能力选择(led_drv_en == 1 生效)

| 定义 | 描述 |
|---|---|

| | |
|---|---|
| 0 | 3ma |
| 1 | 6ma |
| ..... | ..... |
| 7 | 24ma |

**scan_polling_time**：扫描周期

| 定义 | 描述 |
|---|---|
| n | n * pclk 扫描一次 |

**scan_way**：扫描方式

| 定义 | 描述 |
|---|---|
| LL_LED_SCAN_WAY_COM | com 扫描 |
| LL_LED_SCAN_WAY_SEG | seg 扫描 |

**led_polarity**：led 极性

| 定义 | 描述 |
|---|---|
| LL_LED_PLY_COMM_CATHODE | 共阴 |
| LL_LED_PLY_COMM_ANODE | 共阳 |

**sweep_hold_time**：扫描停留时间

| 定义 | 描述 |
|---|---|
| n | n * pclk |

**led_com_map**:led com 映射

| 定义 | 描述 |
|---|---|
| n | Bit0 对应于 com0 |

**led_seg_map**：led seg 映射

| 定义 | 描述 |
| --- | --- |
| n | Bit0 对应于 seg0 |

**led_addr：** LED 显示数据缓存区基址

**com_divt_en：**分时扫描使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

**led_drv_en：**硬件驱动能力控制使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

**auto_scan_en：**自动扫描使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

**led_en：**led 使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

**support_num：**led 支持的数量

## 8.2 函数 ll_led_seg_drv_enable

| 函数名 | void ll_led_seg_drv_enable(LED_TypeDef *p_led) |
| --- | --- |

| | |
|---|---|
| 功能 | Seg 硬件驱动使能 |
| 参数 1 | p_led：led 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 8.3 函数 ll_led_seg_drv_disable

| | |
|---|---|
| 函数名 | void ll_led_seg_drv_disable(LED_TypeDef *p_led) |
| 功能 | 禁用 Seg 硬件驱动 |
| 参数 1 | p_led：led 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 8.4 函数 ll_led_auto_scan_enable

| | |
|---|---|
| 函数名 | void ll_led_auto_scan_enable(LED_TypeDef *p_led) |
| 功能 | 自动扫描使能 |
| 参数 1 | p_led：led 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 8.5 函数 ll_led_auto_scan_disable

| | |
|---|---|
| 函数名 | void ll_led_auto_scan_disable(LED_TypeDef *p_led) |
| 功能 | 禁用自动扫描 |
| 参数 1 | p_led：led 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 8.6 函数 ll_led_enable

| | |
|---|---|
| 函数名 | void ll_led_enable(LED_TypeDef *p_led) |
| 功能 | led 使能 |
| 参数 1 | p_led：led 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 8.7 函数 ll_led_disable

| | |
|---|---|
| 函数名 | void ll_led_disable(LED_TypeDef *p_led) |
| 功能 | 禁用 led |
| 参数 1 | p_led：led 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 8.8 函数 ll_led_com_divt_enable

| | |
|---|---|
| 函数名 | void ll_led_com_divt_enable(LED_TypeDef *p_led) |
| 功能 | led 分时扫描使能 |
| 参数 1 | p_led：led 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 8.9 函数 ll_led_com_divt_disable

| | |
|---|---|
| 函数名 | void ll_led_com_divt_disable(LED_TypeDef *p_led) |
| 功能 | 禁用 led 分时扫描 |

| | |
|---|---|
| 参数 1 | p_led：led 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

## 8.10 模块相关宏定义

| 定义 | 描述 |
|---|---|
| #define LL_LED_SEG_DRV_EN_GET(p_led) | LED SEG 硬件驱动使能获取 |
| #define LL_LED_AUTO_SCAN_EN_GET(p_led) | LED 自动扫描使能获取 |
| #define LL_LED_EN_GET(p_led) | LED 使能获取 |
| #define LL_LED_COM_DIVT_EN_GET(p_led) | LED COM 分时扫描使能获取 |

# 9 低电压检测 lvd

JSH3000 内部集成两个电压检测器，一个检测外部供电 VCC,一个检测内部 LDO 输出 VDD，LDO 采用 capless 结构，封装上 VDD 不可见。两种检测电压均阈值可选。当系统监测到 VCC 或 VDD 电压低于配置电压值时，可以选择触发系统复位或通过使能 PVD 中断进入中断子函数。这一特性可用于用于执行紧急关闭任务。检测信号可以选择经过毛刺滤波电路或直接检测，由 LVDCON. lvdvcc_bps_en 和 LVDCON. lvdvdd_bps_en 来控制。

## 9.1 函数 ll_lvd_init

| | |
|---|---|
| 函数名 | void ll_lvd_init(LVD_TypeDef *p_lvd, TYPE_LL_LVD_INIT *p_init) |
| 功能 | 初始化低电压检测 lvd |
| 参数 1 | p_lvd：lvd 寄存器基地址 |
| 参数 2 | p_init：指向包含了指定外设的配置信息的结构体 |
| 参数 3 | 无 |
| 返回值 | 无 |

```
typedef struct __ll_lvd_init {
    u32 lvd_con_bit_map;
} TYPE_LL_LVD_INIT;
```

**lvd_con_bit_map**：寄存器位映象

## 9.2 函数 ll_lvd_con_set

| | |
|---|---|
| 函数名 | void ll_lvd_con_set(LVD_TypeDef *p_lvd u32 con_bit_map) |
| 功能 | 寄存器配置 |
| 参数 1 | p_lvd：lvd 寄存器基地址 |
| 参数 2 | con_bit_map：寄存器位映射 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 9.3 函数 ll_lvd_con_get

| | |
|---|---|
| 函数名 | u32 ll_lvd_con_get(LVD_TypeDef *p_lvd) |
| 功能 | 获取寄存器配置 |
| 参数 1 | p_lvd：lvd 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 寄存器配置值 |

### 9.4 函数 ll_lvd_vdd_enable

| | |
|---|---|
| 函数名 | void ll_lvd_vdd_enable(LVD_TypeDef *p_lvd) |
| 功能 | 使能 vdd 低电压检测 |
| 参数 1 | p_lvd：lvd 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 9.5 函数 ll_lvd_vdd_disable

| | |
|---|---|
| 函数名 | void ll_lvd_vdd_disable(LVD_TypeDef *p_lvd) |
| 功能 | 禁用 vdd 低电压检测 |
| 参数 1 | p_lvd：lvd 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 9.6 函数 ll_lvd_vcc_enable

| | |
|---|---|
| 函数名 | void ll_lvd_vcc_enable(LVD_TypeDef *p_lvd) |
| 功能 | 使能 vcc 低电压检测 |

| | |
|---|---|
| 参数 1 | p_lvd：lvd 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 9.7 函数 ll_lvd_vcc_disable

| | |
|---|---|
| 函数名 | void ll_lvd_vcc_disable(LVD_TypeDef *p_lvd) |
| 功能 | 禁用 vcc 低电压检测 |
| 参数 1 | p_lvd：lvd 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 9.8 函数 ll_lvd_vdd_debounce_enable

| | |
|---|---|
| 函数名 | void ll_lvd_vdd_debounce_enable(LVD_TypeDef *p_lvd) |
| 功能 | vdd 低压检测器去抖动使能 |
| 参数 1 | p_lvd：lvd 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 9.9 函数 ll_lvd_vdd_debounce_disable

| | |
|---|---|
| 函数名 | void ll_lvd_vdd_debounce_disable(LVD_TypeDef *p_lvd) |
| 功能 | 禁用 vdd 低压检测器去抖动 |
| 参数 1 | p_lvd：lvd 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 9.10 函数 ll_lvd_vcc_debounce_enable

| | |
|---|---|
| 函数名 | void ll_lvd_vcc_debounce_enable(LVD_TypeDef *p_lvd) |
| 功能 | vcc 低压检测器去抖动使能 |
| 参数 1 | p_lvd：lvd 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 9.11 函数 ll_lvd_vcc_debounce_disable

| | |
|---|---|
| 函数名 | void ll_lvd_vcc_debounce_disable(LVD_TypeDef *p_lvd) |
| 功能 | 禁用 vcc 低压检测器去抖动 |
| 参数 1 | p_lvd：lvd 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 9.12 函数 ll_lvd_out_enable

| | |
|---|---|
| 函数名 | void ll_lvd_out_enable(LVD_TypeDef *p_lvd) |
| 功能 | 使能阈值判断触发条件后中断以及复位功能 |
| 参数 1 | p_lvd：lvd 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 9.13 函数 ll_lvd_out_disable

| | |
|---|---|
| 函数名 | void ll_lvd_out_enable(LVD_TypeDef *p_lvd) |
| 功能 | 禁用阈值判断触发条件后中断以及复位功能 |
| 参数 1 | p_lvd：lvd 寄存器基地址 |

| | |
|---|---|
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 9.14 函数 ll_lvd_vdd_reset_enable

| | |
|---|---|
| 函数名 | void ll_lvd_vdd_reset_enable(LVD_TypeDef *p_lvd) |
| 功能 | 触发阈值时，vdd 低压检测器复位使能 |
| 参数 1 | p_lvd：lvd 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 9.15 函数 ll_lvd_vdd_reset_disable

| | |
|---|---|
| 函数名 | void ll_lvd_vdd_reset_disable(LVD_TypeDef *p_lvd) |
| 功能 | 触发阈值时，禁用 vdd 低压检测器复位 |
| 参数 1 | p_lvd：lvd 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 9.16 函数 ll_lvd_vcc_reset_enable

| | |
|---|---|
| 函数名 | void ll_lvd_vcc_reset_enable(LVD_TypeDef *p_lvd) |
| 功能 | 触发阈值时，vcc 低压检测器复位使能 |
| 参数 1 | p_lvd：lvd 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 9.17 函数 ll_lvd_vcc_reset_disable

| | |
|---|---|
| 函数名 | void ll_lvd_vcc_reset_disable(LVD_TypeDef *p_lvd) |
| 功能 | 触发阈值时，禁用 vcc 低压检测器复位 |
| 参数 1 | p_lvd：lvd 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 9.18 函数 ll_lvd_vdd_lvd_get

| | |
|---|---|
| 函数名 | TYPE_ENUM_LL_LVD_VDD_LVD_SET ll_lvd_vdd_lvd_get(LVD_TypeDef *p_lvd) |
| 功能 | 获取 vdd 低压检测器的设置值 |
| 参数 1 | p_lvd：lvd 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | vdd 低压检测器的设置值 |

### 9.19 函数 ll_lvd_vcc_lvd_get

| | |
|---|---|
| 函数名 | TYPE_ENUM_LL_LVD_VDD_LVD_SET ll_lvd_vdd_lvd_get(LVD_TypeDef *p_lvd) |
| 功能 | 获取 vcc 低压检测器的设置值 |
| 参数 1 | p_lvd：lvd 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | vcc 低压检测器的设置值 |

### 9.20 模块相关宏定义

| 定义 | 描述 |
|---|---|

| | |
|---|---|
| #define LL_LVD_VCC_RESET_EN_GET(p_lvd) | 获取阈值触发时，VCC 低压检测器复位使能 |
| #define LL_LVD_VDD_PENDING_GET(p_lvd) | 获取 vdd 低电压检测器中断标志 |
| #define LL_LVD_VDD_PENDING_CLR(p_lvd) | 清除 vdd 低压检测器中断标志 |
| #define LL_LVD_VCC_PENDING_GET(p_lvd) | 获取 vcc 低电压检测器中断标志 |
| #define LL_LVD_VCC_PENDING_CLR(p_lvd) | 清除 vcc 低压检测器中断标志 |

# 10 运算放大器 opam

运算放大器是具有很高放大倍数的电路单元。在实际电路中，通常结合反馈网络共同组成某种功能模块。它是一种带有特殊耦合电路及反馈的放大器。芯片内嵌一个运算放大器，运算放大器的输入和输出都连接到 I/O，通过共享 I/O 可以与 ADC、比较器相连。

## 10.1 函数 ll_opam_init

| | |
|---|---|
| 函数名 | void ll_opam_init(OPAM_TypeDef *p_opam, TYPE_LL_OPAM_INIT *p_init) |
| 功能 | 初始化运算放大器 |
| 参数 1 | p_opam：opam 寄存器基地址 |
| 参数 2 | p_init：指向包含了指定外设的配置信息的结构体 |
| 参数 3 | 无 |
| 返回值 | 无 |

```
typedef struct __ll_opam_init {
    TYPE_ENUM_LL_OPAM_AINX_SEL      ainx_sel;
    TYPE_ENUM_LL_OPAM_BINX_SEL      binx_sel;
    FunctionalState                 trim_en;
    FunctionalState                 low_power_en;
    FunctionalState                 opam_en;
    u8                              trim_value;
} TYPE_LL_OPAM_INIT;
```

**ainx_sel**：运算放大器正级输入选择

| 定义 | 描述 |
|---|---|
| LL_OPAM_AINX_SEL_PA2 | 选择 PA2 作为正极输入 |
| LL_OPAM_AINX_SEL_PA1 | 选择 PA1 作为正极输入 |
| LL_OPAM_AINX_SEL_PC4 | 选择 PC4 作为正极输入 |

**binx_sel**：运算放大器负级输入选择

| 定义 | 描述 |
|---|---|

| | |
|---|---|
| LL_OPAM_BINX_SEL_PA0 | 选择 PA0 作为负极输入 |
| LL_OPAM_BINX_SEL_PC0 | 选择 PC0 作为负极输入 |
| LL_OPAM_BINX_SEL_PC1 | 选择 PC1 作为负极输入 |
| LL_OPAM_BINX_SEL_PC5 | 选择 PC5 作为负极输入 |

**trim_en**：运算放大器适配校准启用或禁用

| 定义 | 描述 |
|---|---|
| LL_OPAM_AINX_SEL_PA2 | 选择 PA2 作为正极输入 |
| LL_OPAM_AINX_SEL_PA1 | 选择 PA1 作为正极输入 |
| LL_OPAM_AINX_SEL_PC4 | 选择 PC4 作为正极输入 |

**low_power_en**：运算放大器低功率启用或禁用

| 定义 | 描述 |
|---|---|
| DISABLE | 不使能 |
| ENABLE | 使能 |

**opam_en**：运算放大器启用或禁用

| 定义 | 描述 |
|---|---|
| DISABLE | 不使能 |
| ENABLE | 使能 |

**trim_value**：运算放大器适配校准值

适配校准值获取方法：详见 jsh3000 用户手册。

## 10.2 函数 ll_opam_enable

| | |
|---|---|
| 函数名 | void ll_opam_enable(OPAM_TypeDef *p_opam) |
| 功能 | 使能运算放大器 |
| 参数 1 | p_opam：opam 寄存器基地址 |

| | |
|---|---|
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 10.3 函数 ll_opam_disable

| | |
|---|---|
| 函数名 | void ll_opam_disable(OPAM_TypeDef *p_opam) |
| 功能 | 禁用运算放大器 |
| 参数 1 | p_opam：opam 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 10.4 函数 ll_opam_trim_enable

| | |
|---|---|
| 函数名 | void ll_opam_trim_enable(OPAM_TypeDef *p_opam) |
| 功能 | 运算放大器适配校准使能 |
| 参数 1 | p_opam：opam 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 10.5 函数 ll_opam_trim_disable

| | |
|---|---|
| 函数名 | void ll_opam_trim_disable(OPAM_TypeDef *p_opam) |
| 功能 | 禁用运算放大器适配校准 |
| 参数 1 | p_opam：opam 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 10.6 函数 ll_opam_lowp_enable

| | |
|---|---|
| 函数名 | void ll_opam_lowp_enable(OPAM_TypeDef *p_opam) |
| 功能 | 运算放大器低功耗使能 |
| 参数 1 | p_opam：opam 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 10.7 函数 ll_opam_lowp_disable

| | |
|---|---|
| 函数名 | void ll_opam_lowp_enable(OPAM_TypeDef *p_opam) |
| 功能 | 禁用运算放大器低功耗 |
| 参数 1 | p_opam：opam 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 10.8 函数 ll_opam_lock_enable

| | |
|---|---|
| 函数名 | void ll_opam_lock_enable(OPAM_TypeDef *p_opam) |
| 功能 | 运算放大器寄存器锁定使能(仅设置一次) |
| 参数 1 | p_opam：opam 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 10.9 模块相关宏定义

| 定义 | 描述 |
|---|---|
| #define LL_OPAM_EN_GET(p_opam) | 获取运算放大器启用状态 |

| | |
|---|---|
| #define<br>LL_OPAM_TRIM_EN_GET(p_opam) | 获取运算放大器适配校准使能 |
| #define<br>LL_OPAM_LOWP_EN_GET(p_opam) | 获取运算放大器适配校准使能 |
| #define<br>LL_OPAM_LOCK_EN_GET(p_opam) | 获取运算放大器寄存器锁定状态 |

# 11 通信接口 spi、iic

本芯片支持 SPI 和 IIC 分时复用。SPI 接口广泛用于不同设备之间的板级通讯，如扩展串行 Flash，ADC 等。许多 IC 制造商生产的器件都支持 SPI 接口。IIC（Inter－Integrated Circuit）总线是由 PHILIPS 公司开发的两线式串行总线，用于连接微控制器及其外围设备。它提供多主模式功能，可以控制所有 IIC 总线特定的序列、协议、仲裁和时序。是微电子通信控制领域广泛采用的一种总线标准。

## 11.1 函数 ll_spi_init

| | |
|---|---|
| 函数名 | void ll_spi_init(SPI_I2C_TypeDef *p_spi, TYPE_LL_SPI_INIT *p_init) |
| 功能 | 初始化 spi |
| 参数 1 | p_spi：spi 寄存器基地址 |
| 参数 2 | p_init：指向包含了指定外设的配置信息的结构体 |
| 参数 3 | 无 |
| 返回值 | 无 |

```
typedef struct __ll_spi_init {
    TYPE_ENUM_LL_SPI_FRAME_SIZE     frame_size;
    TYPE_ENUM_LL_SPI_WIRE_MODE      wire_mode;
    TYPE_ENUM_LL_SPI_MODE           spi_mode;
    TYPE_ENUM_LL_SSP_WORK_MODE      work_mode;
    u16                             delay_cycle_cnt;
    u16                             baud;
    FunctionalState                 cs_en;
    FunctionalState                 cs_rising_ie;
    FunctionalState                 frame_ie;
    FunctionalState                 fifo_ov_ie;
    FunctionalState                 rfifo_not_empty_ie;
    FunctionalState                 tfifo_not_full_ie;
} TYPE_LL_SPI_INIT;
```

**frame_size**：每次发送的数据位数

| 定义 | 描述 |
|---|---|
| LL_SPI_8_BIT | 1 个字节 |

| | |
|---|---|
| LL_SPI_16_BIT | 2 个字节 |
| LL_SPI_24_BIT | 3 个字节 |
| LL_SPI_32_BIT | 4 个字节 |

**wire_mode**：SPI 模块的连接模式

| 定义 | 描述 |
|---|---|
| LL_SPI_NORMAL_MODE | spi 通常数据模式 |
| LL_SPI_THREE_WIRE_MODE | spi 3 数据线模式 |
| LL_SPI_DUAL_MODE | spi 2 数据线模式 |
| LL_SPI_QUAD_MODE | spi 4 数据线模式 |

**spi_mode**：数据采样模式

| 定义 | 描述 |
|---|---|
| LL_SPI_MODE_0 | 时钟 idel 为 0，上升沿采样，下降沿出数据 |
| LL_SPI_MODE_1 | 时钟 idel 为 0，下降沿采样，上升沿出数据 |
| LL_SPI_MODE_2 | 时钟 idle 为 1，下降沿采样，上升沿出数据 |
| LL_SPI_MODE_3 | 时钟 idel 为 1，上升沿采样，下降沿出数据 |

**work_mode**：工作模式

| 定义 | 描述 |
|---|---|
| LL_SSP_MASTER_MODE | 主机模式 |
| LL_SSP_SLAVE_MODE | 从机模式 |

**delay_cycle_cnt**：SPI 主模式，捕获串行数据延时周期

| 定义 | 描述 |
|---|---|
| 0 | 没有延时 |
| 1 | 延时 1 个周期 |
| 2 | 延时 2 个周期 |

| | |
|---|---|
| 3 | 延时 3 个周期 |
| 4 | 延时 4 个周期 |
| 5 | 延时 5 个周期 |
| 6 | 延时 6 个周期 |
| 7 | 延时 7 个周期 |

**Baud**：波特率

| 定义 | 描述 |
|---|---|
| 波特率 = apb0_clk /(2 * (Baud + 1)) | |

**cs_en**： cs 控制

| 定义 | 描述 |
|---|---|
| DISABLE | 不使能 |
| ENABLE | 使能 |

**cs_rising_ie**：spi cs 上升沿检测中断使能

| 定义 | 描述 |
|---|---|
| DISABLE | 不使能 |
| ENABLE | 使能 |

**frame_ie**：发送接受完成一帧数据中断

| 定义 | 描述 |
|---|---|
| DISABLE | 不使能 |
| ENABLE | 使能 |

**fifo_ov_ie**：fifo 溢出中断

| 定义 | 描述 |
|---|---|
| DISABLE | 不使能 |

| | |
|---|---|
| ENABLE | 使能 |

**rfifo_not_empty_ie**：接收不为空中断使能

| 定义 | 描述 |
|---|---|
| DISABLE | 不使能 |
| ENABLE | 使能 |

**tfifo_not_full_ie**：发送不为空中断使能

| 定义 | 描述 |
|---|---|
| DISABLE | 不使能 |
| ENABLE | 使能 |

## 11.2 函数 ll_spi_dma_config

| | |
|---|---|
| 函数名 | void ll_spi_dma_config(SPI_I2C_TypeDef *p_spi, TYPE_LL_SPI_DMA_CFG *p_cfg) |
| 功能 | 配置 spi dma 功能 |
| 参数 1 | p_spi：spi 寄存器基地址 (SPI) |
| 参数 2 | p_cfg：指向包含了指定外设的配置信息的结构体 |
| 参数 3 | 无 |
| 返回值 | 无 |

```
typedef struct __ll_spi_dma_cfg {
    u32                         dma_addr;
    u16                         dma_size;
    TYPE_ENUM_LL_SSP_DIRECTION  dir;
    FunctionalState             dma_ie;
} TYPE_LL_SPI_DMA_CFG;
```

**dma_addr**：dma 数据地址

**dma_size**：dma 数据大小

**dir**：数据传输方向

| 定义 | 描述 |
| --- | --- |
| LL_SSP_TX | 发送 |
| LL_SSP_RX | 接收 |

**dma_ie**:完成中断使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

### 11.3 函数 ll_spi_deinit

| 函数名 | void ll_spi_deinit(SPI_I2C_TypeDef *p_spi) |
| --- | --- |
| 功能 | 释放 spi |
| 参数 1 | p_spi：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.4 函数 ll_iic_init

| 函数名 | void ll_iic_init(SPI_I2C_TypeDef *p_iic, TYPE_LL_IIC_INIT *p_init) |
| --- | --- |
| 功能 | iic 初始化 |
| 参数 1 | p_iic：iic 寄存器基地址 |
| 参数 2 | p_init：指向包含了指定外设的配置信息的结构体 |
| 参数 3 | 无 |
| 返回值 | 无 |

```
typedef struct __ll_iic_init {
    u16                         baud;
```

```
    TYPE_ENUM_LL_SSP_WORK_MODE      work_mode;
    FunctionalState                 al_ie;
    FunctionalState                 stop_ie;
    FunctionalState                 frame_ie;
    FunctionalState                 fifo_ov_ie;
    FunctionalState                 rfifo_not_empty_ie;
    FunctionalState                 tfifo_not_full_ie;
} TYPE_LL_IIC_INIT;
```

**baud**：波特率

| 定义 | 描述 |
| --- | --- |
| 波特率 = apb0_clk /(4* (Baud + 1)) | |

**work_mode**：工作模式

| 定义 | 描述 |
| --- | --- |
| LL_SSP_MASTER_MODE | 主机模式 |
| LL_SSP_SLAVE_MODE | 从机模式 |

**al_ie**:仲裁丢失中断使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

**stop_ie**:接收停止中断使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

**frame_ie**:一帧数据中断

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

**fifo_ov_ie**:fifo 溢出中断使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

**rfifo_not_empty_ie**:接收不为空中断使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

t**fifo_not_full_ie**；发送不满中断使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

## 11.5 函数 ll_iic_dma_irq_config

| 函数名 | void ll_iic_dma_irq_config(SPI_I2C_TypeDef *p_iic, TYPE_IIC_DMA_CFG *p_cfg) |
| --- | --- |
| 功能 | iic dma 配置初始化 |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | p_cfg：指向包含了指定外设的配置信息的结构体 |
| 参数 3 | 无 |
| 返回值 | 无 |

```
typedef struct {
    u32                 dma_addr;
    u16                 dma_size;
    TYPE_ENUM_LL_SSP_DIRECTION dir;
    FunctionalState     dma_ie;
} TYPE_IIC_DMA_CFG;
```

**dma_addr**：dma 数据地址

**dma_size**：dma 数据大小

**dir**：数据传输方向

| 定义 | 描述 |
|---|---|
| LL_SSP_TX | 发送 |
| LL_SSP_RX | 接收 |

**dma_ie**:完成中断使能

| 定义 | 描述 |
|---|---|
| DISABLE | 不使能 |
| ENABLE | 使能 |

## 11.6 函数 ll_iic_slave_mode_config

| 函数名 | void ll_iic_slave_mode_config(SPI_I2C_TypeDef *p_iic,<br>TYPE_IIC_SLAVE_CFG *p_cfg) |
|---|---|
| 功能 | iic 从机模式配置初始化 |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | p_cfg：指向包含了指定外设的配置信息的结构体 |
| 参数 3 | 无 |
| 返回值 | 无 |

```
typedef struct {
    u16                             slave_addr;
    TYPE_ENUM_LL_IIC_ADDR_WIDTH     addr_width;
    FunctionalState                 nack_ie;
    FunctionalState                 broadcast_en;
    FunctionalState                 broadcast_ie;
    FunctionalState                 addr_match_ie;
} TYPE_IIC_SLAVE_CFG;
```

**slave_addr**：从模式地址

**addr_width**：地址宽度

| 定义 | 描述 |
| --- | --- |
| LL_IIC_ADDR_7BIT | 7bit |
| LL_IIC_ADDR_10BIT | 10bit |

**nack_ie**：接收应答中断使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

**broadcast_en**：接收广播使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

**broadcast_ie**：接收广播中断使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

**addr_match_ie**：地址匹配中断使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

### 11.7 函数 ll_iic_deinit

| 函数名 | void ll_iic_deinit(SPI_I2C_TypeDef *p_iic) |
| --- | --- |
| 功能 | 释放 iic |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.8 函数 ll_spi_iic_dma_interrupt_enable

| 函数名 | void ll_spi_iic_dma_interrupt_enable(SPI_I2C_TypeDef *p_ssp) |
| --- | --- |
| 功能 | dma 传输完成中断使能 |
| 参数 1 | p_ssp：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.9 函数 ll_spi_iic_dma_interrupt_disable

| 函数名 | void ll_spi_iic_dma_interrupt_disable(SPI_I2C_TypeDef *p_ssp) |
| --- | --- |
| 功能 | 禁用 dma 传输完成中断 |
| 参数 1 | p_ssp：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.10 函数 ll_spi_iic_fifo_ov_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_spi_iic_fifo_ov_interrupt_enable(SPI_I2C_TypeDef *p_ssp) |
| 功能 | 使能数据溢出中断 |
| 参数 1 | p_ssp：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.11 函数 ll_spi_iic_fifo_ov_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_spi_iic_fifo_ov_interrupt_disable(SPI_I2C_TypeDef *p_ssp) |
| 功能 | 禁用数据溢出中断 |
| 参数 1 | p_ssp：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.12 函数 ll_spi_iic_rfifo_not_empty_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_spi_iic_rfifo_not_empty_interrupt_enable(SPI_I2C_TypeDef *p_ssp) |
| 功能 | 使能接收 fifo 不为空中断 |
| 参数 1 | p_ssp：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.13 函数 ll_spi_iic_rfifo_not_empty_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_spi_iic_rfifo_not_empty_interrupt_disable(SPI_I2C_TypeDef *p_ssp) |
| 功能 | 禁用接收 fifo 不为空中断 |
| 参数 1 | p_ssp：spi 寄存器基地址 |

| | |
|---|---|
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.14 函数 ll_spi_iic_tfifo_not_full_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_spi_iic_tfifo_not_full_interrupt_enable(SPI_I2C_TypeDef *p_ssp) |
| 功能 | 使能发送 fifo 不满中断 |
| 参数 1 | p_ssp：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.15 函数 ll_spi_iic_tfifo_not_full_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_spi_iic_tfifo_not_full_interrupt_disable(SPI_I2C_TypeDef *p_ssp) |
| 功能 | 禁用发送 fifo 不满中断 |
| 参数 1 | p_ssp：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.16 函数 ll_spi_iic_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_spi_iic_interrupt_enable(SPI_I2C_TypeDef *p_ssp) |
| 功能 | 使能 spi iic 中断 |
| 参数 1 | p_ssp：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.17 函数 ll_spi_iic_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_spi_iic_interrupt_disable(SPI_I2C_TypeDef *p_ssp) |
| 功能 | 禁用 spi iic 中断 |
| 参数 1 | p_ssp：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.18 函数 ll_spi_iic_dma_enable

| | |
|---|---|
| 函数名 | void ll_spi_iic_dma_enable(SPI_I2C_TypeDef *p_ssp) |
| 功能 | 使能 spi iic dma 传输 |
| 参数 1 | p_ssp：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.19 函数 ll_spi_iic_dma_disable

| | |
|---|---|
| 函数名 | void ll_spi_iic_dma_disable(SPI_I2C_TypeDef *p_ssp) |
| 功能 | 禁用 spi iic dma 传输 |
| 参数 1 | p_ssp：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.20 函数 ll_spi_iic_tx_enable

| | |
|---|---|
| 函数名 | void ll_spi_iic_tx_enable(SPI_I2C_TypeDef *p_ssp) |
| 功能 | 使能 spi iic 发送 |
| 参数 1 | p_ssp：spi 寄存器基地址 |

| | |
|---|---|
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.21 函数 ll_spi_iic_tx_disable

| | |
|---|---|
| 函数名 | void ll_spi_iic_tx_disable(SPI_I2C_TypeDef *p_ssp) |
| 功能 | 禁用 spi iic 发送 |
| 参数 1 | p_ssp：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.22 函数 ll_spi_iic_enable

| | |
|---|---|
| 函数名 | void ll_spi_iic_enable(SPI_I2C_TypeDef *p_ssp) |
| 功能 | 使能 spi iic |
| 参数 1 | p_ssp：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.23 函数 ll_spi_iic_disable

| | |
|---|---|
| 函数名 | void ll_spi_iic_disable(SPI_I2C_TypeDef *p_ssp) |
| 功能 | 禁用 spi iic |
| 参数 1 | p_ssp：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.24 函数 ll_spi_iic_stop

| | |
|---|---|
| 函数名 | void ll_spi_iic_stop(SPI_I2C_TypeDef *p_spi) |
| 功能 | 停止 spi iic |
| 参数 1 | p_spi：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.25 函数 ll_spi_iic_fifo_empty

| | |
|---|---|
| 函数名 | bool ll_spi_iic_fifo_empty(SPI_I2C_TypeDef *p_spi) |
| 功能 | 判断 fifo 是否为空 |
| 参数 1 | p_spi：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | true ：fifo 未空　　false:fifo 空 |

### 11.26 函数 ll_spi_iic_fifo_full

| | |
|---|---|
| 函数名 | bool ll_spi_iic_fifo_full(SPI_I2C_TypeDef *p_spi) |
| 功能 | 判断 fifo 是否满 |
| 参数 1 | p_spi：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | true ：fifo 满　　false:fifo 未满 |

### 11.27 函数 ll_spi_cs_rising_edge_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_spi_cs_rising_edge_interrupt_enable(SPI_I2C_TypeDef *p_spi) |
| 功能 | 使能 spi cs 引脚上升沿中断 |
| 参数 1 | p_spi：spi 寄存器基地址 |

| | |
|---|---|
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.28 函数 ll_spi_cs_rising_edge_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_spi_cs_rising_edge_interrupt_disable(SPI_I2C_TypeDef *p_spi) |
| 功能 | 禁用 spi cs 引脚上升沿中断 |
| 参数 1 | p_spi：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.29 函数 ll_spi_cs_enable

| | |
|---|---|
| 函数名 | void ll_spi_cs_enable(SPI_I2C_TypeDef *p_spi) |
| 功能 | spi cs 使能 |
| 参数 1 | p_spi：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.30 函数 ll_spi_cs_disable

| | |
|---|---|
| 函数名 | void ll_spi_cs_disable(SPI_I2C_TypeDef *p_spi) |
| 功能 | 禁用 spi cs |
| 参数 1 | p_spi：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.31 函数 ll_spi_slave_sync_enable

| | |
|---|---|
| 函数名 | void ll_spi_slave_sync_enable(SPI_I2C_TypeDef *p_spi) |
| 功能 | SPI 从模式下，使能输入数据同步 |
| 参数 1 | p_spi：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.32 函数 ll_spi_slave_sync_disable

| | |
|---|---|
| 函数名 | void ll_spi_slave_sync_disable(SPI_I2C_TypeDef *p_spi) |
| 功能 | SPI 从模式下，禁用输入数据同步 |
| 参数 1 | p_spi：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.33 函数 ll_spi_master_sync_enable

| | |
|---|---|
| 函数名 | void ll_spi_master_sync_enable(SPI_I2C_TypeDef *p_spi) |
| 功能 | SPI 主模式下，使能输入数据同步 |
| 参数 1 | p_spi：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.34 函数 ll_spi_master_sync_disable

| | |
|---|---|
| 函数名 | void ll_spi_master_sync_disable(SPI_I2C_TypeDef *p_spi) |
| 功能 | SPI 主模式下，禁用输入数据同步 |
| 参数 1 | p_spi：spi 寄存器基地址 |

| | |
|---|---|
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.35 函数 ll_spi_wire_mode_get

| | |
|---|---|
| 函数名 | TYPE_ENUM_LL_SPI_WIRE_MODE<br>ll_spi_wire_mode_get(SPI_I2C_TypeDef *p_spi) |
| 功能 | 获取 SPI 连接方式 |
| 参数 1 | p_spi：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | spi 连接方式 |

| 定义 | 描述 |
|---|---|
| LL_SPI_NORMAL_MODE | spi 通常数据模式 |
| LL_SPI_THREE_WIRE_MODE | spi 3 数据线模式 |
| LL_SPI_DUAL_MODE | spi 2 数据线模式 |
| LL_SPI_QUAD_MODE | spi 4 数据线模式 |

### 11.36 函数 ll_spi_wire_mode_set

| | |
|---|---|
| 函数名 | void ll_spi_wire_mode_set(SPI_I2C_TypeDef *p_spi,<br>TYPE_ENUM_LL_SPI_WIRE_MODE wire_mode) |
| 功能 | 获取 SPI 连接方式 |
| 参数 1 | p_spi：spi 寄存器基地址 |
| 参数 2 | wire_mode：spi 连接方式 |
| 参数 3 | 无 |
| 返回值 | spi 连接方式 |

| 定义 | 描述 |
|---|---|
| LL_SPI_NORMAL_MODE | spi 通常数据模式 |

| | |
|---|---|
| LL_SPI_THREE_WIRE_MODE | spi 3 数据线模式 |
| LL_SPI_DUAL_MODE | spi 2 数据线模式 |
| LL_SPI_QUAD_MODE | spi 4 数据线模式 |

### 11.37 函数 ll_spi_cs_set

| | |
|---|---|
| 函数名 | void ll_spi_cs_set(SPI_I2C_TypeDef *p_spi) |
| 功能 | SPI 主模式下，cs 输出高 |
| 参数 1 | p_spi：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.38 函数 ll_spi_cs_clr

| | |
|---|---|
| 函数名 | void ll_spi_cs_clr(SPI_I2C_TypeDef *p_spi) |
| 功能 | SPI 主模式下，cs 输出低 |
| 参数 1 | p_spi：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.39 函数 ll_iic_rx_nack_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_iic_rx_nack_interrupt_enable(SPI_I2C_TypeDef *p_iic) |
| 功能 | 使能 iic 接收无响应中断 |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.40 函数 ll_iic_rx_nack_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_iic_rx_nack_interrupt_disable(SPI_I2C_TypeDef *p_iic) |
| 功能 | 禁用 iic 接收无响应中断 |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.41 函数 ll_iic_al_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_iic_al_interrupt_enable(SPI_I2C_TypeDef *p_iic) |
| 功能 | 使能 iic 主机仲裁丢失中断 |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.42 函数 ll_iic_al_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_iic_al_interrupt_disable(SPI_I2C_TypeDef *p_iic) |
| 功能 | 禁用 iic 主机仲裁丢失中断 |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.43 函数 ll_iic_stop_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_iic_stop_interrupt_enable(SPI_I2C_TypeDef *p_iic) |
| 功能 | 使能 iic 检测到停止信号中断 |
| 参数 1 | p_iic：spi 寄存器基地址 |

| | |
|---|---|
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.44 函数 ll_iic_stop_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_iic_stop_interrupt_disable(SPI_I2C_TypeDef *p_iic) |
| 功能 | 禁用 iic 检测到停止信号中断 |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.45 函数 ll_iic_addr_match_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_iic_addr_match_interrupt_enable(SPI_I2C_TypeDef *p_iic) |
| 功能 | 使能 IIC 从模式下地址匹配中断 |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.46 函数 ll_iic_addr_match_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_iic_addr_match_interrupt_disable(SPI_I2C_TypeDef *p_iic) |
| 功能 | 禁用 IIC 从模式下地址匹配中断 |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.47 函数 ll_iic_broadcast_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_iic_broadcast_interrupt_enable(SPI_I2C_TypeDef *p_iic) |
| 功能 | 使能 IIC 广播中断 |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.48 函数 ll_iic_broadcast_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_iic_broadcast_interrupt_disable(SPI_I2C_TypeDef *p_iic) |
| 功能 | 禁用 IIC 广播中断 |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.49 函数 ll_iic_broadcast_enable

| | |
|---|---|
| 函数名 | void ll_iic_broadcast_enable(SPI_I2C_TypeDef *p_iic) |
| 功能 | 使能 IIC 广播 |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.50 函数 ll_iic_broadcast_disable

| | |
|---|---|
| 函数名 | void ll_iic_broadcast_disable(SPI_I2C_TypeDef *p_iic) |
| 功能 | 禁用 IIC 广播 |

| | |
|---|---|
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.51 函数 ll_iic_tx_nack_enable

| | |
|---|---|
| 函数名 | void ll_iic_tx_nack_enable(SPI_I2C_TypeDef *p_iic) |
| 功能 | 使能 IIC 响应 NACK |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.52 函数 ll_iic_tx_nack_disable

| | |
|---|---|
| 函数名 | void ll_iic_tx_nack_enable(SPI_I2C_TypeDef *p_iic) |
| 功能 | 禁用 IIC 响应 NACK |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 11.53 函数 ll_iic_bus_is_busy

| | |
|---|---|
| 函数名 | bool ll_iic_bus_is_busy(SPI_I2C_TypeDef *p_iic) |
| 功能 | 判断 iic 是否被占用 |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | True：被占用　　false：未被占用 |

### 11.54 函数 ll_iic_arbitration_is_lost

| | |
|---|---|
| 函数名 | bool ll_iic_arbitration_is_lost(SPI_I2C_TypeDef *p_iic) |
| 功能 | 判断 IIC 仲裁丢失 |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | True：有数据丢失　　false：没有数据丢失 |

### 11.55 函数 ll_iic_slave_addr_response

| | |
|---|---|
| 函数名 | TYPE_ENUM_LL_IIC_ADDR_RESPONSE<br>ll_iic_slave_addr_response(SPI_I2C_TypeDef *p_iic) |
| 功能 | IIC 从模式下地址响应判断（如果地址匹配，则判断写入或读取方向） |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | LL_IIC_NO_RESPONSE：没有响应<br>LL_IIC_READ：读取数据<br>LL_IIC_WRITE：写入数据 |

### 11.56 函数 ll_iic_slave_rx_broadcast

| | |
|---|---|
| 函数名 | bool ll_iic_slave_rx_broadcast(SPI_I2C_TypeDef *p_iic) |
| 功能 | IIC 从模式下接收广播判断 |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | True：检测到广播地址　　false：没有检测到广播地址 |

### 11.57 函数 ll_iic_get_ack_state

| | |
|---|---|
| 函数名 | TYPE_ENUM_LL_IIC_ACK ll_iic_get_ack_state(SPI_I2C_TypeDef *p_iic) |
| 功能 | 获取 iic 应答状态 |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | LL_IIC_ACK：检测应答信号为 ack　　LL_IIC_NACK：检测应答信号为 nack |

### 11.58 函数 ll_iic_slave_rx_byte

| | |
|---|---|
| 函数名 | u8 ll_iic_slave_rx_byte(SPI_I2C_TypeDef *p_iic) |
| 功能 | IIC 从模式下接收 1 字节数据 |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | 无 |
| 参数 3 | 无 |
| 返回值 | 接收的数据 |

### 11.59 函数 ll_iic_slave_tx_byte

| | |
|---|---|
| 函数名 | void ll_iic_slave_tx_byte(SPI_I2C_TypeDef *p_iic, u8 data) |
| 功能 | IIC 从模式下发送 1 字节数据 |
| 参数 1 | p_iic：spi 寄存器基地址 |
| 参数 2 | data：要发送的数据 |
| 返回值 | 无 |

### 11.60 模块相关宏定义

| 定义 | 描述 |
|---|---|
| LL_SPI_IIC_DMA_INTERRUPT_GET(p_ssp) | DMA 完成中断获取 |
| LL_SPI_IIC_FIFO_OV_INTERRUPT_GET(p_ssp) | buf 溢出中断获取 |

| | |
|---|---|
| LL_SPI_IIC_RFIFO_NOT_EMPTY_INTERRUPT_GET(p_ssp) | 接收 buf 不空中断获取 |
| LL_SPI_IIC_TFIFO_NOT_FULL_INTERRUPT_GET(p_ssp) | 发送 buf 不满中断获取 |
| LL_SPI_IIC_INTERRUPT_GET(p_ssp) | SPI_IIC 中断获取 |
| LL_SPI_CS_RISING_EDGE_INTERRUPT_GET(p_spi) | SPI cs 上升沿中断获取 |
| LL_IIC_RX_NACK_INTERRUPT_GET(p_iic) | IIC 接收 NACK 中断获取 |
| LL_IIC_AL_INTERRUPT_GET(p_iic) | IIC 主模式下仲裁丢失中断获取 |
| LL_IIC_STOP_INTERRUPT_GET(p_iic) | 检测到停止信号中断获取 |
| LL_IIC_ADDR_MATCH_INTERRUPT_GET(p_iic) | IIC 从模式下地址匹配中断获取 |
| LL_IIC_BROADCAST_INTERRUPT_GET(p_iic) | IIC 广播中断获取 |
| LL_SPI_IIC_DMA_EN_GET(p_ssp) | DMA 使能获取 |
| LL_SPI_IIC_TX_EN_GET(p_ssp) | 发送使能获取 |
| LL_SPI_IIC_EN_GET(p_ssp) | SPI_IIC 使能获取 |
| LL_SPI_CS_EN_GET(p_spi) | SPI cs 使能获取 |
| LL_SPI_SLAVE_SYNC_EN_GET(p_spi) | SPI 从模式下需要同步使能获取 |
| LL_IIC_BROADCAST_EN_GET(p_iic) | IIC 广播使能获取 |
| LL_IIC_TX_NACK_EN_GET(p_iic) | IIC 响应 nack 使能获取 |
| LL_SPI_IIC_BUSY_PENDING_GET(p_ssp) | IIC 线路被占用状态获取 |
| LL_SPI_IIC_DMA_DONE_PENDING_GET(p_ssp) | DMA 完成状态获取 |
| LL_SPI_IIC_DMA_DONE_PENDING_CLR(p_ssp) | DMA 完成状态清除 |
| LL_SPI_IIC_FIFO_OV_PENDING_GET(p_ssp) | buf 溢出状态获取 |
| LL_SPI_IIC_FIFO_OV_PENDING_CLR(p_ssp) | buf 溢出状态清除 |
| LL_SPI_IIC_FIFO_EMPTY_PENDING_GET(p_ssp) | buf 空状态获取 |
| LL_SPI_IIC_FIFO_FULL_PENDING_GET(p_ssp) | buf 满状态获取 |
| LL_SPI_IIC_DONE_PENDING_GET(p_ssp) | SPI_IIC 完成状态获取 |
| LL_SPI_IIC_DONE_PENDING_CLR(p_ssp) | SPI_IIC 完成状态清除 |

| | |
|---|---|
| LL_SPI_CS_RISING_EDGE_PENDING_GET(p_spi) | SPI cs 上升沿状态获取 |
| LL_SPI_CS_RISING_EDGE_PENDING_GET(p_spi) | SPI cs 下降沿状态获取 |
| LL_IIC_MASTER_RX_BUSY_PENDING_GET(p_iic) | IIC 主模式下接收忙状态获取 |
| LL_IIC_ BUS_BUSY_PENDING_GET(p_iic) | IIC 下线路忙状态获取 |
| LL_IIC_AL_PENDING_GET(p_iic) | IIC 仲裁丢失检测状态获取 |
| LL_IIC_AL_PENDING_CLR(p_iic) | IIC 仲裁丢失检测状态清除 |
| LL_IIC_STOP_PENDING_GET(p_iic) | IIC 检测到停止位状态获取 |
| LL_IIC_STOP_PENDING_CLR(p_iic) | IIC 检测到停止位状态清除 |
| LL_IIC_ADDR_MATCH_PENDING_GET(p_iic) | IIC 地址匹配状态获取 |
| LL_IIC_ADDR_MATCH_PENDING_CLR(p_iic) | IIC 地址匹配状态清除 |
| LL_IIC_BROADCAST_PENDING_GET(p_iic) | IIC 检测到广播地址状态获取 |
| LL_IIC_BROADCAST_PENDING_CLR(p_iic) | IIC 检测到广播地址状态清除 |

# 12 系统控制单元

## 12.1 函数 NVIC_Init

| | |
|---|---|
| 函数名 | void NVIC_Init(NVIC_InitTypeDef* NVIC_InitStruct) |
| 功能 | NVIC 初始化 |
| 参数 1 | NVIC_InitStruct：指向包含了指定外设的配置信息的结构体 |
| 参数 2 | 无 |
| 返回值 | 无 |

**NVIC_IRQChannel**：用或禁用的 IRQ 通道。这个参数可以是 IRQn_Type 的值

**NVIC_IRQChannelPriority**：中断优先级（0-3），数值越低，优先级越高

**NVIC_IRQChannelCmd**：允许或禁用中断

| 定义 | 描述 |
|---|---|
| DISABLE | 不使能 |
| ENABLE | 使能 |

## 12.2 函数 NVIC_SystemLPConfig

| | |
|---|---|
| 函数名 | void NVIC_SystemLPConfig(u8 LowPowerMode, FunctionalState NewState) |
| 功能 | 系统低功耗模式配置 |
| 参数 1 | LowPowerMode：模式选择 |
| 参数 2 | NewState：开启或关闭 |
| 返回值 | 无 |

**LowPowerMode**：模式选择

**NewState**：开启或关闭

| 定义 | 描述 |
|---|---|
| DISABLE | 不使能 |

| | |
|---|---|
| ENABLE | 使能 |

### 12.3 函数 SysTick_CLKSourceConfig

| | |
|---|---|
| 函数名 | void SysTick_CLKSourceConfig(u32 SysTick_CLKSource) |
| 功能 | 配置 SysTick 时钟源 |
| 参数 1 | SysTick_CLKSource：SysTick_CLKSource:时钟源，可以使以下两个值之一 SysTick_CLKSource_HCLK_Div8，SysTick_CLKSource_HCLK |
| 参数 2 | 无 |
| 返回值 | 无 |

### 12.4 函数 sys_clk_rc32k

| | |
|---|---|
| 函数名 | void sys_clk_rc32k(void) |
| 功能 | 系统时钟选择内部低速时钟 rc32KHz |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 12.5 函数 sys_clk_xoscm

| | |
|---|---|
| 函数名 | void sys_clk_xoscm(void) |
| 功能 | 系统时钟选择外部晶振 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 12.6 函数 sys_clk_hirc

| | |
|---|---|
| 函数名 | void sys_clk_hirc(void) |
| 功能 | 系统时钟选择内部高速 26MHz 时钟 |
| 参数 1 | 无 |

| | |
|---|---|
| 参数 2 | 无 |
| 返回值 | 无 |

### 12.7 函数 sys_clk_pll

| | |
|---|---|
| 函数名 | void sys_clk_pll(void) |
| 功能 | 系统时钟选择 PLL 时钟(HIRC26MHZ * 2 = 52MHZ) |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 12.8 函数 sys_special_fun_init

| | |
|---|---|
| 函数名 | void sys_special_fun_init(void) |
| 功能 | 系统特殊功能初始化 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 12.9 函数 delay_ms

| | |
|---|---|
| 函数名 | void delay_ms(u32 n) |
| 功能 | 延时 n * ms（使用系统时钟） |
| 参数 1 | n |
| 参数 2 | 无 |
| 返回值 | 无 |

### 12.10 函数 delay_us

| | |
|---|---|
| 函数名 | void delay_us(u32 n) |
| 功能 | 延时 n * us（使用系统时钟） |
| 参数 1 | n |

| | |
|---|---|
| 参数 2 | 无 |
| 返回值 | 无 |

### 12.11 函数 sys_init

| | |
|---|---|
| 函数名 | void sys_init(void) |
| 功能 | 系统初始化 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 12.12 函数 pll_init

| | |
|---|---|
| 函数名 | void pll_init(void) |
| 功能 | PLL 初始化 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 12.13 函数 SystemInit

| | |
|---|---|
| 函数名 | void SystemInit(void) |
| 功能 | 设置微控制器系统,初始化嵌入式 Flash 接口，pll 和更新 SystemCoreClock 变量。 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 12.14 函数 is_systick_expired

| | |
|---|---|
| 函数名 | u32 is_systick_expired(s32 offset_ticks, s32 Texpire) |
| 功能 | 判断嘀嗒时钟定时时间是否到了 |

| | |
|---|---|
| 参数 1 | offset_ticks：系统嘀嗒时钟当前计数值 |
| 参数 2 | Texpire：总定时时间 |
| 返回值 | true or false |

### 12.15 函数 GetSysTick

| | |
|---|---|
| 函数名 | u32 GetSysTick(void) |
| 功能 | 获取当前计数器值 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 当前计数值 |

### 12.16 函数 SystemTickInit

| | |
|---|---|
| 函数名 | u32 SystemTickInit(void) |
| 功能 | 初始化滴答定时器 1ms |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | true or false |

# 13 定时器 timer

定时器 timer0/1/2/3/5 由一个 16 位的计数器组成，timer4 由一个 32 位的计数器组成。支持定时功能，可选择不同的计数源（系统时钟、内部低速 RC 时钟、外部时钟、GPIO 等），同时支持捕获和 PWM 输出功能,另外 timer5 支持红外发射功能。

## 13.1 函数 ll_timer_init

| | |
|---|---|
| 函数名 | void ll_timer_init(TIMER_TypeDef *p_timer, TYPE_LL_TIMER_INIT *p_init) |
| 功能 | 初始化定时器 |
| 参数 1 | p_timer：timer 寄存器基地址（timer0/1/2/3/5） |
| 参数 2 | p_init：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_timer_init {
    TYPE_ENUM_LL_TIMER_SRC_SEL      timer_src_sel;
    TYPE_ENUM_LL_TIMER_PSC          prescaler;
} TYPE_LL_TIMER_INIT;
```

**timer_src_sel**：时钟源选择

| 定义 | 描述 |
|---|---|
| LL_TIMER_SRC_INC_PIN_RISING | 引脚上升沿 |
| LL_TIMER_SRC_INC_PIN_FALLING | 引脚下降沿 |
| LL_TIMER_SRC_INTERNAL_HIGH_SPEED_RC | 内部时钟 RC 除以 2 时钟上升沿和下降沿 |
| LL_TIMER_SRC_INTERNAL_32KHZ_RC | 内部 32k 除以 2 时钟上升沿和下降沿 |
| LL_TIMER_SRC_EXTERNAL_CRYSTAL | 外部晶振除以 2 时钟上升沿和下降沿 |
| LL_TIMER_SRC_SYS_RISING | 系统时钟上升沿 |

**Prescaler**：预分频系数选择

| 定义 | 描述 |
|---|---|

| | |
|---|---|
| LL_TIMER_PSC_NONE | 不分频 |
| LL_TIMER_PSC_2 | 2 分频 |
| LL_TIMER_PSC_4 | 4 分频 |
| LL_TIMER_PSC_8 | 8 分频 |
| LL_TIMER_PSC_16 | 16 分频 |
| LL_TIMER_PSC_32 | 32 分频 |
| LL_TIMER_PSC_64 | 64 分频 |
| LL_TIMER_PSC_128 | 128 分频 |

### 13.2 函数 ll_timer_deinit

| | |
|---|---|
| 函数名 | void ll_timer_deinit(TIMER_TypeDef *p_timer) |
| 功能 | 释放定时器 |
| 参数 1 | p_timer：timer 寄存器基地址（timer0/1/2/3/5） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 13.3 函数 ll_timer_stop

| | |
|---|---|
| 函数名 | void ll_timer_stop(TIMER_TypeDef *p_timer) |
| 功能 | 禁用定时器 |
| 参数 1 | p_timer：timer 寄存器基地址（timer0/1/2/3/5） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 13.4 函数 ll_timer_start

| | |
|---|---|
| 函数名 | void ll_timer_start(TIMER_TypeDef *p_timer,<br>TYPE_ENUM_LL_TIMER_MODE_SEL mode_sel) |
| 功能 | 启动定时器 |
| 参数 1 | p_timer：timer 寄存器基地址（timer0/1/2/3/5） |

| | |
|---|---|
| 参数 2 | mode_sel：模式选择 |
| 返回值 | 无 |

**mode_sel**：模式选择

| 定义 | 描述 |
|---|---|
| LL_TIMER_MODE_SEL_DISABLE | 禁用计时器 |
| LL_TIMER_MODE_SEL_COUNTER | 计数器模式 |
| LL_TIMER_MODE_SEL_PWM | Pwm 模式 |
| LL_TIMER_MODE_SEL_CAPTURE | 捕获模式 |

### 13.5 函数 ll_timer_cnt_set

| | |
|---|---|
| 函数名 | void ll_timer_cnt_set(TIMER_TypeDef *p_timer, u32 cnt_set) |
| 功能 | 设置定时器计数值 |
| 参数 1 | p_timer：timer 寄存器基地址（timer0/1/2/3/5） |
| 参数 2 | cnt_set：计数值 |
| 返回值 | 无 |

### 13.6 函数 ll_timer_cnt_get

| | |
|---|---|
| 函数名 | u32 ll_timer_cnt_get(TIMER_TypeDef *p_timer) |
| 功能 | 获取定时器计数值 |
| 参数 1 | p_timer：timer 寄存器基地址（timer0/1/2/3/4） |
| 参数 2 | 无 |
| 返回值 | 当前计数值 |

### 13.7 函数 ll_timer_cnt_mode_config

| | |
|---|---|
| 函数名 | void ll_timer_cnt_mode_config(TIMER_TypeDef *p_timer,<br>TYPE_LL_TIMER_CNT_CFG *cnt_cfg) |
| 功能 | 定时器模式配置 |

| | |
|---|---|
| 参数 1 | p_timer：timer 寄存器基地址（timer0/1/2/3/5） |
| 参数 2 | cnt_cfg：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_timer_cnt_cfg {
    u32                 count_period;
    u32                 count_initial;
    FunctionalState     count_ie;
} TYPE_LL_TIMER_CNT_CFG;
```

**count_period**：计数周期

**count_initial**：计数初始值

**count_ie**：计数中断使能

| 定义 | 描述 |
|---|---|
| ENABLE | 使能开 |
| DISABLE | 使能关 |

## 13.8 函数 ll_timer_pwm_mode_config

| | |
|---|---|
| 函数名 | void ll_timer_pwm_mode_config(TIMER_TypeDef *p_timer,<br>TYPE_LL_TIMER_PWM_CFG *pwm_cfg) |
| 功能 | PWM 模式配置 |
| 参数 1 | p_timer：timer 寄存器基地址（timer0/1/2/3/5） |
| 参数 2 | pwm_cfg：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_timer_pwm_cfg {
    u32                 pwm_period;
    u32                 pwm_duty;
} TYPE_LL_TIMER_PWM_CFG;
```

**pwm_period**：PWM 周期

**pwm_duty**：PWM 占空比

### 13.9 函数 ll_timer_cap_mode_config

| | |
|---|---|
| 函数名 | void ll_timer_cap_mode_config(TIMER_TypeDef *p_timer, TYPE_LL_TIMER_CAP_CFG *cap_cfg) |
| 功能 | 捕获模式配置 |
| 参数 1 | p_timer：timer 寄存器基地址（timer0/1/2/3/5） |
| 参数 2 | cap_cfg：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_timer_capture_cfg {
    TYPE_ENUM_LL_TIMER_CAP_EDGE    capture_edge_sel;
    FunctionalState                capture_ie;
} TYPE_LL_TIMER_CAP_CFG;
```

**capture_edge_sel**：触发源选择

| 定义 | 描述 |
|---|---|
| LL_TIMER_EDGE_SEL_RISING | 上升沿触发 |
| LL_TIMER_EDGE_SEL_FALLING | 下降沿触发 |
| LL_TIMER_EDGE_SEL_RISING_FALLING | 上升沿和下降沿触发 |

**capture_ie**：捕获中断

| 定义 | 描述 |
|---|---|
| ENABLE | 使能开 |
| DISABLE | 使能关 |

## 13.10 函数 ll_irtimer_init

| | |
|---|---|
| 函数名 | void ll_irtimer_init(TIMER5_TypeDef *p_timer, TYPE_LL_TIMER_INIT *p_init) |
| 功能 | 红外定时器初始化 |
| 参数 1 | p_timer：timer 寄存器基地址（timer5） |
| 参数 2 | p_init：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_timer_init {
    TYPE_ENUM_LL_TIMER_SRC_SEL      timer_src_sel;
    TYPE_ENUM_LL_TIMER_PSC          prescaler;
} TYPE_LL_TIMER_INIT;
```

**timer_src_sel**：时钟源选择

| 定义 | 描述 |
|---|---|
| LL_TIMER_SRC_INC_PIN_RISING | 引脚上升沿 |
| LL_TIMER_SRC_INC_PIN_FALLING | 引脚下降沿 |
| LL_TIMER_SRC_INTERNAL_HIGH_SPEED_RC | 内部时钟 RC 除以 2 时钟上升沿和下降沿 |
| LL_TIMER_SRC_INTERNAL_32KHZ_RC | 内部 32k 除以 2 时钟上升沿和下降沿 |
| LL_TIMER_SRC_EXTERNAL_CRYSTAL | 外部晶振除以 2 时钟上升沿和下降沿 |
| LL_TIMER_SRC_SYS_RISING | 系统时钟上升沿 |

**Prescaler**：预分频系数选择

| 定义 | 描述 |
|---|---|
| LL_TIMER_PSC_NONE | 不分频 |
| LL_TIMER_PSC_2 | 2 分频 |
| LL_TIMER_PSC_4 | 4 分频 |
| LL_TIMER_PSC_8 | 8 分频 |
| LL_TIMER_PSC_16 | 16 分频 |

| | |
|---|---|
| LL_TIMER_PSC_32 | 32 分频 |
| LL_TIMER_PSC_64 | 64 分频 |
| LL_TIMER_PSC_128 | 128 分频 |

### 11.11 函数 ll_irtimer_stop

| | |
|---|---|
| 函数名 | void ll_irtimer_stop(TIMER5_TypeDef *p_timer) |
| 功能 | 禁用红外定时器 |
| 参数 1 | p_timer：timer 寄存器基地址（timer5） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 13.12 函数 ll_irtimer_start

| | |
|---|---|
| 函数名 | void ll_irtimer_start(TIMER5_TypeDef *p_timer,<br>TYPE_ENUM_LL_TIMER_MODE_SEL mode_sel) |
| 功能 | 启动红外定时器 |
| 参数 1 | p_timer：timer 寄存器基地址（timer5） |
| 参数 2 | mode_sel：模式设置 |
| 返回值 | 无 |

**mode_sel**：模式选择

| 定义 | 描述 |
|---|---|
| LL_TIMER_MODE_SEL_DISABLE | 禁用计时器 |
| LL_TIMER_MODE_SEL_COUNTER | 计数器模式 |
| LL_TIMER_MODE_SEL_PWM | Pwm 模式 |
| LL_TIMER_MODE_SEL_CAPTURE | 捕获模式 |

### 11.13 函数 ll_irtimer_cnt_set

| | |
|---|---|
| 函数名 | void ll_irtimer_cnt_set(TIMER5_TypeDef *p_timer, u32 cnt_set) |

| | |
|---|---|
| 功能 | 设置红外定时器计数值 |
| 参数 1 | p_timer：timer 寄存器基地址（timer5） |
| 参数 2 | cnt_set：设置计数值 |
| 返回值 | 无 |

### 11.14 函数 ll_irtimer_cnt_get

| | |
|---|---|
| 函数名 | u32 ll_irtimer_cnt_get(TIMER5_TypeDef *p_timer) |
| 功能 | 获取红外定时器计数值 |
| 参数 1 | p_timer：timer 寄存器基地址（timer5） |
| 参数 2 | 无 |
| 返回值 | 计数值 |

### 13.15 函数 ll_irtimer_cnt_mode_config

| | |
|---|---|
| 函数名 | void ll_irtimer_cnt_mode_config(TIMER5_TypeDef *p_timer,<br>TYPE_LL_TIMER_CNT_CFG *cnt_cfg) |
| 功能 | 配置红外定时器计数模式 |
| 参数 1 | p_timer：timer 寄存器基地址（timer5） |
| 参数 2 | cnt_cfg：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_timer_cnt_cfg {
    u32                 count_period;
    u32                 count_initial;
    FunctionalState     count_ie;
} TYPE_LL_TIMER_CNT_CFG;
```

**count_period**：计数周期

**count_initial**：计数初始值

**count_ie**：计数中断使能

| 定义 | 描述 |
| --- | --- |
| ENABLE | 使能开 |
| DISABLE | 使能关 |

## 13.16 函数 ll_irtimer_pwm_mode_config

| 函数名 | void ll_irtimer_pwm_mode_config(TIMER5_TypeDef *p_timer,<br>TYPE_LL_TIMER_PWM_CFG *pwm_cfg) |
| --- | --- |
| 功能 | 配置红外定时器 pwm 模式 |
| 参数 1 | p_timer：timer 寄存器基地址（timer5） |
| 参数 2 | pwm_cfg：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_timer_pwm_cfg {
    u32                         pwm_period;
    u32                         pwm_duty;
} TYPE_LL_TIMER_PWM_CFG;
```

**pwm_period**：PWM 周期

**pwm_duty**：PWM 占空比

## 13.17 函数 ll_irtimer_cap_mode_config

| 函数名 | void ll_irtimer_cap_mode_config(TIMER5_TypeDef *p_timer,<br>TYPE_LL_TIMER_CAP_CFG *cap_cfg) |
| --- | --- |
| 功能 | 配置红外定时器捕获模式 |
| 参数 1 | p_timer：timer 寄存器基地址（timer5） |
| 参数 2 | cap_cfg：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_timer_capture_cfg {
    TYPE_ENUM_LL_TIMER_CAP_EDGE     capture_edge_sel;
    FunctionalState                 capture_ie;
} TYPE_LL_TIMER_CAP_CFG;
```

**capture_edge_sel**：触发源选择

| 定义 | 描述 |
|---|---|
| LL_TIMER_EDGE_SEL_RISING | 上升沿触发 |
| LL_TIMER_EDGE_SEL_FALLING | 下降沿触发 |
| LL_TIMER_EDGE_SEL_RISING_FALLING | 上升沿和下降沿触发 |

**capture_ie**：捕获中断

| 定义 | 描述 |
|---|---|
| ENABLE | 使能开 |
| DISABLE | 使能关 |

### 13.18 函数 ll_ir_tx_init

| 函数名 | void ll_ir_tx_init(TIMER5_TypeDef *p_timer, TYPE_LL_IR_TX_CFG *ir_tx_cfg) |
|---|---|
| 功能 | 红外发射初始化 |
| 参数 1 | p_timer：timer 寄存器基地址（timer5） |
| 参数 2 | ir_tx_cfg：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_ir_tx_cfg{
    u16                 carrier_freq;
    u16                 bit_1_period_cnt;
    u16                 bit_1_carrier_cnt;
    u16                 bit_0_period_cnt;
    u16                 bit_0_carrier_cnt;
```

```
    FunctionalState          xor_en;
    FunctionalState          p_pol_en;
} TYPE_LL_IR_TX_CFG;
```

**carrier_freq**：载波频率

**bit_1_period_cnt**：设置红外传输 1 的周期时间

**bit_1_carrier_cnt**：设置红外传输 1 载波的持续时间

**bit_0_period_cnt**：设置红外传输 0 的周期时间

**bit_0_carrier_cnt**：设置红外传输 0 载波的持续时间

**xor_en**：异或使能，用于 RC5 红外发射协议

| 定义 | 描述 |
|---|---|
| ENABLE | 使能开 |
| DISABLE | 使能关 |

**p_pol_en**：控制 PWM 极性

| 定义 | 描述 |
|---|---|
| ENABLE | 使能开 |
| DISABLE | 使能关 |

### 13.19 函数 ll_ir_tx

| 函数名 | void ll_ir_tx(TIMER5_TypeDef *p_timer, TYPE_LL_IR_FRAME_CFG *p_ir_frame) |
|---|---|
| 功能 | 红外发射数据 |
| 参数 1 | p_timer：timer 寄存器基地址（timer5） |
| 参数 2 | p_ir_frame：指向包含了指定外设的配置信息的结构体 |

| | |
|---|---|
| 返回值 | 无 |

```
typedef struct __ll_ir_frame_cfg{
    u16                     start_bit_period_cnt;
    u16                     start_bit_carrier_cnt;
    u16                     *ir_buf;
    u8                      frame_bit_cnt;
} TYPE_LL_IR_FRAME_CFG;
```

**start_bit_period_cnt**：位周期计数值

**start_bit_carrier_cnt**：位载波计数值

***ir_buf**：红外数据缓冲，先发高位

**frame_bit_cnt**：红外帧数据位计数（不包含起始位）

### 11.20 函数 ll_ir_int_tx

| | |
|---|---|
| 函数名 | void ll_ir_tx(TIMER5_TypeDef *p_timer, TYPE_LL_IR_FRAME_CFG *p_ir_frame) |
| 功能 | 红外发射数据（中断模式） |
| 参数 1 | p_timer：timer 寄存器基地址（timer5） |
| 参数 2 | p_ir_frame：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

**start_bit_period_cnt**：位周期计数值

**start_bit_carrier_cnt**：位载波计数值

***ir_buf**：红外数据缓冲，先发高位

**frame_bit_cnt**：红外帧数据位计数（不包含起始位）

### 13.21 函数 ll_timer_cnt_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_timer_cnt_interrupt_enable(TIMER_TypeDef *p_timer) |
| 功能 | 定时器中断使能 |
| 参数 1 | p_timer：timer 寄存器基地址（timer0 ~ timer5） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 13.22 函数 ll_timer_cnt_interrrupt_disable

| | |
|---|---|
| 函数名 | void ll_timer_cnt_interrupt_disable(TIMER_TypeDef *p_timer) |
| 功能 | 禁用定时器中断 |
| 参数 1 | p_timer：timer 寄存器基地址（timer0 ~ timer5） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 13.23 函数 ll_timer_cap_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_timer_cap_interrupt_enable(TIMER_TypeDef *p_timer) |
| 功能 | 使能定时器捕获中断 |
| 参数 1 | p_timer：timer 寄存器基地址（timer0 ~ timer5） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 13.24 函数 ll_timer_cap_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_timer_cap_interrupt_disable(TIMER_TypeDef *p_timer) |
| 功能 | 禁用定时器捕获中断 |
| 参数 1 | p_timer：timer 寄存器基地址（timer0 ~ timer5） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 13.25 函数 ll_irtimer_buf_empty_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_irtimer_buf_empty_interrupt_enable(TIMER5_TypeDef *p_timer) |
| 功能 | 使能红外发送 buff 空中断 |
| 参数 1 | p_timer：timer 寄存器基地址（timer5） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 13.26 函数 ll_irtimer_buf_empty_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_irtimer_buf_empty_interrupt_disable(TIMER5_TypeDef *p_timer) |
| 功能 | 禁用红外发送 buff 空中断 |
| 参数 1 | p_timer：timer 寄存器基地址（timer5） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 13.27 函数 ll_irtimer_tx_done_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_irtimer_tx_done_interrupt_enable(TIMER5_TypeDef *p_timer) |
| 功能 | 使能红外发送完成中断 |
| 参数 1 | p_timer：timer 寄存器基地址（timer5） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 13.28 函数 ll_irtimer_tx_done_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_irtimer_tx_done_interrupt_disable(TIMER5_TypeDef *p_timer) |
| 功能 | 禁用红外发送完成中断 |
| 参数 1 | p_timer：timer 寄存器基地址（timer5） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 13.29 函数 ll_irtimer_ir_enable

| | |
|---|---|
| 函数名 | void ll_irtimer_ir_enable(TIMER5_TypeDef *p_timer) |
| 功能 | 使能红外发送完成中断 |
| 参数 1 | p_timer：timer 寄存器基地址（timer5） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 13.30 函数 ll_irtimer_ir_disable

| | |
|---|---|
| 函数名 | void ll_irtimer_ir_disable(TIMER5_TypeDef *p_timer) |
| 功能 | 禁用红外发送完成中断 |
| 参数 1 | p_timer：timer 寄存器基地址（timer5） |
| 参数 2 | 无 |
| 返回值 | 无 |

# 14 TK 触摸屏

本芯片采用的是电容式触摸按键。它可以穿透绝缘材料外壳 5mm (玻璃、塑料等等)以上，准确无误地侦测到手指的有效触摸。并保证了产品的灵敏度、稳定性、可靠性等不会因环境条件的改变或长期使用而发生变化，并具有防水和强抗干扰能力，超强防护，超强适应温度范围。适用于遥控器、灯具调光、各类开关以及车载、小家电和家用电器控制界面等应用中。

## 14.1 函数 ll_tk_init

| | |
|---|---|
| 函数名 | void ll_tk_init(TK_TypeDef *p_tk, TYPE_LL_TK_INIT *p_init) |
| 功能 | tk 初始化 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | p_init：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_tk_init {
    u8                          max_min_mode;
    TYPE_ENUM_LL_TK_BDEC_MODE       bdec_mode;
    u32                         fsdiv           : 4;
    TYPE_ENUM_LL_TK_CHARGE_MODE     charge_mode;
    TYPE_ENUM_LL_TK_OFFSET_MODE     offset_mode;
    FunctionalState              fixpr;
    u8                          bflt_mode       : 3;
    u8                          flt_mode        : 3;
    u8                          dis_charge_cnt;
    u32                         period_sample_num   : 5;
    u32                         per_sample_clk      : 5;
u16                             tk_wait_time;
u16                             tk_sample_time;
    u16                          mmstbthr;
    u32                          tk_buf;
    FunctionalState               scan_timeout_det_en;
    FunctionalState               scan_done_ie;
    FunctionalState               sample_ov_ie;
    FunctionalState               scan_timeout_ie;
```

```
    FunctionalState                 key_trigger_ie;
    FunctionalState                 auto_scan_en;
    FunctionalState                 rc_inv_en;
    u8                              precyc_num;
} TYPE_LL_TK_INIT;
```

**max_min_mode**：最大最小滤波方式

| 定义 | 描述 |
| --- | --- |
| LL_TK_MAX_MIN_FILTER_NONE | 不做去除最大最小值滤波 |
| LL_TK_MAX_MIN_FILTER_MODE_1 | 去掉一个最大值最小值 |
| LL_TK_MAX_MIN_FILTER_MODE_2 | 去掉两个个最大值最小值 |

**bdec_mode**：基准值采样频率倍速模式

| 定义 | 描述 |
| --- | --- |
| LL_TK_BDEC_MODE_0 | 不倍频 |
| LL_TK_BDEC_MODE_1 | 2 倍频 |
| LL_TK_BDEC_MODE_2 | 4 倍频 |
| LL_TK_BDEC_MODE_3 | 8 倍频 |
| LL_TK_BDEC_MODE_4 | 16 倍频 |
| LL_TK_BDEC_MODE_5 | 32 倍频 |
| LL_TK_BDEC_MODE_6 | 64 倍频 |
| LL_TK_BDEC_MODE_7 | 128 倍频 |

**fsdiv**：电荷转移模式充电开关分频比

| 定义 | 描述 |
| --- | --- |
| n | 分频比 = (n+1)*2 |

**charge_mode**：振荡器模式

| 定义 | 描述 |
| --- | --- |

| | |
|---|---|
| LL_TK_RELAXATION_OSC_MODE | 张弛振荡模式 |
| LL_TK_CAP_CHARGE_MODE | 电荷转移模式 |

**offset_mode**：阈值模式

| 定义 | 描述 |
|---|---|
| LL_TK_OFFSET_ABS_MODE | 绝对值模式 |
| LL_TK_OFFSET_PERCENT_MODE | 百分比模式 |

**fixpr**：按键扫描模式

| 定义 | 描述 |
|---|---|
| DISABLE | 不使能 |
| ENABLE | 使能 |

**bflt_mode**:基准值滤波模式

| 定义 | 描述 |
|---|---|
| n | n 越大，效果越好，响应速度越慢，不能为 0 |

**flt_mode**:采样滤波模式

| 定义 | 描述 |
|---|---|
| n | n 越大，效果越好，响应速度越慢，不能为 7 |

**dis_charge_cnt**：RC 振荡器放电时间

| 定义 | 描述 |
|---|---|
| n | 放电时间 = n * tkclk |

**period_sample_num**：采样周期数

| 定义 | 描述 |
|---|---|

| | |
|---|---|
| n | 每次采样 n 次，n 不为 0 |

**per_sample_clk**:采样时钟

| 定义 | 描述 |
|---|---|
| n | 每次采样 n+1 个上升沿， |

**tk_wait_time:**TK 等待时间

| 定义 | 描述 |
|---|---|
| n | LED 和触摸按键复用时： LED 扫描结束后等待 n 个 tk_clk 才开始下一轮按键扫描。 |

**tk_sample_time**:每次扫描时间

| 定义 | 描述 |
|---|---|
| n | 每个按键的扫描时间为 n 个 tk_hclk 周期。 |

**mmstbthr**:采样稳定阈值

| 定义 | 描述 |
|---|---|
| n | 当前采样稳定阈值：对当前 N 个采样值的最大最小值作差，当差值大于 n 时，认为当前采样不稳定，丢弃采样值，不做滤波计算及触摸事件判断 |

**tk_buf**:TK 缓冲区

**scan_timeout_det_en:**扫描超时检测

| 定义 | 描述 |
|---|---|
| DISABLE | 不使能 |
| ENABLE | 使能 |

**scan_done_ie**:扫描完成中断使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

**sample_ov_ie**:采样溢出中断使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

**scan_timeout_ie**:扫描超时中断使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

**key_trigger_ie**:按键触发中断使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

**auto_scan_en**:自动扫描使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

**rc_inv_en**:RC 信号取反使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |

| | |
|---|---|
| ENABLE | 使能 |

**precyc_num**:模拟 RC 信号稳定周期数

| 定义 | 描述 |
|---|---|
| N | 经过(N + 1)* tkclk 稳定后开始采样，N 最大为 31，电荷转移模式时，N 最大为 7 |

## 14.2 函数 ll_tk_deinit

| | |
|---|---|
| 函数名 | void ll_tk_deinit(TK_TypeDef *p_tk) |
| 功能 | 释放 tk |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

## 14.3 函数 ll_tk_anafrq_rccis_set

| | |
|---|---|
| 函数名 | void ll_tk_anafrq_rccis_set(TK_TypeDef *p_tk, u32 key_num, u8 rccis) |
| 功能 | RC 充电器电流值设置 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | key_num：按键编号 |
| 参数 3 | Rccis：电流值，最大取 7 |
| 返回值 | 无 |

## 14.4 函数 ll_tk_anafrq_rcftune_set

| | |
|---|---|
| 函数名 | void ll_tk_anafrq_rcftune_set(TK_TypeDef *p_tk, u32 key_num, u8 rcftune) |
| 功能 | RC 充电器频率微调设置 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | key_num：按键编号 |
| 参数 3 | rcftune：RC 充电器频率微调值，最大取 3 |

| | |
|---|---|
| 返回值 | 无 |

### 14.5 函数 ll_tk_anafrq_rccis_get

| | |
|---|---|
| 函数名 | bool ll_tk_anafrq_rccis_get(TK_TypeDef *p_tk, u32 key_num, u8 *p_rccis) |
| 功能 | RC 充电器电流值获取 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | key_num：按键编号 |
| 参数 3 | p_rccis：电流值存放地址 |
| 返回值 | false：操作失败　true：操作成功 |

### 14.6 函数 ll_tk_anafrq_rcftune_get

| | |
|---|---|
| 函数名 | bool ll_tk_anafrq_rcftune_get(TK_TypeDef *p_tk, u32 key_num, u8 *p_rcftune) |
| 功能 | RC 充电器频率微调值获取 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | key_num：按键编号 |
| 参数 3 | p_rcftune：RC 充电器频率微调值存放地址 |
| 返回值 | false：操作失败　true：操作成功 |

### 14.7 函数 ll_tk_offset_set

| | |
|---|---|
| 函数名 | void ll_tk_offset_set(TK_TypeDef *p_tk, u32 key_num, u32 offset) |
| 功能 | 阈值设置 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | bit_map：按键编号 |
| 参数 3 | Offset：阈值 |
| 返回值 | 无 |

**Offset**：阈值

| 定义 | 描述 |
| --- | --- |
| N | 最大 4096，百分比模式：N / 4096 |

### 14.8 函数 ll_tk_led_mulitplex

| 函数名 | void ll_tk_led_mulitplex(TK_TypeDef *p_tk, u16 tk_wait_cnt, u16 tk_period_cnt) |
| --- | --- |
| 功能 | Tk 和 LED 复用配置 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | tk_wait_cnt：TK 等待时间　　LED 扫描后等待 N * TKCLK 才开始 tk 扫描 |
| 参数 3 | tk_period_cnt：TK 扫描时间　　每个按键扫描时间为 N * TKCLK |
| 返回值 | 无 |

### 14.9 函数 ll_tk_key_enable

| 函数名 | void ll_tk_key_enable(TK_TypeDef *p_tk, u32 key_bit_map) |
| --- | --- |
| 功能 | TK 通道使能 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | key_bit_map：通道选择：每个位代表一个通道 |
| 参数 3 | 无 |
| 返回值 | 无 |

### 14.10 函数 ll_tk_advanced_config

| 函数名 | void ll_tk_advanced_config(TK_TypeDef *p_tk, TYPE_LL_TK_ADVANCED_CFG *p_cfg) |
| --- | --- |
| 功能 | TK 通道高级配置 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | p_cfg：指向包含了指定外设的配置信息的结构体 |
| 参数 3 | 无 |
| 返回值 | 无 |

```
typedef struct __ll_tk_advanced_cfg {
    u8    key_debounce_level                : 3;
    u8    input_rc_debounce_level           : 4;
    u8    baseline_stable_times             : 3;
    u8    baseline_update_speed;
    u16   base_val_debounce_threshold       : 12;
u16   lt_base_val_debounce_threshold    : 12;
u8    base_val_debounce_level           : 5;
    u8    chswup_speed                      : 1;
    u8    chswup                            : 2;
    FunctionalState       noise_detect_en;
    u16   noise_threshold                   : 12;
    u16   lt_noise_threshold                : 12;
    u32   threshold_sel_bit_field           : 20;
} TYPE_LL_TK_ADVANCED_CFG;
```

**key_debounce_level**：按键输出去抖（次数）

| 定义 | 描述 |
| --- | --- |
| N | 次数：N +1 |

**input_rc_debounce_level**：按键输入去抖（次数）

| 定义 | 描述 |
| --- | --- |
| N | 次数：N +1 |

**baseline_stable_times**：基准值稳定水平(次数)

| 定义 | 描述 |
| --- | --- |
| N | 保持 N 次不变后认为基准值是稳定的 |

**baseline_update_speed**：控制标准值更新速度，数值越大，更新速度越慢,可选择 0 ~ 255

**base_val_debounce_threshold**：基准值超过阈值

| 定义 | 描述 |
| --- | --- |
| N | 按百分比计算，当采样值与基准值的差值超过 BASEVAL*N/4096 时，对基准值更新做防抖处理。 |

**lt_base_val_debounce_threshold**：基准值超过阈值（复用 led 模式）

| 定义 | 描述 |
| --- | --- |
| N | 按百分比计算，当采样值与基准值的差值超过 BASEVAL*N/4096 时，对基准值更新做防抖处理。 |

**base_val_debounce_level**：基准值减持时间

| 定义 | 描述 |
| --- | --- |
| N | 基准值去抖次数：当采样值超过基准值的去抖阈值时，需要满足 N 次去抖后才更新基准值。 |

**chswup_speed**：电荷转移模式，加速模式

| 定义 | 描述 |
| --- | --- |
| 0 | 加速 2 倍 (FSDIV/2) 即 fsdiv:电荷转移模式充电开关分频比 |
| 1 | 加速 4 倍 (FSDIV/4) |

**Chswup**：电荷转移模式，翻转 n 次后加速

| 定义 | 描述 |
| --- | --- |
| 0 | 不翻转 |
| 1 | 翻转 256 次后加速 |
| 2 | 翻转 512 次后加速 |
| 3 | 翻转 1024 次后加速 |

**noise_detect_en**：噪声检测使能

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

**noise_threshold**：噪声阈值

| 定义 | 描述 |
| --- | --- |
| DISABLE | 不使能 |
| ENABLE | 使能 |

**threshold_sel_bit_field**：按键基准值噪声去抖阈值

| 定义 | 描述 |
| --- | --- |
| 0 | 选择 base_val_debounce_threshold 即基准值超过阈值 |
| 1 | 选择 lt_base_val_debounce_threshold 即基准值超过阈值（复用 led 模式） |

### 14.11 函数 ll_tk_scan_status_get

| 函数名 | u32 ll_tk_scan_status_get(TK_TypeDef *p_tk) |
| --- | --- |
| 功能 | 获取按键扫描状态 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 按键扫描状态 |

### 14.12 函数 ll_tk_key_ready_state_get

| 函数名 | bool ll_tk_key_ready_state_get(TK_TypeDef *p_tk, u32 key_num) |
| --- | --- |
| 功能 | 获取按键准备状态（用于测试） |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | key_num：按键编号 |
| 返回值 | False or true |

### 14.13 函数 ll_tk_key_state_get

| | |
|---|---|
| 函数名 | bool ll_tk_key_state_get(TK_TypeDef *p_tk, u32 key_num) |
| 功能 | 获取按键状态 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | key_num：按键编号 |
| 返回值 | False or true |

### 14.14 函数 ll_tk_key_map_state_get

| | |
|---|---|
| 函数名 | u32 ll_tk_key_map_state_get(TK_TypeDef *p_tk) |
| 功能 | 获取触摸按键映射状态 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 按键映射状态 |

### 14.15 函数 ll_tk_filtval_get

| | |
|---|---|
| 函数名 | u16 ll_tk_filtval_get(TK_TypeDef *p_tk, u32 num) |
| 功能 | 获取按键采样值滤波模式 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | key_num：按键编号 |
| 返回值 | 采样滤波模式 |

### 14.16 函数 ll_tk_baseval_get

| | |
|---|---|
| 函数名 | u16 ll_tk_filtval_get(TK_TypeDef *p_tk, u32 num) |
| 功能 | 获取按键基准值滤波模式 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | key_num：按键编号 |
| 返回值 | 按键基准滤波模式 |

### 14.17 函数 ll_tk_scdn_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_tk_scdn_interrupt_enable(TK_TypeDef *p_tk) |
| 功能 | 使能单个按键扫描完成中断 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.18 函数 ll_tk_scdn_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_tk_scdn_interrupt_disable(TK_TypeDef *p_tk) |
| 功能 | 禁用单个按键扫描完成中断 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.19 函数 ll_tk_spovf_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_tk_spovf_interrupt_enable(TK_TypeDef *p_tk) |
| 功能 | 使能按键采样值溢出中断 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.20 函数 ll_tk_spovf_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_tk_spovf_interrupt_disable(TK_TypeDef *p_tk) |
| 功能 | 禁用按键采样值溢出中断 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.21 函数 ll_tk_scnovt_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_tk_scnovt_interrupt_enable(TK_TypeDef *p_tk) |
| 功能 | 使能按键采样超时中断 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.22 函数 ll_tk_scnovt_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_tk_scnovt_interrupt_disable(TK_TypeDef *p_tk) |
| 功能 | 禁用按键采样超时中断 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.23 函数 ll_tk_keyirq_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_tk_keyirq_interrupt_enable(TK_TypeDef *p_tk) |
| 功能 | 使能按键触发中断 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.24 函数 ll_tk_keyirq_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_tk_keyirq_interrupt_disable(TK_TypeDef *p_tk) |
| 功能 | 禁用按键触发中断 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.25 函数 ll_tk_samp_timeout_det_enable

| | |
|---|---|
| 函数名 | void ll_tk_samp_timeout_det_enable(TK_TypeDef *p_tk) |
| 功能 | 使能按键采样超时检测 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.26 函数 ll_tk_samp_timeout_det_disable

| | |
|---|---|
| 函数名 | void ll_tk_samp_timeout_det_disable(TK_TypeDef *p_tk) |
| 功能 | 禁用按键采样超时检测 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.27 函数 ll_tk_rc_inv_enable

| | |
|---|---|
| 函数名 | void ll_tk_rc_inv_enable(TK_TypeDef *p_tk) |
| 功能 | 使能按键 RC 信号取反 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.28 函数 ll_tk_rc_inv_disable

| | |
|---|---|
| 函数名 | void ll_tk_rc_inv_disable(TK_TypeDef *p_tk) |
| 功能 | 禁用按键 RC 信号取反 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.29 函数 ll_tk_autoscan_enable

| | |
|---|---|
| 函数名 | void ll_tk_autoscan_enable(TK_TypeDef *p_tk) |
| 功能 | 使能按键自动取反 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.30 函数 ll_tk_autoscan_disable

| | |
|---|---|
| 函数名 | void ll_tk_autoscan_enable(TK_TypeDef *p_tk) |
| 功能 | 禁用按键自动取反 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.31 函数 ll_tk_fixed_scan_period_enable

| | |
|---|---|
| 函数名 | void ll_tk_fixed_scan_period_enable(TK_TypeDef *p_tk) |
| 功能 | 使能按键周期扫描 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.32 函数 ll_tk_fixed_scan_period_disable

| | |
|---|---|
| 函数名 | void ll_tk_fixed_scan_period_disable(TK_TypeDef *p_tk) |
| 功能 | 禁用按键周期扫描 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.33 函数 ll_tk_fixed_scan_start_enable

| | |
|---|---|
| 函数名 | void ll_tk_fixed_scan_start_enable(TK_TypeDef *p_tk) |
| 功能 | 按键扫描开始 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.34 函数 ll_tk_fixed_scan_start_disable

| | |
|---|---|
| 函数名 | void ll_tk_fixed_scan_start_disable(TK_TypeDef *p_tk) |
| 功能 | 按键扫描停止 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.35 函数 ll_tk_enable

| | |
|---|---|
| 函数名 | void ll_tk_enable(TK_TypeDef *p_tk) |
| 功能 | 使能 TK 模块 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.36 函数 ll_tk_disable

| | |
|---|---|
| 函数名 | void ll_tk_disable(TK_TypeDef *p_tk) |
| 功能 | 禁用 TK 模块 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.37 函数 ll_tk_ana_test_out_enable

| | |
|---|---|
| 函数名 | void ll_tk_ana_test_out_enable(TK_TypeDef *p_tk) |
| 功能 | 使能 TK 模拟 RC 信号输出 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.38 函数 ll_tk_ana_test_out_disable

| | |
|---|---|
| 函数名 | void ll_tk_ana_test_out_enable(TK_TypeDef *p_tk) |
| 功能 | 禁用 TK 模拟 RC 信号输出 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.39 函数 ll_tk_ana_rcen_ccen_hwctl_enable

| | |
|---|---|
| 函数名 | void ll_tk_ana_rcen_ccen_hwctl_enable(TK_TypeDef *p_tk) |
| 功能 | 使能 TK 控制硬件模拟张弛振荡器模式和模拟电荷转移模式 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.40 函数 ll_tk_ana_rcen_ccen_hwctl_disable

| | |
|---|---|
| 函数名 | void ll_tk_ana_rcen_ccen_hwctl_disable(TK_TypeDef *p_tk) |
| 功能 | 禁用 TK 控制硬件模拟张弛振荡器模式和模拟电荷转移模式 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.41 函数 ll_tk_ana_rc_enable

| | |
|---|---|
| 函数名 | void ll_tk_ana_rc_enable(TK_TypeDef *p_tk) |
| 功能 | 使能 TK 控制硬件模拟张弛振荡器模式 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.42 函数 ll_tk_ana_rc_disable

| | |
|---|---|
| 函数名 | void ll_tk_ana_rc_disable(TK_TypeDef *p_tk) |
| 功能 | 禁用 TK 控制硬件模拟张弛振荡器模式 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.43 函数 ll_tk_ana_cc_enable

| | |
|---|---|
| 函数名 | void ll_tk_ana_rc_disable(TK_TypeDef *p_tk) |
| 功能 | 使能 TK 控制硬件模拟电荷转移模式 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 11.44 函数 ll_tk_ana_cc_disable

| | |
|---|---|
| 函数名 | void ll_tk_ana_cc_disable(TK_TypeDef *p_tk) |
| 功能 | 禁用 TK 控制硬件模拟电荷转移模式 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.45 函数 ll_tk_ana_currefsel_enable

| | |
|---|---|
| 函数名 | void ll_tk_ana_currefsel_enable(TK_TypeDef *p_tk) |
| 功能 | 使能 TK 模拟 LD0 电流参考模式 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.46 函数 ll_tk_ana_currefsel_disable

| | |
|---|---|
| 函数名 | void ll_tk_ana_currefsel_disable(TK_TypeDef *p_tk) |
| 功能 | 禁用 TK 模拟 LD0 电流参考模式 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.47 函数 ll_tk_ana_vref_out_to_atsout_enable

| | |
|---|---|
| 函数名 | void ll_tk_ana_vref_out_to_atsout_enable(TK_TypeDef *p_tk) |
| 功能 | 使能 TK 模拟无盖 VREF 输出到 ATSOUT |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.48 函数 ll_tk_ana_vref_out_to_atsout_disable

| | |
|---|---|
| 函数名 | void ll_tk_ana_vref_out_to_atsout_disable(TK_TypeDef *p_tk) |
| 功能 | 禁用 TK 模拟无盖 VREF 输出到 ATSOUT |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.49 函数 ll_tk_ana_vddtk_out_to_atsout_enable

| | |
|---|---|
| 函数名 | void ll_tk_ana_vddtk_out_to_atsout_enable(TK_TypeDef *p_tk) |
| 功能 | 使能 TK 模拟无盖 VDDTK 输出到 ATSOUT |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.50 函数 ll_tk_ana_vddtk_out_to_atsout_disable

| | |
|---|---|
| 函数名 | void ll_tk_ana_vddtk_out_to_atsout_disable(TK_TypeDef *p_tk) |
| 功能 | 禁用 TK 模拟无盖 VDDTK 输出到 ATSOUT |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.51 函数 ll_tk_ana_rccap_enable

| | |
|---|---|
| 函数名 | void ll_tk_ana_rccap_enable(TK_TypeDef *p_tk) |
| 功能 | 使能 TK 模拟 RC 模式内盖 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.52 函数 ll_tk_ana_rccap_disable

| | |
|---|---|
| 函数名 | void ll_tk_ana_rccap_enable(TK_TypeDef *p_tk) |
| 功能 | 禁用 TK 模拟 RC 模式内盖 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.53 函数 ll_tk_ana_ccfb_enable

| | |
|---|---|
| 函数名 | void ll_tk_ana_ccfb_enable(TK_TypeDef *p_tk) |
| 功能 | 使能 TK 模拟充电帽模式局部反馈 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.54 函数 ll_tk_ana_ccfb_disable

| | |
|---|---|
| 函数名 | void ll_tk_ana_ccfb_disable(TK_TypeDef *p_tk) |
| 功能 | 禁用 TK 模拟充电帽模式局部反馈 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.55 函数 ll_tk_ana_ldo_enable

| | |
|---|---|
| 函数名 | void ll_tk_ana_ldo_enable(TK_TypeDef *p_tk) |
| 功能 | 使能 TK 模拟 LDO 信号 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.56 函数 ll_tk_ana_ldo_disable

| | |
|---|---|
| 函数名 | void ll_tk_ana_ldo_enable(TK_TypeDef *p_tk) |
| 功能 | 禁用 TK 模拟 LDO 信号 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.57 函数 ll_tk_noise_detect_enable

| | |
|---|---|
| 函数名 | void ll_tk_noise_detect_enable(TK_TypeDef *p_tk) |
| 功能 | 使能 TK 噪声检测 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.58 函数 ll_tk_noise_detect_disable

| | |
|---|---|
| 函数名 | void ll_tk_noise_detect_enable(TK_TypeDef *p_tk) |
| 功能 | 禁用 TK 噪声检测 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 14.59 函数 ll_tk_keypnd_map_pending_clr

| | |
|---|---|
| 函数名 | void ll_tk_keypnd_map_pending_clr(TK_TypeDef *p_tk, u32 num_bit_map) |
| 功能 | 根据按键编号索引的位图清除按键触发请求中断标志 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | num_bit_map:键号索引位图（key0~key19 对应于 bit0~bit19，置 1 清除） |
| 返回值 | 无 |

### 14.60 函数 ll_tk_scovpnd_map_pending_clr

| | |
|---|---|
| 函数名 | void ll_tk_scovpnd_map_pending_clr(TK_TypeDef *p_tk, u32 num_bit_map) |
| 功能 | 根据按键编号索引的位图清除扫描超时中断标志 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | num_bit_map:键号索引位图（key0~key19 对应于 bit0~bit19，置 1 清除） |
| 返回值 | 无 |

### 14.61 函数 ll_tk_scdopnd_map_pending_clr

| | |
|---|---|
| 函数名 | void ll_tk_scdopnd_map_pending_clr(TK_TypeDef *p_tk, u32 num_bit_map) |
| 功能 | 根据按键编号索引的位图清除扫描超时中断标志 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | num_bit_map:键号索引位图（key0~key19 对应于 bit0~bit19，置 1 清除） |
| 返回值 | 无 |

### 14.62 函数 ll_tk_intpnd_map_pending_clr

| | |
|---|---|
| 函数名 | void ll_tk_intpnd_map_pending_clr(TK_TypeDef *p_tk, u32 int_bit_map) |
| 功能 | 根据键编号索引的位图清除扫描完成中断标志 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | num_bit_map:键号索引位图（key0~key19 对应于 bit0~bit19，置 1 清除） |
| 返回值 | 无 |

### 14.63 函数 ll_tk_rcftune1_set

| | |
|---|---|
| 函数名 | void ll_tk_rcftune1_set(TK_TypeDef *p_tk, u8 rcftune1) |
| 功能 | rcftune1 设置 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | rcftune1:　0 or 1 |
| 返回值 | 无 |

### 14.64 函数 ll_tk_rcftune0_set

| | |
|---|---|
| 函数名 | void ll_tk_rcftune0_set(TK_TypeDef *p_tk, u8 rcftune0) |
| 功能 | Rcftune0 设置 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | Rcftune0:　0 or 1 |
| 返回值 | 无 |

### 14.65 函数 ll_tk_rcftune_set

| | |
|---|---|
| 函数名 | void ll_tk_rcftune_set(TK_TypeDef *p_tk, u8 rcftune) |
| 功能 | Rcftune0　rcftune1 设置 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | rcftune:　最大取 3 |
| 返回值 | 无 |

### 14.66 函数 ll_tk_rcbank1_set

| | |
|---|---|
| 函数名 | void ll_tk_rcbank1_set(TK_TypeDef *p_tk, u8 rcbank1) |
| 功能 | rcbank1 设置 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | rcbank1:　0 or 1 |
| 返回值 | 无 |

### 14.67 函数 ll_tk_rcbank0_set

| | |
|---|---|
| 函数名 | void ll_tk_rcbank0_set(TK_TypeDef *p_tk, u8 rcbank0) |
| 功能 | Rcbank0 设置 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | Rcbank0:　0 or 1 |
| 返回值 | 无 |

### 14.68 函数 ll_tk_rcbank_set

| | |
|---|---|
| 函数名 | void ll_tk_rcbank_set(TK_TypeDef *p_tk, u8 rcbank) |
| 功能 | Rcbank　rcbank1 设置 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | Rcbank:　最大取 3 |

| | |
|---|---|
| 返回值 | 无 |

### 14.69 函数 ll_tk_ccvrs1_set

| | |
|---|---|
| 函数名 | void ll_tk_ccvrs1_set(TK_TypeDef *p_tk, u8 ccvrs1) |
| 功能 | ccvrs1 设置 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | ccvrs1:  0 or 1 |
| 返回值 | 无 |

### 14.70 函数 ll_tk_ccvrs0_set

| | |
|---|---|
| 函数名 | void ll_tk_ccvrs0_set(TK_TypeDef *p_tk, u8 ccvrs0) |
| 功能 | Ccvrs0 设置 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | Ccvrs0:  0 or 1 |
| 返回值 | 无 |

### 14.71 函数 ll_tk_ccvrs_set

| | |
|---|---|
| 函数名 | void ll_tk_ccvrs_set(TK_TypeDef *p_tk, u8 ccvrs) |
| 功能 | Ccvrs0  ccvrs1 设置(比较器电压选择) |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | Ccvrs:  最大取 3 |
| 返回值 | 无 |

| 定义 | 描述 |
|---|---|
| b’00 | (4/8)*vddtk |
| b’01 | (3/8)*vddtk |
| b’10 | (5/8)*vddtk |

| | |
|---|---|
| b'11 | (7/8)*vddtk |

### 14.72 函数 ll_tk_ldo_v_sel

| | |
|---|---|
| 函数名 | void ll_tk_ldo_v_sel(TK_TypeDef *p_tk, u8 ldo_v_sel) |
| 功能 | Ldo 电压选择 |
| 参数 1 | p_tk：tk 寄存器基地址 |
| 参数 2 | ldo_v_sel: 0 ：1.5v 1:1.6v |
| 返回值 | 无 |

### 14.73 模块相关宏定义

| 定义 | 描述 |
|---|---|
| LL_TK_SCDN_INTERRUPT_GET(p_tk) | 单触摸键扫描完成中断获取 |
| LL_TK_SPOVF_INTERRUPT_GET(p_tk) | 采样值溢出中断获取 |
| LL_TK_SCNOVT_INTERRUPT_GET(p_tk) | 采样时间超时中断获取 |
| LL_TK_KEYIRQ_INTERRUPT_GET(p_tk) | 触摸键触发中断获取 |
| LL_TK_SAMP_TIMEOUT_DET_EN_GET(p_tk) | 采样时间超时使能获取 |
| LL_TK_RC_INV_EN_GET(p_tk) | RC 信号取反使能获取 |
| LL_TK_AUTOSCAN_EN_GET(p_tk) | 自动扫描使能获取 |
| LL_TK_FIXED_SCAN_PERIOD_EN_GET(p_tk) | 固定扫描周期使能获取 |
| LL_TK_FIXED_SCAN_START_EN_GET(p_tk) | TK 扫描启动使能获取 |
| LL_TK_EN_GET(p_tk) | TK 模块使能获取 |
| LL_TK_ANA_TEST_OUT_EN_GET(p_tk) | 模拟 RC 信号输出使能获取 |
| LL_TK_ANA_RCEN_CCEN_HWCTL_EN_GET(p_tk) | TK 硬件控制 TKEY_RCEN 和 TKEY_CCEN 使能获取 |
| LL_TK_ANA_RC_EN_GET(p_tk) | 模拟张弛振荡器模式使能获取 |
| LL_TK_ANA_CC_EN_GET(p_tk) | 模拟电荷转移模式使能获取 |
| LL_TK_ANA_CURREFSEL_EN_GET(p_tk) | 模拟 LD0 电流参考模式选择使能获取 |

| | |
|---|---|
| LL_TK_ANA_VREF_OUT_TO_ATSOUT_EN_GET(p_tk) | 模拟无盖 VREF 输出到 ATSOUT 使能获取 |
| LL_TK_ANA_VDDTK_OUT_TO_ATSOUT_EN_GET(p_tk) | 模拟无盖 VDDTK 输出到 ATSOUT 使能获取 |
| LL_TK_ANA_RCCAP_EN_GET(p_tk) | TK 模拟 RC 模式内盖使能获取 |
| LL_TK_ANA_CCFB_EN_GET(p_tk) | TK 模拟充电盖模式，本地反馈使取获取 |
| LL_TK_ANA_LDO_EN_GET(p_tk) | 模拟 LD0 信号使能获取 |
| LL_TK_NOISE_DETECT_EN_GET(p_tk) | TK 噪值检测使能获取 |
| LL_TK_INTPND_SCDN_PENDING_GET(p_tk) | 单触摸键扫描完成状态获取 |
| LL_TK_INTPND_SCDN_PENDING_CLR(p_tk) | 单触摸键扫描完成状态清除 |
| LL_TK_INTPND_SCANOVT_PENDING_GET(p_tk) | 单触摸键扫描超时状态获取 |
| LL_TK_INTPND_SCANOVT_PENDING_CLR(p_tk) | 单触摸键扫描超时状态清除 |
| LL_TK_INTPND_SPOVF_PENDING_GET(p_tk) | 采样累积值溢出状态获取 |
| LL_TK_INTPND_SPOVF_PENDING_CLR(p_tk) | 采样累积值溢出状态清除 |
| LL_TK_INTPND_KEYIRQ_PENDING_GET(p_tk) | 触发请求状态获取 |
| LL_TK_INTPND_KEYIRQ_PENDING_CLR(p_tk) | 触发请求状态清除 |
| LL_TK_KEYPND_MAP_PENDING_GET(p_tk) | 按键触发状态获取，bit0~bit19 对应于 key0~key19 |
| LL_TK_KEYPND_MAP_PENDING_CLR(p_tk,num_bit_map) | 按键触发状态清除， bit0~bit19 对应于 key0~key19, set 1 clr |
| LL_TK_SCOVPND_MAP_PENDING_GET(p_tk) | 扫描完成状态获取，bit0~bit19 对应于 key0~key19 |
| LL_TK_SCOVPND_MAP_PENDING_CLR(p_tk,num_bit_map) | 扫描完成状态清除，bit0~bit19 对应于 key0~key19 |
| LL_TK_BASE_NOISE_PENDING_GET(p_tk) | 基准噪声检测状态获取 |
| LL_TK_SAMP_NOISE_PENDING_GET(p_tk) | 采样噪声检测状态获取 |

# 15 通用异步收发起 uart

通用异步收发器(UART)提供了一种灵活的方法与使用工业标准 NRZ 异步串行数据格式的外部设备之间进行全双工数据交换。UART 利用分数波特率发生器提供宽度范围的波特率选择。使用多缓冲器配置的 DMA 方式，可以实现高速数据通信。

## 15.1 函数 ll_uart_init

| | |
|---|---|
| 函数名 | void ll_uart_init(UART2_TypeDef *p_uart, TYPE_LL_UART_INIT *p_init) |
| 功能 | 初始化串口 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | p_init：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_uart_init {
    u32                                 baudrate;
    TYPE_ENUM_LL_UART_BIT_WIDTH_SEL     bit_width_sel;
    TYPE_ENUM_LL_UART_PARITY_SEL        parity;
    TYPE_ENUM_LL_UART_STOP_BIT_SEL      stop_bit_sel;
    TYPE_ENUM_LL_UART_WORK_MODE         work_mode;
} TYPE_LL_UART_INIT;
```

**Baudrate**：波特率

**bit_width_sel**：数据长度

| 定义 | 描述 |
|---|---|
| LL_UART_WORD_LENGTH_8B | 8 位 |
| LL_UART_WORD_LENGTH_8B | 9 位 |

**Parity**：奇偶校验

| 定义 | 描述 |
|---|---|
| LL_UART_PARITY_NO | 无校验 |

| | |
|---|---|
| LL_UART_PARITY_ODD | 奇校验 |
| LL_UART_PARITY_EVEN | 偶校验 |
| LL_UART_RESVERS | 保留 |

**stop_bit_sel**：停止位位数选择

| 定义 | 描述 |
|---|---|
| LL_UART_STOP_1B | 1 位 |
| LL_UART_STOP_2B | 2 位 |

**work_mode**：工作模式

| 定义 | 描述 |
|---|---|
| LL_UART_WORK_MODE_FULL_DUPLEX | 双工 |
| LL_UART_WORK_MODE_SINGLE_SND | 单工发送 |
| LL_UART_WORK_MODE_SINGLE_RCV | 单工接收 |
| LL_UART_WORK_MODE_SINGLE_AUTO | 单线自动 |

## 15.2 函数 ll_uart_deinit

| | |
|---|---|
| 函数名 | void ll_uart_deinit(UART2_TypeDef *p_uart) |
| 功能 | 释放串口 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | p_init：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

## 15.3 函数 ll_uart_ie_config

| | |
|---|---|
| 函数名 | void ll_uart_ie_config(UART2_TypeDef *p_uart, TYPE_LL_UART_IE_CFG *p_cfg) |
| 功能 | 串口中断配置 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |

| | |
|---|---|
| 参数 2 | p_cfg：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_uart_ie_cfg {
    FunctionalState uart_tx_ie;
    FunctionalState uart_rx_ie;
} TYPE_LL_UART_IE_CFG;
```

**uart_tx_ie**：发送中断

| 定义 | 描述 |
|---|---|
| ENABLE | 使能 |
| DISABLE | 禁用 |

**uart_rx_ie**：接收中断

| 定义 | 描述 |
|---|---|
| ENABLE | 使能 |
| DISABLE | 禁用 |

## 15.4 函数 ll_uart_rs485_config

| | |
|---|---|
| 函数名 | void ll_uart_rs485_config( UART2_TypeDef *p_uart, TYPE_LL_UART_RS485_CFG *p_cfg ) |
| 功能 | 485 配置 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | p_cfg：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_uart_rs485_cfg{
    TYPE_ENUM_LL_UART_RS485_MODE  rs485_mode;
    TYPE_ENUM_LL_UART_RS485_RE_DE_POL  re_pol;
    TYPE_ENUM_LL_UART_RS485_RE_DE_POL  de_pol;
```

```
    u16 de_dat;
    u16 de_at;
    u16 re2de_t;
    u16 de2re_t;
    FunctionalState   re_en;
    FunctionalState   de_en;
    FunctionalState   rs485_en;
} TYPE_LL_UART_RS485_CFG;
```

**rs485_mode**：工作模式

| 定义 | 描述 |
| --- | --- |
| LL_UART_RS485_MODE_MANUAL | 用户软件切换 |
| LL_UART_RS485_MODE_AUTO | 硬件自动切换 |

**re_pol**：re 极性

| 定义 | 描述 |
| --- | --- |
| LL_UART_RS485_RE_DE_POL_H | 高电平有效 |
| LL_UART_RS485_RE_DE_POL_L | 低电平有效 |

**de_pol**：de 极性

| 定义 | 描述 |
| --- | --- |
| LL_UART_RS485_RE_DE_POL_H | 高电平有效 |
| LL_UART_RS485_RE_DE_POL_L | 低电平有效 |

**de_dat**：时间间隔配置（停止位到 de 下降沿）

| 定义 | 描述 |
| --- | --- |
| n | n * uart_clk |

**de_at**：时间间隔配置（de 上升沿到开始位）

| 定义 | 描述 |
| --- | --- |

| | |
|---|---|
| n | n * uart_clk |

**re2de_t**：时间间隔配置（re 下降沿到 de 上升沿）

| 定义 | 描述 |
|---|---|
| n | n * uart_clk |

**de2re_t**：时间间隔配置（de 下降沿到 re 上升沿）

| 定义 | 描述 |
|---|---|
| n | n * uart_clk |

**re_en**：re 使能

| 定义 | 描述 |
|---|---|
| ENABLE | 使能 |
| DISABLE | 禁用 |

**de_en**：de 使能

| 定义 | 描述 |
|---|---|
| ENABLE | 使能 |
| DISABLE | 禁用 |

**rs485_en**：485 使能

| 定义 | 描述 |
|---|---|
| ENABLE | 使能 |
| DISABLE | 禁用 |

## 15.5 函数 ll_uart_dma_config

| | |
|---|---|
| 函数名 | void ll_uart_dma_config(UART2_TypeDef *p_uart,TYPE_LL_UART_DMA_CFG *p_cfg) |
| 功能 | Uart dma 配置 |

| | |
|---|---|
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | p_cfg：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_uart_dma_cfg{
    u32             dma_tx_addr;
    u32             dma_rx_addr;
    u16             dma_want_tx_len;
    u16             dma_want_rx_len;
    FunctionalState dma_rx_perr_ie;
    FunctionalState dma_tx_ie;
    FunctionalState dma_rx_ie;
    FunctionalState dma_tx_en;
    FunctionalState dma_rx_en;
} TYPE_LL_UART_DMA_CFG;
```

**dma_tx_addr**：发送地址

**dma_rx_addr**：接收地址

**dma_want_tx_len**：发送数据长度

**dma_want_rx_len**：接收数据长度

**dma_rx_perr_ie**：奇偶校验出错中断使能

| 定义 | 描述 |
|---|---|
| ENABLE | 使能 |
| DISABLE | 禁用 |

**dma_tx_ie**：发送中断使能

| 定义 | 描述 |
|---|---|

| | |
|---|---|
| ENABLE | 使能 |
| DISABLE | 禁用 |

**dma_rx_ie**：接收中断使能

| 定义 | 描述 |
|---|---|
| ENABLE | 使能 |
| DISABLE | 禁用 |

**dma_tx_en**：发送使能

| 定义 | 描述 |
|---|---|
| ENABLE | 使能 |
| DISABLE | 禁用 |

**dma_rx_en**：接收使能

| 定义 | 描述 |
|---|---|
| ENABLE | 使能 |
| DISABLE | 禁用 |

## 15.6 函数 ll_uart_rx_timeout_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_uart_rx_timeout_interrupt_enable(UART2_TypeDef *p_uart) |
| 功能 | 使能接收超时中断 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

## 15.7 函数 ll_uart_rx_timeout_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_uart_rx_timeout_interrupt_disable(UART2_TypeDef *p_uart) |

| | |
|---|---|
| 功能 | 禁用接收超时中断 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.8 函数 ll_uart_ferr_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_uart_ferr_interrupt_enable(UART2_TypeDef *p_uart) |
| 功能 | 使能帧错误中断 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.9 函数 ll_uart_ferr_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_uart_ferr_interrupt_disable(UART2_TypeDef *p_uart) |
| 功能 | 禁用帧错误中断 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.10 函数 ll_uart_tx_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_uart_tx_interrupt_enable(UART2_TypeDef *p_uart) |
| 功能 | 使能发送中断 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.11 函数 ll_uart_tx_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_uart_tx_interrupt_disable(UART2_TypeDef *p_uart) |

| | |
|---|---|
| 功能 | 禁用发送中断 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.12 函数 ll_uart_rx_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_uart_rx_interrupt_enable(UART2_TypeDef *p_uart) |
| 功能 | 使能接收中断 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.13 函数 ll_uart_rx_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_uart_rx_interrupt_disable(UART2_TypeDef *p_uart) |
| 功能 | 禁用接收中断 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.14 函数 ll_uart1_dma_rx_perr_interrupt_enable

| | |
|---|---|
| 函数名 | void ll_uart1_dma_rx_perr_interrupt_enable(void) |
| 功能 | 使能串口 1 dma 接收帧错误中断 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.15 函数 ll_uart1_dma_rx_perr_interrupt_disable

| | |
|---|---|
| 函数名 | void ll_uart1_dma_rx_perr_interrupt_disable(void) |

| 功能 | 禁用串口 1 dma 接收帧错误中断 |
|---|---|
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.16 函数 ll_uart1_dma_rx_interrupt_enable

| 函数名 | void ll_uart1_dma_rx_interrupt_enable(void) |
|---|---|
| 功能 | 使能串口 1 dma 接收完成中断 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.17 函数 ll_uart1_dma_rx_interrupt_disable

| 函数名 | void ll_uart1_dma_rx_interrupt_disable(void) |
|---|---|
| 功能 | 禁用串口 1 dma 接收完成中断 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.18 函数 ll_uart1_dma_tx_interrupt_enable

| 函数名 | void ll_uart1_dma_tx_interrupt_enable(void) |
|---|---|
| 功能 | 使能串口 1 dma 发送完成中断 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.19 函数 ll_uart1_dma_tx_interrupt_disable

| 函数名 | void ll_uart1_dma_tx_interrupt_disable(void) |
|---|---|

| | |
|---|---|
| 功能 | 禁用串口 1 dma 发送完成中断 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.20 函数 ll_uart_tmr_pwm_enable

| | |
|---|---|
| 函数名 | void ll_uart_tmr_pwm_enable(UART2_TypeDef *p_uart) |
| 功能 | 使能 UART 的输出和 TMR 的 PWM 一起运算后输出 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.21 函数 ll_uart_tmr_pwm_disable

| | |
|---|---|
| 函数名 | void ll_uart_tmr_pwm_disable(UART2_TypeDef *p_uart) |
| 功能 | 禁用 UART 的输出和 TMR 的 PWM 一起运算后输出 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.22 函数 ll_uart_rx_timeout_enable

| | |
|---|---|
| 函数名 | void ll_uart_rx_timeout_enable(UART2_TypeDef *p_uart) |
| 功能 | 使能 UART 接收超时中断 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.23 函数 ll_uart_rx_timeout_disable

| | |
|---|---|
| 函数名 | void ll_uart_rx_timeout_disable(UART2_TypeDef *p_uart) |

| | |
|---|---|
| 功能 | 禁用 UART 接收超时中断 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.24 函数 ll_uart_tx_inv_enable

| | |
|---|---|
| 函数名 | void ll_uart_tx_inv_enable(UART2_TypeDef *p_uart) |
| 功能 | 使能 UART 发送信号取反 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.25 函数 ll_uart_tx_inv_disable

| | |
|---|---|
| 函数名 | void ll_uart_tx_inv_disable(UART2_TypeDef *p_uart) |
| 功能 | 禁用 UART 发送信号取反 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.26 函数 ll_uart_rx_inv_enable

| | |
|---|---|
| 函数名 | void ll_uart_rx_inv_enable(UART2_TypeDef *p_uart) |
| 功能 | 使能 UART 接收信号取反 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.27 函数 ll_uart_rx_inv_disable

| | |
|---|---|
| 函数名 | void ll_uart_rx_inv_disable(UART2_TypeDef *p_uart) |

| | |
|---|---|
| 功能 | 禁用 UART 接收信号取反 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.28 函数 ll_uart_odd_enable

| | |
|---|---|
| 函数名 | void ll_uart_odd_enable(UART2_TypeDef *p_uart) |
| 功能 | 设置 UART 奇校验 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.29 函数 ll_uart_odd_disable

| | |
|---|---|
| 函数名 | void ll_uart_odd_disable(UART2_TypeDef *p_uart) |
| 功能 | 设置 UART 偶校验 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.30 函数 ll_uart_parity_enable

| | |
|---|---|
| 函数名 | void ll_uart_parity_enable(UART2_TypeDef *p_uart) |
| 功能 | 使能 UART 奇校验 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.31 函数 ll_uart_parity_disable

| | |
|---|---|
| 函数名 | void ll_uart_parity_disable(UART2_TypeDef *p_uart) |

| | |
|---|---|
| 功能 | 禁用 UART 奇校验 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.32 函数 ll_uart_9bit_enable

| | |
|---|---|
| 函数名 | void ll_uart_9bit_enable(UART2_TypeDef *p_uart) |
| 功能 | 设置 UART 9 位数据传输 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.33 函数 ll_uart_9bit_disable

| | |
|---|---|
| 函数名 | void ll_uart_9bit_enable(UART2_TypeDef *p_uart) |
| 功能 | 设置 UART 8 位数据传输 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.34 函数 ll_uart_enable

| | |
|---|---|
| 函数名 | void ll_uart_enable(UART2_TypeDef *p_uart) |
| 功能 | 使能 UART |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.35 函数 ll_uart_disable

| | |
|---|---|
| 函数名 | void ll_uart_disable(UART2_TypeDef *p_uart) |

| | |
|---|---|
| 功能 | 禁用 UART |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.36 函数 ll_uart1_dma_rx_enable

| | |
|---|---|
| 函数名 | void ll_uart1_dma_rx_enable(void) |
| 功能 | 使能 UART1 dma 接收 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.37 函数 ll_uart1_dma_rx_disable

| | |
|---|---|
| 函数名 | void ll_uart1_dma_rx_disable(void) |
| 功能 | 禁用 UART1 dma 接收 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.38 函数 ll_uart1_dma_tx_enable

| | |
|---|---|
| 函数名 | void ll_uart1_dma_tx_enable(void) |
| 功能 | 使能 UART1 dma 发送 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.39 函数 ll_uart1_dma_tx_disable

| | |
|---|---|
| 函数名 | void ll_uart1_dma_tx_disable(void) |

| | |
|---|---|
| 功能 | 禁用 UART1　dma 发送 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.40 函数 ll_uart1_rs485_re_enable

| | |
|---|---|
| 函数名 | void ll_uart1_rs485_re_enable(void) |
| 功能 | 使能 rs485 re |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.41 函数 ll_uart1_rs485_re_disable

| | |
|---|---|
| 函数名 | void ll_uart1_rs485_re_disable(void) |
| 功能 | 禁用 rs485 re |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.42 函数 ll_uart1_rs485_de_enable

| | |
|---|---|
| 函数名 | void ll_uart1_rs485_de_enable(void) |
| 功能 | 使能 rs485 de |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.43 函数 ll_uart1_rs485_de_disable

| 函数名 | void ll_uart1_rs485_de_disable(void) |
|---|---|
| 功能 | 禁用 rs485 de |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.44 函数 ll_uart1_rs485_enable

| 函数名 | void ll_uart1_rs485_enable(void) |
|---|---|
| 功能 | UART1 rs485 使能 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.45 函数 ll_uart1_rs485_disable

| 函数名 | void ll_uart1_rs485_disable(void) |
|---|---|
| 功能 | 禁用 UART1 rs485 |
| 参数 1 | 无 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.46 函数 ll_uart_baudrate_set

| 函数名 | void ll_uart_baudrate_set(UART2_TypeDef *p_uart, uint32_t baudrate) |
|---|---|
| 功能 | 设置波特率 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | Baudrate：设置的波特率 |
| 返回值 | 无 |

### 15.47 函数 ll_uart_work_mode_get

| | |
|---|---|
| 函数名 | TYPE_ENUM_LL_UART_WORK_MODE<br>ll_uart_work_mode_get(UART2_TypeDef *p_uart) |
| 功能 | 获取工作模式 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | 工作模式 |

工作模式

| 定义 | 描述 |
|---|---|
| LL_UART_WORK_MODE_FULL_DUPLEX | 双工 |
| LL_UART_WORK_MODE_SINGLE_SND | 单工发送 |
| LL_UART_WORK_MODE_SINGLE_RCV | 单工接收 |
| LL_UART_WORK_MODE_SINGLE_AUTO | 单线自动 |

### 15.48 函数 ll_uart_work_mode_set

| | |
|---|---|
| 函数名 | void ll_uart_work_mode_set(UART2_TypeDef *p_uart,<br>TYPE_ENUM_LL_UART_WORK_MODE work_mode) |
| 功能 | 获取工作模式 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | work_mode：工作模式 |
| 返回值 | 无 |

**work_mode**：工作模式

| 定义 | 描述 |
|---|---|
| LL_UART_WORK_MODE_FULL_DUPLEX | 双工 |
| LL_UART_WORK_MODE_SINGLE_SND | 单工发送 |
| LL_UART_WORK_MODE_SINGLE_RCV | 单工接收 |
| LL_UART_WORK_MODE_SINGLE_AUTO | 单线自动 |

### 15.49 函数 ll_uart_irq_tx

| 函数名 | void ll_uart_irq_tx(UART2_TypeDef *p_uart, u16 tx_data) |
|---|---|
| 功能 | 中断发送数据 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | tx_data：发送的数据 |
| 返回值 | 无 |
| | 注意：发送完成标志需要在中断中清除，而发送下一个数据在中断中 |

### 15.50 函数 ll_uart_tx

| 函数名 | void ll_uart_tx(UART2_TypeDef *p_uart,u16 tx_data) |
|---|---|
| 功能 | 发送数据 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | tx_data：发送的数据 |
| 返回值 | 无 |
| | 注意：发送完成标志需要在这个函数中清除，不需要中断 |

### 15.51 函数 ll_uart_rx

| 函数名 | bool ll_uart_rx(UART2_TypeDef *p_uart, u16 *rx_data) |
|---|---|
| 功能 | 接收数据 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | rx_data：数据存放地址 |
| 返回值 | False：接收失败　true：接收成功 |

### 15.52 函数 ll_uart_tx_inv_en_get

| 函数名 | FunctionalState ll_uart_tx_inv_en_get(UART2_TypeDef *p_uart) |
|---|---|
| 功能 | 获取发送信号取反状态 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |

| | |
|---|---|
| 参数 2 | 无 |
| 返回值 | ENABLE：使能信号取反　　DISABLE：禁用信号取反 |

### 15.53 函数 ll_uart_rx_inv_en_get

| | |
|---|---|
| 函数名 | FunctionalState ll_uart_rx_inv_en_get(UART2_TypeDef *p_uart) |
| 功能 | 获取接收信号取反状态 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | ENABLE：使能信号取反　　DISABLE：禁用信号取反 |

### 15.54 函数 ll_uart_9bit_en_get

| | |
|---|---|
| 函数名 | FunctionalState ll_uart_9bit_en_get(UART2_TypeDef *p_uart) |
| 功能 | 获取 9 位数据传输状态 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0/1） |
| 参数 2 | 无 |
| 返回值 | ENABLE：使能　　DISABLE：禁用 |

### 15.55 函数 ll_uart0_updata_detect_en_get

| | |
|---|---|
| 函数名 | FunctionalState ll_uart0_updata_detect_en_get(UART2_TypeDef *p_uart) |
| 功能 | uart0 升级检测使能获取 |
| 参数 1 | p_uart：uart 寄存器基地址（uart0） |
| 参数 2 | 无 |
| 返回值 | ENABLE：使能　　DISABLE：禁用 |

### 15.56 函数 ll_uart_dma_tx_addr_set

| | |
|---|---|
| 函数名 | void ll_uart_dma_tx_addr_set(UART2_TypeDef *p_uart, u32 addr) |

| | |
|---|---|
| 功能 | uart1 dma 发送地址设置 |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | Addr:地址 |
| 返回值 | 无 |

### 15.57 函数 ll_uart_dma_rx_addr_set

| | |
|---|---|
| 函数名 | void ll_uart_dma_rx_addr_set(UART2_TypeDef *p_uart, u32 addr) |
| 功能 | uart1 dma 接收地址设置 |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | Addr:地址 |
| 返回值 | 无 |

### 15.58 函数 ll_uart_dma_want_tx_len_set

| | |
|---|---|
| 函数名 | void ll_uart_dma_want_tx_len_set(UART2_TypeDef *p_uart, u16 len) |
| 功能 | uart1 dma 发送的数据长度设置 |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | len:数据长度 |
| 返回值 | 无 |

### 15.59 函数 ll_uart_dma_want_rx_len_set

| | |
|---|---|
| 函数名 | void ll_uart_dma_want_rx_len_set(UART2_TypeDef *p_uart, u16 len) |
| 功能 | uart1 dma 接收的数据长度设置 |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | len:数据长度 |
| 返回值 | 无 |

### 15.60 函数 ll_uart_dma_want_tx_len_get

| 函数名 | u16 ll_uart_dma_want_tx_len_get(UART2_TypeDef *p_uart) |
|---|---|
| 功能 | 获取 uart1 dma 发送的数据长度 |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | 无 |
| 返回值 | 数据长度 |

### 15.61 函数 ll_uart_dma_want_rx_len_get

| 函数名 | u16 ll_uart_dma_want_rx_len_get(UART2_TypeDef *p_uart) |
|---|---|
| 功能 | 获取 uart1 dma 接收的数据长度 |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | 无 |
| 返回值 | 数据长度 |

### 15.62 函数 ll_uart_dma_tx_len_get

| 函数名 | u16 ll_uart_dma_tx_len_get(UART2_TypeDef *p_uart) |
|---|---|
| 功能 | 获取 uart1 dma 已发送的数据长度 |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | 无 |
| 返回值 | 数据长度 |

### 15.63 函数 ll_uart_dma_rx_len_get

| 函数名 | u16 ll_uart_dma_rx_len_get(UART2_TypeDef *p_uart) |
|---|---|
| 功能 | 获取 uart1 dma 已接收的数据长度 |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | 无 |
| 返回值 | 数据长度 |

### 15.64 函数 ll_uart_rs485_re_en_set

| | |
|---|---|
| 函数名 | void ll_uart_rs485_re_en_set(UART2_TypeDef *p_uart, FunctionalState enable) |
| 功能 | uart1 rs485 re 使能设置 |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | Enable：设置状态 |
| 返回值 | 无 |

### 15.65 函数 ll_uart_rs485_de_en_set

| | |
|---|---|
| 函数名 | void ll_uart_rs485_de_en_set(UART2_TypeDef *p_uart, FunctionalState enable) |
| 功能 | uart1 rs485 de 使能设置 |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | Enable：设置状态 |
| 返回值 | 无 |

### 15.66 函数 ll_uart_rs485_mode_get

| | |
|---|---|
| 函数名 | TYPE_ENUM_LL_UART_RS485_MODE ll_uart_rs485_mode_get(UART2_TypeDef *p_uart) |
| 功能 | uart1 rs485 模式获取 |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | 无 |
| 返回值 | rs485_mode：工作模式 |

**rs485_mode**：工作模式

| 定义 | 描述 |
|---|---|
| LL_UART_RS485_MODE_MANUAL | 用户软件切换 |
| LL_UART_RS485_MODE_AUTO | 硬件自动切换 |

### 15.67 函数 ll_uart_rs485_mode_set

| | |
|---|---|
| 函数名 | void ll_uart_rs485_mode_set(UART2_TypeDef *p_uart, TYPE_ENUM_LL_UART_RS485_MODE rs485_mode) |
| 功能 | uart1 rs485 模式设置 |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | 无 |
| 返回值 | 无 |

### 15.68 函数 ll_uart_re_pol_set

| | |
|---|---|
| 函数名 | void ll_uart_re_pol_set(UART2_TypeDef *p_uart, TYPE_ENUM_LL_UART_RS485_RE_DE_POL pol) |
| 功能 | uart1 rs485 re 极性设置 |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | Pol：极性 |
| 返回值 | 无 |

**pol**：re 极性

| 定义 | 描述 |
|---|---|
| LL_UART_RS485_RE_DE_POL_H | 高电平有效 |
| LL_UART_RS485_RE_DE_POL_L | 低电平有效 |

### 15.69 函数 ll_uart_de_pol_set

| | |
|---|---|
| 函数名 | void ll_uart_de_pol_set(UART2_TypeDef *p_uart, TYPE_ENUM_LL_UART_RS485_RE_DE_POL pol) |
| 功能 | uart1 rs485 de 极性设置 |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | Pol：极性 |
| 返回值 | 无 |

**pol**：de 极性

| 定义 | 描述 |
| --- | --- |
| LL_UART_RS485_RE_DE_POL_H | 高电平有效 |
| LL_UART_RS485_RE_DE_POL_L | 低电平有效 |

### 15.70 函数 ll_uart_rs485_en_set

| 函数名 | void ll_uart_rs485_en_set(UART2_TypeDef *p_uart, FunctionalState enable) |
| --- | --- |
| 功能 | uart1 rs485 使能设置 |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | enable：状态 |
| 返回值 | 无 |

### 15.71 函数 ll_uart_rs485_det_de_at_set

| 函数名 | void ll_uart_rs485_det_de_at_set(UART2_TypeDef *p_uart, u16 interval) |
| --- | --- |
| 功能 | 时间间隔配置（de 上升沿到开始位） |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | interva：时间间隔 |
| 返回值 | 无 |

### 15.72 函数 ll_uart_rs485_det_de_dat_set

| 函数名 | void ll_uart_rs485_det_de_dat_set(UART2_TypeDef *p_uart,u16 interval) |
| --- | --- |
| 功能 | 时间间隔配置（停止位到 de 下降沿） |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | interva：时间间隔 |
| 返回值 | 无 |

### 15.73 函数 ll_uart_rs485_tat_de2re_t_set

| | |
|---|---|
| 函数名 | void ll_uart_rs485_tat_de2re_t_set(UART2_TypeDef *p_uart, u16 interval) |
| 功能 | 时间间隔配置（de 下降沿到 re 上升沿） |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | interva：时间间隔 |
| 返回值 | 无 |

### 15.74 函数 ll_uart_rs485_tat_re2de_t_set

| | |
|---|---|
| 函数名 | void ll_uart_rs485_tat_re2de_t_set(UART2_TypeDef *p_uart, u16 interval) |
| 功能 | 时间间隔配置（re 下降沿到 de 上升沿） |
| 参数 1 | p_uart：uart 寄存器基地址（uart1） |
| 参数 2 | interva：时间间隔 |
| 返回值 | 无 |

### 15.75 模块相关宏定义

| 定义 | 描述 |
|---|---|
| LL_UART_RX_TIMEOUT_INTERRUPT_GET(p_uart) | UART 接收超时中断获取 |
| LL_UART_FERR_INTERRUPT_GET(p_uart) | UART 帧错误中断获取 |
| LL_UART_TX_INTERRUPT_GET(p_uart) | 发送完成中断获取 |
| LL_UART1_RX_DMA_PERR_INTERRUPT_GET() | UART1 接收 DMA 数据奇偶校验出错中断获取 |
| LL_UART1_RX_DMA_INTERRUPT_GET() | UART1 接收 DMA 完成中断获取 |
| LL_UART1_TX_DMA_INTERRUPT_GET() | UART1 发送 DMA 完成中断获取 |
| LL_UART_TMR_PWM_EN_GET(p_uart) | UART 的输出和 TMR 的 PWM 一起运算后才输出使能获取 |
| LL_UART_RX_TIMEOUT_EN_GET(p_uart) | UART 接收超时使能获取 |

| | |
|---|---|
| LL_UART_TX_INV_EN_GET(p_uart) | UART 发送输出信号取反使能获取 |
| LL_UART_RX_INV_EN_GET(p_uart) | UART 接收输入信号取反使能获取 |
| LL_UART_ODD_EN_GET(p_uart) | UART 奇偶检验选择获取 |
| LL_UART_PARITY_EN_GET(p_uart) | UART 奇偶检验使能获取 |
| LL_UART_9BIT_EN_GET(p_uart) | UART 传输 9bit 数据使能获取 |
| LL_UART_EN_GET(p_uart) | UART 使能获取 |
| LL_UART1_DMA_RX_EN_GET() | UART1 DMA 接收使能获取 |
| LL_UART1_DMA_TX_EN_GET() | UART1 DMA 发送使能获取 |
| LL_UART1_RS485_RE_EN_GET() | UART1 RS485RE 模式使能获取 |
| LL_UART1_RS485_DE_EN_GET() | UART1 RS485DE 模式使能获取 |
| LL_UART1_RS485_EN_GET() | UART1 RS485 模式使能获取 |
| LL_UART_RX_PERR_GET(p_uart) | UART 发送帧错误获取 |
| LL_UART_RX_CNT_GET(p_uart) | UART 接收数据个数获取 |
| LL_UART0_UPDATE_DETECT_G_PENDING_GET() | UART0 检测升级标志获取仅 uart0 |
| LL_UART0_UPDATE_DETECT_PENDING_GET() | UART0 检测升级标志获取 |
| LL_UART0_UPDATE_DETECT_PENDING_CLR() | UART0 检测升级标志清除 |
| LL_UART_RX_TIMEOUT_PENDING_GET(p_uart) | UART 接收数据超时状态获取 |
| LL_UART_RX_TIMEOUT_PENDING_CLR(p_uart) | UART 接收数据超时状态清除 |
| LL_UART_RX_FERR_PENDING_GET(p_uart) | UART 接收数据帧出错状态获取 |
| LL_UART_RX_FERR_PENDING_CLR(p_uart) | UART 接收数据帧出错状态清除 |
| LL_UART_RX_BUF_OV_PENDING_GET(p_uart) | UART 接收数据缓存溢出状态获取 |
| LL_UART_RX_BUF_OV_PENDING_CLR(p_uart) | UART 接收数据缓存溢出状态清除 |

| | |
|---|---|
| LL_UART_RX_BUF_NOT_EMPTY_PENDING_GET(p_uart) | UART 接收数据缓存不空状态获取 |
| LL_UART_RX_BUF_NOT_EMPTY_PENDING_CLR(p_uart) | UART 接收数据缓存不空状态清除 |
| LL_UART_TX_DONE_PENDING_GET(p_uart) | UART 发送数据完成标志获取 |
| LL_UART_TX_DONE_PENDING_CLR(p_uart) | UART 发送数据完成标志清除 |
| LL_UART1_RX_DMA_PERR_PENDING_GET() | UART1DMA 接收数据帧错误标志获取 |
| LL_UART1_RX_DMA_PERR_PENDING_CLR() | UART1 DMA 接收数据帧错误标志清除 |
| LL_UART1_RX_DMA_PENDING_GET() | UART1 DMA 接收数据完成标志获取 |
| LL_UART1_RX_DMA_PENDING_CLR() | UART1 DMA 接收数据完成标志清除 |
| LL_UART1_TX_DMA_PENDING_GET() | UART1 DMA 发送数据完成标志获取 |
| LL_UART1_TX_DMA_PENDING_CLR() | UART1 DMA 发送数据完成标志清除 |

# 16 看门狗 wdt

Watchdog 工作时钟为 32K。默认定时 2S，可以通过修改分频系数改变看门狗超时时间。通过配置寄存器，可以选择当计时溢出时，复位系统或者产生中断。

## 16.1 函数 ll_wdt_feed

| 函数名 | void ll_wdt_feed(WDT_TypeDef *p_wdt) |
|---|---|
| 功能 | 清除看门狗寄存器 |
| 参数 1 | p_wdt：wdt 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

## 16.2 函数 ll_wdt_init

| 函数名 | void ll_wdt_init(WDT_TypeDef *p_wdt, TYPE_LL_WDT_INIT *p_init) |
|---|---|
| 功能 | 看门狗初始化 |
| 参数 1 | p_wdt：wdt 寄存器基地址 |
| 参数 2 | p_init：指向包含了指定外设的配置信息的结构体 |
| 返回值 | 无 |

```
typedef struct __ll_wdt_init {
    TYPE_ENUM_LL_WDT_PSR      psr;
    TYPE_ENUM_LL_WDT_RMODE    mode;
} TYPE_LL_WDT_INIT;
```

**Psr**：预分频

| 定义 | 描述 |
|---|---|
| LL_WDT_PSR_0DIV | 不分频 |
| LL_WDT_PSR_2DIV | 2 分频 |
| ..... | ..... |

| | |
|---|---|
| LL_WDT_PSR_32768DIV | 32768 分频 |

**Mode**：超时模式选择

| 定义 | 描述 |
|---|---|
| LL_WDT_RMODE_RST | 复位 |
| LL_WDT_RMODE_INT | 中断 |

### 16.3 函数 ll_wdt_deinit

| | |
|---|---|
| 函数名 | void ll_wdt_deinit(WDT_TypeDef *p_wdt) |
| 功能 | 释放看门狗 |
| 参数 1 | p_wdt：wdt 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 16.4 函数 ll_wdt_start

| | |
|---|---|
| 函数名 | void ll_wdt_start(WDT_TypeDef *p_wdt) |
| 功能 | 看门狗启动 |
| 参数 1 | p_wdt：wdt 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |

### 16.5 函数 ll_wdt_stop

| | |
|---|---|
| 函数名 | void ll_wdt_stop(WDT_TypeDef *p_wdt) |
| 功能 | 看门狗关闭 |
| 参数 1 | p_wdt：wdt 寄存器基地址 |
| 参数 2 | 无 |
| 返回值 | 无 |