用户手册

32 位基于 ARM 微控制器AIR001 HAL 驱动库和 LL 驱动库函数说明

介绍

HAL 驱动库已经实现了适用于AIR001 系列芯片的一整套 APIs，这些 APIs 能够使应用程序与底层硬件之间的交互更加简单、方便。

在 HAL 驱动库中用户能够调用的 APIs 可以分为两类：通用 APIs 和扩展 APIs。通用 APIs 为所有 AIR001 系列芯片提供通用功能的驱动。扩展 APIs 则根据不同型号提供扩展功能的 APIs。

HAL 驱动库并不是基于 IP 所构建的，而是基于外设的特性和功能实现的。例如，USART 拥有 UART 和 USART 两种功能，每种功能都拥有一组独立的驱动程序来支持，并且它们的驱动程序是相互分离的。

HAL 驱动库函数的入口处均有断言函数，断言函数用来校验输入参数是否合法。这种校验方式提高了驱动程序的健壮性。用户也可以使用断言函数来进行编写和调试应用程序。

HAL 驱动库提供的 APIs 均具有很高的可移植性，并且它们对用户屏蔽了 MCU 和底层硬件实现功能时的复杂流程。

LL 驱动库根据 AIR001 外设特点提供基于寄存器的函数，这些函数准确的反应了硬件功能，并且必须按照 AIR001 参考手册中描述的编程模型进行调用。所以 LL 驱动程序不是基于独立进程的，也不需要任何额外的内存资源来保持它们的状态、计数值、数据指针等。所有操作都是通过修改相关外设寄存器的内容来执行的。

HAL 驱动库和LL 驱动库是互补的，它们涵盖了广泛的应用需求：

- HAL 提供具有高可移植性和面向特性的API，向最终用户隐藏了 MCU 和外设的复杂性。
- LL 提供了寄存器级别的低级 API，具有更好的优化性但可移植性较差，且需要深入了解 MCU 和外设规范。

本手册结构如下：

- 缩写和定义
- HAL 驱动概述
- HAL 驱动程序说明

目录

目录

1 文档和库规范

1.1 缩写

表1-1 缩写

缩写	外设/单元
ADC	模数转换器
COMP	比较器
CRC	循环冗余校验
DMA	直接存储器存取
EXTI	外部中断/事件控制器
FLASH	闪存存储器
GPIO	通用输入输出
I2C	内部集成电路总线
ISR	中断服务程序
IWDG	独立看门狗
NVIC	嵌套中断向量列表控制器
LED	数码管控制器
MSP	芯片级支持包
LPTIM	低功耗定时器
PWR	电源/功耗控制
RCC	复位与时钟控制器
RTC	实时时钟
SPI	串行外设接口
Systick	系统滴答定时器
TIM1	高级控制定时器
TIM	通用定时器
USART	通用同步异步收发器
WWDG	窗口看门狗

2 HAL 驱动库概述

HAL 驱动库设计的目的是为了提供一组用户能够轻松简单地和底层硬件交互的 APIs。每个外设的驱动程序都由一组结构体和函数组成，这些函数涵盖了外设的常见功能。他们由一个句柄驱动，驱动程序中的所有结构体、函数、参数都依赖于该句柄（PPP_HandleTyprDef）。

HAL 驱动库中每个 IP 都与驱动程序一一对应，但是当 IP 具有多个功能时，则该 IP 的每个功能与驱动程序相对应。例如USART 拥有：UART 和 USART，它们都有独立的驱动程序。

HAL 驱动库的主要特点如下：

- 拥有可以跨系列移植的APIs 集，该 APIs 集包含了通用驱动 APIs 和扩展驱动APIs。
- 三种编程模式：轮询，中断，DMA。
- HAL APIs 与 RTOS 兼容。
- 支持同一外设同时定义多个实例（USART1，USART2…），并且这些实例能够并行调用相关 API。
- 所有HAL APIs 都实现了用户回调函数机制：
  - 外设初始化/去初始化时对MCU 底层硬件的初始化/去初始化回调。
  - 外设中断回调。
  - 外设错误回调。
- 对象锁定机制：HAL 锁能够防止程序对共享硬件资源的同时访问。
- 用于所有阻塞进程的超时机制:超时可以是一个简单的计数器或根据时基判断是否超时。

2.1 HAL 和用户应用程序文件

2.1.1 HAL 驱动库文件

HAL 驱动库由以下文件组成：

表2-1 HAL 驱动库文件

文件	说明
air001xx_hal_ppp.c	主要外设的驱动程序。 该类文件包含所有 AIR001 芯片的通用 APIs。 例如：air001xx_hal_adc.c，air001xx_hal_uart.c，……
air001xx_hal_ppp.h	主要外设驱动的头文件。 该类文件包含外设共有数据结构，枚举结构，宏定义，以及通用驱动的函数声明。 例如：air001xx_hal_adc.h，air001xx_hal_uart.h，……

air001xx_hal_ppp_ex.c	外设扩展功能的驱动文件。该类文件包含了特定型号芯片的特殊功能函数。 例如：air001xx_hal_adc_ex.c，air001xx_hal_flash_ex.c
air001xx_hal_ppp_ex.h	扩展驱动的头文件。 包括额外的数据结构、枚举结构，宏定义，以及特定型号的函数声明。 例如：air001xx_hal_adc_ex.h，air001xx_hal_flash_ex.h
air001xx_hal.c	该文件用于 HAL 驱动库初始化，包含了DBGMCU、重映射、基于 SysTick 的延时函数等。
air001xx_hal_h	air001xx_hal.c 的头文件。
air001xx_def.h	该文件包含常用的HAL 驱动库通用资源，如常用的枚举结构、宏。

2.1.2 用户应用程序文件

下表列出了使用 HAL 驱动库构建应用程序所必需的文件：

表2-2 用户应用程序文件

文件	说明
system_air001xx.c	这个文件包含SystemInit()，该函数在复位后，跳转到主函数之前被调用，它配置了系统时钟和异常向量表的偏移地址。
startup_air001xx.s	芯片启动文件。 包含了复位处理和异常向量表初始化，以及配置栈/堆大小用来适应应用程序需求。
air001xx_hal_msp.c	这个文件包含用户应用程序中使用的外设的 MSP 初始化和去初始化。
air001xx_hal_conf.h	用户可以注释/取消注释来定制所用户应用程序中需要使用的外设，也可以修改宏定义参数来适配用户应用程序。 此文件配置不强制修改。应用程序可以使用默认配置。
air001xx_it.c/.h	该文件包含异常处理函数和外设中断服务函数。如果应用程序中使用了基于中断的进程，PPP_IRQHandler()函数必须调用 HAL_PPP_IRQHandler()。 其中SysTick_Handler()会定期调用 HAL_IncTick()来使作为时基的全局变量“uwTick”自增。（缺省情况下，系统 ISR 每 1ms 调用一次该函数)
air001xx_Start_Kit.c	该文件包含了对板级资源外设的初始化函数，例如LED 灯的初始化以及点亮/熄灭函数实现，按键的初始化以及获取按键状态函数。
main.c/.h	这个文件包含主程序需要调用的函数，主要是:

	• HAL_Init()。 • assert_failed()实现。 • 系统时钟配置。 • 外设初始化和用户应用程序代码。

2.2 HAL 数据结构

每个HAL 驱动程序包含以下结构：

- 外设句柄结构。
- 初始化和配置结构。
- 具体的功能函数。

2.2.1 外设句柄结构

外设句柄结构体具有模块化，多实例化的特点，同一句柄结构体能够同时定义多个外设实例。

PPP_HandleTypeDef *handle 是 HAL 驱动程序中的主要结构体。它处理外设参数配置和外设寄存器配置，并包含外设工作流程中所需要的所有结构体和变量。

外设句柄用于以下目的：

- 多实例支持：每个外设实例都有自己的句柄。因此，实例资源是独立的。
- 外设进程间通信:句柄能够管理进程之间共享的数据资源。
  例如：全局指针，DMA 句柄。
- 存储：句柄还用于管理HAL 驱动程序中的全局变量。

外设句柄示例如下：

```
typedef struct __UART_HandleTypeDef
{
USART_TypeDef *Instance; /*!< UART registers base address*/
UART_InitTypeDef Init; /*!< UART communication parameters*/
UART_AdvFeatureInitTypeDef AdvancedInit; /*!< UART Advanced Features initialization parameters
*/
uint8_t *pTxBuffPtr; /*!< Pointer to UART Tx transfer Buffer */
uint16_t TxXferSize; /*!< UART Tx Transfer size*/
__IO uint16_t TxXferCount; /*!< UART Tx Transfer Counter*/
uint8_t *pRxBuffPtr; /*!< Pointer to UART Rx transfer Buffer */
uint16_t RxXferSize; /*!< UART Rx Transfer size*/
__IO uint16_t RxXferCount; /*!< UART Rx Transfer Counter*/
void (*RxISR)(struct __UART_HandleTypeDef *huart);     /*!< Function pointer on Rx IRQ handler */
void (*TxISR)(struct __UART_HandleTypeDef *huart);     /*!< Function pointer on Tx IRQ handler */
```

```
DMA_HandleTypeDef *hdmatx; /*!< UART Tx DMA Handle parameters*/
DMA_HandleTypeDef *hdmarx; /*!< UART Rx DMA Handle parameters*/
HAL_LockTypeDef Lock; /*!< Locking object*/
__IO HAL_UART_StateTypeDef gState; /*!< UART state information related to global Handle management and also related to Tx operations. */
__IO HAL_UART_StateTypeDef RxState; /*!< UART state information related to Rx operations. */
__IO uint32_t ErrorCode; /*!< UART Error code*/
} UART_HandleTypeDef;
```

1. 多实例特性意味着应用程序中使用的所有函数都是可重入函数，因此所有的函数都应该避免使用全局变量。但是在可重入函数中可以处理全局数据，例如读串口缓冲区，不过应该尽量减少这样的操作。同时可重入函数在运行过程中不能修改自己的代码。
2. 使用 DMA 同时管理多个外设实例时，应该将 DMA 接口句柄添加到每个进程的 PPP_HandleTypeDef 中。
3. 对于共享外设和系统外设，不使用句柄或实例对象。涉及的外设如下:
   - GPIO
   - SYSTICK
   - NVIC
   - PWR
   - RCC
   - FLASH

2.2.2 初始化和配置结构

当初始化结构体和配置结构体在同系列所有型号芯片中都适用时，这些结构体被定义在 HAL 通用驱动头文件当中，当在不同型号芯片中有不同的定义时，这些会改变的结构体被定义在 HAL 扩展驱动头文件当中。

Init 结构用于初始化外设。例如:

- Init 结构体:

```
typedef struct
{
uint32_t ClockPrescaler;
uint32_t Resolution;
uint32_t DataAlign;
uint32_t ScanConvMode;
uint32_t EOCSelection;
FunctionalState LowPowerAutoWait;
FunctionalState ContinuousConvMode;
FunctionalState DiscontinuousConvMode;
uint32_t ExternalTrigConv;
```

```
uint32_t ExternalTrigConvEdge;
FunctionalState DMAContinuousRequests;
uint32_t Overrun;
uint32_t SamplingTimeCommon;
} ADC_InitTypeDef;
```

- 配置函数：

```
HAL_StatusTypeDef HAL_ADC_Init(ADC_HandleTypeDef* hadc);
```

Config 结构用于初始化外设的子模块或子实例。例如：

- Config 结构体：

```
typedef struct
{
uint32_t Channel;
uint32_t Rank;
uint32_t SamplingTime;
} ADC_ChannelConfTypeDef;
```

- 配置函数：

```
HAL_ADC_ConfigChannel (ADC_HandleTypeDef* hadc, ADC_ChannelConfTypeDef* sConfig);
```

2.2.3 具体的进程函数

具体的进程函数（通用 APIs）用于实现具体的功能。它们在通用驱动程序头文件中声明。

例子：

```
HAL_StatusTypeDef HAL_ADC_Start(ADC_HandleTypeDef* hadc);
HAL_StatusTypeDef HAL_ADC_Stop(ADC_HandleTypeDef* hadc);
HAL_StatusTypeDef HAL_ADC_PollForConversion(ADC_HandleTypeDef* hadc, uint32_t Timeout);
HAL_StatusTypeDef HAL_ADC_PollForEvent(ADC_HandleTypeDef* hadc, uint32_t EventType, uint32_t Timeout);
```

2.3 API 分类

HAL 的 APIs 可以分为两类：

- **通用型 APIs：**适用于所有 AIR001 芯片的通用 API，这些 API 在相关外设驱动文件的头文件中被声明。

```
HAL_StatusTypeDef HAL_ADC_Init(ADC_HandleTypeDef* hadc);
HAL_StatusTypeDef HAL_ADC_DeInit(ADC_HandleTypeDef *hadc);
HAL_StatusTypeDef HAL_ADC_Start(ADC_HandleTypeDef* hadc);
HAL_StatusTypeDef HAL_ADC_Stop(ADC_HandleTypeDef* hadc);
HAL_StatusTypeDef HAL_ADC_Start_IT(ADC_HandleTypeDef* hadc);
```

```
HAL_StatusTypeDef HAL_ADC_Stop_IT(ADC_HandleTypeDef* hadc);
void HAL_ADC_IRQHandler(ADC_HandleTypeDef* hadc);
```

- **扩展 APIs：**
  应用于特定外设的扩展 APIs。在相关外设扩展驱动文件的头文件声明。(参见下面与 TIM 相关的示例)

```
HAL_StatusTypeDef HAL_TIMEx_HallSensor_Init
(TIM_HandleTypeDef *htim,TIM_HallSensor_InitTypeDef *sConfig);
```

2.4 HAL 驱动库共用资源

在 air001xx_hal_def.h 中定义了常见的 HAL 驱动库资源，例如 HAL 状态的枚举定义，DMA 句柄连接到外设的宏定义。其中最主要的通用枚举定义是HAL_StatusTypeDef。

- **HAL 状态**
  HAL 状态几乎被所有 HAL 驱动库的 APIs 使用。API 执行完毕后会返回 HAL 状态来表示该 API 的执行结果。它有以下四种值：

```
typedef enum
{
HAL_OK          = 0x00U,
HAL_ERROR    = 0x01U,
HAL_BUSY     = 0x02U,
HAL_TIMEOUT  = 0x03U
} HAL_StatusTypeDef;
```

- **HAL 锁**
  所有 HAL 驱动库共享资源都使用 HAL 锁，防止被非法访问。

```
typedef enum
{
HAL_UNLOCKED = 0x00U,
HAL_LOCKED   = 0x01U
} HAL_LockTypeDef;
```

  除了常用资源外，air001xx_hal_def.h 文件还会调用 CMSIS 库中的 air001xx.h 文件来获取所有外设的数据结构和地址映射。

- **常见的宏**
  - 宏定义 NULL

```
#undef NULL
#define NULL 0
```

  - 宏定义 HAL_MAX_DELAY

```
#define HAL_MAX_DELAY      0xFFFFFFFFU
```

  - 将 PPP 外设链接到 DMA 句柄的宏:

```
#define __HAL_LINKDMA(__HANDLE__, __PPP_DMA_FIELD__, __DMA_HANDLE__)\
```

```
do{\
(__HANDLE__)->__PPP_DMA_FIELD___= &(__DMA_HANDLE__); \
(__DMA_HANDLE__).Parent = (__HANDLE__);\
} while(0U)
```

2.5 HAL 配置

用户可以通过配置文件“air001xx_hal_conf.h”为应用程序定制 HAL 驱动库全局配置。

若不修改此配置文件，应用程序将使用默认配置。如果需要修改配置文件，用户应该通过注释/取消注释或修改相关定义语句的值来禁用/启用或修改某些选项，部分定义语句如下表所示:

表2-3 定义用于 HAL 配置的声明

配置项	说明	默认值
HSE_VALUE	定义外部振荡器(HSE)的值(Hz)。 当使用不同的晶振时，用户必须调整此宏定义的值。	24,000,000 (Hz)
HSE_STARTUP_TIMEOUT	HSE 启动超时时间。	200 (ms)
HSI_VALUE	定义内部振荡器(HSI)的值(Hz)。	8,000,000 (Hz)
LSE_VALUE	定义外部振荡器(LSE)的值(Hz)。当使用不同的晶振时，用户必须调整此宏定义的值。	32,768 (Hz)
LSE_STARTUP_TIMEOUT	HSE 启动超时时间。	5,000 (ms)
LSI_VALUE	定义内部低速振荡器的值(Hz)。实际值可能会随着电压和温度的变化而变化。	32,768 (Hz)
VDD_VALUE	VDD 值	3,300 (mV)
PRIORITY_HIGHEST	中断优先级：最高	0
PRIORITY_HIGH	中断优先级：高	1
PRIORITY_LOW	中断优先级：低	2
PRIORITY_LOWEST	中断优先级：最低	3
TICK_INT_PRIORITY	TICK 中断优先级	PRIORITY_LOWEST
USE_RTOS	启用RTOS 功能	FALSE
PREFETCH_ENABLE	启用FLASH 预取功能	TRUE

默认情况下，air001xx_hal_conf.h 文件中定义的值与示例中使用的值相同。并且 HAL 驱动库中所有的 C 文件都包含该头文件，因此这些宏定义可以在用户代码中直接使用。

3 LL 驱动库概述

LL 驱动程序与HAL 驱动程序相比，提供了更接近硬件、程序体积更小、执行效率更高的驱动程序。

LL 驱动程序具有以下特点：

- 一组函数，可以根据结构体中定义的参数，初始化外设主要功能；
- 一组函数，用于将初始化结构体中的字段填充为重置值；
- 去初始化函数（外设寄存器恢复到默认值）；
- 一组内联函数，其用于直接寄存器访问和原子寄存器访问；
- 完全独立于HAL 库，LL 驱动可以独立使用（没有 HAL 驱动），也可以混合使用（有 HAL 驱动）；
- 覆盖所有外设支持的功能。

LL 驱动库根据 AIR001 外设特点提供基于寄存器的函数，这些函数准确的反应了硬件功能，所以必须按照 AIR001 参考手册中描述的编程模型进行调用。并且这些函数都是一次性操作，不会执行任何对调用结果的处理。所以不需要任何额外的内存资源来保存他们的状态、计数值、数据指针等。所有的操作都是通过修改相关外设的寄存器来执行的。

3.1 LL 驱动文件

LL 驱动库由外设驱动头文件、C 文件，系统和Cortex 相关功能文件构成。

表3-1 LL 驱动文件

文件	说明
air001xx_ll_bus.h	这个文件包含用于核心总线控制和外设时钟的使能与禁用的函数。 例如：LL_AHB1_GRP1_EnableClock…
air001xx_ll_ppp.h/.c	air001xx_ll_ppp.c 提供了LL_PPP_Init()、LL_PPP_StructInit()、LL_PPP_DeInit()等外设初始化函数。所有其他 API 都在 air001xx_ll_ppp.h 中定义。
air001xx_ll_cortex.h	Cortex-M 相关寄存器操作API，包括 SysTick、Low power 等。
air001xx_ll_utils.h/.c	该文件包含了读取设备唯一 ID 和电子签名、时基和延迟管理、系统时钟配置的函数。
air001xx_ll_system.h	系统相关操作。 例如：LL_SYSCFG_xxxx、LL_FLASH_xxxx…
air001_assert.h	该文件定义了断言函数模板，当用户要使用断言时在该文件中定义断言函数。

4 HAL 系统驱动程序

4.1 HAL 库系统驱动程序 API 描述

下面的部分列出了HAL 库系统驱动的各种功能。

4.1.1 如何使用 HAL 库系统驱动程序

HAL 系统驱动程序包含一组通用的 APIs，可以被外设驱动程序和用户调用。

HAL 包含两类APIs：

- HAL 初始化和去初始化功能。
- HAL 控制功能。

4.1.2 初始化和去初始化函数

本节描述以下功能：

- 初始化 NVIC 配置和时钟基准源配置。
- 将 HAL 的通用功能寄存器设为缺省值。
- 使用SysTick 中断产生 1ms 时基，SysTick 中断优先级默认配置为最低优先级。
  - 默认情况下，使用系统时钟作为 SysTick 的时钟源，但是用户也可以配置合适的时钟作为 SysTick 时钟源(例如通用计时器)， 要注意 SysTick 计数时间应该保持为 1ms， 因为 PPP_TIMEOUT_VALUEs 是以毫秒为单位进行定义和处理的。
  - SysTick 配置函数(HAL_InitTick())在系统复位后重启时由 HAL_Init()自动调用，或者在配置时钟时由 HAL_RCC_ClockConfig()在配置时钟后调用。
  - SysTick 被配置成每经过一段固定的时间后产生中断。如果外设 ISR 进程调用 HAL_Delay()，必须注意，SysTick 中断必须比外设中断具有更高的优先级(数字上更低)。否则，外设 ISR 进程将被阻塞。
  - 影响SysTick配置的函数被声明为__Weak，所以可以在用户文件中重写它满足应用程序需求。

本节描述包含以下 APIs：

表4-1 HAL 初始化和去初始化函数说明

函数名	描述
HAL_Init	配置时基源、NVIC 和 MCU 底层硬件
HAL_DeInit	将 HAL 的通用功能寄存器设为缺省值，并停止 SysTick
HAL_MspInit	初始化全局 MSP
HAL_MspDeInit	将全局 MSP 设为缺省值
HAL_InitTick	配置SysTick，NVIC

4.1.3 HAL 控制功能

表4-2 HAL 控制功能函数说明

函数名	描述
HAL_IncTick	使全局变量“uwTick”自增
HAL_GetTick	获取SysTick 的计数值，单位：毫秒
HAL_Delay	提供以毫秒为单位的阻塞延迟
HAL_SuspendTick	暂停Systick 中断
HAL_ResumeTick	恢复Systick 中断
HAL_GetHalVersion	获取HAL API 版本
HAL_GetREVID	获取版本标识符
HAL_GetDEVID	获取设备标识符
HAL_GetUIDw0	获取设备版本标识符第一个字
HAL_GetUIDw1	获取设备版本标识符第二个字
HAL_GetUIDw2	获取设备版本标识符第三个字

4.2 功能详细说明

4.2.1 函数 HAL_Init

描述了函数 HAL_Init

表4-3 函数 HAL_Init

函数名	HAL_Init
函数原形	HAL_StatusTypeDef HAL_Init(void)
功能描述	配置 SysTick 作为时基源，NVIC 中断优先级和 MCU 底层硬件初始化
输入参数	无
输出参数	无
返回值	HAL 状态
先决条件	无

4.2.2 函数 HAL_DeInit

描述了函数 HAL_DeInit

表4-4 函数 HAL_DeInit

函数名	HAL_DeInit
函数原形	HAL_StatusTypeDef HAL_DeInit (void)
功能描述	这个函数将 HAL 的通用功能寄存器设为缺省值，并停止 SysTick
输入参数	无

输出参数	无
返回值	HAL 状态
先决条件	无

4.2.3 函数 HAL_MspInit

描述了函数 HAL_MspInit

表4-5 函数 HAL_MspInit

函数名	HAL_MspInit
函数原形	void HAL_MspInit (void)
功能描述	初始化全局 MSP
输入参数	无
输出参数	无
返回值	无
先决条件	无

4.2.4 函数 HAL_MspDeInit

描述了函数 HAL_MspDeInit

表4-6 函数 HAL_MspDeInit

函数名	HAL_MspDeInit
函数原形	void HAL_MspDeInit (void)
功能描述	将全局 MSP 设为缺省值
输入参数	无
输出参数	无
返回值	无
先决条件	无

4.2.5 函数 HAL_InitTick

描述了函数 HAL_InitTick

表4-7 函数 HAL_InitTick

函数名	HAL_InitTick
函数原形	HAL_StatusTypeDef HAL_InitTick (uint32_t TickPriority)
功能描述	配置 SysTick，NVIC
输入参数	TickPriority：中断优先级
输出参数	无
返回值	HAL 状态
先决条件	无

4.2.6 函数 HAL_IncTick

描述了函数 HAL_IncTick

表4-8 函数 HAL_IncTick

函数名	HAL_IncTick
函数原形	void HAL_IncTick (void)
功能描述	使全局变量“uwTick”自增
输入参数	无
输出参数	无
返回值	无
先决条件	无

4.2.7 函数 HAL_Delay

描述了函数 HAL_Delay

表4-9 函数 HAL_Delay

函数名	HAL_Delay
函数原形	void HAL_Delay (__IO uint32_t Delay)
功能描述	提供以毫秒为单位的阻塞延迟
输入参数	Delay：延时时间的值，单位：毫秒
输出参数	无
返回值	无
先决条件	无

4.2.8 函数 HAL_GetTick

描述了函数 HAL_GetTick

表4-10 函数 HAL_GetTick

函数名	HAL_GetTick
函数原形	uint32_t HAL_GetTick (void)
功能描述	获取 SysTick 的计数值，单位：毫秒
输入参数	无
输出参数	无
返回值	Tick：计数值
先决条件	无

4.2.9 函数 HAL_SuspendTick

描述了函数 HAL_SuspendTick

表4-11 函数 HAL_SuspendTick

函数名	HAL_SuspendTick
函数原形	void HAL_SuspendTick (void)
功能描述	暂停 SysTick 中断
输入参数	无
输出参数	无
返回值	无
先决条件	无

4.2.10 函数 HAL_ResumeTick

描述了函数 HAL_ResumeTick

表4-12 函数 HAL_ResumeTick

函数名	HAL_ResumeTick
函数原形	void HAL_ResumeTick (void)
功能描述	恢复 SysTick 中断
输入参数	无
输出参数	无
返回值	无
先决条件	无

4.2.11 函数 HAL_GetHalVersion

描述了函数 HAL_GetHalVersion

表4-13 函数 HAL_GetHalVersion

函数名	HAL_GetHalVersion
函数原形	uint32_t HAL_GetHalVersion (void)
功能描述	获取 HAL API 版本
输入参数	无
输出参数	无
返回值	版本号 SysTick
先决条件	无

4.2.12 函数 HAL_GetREVID

描述了函数 HAL_GetREVID

表4-14 函数 HAL_GetREVID

函数名	HAL_GetREVID
函数原形	uint32_t HAL_GetREVID (void)

功能描述	返回版本标识符
输入参数	无
输出参数	无
返回值	版本标识符
先决条件	无

4.2.13 函数 HAL_GetDEVID

描述了函数 HAL_GetDEVID

表4-15 函数 HAL_GetDEVID

函数名	HAL_GetDEVID
函数原形	uint32_t HAL_GetDEVID (void)
功能描述	返回设备标识符
输入参数	无
输出参数	无
返回值	设备标识符
先决条件	无

4.2.14 函数 HAL_GetUIDw0

描述了函数 HAL_GetUIDw0

表4-16 函数 HAL_GetUIDw0

函数名	HAL_GetUIDw0
函数原形	uint32_t HAL_GetUIDw0 (void)
功能描述	获取设备版本标识符第一个字
输入参数	无
输出参数	无
返回值	设备标识符
先决条件	无

4.2.15 函数 HAL_GetUIDw1

描述了函数 HAL_GetUIDw1

表4-17 函数 HAL_GetUIDw1

函数名	HAL_GetUIDw1
函数原形	uint32_t HAL_GetDEVID (void)
功能描述	获取设备版本标识符第二个字
输入参数	无
输出参数	无

返回值	设备标识符
先决条件	无

4.2.16 函数 HAL_GetUIDw2

描述了函数 HAL_GetUIDw2

表4-18 函数 HAL_GetUIDw2

函数名	HAL_GetUIDw2
函数原形	uint32_t HAL_GetUIDw2 (void)
功能描述	获取设备版本标识符第三个字
输入参数	无
输出参数	无
返回值	设备标识符
先决条件	无

5 HAL 模拟/数字转换器通用驱动程序（ADC）

12 位 ADC 是逐次逼近型模数转换器。它具有多达 12 个复用通道，可测量来自 10 个外部源和 2 个内部源的信号。

各种不同通道的 A/D 转换可在单次、连续、扫描或不连续采样模式下进行。ADC 的结果存储在一个左对齐或右对齐的 16 位数据寄存器中。

ADC 具有模拟看门狗特性，允许应用检测输入电压是否超过了用户自定义的阈值上限或下限。

ADC 实现了在低频率下运行，可获得极低的功耗。

5.1 ADC 固件驱动寄存器结构

5.1.1 ADC_InitTypeDef

ADC_InitTypeDef，定义于文件“air001xx_hal_adc.h”如下：

```
typedef struct
{
uint32_t ClockPrescaler;
uint32_t Resolution;
uint32_t DataAlign;
uint32_t ScanConvMode;
uint32_t EOCSelection;
FunctionalState LowPowerAutoWait;
FunctionalState ContinuousConvMode;
FunctionalState DiscontinuousConvMode;
uint32_t ExternalTrigConv;
uint32_t ExternalTrigConvEdge;
FunctionalState DMAContinuousRequests;
uint32_t Overrun;
uint32_t SamplingTimeCommon;
}ADC_InitTypeDef;
```

字段说明：

表5-1 ADC_InitTypeDef 字段说明

字段	描述
ClockPrescaler	配置 ADC 时钟源(源自 APB 的同步时钟或源自 HSI RC 振荡器的异步时钟)和时钟分频系数
Resolution	配置ADC 分辨率

DataAlign	配置ADC 数据对齐方式
ScanConvMode	配置规则组扫描序列方向
EOCSelection	配置轮询/中断方式的结束/中断触发的标志
LowPowerAutoWait	配置动态低功耗自动延迟
ContinuousConvMode	配置连续转换模式
DiscontinuousConvMode	配置非连续转换模式
ExternalTrigConv	配置用于触发采样转换开始的外部事件
ExternalTrigConvEdge	配置触发采样的外部触发边沿
DMAContinuousRequests	配置DMA 循环模式
Overrun	配置过载管理模式
SamplingTimeCommon	配置所选扫描序列的采样时间

注意：连续转换模式（ContinuousConvMode）和非连续转换模式（DiscontinuousConvMode）不能同时开启（ENABLE），如果都不开启则为单次模式。

参数说明：

ClockPrescaler 可选参数：

表5-2 ClockPrescaler 可选参数

参数	描述
ADC_CLOCK_SYNC_PCLK_DIV1	选择 APB 作为时钟源，不分频
ADC_CLOCK_SYNC_PCLK_DIV2	选择 APB 作为时钟源，2 分频
ADC_CLOCK_SYNC_PCLK_DIV4	选择 APB 作为时钟源，4 分频
ADC_CLOCK_SYNC_PCLK_DIV8	选择 APB 作为时钟源，8 分频
ADC_CLOCK_SYNC_PCLK_DIV16	选择 APB 作为时钟源，16 分频
ADC_CLOCK_SYNC_PCLK_DIV32	选择 APB 作为时钟源，32 分频
ADC_CLOCK_SYNC_PCLK_DIV64	选择 APB 作为时钟源，64 分频
ADC_CLOCK_ASYNC_HSI_DIV1	选择 HSI 作为时钟源，不分频
ADC_CLOCK_ASYNC_HSI_DIV2	选择 HSI 作为时钟源，2 分频
ADC_CLOCK_ASYNC_HSI_DIV4	选择 HSI 作为时钟源，4 分频
ADC_CLOCK_ASYNC_HSI_DIV8	选择 HSI 作为时钟源，8 分频
ADC_CLOCK_ASYNC_HSI_DIV16	选择 HSI 作为时钟源，16 分频
ADC_CLOCK_ASYNC_HSI_DIV32	选择 HSI 作为时钟源，32 分频
ADC_CLOCK_ASYNC_HSI_DIV64	选择 HSI 作为时钟源，64 分频

Resolution 可选参数：

表5-3 Resolution 可选参数

参数	描述
ADC_RESOLUTION_12B	设置 12 位分辨率
ADC_RESOLUTION_10B	设置 10 位分辨率
ADC_RESOLUTION_8B	设置 8 位分辨率
ADC_RESOLUTION_6B	设置 6 位分辨率

DataAlign 可选参数：

表5-4 DataAlign 可选参数

参数	描述
ADC_DATAALIGN_RIGHT	数据右对齐
ADC_DATAALIGN_LEFT	数据左对齐

ScanConvMode 可选参数：

表5-5 ScanConvMode 可选参数

参数	描述
ADC_SCAN_DIRECTION_FORWARD	向上扫描，从通道 0 到 11
ADC_SCAN_DIRECTION_BACKWARD	向下扫描，从通道 11 到 0

EOCSelection 可选参数：

表5-6 EOCSelection 可选参数

参数	描述
ADC_EOC_SINGLE_CONV	将转换结束标志作为结束/中断标志位
ADC_EOC_SEQ_CONV	将序列结束标志作为结束/中断标志位

LowPowerAutoWait 可选参数：

表5-7 LowPowerAutoWait 可选参数

参数	描述
ENABLE	开启动态低功耗自动延迟
DISABLE	关闭动态低功耗自动延迟

ContinuousConvMode 可选参数：

表5-8 ContinuousConvMode 可选参数

参数	描述
ENABLE	开启连续转换模式
DISABLE	关闭连续转换模式

DiscontinuousConvMode 可选参数：

表5-9 DiscontinuousConvMode 可选参数

参数	描述
ENABLE	开启非连续转换模式
DISABLE	关闭非连续转换模式

ExternalTrigConv 可选参数：

表5-10 ExternalTrigConv 可选参数

参数	描述
ADC_EXTERNALTRIGCONV_T1_TRGO	选择外部事件 TIM1_TRGO 触发转换启动
ADC_EXTERNALTRIGCONV_T1_CC4	选择外部事件 TIM1_CC4 触发转换启动
ADC_EXTERNALTRIGCONV_T3_TRGO	选择外部事件 TIM3_TRGO 触发转换启动
ADC_SOFTWARE_START	选择软件触发

ExternalTrigConvEdge 可选参数：

表5-11 ExternalTrigConvEdge 可选参数

参数	描述
ADC_EXTERNALTRIGCONVEDGE_NONE	禁用硬件触发检测（可以通过软件开始转换）
ADC_EXTERNALTRIGCONVEDGE_RISING	开启硬件上升沿检测
ADC_EXTERNALTRIGCONVEDGE_FALLING	开启硬件下降沿检测
ADC_EXTERNALTRIGCONVEDGE_RISINGFALLING	开启硬件上升沿和下降沿检测

DMAContinuousRequests 可选参数：

表5-12 DMAContinuousRequests 可选参数

参数	描述
ENABLE	开启 DMA 循环模式
DISABLE	关闭 DMA 循环模式

Overrun 可选参数：

表5-13 Overrun 可选参数

参数	描述
ADC_OVR_DATA_OVERWRITTEN	使用新的转换数据覆盖原有数据
ADC_OVR_DATA_PRESERVED	保持原有数据，丢弃新的转换数据

SamplingTimeCommon 可选参数：

表5-14 SamplingTimeCommon 可选参数

参数	描述
ADC_SAMPLETIME_3CYCLES_5	采样时间为 3.5 个 ADC 时钟周期
ADC_SAMPLETIME_5CYCLES_5	采样时间为 5.5 个 ADC 时钟周期

ADC_SAMPLETIME_7CYCLES_5	采样时间为 7.5 个 ADC 时钟周期
ADC_SAMPLETIME_13CYCLES_5	采样时间为 13.5 个 ADC 时钟周期
ADC_SAMPLETIME_28CYCLES_5	采样时间为 28.5 个 ADC 时钟周期
ADC_SAMPLETIME_41CYCLES_5	采样时间为 41.5 个 ADC 时钟周期
ADC_SAMPLETIME_71CYCLES_5	采样时间为 71.5 个 ADC 时钟周期
ADC_SAMPLETIME_239CYCLES_5	采样时间为 239.5 个 ADC 时钟周期

5.1.2 ADC_ChannelConfTypeDef

ADC_ChannelConfTypeDef，定义于文件“air001xx_hal_adc.h”如下：

```
typedef struct
{
uint32_t Channel;
uint32_t Rank;
uint32_t SamplingTime;
}ADC_ChannelConfTypeDef;
```

字段说明：

表5-15 ADC_ChannelConfTypeDef 字段说明

字段	描述
Channel	指定要配置到 ADC 规则组中的通道
Rank	配置该通道是否开启
SamplingTime	配置所选通道采样时间

参数说明：

Channel 可选参数：

表5-16 Channel 可选参数

参数	描述
ADC_CHANNEL_0	通道 0
ADC_CHANNEL_1	通道 1
ADC_CHANNEL_2	通道 2
ADC_CHANNEL_3	通道 3
ADC_CHANNEL_4	通道 4
ADC_CHANNEL_5	通道 5
ADC_CHANNEL_6	通道 6
ADC_CHANNEL_7	通道 7
ADC_CHANNEL_8	通道 8
ADC_CHANNEL_9	通道 9

ADC_CHANNEL_11	通道 11
ADC_CHANNEL_12	通道 12

Rank 可选参数：

表5-17 Rank 可选参数

参数	描述
ADC_RANK_CHANNEL_NUMBER	开启该通道
ADC_RANK_NONE	关闭该通道

SamplingTime 可选参数：

表5-18 SamplingTime 可选参数

参数	描述
ADC_SAMPLETIME_3CYCLES_5	采样时间为 3.5 个 ADC 时钟周期
ADC_SAMPLETIME_5CYCLES_5	采样时间为 5.5 个 ADC 时钟周期
ADC_SAMPLETIME_7CYCLES_5	采样时间为 7.5 个 ADC 时钟周期
ADC_SAMPLETIME_13CYCLES_5	采样时间为 13.5 个 ADC 时钟周期
ADC_SAMPLETIME_28CYCLES_5	采样时间为 28.5 个 ADC 时钟周期
ADC_SAMPLETIME_41CYCLES_5	采样时间为 41.5 个 ADC 时钟周期
ADC_SAMPLETIME_71CYCLES_5	采样时间为 71.5 个 ADC 时钟周期
ADC_SAMPLETIME_239CYCLES_5	采样时间为 239.5 个 ADC 时钟周期

5.1.3 ADC_AnalogWDGConfTypeDef

ADC_AnalogWDGConfTypeDef，定义于文件“air001xx_hal_adc.h”如下：

```
typedef struct
{
uint32_t WatchdogMode;
uint32_t Channel;
FunctionalState ITMode;
uint32_t HighThreshold;
uint32_t LowThreshold;
}ADC_AnalogWDGConfTypeDef;
```

字段说明：

表5-19 ADC_AnalogWDGConfTypeDef 字段说明

字段	描述
WatchdogMode	配置ADC 模拟看门狗模式：单一通道/所有通道/无
Channel	配置ADC 模拟看门狗监控的通道
ITMode	配置ADC 模拟看门狗中断

HighThreshold	配置ADC 模拟看门狗的阈值高位
LowThreshold	配置ADC 模拟看门狗的阈值低位

参数说明：

WatchdogMode 可选参数：

表5-20 WatchdogMode 可选参数

参数	描述
ADC_ANALOGWATCHDOG_NONE	不开启模拟看门狗
ADC_ANALOGWATCHDOG_SINGLE_REG	在一个通道上开启模拟看门狗
ADC_ANALOGWATCHDOG_ALL_REG	在所有通道上开启模拟看门狗

Channel 可选参数：

表5-21 Channel 可选参数

参数	描述
ADC_CHANNEL_0	通道 0
ADC_CHANNEL_1	通道 1
ADC_CHANNEL_2	通道 2
ADC_CHANNEL_3	通道 3
ADC_CHANNEL_4	通道 4
ADC_CHANNEL_5	通道 5
ADC_CHANNEL_6	通道 6
ADC_CHANNEL_7	通道 7
ADC_CHANNEL_8	通道 8
ADC_CHANNEL_9	通道 9
ADC_CHANNEL_11	通道 11
ADC_CHANNEL_12	通道 12

ITMode 可选参数：

表5-22 ITMode 可选参数

参数	描述
ENABLE	开启模拟看门狗中断
DISABLE	关闭模拟看门狗中断

HighThreshold 可选参数：

表5-23 HighThreshold 可选参数

参数	描述
0x000~0xFFF/0x3FF/0xFF/0x3F	模拟看门狗阈值高位可选范围

	当 ADC 分辨率为 12 位时：0x000~0xFFF 当 ADC 分辨率为 10 位时：0x000~0x3FF 当 ADC 分辨率为 8 位时：0x000~0xFF 当 ADC 分辨率为 6 位时：0x000~0x3F

LowThreshold 可选参数：

表5-24 LowThreshold 可选参数

参数	描述
0x000~0xFFF/0x3FF/0xFF/0x3F	模拟看门狗阈值低位可选范围（同阈值高位）

5.1.4 ADC_HandleTypeDef

ADC_HandleTypeDef，定义于文件“air001xx_hal_adc.h”如下：

```
typedef struct __ADC_HandleTypeDef
{
ADC_TypeDef *Instance;
ADC_InitTypeDef Init;
DMA_HandleTypeDef *DMA_Handle;
HAL_LockTypeDef Lock;
__IO uint32_t State;
__IO uint32_t ErrorCode;
}ADC_HandleTypeDef;
```

字段说明：

表5-25 ADC_HandleTypeDef 字段说明

字段	描述
Instance	外设寄存器基地址
Init	外设初始化参数结构体指针
DMA_Handle	DMA 句柄指针
Lock	HAL 锁状态
State	外设状态信息
ErrorCode	错误代码

5.2 ADC 固件库函数

表5-26 ADC 固件库函数说明

函数名	描述
HAL_ADC_Init	初始化 ADC
HAL_ADC_DeInit	将 ADC 配置设为缺省值

HAL_ADC_MspInit	初始化与ADC 相关的 MSP
HAL_ADC_MspDeInit	将与ADC 相关的 MSP 设置为缺省值
HAL_ADC_Start	开启ADC，使用轮询模式开始规则组转换
HAL_ADC_Stop	停止转换序列，关闭 ADC
HAL_ADC_PollForConversion	等待转换序列完成
HAL_ADC_PollForEvent	轮询方式获取转换事件
HAL_ADC_Start_IT	开启ADC 并开启中断
HAL_ADC_Stop_IT	停止规则组转换，关闭中断，关闭ADC
HAL_ADC_Start_DMA	开启ADC，开始规则组转换，开启DMA 请求
HAL_ADC_Stop_DMA	停止规则组转换，关闭DMA 请求，关闭ADC
HAL_ADC_GetValue	获取ADC 转换结束的结果
HAL_ADC_IRQHandler	处理ADC 中断请求
HAL_ADC_ConvCpltCallback	DMA 模式转换完成回调函数
HAL_ADC_ConvHalfCpltCallback	DMA 模式 DMA 半传输完成回调函数
HAL_ADC_LevelOutOfWindowCallback	中断模式模拟看门狗回调函数
HAL_ADC_ErrorCallback	ADC 错误回调函数
HAL_ADC_Calibration_Start	ADC 自动校准
HAL_ADC_ConfigChannel	配置所选通道到 ADC 规则组
HAL_ADC_AnalogWDGConfig	配置模拟看门狗
HAL_ADC_GetState	获取ADC 状态
HAL_ADC_GetError	获取ADC 错误代码

5.2.1 函数 HAL_ADC_Init

描述了函数 HAL_ADC_Init

表5-27 函数 HAL_ADC_Init

函数名	HAL_ADC_Init
函数原形	HAL_StatusTypeDef HAL_ADC_Init (ADC_HandleTypeDef * hadc)
功能描述	初始化 ADC
输入参数	hadc：ADC 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

5.2.2 函数 HAL_ADC_DeInit

描述了函数 HAL_ADC_DeInit

表5-28 函数 HAL_ADC_DeInit

函数名	HAL_ADC_DeInit
函数原形	HAL_StatusTypeDef HAL_ADC_DeInit (ADC_HandleTypeDef * hadc)
功能描述	将 ADC 配置设为缺省值
输入参数	hadc：ADC 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

5.2.3 函数 HAL_ADC_MspInit

描述了函数 HAL_ADC_MspInit

表5-29 函数 HAL_ADC_MspInit

函数名	HAL_ADC_MspInit
函数原形	void HAL_ADC_MspInit (ADC_HandleTypeDef * hadc)
功能描述	初始化与 ADC 相关的 MSP
输入参数	hadc：ADC 句柄
输出参数	无
返回值	无
先决条件	无

5.2.4 函数 HAL_ADC_MspDeInit

描述了函数 HAL_ADC_MspDeInit

表5-30 函数 HAL_ADC_MspDeInit

函数名	HAL_ADC_MspDeInit
函数原形	void HAL_ADC_MspDeInit (ADC_HandleTypeDef * hadc)
功能描述	将与 ADC 相关的 MSP 设为缺省值
输入参数	hadc：ADC 句柄
输出参数	无
返回值	无
先决条件	无

5.2.5 函数 HAL_ADC_Start

描述了函数 HAL_ADC_Start

表5-31 函数 HAL_ADC_Start

函数名	HAL_ADC_Start
函数原形	HAL_StatusTypeDef HAL_ADC_Start (ADC_HandleTypeDef * hadc)
功能描述	开启 ADC，使用轮询模式开始规则组转换
输入参数	hadc：ADC 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

5.2.6 函数 HAL_ADC_Stop

描述了函数 HAL_ADC_Stop

表5-32 函数 HAL_ADC_Stop

函数名	HAL_ADC_Stop
函数原形	HAL_StatusTypeDef HAL_ADC_Stop (ADC_HandleTypeDef * hadc)
功能描述	停止 ADC 规则组转换，关闭 ADC
输入参数	hadc：ADC 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

5.2.7 函数 HAL_ADC_PollForConversion

描述了函数 HAL_ADC_PollForConversion

表5-33 函数 HAL_ADC_PollForConversion

函数名	HAL_ADC_PollForConversion
函数原形	HAL_StatusTypeDef HAL_ADC_PollForConversion (ADC_HandleTypeDef * hadc, uint32_t Timeout)
功能描述	等待转换序列完成
输入参数 1	hadc：ADC 句柄
输入参数 2	Timeout：等待超时值，单位：毫秒
输出参数	无
返回值	HAL 状态
先决条件	无

5.2.8 函数 HAL_ADC_PollForEvent

描述了函数 HAL_ADC_PollForEvent

表5-34 函数 HAL_ADC_PollForEvent

函数名	HAL_ADC_PollForEvent

函数原形	HAL_StatusTypeDef HAL_ADC_PollForEvent (ADC_HandleTypeDef * hadc, uint32_t EventType, uint32_t Timeout)
功能描述	轮询方式查询转换事件
输入参数 1	hadc：ADC 句柄
输入参数 2	EventType：ADC 事件类型
输入参数 3	Timeout：等待超时时间，单位：毫秒
输出参数	无
返回值	HAL 状态
先决条件	无

5.2.9 函数 HAL_ADC_Start_IT

描述了函数 HAL_ADC_Start_IT

表5-35 函数 HAL_ADC_Start_IT

函数名	HAL_ADC_Start_IT
函数原形	HAL_StatusTypeDef HAL_ADC_Start_IT (ADC_HandleTypeDef * hadc)
功能描述	开启 ADC 并开启中断
输入参数	hadc：ADC 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

5.2.10 函数 HAL_ADC_Stop_IT

描述了函数 HAL_ADC_Stop_IT

表5-36 函数 HAL_ADC_Stop_IT

函数名	HAL_ADC_Stop_IT
函数原形	HAL_StatusTypeDef HAL_ADC_Stop_IT (ADC_HandleTypeDef * hadc)
功能描述	停止规则组转换，关闭中断，关闭 ADC
输入参数	hadc：ADC 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

5.2.11 函数 HAL_ADC_Start_DMA

描述了函数 HAL_ADC_Start_DMA

表5-37 函数 HAL_ADC_Start_DMA

函数名	HAL_ADC_Start_DMA
函数原形	HAL_StatusTypeDef HAL_ADC_Start_DMA (ADC_HandleTypeDef * hadc, uint32_t * pData, uint32_t Length)

功能描述	开启 ADC，开始规则组转换，开启 DMA 请求
输入参数 1	hadc：ADC 句柄
输入参数 2	pData：数据存储缓冲区
输入参数 3	Length：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

5.2.12 函数 HAL_ADC_Stop_DMA

描述了函数 HAL_ADC_Stop_DMA

表5-38 函数 HAL_ADC_Stop_DMA

函数名	HAL_ADC_Stop_DMA
函数原形	HAL_StatusTypeDef HAL_ADC_Stop_DMA (ADC_HandleTypeDef * hadc)
功能描述	停止规则组转换，关闭 DMA 请求，关闭 ADC
输入参数	hadc：ADC 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

5.2.13 函数 HAL_ADC_GetValue

描述了函数 HAL_ADC_GetValue

表5-39 函数 HAL_ADC_GetValue

函数名	HAL_ADC_GetValue
函数原形	uint32_t HAL_ADC_GetValue (ADC_HandleTypeDef * hadc)
功能描述	获取 ADC 规则组的转换结果
输入参数	hadc：ADC 句柄
输出参数	无
返回值	转换结果数据
先决条件	无

5.2.14 函数 HAL_ADC_IRQHandler

描述了函数 HAL_ADC_IRQHandler

表5-40 函数 HAL_ADC_IRQHandler

函数名	HAL_ADC_IRQHandler
函数原形	void HAL_ADC_IRQHandler (ADC_HandleTypeDef * hadc)
功能描述	ADC 中断请求处理

输入参数	hadc：ADC 句柄
输出参数	无
返回值	无
先决条件	无

5.2.15 函数 HAL_ADC_ConvCpltCallback

描述了函数 HAL_ADC_ConvCpltCallback

表5-41 函数 HAL_ADC_ConvCpltCallback

函数名	HAL_ADC_ConvCpltCallback
函数原形	void HAL_ADC_ConvCpltCallback (ADC_HandleTypeDef * hadc)
功能描述	DMA 模式下转换完成的回调函数
输入参数	hadc：ADC 句柄
输出参数	无
返回值	无
先决条件	无

5.2.16 函数 HAL_ADC_ConvHalfCpltCallback

描述了函数 HAL_ADC_ConvHalfCpltCallback

表5-42 函数 HAL_ADC_ConvHalfCpltCallback

函数名	HAL_ADC_ConvHalfCpltCallback
函数原形	void HAL_ADC_ConvHalfCpltCallback (ADC_HandleTypeDef * hadc)
功能描述	DMA 模式下的 DMA 半传输完成回调函数
输入参数	hadc：ADC 句柄
输出参数	无
返回值	无
先决条件	无

5.2.17 函数 HAL_ADC_LevelOutOfWindowCallback

描述了函数 HAL_ADC_LevelOutOfWindowCallback

表5-43 函数 HAL_ADC_LevelOutOfWindowCallback

函数名	HAL_ADC_LevelOutOfWindowCallback
函数原形	void HAL_ADC_LevelOutOfWindowCallback (ADC_HandleTypeDef * hadc)
功能描述	模拟看门狗中断回调函数
输入参数	hadc：ADC 句柄
输出参数	无
返回值	无

先决条件	无

5.2.18 函数 HAL_ADC_ErrorCallback

描述了函数 HAL_ADC_ErrorCallback

表5-44 函数 HAL_ADC_ErrorCallback

函数名	HAL_ADC_ErrorCallback
函数原形	void HAL_ADC_ErrorCallback (ADC_HandleTypeDef * hadc)
功能描述	中断模式下 ADC 错误回调函数
输入参数	hadc：ADC 句柄
输出参数	无
返回值	无
先决条件	无

5.2.19 函数 HAL_ADC_Calibration_Start

描述了函数 HAL_ADC_Calibration_Start

表5-45 函数 HAL_ADC_Calibration_Start

函数名	HAL_ADC_Calibration_Start
函数原形	HAL_StatusTypeDef　HAL_ADC_Calibration_Start(ADC_HandleTypeDef*　hadc)
功能描述	ADC 自动校准
输入参数 1	hadc：ADC 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

5.2.20 函数 HAL_ADC_ConfigChannel

描述了函数 HAL_ADC_ConfigChannel

表5-46 函数 HAL_ADC_ConfigChannel

函数名	HAL_ADC_ConfigChannel
函数原形	HAL_StatusTypeDef HAL_ADC_ConfigChannel (ADC_HandleTypeDef * hadc,ADC_ChannelConfTypeDef * sConfig)
功能描述	配置选择的通道
输入参数 1	hadc：ADC 句柄
输入参数 2	sConfig：ADC 通道配置结构体
输出参数	无
返回值	HAL 状态
先决条件	无

5.2.21 函数 HAL_ADC_AnalogWDGConfig

描述了函数 HAL_ADC_AnalogWDGConfig

表5-47 函数 HAL_ADC_AnalogWDGConfig

函数名	HAL_ADC_AnalogWDGConfig
函数原形	HAL_StatusTypeDef HAL_ADC_AnalogWDGConfig (ADC_HandleTypeDef * hadc, ADC_AnalogWDGConfTypeDef * AnalogWDGConfig)
功能描述	配置模拟看门狗
输入参数 1	hadc：ADC 句柄
输入参数 2	AnalogWDGConfig：模拟看门狗配置结构体
输出参数	无
返回值	HAL 状态
先决条件	无

5.2.22 函数 HAL_ADC_GetState

描述了函数 HAL_ADC_GetState

表5-48 函数 HAL_ADC_GetState

函数名	HAL_ADC_GetState
函数原形	uint32_t HAL_ADC_GetState (ADC_HandleTypeDef * hadc)
功能描述	获取 ADC 状态
输入参数	hadc：ADC 句柄
输出参数	无
返回值	ADC 状态
先决条件	无

5.2.23 函数 HAL_ADC_GetError

描述了函数 HAL_ADC_GetError

表5-49 函数 HAL_ADC_GetError

函数名	HAL_ADC_GetError
函数原形	uint32_t HAL_ADC_GetError (ADC_HandleTypeDef * hadc)
功能描述	获取 ADC 错误代码
输入参数	hadc：ADC 句柄
输出参数	无
返回值	错误代码
先决条件	无

6 HAL 比较器通用驱动程序（COMP）

芯片内集成了 2 个通用比较器（general purpose comparators）COMP，分别是 COMP1 和 COMP2。这两个模块可以作为单独的模块，也可以与TIM 组合在一起使用。

比较器可以实现如下功能：

- 被模拟信号触发，产生低功耗模式唤醒功能
- 模拟信号调节
- 当与来自TIM 的 PWM 输出连接时，构成逐周期电流控制环路。

6.1 比较器固件驱动寄存器结构

6.1.1 COMP_InitTypeDef

COMP_InitTypeDef，定义于文件“air001xx_hal_comp.h”如下：

```
typedef struct
{
uint32_t WindowMode;
uint32_t Mode;
uint32_t InputPlus;
uint32_t InputMinus;
uint32_t Hysteresis;
uint32_t OutputPol;
uint32_t DigitalFilter;
uint32_t TriggerMode;
} COMP_InitTypeDef;
```

字段说明：

表6-1 COMP_InitTypeDef 字段说明

字段	描述
WindowMode	设置比较器 1 和比较器 2 的窗口模式
Mode	设置比较器工作模式
InputPlus	设置比较器正相输入
InputMinus	设置比较器反相输入
Hysteresis	设置反相输入比较器的迟滞模式
OutputPol	设置比较器输出极性
DigitalFilter	设置比较器数字滤波器
TriggerMode	设置比较器输出外部中断/事件的触发边沿

参数说明：

WindowMode 可选参数：

表6-2 WindowMode 可选参数

参数	描述
COMP_WINDOWMODE_DISABLE	关闭窗口模式
COMP_WINDOWMODE_COMP1_INPUT_PLUS_COMMON	COMP1 正相输入作为检测输入端口
COMP_WINDOWMODE_COMP2_INPUT_PLUS_COMMON	COMP2 正相输入作为检测输入端口

Mode 可选参数：

表6-3 Mode 可选参数

参数	描述
COMP_POWERMODE_HIGHSPEED	高速模式
COMP_POWERMODE_MEDIUMSPEED	中等速度模式

InputPlus 可选参数：

表6-4 InputPlus 可选参数

参数	描述
COMP_INPUT_PLUS_IO1	选择 COMP 输入引脚 IO1（COMP1：PB8，COMP2：PB4）
COMP_INPUT_PLUS_IO2	选择 COMP 输入引脚 IO2（COMP1：PB2，COMP2：PB6）
COMP_INPUT_PLUS_IO3	选择 COMP 输入引脚 IO3（COMP1：PA1，COMP2：PA3）
COMP_INPUT_PLUS_IO4	选择 COMP 输入引脚 IO4（COMP1：无，COMP2：PF3）

InputMinus 可选参数：

表6-5 InputMinus 可选参数

参数	描述
COMP_INPUT_MINUS_1_4VREFINT	选择反相输入电压为 1/4VREFINT
COMP_INPUT_MINUS_1_2VREFINT	选择反相输入电压为 1/2VREFINT
COMP_INPUT_MINUS_3_4VREFINT	选择反相输入电压为 3/4VREFINT
COMP_INPUT_MINUS_VREFINT	选择反相输入电压为 VREFINT
COMP_INPUT_MINUS_VCC	选择反相输入电压为 VCC
COMP_INPUT_MINUS_TS	选择反相输入电压为温度传感器输出电压
COMP_INPUT_MINUS_IO1	选择反相输入电压为 IO1（COMP1：PB1，COMP2：PB3
COMP_INPUT_MINUS_IO2	选择反相输入电压为 IO2（COMP1：无，COMP2：PB7）
COMP_INPUT_MINUS_IO3	选择反相输入电压为 IO3（COMP1：PA0，COMP2：PA2

Hysteresis 可选参数：

表6-6 Hysteresis 可选参数

参数	描述
COMP_HYSTERESIS_DISABLE	关闭比较器迟滞功能
COMP_HYSTERESIS_ENABLE	开启比较器迟滞功能

OutputPol 可选参数：

表6-7 OutputPol 可选参数

参数	描述
COMP_OUTPUTPOL_NONINVERTED	关闭比较器输出反向
COMP_OUTPUTPOL_INVERTED	开启比较器输出反向

DigitalFilter 可选参数：

表6-8 DigitalFilter 可选参数

参数	描述
0~0xFFFF	滤波响应时间，单位：机器周期

TriggerMode 可选参数：

表6-9 TriggerMode 可选参数

参数	描述
COMP_TRIGGERMODE_NONE	不开启 COMP 中断
COMP_TRIGGERMODE_IT_RISING	开启 COMP 上升沿触发中断
COMP_TRIGGERMODE_IT_FALLING	开启 COMP 下降沿触发中断
COMP_TRIGGERMODE_IT_RISING_FALLING	开启 COMP 上升沿和下降沿触发中断
COMP_TRIGGERMODE_EVENT_RISING	开启 COMP 上升沿触发事件
COMP_TRIGGERMODE_EVENT_FALLING	开启 COMP 下降沿触发事件
COMP_TRIGGERMODE_EVENT_RISING_FALLING	开启 COMP 上升沿和下降沿触发事件

6.1.2 COMP_HandleTypeDef

COMP_HandleTypeDef，定义于文件“air001xx_hal_comp.h”如下：

```
typedef struct
{
COMP_TypeDef *Instance;
COMP_InitTypeDef Init;
HAL_LockTypeDef Lock;
__IO HAL_COMP_StateTypeDef State;
__IO uint32_t ErrorCode;
} COMP_HandleTypeDef;
```

字段说明：

表6-10 COMP_HandleTypeDef 字段说明

字段	描述
Instance	外设寄存器基地址
Init	初始化参数结构体指针
Lock	HAL 锁
State	COMP 状态
ErrorCode	错误代码

6.2 COMP 固件库函数

表6-11 COMP 固件库函数说明

函数名	描述
HAL_COMP_Init	初始化 COMP
HAL_COMP_DeInit	将 COMP 参数设为缺省值
HAL_COMP_MspInit	初始化 COMP 相关的 MSP
HAL_COMP_MspDeInit	将 COMP 相关的 MSP 设置为缺省值
HAL_COMP_Start	启动COMP
HAL_COMP_Stop	停止COMP
HAL_COMP_IRQHandler	COMP 中断请求处理
HAL_COMP_Lock	锁定选 COMP 配置
HAL_COMP_GetOutputLevel	获取COMP 输出的电平
HAL_COMP_TriggerCallback	触发回调函数
HAL_COMP_GetState	获取COMP 状态信息
HAL_COMP_GetError	获取错误代码

6.2.1 函数 HAL_COMP_Init

描述了函数 HAL_COMP_Init

表6-12 函数 HAL_COMP_Init

函数名	HAL_COMP_Init
函数原形	HAL_StatusTypeDef HAL_COMP_Init(COMP_HandleTypeDef *hcomp)
功能描述	初始化 COMP
输入参数	hcomp：COMP 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

6.2.2 函数 HAL_COMP_DeInit

描述了函数 HAL_COMP_DeInit

表6-13 函数 HAL_COMP_DeInit

函数名	HAL_COMP_DeInit
函数原形	HAL_StatusTypeDef HAL_COMP_DeInit(COMP_HandleTypeDef *hcomp)
功能描述	将 COMP 参数设为缺省值
输入参数	hcomp：COMP 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

6.2.3 函数 HAL_COMP_MspInit

描述了函数 HAL_COMP_MspInit

表6-14 函数 HAL_COMP_MspInit

函数名	HAL_COMP_MspInit
函数原形	void HAL_COMP_MspInit(COMP_HandleTypeDef *hcomp)
功能描述	初始化 COMP 相关的 MSP
输入参数	hcomp：COMP 句柄
输出参数	无
返回值	无
先决条件	无

6.2.4 函数 HAL_COMP_MspDeInit

描述了函数 HAL_COMP_MspDeInit

表6-15 函数 HAL_COMP_MspDeInit

函数名	HAL_COMP_MspDeInit
函数原形	void HAL_COMP_MspDeInit(COMP_HandleTypeDef *hcomp)
功能描述	将 COMP 相关的 MSP 设置为缺省值
输入参数	hcomp：COMP 句柄
输出参数	无
返回值	无
先决条件	无

6.2.5 函数 HAL_COMP_Start

描述了函数 HAL_COMP_Start

表6-16 函数 HAL_COMP_Start

函数名	HAL_COMP_Start
函数原形	HAL_StatusTypeDef HAL_COMP_Start(COMP_HandleTypeDef *hcomp)
功能描述	启动 COMP
输入参数	hcomp：COMP 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

6.2.6 函数 HAL_COMP_Stop

描述了函数 HAL_COMP_Stop

表6-17 函数 HAL_COMP_Stop

函数名	HAL_COMP_Stop
函数原形	HAL_StatusTypeDef HAL_COMP_Stop(COMP_HandleTypeDef *hcomp)
功能描述	停止 COMP
输入参数	hcomp：COMP 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

6.2.7 函数 HAL_COMP_IRQHandler

描述了函数 HAL_COMP_IRQHandler

表6-18 函数 HAL_COMP_IRQHandler

函数名	HAL_COMP_IRQHandler
函数原形	void HAL_COMP_IRQHandler(COMP_HandleTypeDef *hcomp)
功能描述	COMP 中断请求处理
输入参数	hcomp：COMP 句柄
输出参数	无
返回值	无
先决条件	无

6.2.8 函数 HAL_COMP_Lock

描述了函数 HAL_COMP_Lock

表6-19 函数 HAL_COMP_Lock

函数名	HAL_COMP_Lock
函数原形	HAL_StatusTypeDef HAL_COMP_Lock(COMP_HandleTypeDef *hcomp)

功能描述	锁定 COMP 配置
输入参数	hcomp：COMP 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

6.2.9 函数 HAL_COMP_GetOutputLevel

描述了函数 HAL_COMP_GetOutputLevel

表6-20 函数 HAL_COMP_GetOutputLevel

函数名	HAL_COMP_GetOutputLevel
函数原形	uint32_t HAL_COMP_GetOutputLevel(COMP_HandleTypeDef *hcomp)
功能描述	获取 COMP 输出的电平
输入参数	hcomp：COMP 句柄
输出参数	无
返回值	电平状态（高 1 或低 0）
先决条件	无

6.2.10 函数 HAL_COMP_TriggerCallback

描述了函数 HAL_COMP_TriggerCallback

表6-21 函数 HAL_COMP_TriggerCallback

函数名	HAL_COMP_TriggerCallback
函数原形	void HAL_COMP_TriggerCallback(COMP_HandleTypeDef *hcomp)
功能描述	触发回调函数
输入参数	hcomp：COMP 句柄
输出参数	无
返回值	无
先决条件	无

6.2.11 函数 HAL_COMP_GetState

描述了函数 HAL_COMP_GetState

表6-22 函数 HAL_COMP_GetState

函数名	HAL_COMP_GetState
函数原形	HAL_COMP_StateTypeDef HAL_COMP_GetState(COMP_HandleTypeDef *hcomp)
功能描述	获取 COMP 状态信息
输入参数	hcomp：COMP 句柄
输出参数	无

返回值	COMP 状态
先决条件	无

6.2.12 函数 HAL_COMP_GetError

描述了函数 HAL_COMP_GetError

表6-23 函数 HAL_COMP_GetError

函数名	HAL_COMP_GetError
函数原形	uint32_t HAL_COMP_GetError(COMP_HandleTypeDef *hcomp)
功能描述	获取错误代码
输入参数	hcomp：COMP 句柄
输出参数	无
返回值	错误代码
先决条件	无

7 HAL Cortex 通用驱动程序（CORTEX）

Cortex 包含对SysTick 定时器、SCB 系统控制的配置，以及获取CPU ID。

7.1 Cortex 固件库函数

表7-1 Cortex 固件库函数说明

函数名	描述
HAL_NVIC_SetPriority	设置中断优先级
HAL_NVIC_EnableIRQ	开启NVIC 中断控制器中特定设备的中断请求
HAL_NVIC_DisableIRQ	关闭NVIC 中断控制器中特定设备的中断请求
HAL_NVIC_SystemReset	发起系统复位请求以复位MCU
HAL_SYSTICK_Config	初始化 SysTick 及其中断并启动SysTick
HAL_NVIC_GetPriority	获取中断优先级
HAL_NVIC_GetPendingIRQ	检查指定外部中断是否挂起
HAL_NVIC_SetPendingIRQ	置位指定外部中断挂起位
HAL_NVIC_ClearPendingIRQ	清除指定外部中断挂起位
HAL_SYSTICK_CLKSourceConfig	配置SysTick 时钟源
HAL_SYSTICK_IRQHandler	处理SysTick 中断请求
HAL_SYSTICK_Callback	SysTick 中断回调函数

7.1.1 函数 HAL_NVIC_SetPriority

描述了函数 HAL_NVIC_SetPriority

表7-2 函数 HAL_NVIC_SetPriority

函数名	HAL_NVIC_SetPriority
函数原形	void HAL_NVIC_SetPriority(IRQn_Type IRQn,uint32_t PreemptPriority, uint32_t SubPriority)
功能描述	设置指定外部中断的抢占优先级和相应优先级
输入参数 1	IRQn：外设中断序号
输入参数 2	PreemptPriority：抢占优先级
输入参数 3	SubPriority：相应优先级
输出参数	无
返回值	无
先决条件	无

参数说明:

IRQn 可选参数:

表7-3 IRQn 可选参数

参数	描述
WWDG_IRQn	看门狗中断
PVD_IRQn	PVD 中断
RTC_IRQn	RTC 中断
FLASH_IRQn	FLASH 中断
RCC_IRQn	RCC 中断
EXTI0_1_IRQn	外部中断线 0~1 中断
EXTI2_3_IRQn	外部中断线 2~3 中断
EXTI4_15_IRQn	外部中断线 4~15 中断
DMA1_Channel1_IRQn	DMA 通道 1 中断
DMA1_Channel2_3_IRQn	DMA 通道 2~3 中断
ADC_COMP_IRQn	ADC 和 COMP 中断
TIM1_BRK_UP_TRG_COM_IRQn	TIM1 刹车、更新、触发、换向中断
TIM1_CC_IRQn	TIM1 捕获/比较中断
TIM3_IRQn	TIM3 全局中断
LPTIM1_IRQn	LPTIM1 全局中断
TIM14_IRQn	TIM14 全局中断
TIM16_IRQn	TIM16 全局中断
TIM17_IRQn	TIM17 全局中断
I2C1_IRQn	I2C 中断
SPI1_IRQn	SPI1 中断
SPI2_IRQn	SPI2 中断
USART1_IRQn	USART1 中断
USART2_IRQn	USART2 中断
LED_IRQn	LED 全局中断

7.1.2 函数 HAL_NVIC_EnableIRQ

描述了函数 HAL_NVIC_EnableIRQ

表7-4 函数 HAL_NVIC_EnableIRQ

函数名	HAL_NVIC_EnableIRQ
函数原形	void HAL_NVIC_EnableIRQ(IRQn_Type IRQn)
功能描述	开启 NVIC 中断控制器中特定设备的中断请求
输入参数	IRQn：外设中断序号
输出参数	无

返回值	无
先决条件	无

7.1.3 函数 HAL_NVIC_DisableIRQ

描述了函数 HAL_NVIC_DisableIRQ

表7-5 函数 HAL_NVIC_DisableIRQ

函数名	HAL_NVIC_DisableIRQ
函数原形	void HAL_NVIC_DisableIRQ(IRQn_Type IRQn)
功能描述	关闭 NVIC 中断控制器中特定设备的中断请求
输入参数	IRQn：外设中断序号
输出参数	无
返回值	无
先决条件	无

7.1.4 函数 HAL_NVIC_SystemReset

描述了函数 HAL_NVIC_SystemReset

表7-6 函数 HAL_NVIC_SystemReset

函数名	HAL_NVIC_SystemReset
函数原形	void HAL_NVIC_SystemReset(void)
功能描述	发起系统复位请求，复位 MCU
输入参数	无
输出参数	无
返回值	无
先决条件	无

7.1.5 函数 HAL_SYSTICK_Config

描述了函数 HAL_SYSTICK_Config

表7-7 函数 HAL_SYSTICK_Config

函数名	HAL_SYSTICK_Config
函数原形	uint32_t HAL_SYSTICK_Config(uint32_t TicksNumb)
功能描述	初始化 SysTick 及其中断并启动 Systick
输入参数	TicksNumb：SysTick 产生两次中断间的计数值
输出参数	无
返回值	函数执行结果（成功 0 或失败 1）
先决条件	无

7.1.6 函数 HAL_NVIC_GetPriority

描述了函数 HAL_NVIC_GetPriority

表7-8 函数 HAL_NVIC_GetPriority

函数名	HAL_NVIC_GetPriority
函数原形	uint32_t HAL_NVIC_GetPriority(IRQn_Type IRQn)
功能描述	获取指定外设的中断优先级
输入参数	IRQn：外设中断序号
输出参数	无
返回值	中断优先级
先决条件	无

7.1.7 函数 HAL_NVIC_GetPendingIRQ

描述了函数 HAL_NVIC_GetPendingIRQ

表7-9 函数 HAL_NVIC_GetPendingIRQ

函数名	HAL_NVIC_GetPendingIRQ
函数原形	uint32_t HAL_NVIC_GetPendingIRQ(IRQn_Type IRQn)
功能描述	检查指定外部中断是否挂起
输入参数	IRQn：外设中断序号
输出参数	无
返回值	中断被挂起（1）/中断未挂起（0）
先决条件	无

7.1.8 函数 HAL_NVIC_SetPendingIRQ

描述了函数 HAL_NVIC_SetPendingIRQ

表7-10 函数 HAL_NVIC_SetPendingIRQ

函数名	HAL_NVIC_SetPendingIRQ
函数原形	void HAL_NVIC_SetPendingIRQ(IRQn_Type IRQn)
功能描述	置位指定外部中断的挂起标志位
输入参数	IRQn：外设中断序号
输出参数	无
返回值	无
先决条件	无

7.1.9 函数 HAL_NVIC_ClearPendingIRQ

描述了函数 HAL_NVIC_ClearPendingIRQ

表7-11 函数 HAL_NVIC_ClearPendingIRQ

函数名	HAL_NVIC_ClearPendingIRQ
函数原形	void HAL_NVIC_ClearPendingIRQ(IRQn_Type IRQn)
功能描述	清除指定外部中断的挂起标志位
输入参数	IRQn：外设中断序号
输出参数	无
返回值	无
先决条件	无

7.1.10 函数 HAL_SYSTICK_CLKSourceConfig

描述了函数 HAL_SYSTICK_CLKSourceConfig

表7-12 函数 HAL_SYSTICK_CLKSourceConfig

函数名	HAL_SYSTICK_CLKSourceConfig
函数原形	void HAL_SYSTICK_CLKSourceConfig(uint32_t CLKSource)
功能描述	设置 SysTick 时钟源
输入参数	CLKSource：Systick 时钟源
输出参数	无
返回值	无
先决条件	无

参数说明:

CLKSource 可选参数:

表7-13 CLKSource 可选参数

参数	描述
SYSTICK_CLKSOURCE_HCLK_DIV8	设置 SysTick 时钟源为 AHB 时钟 8 分频
SYSTICK_CLKSOURCE_HCLK	设置 SysTick 时钟源为 AHB 时钟

7.1.11 函数 HAL_SYSTICK_IRQHandler

描述了函数 HAL_SYSTICK_IRQHandler

表7-14 函数 HAL_SYSTICK_IRQHandler

函数名	HAL_SYSTICK_IRQHandler
函数原形	void HAL_SYSTICK_IRQHandler(void)
功能描述	处理 Systick 中断请求
输入参数	无
输出参数	无
返回值	无

先决条件	无

7.1.12 函数 HAL_SYSTICK_Callback

描述了函数 HAL_SYSTICK_Callback

表7-15 函数 HAL_SYSTICK_Callback

函数名	HAL_SYSTICK_Callback
函数原形	void HAL_SYSTICK_Callback(void)
功能描述	Systick 中断回调函数
输入参数	无
输出参数	无
返回值	无
先决条件	无

8 HAL 循环冗余校验通用驱动程序（CRC）

根据生成多项式，CRC 计算单元将输入的 32 位数据，运算产生一个 CRC 结果。

8.1 CRC 固件驱动寄存器结构

8.1.1 CRC_HandleTypeDef

CRC_HandleTypeDef，定义于文件“air001xx_hal_crc.h”如下：

```
typedef struct
{
CRC_TypeDef *Instance;
HAL_LockTypeDef Lock;
__IO HAL_CRC_StateTypeDef State;
} CRC_HandleTypeDef;
```

字段说明：

表8-1 CRC_HandleTypeDef 字段说明

字段	描述
Instance	外设基地址
Lock	外设锁状态
State	外设状态信息

8.2 CRC 固件库函数

表8-2 CRC 固件库函数说明

函数名	描述
HAL_CRC_Init	初始化 CRC
HAL_CRC_DeInit	将 CRC 参数设为缺省值
HAL_CRC_MspInit	初始化 CRC 相关的 MSP
HAL_CRC_MspDeInit	将 CRC 相关的 MSP 设为缺省值
HAL_CRC_Accumulate	将上一次计算的 CRC 值作为初始值，计算 32 位数据缓冲区的 CRC 值
HAL_CRC_Calculate	将重置值（0xFFFFFFFF）作为初始值，计算 32 位数据缓冲区 CRC 值
HAL_CRC_GetState	获取 CRC 的状态信息

8.2.1 函数 HAL_CRC_Init

描述了函数 HAL_CRC_Init

表8-3 函数 HAL_CRC_Init

函数名	HAL_CRC_Init
函数原形	HAL_StatusTypeDef HAL_CRC_Init(CRC_HandleTypeDef *hcrc)
功能描述	初始化 CRC
输入参数	hcrc：CRC 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

8.2.2 函数 HAL_CRC_DeInit

描述了函数 HAL_CRC_DeInit

表8-4 函数 HAL_CRC_DeInit

函数名	HAL_CRC_DeInit
函数原形	HAL_StatusTypeDef HAL_CRC_DeInit(CRC_HandleTypeDef *hcrc)
功能描述	将 CRC 参数设为缺省值
输入参数	hcrc：CRC 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

8.2.3 函数 HAL_CRC_MspInit

描述了函数 HAL_CRC_MspInit

表8-5 函数 HAL_CRC_MspInit

函数名	HAL_CRC_MspInit
函数原形	void HAL_CRC_MspInit(CRC_HandleTypeDef *hcrc)
功能描述	初始化 CRC 相关的 MSP
输入参数	hcrc：CRC 句柄
输出参数	无
返回值	无
先决条件	无

8.2.4 函数 HAL_CRC_MspDeInit

描述了函数 HAL_CRC_MspDeInit

表8-6 函数 HAL_CRC_MspDeInit

函数名	HAL_CRC_MspDeInit
函数原形	void HAL_CRC_MspDeInit(CRC_HandleTypeDef *hcrc)

功能描述	将 CRC 相关的 MSP 设为缺省值
输入参数	hcrc：CRC 句柄
输出参数	无
返回值	无
先决条件	无

8.2.5 函数 HAL_CRC_Accumulate

描述了函数 HAL_CRC_Accumulate

表8-7 函数 HAL_CRC_Accumulate

函数名	HAL_CRC_Accumulate
函数原形	uint32_t HAL_CRC_Accumulate(CRC_HandleTypeDef *hcrc, uint32_t pBuffer[], uint32_t BufferLength)
功能描述	将上一次计算的 CRC 值作为初始值，计算 32 位数据缓冲区的 CRC 值
输入参数 1	hcrc：CRC 句柄
输入参数 2	pBuffer：数据缓冲区指针
输入参数 3	BufferLength：数据缓冲区数据长度
输出参数	无
返回值	CRC 计算结果值
先决条件	无

8.2.6 函数 HAL_CRC_Calculate

描述了函数 HAL_CRC_Calculate

表8-8 函数 HAL_CRC_Calculate

函数名	HAL_CRC_Calculate
函数原形	uint32_t HAL_CRC_Calculate(CRC_HandleTypeDef * hcrc, uint32_t pBuffer[], uint32_t BufferLength)
功能描述	将重置值（0xFFFFFFFF）作为初始值，计算 32 位数据缓冲区 CRC 值
输入参数 1	hcrc：CRC 句柄
输入参数 2	pBuffer：数据缓冲区指针
输入参数 3	BufferLength：数据缓冲区数据长度
输出参数	无
返回值	CRC 计算结果值
先决条件	无

8.2.7 函数 HAL_CRC_GetState

描述了函数 HAL_CRC_GetState

表8-9 函数 HAL_CRC_GetState

函数名	HAL_CRC_GetState

函数原形	HAL_CRC_StateTypeDef　HAL_CRC_GetState(CRC_HandleTypeDef　*hcrc)
功能描述	获取 CRC 的状态信息
输入参数	hcrc：CRC 句柄
输出参数	无
返回值	CRC 状态
先决条件	无

9 HAL DMA 控制器通用驱动程序（DMA）

直接存储器存取(DMA)用来提供在外设和存储器之间或者存储器和存储器之间的高速数据传输。无须CPU 干预，数据可以通过DMA 快速地移动，节省了CPU 的资源,进行其他操作。

DMA 控制器有 3 条 DMA 通道，每条通道负责管理来自 1 个或者多个外设对存储器访问的请求。DMA 控制器包括处理 DMA 请求的仲裁器，用于处理各个 DMA 请求的优先级。

9.1 DMA 固件驱动寄存器结构

9.1.1 DMA_InitTypeDef

DMA_InitTypeDef，定义于文件”air001xx_hal_dma.h”如下：

```
typedef struct
{
uint32_t  Direction;
uint32_t PeriphInc;
uint32_t MemInc;
uint32_t PeriphDataAlignment;
uint32_t MemDataAlignment;
uint32_t Mode;
uint32_t Priority;
} DMA_InitTypeDef;
```

字段说明：

表9-1 DMA_InitTypeDef 字段说明

字段	描述
Direction	配置数据在存储器和外设之间的传输方向
PeriphInc	配置外设地址增量模式
MemInc	配置存储器地址增量模式
PeriphDataAlignment	配置外设数据宽度
MemDataAlignment	配置存储器数据宽度
Mode	配置DMA 通道的循环模式
Priority	配置DMA 通道的软件优先级

参数说明：

Direction 可选参数：

表9-2 Direction 可选参数

参数	描述
DMA_PERIPH_TO_MEMORY	设置 DMA 传输方向为外设到存储器
DMA_MEMORY_TO_PERIPH	设置 DMA 传输方向为存储器到外设
DMA_MEMORY_TO_MEMORY	设置 DMA 传输方向为存储器到存储器

PeriphInc 可选参数：

表9-3 PeriphInc 可选参数

参数	描述
DMA_PINC_ENABLE	外设地址递增
DMA_PINC_DISABLE	外设地址不变

MemInc 可选参数：

表9-4 MemInc 可选参数

参数	描述
DMA_MINC_ENABLE	存储器地址递增
DMA_MINC_DISABLE	存储器地址不变

PeriphDataAlignment 可选参数：

表9-5 PeriphDataAlignment 可选参数

参数	描述
DMA_PDATAALIGN_BYTE	外设数据按字节传输
DMA_PDATAALIGN_HALFWORD	外设数据按半字传输
DMA_PDATAALIGN_WORD	外设数据按字传输

MemDataAlignment 可选参数：

表9-6 MemDataAlignment 可选参数

参数	描述
DMA_MDATAALIGN_BYTE	存储器数据按字节传输
DMA_MDATAALIGN_HALFWORD	存储器数据按半字传输
DMA_MDATAALIGN_WORD	存储器数据按字传输

Mode 可选参数：

表9-7 Mode 可选参数

参数	描述
DMA_NORMAL	普通模式
DMA_CIRCULAR	循环模式

Priority 可选参数：

表9-8 Priority 可选参数

参数	描述
DMA_PRIORITY_LOW	优先级：低
DMA_PRIORITY_MEDIUM	优先级：中
DMA_PRIORITY_HIGH	优先级：高
DMA_PRIORITY_VERY_HIGH	优先级：非常高

9.1.2 DMA_HandleTypeDef

DMA_HandleTypeDef，定义于文件”air001xx_hal_dma.h”如下：

```
typedef struct __DMA_HandleTypeDef
{
DMA_Channel_TypeDef *Instance;
DMA_InitTypeDef Init;
HAL_LockTypeDef Lock;
__IO HAL_DMA_StateTypeDef State;
void *Parent;
void (* XferCpltCallback)( struct __DMA_HandleTypeDef * hdma);
void (* XferHalfCpltCallback)( struct __DMA_HandleTypeDef * hdma);
void (* XferErrorCallback)( struct __DMA_HandleTypeDef * hdma);
void (* XferAbortCallback)( struct __DMA_HandleTypeDef * hdma);
__IO uint32_t ErrorCode;
DMA_TypeDef *DmaBaseAddress;
uint32_t ChannelIndex;
} DMA_HandleTypeDef;
```

字段说明：

表9-9 DMA_HandleTypeDef 字段说明

字段	描述
Instance	外设寄存器基地址
Init	外设初始化参数结构体指针
Lock	外设锁定状态
State	外设状态信息
Parent	父对象状态
(*XferCpltCallback)(struct __DMA_HandleTypeDef * hdma)	传输完成回调函数指针
(*XferHalfCpltCallback)(struct __DMA_HandleTypeDef * hdma)	半传输完成回调函数指针

(*XferErrorCallback)(struct __DMA_HandleTypeDef * hdma)	传输错误回调函数指针
(*XferAbortCallback)(struct __DMA_HandleTypeDef * hdma)	传输中止回调函数指针
ErrorCode	错误代码
DmaBaseAddress	DMA 通道基地址
ChannelIndex	DMA 通道索引

9.2 DMA 固件库函数

表9-10 DMA 固件库函数说明

函数名	描述
HAL_DMA_Init	初始化 DMA
HAL_DMA_DeInit	将 DMA 参数设为缺省值
HAL_DMA_Start	启动DMA 传输
HAL_DMA_Start_IT	在中断模式下启动DMA 传输
HAL_DMA_Abort	中止DMA 传输
HAL_DMA_Abort_IT	中止在中断模式下的 DMA 传输
HAL_DMA_PollForTransfer	使用轮询的方式处理已完成的传输
HAL_DMA_IRQHandler	处理DMA 中断请求
HAL_DMA_RegisterCallback	注册DMA 用户回调函数
HAL_DMA_UnRegisterCallback	注销DMA 用户回调函数
HAL_DMA_ChannelMap	DMA 通道映射
HAL_DMA_GetState	获取DMA 状态信息
HAL_DMA_GetError	获取DMA 错误代码

9.2.1 函数 HAL_DMA_Init

描述了函数 HAL_DMA_Init

表9-11 函数 HAL_DMA_Init

函数名	HAL_DMA_Init
函数原形	HAL_StatusTypeDef HAL_DMA_Init(DMA_HandleTypeDef *hdma)
功能描述	初始化 DMA
输入参数	hdma：DMA 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

9.2.2 函数 HAL_DMA_DeInit

描述了函数 HAL_DMA_DeInit

表9-12 函数 HAL_DMA_DeInit

函数名	HAL_DMA_DeInit
函数原形	HAL_StatusTypeDef HAL_DMA_DeInit (DMA_HandleTypeDef *hdma)
功能描述	将 DMA 参数设为缺省值
输入参数	hdma：DMA 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

9.2.3 函数 HAL_DMA_Start

描述了函数 HAL_DMA_Start

表9-13 函数 HAL_DMA_Start

函数名	HAL_DMA_Start
函数原形	HAL_StatusTypeDef HAL_DMA_Start (DMA_HandleTypeDef *hdma, uint32_t SrcAddress, uint32_t DstAddress, uint32_t DataLength)
功能描述	启动 DMA 传输
输入参数 1	hdma：DMA 句柄
输入参数 2	SrcAddress：发送缓冲区地址
输入参数 3	DstAddress：接收缓冲区地址
输入参数 4	DataLength：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

9.2.4 函数 HAL_DMA_Start_IT

描述了函数 HAL_DMA_Start_IT

表9-14 函数 HAL_DMA_Start_IT

函数名	HAL_DMA_Start_IT
函数原形	HAL_StatusTypeDef HAL_DMA_Start_IT(DMA_HandleTypeDef *hdma, uint32_t SrcAddress, uint32_t DstAddress, uint32_t DataLength)
功能描述	在中断模式下启动 DMA 传输
输入参数 1	hdma：DMA 句柄
输入参数 2	SrcAddress：发送缓冲区地址
输入参数 3	DstAddress：接收缓冲区地址
输入参数 4	DataLength：数据长度

输出参数	无
返回值	HAL 状态
先决条件	无

9.2.5 函数 HAL_DMA_Abort

描述了函数 HAL_DMA_Abort

表9-15 函数 HAL_DMA_Abort

函数名	HAL_DMA_Abort
函数原形	HAL_StatusTypeDef HAL_DMA_Abort(DMA_HandleTypeDef * hdma)
功能描述	中止 DMA 传输
输入参数	hdma：DMA 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

9.2.6 函数 HAL_DMA_Abort_IT

描述了函数 HAL_DMA_Abort_IT

表9-16 函数 HAL_DMA_Abort_IT

函数名	HAL_DMA_Abort_IT
函数原形	HAL_StatusTypeDef HAL_DMA_Abort_IT(DMA_HandleTypeDef *hdma)
功能描述	中止中断模式下的 DMA 传输
输入参数	hdam：DMA 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

9.2.7 函数 HAL_DMA_PollForTransfer

描述了函数 HAL_DMA_PollForTransfer

表9-17 函数 HAL_DMA_PollForTransfer 1

函数名	HAL_DMA_PollForTransfer
函数原形	HAL_StatusTypeDef HAL_DMA_PollForTransfer(DMA_HandleTypeDef *hdma, uint32_t CompleteLevel, uint32_t Timeout)
功能描述	使用轮询的方式处理已完成的传输
输入参数 1	hdma：DMA 句柄
输入参数 2	CompleteLevel：传输完成认定等级（半传输完成或全传输完成）
输入参数 3	Timeout：完成处理超时时间
输出参数	无

返回值	HAL 状态
先决条件	无

9.2.8 函数 HAL_DMA_IRQHandler

描述了函数 HAL_DMA_IRQHandler

表9-18 函数 HAL_DMA_IRQHandler

函数名	HAL_DMA_IRQHandler
函数原形	void HAL_DMA_IRQHandler(DMA_HandleTypeDef *hdma)
功能描述	处理 DMA 中断请求
输入参数	hdma：DMA 句柄
输出参数	无
返回值	无
先决条件	无

9.2.9 函数 HAL_DMA_RegisterCallback

描述了函数 HAL_DMA_RegisterCallback

表9-19 函数 HAL_DMA_RegisterCallback

函数名	HAL_DMA_RegisterCallback
函数原形	HAL_StatusTypeDef HAL_DMA_RegisterCallback(DMA_HandleTypeDef *hdma, HAL_DMA_CallbackIDTypeDef CallbackID, void (* pCallback)(DMA_HandleTypeDef * _hdma))
功能描述	注册 DMA 用户回调函数
输入参数 1	hdma：DMA 句柄
输入参数 2	CallbackID：回调标识 ID
输入参数 3	pCallback：用户回调函数指针
输出参数	无
返回值	HAL 状态
先决条件	无

9.2.10 函数 HAL_DMA_ChannelMap

描述了函数 HAL_DMA_ChannelMap

表9-20 函数 HAL_DMA_ChannelMap

函数名	HAL_DMA_ChannelMap
函数原形	void HAL_DMA_ChannelMap(DMA_HandleTypeDef *hdma, uint32_t MapReqNum)
功能描述	注销 DMA 用户回调函数
输入参数 1	hdma：DMA 句柄
输入参数 2	MapReqNum：请求 ID

输出参数	无
返回值	无
先决条件	无

表9-21 MapReqNum 可选参数

参数	描述
DMA_CHANNEL_MAP_ADC	将 ADC DMA 请求映射当前通道
DMA_CHANNEL_MAP_SPI1_TX	将 SPI1_TX DMA 请求映射当前通道
DMA_CHANNEL_MAP_SPI1_RX	将 SPI1_RX DMA 请求映射当前通道
DMA_CHANNEL_MAP_SPI2_TX	将 SPI2_TX DMA 请求映射当前通道
DMA_CHANNEL_MAP_SPI2_RX	将 SPI2_RX DMA 请求映射当前通道
DMA_CHANNEL_MAP_USART1_TX	将 USART1_TX DMA 请求映射当前通道
DMA_CHANNEL_MAP_USART1_RX	将 USART1_RX DMA 请求映射当前通道
DMA_CHANNEL_MAP_USART2_TX	将 USART2_TX DMA 请求映射当前通道
DMA_CHANNEL_MAP_USART2_RX	将 USART2_RX DMA 请求映射当前通道
DMA_CHANNEL_MAP_I2C_TX	将 I2C_TX DMA 请求映射当前通道
DMA_CHANNEL_MAP_I2C_RX	将 I2C_RX DMA 请求映射当前通道
DMA_CHANNEL_MAP_TIM1_CH1	将 TIM1_CH1 DMA 请求映射当前通道
DMA_CHANNEL_MAP_TIM1_CH2	将 TIM1_CH2 DMA 请求映射当前通道
DMA_CHANNEL_MAP_TIM1_CH3	将 TIM1_CH3 DMA 请求映射当前通道
DMA_CHANNEL_MAP_TIM1_CH4	将 TIM1_CH4 DMA 请求映射当前通道
DMA_CHANNEL_MAP_TIM1_COM	将 TIM1_COM DMA 请求映射当前通道
DMA_CHANNEL_MAP_TIM1_UP	将 TIM1_UP DMA 请求映射当前通道
DMA_CHANNEL_MAP_TIM1_TRIG	将 TIM1_TRIG DMA 请求映射当前通道
DMA_CHANNEL_MAP_TIM3_CH1	将 TIM3_CH1 DMA 请求映射当前通道
DMA_CHANNEL_MAP_TIM3_CH3	将 TIM3_CH3 DMA 请求映射当前通道
DMA_CHANNEL_MAP_TIM3_CH4	将 TIM3_CH4 DMA 请求映射当前通道
DMA_CHANNEL_MAP_TIM3_TRIG	将 TIM3_TRIG DMA 请求映射当前通道
DMA_CHANNEL_MAP_TIM3_UP	将 TIM3_UP DMA 请求映射当前通道
DMA_CHANNEL_MAP_TIM16_CH1	将 TIM16_CH1 DMA 请求映射当前通道
DMA_CHANNEL_MAP_TIM16_UP	将 TIM16_UP DMA 请求映射当前通道
DMA_CHANNEL_MAP_TIM17_CH1	将 TIM17_CH1 DMA 请求映射当前通道
DMA_CHANNEL_MAP_TIM17_UP	将 TIM17_UP DMA 请求映射当前通道

9.2.11 函数 HAL_DMA_UnRegisterCallback

描述了函数 HAL_DMA_UnRegisterCallback

表9-22 函数 HAL_DMA_UnRegisterCallback

函数名	HAL_DMA_UnRegisterCallback
函数原形	HAL_StatusTypeDef HAL_DMA_UnRegisterCallback(DMA_HandleTypeDef *hdma, HAL_DMA_CallbackIDTypeDef CallbackID)
功能描述	注销 DMA 用户回调函数
输入参数 1	hdma：DMA 句柄
输入参数 2	CallbackID：回调标识 ID
输出参数	无
返回值	HAL 状态
先决条件	无

9.2.12 函数 HAL_DMA_GetState

描述了函数 HAL_DMA_GetState

表9-23 函数 HAL_DMA_GetState

函数名	HAL_DMA_GetState
函数原形	HAL_DMA_StateTypeDef HAL_DMA_GetState(DMA_HandleTypeDef *hdma)
功能描述	获取 DMA 状态信息
输入参数	hdma：DMA 句柄
输出参数	无
返回值	DMA 状态
先决条件	无

9.2.13 函数 HAL_DMA_GetError

描述了函数 HAL_DMA_GetError

表9-24 函数 HAL_DMA_GetError

函数名	HAL_DMA_GetError
函数原形	uint32_t HAL_DMA_GetError(DMA_HandleTypeDef *hdma)
功能描述	获取 DMA 错误代码
输入参数	hdma：DMA 句柄
输出参数	无
返回值	错误代码
先决条件	无

10 HAL 外部中断/事件控制器通用驱动程序（EXTI）

外部中断/事件控制器管理 21 个中断输入（19 个可配置(configurable)中断和 2 个直接(direct)中断）。可配置中断能够选择触发端口、触发边沿和触发模式（中断/事件），直接中断直接由指定外设触发。

10.1 EXTI 固件驱动寄存器结构

10.1.1 EXTI_HandleTypeDef

EXTI_HandleTypeDef，定义于文件"air001xx_hal_exti.h"如下：

```
typedef struct
{
uint32_t Line;
void (* PendingCallback)(void);
} EXTI_HandleTypeDef;
```

字段说明：

表10-1 EXTI_HandleTypeDef 字段说明

字段	描述
Line	配置外部的中断线
PendingCallback	外部中断线挂起回调函数

10.1.2 EXTI_ConfigTypeDef

EXTI_ConfigTypeDef，定义于文件"air001xx_hal_exti.h"如下：

```
typedef struct
{
uint32_t Line;
uint32_t Mode;
uint32_t Trigger;
uint32_t GPIOSel;
} EXTI_ConfigTypeDef;
```

字段说明：

表10-2 EXTI_ConfigTypeDef 字段说明

字段	描述
Line	要配置的外部中断线
Mode	配置内核处理外部中断的模式（中断/事件/无）
Trigger	配置外部中断触发边沿
GPIOSel	GPIO 外部中断端口选择

参数说明：

Line 可选参数：

表10-3 Line 可选参数

参数	描述
EXTI_LINE_0	0 号外部中断线
EXTI_LINE_1	1 号外部中断线
EXTI_LINE_2	2 号外部中断线
EXTI_LINE_3	3 号外部中断线
EXTI_LINE_4	4 号外部中断线
EXTI_LINE_5	5 号外部中断线
EXTI_LINE_6	6 号外部中断线
EXTI_LINE_7	7 号外部中断线
EXTI_LINE_8	8 号外部中断线
EXTI_LINE_9	9 号外部中断线
EXTI_LINE_10	10 号外部中断线
EXTI_LINE_11	11 号外部中断线
EXTI_LINE_12	12 号外部中断线
EXTI_LINE_13	13 号外部中断线
EXTI_LINE_14	14 号外部中断线
EXTI_LINE_15	15 号外部中断线
EXTI_LINE_16	16 号外部中断线
EXTI_LINE_17	17 号外部中断线
EXTI_LINE_18	18 号外部中断线
EXTI_LINE_19	19 号外部中断线
EXTI_LINE_29	29 号外部中断线

Mode 可选参数：

表10-4 Mode 可选参数

参数	描述
EXTI_MODE_NONE	不响应外部触发
EXTI_MODE_INTERRUPT	外部触发中断
EXTI_MODE_EVENT	外部触发事件

Trigger 可选参数：

表10-5 Trigger 可选参数

参数	描述

EXTI_TRIGGER_NONE	不触发中断
EXTI_TRIGGER_RISING	上升沿触发中断
EXTI_TRIGGER_FALLING	下降沿触发中断
EXTI_TRIGGER_RISING_FALLING	上升沿和下降沿触发中断

GPIOSel 可选参数：

表10-6 GPIOSel 可选参数

参数	描述
EXTI_GPIOA	设置中断线对应 A 端口
EXTI_GPIOB	设置中断线对应 B 端口
EXTI_GPIOF	设置中断线对应 F 端口

10.2 EXTI 固件库函数

表10-7 EXTI 固件库函数说明

函数名	描述
HAL_EXTI_SetConfigLine	配置指定的 EXTI 线路
HAL_EXTI_GetConfigLine	获取指定的 EXTI 线路的配置
HAL_EXTI_ClearConfigLine	清除指定的 EXTI 线路的整体配置
HAL_EXTI_RegisterCallback	注册 EXTI 用户回调函数
HAL_EXTI_GetHandle	将 EXTI 线序号保存到EXTI 句柄中
HAL_EXTI_IRQHandler	处理 EXTI 中断请求
HAL_EXTI_GetPending	获取指定EXTI 线路的挂起位
HAL_EXTI_ClearPending	清除指定EXTI 线路的挂起位
HAL_EXTI_GenerateSWI	在指定线路上产生软件中断

10.2.1 函数 HAL_EXTI_SetConfigLine

描述了函数 HAL_EXTI_SetConfigLine

表10-8 函数 HAL_EXTI_SetConfigLine

函数名	HAL_EXTI_SetConfigLine
函数原形	HAL_StatusTypeDef HAL_EXTI_SetConfigLine(EXTI_HandleTypeDef *hexti, EXTI_ConfigTypeDef *pExtiConfig)
功能描述	配置指定的 EXTI 线路
输入参数 1	hexti：EXTI 句柄
输入参数 2	pExtiConfig：EXTI 配置结构体
输出参数	无
返回值	HAL 状态

先决条件	无

10.2.2 函数 HAL_EXTI_GetConfigLine

描述了函数 HAL_EXTI_GetConfigLine

表10-9 函数 HAL_EXTI_GetConfigLine

函数名	HAL_EXTI_GetConfigLine
函数原形	HAL_StatusTypeDef HAL_EXTI_GetConfigLine(EXTI_HandleTypeDef *hexti, EXTI_ConfigTypeDef *pExtiConfig)
功能描述	获取指定 EXTI 线路的配置
输入参数 1	hexti：EXTI 句柄
输入参数 2	pExtiConfig：EXTI 配置结构体
输出参数	pExtiConfig
返回值	HAL 状态
先决条件	无

10.2.3 函数 HAL_EXTI_ClearConfigLine

描述了函数 HAL_EXTI_ClearConfigLine

表10-10 函数 HAL_EXTI_ClearConfigLine

函数名	HAL_EXTI_ClearConfigLine
函数原形	HAL_StatusTypeDef HAL_EXTI_ClearConfigLine(EXTI_HandleTypeDef *hexti)
功能描述	清除指定线路的所有配置
输入参数	hexti：EXTI 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

10.2.4 函数 HAL_EXTI_RegisterCallback

描述了函数 HAL_EXTI_RegisterCallback

表10-11 函数 HAL_EXTI_RegisterCallback

函数名	HAL_EXTI_RegisterCallback
函数原形	HAL_StatusTypeDef HAL_EXTI_RegisterCallback(EXTI_HandleTypeDef *hexti, EXTI_CallbackIDTypeDef CallbackID, void (*pPendingCbfn)(void))
功能描述	注册 EXTI 回调函数
输入参数 1	hexti：EXTI 句柄
输入参数 2	CallbackID：回调 ID
输入参数 3	pPendingCbfn：回调函数指针
输出参数	无
返回值	HAL 状态

先决条件	无

10.2.5 函数 HAL_EXTI_GetHandle

描述了函数 HAL_EXTI_GetHandle

表10-12 函数 HAL_EXTI_GetHandle

函数名	HAL_EXTI_GetHandle
函数原形	HAL_StatusTypeDef HAL_EXTI_GetHandle(EXTI_HandleTypeDef *hexti, uint32_t ExtiLine)
功能描述	将指定 EXTI 中断线序号存入 EXTI 句柄中
输入参数 1	hexti：EXTI 句柄
输入参数 2	ExtiLine：EXTI 中断线序号
输出参数	hexti
返回值	HAL 状态
先决条件	无

10.2.6 函数 HAL_EXTI_IRQHandler

描述了函数 HAL_EXTI_IRQHandler

表10-13 函数 HAL_EXTI_IRQHandler

函数名	HAL_EXTI_IRQHandler
函数原形	void HAL_EXTI_IRQHandler(EXTI_HandleTypeDef *hexti)
功能描述	EXTI 中断请求处理
输入参数	hexti：EXTI 句柄
输出参数	无
返回值	无
先决条件	无

10.2.7 函数 HAL_EXTI_GetPending

描述了函数 HAL_EXTI_GetPending

表10-14 函数 HAL_EXTI_GetPending

函数名	HAL_EXTI_GetPending
函数原形	uint32_t HAL_EXTI_GetPending(EXTI_HandleTypeDef *hexti, uint32_t Edge)
功能描述	获取指定 EXTI 中断线路的挂起位
输入参数 1	hexti：EXTI 句柄
输入参数 2	Edge：中断触发沿
输出参数	无
返回值	被挂起（1）/未挂起（0）
先决条件	无

Edge 可选参数：

表10-15 Edge 可选参数

参数	描述
EXTI_TRIGGER_RISING_FALLING	上升沿和下降沿中断挂起

10.2.8 函数 HAL_EXTI_ClearPending

描述了函数 HAL_EXTI_ClearPending

函数名	HAL_EXTI_ClearPending
函数原形	void HAL_EXTI_ClearPending(EXTI_HandleTypeDef *hexti, uint32_t Edge)
功能描述	清除指定 EXTI 中断线路的挂起位
输入参数 1	hexti：EXTI 句柄
输入参数 2	Edge：中断触发沿
输出参数	无
返回值	无
先决条件	无

Edge 可选参数：

表10-16 Edge 可选参数

参数	描述
EXTI_TRIGGER_RISING_FALLING	上升沿和下降沿中断挂起

10.2.9 函数 HAL_EXTI_GenerateSWI

描述了函数 HAL_EXTI_GenerateSWI

表10-17 函数 HAL_EXTI_GenerateSWI

函数名	HAL_EXTI_GenerateSWI
函数原形	void HAL_EXTI_GenerateSWI(EXTI_HandleTypeDef * hexti)
功能描述	在指定的 EXTI 中断线路上产生软件中断
输入参数	hexti：EXTI 句柄
输出参数	无
返回值	无
先决条件	无

11 HAL 闪存存储器通用驱动程序（FLASH）

Flash 接口可管理CPU(Cortex®-M0+)AHB 对 Flash 进行的访问。该接口可针对 Flash 执行擦除和编程操作，读写保护和安全机制。

Flash 接口通过指令预取和缓存机制加速代码执行。

11.1 FLASH 固件驱动寄存器结构

11.1.1 FLASH_EraseInitTypeDef

FLASH_EraseInitTypeDef，定义于文件"air001xx_hal_flash.h"如下：

```
typedef struct
{
uint32_t TypeErase;
uint32_t PageAddress;
uint32_t NbPages;
uint32_t SectorAddress;
uint32_t NbSectors;
} FLASH_EraseInitTypeDef;
```

字段说明：

表11-1 FLASH_EraseInitTypeDef 字段说明

字段	描述
TypeErase	配置擦除类型
PageAddress	配置需要擦除页的起始地址
NbPages	配置需要擦除的页数（1~（最大页数-所选页数））
SectorAddress	配置需要擦除扇区的起始地址
NbSectors	配置需要擦除的扇区数（1~（最大扇区数-所选扇区数））

参数说明：

TypeErase 可选参数：

表11-2 TypeErase 可选参数

参数	描述
FLASH_TYPEERASE_MASSERASE	MASS 擦除
FLASH_TYPEERASE_PAGEERASE	PAGE 擦除
FLASH_TYPEERASE_SECTORERASE	SECTOR 擦除

11.1.2 FLASH_OBProgramInitTypeDef

FLASH_OBProgramInitTypeDef，定义于文件"air001xx_hal_flash.h"如下：

```
typedef struct
{
uint32_t    OptionType;
uint32_t    WRPSector;
uint32_t    SDKStartAddr;
uint32_t    SDKEndAddr;
uint32_t    RDPLevel;
uint32_t    USERType;
uint32_t    USERConfig;
} FLASH_OBProgramInitTypeDef;
```

字段说明：

表11-3 FLASH_OBProgramInitTypeDef 字段说明

字段	描述
OptionType	配置选项字节
WRPSector	配置指定扇区写保护
SDKStartAddr	SDK 写保护起始地址
SDKEndAddr	SDK 写保护结束地址
RDPLevel	设置读保护级别
USERType	需要配置的 USER 选项字节
USERConfig	用户选项字节配置值

参数说明：

OptionType 可选参数：

表11-4 OptionType 可选参数

参数	描述
OPTIONBYTE_WRP	配置WRP 选项字节
OPTIONBYTE_SDK	配置 SDK 选项字节
OPTIONBYTE_RDP	配置 RDP 选项字节
OPTIONBYTE_USER	配置 USER 选项字节

WRPSector 可选参数：

表11-5 WRPSector 可选参数

参数	描述

OB_WRP_SECTOR_0	设置扇区 0 写保护
OB_WRP_SECTOR_1	设置扇区 1 写保护
OB_WRP_SECTOR_2	设置扇区 2 写保护
OB_WRP_SECTOR_3	设置扇区 3 写保护
OB_WRP_SECTOR_4	设置扇区 4 写保护
OB_WRP_SECTOR_5	设置扇区 5 写保护
OB_WRP_SECTOR_6	设置扇区 6 写保护
OB_WRP_SECTOR_7	设置扇区 7 写保护
OB_WRP_SECTOR_8	设置扇区 8 写保护
OB_WRP_SECTOR_9	设置扇区 9 写保护
OB_WRP_SECTOR_10	设置扇区 10 写保护
OB_WRP_SECTOR_11	设置扇区 11 写保护
OB_WRP_SECTOR_12	设置扇区 12 写保护
OB_WRP_SECTOR_13	设置扇区 13 写保护
OB_WRP_SECTOR_14	设置扇区 14 写保护
OB_WRP_SECTOR_15	设置扇区 15 写保护
OB_WRP_Pages0to31	设置 0~31 页写保护
OB_WRP_Pages32to63	设置 32~63 页写保护
OB_WRP_Pages64to95	设置 64~95 页写保护
OB_WRP_Pages96to127	设置 96~127 页写保护
OB_WRP_Pages128to159	设置 128~159 页写保护
OB_WRP_Pages160to191	设置 160~191 页写保护
OB_WRP_Pages192to223	设置 192~223 页写保护
OB_WRP_Pages224to255	设置 224~255 页写保护
OB_WRP_Pages256to287	设置 256~287 页写保护
OB_WRP_Pages288to319	设置 288~319 页写保护
OB_WRP_Pages320to351	设置 320~351 页写保护
OB_WRP_Pages352to383	设置 352~383 页写保护
OB_WRP_Pages384to415	设置 384~415 页写保护
OB_WRP_Pages416to447	设置 416~447 页写保护
OB_WRP_Pages448to479	设置 448~479 页写保护
OB_WRP_Pages480to511	设置 480~511 页写保护
OB_WRP_AllPages	设置所有页写保护

RDPLevel 可选参数:

表11-6 RDPLevel 可选参数

参数	描述
OB_RDP_LEVEL_0	读保护级别 0
OB_RDP_LEVEL_1	读保护级别 1

USERType 可选参数：

表11-7 USERType 可选参数

参数	描述
OB_USER_BOR_EN	配置 BOR_EN
OB_USER_BOR_LEV	配置 BOR_LEV
OB_USER_IWDG_SW	配置 IWDG
OB_USER_WWDG_SW	配置WWDG
OB_USER_NRST_MODE	配置 NRST
OB_USER_nBOOT1	配置 nBOOT1
OB_USER_ALL	配置所有用户选项字节

USERConfig 可选参数：

表11-8 USERConfig 可选参数

参数	描述
OB_BOR_DISABLE	关闭 BOR 检测
OB_BOR_ENABLE	开启 BOR 检测
OB_BOR_LEVEL_1p7_1p8	设置 BOR 下降阈值 1.7V，上升阈值 1.8V
OB_BOR_LEVEL_1p9_2p0	设置 BOR 下降阈值 1.9V，上升阈值 2.0V
OB_BOR_LEVEL_2p1_2p2	设置 BOR 下降阈值 2.1V，上升阈值 2.2V
OB_BOR_LEVEL_2p3_2p4	设置 BOR 下降阈值 2.3V，上升阈值 2.4V
OB_BOR_LEVEL_2p5_2p6	设置 BOR 下降阈值 2.5V，上升阈值 2.6V
OB_BOR_LEVEL_2p7_2p8	设置 BOR 下降阈值 2.7V，上升阈值 2.8V
OB_BOR_LEVEL_2p9_3p0	设置 BOR 下降阈值 2.9V，上升阈值 3.0V
OB_BOR_LEVEL_3p1_3p2	设置 BOR 下降阈值 3.1V，上升阈值 3.2V
OB_RESET_MODE_RESET	复位引脚仅作为复位输入功能
OB_RESET_MODE_GPIO	复位引脚仅作为 GPIO 功能
OB_IWDG_SW	选择软件 IWDG
OB_IWDG_HW	选择硬件 IWDG
OB_WWDG_SW	选择软件WWDG
OB_WWDG_HW	选择硬件WWDG
OB_BOOT1_SRAM	nBOOT1 清零
OB_BOOT1_SYSTEM	nBOOT1 置位

11.1.3 FLASH_ProcessTypeDef

FLASH_ProcessTypeDef，定义于文件"air001xx_hal_flash.h"如下：

```
typedef struct
{
HAL_LockTypeDef Lock;
uint32_t ErrorCode;
uint32_t ProcedureOnGoing;
uint32_t Address;
uint32_t PageOrSector;
uint32_t NbPagesSectorsToErase;
} FLASH_ProcessTypeDef;
```

字段说明：

表11-9 FLASH_ProcessTypeDef 字段说明

字段	描述
Lock	HAL 锁状态
ErrorCode	错误代码
ProcedureOnGoing	当前正在中断模式运行的程序
Address	在启用中断的情况下当前正在擦除的地址
PageOrSector	在启用中断的情况下当前正在擦除的页/扇区
NbPagesSectorsToErase	在启用中断的情况下总共要擦除的页/扇区数

11.2 FLASH 固件库函数

表11-10 FLASH 固件库函数说明

函数名	描述
HAL_FLASH_Init	初始化 FLASH
HAL_FLASH_PageProgram	启动FLASH 页编程
HAL_FLASH_PageProgram_IT	启动FLASH 页编程并开启中断
HAL_FLASH_IRQHandler	FLASH 中断请求处理
HAL_FLASH_EndOfOperationCallback	FLASH 编程结束回调函数
HAL_FLASH_OperationErrorCallback	FLASH 编程错误回调函数
HAL_FLASH_Erase	执行FLASH 擦除
HAL_FLASH_Erase_IT	执行FLASH 擦除，并开启中断
HAL_FLASH_Unlock	解锁FLASH 控制寄存器访问

HAL_FLASH_Lock	锁定FLASH 控制寄存器访问
HAL_FLASH_OB_Unlock	解锁FLASH 选项字节寄存器访问
HAL_FLASH_OB_Lock	锁定FLASH 选项字节寄存器访问
HAL_FLASH_OB_Launch	重装载选项字节配置
HAL_FLASH_OBProgram	选项字节编程
HAL_FLASH_OBGetConfig	获取选项字节设置
HAL_OB_RDP_LevelConfig	设置读保护等级
HAL_FLASH_GetError	获取错误代码

11.2.1 函数 HAL_FLASH_Init

描述了函数 HAL_FLASH_Init

表11-11 函数 HAL_FLASH_Init

函数名	HAL_FLASH_Init
函数原形	void HAL_FLASH_Init(uint32_t u32ClockID)
功能描述	初始化 FLASH
输入参数	u32ClockID：选择当前 HSI 时钟频率
输出参数	无
返回值	无
先决条件	无

u32ClockID 可选参数：

表11-12 u32ClockID 可选参数

参数	描述
FLASH_PROGRAM_ERASE_CLOCK_4MHZ	HIS 当前频率为 4MHz
FLASH_PROGRAM_ERASE_CLOCK_8MHZ	HIS 当前频率为 8MHz
FLASH_PROGRAM_ERASE_CLOCK_16MHZ	HIS 当前频率为 16MHz
FLASH_PROGRAM_ERASE_CLOCK_22P12MHZ	HIS 当前频率为 22.12MHz
FLASH_PROGRAM_ERASE_CLOCK_24MHZ	HIS 当前频率为 24MHz

11.2.2 函数 HAL_FLASH_PageProgram

描述了函数 HAL_FLASH_PageProgram

表11-13 函数 HAL_FLASH_PageProgram

函数名	HAL_FLASH_PageProgram
函数原形	HAL_StatusTypeDef HAL_FLASH_PageProgram (uint32_t Address, uint32_t *DataAddr)
功能描述	将指定大小的数据写入 FLASH 指定页

输入参数 1	Address：写入页的起始地址
输入参数 2	DataAddr：写入数据的起始地址
输出参数	无
返回值	HAL 状态
先决条件	无

11.2.3 函数 HAL_FLASH_PageProgram_IT

描述了函数 HAL_FLASH_PageProgram_IT

表11-14 函数 HAL_FLASH_PageProgram_IT

函数名	HAL_FLASH_PageProgram_IT
函数原形	HAL_StatusTypeDef HAL_FLASH_PageProgram_IT (uint32_t Address, uint64_t Data)
功能描述	启动 FLASH 页编程并开启中断
输入参数 1	Address：写入页的起始地址
输入参数 2	DataAddr：写入数据的起始地址
输出参数	无
返回值	HAL 状态
先决条件	无

11.2.4 函数 HAL_FLASH_IRQHandler

描述了函数 HAL_FLASH_IRQHandler

表11-15 函数 HAL_FLASH_IRQHandler

函数名	HAL_FLASH_IRQHandler
函数原形	void HAL_FLASH_IRQHandler(void)
功能描述	中断请求处理
输入参数	无
输出参数	无
返回值	无
先决条件	无

11.2.5 函数 HAL_FLASH_EndOfOperationCallback

描述了函数 HAL_FLASH_EndOfOperationCallback

表11-16 函数 HAL_FLASH_EndOfOperationCallback

函数名	HAL_FLASH_EndOfOperationCallback
函数原形	void HAL_FLASH_EndOfOperationCallback(uint32_t ReturnValue)
功能描述	FLASH 编程结束中断回调函数
输入参数	ReturnValue：与当前 FLASH 运行的功能有关

输出参数	无
返回值	无
先决条件	无

11.2.6 函数 HAL_FLASH_OperationErrorCallback

描述了函数 HAL_FLASH_OperationErrorCallback

表11-17 函数 HAL_FLASH_OperationErrorCallback

函数名	HAL_FLASH_OperationErrorCallback
函数原形	void HAL_FLASH_OperationErrorCallback(uint32_t ReturnValue)
功能描述	FLASH 编程错误中断回调函数
输入参数	ReturnValue：与当前 FLASH 运行的功能有关
输出参数	无
返回值	无
先决条件	无

11.2.7 函数 HAL_FLASH_Erase

描述了函数 HAL_FLASH_Erase

表11-18 函数 HAL_FLASH_Erase

函数名	HAL_FLASH_Erase
函数原形	HAL_StatusTypeDef HAL_FLASH_Erase(FLASH_EraseInitTypeDef *pEraseInit, uint32_t *PageError)
功能描述	执行全擦除/块擦除/页擦除
输入参数	pEraseInit：擦除初始化结构体
输出参数	PageError：指向操作出错的 FLASH 页地址的指针（未出错则指向 0xFFFFFFFF）
返回值	HAL 状态
先决条件	无

11.2.8 函数 HAL_FLASH_Erase_IT

描述了函数 HAL_FLASH_Erase_IT

表11-19 函数 HAL_FLASH_Erase_IT

函数名	HAL_FLASH_Erase_IT
函数原形	HAL_StatusTypeDef HAL_FLASH_Erase_IT(FLASH_EraseInitTypeDef *pEraseInit)
功能描述	执行 FLASH 擦除，并开启中断
输入参数	pEraseInit：擦除初始化结构体
输出参数	无
返回值	HAL 状态

先决条件	无

11.2.9 函数 HAL_FLASH_Unlock

描述了函数 HAL_FLASH_Unlock

表11-20 函数 HAL_FLASH_Unlock

函数名	HAL_FLASH_Unlock
函数原形	HAL_StatusTypeDef HAL_FLASH_Unlock(void)
功能描述	解锁 FLASH 控制器访问
输入参数	无
输出参数	无
返回值	HAL 状态
先决条件	无

11.2.10 函数 HAL_FLASH_Lock

描述了函数 HAL_FLASH_Lock

表11-21 函数 HAL_FLASH_Lock

函数名	HAL_FLASH_Lock
函数原形	HAL_StatusTypeDef HAL_FLASH_Lock(void)
功能描述	锁定 FLASH 控制器访问
输入参数	无
输出参数	无
返回值	HAL 状态
先决条件	无

11.2.11 函数 HAL_FLASH_OB_Unlock

描述了函数 HAL_FLASH_OB_Unlock

表11-22 函数 HAL_FLASH_OB_Unlock

函数名	HAL_FLASH_OB_Unlock
函数原形	HAL_StatusTypeDef HAL_FLASH_OB_Unlock(void)
功能描述	解锁 FLASH 选项字节寄存器访问
输入参数	无
输出参数	无
返回值	HAL 状态
先决条件	无

11.2.12 函数 HAL_FLASH_OB_Lock

描述了函数 HAL_FLASH_OB_Lock

表11-23 函数 HAL_FLASH_OB_Lock

函数名	HAL_FLASH_OB_Lock
函数原形	HAL_StatusTypeDef HAL_FLASH_OB_Lock(void)
功能描述	锁定 FLASH 选项字节寄存器访问
输入参数	无
输出参数	无
返回值	HAL 状态
先决条件	无

11.2.13 函数 HAL_FLASH_OB_Launch

描述了函数 HAL_FLASH_OB_Launch

表11-24 函数 HAL_FLASH_OB_Launch

函数名	HAL_FLASH_OB_Launch
函数原形	HAL_StatusTypeDef HAL_FLASH_OB_Launch(void)
功能描述	重装载选项字节配置
输入参数	无
输出参数	无
返回值	HAL 状态
先决条件	无

11.2.14 函数 HAL_FLASH_OBProgram

描述了函数 HAL_FLASH_OBProgram

表11-25 函数 HAL_FLASH_OBProgram

函数名	HAL_FLASH_OBProgram
函数原形	HAL_StatusTypeDef HAL_FLASH_OBProgram(FLASH_OBProgramInitTypeDef *pOBInit)
功能描述	启动选项字节编程
输入参数	pOBInit：选项字节初始化结构体指针
输出参数	无
返回值	HAL 状态
先决条件	无

11.2.15 函数 HAL_FLASH_OBGetConfig

描述了函数 HAL_FLASH_OBGetConfig

表11-26 函数 HAL_FLASH_OBGetConfig

函数名	HAL_FLASH_OBGetConfig
函数原形	void HAL_FLASH_OBGetConfig(FLASH_OBProgramInitTypeDef *pOBInit)

功能描述	获取选项字节的配置值，保存在 pOBInit 中
输入参数	pOBInit：选项字节初始化结构体指针
输出参数	pOBInit：选项字节初始化结构体指针
返回值	无
先决条件	无

11.2.16 函数 HAL_OB_RDP_LevelConfig

描述了函数 HAL_OB_RDP_LevelConfig

表11-27 函数 HAL_OB_RDP_LevelConfig

函数名	HAL_OB_RDP_LevelConfig
函数原形	HAL_StatusTypeDef LASH_OB_RDP_LevelConfig(uint8_t ReadProtectLevel)
功能描述	配置读保护等级
输入参数	ReadProtectLevel：读保护等级
输出参数	无
返回值	HAL 状态
先决条件	无

11.2.17 函数 HAL_FLASH_GetError

描述了函数 HAL_FLASH_GetError

表11-28 函数 HAL_FLASH_GetError

函数名	HAL_FLASH_GetError
函数原形	uint32_t HAL_FLASH_GetError(void)
功能描述	获取错误代码
输入参数	无
输出参数	无
返回值	错误代码
先决条件	无

12 HAL 通用输入/输出通用驱动程序（GPIO）

每个 GPIO 端口有：
4 个 32 位配置寄存器(GPIOx_MODER,GPIOx_OTYPER,GPIOx_OSPEEDR, GPIOx_PUPDR)
2 个 32 位数据寄存器 (GPIOx_IDR 和 GPIOx_ODR)
1 个 32 位置位/复位寄存器(GPIOx_BSRR)
1 个 32 位锁定寄存器(GPIOx_LCKR)
2 个复用功能选择寄存器(GPIOx_AFRH 和 GPIOx_AFRL)。

12.1 GPIO 寄存器结构

12.1.1 GPIO_InitTypeDef

GPIO_InitTypeDef，定义于文件”air001xx_hal_gpio.h”如下：

```
typedef struct
{
uint32_t Pin;
uint32_t Mode;
uint32_t Pull;
uint32_t Speed;
uint32_t Alternate;
} GPIO_InitTypeDef;
```

字段说明：

表12-1 GPIO_InitTypeDef 字段说明

字段	描述
Pin	选择需要配置的引脚
Mode	配置指定引脚的模式
Pull	配置指定引脚上拉或下拉
Speed	配置指定引脚的速度
Alternate	配置指定引脚需要连接到的外设

参数说明：

Pin 可选参数：

表12-2 Pin 可选参数

参数	描述
GPIO_PIN_0	引脚 0
GPIO_PIN_1	引脚 1

GPIO_PIN_2	引脚 2
GPIO_PIN_3	引脚 3
GPIO_PIN_4	引脚 4
GPIO_PIN_5	引脚 5
GPIO_PIN_6	引脚 6
GPIO_PIN_7	引脚 7
GPIO_PIN_8	引脚 8
GPIO_PIN_9	引脚 9
GPIO_PIN_10	引脚 10
GPIO_PIN_11	引脚 11
GPIO_PIN_12	引脚 12
GPIO_PIN_13	引脚 13
GPIO_PIN_14	引脚 14
GPIO_PIN_15	引脚 15
GPIO_PIN_All	全部引脚

Mode 可选参数：

表12-3 Mode 可选参数

参数	描述
GPIO_MODE_INPUT	输入模式
GPIO_MODE_OUTPUT_PP	推挽输出模式
GPIO_MODE_OUTPUT_OD	开漏输出模式
GPIO_MODE_AF_PP	推挽复用模式
GPIO_MODE_AF_OD	开漏复用模式
GPIO_MODE_ANALOG	模拟模式
GPIO_MODE_IT_RISING	外部中断，上升沿触发模式
GPIO_MODE_IT_FALLING	外部中断，下降沿触发模式
GPIO_MODE_IT_RISING_FALLING	外部中断，上升沿和下降沿触发模式
GPIO_MODE_EVT_RISING	外部事件，上升沿触发模式
GPIO_MODE_EVT_FALLING	外部事件，下降沿触发模式
GPIO_MODE_EVT_RISING_FALLING	外部事件，上升沿下降沿触发模式

Pull 可选参数：

表12-4 Pull 可选参数

参数	描述
GPIO_NOPULL	无上拉或下拉
GPIO_PULLUP	引脚上拉

GPIO_PULLDOWN	引脚下拉

Speed 可选参数：

表12-5 Speed 可选参数

参数	描述
GPIO_SPEED_FREQ_LOW	低速度模式
GPIO_SPEED_FREQ_MEDIUM	中速度模式
GPIO_SPEED_FREQ_HIGH	高速度模式
GPIO_SPEED_FREQ_VERY_HIGH	最高速度模式

Alternate 可选参数：

表12-6 Alternate 可选参数

参数	描述
GPIO_AF0_SWJ	复用模式 AF 0：连接到 SWDIO
GPIO_AF0_SPI1	复用模式 AF 0：连接到 SPI1
GPIO_AF0_SPI2	复用模式 AF 0：连接到 SPI2
GPIO_AF0_TIM14	复用模式 AF 0：连接到 TIM14
GPIO_AF0_USART1	复用模式 AF 0：连接到 USART1
GPIO_AF0_USART2	复用模式 AF 0：连接到 USART2
GPIO_AF1_IR	复用模式 AF 1：连接到 IRTIM
GPIO_AF1_SPI2	复用模式 AF 1：连接到 SPI2
GPIO_AF1_TIM1	复用模式 AF 1：连接到 TIM1
GPIO_AF1_TIM3	复用模式 AF 1：连接到 TIM3
GPIO_AF1_USART1	复用模式 AF 1：连接到 USART1
GPIO_AF1_USART2	复用模式 AF 1：连接到 USART2
GPIO_AF2_SPI2	复用模式 AF 2：连接到 SPI2
GPIO_AF2_TIM2	复用模式 AF 2：连接到 TIM2
GPIO_AF2_TIM14	复用模式 AF 2：连接到 TIM14
GPIO_AF2_TIM16	复用模式 AF 2：连接到 TIM16
GPIO_AF2_TIM17	复用模式 AF 2：连接到 TIM17
GPIO_AF3_LED	复用模式 AF 3：连接到 LED
GPIO_AF3_USART1	复用模式 AF 3：连接到 USART1
GPIO_AF3_USART2	复用模式 AF 3：连接到 USART2
GPIO_AF3_SPI2	复用模式 AF 3：连接到 SPI2
GPIO_AF4_TIM14	复用模式 AF 4：连接到 TIM14
GPIO_AF4_USART2	复用模式 AF 4：连接到 USART2
GPIO_AF5_LPTIM	复用模式 AF 5：连接到 LPTIM

GPIO_AF5_USART2	复用模式 AF 5：连接到 USART2
GPIO_AF5_TIM16	复用模式 AF 5：连接到 TIM16
GPIO_AF5_TIM17	复用模式 AF 5：连接到 TIM17
GPIO_AF5_EVENTOUT	复用模式 AF 5：连接到 EVENT OUT
GPIO_AF5_MCO	复用模式 AF 5：连接到 MCO
GPIO_AF6_I2C	复用模式 AF 6：连接到 I2C
GPIO_AF6_LED	复用模式 AF 6：连接到 LED
GPIO_AF6_MCO	复用模式 AF 6：连接到 MCO
GPIO_AF6_EVENTOUT	复用模式 AF 6：连接到 EVENT OUT
GPIO_AF7_EVENTOUT	复用模式 AF 7：连接到 EVENT OUT
GPIO_AF7_COMP1	复用模式 AF 7：连接到 COMP1
GPIO_AF7_COMP2	复用模式 AF 7：连接到 COMP2
GPIO_AF8_USART1	复用模式 AF 8：连接到 USART1
GPIO_AF9_USART2	复用模式 AF 9：连接到 USART2
GPIO_AF10_SPI1	复用模式 AF 10：连接到 SPI1
GPIO_AF11_SPI2	复用模式 AF 11：连接到 SPI2
GPIO_AF12_I2C	复用模式 AF 12：连接到 I2C
GPIO_AF13_TIM1	复用模式 AF 13：连接到 TIM1
GPIO_AF13_TIM3	复用模式 AF 13：连接到 TIM3
GPIO_AF13_TIM14	复用模式 AF 13：连接到 TIM14
GPIO_AF13_TIM17	复用模式 AF 13：连接到 TIM17
GPIO_AF14_TIM1	复用模式 AF 14：连接到 TIM1
GPIO_AF15_RTCOUT	复用模式 AF 15：连接到 RTC OUT
GPIO_AF15_MCO	复用模式 AF 15：连接到 MCO
GPIO_AF15_IR	复用模式 AF 15：连接到 IRTIM

12.2 GPIO 固件库函数

表12-7 GPIO 固件库函数说明

函数名	描述
HAL_GPIO_Init	初始化指定 GPIO 引脚
HAL_GPIO_DeInit	将指定 GPIO 引脚设为缺省值
HAL_GPIO_ReadPin	读取指定引脚电平状态
HAL_GPIO_WritePin	配置指定引脚电平状态
HAL_GPIO_TogglePin	翻转指定引脚电平状态
HAL_GPIO_LockPin	锁定指定引脚配置

HAL_GPIO_EXTI_IRQHandler	GPIO 中断请求处理
HAL_GPIO_EXTI_Callback	GPIO 中断回调函数

12.2.1 函数 HAL_GPIO_Init

描述了函数 HAL_GPIO_Init

表12-8 函数 HAL_GPIO_Init

函数名	HAL_GPIO_Init
函数原形	void HAL_GPIO_Init(GPIO_TypeDef *GPIOx, GPIO_InitTypeDef *GPIO_Init)
功能描述	初始化指定 GPIO 引脚
输入参数 1	GPIOx：GPIO 端口
输入参数 2	GPIO_Init：初始化参数结构体
输出参数	无
返回值	无
先决条件	无

GPIOx 可选参数：

表12-9 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

12.2.2 函数 HAL_GPIO_DeInit

描述了函数 HAL_GPIO_DeInit

表12-10 函数 HAL_GPIO_DeInit

函数名	HAL_GPIO_DeInit
函数原形	void HAL_GPIO_DeInit(GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
功能描述	将指定引脚的配置设为缺省值
输入参数 1	GPIOx：GPIO 端口
输入参数 2	GPIO_Pin：引脚号
输出参数	无
返回值	无
先决条件	无

GPIOx 可选参数：

表12-11 GPIOx 可选参数

参数	描述

GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

GPIO_Pin 可选参数:

表12-12 GPIO_Pin 可选参数

参数	描述
GPIO_PIN_0	引脚 0
GPIO_PIN_1	引脚 1
GPIO_PIN_2	引脚 2
GPIO_PIN_3	引脚 3
GPIO_PIN_4	引脚 4
GPIO_PIN_5	引脚 5
GPIO_PIN_6	引脚 6
GPIO_PIN_7	引脚 7
GPIO_PIN_8	引脚 8
GPIO_PIN_9	引脚 9
GPIO_PIN_10	引脚 10
GPIO_PIN_11	引脚 11
GPIO_PIN_12	引脚 12
GPIO_PIN_13	引脚 13
GPIO_PIN_14	引脚 14
GPIO_PIN_15	引脚 15
GPIO_PIN_ALL	全部引脚

12.2.3 函数 HAL_GPIO_ReadPin

描述了函数 HAL_GPIO_ReadPin

表12-13 函数 HAL_GPIO_ReadPin

函数名	HAL_GPIO_ReadPin
函数原形	GPIO_PinState AL_GPIO_ReadPin(GPIO_TypeDef *GPIOx,uint16_t GPIO_Pin)
功能描述	获取指定引脚的电平状态
输入参数 1	GPIOx：GPIO 端口
输入参数 2	GPIO_Pin：引脚号
输出参数	无
返回值	电平状态：GPIO_PIN_RESET/ GPIO_PIN_SET
先决条件	无

GPIOx 可选参数：

表12-14 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

GPIO_Pin 可选参数：

表12-15 GPIO_Pin 可选参数

参数	描述
GPIO_PIN_0	引脚 0
GPIO_PIN_1	引脚 1
GPIO_PIN_2	引脚 2
GPIO_PIN_3	引脚 3
GPIO_PIN_4	引脚 4
GPIO_PIN_5	引脚 5
GPIO_PIN_6	引脚 6
GPIO_PIN_7	引脚 7
GPIO_PIN_8	引脚 8
GPIO_PIN_9	引脚 9
GPIO_PIN_10	引脚 10
GPIO_PIN_11	引脚 11
GPIO_PIN_12	引脚 12
GPIO_PIN_13	引脚 13
GPIO_PIN_14	引脚 14
GPIO_PIN_15	引脚 15
GPIO_PIN_ALL	全部引脚

12.2.4 函数 HAL_GPIO_WritePin

描述了函数 HAL_GPIO_WritePin

表12-16 函数 HAL_GPIO_WritePin

函数名	HAL_GPIO_WritePin
函数原形	void HAL_GPIO_WritePin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)
功能描述	配置指定引脚的电平状态
输入参数 1	GPIOx：GPIO 端口
输入参数 2	GPIO_Pin：引脚号

输入参数 3	PinState：电平状态
输出参数	无
返回值	无
先决条件	无

GPIOx 可选参数：

表12-17 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

GPIO_Pin 可选参数：

表12-18 GPIO_Pin 可选参数

参数	描述
GPIO_PIN_0	引脚 0
GPIO_PIN_1	引脚 1
GPIO_PIN_2	引脚 2
GPIO_PIN_3	引脚 3
GPIO_PIN_4	引脚 4
GPIO_PIN_5	引脚 5
GPIO_PIN_6	引脚 6
GPIO_PIN_7	引脚 7
GPIO_PIN_8	引脚 8
GPIO_PIN_9	引脚 9
GPIO_PIN_10	引脚 10
GPIO_PIN_11	引脚 11
GPIO_PIN_12	引脚 12
GPIO_PIN_13	引脚 13
GPIO_PIN_14	引脚 14
GPIO_PIN_15	引脚 15
GPIO_PIN_ALL	全部引脚

12.2.5 函数 HAL_GPIO_TogglePin

描述了函数 HAL_GPIO_TogglePin

表12-19 函数 HAL_GPIO_TogglePin

函数名	HAL_GPIO_TogglePin

函数原形	void HAL_GPIO_TogglePin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)
功能描述	翻转指定引脚的电平状态
输入参数 1	GPIOx：GPIO 端口
输入参数 2	GPIO_Pin：引脚号
输出参数	无
返回值	无
先决条件	无

GPIOx 可选参数:

表12-20 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

GPIO_Pin 可选参数:

表12-21 GPIO_Pin 可选参数

参数	描述
GPIO_PIN_0	引脚 0
GPIO_PIN_1	引脚 1
GPIO_PIN_2	引脚 2
GPIO_PIN_3	引脚 3
GPIO_PIN_4	引脚 4
GPIO_PIN_5	引脚 5
GPIO_PIN_6	引脚 6
GPIO_PIN_7	引脚 7
GPIO_PIN_8	引脚 8
GPIO_PIN_9	引脚 9
GPIO_PIN_10	引脚 10
GPIO_PIN_11	引脚 11
GPIO_PIN_12	引脚 12
GPIO_PIN_13	引脚 13
GPIO_PIN_14	引脚 14
GPIO_PIN_15	引脚 15
GPIO_PIN_ALL	全部引脚

12.2.6 函数 HAL_GPIO_LockPin

描述了函数 HAL_GPIO_LockPin

表12-22 函数 HAL_GPIO_LockPin

函数名	HAL_GPIO_LockPin
函数原形	HAL_StatusTypeDef HAL_GPIO_LockPin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)
功能描述	锁定指定引脚的配置
输入参数 1	GPIOx：GPIO 端口
输入参数 2	GPIO_Pin：引脚号
输出参数	无
返回值	HAL 状态
先决条件	无

GPIOx 可选参数：

表12-23 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

GPIO_Pin 可选参数：

表12-24 GPIO_Pin 可选参数

参数	描述
GPIO_PIN_0	引脚 0
GPIO_PIN_1	引脚 1
GPIO_PIN_2	引脚 2
GPIO_PIN_3	引脚 3
GPIO_PIN_4	引脚 4
GPIO_PIN_5	引脚 5
GPIO_PIN_6	引脚 6
GPIO_PIN_7	引脚 7
GPIO_PIN_8	引脚 8
GPIO_PIN_9	引脚 9
GPIO_PIN_10	引脚 10
GPIO_PIN_11	引脚 11
GPIO_PIN_12	引脚 12
GPIO_PIN_13	引脚 13

GPIO_PIN_14	引脚 14
GPIO_PIN_15	引脚 15
GPIO_PIN_ALL	全部引脚

12.2.7 函数 HAL_GPIO_EXTI_IRQHandler

描述了函数 HAL_GPIO_EXTI_IRQHandler

表12-25 函数 HAL_GPIO_EXTI_IRQHandler

函数名	HAL_GPIO_EXTI_IRQHandler
函数原形	void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
功能描述	GPIO 中断请求处理
输入参数	GPIO_Pin：引脚号
输出参数	无
返回值	无
先决条件	无

12.2.8 函数 HAL_GPIO_EXTI_Callback

描述了函数 HAL_GPIO_EXTI_Callback

表12-26 函数 HAL_GPIO_EXTI_Callback

函数名	HAL_GPIO_EXTI_Callback
函数原形	void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
功能描述	GPIO 回调函数
输入参数	GPIO_Pin：引脚号
输出参数	无
返回值	无

13 HAL 内部集成电路总线通用驱动程序（I2C）

I2C（内部集成电路）总线接口处理微控制器与串行 I2C 总线间的通信。它提供多主模式功能，可以控制所有 I2C 总线特定的序列、协议、仲裁和时序。它支持标准模式 (Sm)、快速模式 (Fm)。

13.1 I2C 固件驱动寄存器结构

13.1.1 I2C_InitTypeDef

I2C_InitTypeDef，定于文件”air001xx_hal_i2c.h”

```
typedef struct
{
uint32_t ClockSpeed;
uint32_t DutyCycle;
uint32_t OwnAddress1;
uint32_t GeneralCallMode;
uint32_t NoStretchMode;
} I2C_InitTypeDef;
```

字段说明：

表13-1 I2C_InitTypeDef 字段说明

字段	描述
ClockSpeed	配置时钟频率
DutyCycle	配置快速模式下的占空比
OwnAddress1	配置设备的地址
GeneralCallMode	配置广播模式
NoStretchMode	配置时钟拉长

参数说明：

ClockSpeed 可选参数：

表13-2 ClockSpeed 可选参数

参数	描述
0~400000	时钟频率最高 400kHz

DutyCycle 可选参数：

表13-3 DutyCycle 可选参数

参数	描述
I2C_DUTYCYCLE_2	快速模式下占空比：$T_{low}/T_{high}=2$

I2C_DUTYCYCLE_16_9	快速模式下占空比：T_{low}/T_{high}=16/9

GeneralCallMode 可选参数：

表13-4 GeneralCallMode 可选参数

参数	描述
I2C_GENERALCALL_DISABLE	关闭广播模式
I2C_GENERALCALL_ENABLE	使能广播模式

NoStretchMode 可选参数：

表13-5 NoStretchMode 可选参数

参数	描述
I2C_NOSTRETCH_DISABLE	关闭时钟拉长
I2C_NOSTRETCH_ENABLE	开启时钟拉长

13.1.2 I2C_HandleTypeDef

I2C_HandleTypeDef，定于文件”air001xx_hal_i2c.h”

```
typedef struct __I2C_HandleTypeDef
{
I2C_TypeDef *Instance;
I2C_InitTypeDef Init;
uint8_t *pBuffPtr;
uint16_t XferSize;
__IO uint16_t XferCount;
__IO uint32_t XferOptions;
__IO uint32_t PreviousState;
DMA_HandleTypeDef *hdmatx;
DMA_HandleTypeDef *hdmarx;
HAL_LockTypeDef Lock;
__IO HAL_I2C_StateTypeDef State;
__IO HAL_I2C_ModeTypeDef Mode;
__IO uint32_t ErrorCode;
__IO uint32_t Devaddress;
__IO uint32_t Memaddress;
__IO uint32_t MemaddSize;
__IO uint32_t EventCount;
} I2C_HandleTypeDef;
```

字段说明：

表13-6 I2C_HandleTypeDef 字段说明

字段	描述
Instance	外设基地址
Init	初始化结构体
pBuffPtr	数据缓冲区指针
XferSize	传输数量总量
XferCount	未传输的数据数量
XferOptions	传输选项
PreviousState	前一次通信的状态
hdmatx	DMA 发送句柄
hdmarx	DMA 接收句柄
Lock	HAL 锁
State	HAL 状态
Mode	通信模式
ErrorCode	错误代码
Devaddress	目标设备地址
Memaddress	目标存储器地址
MemaddSize	目标存储器地址大小
EventCount	事件计数

13.2 I2C 固件库函数

表13-7 I2C 固件库函数说明

函数名	描述
HAL_I2C_Init	初始化 I2C
HAL_I2C_DeInit	将 I2C 配置设为缺省值
HAL_I2C_MspInit	I2C 相关 MSP 初始化
HAL_I2C_MspDeInit	I2C 相关 MSP 去初始化
HAL_I2C_Master_Transmit	在主模式下使用轮询的方式发送数据
HAL_I2C_Master_Receive	在主模式下使用轮询的方式接收数据
HAL_I2C_Slave_Transmit	在从模式下使用轮询的方式发送数据
HAL_I2C_Slave_Receive	在从模式下使用轮询的方式接收数据
HAL_I2C_Mem_Write	使用轮询的方式将数据写入指定存储器地址

HAL_I2C_Mem_Read	使用轮询的方式将指定存储器地址中的数据读出
HAL_I2C_IsDeviceReady	检查目标设备是否准备好通信
HAL_I2C_Master_Transmit_IT	在主模式下使用中断的方式发送数据
HAL_I2C_Master_Receive_IT	在主模式下使用中断的方式接收数据
HAL_I2C_Slave_Transmit_IT	在从模式下使用中断的方式发送数据
HAL_I2C_Slave_Receive_IT	在从模式下使用中断的方式接收数据
HAL_I2C_Mem_Write_IT	使用中断的方式将数据写入指定存储器地址
HAL_I2C_Mem_Read_IT	使用中断的方式将指定存储器地址中的数据读出
HAL_I2C_EnableListen_IT	使能在中断模式下 I2C 地址监听
HAL_I2C_DisableListen_IT	关闭在中断模式下 I2C 地址监听
HAL_I2C_Master_Abort_IT	使用中断的方式中止 I2C 的中断传输或DMA 传输
HAL_I2C_Master_Transmit_DMA	在主模式下使用 DMA 的方式发送数据
HAL_I2C_Master_Receive_DMA	在主模式下使用 DMA 的方式接收数据
HAL_I2C_Slave_Transmit_DMA	在从模式下使用 DMA 的方式发送数据
HAL_I2C_Slave_Receive_DMA	在从模式下使用 DMA 的方式接收数据
HAL_I2C_Mem_Write_DMA	使用DMA 的方式将数据写入指定存储器地址
HAL_I2C_Mem_Read_DMA	使用DMA 的方式将指定存储器地址中的数据读出
HAL_I2C_EV_IRQHandler	事件中断请求处理
HAL_I2C_ER_IRQHandler	错误中断请求处理
HAL_I2C_MasterTxCpltCallback	主模式发送完成回调函数
HAL_I2C_MasterRxCpltCallback	主模式接收完成回调函数
HAL_I2C_SlaveTxCpltCallback	从模式发送完成回调函数
HAL_I2C_SlaveRxCpltCallback	从模式接收完成回调函数
HAL_I2C_AddrCallback	从模式地址匹配回调函数
HAL_I2C_ListenCpltCallback	通信完成回调函数
HAL_I2C_MemTxCpltCallback	写入存储器完成回调函数
HAL_I2C_MemRxCpltCallback	读取内容完成回调函数
HAL_I2C_ErrorCallback	错误回调函数
HAL_I2C_AbortCpltCallback	中止完成回调函数
HAL_I2C_GetState	获取通信状态
HAL_I2C_GetMode	获取传输模式（主机，从机，存储器，无)

HAL_I2C_GetError	获取错误代码

13.2.1 函数 HAL_I2C_Init

描述了函数 HAL_I2C_Init

表13-8 函数 HAL_I2C_Init

函数名	HAL_I2C_Init
函数原形	HAL_StatusTypeDef HAL_I2C_Init(I2C_HandleTypeDef *hi2c)
功能描述	初始化 I2C
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.2 函数 HAL_I2C_DeInit

描述了函数 HAL_I2C_DeInit

表13-9 函数 HAL_I2C_DeInit

函数名	HAL_I2C_DeInit
函数原形	HAL_StatusTypeDef HAL_I2C_DeInit(I2C_HandleTypeDef *hi2c)
功能描述	将 I2C 配置设为缺省值
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.3 函数 HAL_I2C_MspInit

描述了函数 HAL_I2C_MspInit

表13-10 函数 HAL_I2C_MspInit

函数名	HAL_I2C_MspInit
函数原形	void HAL_I2C_MspInit(I2C_HandleTypeDef *hi2c)
功能描述	初始化 I2C 相关的 MSP
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	无
先决条件	无

13.2.4 函数 HAL_I2C_MspDeInit

描述了函数 HAL_I2C_MspDeInit

表13-11 函数 HAL_I2C_MspDeInit

函数名	HAL_I2C_MspDeInit
函数原形	void HAL_I2C_MspDeInit(I2C_HandleTypeDef *hi2c)
功能描述	将 I2C 相关的 MSP 设为缺省值
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	无
先决条件	无

13.2.5 函数 HAL_I2C_Master_Transmit

描述了函数 HAL_I2C_Master_Transmit

表13-12 函数 HAL_I2C_Master_Transmit

函数名	HAL_I2C_Master_Transmit
函数原形	HAL_StatusTypeDef HAL_I2C_Master_Transmit(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint8_t *pData, uint16_t Size, uint32_t Timeout)
功能描述	在主模式下使用轮询的方式发送数据
输入参数 1	hi2c：I2C 句柄
输入参数 2	DevAddress：从设备地址
输入参数 3	pData：数据缓冲区指针
输入参数 4	Size：数据长度
输入参数 5	Timeout：超时时间
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.6 函数 HAL_I2C_Master_Receive

描述了函数 HAL_I2C_Master_Receive

表13-13 函数 HAL_I2C_Master_Receive

函数名	HAL_I2C_Master_Receive
函数原形	HAL_StatusTypeDef HAL_I2C_Master_Receive(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint8_t *pData, uint16_t Size, uint32_t Timeout)
功能描述	在主模式下使用轮询的方式接收数据
输入参数 1	hi2c：I2C 句柄
输入参数 2	DevAddress：从设备地址
输入参数 3	pData：数据缓冲区指针

输入参数 4	Size：数据长度
输入参数 5	Timeout：超时时间
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.7 函数 HAL_I2C_Slave_Transmit

描述了函数 HAL_I2C_Slave_Transmit

表13-14 函数 HAL_I2C_Slave_Transmit

函数名	HAL_I2C_Slave_Transmit
函数原形	HAL_StatusTypeDef HAL_I2C_Slave_Transmit(I2C_HandleTypeDef *hi2c, uint8_t *pData, uint16_t Size, uint32_t Timeout)
功能描述	在从模式下使用轮询的方式发送数据
输入参数 1	hi2c：I2C 句柄
输入参数 2	pData：数据缓冲区指针
输入参数 3	Size：数据长度
输入参数 4	Timeout：超时时间
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.8 函数 HAL_I2C_Slave_Receive

描述了函数 HAL_I2C_Slave_Receive

表13-15 函数 HAL_I2C_Slave_Receive

函数名	HAL_I2C_Slave_Receive
函数原形	HAL_StatusTypeDef HAL_I2C_Slave_Receive(I2C_HandleTypeDef *hi2c, uint8_t *pData, uint16_t Size, uint32_t Timeout)
功能描述	在从模式下使用轮询的方式接收数据
输入参数 1	hi2c：I2C 句柄
输入参数 2	pData：数据缓冲区指针
输入参数 3	Size：数据长度
输入参数 4	Timeout：超时时间
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.9 函数 HAL_I2C_Mem_Write

描述了函数 HAL_I2C_Mem_Write

表13-16 函数 HAL_I2C_Mem_Write

函数名	HAL_I2C_Mem_Write
函数原形	HAL_StatusTypeDef HAL_I2C_Mem_Write(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t *pData, uint16_t Size, uint32_t Timeout)
功能描述	使用轮询的方式将数据写入指定存储器地址
输入参数 1	hi2c：I2C 句柄
输入参数 2	DevAddress：目标设备地址
输入参数 3	MemAddress：存储器地址
输入参数 4	MemAddSize：存储器地址数据宽度
输入参数 5	pData：数据缓冲区指针
输入参数 6	Size：数据长度
输出参数 7	Timeout：超时时间
返回值	HAL 状态
先决条件	无

13.2.10 函数 HAL_I2C_Mem_Read

描述了函数 HAL_I2C_Mem_Read

表13-17 函数 HAL_I2C_Mem_Read

函数名	HAL_I2C_Mem_Read
函数原形	HAL_StatusTypeDef HAL_I2C_Mem_Read(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t *pData, uint16_t Size, uint32_t Timeout)
功能描述	使用轮询的方式读取指定存储器地址的数据
输入参数 1	hi2c：I2C 句柄
输入参数 2	DevAddress：目标设备地址
输入参数 3	MemAddress：存储器地址
输入参数 4	MemAddSize：存储器地址数据宽度
输入参数 5	pData：数据缓冲区指针
输入参数 6	Size：数据长度
输出参数 7	Timeout：超时时间
返回值	HAL 状态
先决条件	无

13.2.11 函数 HAL_I2C_IsDeviceReady

描述了函数 HAL_I2C_IsDeviceReady

表13-18 函数 HAL_I2C_IsDeviceReady

函数名	HAL_I2C_IsDeviceReady

函数原形	HAL_StatusTypeDef HAL_I2C_IsDeviceReady(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint32_t Trials, uint32_t Timeout)
功能描述	检查目标设备是否准备好通信
输入参数 1	hi2c：I2C 句柄
输入参数 2	DevAddress：目标设备地址
输入参数 3	Trials：尝试次数
输入参数 4	Timeout：超时时间
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.12 函数 HAL_I2C_Master_Transmit_IT

描述了函数 HAL_I2C_Master_Transmit_IT

表13-19 函数 HAL_I2C_Master_Transmit_IT

函数名	HAL_I2C_Master_Transmit_IT
函数原形	HAL_StatusTypeDef HAL_I2C_Master_Transmit_IT(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint8_t *pData, uint16_t Size)
功能描述	在主模式下使用中断的方式发送数据
输入参数 1	hi2c：I2C 句柄
输入参数 2	DevAddress：从设备地址
输入参数 3	pData：数据缓冲区指针
输入参数 4	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.13 函数 HAL_I2C_Master_Receive_IT

描述了函数 HAL_I2C_Master_Receive_IT

表13-20 函数 HAL_I2C_Master_Receive_IT

函数名	HAL_I2C_Master_Receive_IT
函数原形	HAL_StatusTypeDef HAL_I2C_Master_Receive_IT(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint8_t *pData, uint16_t Size)
输入参数 1	在主模式下使用中断的方式接收数据
输入参数 2	hi2c：I2C 句柄
输入参数 3	DevAddress：从设备地址
输入参数 4	pData：数据缓冲区指针
输入参数	Size：数据长度
输出参数	无

返回值	HAL 状态
先决条件	无

13.2.14 函数 HAL_I2C_Slave_Transmit_IT

描述了函数 HAL_I2C_Slave_Transmit_IT

表13-21 函数 HAL_I2C_Slave_Transmit_IT

函数名	HAL_I2C_Slave_Transmit_IT
函数原形	HAL_StatusTypeDef HAL_I2C_Slave_Transmit_IT(I2C_HandleTypeDef *hi2c, uint8_t *pData, uint16_t Size)
功能描述	在从模式下使用中断的方式发送数据
输入参数 1	hi2c：I2C 句柄
输入参数 2	pData：数据缓冲区指针
输入参数 3	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.15 函数 HAL_I2C_Slave_Receive_IT

描述了函数 HAL_I2C_Slave_Receive_IT

表13-22 函数 HAL_I2C_Slave_Receive_IT

函数名	HAL_I2C_Slave_Receive_IT
函数原形	HAL_StatusTypeDef HAL_I2C_Slave_Receive_IT(I2C_HandleTypeDef *hi2c, uint8_t *pData, uint16_t Size)
功能描述	在从模式下使用中断的方式接收数据
输入参数 1	hi2c：I2C 句柄
输入参数 2	pData：数据缓冲区指针
输入参数 3	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.16 函数 HAL_I2C_Mem_Write_IT

描述了函数 HAL_I2C_Mem_Write_IT

表13-23 函数 HAL_I2C_Mem_Write_IT

函数名	HAL_I2C_Mem_Write_IT
函数原形	HAL_StatusTypeDef HAL_I2C_Mem_Write_IT(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t *pData, uint16_t Size)
功能描述	使用中断的方式将数据写入指定存储器地址

输入参数 1	hi2c：I2C 句柄
输入参数 2	DevAddress：目标设备地址
输入参数 3	MemAddress：存储器地址
输入参数 4	MemAddSize：存储器地址数据宽度
输入参数 5	pData：数据缓冲区指针
输入参数 6	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.17 函数 HAL_I2C_Mem_Read_IT

描述了函数 HAL_I2C_Mem_Read_IT

表13-24 函数 HAL_I2C_Mem_Read_IT

函数名	HAL_I2C_Mem_Read_IT
函数原形	HAL_StatusTypeDef HAL_I2C_Mem_Read_IT(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t *pData, uint16_t Size)
功能描述	使用中断的方式读取指定存储器地址的数据
输入参数 1	hi2c：I2C 句柄
输入参数 2	DevAddress：目标设备地址
输入参数 3	MemAddress：存储器地址
输入参数 4	MemAddSize：存储器地址数据宽度
输入参数 5	pData：数据缓冲区指针
输入参数 6	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.18 函数 HAL_I2C_EnableListen_IT

描述了函数 HAL_I2C_EnableListen_IT

表13-25 函数 HAL_I2C_EnableListen_IT

函数名	HAL_I2C_EnableListen_IT
函数原形	HAL_StatusTypeDef HAL_I2C_EnableListen_IT(I2C_HandleTypeDef *hi2c)
功能描述	使能在中断模式下 I2C 地址监听
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	HAL 状态

先决条件	无

13.2.19 函数 HAL_I2C_DisableListen_IT

描述了函数 HAL_I2C_DisableListen_IT

表13-26 函数 HAL_I2C_DisableListen_IT

函数名	HAL_I2C_DisableListen_IT
函数原形	HAL_StatusTypeDef HAL_I2C_DisableListen_IT(I2C_HandleTypeDef *hi2c)
功能描述	关闭在中断模式下 I2C 地址监听
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.20 函数 HAL_I2C_Master_Abort_IT

描述了函数 HAL_I2C_Master_Abort_IT

表13-27 函数 HAL_I2C_Master_Abort_IT

函数名	HAL_I2C_Master_Abort_IT
函数原形	HAL_StatusTypeDef HAL_I2C_Master_Abort_IT(I2C_HandleTypeDef *hi2c, uint16_t DevAddress)
功能描述	中止 I2C 的传输并关闭中断
输入参数 1	hi2c：I2C 句柄
输入参数 2	DevAddress：从设备地址
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.21 函数 HAL_I2C_Master_Transmit_DMA

描述了函数 HAL_I2C_Master_Transmit_DMA

表13-28 HAL_I2C_Master_Transmit_DMA

函数名	HAL_I2C_Master_Transmit_DMA
函数原形	HAL_StatusTypeDef HAL_I2C_Master_Transmit_DMA(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint8_t *pData, uint16_t Size)
功能描述	在主模式下使用 DMA 的方式发送数据
输入参数 1	hi2c：I2C 句柄
输入参数 2	DevAddress：从设备地址
输入参数 3	pData：数据缓冲区指针
输入参数 4	Size：数据长度
输出参数	无

返回值	HAL 状态
先决条件	无

13.2.22 函数 HAL_I2C_Master_Receive_DMA

描述了函数 HAL_I2C_Master_Receive_DMA

表13-29 函数 HAL_I2C_Master_Receive_DMA

函数名	HAL_I2C_Master_Receive_DMA
函数原形	HAL_StatusTypeDef HAL_I2C_Master_Receive_DMA(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint8_t *pData, uint16_t Size)
功能描述	在主模式下使用 DMA 的方式接收数据
输入参数 1	hi2c：I2C 句柄
输入参数 2	DevAddress：从设备地址
输入参数 3	pData：数据缓冲区指针
输入参数 4	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.23 函数 HAL_I2C_Slave_Transmit_DMA

描述了函数 HAL_I2C_Slave_Transmit_DMA

表13-30 函数 HAL_I2C_Slave_Transmit_DMA

函数名	HAL_I2C_Slave_Transmit_DMA
函数原形	HAL_StatusTypeDef HAL_I2C_Slave_Transmit_DMA(I2C_HandleTypeDef *hi2c, uint8_t *pData, uint16_t Size)
功能描述	在从模式下使用 DMA 的方式发送数据
输入参数 1	hi2c：I2C 句柄
输入参数 2	pData：数据缓冲区指针
输入参数 3	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.24 函数 HAL_I2C_Slave_Receive_DMA

描述了函数 HAL_I2C_Slave_Receive_DMA

表13-31 函数 HAL_I2C_Slave_Receive_DMA

函数名	HAL_I2C_Slave_Receive_DMA
函数原形	HAL_StatusTypeDef HAL_I2C_Slave_Receive_DMA(I2C_HandleTypeDef *hi2c, uint8_t *pData, uint16_t Size)
功能描述	在从模式下使用 DMA 的方式接收数据

输入参数 1	hi2c：I2C 句柄
输入参数 2	pData：数据缓冲区指针
输入参数 3	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.25 函数 HAL_I2C_Mem_Write_DMA

描述了函数 HAL_I2C_Mem_Write_DMA

表13-32 函数 HAL_I2C_Mem_Write_DMA

函数名	HAL_I2C_Mem_Write_DMA
函数原形	HAL_StatusTypeDef HAL_I2C_Mem_Write_DMA(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t *pData, uint16_t Size)
功能描述	使用 DMA 的方式将数据写入指定存储器地址
输入参数 1	hi2c：I2C 句柄
输入参数 2	DevAddress：目标设备地址
输入参数 3	MemAddress：存储器地址
输入参数 4	MemAddSize：存储器地址数据宽度
输入参数 5	pData：数据缓冲区指针
输入参数 6	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.26 函数 HAL_I2C_Mem_Read_DMA

描述了函数 HAL_I2C_Mem_Read_DMA

表13-33 函数 HAL_I2C_Mem_Read_DMA

函数名	HAL_I2C_Mem_Read_DMA
函数原形	HAL_StatusTypeDef HAL_I2C_Mem_Read_DMA(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t *pData, uint16_t Size)
功能描述	使用 DMA 的方式读取指定存储器地址的数据
输入参数 1	hi2c：I2C 句柄
输入参数 2	DevAddress：目标设备地址
输入参数 3	MemAddress：存储器地址
输入参数 4	MemAddSize：存储器地址数据宽度
输入参数 5	pData：数据缓冲区指针

输入参数 6	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

13.2.27 函数 HAL_I2C_EV_IRQHandler

描述了函数 HAL_I2C_EV_IRQHandler

表13-34 函数 HAL_I2C_EV_IRQHandler

函数名	HAL_I2C_EV_IRQHandler
函数原形	void HAL_I2C_EV_IRQHandler(I2C_HandleTypeDef *hi2c)
功能描述	I2C 事件中断请求处理
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	无
先决条件	无

13.2.28 函数 HAL_I2C_ER_IRQHandler

描述了函数 HAL_I2C_ER_IRQHandler

表13-35 函数 HAL_I2C_ER_IRQHandler

函数名	HAL_I2C_ER_IRQHandler
函数原形	void HAL_I2C_ER_IRQHandler(I2C_HandleTypeDef *hi2c)
功能描述	I2C 错误中断请求处理
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	无
先决条件	无

13.2.29 函数 HAL_I2C_MasterTxCpltCallback

描述了函数 HAL_I2C_MasterTxCpltCallback

表13-36 函数 HAL_I2C_MasterTxCpltCallback

函数名	HAL_I2C_MasterTxCpltCallback
函数原形	void HAL_I2C_MasterTxCpltCallback(I2C_HandleTypeDef *hi2c)
功能描述	主模式发送完成回调函数
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	无

先决条件	无

13.2.30 函数 HAL_I2C_MasterRxCpltCallback

描述了函数 HAL_I2C_MasterRxCpltCallback

表13-37　函数 HAL_I2C_MasterRxCpltCallback

函数名	HAL_I2C_MasterRxCpltCallback
函数原形	void　HAL_I2C_MasterRxCpltCallback(I2C_HandleTypeDef　*hi2c)
功能描述	主模式接收完成回调函数
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	无
先决条件	无

13.2.31 函数 HAL_I2C_SlaveTxCpltCallback

描述了函数 HAL_I2C_SlaveTxCpltCallback

表13-38　函数 HAL_I2C_SlaveTxCpltCallback

函数名	HAL_I2C_SlaveTxCpltCallback
函数原形	void HAL_I2C_SlaveTxCpltCallback(I2C_HandleTypeDef *hi2c)
功能描述	从模式发送完成回调函数
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	无
先决条件	无

13.2.32 函数 HAL_I2C_SlaveRxCpltCallback

描述了函数 HAL_I2C_SlaveRxCpltCallback

表13-39　函数 HAL_I2C_SlaveRxCpltCallback

函数名	HAL_I2C_SlaveRxCpltCallback
函数原形	void HAL_I2C_SlaveRxCpltCallback(I2C_HandleTypeDef *hi2c)
功能描述	从模式接收完成回调函数
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	无
先决条件	无

13.2.33 函数 HAL_I2C_AddrCallback

描述了函数 HAL_I2C_AddrCallback

表13-40　函数 HAL_I2C_AddrCallback

函数名	HAL_I2C_AddrCallback
函数原形	void HAL_I2C_AddrCallback(I2C_HandleTypeDef *hi2c, uint8_t TransferDirection, uint16_t AddrMatchCode)
功能描述	从模式地址匹配回调函数
输入参数 1	hi2c：I2C 句柄
输入参数 2	TransferDirection：主机发起的传输方向（发送/接收）
输入参数 3	AddrMatchCode：本机地址码
输出参数	无
返回值	无
先决条件	无

13.2.34 函数 HAL_I2C_ListenCpltCallback

描述了函数 HAL_I2C_ListenCpltCallback

表13-41　函数 HAL_I2C_ListenCpltCallback

函数名	HAL_I2C_ListenCpltCallback
函数原形	void HAL_I2C_ListenCpltCallback(I2C_HandleTypeDef *hi2c)
功能描述	通信完成回调函数
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	无
先决条件	无

13.2.35 函数 HAL_I2C_MemTxCpltCallback

描述了函数 HAL_I2C_MemTxCpltCallback

表13-42　函数 HAL_I2C_MemTxCpltCallback

函数名	HAL_I2C_MemTxCpltCallback
函数原形	void HAL_I2C_MemTxCpltCallback(I2C_HandleTypeDef *hi2c)
功能描述	写入存储器完成回调函数
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	无
先决条件	无

13.2.36 函数 HAL_I2C_MemRxCpltCallback

描述了函数 HAL_I2C_MemRxCpltCallback

表13-43 函数 HAL_I2C_MemRxCpltCallback

函数名	HAL_I2C_MemRxCpltCallback
函数原形	void HAL_I2C_MemRxCpltCallback(I2C_HandleTypeDef *hi2c)
功能描述	读取存储器完成回调函数
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	无
先决条件	无

13.2.37 函数 HAL_I2C_ErrorCallback

描述了函数 HAL_I2C_ErrorCallback

表13-44 函数 HAL_I2C_ErrorCallback

函数名	HAL_I2C_ErrorCallback
函数原形	void HAL_I2C_ErrorCallback(I2C_HandleTypeDef *hi2c)
功能描述	错误回调函数
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	无
先决条件	无

13.2.38 函数 HAL_I2C_AbortCpltCallback

描述了函数 HAL_I2C_AbortCpltCallback

表13-45 函数 HAL_I2C_AbortCpltCallback

函数名	HAL_I2C_AbortCpltCallback
函数原形	void HAL_I2C_AbortCpltCallback(I2C_HandleTypeDef *hi2c)
功能描述	中止完成回调函数
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	无
先决条件	无

13.2.39 函数 HAL_I2C_GetState

描述了函数 HAL_I2C_GetState

表13-46 函数 HAL_I2C_GetState

函数名	HAL_I2C_GetState
函数原形	HAL_I2C_StateTypeDef HAL_I2C_GetState(I2C_HandleTypeDef *hi2c)

功能描述	获取 I2C 的通信状态
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	I2C 通信状态
先决条件	无

13.2.40 函数 HAL_I2C_GetMode

描述了函数 HAL_I2C_GetMode

表13-47 函数 HAL_I2C_GetMode

函数名	HAL_I2C_GetMode
函数原形	HAL_I2C_ModeTypeDef HAL_I2C_GetMode(I2C_HandleTypeDef *hi2c)
功能描述	获取 I2C 通信模式
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	I2C 通信模式（主机/从机/存储器/无）
先决条件	无

13.2.41 函数 HAL_I2C_GetError

描述了函数 HAL_I2C_GetError

表13-48 函数 HAL_I2C_GetError

函数名	HAL_I2C_GetError
函数原形	uint32_t HAL_I2C_GetError(I2C_HandleTypeDef *hi2c)
功能描述	获取错误代码
输入参数	hi2c：I2C 句柄
输出参数	无
返回值	错误代码
先决条件	无

14 HAL 独立看门狗通用驱动程序（IWDG）

芯片内集成了一个 Independent watchdog（简称 IWDG），该模块具有安全性高、定时准确及使用灵活的优点。此独立看门狗外设可检测并解决由软件错误导致的故障，并在计数器达到给定的超时值时触发系统复位。

独立看门狗(IWDG)由其专用低速时钟(LSI)驱动，因此即便在主时钟发生故障时仍然保持工作状态。

IWDG 最适合应用于需要看门狗作为一个在主程序之外，能够完全独立工作，并且对时间精度要求较低的应用。

14.1 IWDG 固件驱动寄存器结构

14.1.1 IWDG_InitTypeDef

IWDG_InitTypeDef，定义于文件"air001xx_hal_iwdg.h"如下：

```
typedef struct
{
uint32_t Prescaler;
uint32_t Reload;
} IWDG_InitTypeDef;
```

字段说明：

表14-1 IWDG_InitTypeDef 字段说明

字段	描述
Prescaler	设置 IWDG 预分频值
Reload	设置 IWDG 计数重装载值（0x0000~0x0FFF）

参数说明：

Prescaler 可选参数：

表14-2 Prescaler 可选参数

参数	描述
IWDG_PRESCALER_4	LSI 4 分频
IWDG_PRESCALER_8	LSI 8 分频
IWDG_PRESCALER_16	LSI 16 分频
IWDG_PRESCALER_32	LSI 32 分频
IWDG_PRESCALER_64	LSI 64 分频
IWDG_PRESCALER_128	LSI 128 分频
IWDG_PRESCALER_256	LSI 256 分频

14.1.2 IWDG_HandleTypeDef

IWDG_HandleTypeDef，定义于文件"air001xx_hal_iwdg.h"如下：

```
typedef struct
{
IWDG_TypeDef *Instance;
IWDG_InitTypeDef Init;
} IWDG_HandleTypeDef;
```

字段说明：

表14-3 IWDG_HandleTypeDef 字段说明

字段	描述
Instance	外设基地址
Init	初始化结构体

14.2 IWDG 固件库函数

表14-4 IWDG 固件库函数说明

函数名	描述
HAL_IWDG_Init	初始化 IWDG
HAL_IWDG_Refresh	刷新 IWDG 计数器

14.2.1 函数 HAL_IWDG_Init

描述了函数 HAL_IWDG_Init

表14-5 函数 HAL_IWDG_Init

函数名	HAL_IWDG_Init
函数原形	HAL_StatusTypeDef HAL_IWDG_Init(IWDG_HandleTypeDef *hiwdg)
功能描述	初始化 IWDG
输入参数	hiwdg：IWDG 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

14.2.2 函数 HAL_IWDG_Refresh

描述了函数 HAL_IWDG_Refresh

表14-6 函数 HAL_IWDG_Refresh

函数名	HAL_IWDG_Refresh
函数原形	HAL_StatusTypeDef HAL_IWDG_Refresh(IWDG_HandleTypeDef *hiwdg)
功能描述	刷新 IWDG 计数器

输入参数	hiwdg：IWDG 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

15 HAL 数码管控制器通用驱动程序（LED）

本芯片支持 1~4 个 8 段式共阴极 LED 数码管的控制功能。该控制器通过 4 个支持超大灌电流（80mA/60mA/40mA/20mA）的管脚，输出对应点亮 4 个 7 段式数码管，同一时间只点亮一个数字。

15.1 LED 固件驱动寄存器结构

15.1.1 LED_InitTypeDef

LED_InitTypeDef，定义于文档”air001xx_hal_led.h”如下：

```
typedef struct
{
uint32_t  ComDrive;
uint32_t   Prescaler;
uint32_t   ComNum;
uint32_t  LightTime;
uint32_t DeadTime;
} LED_InitTypeDef;
```

字段说明：

表15-1 LED_InitTypeDef 字段说明

字段	描述
ComDrive	LED 输出驱动能力
Prescaler	LED 时钟预分频值（0x00~0xFF）
ComNum	选择需要打开 LED 的数量
LightTime	LED 点亮时间（1~0xFF）
DeadTime	LED 熄灭时间（0~0xFF）

参数说明：

ComDrive 可选参数：

表15-2 ComDrive 可选参数

参数	描述
LED_COMDRIVE_LOW	弱输出
LED_COMDRIVE_HIGH	强输出

ComNum 可选参数：

表15-3 ComNum 可选参数

参数	描述

0~3	选择开启通道数量（0 为 1 个通道，3 为 4 个通道）

15.1.2 LED_HandleTypeDef

LED_HandleTypeDef，定义于文档”air001xx_hal_led.h”如下：

```
typedef struct
{
LED_TypeDef *Instance;
LED_InitTypeDef Init;
HAL_LockTypeDef Lock;
__IO HAL_LED_StateTypeDef State;
} LED_HandleTypeDef;
```

字段说明：

表15-4 LED_HandleTypeDef 字段说明

字段	描述
Instance	外设基地址
Init	初始化结构体
Lock	HAL 锁
State	LED 状态

15.2 LED 固件库函数

表15-5 LED 固件库函数说明

函数名	描述
HAL_LED_Init	初始化 LED
HAL_LED_MspInit	初始化 LED 相关 MSP
HAL_LED_SetComDisplay	设置指定COM 口的显示值
HAL_LED_LightCompleteCallback	LED 中断回调函数
HAL_LED_IRQHandler	中断请求处理

15.2.1 函数 HAL_LED_Init

描述了函数 HAL_LED_Init

表15-6 函数 HAL_LED_Init

函数名	HAL_LED_Init
函数原形	HAL_StatusTypeDef HAL_LED_Init(LED_HandleTypeDef *hled)
功能描述	初始化 LED
输入参数	hled：LED 句柄
输出参数	无

返回值	HAL 状态
先决条件	无

15.2.2 函数 HAL_LED_MspInit

描述了函数 HAL_LED_MspInit

表15-7 函数 HAL_LED_MspInit

函数名	HAL_LED_MspInit
函数原形	void HAL_LED_MspInit(LED_HandleTypeDef *hled)
功能描述	初始化 LED 相关的 MSP 配置
输入参数	hled：LED 句柄
输出参数	无
返回值	无
先决条件	无

15.2.3 函数 HAL_LED_SetComDisplay

描述了函数 HAL_LED_SetComDisplay

表15-8 函数 HAL_LED_SetComDisplay

函数名	HAL_LED_SetComDisplay
函数原形	HAL_StatusTypeDef HAL_LED_SetComDisplay(LED_HandleTypeDef *hled, uint8_t comCh, uint8_t data)
功能描述	设置指定 COM 口的显示数值
输入参数 1	hled：LED 句柄
输入参数 2	comCh：COM 通道号
输入参数 3	data：需要显示的值
输出参数	无
返回值	HAL 状态
先决条件	无

comCh 可选参数：

表15-9 comCh 可选参数

参数	描述
LED_COM0	选择通道 0
LED_COM1	选择通道 1
LED_COM2	选择通道 2
LED_COM3	选择通道 3
LED_COM_ALL	选择所有通道

data 可选参数：

表15-10　data 可选参数

参数	描述
LED_DISP_NONE	关闭数码管
LED_DISP_FULL	点亮数码管
LED_DISP_0	显示 0
LED_DISP_1	显示 1
LED_DISP_2	显示 2
LED_DISP_3	显示 3
LED_DISP_4	显示 4
LED_DISP_5	显示 5
LED_DISP_6	显示 6
LED_DISP_7	显示 7
LED_DISP_8	显示 8
LED_DISP_9	显示 9
LED_DISP_A	显示 A
LED_DISP_B	显示 B
LED_DISP_C	显示 C
LED_DISP_D	显示 D
LED_DISP_E	显示 E
LED_DISP_F	显示 F
LED_DISP_H	显示 H
LED_DISP_P	显示 P
LED_DISP_U	显示 U
LED_DISP_DOT	显示 .

15.2.4 函数 HAL_LED_LightCompleteCallback

描述了函数 HAL_LED_LightCompleteCallback

表15-11　函数 HAL_LED_LightCompleteCallback

函数名	HAL_LED_LightCompleteCallback
函数原形	void　HAL_LED_LightCompleteCallback(LED_HandleTypeDef　*hled)
功能描述	LED 回调函数
输入参数	hled：LED 句柄
输出参数	无
返回值	无
先决条件	无

15.2.5 函数 HAL_LED_IRQHandler

描述了函数 HAL_LED_IRQHandler

表15-12　函数 HAL_LED_IRQHandler

函数名	HAL_LED_IRQHandler
函数原形	void　HAL_LED_IRQHandler(LED_HandleTypeDef　*hled)
功能描述	LED 中断请求处理
输入参数	hled：LED 句柄
输出参数	无
返回值	无
先决条件	无

16 HAL 低功耗定时器通用驱动程序（LPTIM）

LPTIM 是一款 16 位定时器。LPTIM 将系统从低功耗模式中唤醒的能力使得它适合于实现低功耗应用。

LPTIM 引入了一种灵活的时钟方案，可提供所需的功能和性能，同时将功耗降至最低。

16.1 LPTIM 固件驱动寄存器结构

16.1.1 LPTIM_InitTypeDef

LPTIM_InitTypeDef，定义于文件"air001xx_hal_lptim.h"如下：

```
typedef struct
{
uint32_t Prescaler;
uint32_t UpdateMode;
} LPTIM_InitTypeDef;
```

字段说明：

表16-1 LPTIM_InitTypeDef 字段说明

字段	描述
Prescaler	LPTIM 时钟预分频值
UpdateMode	重装载值更新模式

参数说明：

Prescaler 可选参数：

表16-2 Prescaler 可选参数

参数	描述
LPTIM_PRESCALER_DIV1	1 分频
LPTIM_PRESCALER_DIV2	2 分频
LPTIM_PRESCALER_DIV4	4 分频
LPTIM_PRESCALER_DIV8	8 分频
LPTIM_PRESCALER_DIV16	16 分频
LPTIM_PRESCALER_DIV32	32 分频
LPTIM_PRESCALER_DIV64	64 分频
LPTIM_PRESCALER_DIV128	128 分频

UpdateMode 可选参数：

表16-3 UpdateMode 可选参数

参数	描述

LPTIM_UPDATE_IMMEDIATE	立即更新重装载值
LPTIM_UPDATE_ENDOFPERIOD	当前周期结束后更新重装载值

16.1.2 LPTIM_HandleTypeDef

LPTIM_HandleTypeDef，定义于文件"air001xx_hal_lptim.h"如下：

```
typedef struct
{
LPTIM_TypeDef *Instance;
LPTIM_InitTypeDef Init;
HAL_LockTypeDef Lock;
__IO    HAL_LPTIM_StateTypeDef  State;
} LPTIM_HandleTypeDef;
```

字段说明：

表16-4 LPTIM_HandleTypeDef 字段说明

字段	描述
Instance	外设基地址
Init	初始化参数结构体
Lock	HAL 锁
State	LPTIM 状态

16.2 LPTIM 固件库函数

表16-5 LPTIM 固件库函数说明

函数名	描述
HAL_LPTIM_Init	初始化 LPTIM
HAL_LPTIM_DeInit	将 LPTIM 配置设为缺省值
HAL_LPTIM_MspInit	初始化 LPTIM 相关的 MSP
HAL_LPTIM_MspDeInit	将 LPTIM 相关的 MSP 配置设为缺省值
HAL_LPTIM_SetOnce_Start	启动 LPTIM 计数模式
HAL_LPTIM_SetOnce_Stop	停止 LPTIM 计数模式
HAL_LPTIM_SetOnce_Start_IT	启动 LPTIM 计数模式并开启中断
HAL_LPTIM_SetOnce_Stop_IT	停止 LPTIM 计数模式并关闭中断
HAL_LPTIM_ReadCounter	返回当前 LPTIM 计数值
HAL_LPTIM_ReadAutoReload	返回自动重装载值
HAL_LPTIM_IRQHandler	中断请求处理

HAL_LPTIM_AutoReloadMatchCallback	自动重装载回调函数
HAL_LPTIM_GetState	获取 LPTIM 状态

16.2.1 函数 HAL_LPTIM_Init

描述了函数 HAL_LPTIM_Init

表16-6 函数 HAL_LPTIM_Init

函数名	HAL_LPTIM_Init
函数原形	HAL_StatusTypeDef　HAL_LPTIM_Init(LPTIM_HandleTypeDef　*hlptim)
功能描述	初始化 LPTIM
输入参数	hlptim：LPTIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

16.2.2 函数 HAL_LPTIM_DeInit

描述了函数 HAL_LPTIM_DeInit

表16-7 函数 HAL_LPTIM_DeInit

函数名	HAL_LPTIM_DeInit
函数原形	HAL_StatusTypeDef　HAL_LPTIM_DeInit(LPTIM_HandleTypeDef　*hlptim)
功能描述	将 LPTIM 配置设为缺省值
输入参数	hlptim：LPTIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

16.2.3 函数 HAL_LPTIM_MspInit

描述了函数 HAL_LPTIM_MspInit

表16-8 函数 HAL_LPTIM_MspInit

函数名	HAL_LPTIM_MspInit
函数原形	void HAL_LPTIM_MspInit(LPTIM_HandleTypeDef *hlptim)
功能描述	初始化 LPTIM 相关的 MSP
输入参数	hlptim：LPTIM 句柄
输出参数	无
返回值	无
先决条件	无

16.2.4 函数 HAL_LPTIM_MspDeInit

描述了函数 HAL_LPTIM_MspDeInit

表16-9 函数 HAL_LPTIM_MspDeInit

函数名	HAL_LPTIM_MspDeInit
函数原形	void HAL_LPTIM_MspDeInit(LPTIM_HandleTypeDef *hlptim)
功能描述	将 LPTIM 相关的 MSP 配置设为缺省值
输入参数	hlptim：LPTIM 句柄
输出参数	无
返回值	无
先决条件	无

16.2.5 函数 HAL_LPTIM_SetOnce_Start

描述了函数 HAL_LPTIM_SetOnce_Start

表16-10 函数 HAL_LPTIM_SetOnce_Start

函数名	HAL_LPTIM_SetOnce_Start
函数原形	HAL_StatusTypeDef HAL_LPTIM_SetOnce_Start(LPTIM_HandleTypeDef *hlptim, uint32_t Period)
功能描述	启动 LPTIM 计数模式
输入参数 1	hlptim：LPTIM 句柄
输入参数 2	Period：自动重装载值
输出参数	无
返回值	HAL 状态
先决条件	无

16.2.6 函数 HAL_LPTIM_SetOnce_Stop

描述了函数 HAL_LPTIM_SetOnce_Stop

表16-11 函数 HAL_LPTIM_SetOnce_Stop

函数名	HAL_LPTIM_SetOnce_Stop
函数原形	HAL_StatusTypeDef HAL_LPTIM_SetOnce_Stop(LPTIM_HandleTypeDef *hlptim)
功能描述	停止 LPTIM 计数
输入参数	hlptim：LPTIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

16.2.7 函数 HAL_LPTIM_SetOnce_Start_IT

描述了函数 HAL_LPTIM_SetOnce_Start_IT

表16-12 函数 HAL_LPTIM_SetOnce_Start_IT

函数名	HAL_LPTIM_SetOnce_Start_IT
函数原形	HAL_StatusTypeDef HAL_LPTIM_SetOnce_Start_IT(LPTIM_HandleTypeDef *hlptim, uint32_t Period)

功能描述	启动 LPTIM 计数模式并开启中断
输入参数 1	hlptim：LPTIM 句柄
输入参数 2	Period：自动重装载值
输出参数	无
返回值	HAL 状态
先决条件	无

16.2.8 函数 HAL_LPTIM_SetOnce_Stop_IT

描述了函数 HAL_LPTIM_SetOnce_Stop_IT

表16-13 函数 HAL_LPTIM_SetOnce_Stop_IT

函数名	HAL_LPTIM_SetOnce_Stop_IT
函数原形	HAL_StatusTypeDef HAL_LPTIM_SetOnce_Stop_IT(LPTIM_HandleTypeDef *hlptim)
功能描述	停止 LPTIM 计数模式并关闭中断
输入参数	hlptim：LPTIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

16.2.9 函数 HAL_LPTIM_ReadCounter

描述了函数 HAL_LPTIM_ReadCounter

表16-14 函数 HAL_LPTIM_ReadCounter

函数名	HAL_LPTIM_ReadCounter
函数原形	uint32_t HAL_LPTIM_ReadCounter(LPTIM_HandleTypeDef *hlptim)
功能描述	获取 LPTIM 当前计数值
输入参数	hlptim：LPTIM 句柄
输出参数	无
返回值	当前计数值
先决条件	无

16.2.10 函数 HAL_LPTIM_ReadAutoReload

描述了函数 HAL_LPTIM_ReadAutoReload

表16-15 函数 HAL_LPTIM_ReadAutoReload

函数名	HAL_LPTIM_ReadAutoReload
函数原形	uint32_t HAL_LPTIM_ReadAutoReload(LPTIM_HandleTypeDef *hlptim)
功能描述	获取 LPTIM 计数重装载值
输入参数	hlptim：LPTIM 句柄
输出参数	无
返回值	计数重装载值

先决条件	无

16.2.11 函数 HAL_LPTIM_IRQHandler

描述了函数 HAL_LPTIM_IRQHandler

表16-16 函数 HAL_LPTIM_IRQHandler

函数名	HAL_LPTIM_IRQHandler
函数原形	void HAL_LPTIM_IRQHandler(LPTIM_HandleTypeDef *hlptim)
功能描述	LPTIM 中断请求处理
输入参数	hlptim：LPTIM 句柄
输出参数	无
返回值	无
先决条件	无

16.2.12 函数 HAL_LPTIM_AutoReloadMatchCallback

描述了函数 HAL_LPTIM_AutoReloadMatchCallback

表16-17 函数 HAL_LPTIM_AutoReloadMatchCallback

函数名	HAL_LPTIM_AutoReloadMatchCallback
函数原形	void HAL_LPTIM_AutoReloadMatchCallback(LPTIM_HandleTypeDef *hlptim)
功能描述	自动重装载中断回调函数
输入参数	hlptim：LPTIM 句柄
输出参数	无
返回值	无
先决条件	无

16.2.13 函数 HAL_LPTIM_GetState

描述了函数 HAL_LPTIM_GetState

表16-18 函数 HAL_LPTIM_GetState

函数名	HAL_LPTIM_GetState
函数原形	HAL_LPTIM_StateTypeDef HAL_LPTIM_GetState(LPTIM_HandleTypeDef *hlptim)
功能描述	获取 LPTIM 状态
输入参数	hlptim：LPTIM 句柄
输出参数	无
返回值	LPTIM 状态
先决条件	无

17 HAL 电源功耗控制通用驱动程序（PWR）

芯片拥有两种低功耗模式Sleep Mode、 Stop Mode，可以配置通过事件或中断唤醒。

17.1 PWR 固件驱动寄存器结构

17.1.1 PWR_PVDTypeDef

PWR_PVDTypeDef，定义于文件”air001xx_hal_pwr.h”如下：

```
typedef struct
{
uint32_t PVDSource;
uint32_t PVDFilter;
uint32_t PVDLevel;
uint32_t Mode;
}PWR_PVDTypeDef;
```

字段说明：

表17-1 PWR_PVDTypeDef 字段说明

字段	描述
PVDSource	配置PVD 检测源
PVDFilter	配置PVD 数字滤波值
PVDLevel	配置PVD 的检测阈值
Mode	配置PVD 信号触发模式

参数说明：

PVDSource 可选参数：

表17-2 PVDSource 可选参数

参数	描述
PWR_PVD_SOURCE_VCC	VCC 作为 PVD 检测源
PWR_PVD_SOURCE_PB07	PB7 作为 PVD 检测源

PVDFilter 可选参数：

表17-3 PVDFilter 可选参数

参数	描述
PWR_PVD_FILTER_NONE	无滤波
PWR_PVD_FILTER_1CLOCK	滤波时间：1 机器周期
PWR_PVD_FILTER_2CLOCK	滤波时间：2 机器周期
PWR_PVD_FILTER_4CLOCK	滤波时间：4 机器周期

PWR_PVD_FILTER_16CLOCK	滤波时间：16 机器周期
PWR_PVD_FILTER_64CLOCK	滤波时间：64 机器周期
PWR_PVD_FILTER_128CLOCK	滤波时间：128 机器周期
PWR_PVD_FILTER_1024CLOCK	滤波时间：1024 机器周期

PVDLevel 可选参数：

表17-4 PVDLevel 可选参数

参数	描述
PWR_PVDLEVEL_0	PVD 检测等级 0 上升沿检测阈值 1.8V，下降沿相应减少 0.1V
PWR_PVDLEVEL_1	PVD 检测等级 1 上升沿检测阈值 2.0V，下降沿相应减少 0.1V
PWR_PVDLEVEL_2	PVD 检测等级 2 上升沿检测阈值 2.2V，下降沿相应减少 0.1V
PWR_PVDLEVEL_3	PVD 检测等级 3 上升沿检测阈值 2.4V，下降沿相应减少 0.1V
PWR_PVDLEVEL_4	PVD 检测等级 4 上升沿检测阈值 2.6V，下降沿相应减少 0.1V
PWR_PVDLEVEL_5	PVD 检测等级 5 上升沿检测阈值 2.8V，下降沿相应减少 0.1V
PWR_PVDLEVEL_6	PVD 检测等级 6 上升沿检测阈值 3.0V，下降沿相应减少 0.1V
PWR_PVDLEVEL_7	PVD 检测等级 7 上升沿检测阈值 3.2V，下降沿相应减少 0.1V

Mode 可选参数：

表17-5 Mode 可选参数

参数	描述
PWR_PVD_MODE_NORMAL	使用普通模式
PWR_PVD_MODE_IT_RISING	使用上升沿检测的外部中断模式
PWR_PVD_MODE_IT_FALLING	使用下降沿检测的外部中断模式
PWR_PVD_MODE_IT_RISING_FALLING	使用上升沿和下降沿检测的外部中断模式
PWR_PVD_MODE_EVENT_RISING	使用上升沿检测的事件模式
PWR_PVD_MODE_EVENT_FALLING	使用下降沿检测的事件模式
PWR_PVD_MODE_EVENT_RISING_FALLING	使用上升沿和下降沿检测的事件模式

17.2 PWR 固件库函数

表17-6 PWR 固件库函数说明

函数名	描述
HAL_PWR_DeInit	将 PWR 配置设为缺省值
HAL_PWR_EnableBkUpAccess	解锁备份域寄存器
HAL_PWR_DisableBkUpAccess	锁定备份域寄存器
HAL_PWR_ConfigPVD	配置PVD
HAL_PWR_EnablePVD	开启PVD
HAL_PWR_DisablePVD	关闭PVD
HAL_PWR_EnterSLEEPMode	进入SLEEP 模式
HAL_PWR_EnterSTOPMode	进入STOP 模式
HAL_PWR_EnableSleepOnExit	开启退出中断时内核进入SLEEP
HAL_PWR_DisableSleepOnExit	关闭退出中断时内核进入SLEEP
HAL_PWR_EnableSEVOnPend	开启中断挂起事件唤醒
HAL_PWR_DisableSEVOnPend	关闭中断挂起事件唤醒
HAL_PWR_PVD_IRQHandler	中断请求处理
HAL_PWR_PVD_Callback	PVD 中断回调函数

17.2.1 函数 HAL_PWR_DeInit

描述了函数 HAL_PWR_DeInit

表17-7 函数 HAL_PWR_DeInit

函数名	HAL_PWR_DeInit
函数原形	void HAL_PWR_DeInit(void)
功能描述	将 PWR 配置设为缺省值
输入参数	无
输出参数	无
返回值	无
先决条件	无

17.2.2 函数 HAL_PWR_EnableBkUpAccess

描述了函数 HAL_PWR_EnableBkUpAccess

表17-8 函数 HAL_PWR_EnableBkUpAccess

函数名	HAL_PWR_EnableBkUpAccess
函数原形	void HAL_PWR_EnableBkUpAccess(void)
功能描述	解锁备份域寄存器

输入参数	无
输出参数	无
返回值	无
先决条件	无

17.2.3 函数 HAL_PWR_DisableBkUpAccess

描述了函数 HAL_PWR_DisableBkUpAccess

表17-9 函数 HAL_PWR_DisableBkUpAccess

函数名	HAL_PWR_DisableBkUpAccess
函数原形	void HAL_PWR_DisableBkUpAccess(void)
功能描述	锁定备份域寄存器
输入参数	无
输出参数	无
返回值	无
先决条件	无

17.2.4 函数 HAL_PWR_ConfigPVD

描述了函数 HAL_PWR_ConfigPVD

表17-10 函数 HAL_PWR_ConfigPVD

函数名	HAL_PWR_ConfigPVD
函数原形	HAL_StatusTypeDef HAL_PWR_ConfigPVD(PWR_PVDTypeDef *sConfigPVD)
功能描述	配置 PVD
输入参数	sConfigPVD：配置参数 结构体
输出参数	无
返回值	HAL 状态
先决条件	无

17.2.5 函数 HAL_PWR_EnablePVD

描述了函数 HAL_PWR_EnablePVD

表17-11 函数 HAL_PWR_EnablePVD

函数名	HAL_PWR_EnablePVD
函数原形	void HAL_PWR_EnablePVD(void)
功能描述	开启 PVD
输入参数	无
输出参数	无
返回值	无
先决条件	无

17.2.6 函数 HAL_PWR_DisablePVD

描述了函数 HAL_PWR_DisablePVD

表17-12 函数 HAL_PWR_DisablePVD

函数名	HAL_PWR_DisablePVD
函数原形	void HAL_PWR_DisablePVD(void)
功能描述	关闭 PVD
输入参数	无
输出参数	无
返回值	无
先决条件	无

17.2.7 函数 HAL_PWR_EnterSLEEPMode

描述了函数 HAL_PWR_EnterSLEEPMode

表17-13 函数 HAL_PWR_EnterSLEEPMode

函数名	HAL_PWR_EnterSLEEPMode
函数原形	void HAL_PWR_EnterSLEEPMode(uint8_t SLEEPEntry)
功能描述	进入 SLEEP 模式
输入参数 1	SLEEPEntry：选择唤醒 SLEEP 模式的方式
输出参数	无
返回值	无
先决条件	无

SLEEPEntry 可选参数：

表17-14 SLEEPEntry 可选参数

参数	描述
WFI	使用中断唤醒 SLEEP 模式
WFE	使用事件唤醒 SLEEP 模式

17.2.8 函数 HAL_PWR_EnterSTOPMode

描述了函数 HAL_PWR_EnterSTOPMode

表17-15 函数 HAL_PWR_EnterSTOPMode

函数名	HAL_PWR_EnterSTOPMode
函数原形	void HAL_PWR_EnterSTOPMode(uint32_t Regulator, uint8_t STOPEntry)
功能描述	进入 STOP 模式
输入参数 1	Regulator：选择电源调节器模式
输入参数 2	STOPEntry：选择唤醒 STOP 模式的方式
输出参数	无
返回值	无

先决条件	无

Regulator 可选参数：

表17-16 Regulator 可选参数

参数	描述
PWR_MAINREGULATOR_ON	主调节器
PWR_LOWPOWERREGULATOR_ON	低功率调节器

STOPEntry 可选参数：

表17-17 STOPEntry 可选参数

参数	描述
WFI	使用中断唤醒 STOP 模式
WFE	使用事件唤醒 STOP 模式

17.2.9 函数 HAL_PWR_EnableSleepOnExit

描述了函数 HAL_PWR_EnableSleepOnExit

函数名	HAL_PWR_EnableSleepOnExit
函数原形	void HAL_PWR_EnableSleepOnExit(void)
功能描述	开启退出中断时内核进入 SLEEP
输入参数	无
输出参数	无
返回值	无
先决条件	无

17.2.10 函数 HAL_PWR_DisableSleepOnExit

描述了函数 HAL_PWR_DisableSleepOnExit

表17-18 函数 HAL_PWR_DisableSleepOnExit

函数名	HAL_PWR_DisableSleepOnExit
函数原形	void HAL_PWR_DisableSleepOnExit(void)
功能描述	关闭退出中断时内核进入 SLEEP
输入参数	无
输出参数	无
返回值	无
先决条件	无

17.2.11 函数 HAL_PWR_EnableSEVOnPend

描述了函数 HAL_PWR_EnableSEVOnPend

表17-19 函数 HAL_PWR_EnableSEVOnPend

函数名	HAL_PWR_EnableSEVOnPend
函数原形	void HAL_PWR_EnableSEVOnPend(void)

功能描述	开启中断挂起事件唤醒
输入参数	无
输出参数	无
返回值	无
先决条件	无

17.2.12 函数 HAL_PWR_DisableSEVOnPend

描述了函数 HAL_PWR_DisableSEVOnPend

表17-20 函数 HAL_PWR_DisableSEVOnPend

函数名	HAL_PWR_DisableSEVOnPend
函数原形	void HAL_PWR_DisableSEVOnPend(void)
功能描述	关闭中断挂起事件唤醒
输入参数	无
输出参数	无
返回值	无
先决条件	无

17.2.13 函数 HAL_PWR_PVD_IRQHandler

描述了函数 HAL_PWR_PVD_IRQHandler

表17-21 函数 HAL_PWR_PVD_IRQHandler

函数名	HAL_PWR_PVD_IRQHandler
函数原形	void HAL_PWR_PVD_IRQHandler(void)
功能描述	PVD 中断请求处理
输入参数	无
输出参数	无
返回值	无
先决条件	无

17.2.14 函数 HAL_PWR_PVD_Callback

描述了函数 HAL_PWR_PVD_Callback

表17-22 函数 HAL_PWR_PVD_Callback

函数名	HAL_PWR_PVD_Callback
函数原形	void HAL_PWR_PVD_Callback(void)
功能描述	PVD 中断回调函数
输入参数	无
输出参数	无
返回值	无
先决条件	无

18 HAL 复位和时钟通驱动程序（RCC）

复位一共有三种类型的复位，分别为系统复位、电源复位和 RTC 域复位。

芯片提供以下时钟源，可以产生主时钟：

- 外部高速时钟 HSE
- 外部低速时钟 LSE
- 内部高速时钟 HIS
- 内部低速时钟 LSI
- PLL

18.1 RCC 固件驱动寄存器结构

18.1.1 RCC_PLLInitTypeDef

RCC_PLLInitTypeDef，定义于文件"air001xx_hal_rcc.h"如下：

```
typedef struct
{
uint32_t PLLState;
uint32_t PLLSource;
} RCC_PLLInitTypeDef;
```

字段说明：

表18-1 RCC_PLLInitTypeDef 字段说明

字段	描述
PLLState	配置PLL 状态
PLLSource	配置PLL 时钟源

参数说明：

PLLState 可选参数：

表18-2 PLLState 可选参数

参数	描述
RCC_PLL_NONE	不修改 PLL
RCC_PLL_OFF	关闭 PLL
RCC_PLL_ON	启动 PLL

PLLSource 可选参数：

表18-3 PLLSource 可选参数

参数	描述

RCC_PLLSOURCE_NONE	不选择时钟源
RCC_PLLSOURCE_HSI	选择 HSI 作为 PLL 时钟源
RCC_PLLSOURCE_HSE	选择 HSE 作为 PLL 时钟源

18.1.2 RCC_OscInitTypeDef

RCC_OscInitTypeDef，定义于文件"air001xx_hal_rcc.h"如下：

```
typedef struct
{
uint32_t OscillatorType;
uint32_t HSEState;
uint32_t HSEFreq;
uint32_t LSEState;
uint32_t LSEDriver;
uint32_t HSIState;
uint32_t HSIDiv;
uint32_t HSICalibrationValue;
uint32_t LSIState;
RCC_PLLInitTypeDef PLL;
} RCC_OscInitTypeDef;
```

字段说明：

表18-4 RCC_OscInitTypeDef 字段说明

字段	描述
OscillatorType	需要配置的振荡器
HSEState	HSE 状态
HSEFreq	HSE 频率
LSEState	LSE 状态
LSEDriver	LSE 驱动能力
HSIState	HSI 状态
HSIDiv	HSI 分频值
HSICalibrationValue	HSI 校准频率
LSIState	LSI 状态
PLL	PLL 配置结构体

参数说明：

OscillatorType 可选参数：

表18-5 OscillatorType 可选参数

参数	描述
RCC_OSCILLATORTYPE_NONE	不配置
RCC_OSCILLATORTYPE_HSE	配置 HSE
RCC_OSCILLATORTYPE_HSI	配置 HSI
RCC_OSCILLATORTYPE_LSE	配置 LSE
RCC_OSCILLATORTYPE_LSI	配置 LSI

HSEState 可选参数：

表18-6 HSEState 可选参数

参数	描述
RCC_HSE_OFF	关闭 HSE
RCC_HSE_ON	打开 HSE
RCC_HSE_BYPASS	使用外接时钟源

HSEFreq 可选参数：

表18-7 HSEFreq 可选参数

参数	描述
RCC_HSE_4_8MHz	HSE 频率为 4~8Mhz
RCC_HSE_8_16MHz	HSE 频率为 8~16Mhz
RCC_HSE_16_32MHz	HSE 频率为 16~32Mhz

LSEState 可选参数：

表18-8 LSEState 可选参数

参数	描述
RCC_LSE_OFF	关闭 LSE
RCC_LSE_ON	打开 LSE
RCC_LSE_BYPASS	使用外接时钟源

LSEDriver 可选参数：

表18-9 LSEDriver 可选参数

参数	描述
RCC_LSEDRIVE_LOW	LSE 低驱动能力
RCC_LSEDRIVE_MEDIUM	LSE 中驱动能力
RCC_LSEDRIVE_HIGH	LSE 高驱动能力

HSIState 可选参数：

表18-10 HSIState 可选参数

参数	描述
RCC_HSI_OFF	关闭 HSI
RCC_HSI_ON	打开 HSI

HSIDiv 可选参数：

表18-11 HSIDiv 可选参数

参数	描述
RCC_HSI_DIV1	HSI 1 分频
RCC_HSI_DIV2	HSI 2 分频
RCC_HSI_DIV4	HSI 4 分频
RCC_HSI_DIV8	HSI 8 分频
RCC_HSI_DIV16	HSI 16 分频
RCC_HSI_DIV32	HSI 32 分频
RCC_HSI_DIV64	HSI 64 分频
RCC_HSI_DIV128	HSI 128 分频

HSICalibrationValue 可选参数：

表18-12 HSICalibrationValue 可选参数

参数	描述
RCC_HSICALIBRATION_4MHz	校准频率：4MHz
RCC_HSICALIBRATION_8MHz	校准频率：8MHz
RCC_HSICALIBRATION_16MHz	校准频率：16MHz
RCC_HSICALIBRATION_22p12MHz	校准频率：22.12MHz
RCC_HSICALIBRATION_24MHz	校准频率：24MHz

LSIState 可选参数：

表18-13 LSIState 可选参数

参数	描述
RCC_LSI_OFF	关闭 LSI
RCC_LSI_ON	打开 LSI

18.1.3 RCC_ClkInitTypeDef

RCC_ClkInitTypeDef，定义于文件”air001xx_hal_rcc.h”如下：

```
typedef struct
{
uint32_t ClockType;
uint32_t SYSCLKSource;
```

```
uint32_t AHBCLKDivider;
uint32_t APB1CLKDivider;
} RCC_ClkInitTypeDef;
```

字段说明：

表18-14 RCC_ClkInitTypeDef 字段说明

字段	描述
ClockType	要配置的时钟
SYSCLKSource	系统时钟源
AHBCLKDivider	AHB 时钟源
APB1CLKDivider	APB 时钟源

参数说明：

ClockType 可选参数：

表18-15 ClockType 可选参数

参数	描述
RCC_CLOCKTYPE_SYSCLK	配置系统时钟
RCC_CLOCKTYPE_HCLK	配置 AHB 时钟
RCC_CLOCKTYPE_PCLK1	配置 APB 时钟

SYSCLKSource 可选参数：

表18-16 SYSCLKSource 可选参数

参数	描述
RCC_SYSCLKSOURCE_HSI	配置 HSI 为系统时钟源
RCC_SYSCLKSOURCE_HSE	配置 HSE 为系统时钟源
RCC_SYSCLKSOURCE_PLLCLK	配置 PLL 为系统时钟源
RCC_SYSCLKSOURCE_LSI	配置 LSI 为系统时钟源
RCC_SYSCLKSOURCE_LSE	配置 LSE 为系统时钟源

AHBCLKDivider 可选参数：

表18-17 AHBCLKDivider 可选参数

参数	描述
RCC_SYSCLK_DIV1	系统时钟 1 分频作为 AHB 时钟源
RCC_SYSCLK_DIV2	系统时钟 2 分频作为 AHB 时钟源
RCC_SYSCLK_DIV4	系统时钟 4 分频作为 AHB 时钟源
RCC_SYSCLK_DIV8	系统时钟 8 分频作为 AHB 时钟源
RCC_SYSCLK_DIV16	系统时钟 16 分频作为 AHB 时钟源

RCC_SYSCLK_DIV64	系统时钟 64 分频作为 AHB 时钟源
RCC_SYSCLK_DIV128	系统时钟 128 分频作为 AHB 时钟源
RCC_SYSCLK_DIV256	系统时钟 256 分频作为 AHB 时钟源
RCC_SYSCLK_DIV512	系统时钟 512 分频作为 AHB 时钟源

APB1CLKDivider 可选参数:

表18-18 APB1CLKDivider 可选参数

参数	描述
RCC_HCLK_DIV1	AHB 1 分频作为 APB 时钟源
RCC_HCLK_DIV2	AHB 2 分频作为 APB 时钟源
RCC_HCLK_DIV4	AHB 4 分频作为 APB 时钟源
RCC_HCLK_DIV8	AHB 8 分频作为 APB 时钟源
RCC_HCLK_DIV16	AHB 16 分频作为 APB 时钟源

18.2 RCC 固件库函数

表18-19 RCC 固件库函数说明

函数名	描述
HAL_RCC_DeInit	将 RCC 配置设为缺省值
HAL_RCC_OscConfig	配置振荡器
HAL_RCC_ClockConfig	配置时钟和 FLASH 访问延迟
HAL_RCC_MCOConfig	配置 MCO
HAL_RCC_EnableCSS	使能CSS 功能
HAL_RCC_EnableLSECSS	使能 LSECSS 功能
HAL_RCC_DisableLSECSS	关闭 LSECSS 功能
HAL_RCC_GetSysClockFreq	获取系统时钟频率
HAL_RCC_GetHCLKFreq	获取AHP 时钟频率
HAL_RCC_GetPCLK1Freq	获取APB 时钟频率
HAL_RCC_GetOscConfig	获取振荡器配置
HAL_RCC_GetClockConfig	获取时钟配置
HAL_RCC_NMI_IRQHandler	不可屏蔽中断请求处理
HAL_RCC_CSSCallback	CSS 中断回调函数
HAL_RCC_LSECSSCallback	LSECSS 中断回调函数

18.2.1 函数 HAL_RCC_DeInit

描述了函数 HAL_RCC_DeInit

表18-20 函数 HAL_RCC_DeInit

函数名	HAL_RCC_DeInit
函数原形	HAL_StatusTypeDef HAL_RCC_DeInit(void)
功能描述	将 RCC 配置设为缺省值
输入参数	无
输出参数	无
返回值	HAL 状态
先决条件	无

18.2.2 函数 HAL_RCC_OscConfig

描述了函数 HAL_RCC_OscConfig

表18-21 函数 HAL_RCC_OscConfig

函数名	HAL_RCC_OscConfig
函数原形	HAL_StatusTypeDef HAL_RCC_OscConfig(RCC_OscInitTypeDef *RCC_OscInitStruct)
功能描述	配置振荡器
输入参数	RCC_OscInitStruct：振荡器配置参数结构体
输出参数	无
返回值	HAL 状态
先决条件	无

18.2.3 函数 HAL_RCC_ClockConfig

描述了函数 HAL_RCC_ClockConfig

表18-22 函数 HAL_RCC_ClockConfig

函数名	HAL_RCC_ClockConfig
函数原形	HAL_StatusTypeDef HAL_RCC_ClockConfig(RCC_ClkInitTypeDef *RCC_ClkInitStruct, uint32_t FLatency)
功能描述	配置时钟和 FLASH 访问延迟
输入参数 1	RCC_ClkInitStruct：时钟配置参数结构体
输入参数 2	FLatency：FLASH 访问延迟
输出参数	无
返回值	HAL 状态
先决条件	无

FLatency 可选参数：

表18-23 FLatency 可选参数

参数	描述
FLASH_LATENCY_0	关闭访问延迟
FLASH_LATENCY_1	开启访问延迟

18.2.4 函数 HAL_RCC_MCOConfig

描述了函数 HAL_RCC_MCOConfig

表18-24 函数 HAL_RCC_MCOConfig

函数名	HAL_RCC_MCOConfig
函数原形	void HAL_RCC_MCOConfig(uint32_t RCC_MCOx, uint32_t RCC_MCOSource, uint32_t RCC_MCODiv)
功能描述	配置 MCO
输入参数 1	RCC_MCOx：时钟源的输出管脚
输入参数 2	RCC_MCOSource：要输出的时钟源
输入参数 3	RCC_MCODiv：MCO 输出分频
输出参数	无
返回值	无
先决条件	无

RCC_MCOx 可选参数：

表18-25 RCC_MCOx 可选参数

参数	描述
RCC_MCO1	输出到 MCO1（PA8）
RCC_MCO2	输出到 MCO2（PA1）
RCC_MCO3	输出到 MCO3（PA5）
RCC_MCO4	输出到 MCO4（PA9）
RCC_MCO5	输出到 MCO5（PA13）
RCC_MCO6	输出到 MCO6（PA14）
RCC_MCO7	输出到 MCO7（PF2）

RCC_MCOSource 可选参数：

表18-26 RCC_MCOSource 可选参数

参数	描述
RCC_MCO1SOURCE_NOCLOCK	无时钟输出
RCC_MCO1SOURCE_SYSCLK	输出系统时钟
RCC_MCO1SOURCE_HSI	输出 HSI 时钟
RCC_MCO1SOURCE_HSE	输出 HSE 时钟
RCC_MCO1SOURCE_PLLCLK	输出 PLL 时钟
RCC_MCO1SOURCE_LSI	输出 LSI 时钟
RCC_MCO1SOURCE_LSE	输出 LSE 时钟

RCC_MCODiv 可选参数：

表18-27 RCC_MCODiv 可选参数

参数	描述
RCC_MCODIV_1	MCO 输出无分频
RCC_MCODIV_2	MCO 输出 2 分频
RCC_MCODIV_4	MCO 输出 4 分频
RCC_MCODIV_8	MCO 输出 8 分频
RCC_MCODIV_16	MCO 输出 16 分频
RCC_MCODIV_32	MCO 输出 32 分频
RCC_MCODIV_64	MCO 输出 64 分频
RCC_MCODIV_128	MCO 输出 128 分频

18.2.5 函数 HAL_RCC_EnableCSS

描述了函数 HAL_RCC_EnableCSS

表18-28 函数 HAL_RCC_EnableCSS

函数名	HAL_RCC_EnableCSS
函数原形	void HAL_RCC_EnableCSS(void)
功能描述	开启 CSS 功能
输入参数	无
输出参数	无
返回值	无
先决条件	无

18.2.6 函数 HAL_RCC_EnableLSECSS

描述了函数 HAL_RCC_EnableLSECSS

表18-29 函数 HAL_RCC_EnableLSECSS

函数名	HAL_RCC_EnableLSECSS
函数原形	void HAL_RCC_EnableLSECSS(void)
功能描述	开启 LSECSS 功能
输入参数	无
输出参数	无
返回值	无
先决条件	无

18.2.7 函数 HAL_RCC_DisableLSECSS

描述了函数 HAL_RCC_DisableLSECSS

表18-30 函数 HAL_RCC_DisableLSECSS

函数名	HAL_RCC_DisableLSECSS
函数原形	void HAL_RCC_DisableLSECSS(void)

功能描述	关闭 LSECSS 功能
输入参数	无
输出参数	无
返回值	无
先决条件	无

18.2.8 函数 HAL_RCC_GetSysClockFreq

描述了函数 HAL_RCC_GetSysClockFreq

表18-31 函数 HAL_RCC_GetSysClockFreq 2

函数名	HAL_RCC_GetSysClockFreq
函数原形	uint32_t HAL_RCC_GetSysClockFreq(void)
功能描述	获取系统时钟频率
输入参数	无
输出参数	无
返回值	系统时钟频率
先决条件	无

18.2.9 函数 HAL_RCC_GetHCLKFreq

描述了函数 HAL_RCC_GetHCLKFreq

表18-32 函数 HAL_RCC_GetHCLKFreq

函数名	HAL_RCC_GetHCLKFreq
函数原形	uint32_t HAL_RCC_GetHCLKFreq(void)
功能描述	获取 AHB 时钟频率
输入参数	无
输出参数	无
返回值	AHB 时钟频率
先决条件	无

18.2.10 函数 HAL_RCC_GetPCLK1Freq

描述了函数 HAL_RCC_GetPCLK1Freq

表18-33 函数 HAL_RCC_GetPCLK1Freq

函数名	HAL_RCC_GetPCLK1Freq
函数原形	uint32_t HAL_RCC_GetPCLK1Freq(void)
功能描述	获取 APB 时钟频率
输入参数	无
输出参数	无
返回值	APB 时钟频率
先决条件	无

18.2.11 函数 HAL_RCC_GetOscConfig

描述了函数 HAL_RCC_GetOscConfig

表18-34 函数 HAL_RCC_GetOscConfig

函数名	HAL_RCC_GetOscConfig
函数原形	void HAL_RCC_GetOscConfig(RCC_OscInitTypeDef * RCC_OscInitStruct)
功能描述	获取振荡器配置
输入参数	RCC_OscInitStruct：振荡器配置参数结构体
输出参数	RCC_OscInitStruct：振荡器配置参数结构体
返回值	无
先决条件	无

18.2.12 函数 HAL_RCC_GetClockConfig

描述了函数 HAL_RCC_GetClockConfig

表18-35 函数 HAL_RCC_GetClockConfig

函数名	HAL_RCC_GetClockConfig
函数原形	void HAL_RCC_GetClockConfig(RCC_ClkInitTypeDef *RCC_ClkInitStruct, uint32_t *pFLatency)
功能描述	获取时钟配置
输入参数 1	RCC_ClkInitStruct：时钟配置参数结构体
输入参数 2	pFLatency：FLASH 延迟参数指针
输出参数 1	RCC_ClkInitStruct：时钟配置参数结构体
输出参数 2	pFLatency：FLASH 延迟参数指针
返回值	无
先决条件	无

18.2.13 函数 HAL_RCC_NMI_IRQHandler

描述了函数 HAL_RCC_NMI_IRQHandler

表18-36 函数 HAL_RCC_NMI_IRQHandler

函数名	HAL_RCC_NMI_IRQHandler
函数原形	void HAL_RCC_NMI_IRQHandler(void)
功能描述	不可屏蔽中断，中断请求处理
输入参数	无
输出参数	无
返回值	无
先决条件	无

18.2.14 函数 HAL_RCC_CSSCallback

描述了函数 HAL_RCC_CSSCallback

表18-37　函数 HAL_RCC_CSSCallback

函数名	HAL_RCC_CSSCallback
函数原形	void HAL_RCC_CSSCallback(void)
功能描述	CSS 回调函数
输入参数	无
输出参数	无
返回值	无
先决条件	无

18.2.15 函数 HAL_RCC_LSECSSCallback

描述了函数 HAL_RCC_LSECSSCallback

表18-38　函数 HAL_RCC_LSECSSCallback

函数名	HAL_RCC_LSECSSCallback
函数原形	void HAL_RCC_LSECSSCallback(void)
功能描述	LSECSS 回调函数
输入参数	无
输出参数	无
返回值	无
先决条件	无

19 HAL 复位和时钟扩展驱动程序（RCC_Ex）

19.1 RCC_Ex 固件驱动寄存器结构

19.1.1 RCC_PeriphCLKInitTypeDef

RCC_PeriphCLKInitTypeDef，定义于文件"air001xx_hal_rcc_ex.h"如下：

```
typedef struct
{
uint32_t PeriphClockSelection;
uint32_t PvdClockSelection;
uint32_t Comp1ClockSelection;
uint32_t Comp2ClockSelection;
uint32_t LptimClockSelection;
uint32_t RTCClockSelection;
} RCC_PeriphCLKInitTypeDef;
```

字段说明：

表19-1 RCC_PeriphCLKInitTypeDef 字段说明

字段	描述
PeriphClockSelection	需要配置的外设时钟源
PvdClockSelection	配置 PVD 时钟源
Comp1ClockSelection	配置 COMP1 时钟源
Comp2ClockSelection	配置 COMP2 时钟源
LptimClockSelection	配置 LPTIM 时钟源
RTCClockSelection	配置 RTC 时钟源

参数说明：

PeriphClockSelection 可选参数：

表19-2 PeriphClockSelection 可选参数

参数	描述
RCC_PERIPHCLK_PVD	配置 PVD 时钟源
RCC_PERIPHCLK_COMP1	配置 COMP1 时钟源
RCC_PERIPHCLK_COMP2	配置 COMP2 时钟源
RCC_PERIPHCLK_LPTIM	配置 LPTIM 时钟源
RCC_PERIPHCLK_RTC	配置 RTC 时钟源

PvdClockSelection 可选参数：

表19-3 PvdClockSelection 可选参数

参数	描述
RCC_PVDCLKSOURCE_PCLK	选择 APB 时钟作为时钟源
RCC_PVDCLKSOURCE_LSC	选择 LSC（低速时钟）作为时钟源

Comp1ClockSelection 可选参数：

表19-4 Comp1ClockSelection 可选参数

参数	描述
RCC_COMP1CLKSOURCE_PCLK	选择 APB 时钟作为时钟源
RCC_COMP1CLKSOURCE_LSC	选择 LSC（低速时钟）作为时钟源

Comp2ClockSelection 可选参数：

表19-5 Comp2ClockSelection 可选参数

参数	描述
RCC_COMP2CLKSOURCE_PCLK	选择 APB 时钟作为时钟源
RCC_COMP2CLKSOURCE_LSC	选择 LSC（低速时钟）作为时钟源

LptimClockSelection 可选参数：

表19-6 LptimClockSelection 可选参数

参数	描述
RCC_LPTIMCLKSOURCE_PCLK1	选择 APB 时钟作为时钟源
RCC_LPTIMCLKSOURCE_LSI	选择 LSI 时钟作为时钟源
RCC_LPTIMCLKSOURCE_LSE	选择 LSE 时钟作为时钟源

RTCClockSelection 可选参数：

表19-7 RTCClockSelection 可选参数

参数	描述
RCC_RTCCLKSOURCE_NONE	不配置时钟
RCC_RTCCLKSOURCE_LSE	选择 LSE 作为时钟源
RCC_RTCCLKSOURCE_LSI	选择 LSI 作为时钟源
RCC_RTCCLKSOURCE_HSE_DIV128	选择 HSE128 分频作为时钟源

19.2 RCC_Ex 固件库函数

表19-8 RCC_Ex 固件库函数说明

函数名	描述
HAL_RCCEx_PeriphCLKConfig	配置外设时钟
HAL_RCCEx_GetPeriphCLKConfig	获取外设时钟配置信息

HAL_RCCEx_GetPeriphCLKFreq	获取外设时钟频率
HAL_RCCEx_EnableLSCO	开启低速时钟
HAL_RCCEx_DisableLSCO	关闭低速时钟

19.2.1 函数 HAL_RCCEx_PeriphCLKConfig

描述了函数 HAL_RCCEx_PeriphCLKConfig

表19-9 函数 HAL_RCCEx_PeriphCLKConfig

函数名	HAL_RCCEx_PeriphCLKConfig
函数原形	HAL_StatusTypeDef HAL_RCCEx_PeriphCLKConfig(RCC_PeriphCLKInitTypeDef *PeriphClkInit)
功能描述	配置外设时钟
输入参数	PeriphClkInit：外设时钟初始化配置结构体
输出参数	无
返回值	HAL 状态
先决条件	无

19.2.2 函数 HAL_RCCEx_GetPeriphCLKConfig

描述了函数 HAL_RCCEx_GetPeriphCLKConfig

表19-10 函数 HAL_RCCEx_GetPeriphCLKConfig

函数名	HAL_RCCEx_GetPeriphCLKConfig
函数原形	void HAL_RCCEx_GetPeriphCLKConfig(RCC_PeriphCLKInitTypeDef *PeriphClkInit)
功能描述	获取外设时钟配置参数，并保存在 PeriphClkInit 中
输入参数	PeriphClkInit：外设时钟初始化配置结构体
输出参数	PeriphClkInit
返回值	无
先决条件	无

19.2.3 函数 HAL_RCCEx_GetPeriphCLKFreq

描述了函数 HAL_RCCEx_GetPeriphCLKFreq

表19-11 函数 HAL_RCCEx_GetPeriphCLKFreq

函数名	HAL_RCCEx_GetPeriphCLKFreq
函数原形	uint32_t HAL_RCCEx_GetPeriphCLKFreq(uint32_t PeriphClk)
功能描述	获取外设时钟频率
输入参数	PeriphClk：指定外设
输出参数	无
返回值	时钟频率
先决条件	无

PeriphClk 可选参数：

表19-12 PeriphClk 可选参数

参数	描述
RCC_PERIPHCLK_RTC	RTC 时钟频率
RCC_PERIPHCLK_PVD	PVD 时钟频率
RCC_PERIPHCLK_COMP1	COMP1 时钟频率
RCC_PERIPHCLK_COMP2	COMP2 时钟频率
RCC_PERIPHCLK_LPTIM	LPTIM 时钟频率

19.2.4 函数 HAL_RCCEx_EnableLSCO

描述了函数 HAL_RCCEx_EnableLSCO

表19-13 函数 HAL_RCCEx_EnableLSCO

函数名	HAL_RCCEx_EnableLSCO
函数原形	void HAL_RCCEx_EnableLSCO(uint32_t LSCOSource)
功能描述	开启低速时钟
输入参数	LSCOSource：低速时钟时钟源
输出参数	无
返回值	无
先决条件	无

LSCOSource 可选参数：

表19-14 LSCOSource 可选参数

参数	描述
RCC_LSCOSOURCE_LSI	选择 LSI 作为时钟源
RCC_LSCOSOURCE_LSE	选择 LSE 作为时钟源

19.2.5 函数 HAL_RCCEx_DisableLSCO

描述了函数 HAL_RCCEx_DisableLSCO

表19-15 函数 HAL_RCCEx_DisableLSCO

函数名	HAL_RCCEx_DisableLSCO
函数原形	void HAL_RCCEx_DisableLSCO(void)
功能描述	关闭低速时钟
输入参数	无
输出参数	无
返回值	无
先决条件	无

20 HAL 实时时钟通用驱动程序（RTC）

实时时钟（real time clock）是一个独立的定时器。RTC 模块拥有一组连续计数的计数器，在相应软件配置下，可提供时钟日历的功能。修改计数器的值可以重新设置系统当前的时间和日期。RTC 模块和时钟配置系统处于后备区域，即在系统复位后，RTC 的设置和时间维持不变。

20.1 RTC 固件驱动寄存器结构

20.1.1 RTC_TimeTypeDef

RTC_TimeTypeDef，定义于文件”air001xx_hal_rtc.h”如下：

```
typedef struct
{
uint8_t Hours;
uint8_t Minutes;
uint8_t Seconds;
} RTC_TimeTypeDef;
```

字段说明：

表20-1 RTC_TimeTypeDef 字段说明

字段	描述
Hours	设置RTC 时间：时
Minutes	设置RTC 时间：分
Seconds	设置RTC 时间：秒

20.1.2 RTC_AlarmTypeDef

RTC_AlarmTypeDef，定义于文件”air001xx_hal_rtc.h”如下：

```
typedef struct
{
RTC_TimeTypeDef AlarmTime;
} RTC_AlarmTypeDef;
```

字段说明：

表20-2 RTC_AlarmTypeDef 字段说明

字段	描述
AlarmTime	闹钟时间结构体

20.1.3 RTC_InitTypeDef

RTC_InitTypeDef，定义于文件”air001xx_hal_rtc.h”如下：

```
typedef struct
```

```
{
uint32_t AsynchPrediv;
uint32_t OutPut;
} RTC_InitTypeDef;
```

字段说明：

表20-3 RTC_InitTypeDef 字段说明

字段	描述
AsynchPrediv	RTC 预分频值，0x00-0xFFFF（如果使用 RTC_AUTO_1_SECOND 则自动设置为 1s 的时间基准）
OutPut	配置RTC 输出信号源

参数说明：

OutPut 可选参数：

表20-4 OutPut 可选参数

参数	描述
RTC_OUTPUTSOURCE_NONE	不选择输出源
RTC_OUTPUTSOURCE_CALIBCLOCK	引脚输出 RTC 时钟 64 分频
RTC_OUTPUTSOURCE_ALARM	引脚输出 Alarm 脉冲信号
RTC_OUTPUTSOURCE_SECOND	引脚输出秒脉冲信号

20.1.4 RTC_DateTypeDef

RTC_DateTypeDef，定义于文件”air001xx_hal_rtc.h”如下：

```
typedef struct
{
uint8_t WeekDay;
uint8_t Month;
uint8_t Date;
uint8_t Year;
} RTC_DateTypeDef;
```

字段说明：

表20-5 RTC_DateTypeDef 字段说明

字段	描述
WeekDay	周
Month	月
Date	日

Year	年

20.1.5 RTC_HandleTypeDef

RTC_HandleTypeDef，定义于文件"air001xx_hal_rtc.h"如下：

```
typedef struct
{
RTC_TypeDef *Instance;
RTC_InitTypeDef Init;
RTC_DateTypeDef DateToUpdate;
HAL_LockTypeDef Lock;
__IO HAL_RTCStateTypeDef State;
} RTC_HandleTypeDef;
```

字段说明：

表20-6 RTC_HandleTypeDef 字段说明

字段	描述
Instance	外设基地址
Init	初始化参数结构体
DateToUpdate	当前日期
Lock	HAL 锁
State	HAL 状态

20.2 RTC 固件库函数

表20-7 RTC 固件库函数说明

函数名	描述
HAL_RTC_Init	RTC 初始化
HAL_RTC_DeInit	将 RTC 配置设为缺省值
HAL_RTC_MspInit	初始 RTC 相关 MSP
HAL_RTC_MspDeInit	将 RTC 相关 MSP 配置设为缺省值
HAL_RTC_SetTime	设置时间
HAL_RTC_GetTime	获取当前时间
HAL_RTC_SetDate	设置日期
HAL_RTC_GetDate	获取当前日期
HAL_RTC_SetAlarm	设置闹钟时间
HAL_RTC_SetAlarm_IT	设置闹钟并开启闹钟中断

HAL_RTC_DeactivateAlarm	关闭闹钟及其中断
HAL_RTC_AlarmIRQHandler	闹钟中断请求处理
HAL_RTC_PollForAlarmAEvent	使用轮询的方式查询闹钟是否触发
HAL_RTC_AlarmAEventCallback	闹钟事件回调函数
HAL_RTC_GetState	获取RTC 状态
HAL_RTC_WaitForSynchro	等待RTC 寄存器与APB 时钟同步

20.2.1 函数 HAL_RTC_Init

描述了函数 HAL_RTC_Init

表20-8 函数 HAL_RTC_Init

函数名	HAL_RTC_Init
函数原形	HAL_StatusTypeDef HAL_RTC_Init(RTC_HandleTypeDef * hrtc)
功能描述	初始化 RTC
输入参数	hrtc：RTC 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

20.2.2 函数 HAL_RTC_DeInit

描述了函数 HAL_RTC_DeInit

表20-9 函数 HAL_RTC_DeInit

函数名	HAL_RTC_DeInit
函数原形	HAL_StatusTypeDef HAL_RTC_DeInit(RTC_HandleTypeDef *hrtc)
功能描述	将 RTC 配置设为缺省值
输入参数	hrtc：RTC 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

20.2.3 函数 HAL_RTC_MspInit

描述了函数 HAL_RTC_MspInit

表20-10 函数 HAL_RTC_MspInit

函数名	HAL_RTC_MspInit
函数原形	void HAL_RTC_MspInit(RTC_HandleTypeDef * hrtc)
功能描述	初始化 RTC 相关的 MSP
输入参数	hrtc：RTC 句柄
输出参数	无

返回值	无
先决条件	无

20.2.4 函数 HAL_RTC_MspDeInit

描述了函数 HAL_RTC_MspDeInit

表20-11 HAL_RTC_MspDeInit

函数名	HAL_RTC_MspDeInit
函数原形	void HAL_RTC_MspDeInit(RTC_HandleTypeDef *hrtc)
功能描述	将 RTC 相关的 MSP 设为缺省值
输入参数	hrtc：RTC 句柄
输出参数	无
返回值	无
先决条件	无

20.2.5 函数 HAL_RTC_SetTime

描述了函数 HAL_RTC_SetTime

表20-12 函数 HAL_RTC_SetTime

函数名	HAL_RTC_SetTime
函数原形	HAL_StatusTypeDef HAL_RTC_SetTime(RTC_HandleTypeDef *hrtc, RTC_TimeTypeDef *sTime, uint32_t Format)
功能描述	设置 RTC 当前时间
输入参数 1	hrtc：RTC 句柄
输入参数 2	sTime：时间配置结构体
输入参数 3	Format：选择输入参数的格式
输出参数	无
返回值	HAL 状态
先决条件	无

Format 可选参数：

表20-13 Format 可选参数

参数	描述
RTC_FORMAT_BIN	二进制格式
RTC_FORMAT_BCD	BCD 码格式

20.2.6 函数 HAL_RTC_GetTime

描述了函数 HAL_RTC_GetTime

表20-14 函数 HAL_RTC_GetTime

函数名	HAL_RTC_GetTime
函数原形	HAL_StatusTypeDef HAL_RTC_GetTime(RTC_HandleTypeDef *hrtc, RTC_TimeTypeDef *sTime, uint32_t Format)

功能描述	获取 RTC 当前时间，保存在 sTime 结构体中
输入参数 1	hrtc：RTC 句柄
输入参数 2	sTime：时间配置结构体
输入参数 3	Format：选择输入参数的格式
输出参数	无
返回值	HAL 状态
先决条件	无

Format 可选参数：

表20-15 Format 可选参数

参数	描述
RTC_FORMAT_BIN	二进制格式
RTC_FORMAT_BCD	BCD 码格式

20.2.7 函数 HAL_RTC_SetDate

描述了函数 HAL_RTC_SetDate

表20-16 函数 HAL_RTC_SetDate

函数名	HAL_RTC_SetDate
函数原形	HAL_StatusTypeDef HAL_RTC_SetDate(RTC_HandleTypeDef *hrtc, RTC_DateTypeDef *sDate, uint32_t Format)
功能描述	设置 RTC 当前日期
输入参数 1	hrtc：RTC 句柄
输入参数 2	sDate：日期配置结构体
输入参数 3	Format：选择输入参数的格式
输出参数	无
返回值	HAL 状态
先决条件	无

Format 可选参数：

表20-17 Format 可选参数

参数	描述
RTC_FORMAT_BIN	二进制格式
RTC_FORMAT_BCD	BCD 码格式

20.2.8 函数 HAL_RTC_GetDate

描述了函数 HAL_RTC_GetDate

表20-18 函数 HAL_RTC_GetDate

函数名	HAL_RTC_GetDate
函数原形	HAL_StatusTypeDef HAL_RTC_GetDate(RTC_HandleTypeDef *hrtc, RTC_DateTypeDef *sDate, uint32_t Format)

功能描述	获取 RTC 当前日期，保存在 sDate 结构体中
输入参数 1	hrtc：RTC 句柄
输入参数 2	sDate：日期配置结构体
输入参数 3	Format：选择输入参数的格式
输出参数	无
返回值	HAL 状态
先决条件	无

Format 可选参数：

表20-19 Format 可选参数

参数	描述
RTC_FORMAT_BIN	二进制格式
RTC_FORMAT_BCD	BCD 码格式

20.2.9 函数 HAL_RTC_SetAlarm

描述了函数 HAL_RTC_SetAlarm

表20-20 函数 HAL_RTC_SetAlarm

函数名	HAL_RTC_SetAlarm
函数原形	HAL_StatusTypeDef HAL_RTC_SetAlarm(RTC_HandleTypeDef *hrtc, RTC_AlarmTypeDef *sAlarm, uint32_t Format)
功能描述	设置闹钟时间
输入参数 1	hrtc：RTC 句柄
输入参数 2	sAlarm：闹钟结构体指针
输入参数 3	Format：选择输入参数的格式
输出参数	无
返回值	HAL 状态
先决条件	无

Format 可选参数：

表20-21 Format 可选参数

参数	描述
RTC_FORMAT_BIN	二进制格式
RTC_FORMAT_BCD	BCD 码格式

20.2.10 函数 HAL_RTC_SetAlarm_IT

描述了函数 HAL_RTC_SetAlarm_IT

表20-22 函数 HAL_RTC_SetAlarm_IT

函数名	HAL_RTC_SetAlarm_IT
函数原形	HAL_StatusTypeDef HAL_RTC_SetAlarm_IT(RTC_HandleTypeDef *hrtc, RTC_AlarmTypeDef *sAlarm, uint32_t Format)

功能描述	设置闹钟时间并开启中断
输入参数 1	hrtc：RTC 句柄
输入参数 2	sAlarm：闹钟结构体指针
输入参数 3	Format：选择输入参数的格式
输出参数	无
返回值	HAL 状态
先决条件	无

Format 可选参数：

表20-23 Format 可选参数

参数	描述
RTC_FORMAT_BIN	二进制格式
RTC_FORMAT_BCD	BCD 码格式

20.2.11 函数 HAL_RTC_DeactivateAlarm

描述了函数 HAL_RTC_DeactivateAlarm

表20-24 函数 HAL_RTC_DeactivateAlarm

函数名	HAL_RTC_DeactivateAlarm
函数原形	HAL_StatusTypeDef HAL_RTC_DeactivateAlarm(RTC_HandleTypeDef *hrtc)
功能描述	关闭闹钟及其中断
输入参数	hrtc：RTC 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

20.2.12 函数 HAL_RTC_AlarmIRQHandler

描述了函数 HAL_RTC_AlarmIRQHandler

表20-25 函数 HAL_RTC_AlarmIRQHandler

函数名	HAL_RTC_AlarmIRQHandler
函数原形	void HAL_RTC_AlarmIRQHandler(RTC_HandleTypeDef *hrtc)
功能描述	闹钟中断请求处理
输入参数	hrtc：RTC 句柄
输出参数	无
返回值	无
先决条件	无

20.2.13 函数 HAL_RTC_PollForAlarmAEvent

描述了函数 HAL_RTC_PollForAlarmAEvent

表20-26　函数 HAL_RTC_PollForAlarmAEvent

函数名	HAL_RTC_PollForAlarmAEvent
函数原形	HAL_StatusTypeDef HAL_RTC_PollForAlarmAEvent(RTC_HandleTypeDef *hrtc, uint32_t Timeout)
功能描述	使用轮询的方式查询闹钟事件
输入参数 1	hrtc：RTC 句柄
输入参数 2	Timeout：超时时间
输出参数	无
返回值	HAL 状态
先决条件	无

20.2.14 函数 HAL_RTC_AlarmAEventCallback

描述了函数 HAL_RTC_AlarmAEventCallback

表20-27　函数 HAL_RTC_AlarmAEventCallback

函数名	HAL_RTC_AlarmAEventCallback
函数原形	void HAL_RTC_AlarmAEventCallback(RTC_HandleTypeDef *hrtc)
功能描述	闹钟事件回调函数
输入参数	hrtc：RTC 句柄
输出参数	无
返回值	无
先决条件	无

20.2.15 函数 HAL_RTC_GetState

描述了函数 HAL_RTC_GetState

表20-28　函数 HAL_RTC_GetState

函数名	HAL_RTC_GetState
函数原形	HAL_RTCStateTypeDef HAL_RTC_GetState(RTC_HandleTypeDef *hrtc)
功能描述	获取 RTC 状态
输入参数	hrtc：RTC 句柄
输出参数	无
返回值	RTC 状态
先决条件	无

20.2.16 函数 HAL_RTC_WaitForSynchro

描述了函数 HAL_RTC_WaitForSynchro

表20-29　函数 HAL_RTC_WaitForSynchro

函数名	HAL_RTC_WaitForSynchro
函数原形	HAL_StatusTypeDef HAL_RTC_WaitForSynchro(RTC_HandleTypeDef *hrtc)
功能描述	等待 RTC 寄存器与 APB 时钟同步

输入参数	hrtc：RTC 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

21 HAL 实时时钟扩展驱动程序（RTC_Ex）

21.1 RTC_Ex 固件库函数

表21-1 RTC_Ex 固件库函数说明

函数名	描述
HAL_RTCEx_SetSecond_IT	开启秒中断
HAL_RTCEx_DeactivateSecond	关闭秒中断
HAL_RTCEx_RTCIRQHandler	RTC 中断请求处理
HAL_RTCEx_RTCEventCallback	RTC 事件回调函数
HAL_RTCEx_RTCEventErrorCallback	RTC 错误事件回调函数
HAL_RTCEx_SetSmoothCalib	设置平滑校准参数

21.1.1 函数 HAL_RTCEx_SetSecond_IT

描述了函数 HAL_RTCEx_SetSecond_IT

表21-2 函数 HAL_RTCEx_SetSecond_IT

函数名	HAL_RTCEx_SetSecond_IT
函数原形	HAL_StatusTypeDef HAL_RTCEx_SetSecond_IT(RTC_HandleTypeDef *hrtc)
功能描述	开启秒中断
输入参数	hrtc：RTC 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

21.1.2 函数 HAL_RTCEx_DeactivateSecond

描述了函数 HAL_RTCEx_DeactivateSecond

表21-3 函数 HAL_RTCEx_DeactivateSecond

函数名	HAL_RTCEx_DeactivateSecond
函数原形	HAL_StatusTypeDef HAL_RTCEx_DeactivateSecond(RTC_HandleTypeDef *hrtc)
功能描述	关闭秒中断
输入参数	hrtc：RTC 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

21.1.3 函数 HAL_RTCEx_RTCIRQHandler

描述了函数 HAL_RTCEx_RTCIRQHandler

表21-4 函数 HAL_RTCEx_RTCIRQHandler

函数名	HAL_RTCEx_RTCIRQHandler
函数原形	void HAL_RTCEx_RTCIRQHandler(RTC_HandleTypeDef *hrtc)
功能描述	RTC 中断请求处理
输入参数	hrtc：RTC 句柄
输出参数	无
返回值	无
先决条件	无

21.1.4 函数 HAL_RTCEx_RTCEventCallback

描述了函数 HAL_RTCEx_RTCEventCallback

表21-5 函数 HAL_RTCEx_RTCEventCallback

函数名	HAL_RTCEx_RTCEventCallback
函数原形	void HAL_RTCEx_RTCEventCallback(RTC_HandleTypeDef *hrtc)
功能描述	RTC 事件回调函数
输入参数	hrtc：RTC 句柄
输出参数	无
返回值	无
先决条件	无

21.1.5 函数 HAL_RTCEx_RTCEventErrorCallback

描述了函数 HAL_RTCEx_RTCEventErrorCallback

表21-6 函数 HAL_RTCEx_RTCEventErrorCallback

函数名	HAL_RTCEx_RTCEventErrorCallback
函数原形	void HAL_RTCEx_RTCEventErrorCallback(RTC_HandleTypeDef *hrtc)
功能描述	RTC 错误事件回调函数
输入参数	hrtc：RTC 句柄
输出参数	无
返回值	无
先决条件	无

21.1.6 函数 HAL_RTCEx_SetSmoothCalib

描述了函数 HAL_RTCEx_SetSmoothCalib

表21-7 函数 HAL_RTCEx_SetSmoothCalib

函数名	HAL_RTCEx_SetSmoothCalib
函数原形	HAL_StatusTypeDef HAL_RTCEx_SetSmoothCalib(RTC_HandleTypeDef *hrtc, uint32_t SmoothCalibPeriod, uint32_t SmoothCalibPlusPulses, uint32_t SmouthCalibMinusPulsesValue)
功能描述	设置平滑校准参数

输入参数 1	hrtc：RTC 句柄
输入参数 2	SmoothCalibPeriod：无作用（仅为与其他系列兼容）
输入参数 3	SmoothCalibPlusPulses：无作用（仅为与其他系列兼容）
输入参数 4	SmouthCalibMinusPulsesValue：校准值（0x00~0x7F）
输出参数	无
返回值	HAL 状态
先决条件	无

22 HAL 串行外设接口通用驱动程序（SPI）

串行外设接口(SPI)允许芯片与外部设备以半双工、全双工、单工同步的串行方式通信。此接口可以被配置成主模式/从模式，作为主模式时，能够为外部从设备提供通信时钟(SCK)。接口还能以多主配置方式工作。

22.1 SPI 固件驱动寄存器结构

22.1.1 SPI_InitTypeDef

SPI_InitTypeDef，定义于文件"air001xx_hal_spi.h"如下：

```
typedef struct
{
uint32_t Mode;
uint32_t Direction;
uint32_t DataSize;
uint32_t CLKPolarity;
uint32_t CLKPhase;
uint32_t NSS;
uint32_t BaudRatePrescaler;
uint32_t FirstBit;
uint32_t SlaveFastMode;
} SPI_InitTypeDef;
```

字段说明：

表22-1 SPI_InitTypeDef 字段说明

字段	描述
Mode	通信模式
Direction	传输方向
DataSize	数据长度
CLKPolarity	时钟极性
CLKPhase	时钟相位
NSS	NSS 信号模式
BaudRatePrescaler	波特率预分频值
FirstBit	MSB 在前或 LSB 在前
SlaveFastMode	从机模式下管理快速模式

参数说明：

Mode 可选参数：

表22-2 Mode 可选参数

参数	描述
SPI_MODE_SLAVE	从机模式
SPI_MODE_MASTER	主机模式

Direction 可选参数：

表22-3 Direction 可选参数

参数	描述
SPI_DIRECTION_2LINES	全双工模式
SPI_DIRECTION_2LINES_RXONLY	单工模式
SPI_DIRECTION_1LINE	半双工模式

DataSize 可选参数：

表22-4 DataSize 可选参数

参数	描述
SPI_DATASIZE_16BIT	16 位数据长度
SPI_DATASIZE_8BIT	8 位数据长度

CLKPolarity 可选参数：

表22-5 CLKPolarity 可选参数

参数	描述
SPI_POLARITY_LOW	空闲状态时，SCK 保持低电平
SPI_POLARITY_HIGH	空闲状态时，SCK 保持高电平

CLKPhase 可选参数：

表22-6 CLKPhase 可选参数

参数	描述
SPI_PHASE_1EDGE	第一个时钟边沿采样
SPI_PHASE_2EDGE	第二个时钟边沿采样

NSS 可选参数：

表22-7 NSS 可选参数

参数	描述
SPI_NSS_SOFT	软件 NSS
SPI_NSS_HARD_INPUT	硬件 NSS 输入模式
SPI_NSS_HARD_OUTPUT	硬件 NSS 输出模式

BaudRatePrescaler 可选参数：

表22-8 BaudRatePrescaler 可选参数

参数	描述
SPI_BAUDRATEPRESCALER_2	波特率为时钟 2 分频
SPI_BAUDRATEPRESCALER_4	波特率为时钟 4 分频
SPI_BAUDRATEPRESCALER_8	波特率为时钟 8 分频
SPI_BAUDRATEPRESCALER_16	波特率为时钟 16 分频
SPI_BAUDRATEPRESCALER_32	波特率为时钟 32 分频
SPI_BAUDRATEPRESCALER_64	波特率为时钟 64 分频
SPI_BAUDRATEPRESCALER_128	波特率为时钟 128 分频
SPI_BAUDRATEPRESCALER_256	波特率为时钟 256 分频

FirstBit 可选参数：

表22-9 FirstBit 可选参数

参数	描述
SPI_FIRSTBIT_MSB	从 MSB 开始传输
SPI_FIRSTBIT_LSB	从 LSB 开始传输

SlaveFastMode 可选参数：

表22-10 SlaveFastMode 可选参数

参数	描述
SPI_SLAVE_FAST_MODE_DISABLE	关闭从机快速模式
SPI_SLAVE_FAST_MODE_ENABLE	开启从机快速模式

22.1.2 SPI_HandleTypeDef

SPI_HandleTypeDef，定义于文件”air001xx_hal_spi.h”如下：

```
typedef struct __SPI_HandleTypeDef
{
SPI_TypeDef *Instance;
SPI_InitTypeDef Init;
uint8_t *pTxBuffPtr;
uint16_t TxXferSize;
__IO uint16_t TxXferCount;
uint8_t *pRxBuffPtr;
uint16_t RxXferSize;
__IO uint16_t RxXferCount;
void (*RxISR)(struct __SPI_HandleTypeDef *hspi);
void (*TxISR)(struct __SPI_HandleTypeDef *hspi);
DMA_HandleTypeDef *hdmatx;
```

```
DMA_HandleTypeDef *hdmarx;
HAL_LockTypeDef Lock;
__IO HAL_SPI_StateTypeDef State;
__IO uint32_t ErrorCode;
} SPI_HandleTypeDef;
```

字段说明：

表22-11 SPI_HandleTypeDef 字段说明

字段	描述
Instance	SPI 基地址
Init	初始化结构体
pTxBuffPtr	发送数据缓冲区指针
TxXferSize	发送数据总量
TxXferCount	未发送的数据数量
pRxBuffPtr	接收数据缓冲区指针
RxXferSize	接收数据总量
RxXferCount	未接收的数据数量
(*RxISR)(struct __SPI_HandleTypeDef *hspi)	接收中断服务处理函数指针
(*TxISR)(struct __SPI_HandleTypeDef *hspi)	发送中断服务处理函数指针
hdmatx	发送DMA 句柄指针
hdmarx	接收DMA 句柄指针
Lock	HAL 锁
State	SPI 通信状态
ErrorCode	错误代码

22.2 SPI 固件库函数

表22-12 SPI 固件库函数说明

函数名	描述
HAL_SPI_Init	初始化 SPI
HAL_SPI_DeInit	将 SPI 配置设为缺省值
HAL_SPI_MspInit	初始化 SPI 相关的 MSP
HAL_SPI_MspDeInit	将 SPI 相关的 MSP 设为缺省值
HAL_SPI_Transmit	使用轮询的方式发送数据

HAL_SPI_Receive	使用轮询的方式接收数据
HAL_SPI_TransmitReceive	使用轮询的方式同时接收和发送数据
HAL_SPI_Transmit_IT	使用中断的方式发送数据
HAL_SPI_Receive_IT	使用中断的方式接收数据
HAL_SPI_TransmitReceive_IT	使用中断的方式同时发送和接收数据
HAL_SPI_Transmit_DMA	使用DMA 的方式发送数据
HAL_SPI_Receive_DMA	使用DMA 的方式接收数据
HAL_SPI_TransmitReceive_DMA	使用DMA 的方式同时发送和接收数据
HAL_SPI_DMAPause	暂停DMA 传输
HAL_SPI_DMAResume	恢复DMA 传输
HAL_SPI_DMAStop	停止DMA 传输
HAL_SPI_Abort	中止SPI 传输
HAL_SPI_Abort_IT	中止SPI 传输并关闭中断
HAL_SPI_IRQHandler	SPI 中断请求处理
HAL_SPI_TxCpltCallback	发送完成回调函数
HAL_SPI_RxCpltCallback	接收完成回调函数
HAL_SPI_TxRxCpltCallback	同时发送和接收完成回调函数
HAL_SPI_TxHalfCpltCallback	发送半完成回调函数
HAL_SPI_RxHalfCpltCallback	接收半完成回调函数
HAL_SPI_TxRxHalfCpltCallback	同时接收和发送半完成回调函数
HAL_SPI_ErrorCallback	错误回调函数
HAL_SPI_AbortCpltCallback	中止完成回调函数
HAL_SPI_GetState	获取SPI 通信状态
HAL_SPI_GetError	获取错误代码

22.2.1 函数 HAL_SPI_Init

描述了函数 HAL_SPI_Init

表22-13　函数 HAL_SPI_Init

函数名	HAL_SPI_Init
函数原形	HAL_StatusTypeDef HAL_SPI_Init(SPI_HandleTypeDef *hspi)
功能描述	初始化 SPI
输入参数	hspi：SPI 句柄
输出参数	无

返回值	HAL 状态
先决条件	无

22.2.2 函数 HAL_SPI_DeInit

描述了函数 HAL_SPI_DeInit

表22-14 函数 HAL_SPI_DeInit

函数名	HAL_SPI_DeInit
函数原形	HAL_StatusTypeDef HAL_SPI_DeInit(SPI_HandleTypeDef *hspi)
功能描述	将 SPI 配置设为缺省值
输入参数	hspi：SPI 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

22.2.3 函数 HAL_SPI_MspInit

描述了函数 HAL_SPI_MspInit

表22-15 函数 HAL_SPI_MspInit

函数名	HAL_SPI_MspInit
函数原形	void HAL_SPI_MspInit(SPI_HandleTypeDef *hspi)
功能描述	初始化 SPI 相关的 MSP
输入参数	hspi：SPI 句柄
输出参数	无
返回值	无
先决条件	无

22.2.4 函数 HAL_SPI_MspDeInit

描述了函数 HAL_SPI_MspDeInit

表22-16 函数 HAL_SPI_MspDeInit

函数名	HAL_SPI_MspDeInit
函数原形	void HAL_SPI_MspDeInit(SPI_HandleTypeDef *hspi)
功能描述	将 SPI 相关的 MSP 设为缺省值
输入参数	hspi：SPI 句柄
输出参数	无
返回值	无
先决条件	无

22.2.5 函数 HAL_SPI_Transmit

描述了函数 HAL_SPI_Transmit

表22-17 函数 HAL_SPI_Transmit

函数名	HAL_SPI_Transmit
函数原形	HAL_StatusTypeDef HAL_SPI_Transmit(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size, uint32_t Timeout)
功能描述	使用轮询的方式发送数据
输入参数 1	hspi：SPI 句柄
输入参数 2	pData：数据缓冲区指针
输入参数 3	Size：数据长度
输入参数 4	Timeout：超时时间
输出参数	无
返回值	HAL 状态
先决条件	无

22.2.6 函数 HAL_SPI_Receive

描述了函数 HAL_SPI_Receive

表22-18 函数 HAL_SPI_Receive

函数名	HAL_SPI_Receive
函数原形	HAL_StatusTypeDef HAL_SPI_Receive(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size, uint32_t Timeout)
功能描述	使用轮询的方式接收数据
输入参数 1	hspi：SPI 句柄
输入参数 2	pData：数据缓冲区指针
输入参数 3	Size：数据长度
输入参数 4	Timeout：超时时间
输出参数	无
返回值	HAL 状态
先决条件	无

22.2.7 函数 HAL_SPI_TransmitReceive

描述了函数 HAL_SPI_TransmitReceive

表22-19 函数 HAL_SPI_TransmitReceive

函数名	HAL_SPI_TransmitReceive
函数原形	HAL_StatusTypeDef HAL_SPI_TransmitReceive (SPI_HandleTypeDef *hspi, uint8_t *pTxData, uint8_t *pRxData, uint16_t Size, uint32_t Timeout)
功能描述	使用轮询的方式同时发送和接收数据
输入参数 1	hspi：SPI 句柄
输入参数 2	pTxData：数据发送缓冲区指针
输入参数 3	pRxData：数据接收缓冲区指针
输入参数 4	Size：数据长度

输入参数 5	Timeout：超时时间
输出参数	无
返回值	HAL 状态
先决条件	无

22.2.8 函数 HAL_SPI_Transmit_IT

描述了函数 HAL_SPI_Transmit_IT

表22-20 函数 HAL_SPI_Transmit_IT

函数名	HAL_SPI_Transmit_IT
函数原形	HAL_StatusTypeDef HAL_SPI_Transmit_IT(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size)
功能描述	使用中断的方式发送数据
输入参数 1	hspi：SPI 句柄
输入参数 2	pData：数据缓冲区指针
输入参数 3	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

22.2.9 函数 HAL_SPI_Receive_IT

描述了函数 HAL_SPI_Receive_IT

表22-21 函数 HAL_SPI_Receive_IT

函数名	HAL_SPI_Receive_IT
函数原形	HAL_StatusTypeDef HAL_SPI_Receive_IT(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size)
功能描述	使用中断的方式接收数据
输入参数 1	hspi：SPI 句柄
输入参数 2	pData：数据缓冲区指针
输入参数 3	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

22.2.10 函数 HAL_SPI_TransmitReceive_IT

描述了函数 HAL_SPI_TransmitReceive_IT

表22-22 函数 HAL_SPI_TransmitReceive_IT

函数名	HAL_SPI_TransmitReceive_IT
函数原形	HAL_StatusTypeDef HAL_SPI_TransmitReceive_IT (SPI_HandleTypeDef *hspi, uint8_t *pTxData, uint8_t *pRxData,uint16_t Size)
功能描述	使用中断的方式同时发送和接收数据

输入参数 1	hspi：SPI 句柄
输入参数 2	pTxData：数据发送缓冲区指针
输入参数 3	pRxData：数据接收缓冲区指针
输入参数 4	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

22.2.11 函数 HAL_SPI_Transmit_DMA

描述了函数 HAL_SPI_Transmit_DMA

表22-23 函数 HAL_SPI_Transmit_DMA

函数名	HAL_SPI_Transmit_DMA
函数原形	HAL_StatusTypeDef HAL_SPI_Transmit_DMA(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size)
功能描述	使用 DMA 的方式发送数据
输入参数 1	hspi：SPI 句柄
输入参数 2	pData：数据缓冲区指针
输入参数 3	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

22.2.12 函数 HAL_SPI_Receive_DMA

描述了函数 HAL_SPI_Receive_DMA

表22-24 函数 HAL_SPI_Receive_DMA

函数名	HAL_SPI_Receive_DMA
函数原形	HAL_StatusTypeDef HAL_SPI_Receive_DMA(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size)
功能描述	使用 DMA 的方式接收数据
输入参数 1	hspi：SPI 句柄
输入参数 2	pData：数据缓冲区指针
输入参数 3	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

22.2.13 函数 HAL_SPI_TransmitReceive_DMA

描述了函数 HAL_SPI_TransmitReceive_DMA

表22-25 函数 HAL_SPI_TransmitReceive_DMA

函数名	HAL_SPI_TransmitReceive_DMA
函数原形	HAL_StatusTypeDef HAL_SPI_TransmitReceive_DMA(SPI_HandleTypeDef *hspi, uint8_t *pTxData, uint8_t *pRxData,
功能描述	使用 DMA 的方式同时发送和接收数据
输入参数 1	hspi：SPI 句柄
输入参数 2	pTxData：数据发送缓冲区指针
输入参数 3	pRxData：数据接收缓冲区指针
输入参数 4	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

22.2.14 函数 HAL_SPI_DMAPause

描述了函数 HAL_SPI_DMAPause

表22-26 函数 HAL_SPI_DMAPause

函数名	HAL_SPI_DMAPause
函数原形	HAL_StatusTypeDef HAL_SPI_DMAPause(SPI_HandleTypeDef *hspi)
功能描述	暂停正在进行的 DMA 传输
输入参数	hspi：SPI 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

22.2.15 函数 HAL_SPI_DMAResume

描述了函数 HAL_SPI_DMAResume

表22-27 函数 HAL_SPI_DMAResume

函数名	HAL_SPI_DMAResume
函数原形	HAL_StatusTypeDef HAL_SPI_DMAResume(SPI_HandleTypeDef *hspi)
功能描述	恢复暂停的 DMA 传输
输入参数	hspi：SPI 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

22.2.16 函数 HAL_SPI_DMAStop

描述了函数 HAL_SPI_DMAStop

表22-28 函数 HAL_SPI_DMAStop

函数名	HAL_SPI_DMAStop

函数原形	HAL_StatusTypeDef HAL_SPI_DMAStop(SPI_HandleTypeDef *hspi)
功能描述	停止 DMA 传输
输入参数	hspi：SPI 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

22.2.17 函数 HAL_SPI_Abort

描述了函数 HAL_SPI_Abort

表22-29 函数 HAL_SPI_Abort

函数名	HAL_SPI_Abort
函数原形	HAL_StatusTypeDef HAL_SPI_Abort(SPI_HandleTypeDef *hspi)
功能描述	中止 SPI 传输
输入参数	hspi：SPI 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

22.2.18 函数 HAL_SPI_Abort_IT

描述了函数 HAL_SPI_Abort_IT

表22-30 函数 HAL_SPI_Abort_IT

函数名	HAL_SPI_Abort_IT
函数原形	HAL_StatusTypeDef HAL_SPI_Abort_IT(SPI_HandleTypeDef *hspi)
功能描述	中止 SPI 传输并关闭中断
输入参数	hspi：SPI 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

22.2.19 函数 HAL_SPI_IRQHandler

描述了函数 HAL_SPI_IRQHandler

表22-31 函数 HAL_SPI_IRQHandler

函数名	HAL_SPI_IRQHandler
函数原形	void HAL_SPI_IRQHandler(SPI_HandleTypeDef *hspi)
功能描述	SPI 中断请求处理
输入参数	hspi：SPI 句柄
输出参数	无
返回值	无

先决条件	无

22.2.20 函数 HAL_SPI_TxCpltCallback

描述了函数 HAL_SPI_TxCpltCallback

表22-32 函数 HAL_SPI_TxCpltCallback

函数名	HAL_SPI_TxCpltCallback
函数原形	void HAL_SPI_TxCpltCallback(SPI_HandleTypeDef *hspi)
功能描述	发送完成回调函数
输入参数	hspi：SPI 句柄
输出参数	无
返回值	无
先决条件	无

22.2.21 函数 HAL_SPI_RxCpltCallback

描述了函数 HAL_SPI_RxCpltCallback

表22-33 函数 HAL_SPI_RxCpltCallback

函数名	HAL_SPI_RxCpltCallback
函数原形	void HAL_SPI_RxCpltCallback(SPI_HandleTypeDef *hspi)
功能描述	接收完成回调函数
输入参数	hspi：SPI 句柄
输出参数	无
返回值	无
先决条件	无

22.2.22 函数 HAL_SPI_TxRxCpltCallback

描述了函数 HAL_SPI_TxRxCpltCallback

表22-34 函数 HAL_SPI_TxRxCpltCallback

函数名	HAL_SPI_TxRxCpltCallback
函数原形	void HAL_SPI_TxRxCpltCallback(SPI_HandleTypeDef *hspi)
功能描述	同时发送和接收完成回调函数
输入参数	hspi：SPI 句柄
输出参数	无
返回值	无
先决条件	无

22.2.23 函数 HAL_SPI_TxHalfCpltCallback

描述了函数 HAL_SPI_TxHalfCpltCallback

表22-35 函数 HAL_SPI_TxHalfCpltCallback

函数名	HAL_SPI_TxHalfCpltCallback

函数原形	void HAL_SPI_TxHalfCpltCallback(SPI_HandleTypeDef *hspi)
功能描述	发送半完成回调函数
输入参数	hspi：SPI 句柄
输出参数	无
返回值	无
先决条件	无

22.2.24 函数 HAL_SPI_RxHalfCpltCallback

描述了函数 HAL_SPI_RxHalfCpltCallback

表22-36 函数 HAL_SPI_RxHalfCpltCallback

函数名	HAL_SPI_RxHalfCpltCallback
函数原形	void HAL_SPI_RxHalfCpltCallback(SPI_HandleTypeDef *hspi)
功能描述	接收半完成回调函数
输入参数	hspi：SPI 句柄
输出参数	无
返回值	无
先决条件	无

22.2.25 函数 HAL_SPI_TxRxHalfCpltCallback

描述了函数 HAL_SPI_TxRxHalfCpltCallback

表22-37 函数 HAL_SPI_TxRxHalfCpltCallback

函数名	HAL_SPI_TxRxHalfCpltCallback
函数原形	void HAL_SPI_TxRxHalfCpltCallback(SPI_HandleTypeDef *hspi)
功能描述	同时发送和接收半完成回调函数
输入参数	hspi：SPI 句柄
输出参数	无
返回值	无
先决条件	无

22.2.26 函数 HAL_SPI_ErrorCallback

描述了函数 HAL_SPI_ErrorCallback

表22-38 函数 HAL_SPI_ErrorCallback

函数名	HAL_SPI_ErrorCallback
函数原形	void HAL_SPI_ErrorCallback(SPI_HandleTypeDef *hspi)
功能描述	错误回调函数
输入参数	hspi：SPI 句柄
输出参数	无
返回值	无

先决条件	无

22.2.27 函数 HAL_SPI_AbortCpltCallback

描述了函数 HAL_SPI_AbortCpltCallback

表22-39 函数 HAL_SPI_AbortCpltCallback

函数名	HAL_SPI_AbortCpltCallback
函数原形	void HAL_SPI_AbortCpltCallback(SPI_HandleTypeDef *hspi)
功能描述	中止完成回调函数
输入参数	hspi：SPI 句柄
输出参数	无
返回值	无
先决条件	无

22.2.28 函数 HAL_SPI_GetState

描述了函数 HAL_SPI_GetState

表22-40 函数 HAL_SPI_GetState

函数名	HAL_SPI_GetState
函数原形	HAL_SPI_StateTypeDef HAL_SPI_GetState(SPI_HandleTypeDef *hspi)
功能描述	获取 SPI 通信状态
输入参数	hspi：SPI 句柄
输出参数	无
返回值	SPI 状态
先决条件	无

22.2.29 函数 HAL_SPI_GetError

描述了函数 HAL_SPI_GetError

表22-41 函数 HAL_SPI_GetError

函数名	HAL_SPI_GetError
函数原形	uint32_t HAL_SPI_GetError(SPI_HandleTypeDef *hspi)
功能描述	获取错误代码
输入参数	hspi：SPI 句柄
输出参数	无
返回值	错误代码
先决条件	无

23 HAL 定时器通用驱动程序（TIM）

TIM 包含一个 16 位自动重载计数器，该计数器由可编程预分频器驱动。它可以被用作各种场景，包括：输入信号（输入捕获）的脉冲长度测量，或者产生输出波形（输出比较、输出 PWM、带死区插入的互补 PWM）。

使用定时器预分频器和 RCC 时钟控制器预分频器，可将脉冲宽度和波形周期从几微秒调制到几毫秒。

高级控制定时器(TIM1)和通用定时器(TIMy)彼此完全独立，不共享任何资源。

23.1 TIM 固件驱动寄存器结构

23.1.1 TIM_Base_InitTypeDef

TIM_Base_InitTypeDef，定义于文件"air001xx_hal_tim.h"如下：

```
typedef struct
{
uint32_t Prescaler;
uint32_t CounterMode;
uint32_t Period;
uint32_t ClockDivision;
uint32_t RepetitionCounter;
uint32_t AutoReloadPreload;
} TIM_Base_InitTypeDef;
```

字段说明：

表23-1 TIM_Base_InitTypeDef 字段说明

字段	描述
Prescaler	时钟预分频值（0x0000~0xFFFF）
CounterMode	计数模式
Period	自动重装载值（0x0000~0xFFFF）
ClockDivision	时钟分频因子
RepetitionCounter	重复计数值（通用定时器：0x00~0xFF 高级定时器：0x0000~0xFFFF
AutoReloadPreload	自动重装载预装载

参数说明：

CounterMode 可选参数：

表23-2 CounterMode 可选参数

参数	描述
TIM_COUNTERMODE_UP	向上计数模式
TIM_COUNTERMODE_DOWN	向下计数模式
TIM_COUNTERMODE_CENTERALIGNED1	中央对齐模式 1
TIM_COUNTERMODE_CENTERALIGNED2	中央对齐模式 2
TIM_COUNTERMODE_CENTERALIGNED3	中央对齐模式 3

ClockDivision 可选参数：

表23-3 ClockDivision 可选参数

参数	描述
TIM_CLOCKDIVISION_DIV1	tDTS=tCK_INT
TIM_CLOCKDIVISION_DIV2	tDTS=2*tCK_INT
TIM_CLOCKDIVISION_DIV4	tDTS=4*tCK_INT

AutoReloadPreload 可选参数：

表23-4 AutoReloadPreload 可选参数

参数	描述
TIM_AUTORELOAD_PRELOAD_DISABLE	关闭自动重装载预装载
TIM_AUTORELOAD_PRELOAD_ENABLE	开启自动重装载预装载

23.1.2 TIM_OC_InitTypeDef

TIM_OC_InitTypeDef，定义于文件"air001xx_hal_tim.h"如下：

```
typedef struct
{
uint32_t OCMode;
uint32_t Pulse;
uint32_t OCPolarity;
uint32_t OCNPolarity;
uint32_t OCFastMode;
uint32_t OCIdleState;
uint32_t OCNIdleState;
} TIM_OC_InitTypeDef;
```

字段说明：

表23-5 TIM_OC_InitTypeDef 字段说明

字段	描述
OCMode	输出比较模式
Pulse	比较值（0x0000~0xFFFF）

OCPolarity	输出引脚极性
OCNPolarity	互补输出引脚极性
OCFastMode	输出比较快速使能（仅在 PWM1 和 PWM2 模式下可用）
OCIdleState	空闲期间输出引脚状态
OCNIdleState	空闲期间互补输出引脚状态

参数说明：

OCMode 可选参数：

表23-6 OCMode 可选参数

参数	描述
TIM_OCMODE_TIMING	冻结，计数器与比较值的比较对 OCREF 不起作用
TIM_OCMODE_ACTIVE	匹配时设置通道电平为有效电平
TIM_OCMODE_INACTIVE	匹配时设置通道电平为无效电平
TIM_OCMODE_TOGGLE	匹配时翻转通道电平
TIM_OCMODE_PWM1	PWM1 模式
TIM_OCMODE_PWM2	PWM2 模式
TIM_OCMODE_FORCED_ACTIVE	强制输出有效电平
TIM_OCMODE_FORCED_INACTIVE	强制输出无效电平

OCPolarity 可选参数：

表23-7 OCPolarity 可选参数

参数	描述
TIM_OCPOLARITY_HIGH	高电平作为 OC 有效电平
TIM_OCPOLARITY_LOW	低电平作为 OC 有效电平

OCNPolarity 可选参数：

表23-8 OCNPolarity 可选参数

参数	描述
TIM_OCNPOLARITY_HIGH	高电平作为 OCN 有效电平
TIM_OCNPOLARITY_LOW	低电平作为 OCN 有效电平

OCFastMode 可选参数：

表23-9 OCFastMode 可选参数

参数	描述
TIM_OCFAST_DISABLE	关闭快速使能
TIM_OCFAST_ENABLE	开启快速使能

OCIdleState 可选参数：

表23-10 OCIdleState 可选参数

参数	描述
TIM_OCIDLESTATE_SET	空闲时 OC=1
TIM_OCIDLESTATE_RESET	空闲时 OC=0

OCNIdleState 可选参数：

表23-11 OCNIdleState 可选参数

参数	描述
TIM_OCNIDLESTATE_SET	空闲时 OCN=1
TIM_OCNIDLESTATE_RESET	空闲时 OCN=0

23.1.3 TIM_OnePulse_InitTypeDef

TIM_OnePulse_InitTypeDef，定义于文件"air001xx_hal_tim.h"如下：

```
typedef struct
{
uint32_t OCMode;
uint32_t Pulse;
uint32_t OCPolarity;
uint32_t OCNPolarity;
uint32_t OCIdleState;
uint32_t OCNIdleState;
uint32_t ICPolarity;
uint32_t ICSelection;
uint32_t ICFilter;
} TIM_OnePulse_InitTypeDef;
```

字段说明：

表23-12 TIM_OnePulse_InitTypeDef 字段说明

字段	描述
OCMode	单脉冲模式
Pulse	比较值（0x0000~0xFFFF）
OCPolarity	输出引脚极性
OCNPolarity	互补输出引脚极性
OCIdleState	空闲期间输出引脚状态
OCNIdleState	空闲期间互补输出引脚状态
ICPolarity	输入捕获信号的极性配置

ICSelection	选择输入通道与捕获通道的映射关系
ICFilter	输入捕获滤波器值（0x0~0xF）

参数说明：

OCMode 可选参数：

表23-13 OCMode 可选参数

参数	描述
TIM_OCMODE_TIMING	冻结，计数器与比较值的比较对 OCREF 不起作用
TIM_OCMODE_ACTIVE	匹配时设置通道电平为有效电平
TIM_OCMODE_INACTIVE	匹配时设置通道电平为无效电平
TIM_OCMODE_TOGGLE	匹配时翻转通道电平
TIM_OCMODE_PWM1	PWM1 模式
TIM_OCMODE_PWM2	PWM2 模式
TIM_OCMODE_FORCED_ACTIVE	强制输出有效电平
TIM_OCMODE_FORCED_INACTIVE	强制输出无效电平

OCPolarity 可选参数：

表23-14 OCPolarity 可选参数

参数	描述
TIM_OCPOLARITY_HIGH	高电平作为 OC 有效电平
TIM_OCPOLARITY_LOW	低电平作为 OC 有效电平

OCNPolarity 可选参数：

表23-15 OCNPolarity 可选参数

参数	描述
TIM_OCNPOLARITY_HIGH	高电平作为 OCN 有效电平
TIM_OCNPOLARITY_LOW	低电平作为 OCN 有效电平

OCIdleState 可选参数：

表23-16 OCIdleState 可选参数

参数	描述
TIM_OCIDLESTATE_SET	空闲时 OC=1
TIM_OCIDLESTATE_RESET	空闲时 OC=0

OCNIdleState 可选参数：

表23-17 OCNIdleState 可选参数

参数	描述

TIM_OCNIDLESTATE_SET	空闲时 OCN=1
TIM_OCNIDLESTATE_RESET	空闲时 OCN=0

ICPolarity 可选参数：

表23-18 ICPolarity 可选参数

参数	描述
TIM_ICPOLARITY_RISING	不反相，上升沿触发捕获
TIM_ICPOLARITY_FALLING	反相，下降沿触发捕获
TIM_ICPOLARITY_BOTHEDGE	上升沿和下降沿都能触发捕获

ICSelection 可选参数：

表23-19 ICSelection 可选参数

参数	描述
TIM_ICSELECTION_DIRECTTI	CCx 通道配置为输入，ICx 映射在 TIx 上
TIM_ICSELECTION_INDIRECTTI	CCx 通道配置为输入，ICx 映射在 TIy 上
TIM_ICSELECTION_TRC	CCx 通道配置为输入，ICx 映射在 TRC 上 此模式仅在内部触发器输入被选中时可用

注意：当 x=1 时，y=2；当 x=2 时，y=1；当 x=3 时，y=4；当 x=4 时 y=3。

23.1.4 TIM_IC_InitTypeDef

TIM_IC_InitTypeDef，定义于文件"air001xx_hal_tim.h"如下：

```
typedef struct
{
uint32_tICPolarity;
uint32_t ICSelection;
uint32_t ICPrescaler;
uint32_t ICFilter;
} TIM_IC_InitTypeDef;
```

字段说明：

表23-20 TIM_IC_InitTypeDef 字段说明

字段	描述
tICPolarity	输入捕获信号的极性
ICSelection	输入通道和捕获通道的映射关系
ICPrescaler	输入捕获预分频值
ICFilter	输入捕获滤波器值（0x0~0xF）

参数说明：

ICPolarity 可选参数：

表23-21 ICPolarity 可选参数

参数	描述
TIM_ICPOLARITY_RISING	不反相，上升沿触发捕获
TIM_ICPOLARITY_FALLING	反相，下降沿触发捕获
TIM_ICPOLARITY_BOTHEDGE	上升沿和下降沿都能触发捕获

ICSelection 可选参数：

表23-22 ICSelection 可选参数

参数	描述
TIM_ICSELECTION_DIRECTTI	CCx 通道配置为输入，ICx 映射在 TIx 上
TIM_ICSELECTION_INDIRECTTI	CCx 通道配置为输入，ICx 映射在 TIy 上
TIM_ICSELECTION_TRC	CCx 通道配置为输入，ICx 映射在 TRC 上 此模式仅在内部触发器输入被选中时可用

注意：当 x=1 时，y=2；当 x=2 时，y=1；当 x=3 时，y=4；当 x=4 时 y=3。

ICPrescaler 可选参数：

表23-23 ICPrescaler 可选参数

参数	描述
TIM_ICPSC_DIV1	每个事件都触发捕获
TIM_ICPSC_DIV2	每 2 个事件触发一次捕获
TIM_ICPSC_DIV4	每 4 个事件触发一次捕获
TIM_ICPSC_DIV8	每 8 个事件触发一次捕获

23.1.5 TIM_Encoder_InitTypeDef

TIM_Encoder_InitTypeDef，定义于文件”air001xx_hal_tim.h”如下：

```
typedef struct
{
uint32_t EncoderMode;
uint32_t IC1Polarity;
uint32_t IC1Selection;
uint32_t IC1Prescaler;
uint32_t IC1Filter;
uint32_t IC2Polarity;
uint32_t IC2Selection;
uint32_t IC2Prescaler;
uint32_t IC2Filter;
} TIM_Encoder_InitTypeDef;
```

字段说明：

表23-24 TIM_Encoder_InitTypeDef 字段说明

字段	描述
EncoderMode	编码器接口模式
IC1Polarity	输入捕获信号 1 的极性
IC1Selection	输入通道 1 和捕获通道的映射关系
IC1Prescaler	输入信号 1 的预分频值
IC1Filter	输入信号 1 的滤波器值（0x0~0xF）
IC2Polarity	输入捕获信号 2 的极性
IC2Selection	输入通道 2 和捕获通道的映射关系
IC2Prescaler	输入信号 2 的预分频值
IC2Filter	输入信号 2 的滤波器值（0x0~0xF）

参数说明：

EncoderMode 可选参数：

表23-25 EncoderMode 可选参数

参数	描述
TIM_ENCODERMODE_TI1	模式 1，根据 TI1FP1 电平在 TI2FP2 边沿上向上/向下计数
TIM_ENCODERMODE_TI2	模式 2，根据 TI2FP2 电平在 TI1FP1 边沿上向上/向下计数
TIM_ENCODERMODE_TI12	模式 3，模式 1 和模式 2 的综合

IC1Polarity 可选参数：

表23-26 IC1Polarity 可选参数

参数	描述
TIM_ENCODERINPUTPOLARITY_RISING	不反相，上升沿触发捕获
TIM_ENCODERINPUTPOLARITY_FALLING	反相，下降沿触发捕获

IC1Selection 可选参数：

表23-27 IC1Selection 可选参数

参数	描述
TIM_ICSELECTION_DIRECTTI	CC1 通道配置为输入，IC1 映射在 TI1 上
TIM_ICSELECTION_INDIRECTTI	CC1 通道配置为输入，IC1 映射在 TI2 上

IC1Prescaler 可选参数：

表23-28 IC1Prescaler 可选参数

参数	描述

TIM_ICPSC_DIV1	每个事件都触发捕获
TIM_ICPSC_DIV2	每 2 个事件触发一次捕获
TIM_ICPSC_DIV4	每 4 个事件触发一次捕获
TIM_ICPSC_DIV8	每 8 个事件触发一次捕获

IC2Polarity 可选参数：

表23-29 IC2Polarity 可选参数

参数	描述
TIM_ENCODERINPUTPOLARITY_RISING	不反相，上升沿触发捕获
TIM_ENCODERINPUTPOLARITY_FALLING	反相，下降沿触发捕获

IC2Selection 可选参数：

表23-30 IC2Selection 可选参数

参数	描述
TIM_ICSELECTION_DIRECTTI	CC2 通道配置为输入，IC2 映射在 TI2 上
TIM_ICSELECTION_INDIRECTTI	CC2 通道配置为输入，IC2 映射在 TI1 上

IC2Prescaler 可选参数：

表23-31 IC2Prescaler 可选参数

参数	描述
TIM_ICPSC_DIV1	每个事件都触发捕获
TIM_ICPSC_DIV2	每 2 个事件触发一次捕获
TIM_ICPSC_DIV4	每 4 个事件触发一次捕获
TIM_ICPSC_DIV8	每 8 个事件触发一次捕获

23.1.6 TIM_ClockConfigTypeDef

TIM_ClockConfigTypeDef，定义于文件”air001xx_hal_tim.h”如下：

```
typedef struct
{
uint32_t ClockSource;
uint32_t ClockPolarity;
uint32_t ClockPrescaler;
uint32_t ClockFilter;
} TIM_ClockConfigTypeDef;
```

字段说明：

表23-32 TIM_ClockConfigTypeDef 字段说明

字段	描述

ClockSource	时钟源
ClockPolarity	时钟极性
ClockPrescaler	时钟预分频值
ClockFilter	时钟滤波器值（0x0~0xF）

参数说明：

ClockSource 可选参数：

表23-33 ClockSource 可选参数

参数	描述
TIM_CLOCKSOURCE_ETRMODE2	外部时钟模式 2（ETRF）
TIM_CLOCKSOURCE_INTERNAL	内部时钟模式（CK_INT）
TIM_CLOCKSOURCE_ITR0	外部时钟模式 1（ITR0）
TIM_CLOCKSOURCE_ITR1	外部时钟模式 1（ITR1）
TIM_CLOCKSOURCE_ITR2	外部时钟模式 1（ITR2）
TIM_CLOCKSOURCE_ITR3	外部时钟模式 1（ITR3）
TIM_CLOCKSOURCE_TI1ED	外部时钟模式 1（通过边沿检测的 TI1FP1）
TIM_CLOCKSOURCE_TI1	外部时钟模式 1（TI1FP1）
TIM_CLOCKSOURCE_TI2	外部时钟模式 1（TI2FP2）
TIM_CLOCKSOURCE_ETRMODE1	外部时钟模式 1（ETRF）

ClockPolarity 可选参数：

表23-34 ClockPolarity 可选参数

参数	描述
TIM_CLOCKPOLARITY_INVERTED	ETR 反相，低电平或下降沿有效
TIM_CLOCKPOLARITY_NONINVERTED	ETR 不反相，高电平或上升沿有效
TIM_CLOCKPOLARITY_RISING	TIx 不反相，上升沿有效
TIM_CLOCKPOLARITY_FALLING	TIx 反相，下降沿有效
TIM_CLOCKPOLARITY_BOTHEDGE	TIx 上升沿和下降沿都有效

ClockPrescaler 可选参数：

表23-35 ClockPrescaler 可选参数

参数	描述
TIM_CLOCKPRESCALER_DIV1	每个事件都触发捕获
TIM_CLOCKPRESCALER_DIV2	每 2 个事件触发一次捕获
TIM_CLOCKPRESCALER_DIV4	每 4 个事件触发一次捕获
TIM_CLOCKPRESCALER_DIV8	每 8 个事件触发一次捕获

23.1.7 TIM_ClearInputConfigTypeDef

TIM_ClearInputConfigTypeDef，定义于文件"air001xx_hal_tim.h"如下：

```
typedef struct
{
uint32_t ClearInputState;
uint32_t ClearInputSource;
uint32_t ClearInputPolarity;
uint32_t ClearInputPrescaler;
uint32_t ClearInputFilter;
} TIM_ClearInputConfigTypeDef;
```

字段说明：

表23-36 TIM_ClearInputConfigTypeDef 字段说明

字段	描述
ClearInputState	外部事件清除 OCREF
ClearInputSource	清除信号源
ClearInputPolarity	清除信号极性
ClearInputPrescaler	清除信号预分频值（OCRef 清除和 ETR 源一起使用时该参数必须为 0）
ClearInputFilter	清除信号滤波器值（0x0~0xF）

参数说明：

ClearInputState 可选参数：

表23-37 ClearInputState 可选参数

参数	描述
ENABLE	开启 OCREF 清除功能
DISABLE	关闭 OCREF 清除功能

ClearInputSource 可选参数：

表23-38 ClearInputSource 可选参数

参数	描述
TIM_CLEARINPUTSOURCE_NONE	无清除信号源
TIM_CLEARINPUTSOURCE_ETR	ETRF 作为清除信号源
TIM_CLEARINPUTSOURCE_OCREFCLR	OCREF_CLR 作为清除信号源

ClearInputPolarity 可选参数：

表23-39 ClearInputPolarity 可选参数

参数	描述
TIM_CLEARINPUTPOLARITY_INVERTED	反相，低电平有效
TIM_CLEARINPUTPOLARITY_NONINVERTED	不反相，高电平有效

23.1.8 TIM_MasterConfigTypeDef

TIM_MasterConfigTypeDef，定义于文件"air001xx_hal_tim.h"如下：

```
typedef struct
{
uint32_t MasterOutputTrigger;
uint32_t MasterSlaveMode;
} TIM_MasterConfigTypeDef;
```

字段说明：

表23-40 TIM_MasterConfigTypeDef 字段说明

字段	描述
MasterOutputTrigger	选择主模式下送到从定时器的同步信息
MasterSlaveMode	开启主/从模式

参数说明：

MasterOutputTrigger 可选参数：

表23-41 MasterOutputTrigger 可选参数

参数	描述
TIM_TRGO_RESET	复位信号产生触发输出（TRGO）
TIM_TRGO_ENABLE	开启信号产生触发输出（TRGO）
TIM_TRGO_UPDATE	更新事件产生触发输出（TRGO）
TIM_TRGO_OC1	一次捕获或比较成功产生触发输出（TRGO）
TIM_TRGO_OC1REF	OC1REF 信号产生触发输出（TRGO）
TIM_TRGO_OC2REF	OC2REF 信号产生触发输出（TRGO）
TIM_TRGO_OC3REF	OC3REF 信号产生触发输出（TRGO）
TIM_TRGO_OC4REF	OC4REF 信号产生触发输出（TRGO）

MasterSlaveMode 可选参数：

表23-42 MasterSlaveMode 可选参数

参数	描述
TIM_MASTERSLAVEMODE_ENABLE	开启主/从模式
TIM_MASTERSLAVEMODE_DISABLE	关闭主/从模式

23.1.9 TIM_SlaveConfigTypeDef

TIM_SlaveConfigTypeDef，定义于文件"air001xx_hal_tim.h"如下：

```
typedef struct
{
uint32_t SlaveMode;
uint32_t InputTrigger;
uint32_t TriggerPolarity;
uint32_t TriggerPrescaler;
uint32_t TriggerFilter;
} TIM_SlaveConfigTypeDef;
```

字段说明：

表23-43 TIM_SlaveConfigTypeDef 字段说明

字段	描述
SlaveMode	从模式选择
InputTrigger	输入触发源
TriggerPolarity	触发源极性
TriggerPrescaler	触发源预分频值
TriggerFilter	触发源滤波器值（0x0~0xF）

参数说明:

SlaveMode 可选参数：

表23-44 SlaveMode 可选参数

参数	描述
TIM_SLAVEMODE_DISABLE	关闭从模式
TIM_SLAVEMODE_RESET	复位模式
TIM_SLAVEMODE_GATED	门控模式
TIM_SLAVEMODE_TRIGGER	触发模式
TIM_SLAVEMODE_EXTERNAL1	外部时钟模式 1

InputTrigger 可选参数：

表23-45 InputTrigger 可选参数

参数	描述
TIM_TS_ITR0	内部触发 0（ITR0）
TIM_TS_ITR1	内部触发 1（ITR1）
TIM_TS_ITR2	内部触发 2（ITR2）

TIM_TS_ITR3	内部触发 3（ITR3）
TIM_TS_TI1F_ED	边沿检测后的 TI1（TI1F_ED）
TIM_TS_TI1FP1	滤波后的定时器输入 1（TI1FP1）
TIM_TS_TI2FP2	滤波后的定时器输入 2（TI2FP2）
TIM_TS_ETRF	滤波后的外部触发输入（ETRF）
TIM_TS_NONE	不选择输入触发源

TriggerPolarity 可选参数：

表23-46 TriggerPolarity 可选参数

参数	描述
TIM_TRIGGERPOLARITY_INVERTED	ETR 反相，低电平或下降沿有效
TIM_TRIGGERPOLARITY_NONINVERTED	ETR 不反相，高电平或上升沿有效
TIM_TRIGGERPOLARITY_RISING	TIx 不反相，上升沿有效
TIM_TRIGGERPOLARITY_FALLING	TIx 反相，下降沿有效
TIM_TRIGGERPOLARITY_BOTHEDGE	TIx 上升沿和下降沿都有效

TriggerPrescaler 可选参数：

表23-47 TriggerPrescaler 可选参数

参数	描述
TIM_TRIGGERPRESCALER_DIV1	每个 ETR 事件都触发捕获
TIM_TRIGGERPRESCALER_DIV2	每 2 个 ETR 事件触发一次捕获
TIM_TRIGGERPRESCALER_DIV4	每 4 个 ETR 事件触发一次捕获
TIM_TRIGGERPRESCALER_DIV8	每 8 个 ETR 事件触发一次捕获

23.1.10 TIM_BreakDeadTimeConfigTypeDef

TIM_BreakDeadTimeConfigTypeDef，定义于文件”air001xx_hal_tim.h”如下：

```
typedef struct
{
uint32_t OffStateRunMode;
uint32_t OffStateIDLEMode;
uint32_t LockLevel
uint32_t DeadTime;
uint32_t BreakState;
uint32_t BreakPolarity;
uint32_t BreakFilter;
uint32_t AutomaticOutput;
} TIM_BreakDeadTimeConfigTypeDef;
```

字段说明：

表23-48 TIM_BreakDeadTimeConfigTypeDef

字段	描述
OffStateRunMode	运行模式下“关闭状态”选择
OffStateIDLEMode	空闲模式下“关闭状态”选择
LockLevel	锁定级别配置
DeadTime	死区持续时间（0x00~0xFF）
BreakState	刹车输入功能选择
BreakPolarity	刹车输入极性选择
BreakFilter	刹车输入数字滤波器值（0x0~0xF）
AutomaticOutput	自动输出使能

参数说明：

OffStateRunMode 可选参数：

表23-49 OffStateRunMode 可选参数

参数	描述
TIM_OSSR_ENABLE	当输出关闭时，OC/OCN 输出由 TIM 控制
TIM_OSSR_DISABLE	当输出关闭时，OC/OCN 输出被禁用

OffStateIDLEMode 可选参数：

表23-50 OffStateIDLEMode 可选参数

参数	描述
TIM_OSSI_ENABLE	当输出关闭时，OC/OCN 输出由 TIM 控制
TIM_OSSI_DISABLE	当输出关闭时，OC/OCN 输出被禁用

LockLevel 可选参数：

表23-51 LockLevel 可选参数

参数	描述
TIM_LOCKLEVEL_OFF	锁定关闭
TIM_LOCKLEVEL_1	锁定级别 1
TIM_LOCKLEVEL_2	锁定级别 2
TIM_LOCKLEVEL_3	锁定级别 3

BreakState 可选参数：

表23-52 BreakState 可选参数

参数	描述
TIM_BREAK_ENABLE	开启刹车功能

TIM_BREAK_DISABLE	关闭刹车功能

BreakPolarity 可选参数：

表23-53 BreakPolarity 可选参数

参数	描述
TIM_BREAKPOLARITY_LOW	刹车输入低电平有效
TIM_BREAKPOLARITY_HIGH	刹车输入高电平有效

AutomaticOutput 可选参数：

表23-54 AutomaticOutput 可选参数

参数	描述
TIM_AUTOMATICOUTPUT_DISABLE	关闭自动输出模式
TIM_AUTOMATICOUTPUT_ENABLE	开启自动输出模式

23.1.11 TIM_HandleTypeDef

TIM_HandleTypeDef，定义于文件"air001xx_hal_tim.h"如下：

```
typedef struct
{
TIM_TypeDef *Instance;
TIM_Base_InitTypeDef Init;
HAL_TIM_ActiveChannel Channel;
DMA_HandleTypeDef *hdma[7];
HAL_LockTypeDef Lock;
__IO HAL_TIM_StateTypeDef State;
} TIM_HandleTypeDef;
```

字段说明：

表23-55 TIM_HandleTypeDef 字段说明

字段	描述
Instance	TIM 基地址
Init	初始化参数结构体
Channel	有效通道
hdma	DMA 句柄数组
Lock	HAL 锁
State	TIM 状态

23.2 TIM 固件库函数

表23-56 TIM 固件库函数说明

函数名	描述
HAL_TIM_Base_Init	初始化时基单元
HAL_TIM_Base_DeInit	将时基单元配置设为缺省值
HAL_TIM_Base_MspInit	初始化时单元式相关的 MSP
HAL_TIM_Base_MspDeInit	将时基单元相关的 MSP 设为缺省值
HAL_TIM_Base_Start	开启时基的产生
HAL_TIM_Base_Stop	关闭时基的产生
HAL_TIM_Base_Start_IT	开启时基的产生和更新中断
HAL_TIM_Base_Stop_IT	关闭时基的产生和更新中断
HAL_TIM_Base_Start_DMA	开启时基的产生和产生更新事件时的 DMA 请求
HAL_TIM_Base_Stop_DMA	关闭时基的产生和产生更新事件时的 DMA 请求
HAL_TIM_OC_Init	初始化输出比较模式
HAL_TIM_OC_DeInit	将输出比较模式配置设为缺省值
HAL_TIM_OC_MspInit	初始化输出比较模式相关的 MSP
HAL_TIM_OC_MspDeInit	将输出比较相关的 MSP 设为缺省值
HAL_TIM_OC_Start	开启输出比较模式
HAL_TIM_OC_Stop	关闭输出比较模式
HAL_TIM_OC_Start_IT	开启输出比较模式和比较中断
HAL_TIM_OC_Stop_IT	关闭输出比较模式和比较中断
HAL_TIM_OC_Start_DMA	开启输出比较模式和产生比较事件时的DMA 请求
HAL_TIM_OC_Stop_DMA	关闭输出比较模式和产生比较事件时的DMA 请求
HAL_TIM_PWM_Init	初始化 PWM 模式
HAL_TIM_PWM_DeInit	将 PWM 模式配置设为缺省值
HAL_TIM_PWM_MspInit	初始化 PWM 模式相关 MSP
HAL_TIM_PWM_MspDeInit	将 PWM 模式相关的MSP 设为缺省值
HAL_TIM_PWM_Start	开启PWM 模式
HAL_TIM_PWM_Stop	关闭PWM 模式
HAL_TIM_PWM_Start_IT	开启PWM 模式和捕获/比较中断
HAL_TIM_PWM_Stop_IT	关闭PWM 模式和捕获/比较中断

HAL_TIM_PWM_Start_DMA	开启PWM 模式和产生比较事件时的 DMA 请求
HAL_TIM_PWM_Stop_DMA	关闭PWM 模式和产生比较事件时的 DMA 请求
HAL_TIM_IC_Init	初始化输入捕获模式
HAL_TIM_IC_DeInit	将输入捕获模式配置设为缺省值
HAL_TIM_IC_MspInit	初始化输入捕获模式相关的 MSP
HAL_TIM_IC_MspDeInit	将输入捕获模式相关的 MSP 设为缺省值
HAL_TIM_IC_Start	开启输入捕获模式
HAL_TIM_IC_Stop	关闭输入捕获模式
HAL_TIM_IC_Start_IT	开启输入捕获模式和捕获中断
HAL_TIM_IC_Stop_IT	关闭输入捕获模式和捕获中断
HAL_TIM_IC_Start_DMA	开启输入捕获模式和产生捕获事件时 DMA 请求
HAL_TIM_IC_Stop_DMA	关闭输入捕获模式和产生捕获事件时 DMA 请求
HAL_TIM_OnePulse_Init	初始化单脉冲模式
HAL_TIM_OnePulse_DeInit	将单脉冲模式配置设为缺省值
HAL_TIM_OnePulse_MspInit	初始化单脉冲相关的 MSP
HAL_TIM_OnePulse_MspDeInit	将单脉冲相关的 MSP 设为缺省值
HAL_TIM_OnePulse_Start	开启单脉冲模式
HAL_TIM_OnePulse_Stop	关闭单脉冲模式
HAL_TIM_OnePulse_Start_IT	开启单脉冲模式和捕获/比较中断
HAL_TIM_OnePulse_Stop_IT	关闭单脉冲模式和捕获/比较中断
HAL_TIM_Encoder_Init	初始化编码器接口模式
HAL_TIM_Encoder_DeInit	将编码器接口模式配置设为缺省值
HAL_TIM_Encoder_MspInit	初始化编码器接口模式相关的 MSP
HAL_TIM_Encoder_MspDeInit	将编码器接口模式相关的MSP 设为缺省值
HAL_TIM_Encoder_Start	开启编码器接口模式
HAL_TIM_Encoder_Stop	关闭编码器接口模式
HAL_TIM_Encoder_Start_IT	开启编码器接口模式和捕获/比较中断
HAL_TIM_Encoder_Stop_IT	关闭编码器接口模式和捕获/比较中断
HAL_TIM_Encoder_Start_DMA	开启编码器接口模式和产生捕获事件时DMA 请求
HAL_TIM_Encoder_Stop_DMA	关闭编码器接口模式和产生捕获事件时DMA 请求
HAL_TIM_IRQHandler	中断请求处理

HAL_TIM_OC_ConfigChannel	输出比较模式通道配置
HAL_TIM_PWM_ConfigChannel	PWM 模式通道配置
HAL_TIM_IC_ConfigChannel	输入捕获模式通道配置
HAL_TIM_OnePulse_ConfigChannel	单脉冲模式通道配置
HAL_TIM_ConfigOCrefClear	配置外部事件清除 OCREF 信号
HAL_TIM_ConfigClockSource	配置时钟源
HAL_TIM_ConfigTI1Input	配置TI1 的输入信号
HAL_TIM_SlaveConfigSynchro	配置从模式
HAL_TIM_SlaveConfigSynchro_IT	配置从模式并开启触发中断
HAL_TIM_DMABurst_WriteStart	使用DMA 突发模式将数据写入到TIM 寄存器
HAL_TIM_DMABurst_MultiWriteStart	使用DMA 突发模式将多个数据写入到TIM 寄存器
HAL_TIM_DMABurst_WriteStop	停止DMA 突发模式写入数据
HAL_TIM_DMABurst_ReadStart	使用DMA 突发模式将数据从TIM 读取到存储器
HAL_TIM_DMABurst_MultiReadStart	使用 DMA 突发模式将大量数据从 TIM 读取到存储器
HAL_TIM_DMABurst_ReadStop	停止DMA 突发模式读取数据
HAL_TIM_GenerateEvent	软件产生一个事件
HAL_TIM_ReadCapturedValue	读取捕获/比较寄存器值
HAL_TIM_PeriodElapsedCallback	计数周期回调函数
HAL_TIM_PeriodElapsedHalfCpltCallback	计数周期半完成回调函数
HAL_TIM_OC_DelayElapsedCallback	输出比较回调函数
HAL_TIM_IC_CaptureCallback	输入捕获回调函数
HAL_TIM_IC_CaptureHalfCpltCallback	输入捕获半完成回调函数
HAL_TIM_PWM_PulseFinishedCallback	PWM 脉冲回调函数
HAL_TIM_PWM_PulseFinishedHalfCpltCallback	PWM 脉冲半完成回调函数
HAL_TIM_TriggerCallback	霍尔触发检测回调函数
HAL_TIM_TriggerHalfCpltCallback	霍尔触发检测半完成回调函数
HAL_TIM_ErrorCallback	错误回调函数
HAL_TIM_Base_GetState	获取时基单元状态
HAL_TIM_OC_GetState	获取输出比较模式状态
HAL_TIM_PWM_GetState	获取PWM 模式状态

HAL_TIM_IC_GetState	获取输入捕获模式状态
HAL_TIM_OnePulse_GetState	获取单脉冲模式状态
HAL_TIM_Encoder_GetState	获取编码器接口模式状态

23.2.1 函数 HAL_TIM_Base_Init

描述了函数 HAL_TIM_Base_Init

表23-57 函数 HAL_TIM_Base_Init

函数名	HAL_TIM_Base_Init
函数原形	HAL_StatusTypeDef HAL_TIM_Base_Init(TIM_HandleTypeDef *htim)
功能描述	初始化时基单元
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.2 函数 HAL_TIM_Base_DeInit

描述了函数 HAL_TIM_Base_DeInit

表23-58 函数 HAL_TIM_Base_DeInit

函数名	HAL_TIM_Base_DeInit
函数原形	HAL_StatusTypeDef HAL_TIM_Base_DeInit(TIM_HandleTypeDef *htim)
功能描述	将时基单元配置设为缺省值
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.3 函数 HAL_TIM_Base_MspInit

描述了函数 HAL_TIM_Base_MspInit

表23-59 函数 HAL_TIM_Base_MspInit

函数名	HAL_TIM_Base_MspInit
函数原形	void HAL_TIM_Base_MspInit(TIM_HandleTypeDef *htim)
功能描述	初始化时单元式相关的 MSP
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.4 函数 HAL_TIM_Base_MspDeInit

描述了函数 HAL_TIM_Base_MspDeInit

表23-60 函数 HAL_TIM_Base_MspDeInit

函数名	HAL_TIM_Base_MspDeInit
函数原形	void HAL_TIM_Base_MspDeInit(TIM_HandleTypeDef *htim)
功能描述	将时基单元相关的 MSP 设为缺省值
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.5 函数 HAL_TIM_Base_Start

描述了函数 HAL_TIM_Base_Start

表23-61 函数 HAL_TIM_Base_Start

函数名	HAL_TIM_Base_Start
函数原形	HAL_StatusTypeDef HAL_TIM_Base_Start(TIM_HandleTypeDef *htim)
功能描述	开启时基的产生
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.6 函数 HAL_TIM_Base_Stop

描述了函数 HAL_TIM_Base_Stop

表23-62 函数 HAL_TIM_Base_Stop

函数名	HAL_TIM_Base_Stop
函数原形	HAL_StatusTypeDef HAL_TIM_Base_Stop(TIM_HandleTypeDef *htim)
功能描述	关闭时基的产生
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.7 函数 HAL_TIM_Base_Start_IT

描述了函数 HAL_TIM_Base_Start_IT

表23-63 函数 HAL_TIM_Base_Start_IT

函数名	HAL_TIM_Base_Start_IT
函数原形	HAL_StatusTypeDef HAL_TIM_Base_Start_IT(TIM_HandleTypeDef *htim)

功能描述	开启时基的产生和更新中断
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.8 函数 HAL_TIM_Base_Stop_IT

描述了函数 HAL_TIM_Base_Stop_IT

表23-64 函数 HAL_TIM_Base_Stop_IT

函数名	HAL_TIM_Base_Stop_IT
函数原形	HAL_StatusTypeDef HAL_TIM_Base_Stop_IT(TIM_HandleTypeDef *htim)
功能描述	关闭时基的产生和更新中断
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.9 函数 HAL_TIM_Base_Start_DMA

描述了函数 HAL_TIM_Base_Start_DMA

表23-65 函数 HAL_TIM_Base_Start_DMA

函数名	HAL_TIM_Base_Start_DMA
函数原形	HAL_StatusTypeDef HAL_TIM_Base_Start_DMA(TIM_HandleTypeDef *htim, uint32_t *pData, uint16_t Length)
功能描述	开启时基的产生和产生更新事件时的 DMA 请求
输入参数 1	htim：TIM 句柄
输入参数 2	pData：数据缓冲区指针
输入参数 3	Length：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.10 函数 HAL_TIM_Base_Stop_DMA

描述了函数 HAL_TIM_Base_Stop_DMA

表23-66 函数 HAL_TIM_Base_Stop_DMA

函数名	HAL_TIM_Base_Stop_DMA
函数原形	HAL_StatusTypeDef HAL_TIM_Base_Stop_DMA(TIM_HandleTypeDef *htim)
功能描述	关闭时基的产生和产生更新事件时的 DMA 请求
输入参数	htim：TIM 句柄
输出参数	无

返回值	HAL 状态
先决条件	无

23.2.11 函数 HAL_TIM_OC_Init

描述了函数 HAL_TIM_OC_Init

表23-67 函数 HAL_TIM_OC_Init

函数名	HAL_TIM_OC_Init
函数原形	HAL_StatusTypeDef HAL_TIM_OC_Init(TIM_HandleTypeDef *htim)
功能描述	初始化输出比较模式
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.12 函数 HAL_TIM_OC_DeInit

描述了函数 HAL_TIM_OC_DeInit

表23-68 函数 HAL_TIM_OC_DeInit

函数名	HAL_TIM_OC_DeInit
函数原形	HAL_StatusTypeDef HAL_TIM_OC_DeInit(TIM_HandleTypeDef *htim)
功能描述	将输出比较模式配置设为缺省值
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.13 函数 HAL_TIM_OC_MspInit

描述了函数 HAL_TIM_OC_MspInit

表23-69 函数 HAL_TIM_OC_MspInit

函数名	HAL_TIM_OC_MspInit
函数原形	void HAL_TIM_OC_MspInit(TIM_HandleTypeDef *htim)
功能描述	初始化输出比较模式相关的 MSP
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.14 函数 HAL_TIM_OC_MspDeInit

描述了函数 HAL_TIM_OC_MspDeInit

表23-70 函数 HAL_TIM_OC_MspDeInit

函数名	HAL_TIM_OC_MspDeInit
函数原形	void HAL_TIM_OC_MspDeInit(TIM_HandleTypeDef *htim)
功能描述	将输出比较相关的 MSP 设为缺省值
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.15 函数 HAL_TIM_OC_Start

描述了函数 HAL_TIM_OC_Start

表23-71 函数 HAL_TIM_OC_Start

函数名	HAL_TIM_OC_Start
函数原形	HAL_StatusTypeDef HAL_TIM_OC_Start(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	开启输出比较模式
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数:

表23-72 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.16 函数 HAL_TIM_OC_Stop

描述了函数 HAL_TIM_OC_Stop

表23-73 函数 HAL_TIM_OC_Stop

函数名	HAL_TIM_OC_Stop
函数原形	HAL_StatusTypeDef HAL_TIM_OC_Stop(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	关闭输出比较模式
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道

输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-74 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.17 函数 HAL_TIM_OC_Start_IT

描述了函数 HAL_TIM_OC_Start_IT

表23-75 函数 HAL_TIM_OC_Start_IT

函数名	HAL_TIM_OC_Start_IT
函数原形	HAL_StatusTypeDef HAL_TIM_OC_Start_IT(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	开启输出比较模式和捕获/比较中断
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-76 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.18 函数 HAL_TIM_OC_Stop_IT

描述了函数 HAL_TIM_OC_Stop_IT

表23-77 函数 HAL_TIM_OC_Stop_IT

函数名	HAL_TIM_OC_Stop_IT
函数原形	HAL_StatusTypeDef HAL_TIM_OC_Stop_IT(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	关闭输出比较模式和比较中断

输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-78 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.19 函数 HAL_TIM_OC_Start_DMA

描述了函数 HAL_TIM_OC_Start_DMA

表23-79 函数 HAL_TIM_OC_Start_DMA

函数名	HAL_TIM_OC_Start_DMA
函数原形	HAL_StatusTypeDef HAL_TIM_OC_Start_DMA(TIM_HandleTypeDef *htim, uint32_t Channel, uint32_t *pData, uint16_t Length)
功能描述	开启输出比较模式和产生比较事件时 DMA 请求
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输入参数 3	pData：数据缓冲区指针
输入参数 4	Length：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-80 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.20 函数 HAL_TIM_OC_Stop_DMA

描述了函数 HAL_TIM_OC_Stop_DMA

表23-81 函数 HAL_TIM_OC_Stop_DMA

函数名	HAL_TIM_OC_Stop_DMA
函数原形	HAL_StatusTypeDef HAL_TIM_OC_Stop_DMA(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	关闭输出比较模式和产生比较事件时 DMA 请求
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-82 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.21 函数 HAL_TIM_PWM_Init

描述了函数 HAL_TIM_PWM_Init

表23-83 函数 HAL_TIM_PWM_Init

函数名	HAL_TIM_PWM_Init
函数原形	HAL_StatusTypeDef HAL_TIM_PWM_Init(TIM_HandleTypeDef *htim)
功能描述	初始化 PWM 模式
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.22 函数 HAL_TIM_PWM_DeInit

描述了函数 HAL_TIM_PWM_DeInit

表23-84 函数 HAL_TIM_PWM_DeInit

函数名	HAL_TIM_PWM_DeInit
函数原形	HAL_StatusTypeDef HAL_TIM_PWM_DeInit(TIM_HandleTypeDef *htim)
功能描述	将 PWM 模式配置设为缺省值
输入参数	htim：TIM 句柄
输出参数	无

返回值	HAL 状态
先决条件	无

23.2.23 函数 HAL_TIM_PWM_MspInit

描述了函数 HAL_TIM_PWM_MspInit

表23-85 函数 HAL_TIM_PWM_MspInit

函数名	HAL_TIM_PWM_MspInit
函数原形	void HAL_TIM_PWM_MspInit(TIM_HandleTypeDef *htim)
功能描述	初始化 PWM 模式相关 MSP
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.24 函数 HAL_TIM_PWM_MspDeInit

描述了函数 HAL_TIM_PWM_MspDeInit

表23-86 函数 HAL_TIM_PWM_MspDeInit

函数名	HAL_TIM_PWM_MspDeInit
函数原形	void HAL_TIM_PWM_MspDeInit(TIM_HandleTypeDef *htim)
功能描述	将 PWM 模式相关的 MSP 设为缺省值
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.25 函数 HAL_TIM_PWM_Start

描述了函数 HAL_TIM_PWM_Start

表23-87 函数 HAL_TIM_PWM_Start

函数名	HAL_TIM_PWM_Start
函数原形	HAL_StatusTypeDef HAL_TIM_PWM_Start(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	开启 PWM 模式
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-88 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.26 函数 HAL_TIM_PWM_Stop

描述了函数 HAL_TIM_PWM_Stop

表23-89 函数 HAL_TIM_PWM_Stop

函数名	HAL_TIM_PWM_Stop
函数原形	HAL_StatusTypeDef HAL_TIM_PWM_Stop(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	关闭 PWM 模式
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-90 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.27 函数 HAL_TIM_PWM_Start_IT

描述了函数 HAL_TIM_PWM_Start_IT

表23-91 函数 HAL_TIM_PWM_Start_IT

函数名	HAL_TIM_PWM_Start_IT
函数原形	HAL_StatusTypeDef HAL_TIM_PWM_Start_IT(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	开启 PWM 模式和比较中断
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态

先决条件	无

Channel 可选参数：

表23-92 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.28 函数 HAL_TIM_PWM_Stop_IT

描述了函数 HAL_TIM_PWM_Stop_IT

表23-93 函数 HAL_TIM_PWM_Stop_IT

函数名	HAL_TIM_PWM_Stop_IT
函数原形	HAL_StatusTypeDef HAL_TIM_PWM_Stop_IT(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	关闭 PWM 模式和比较中断
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-94 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.29 函数 HAL_TIM_PWM_Start_DMA

描述了函数 HAL_TIM_PWM_Start_DMA

表23-95 函数 HAL_TIM_PWM_Start_DMA

函数名	HAL_TIM_PWM_Start_DMA
函数原形	HAL_StatusTypeDef HAL_TIM_PWM_Start_DMA(TIM_HandleTypeDef *htim, uint32_t Channel, uint32_t *pData, uint16_t Length)
功能描述	开启 PWM 模式和产生比较事件时的 DMA 请求
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道

输入参数 3	pData：数据缓冲区指针
输入参数 4	Length：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-96 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.30 函数 HAL_TIM_PWM_Stop_DMA

描述了函数 HAL_TIM_PWM_Stop_DMA

表23-97 函数 HAL_TIM_PWM_Stop_DMA

函数名	HAL_TIM_PWM_Stop_DMA
函数原形	HAL_StatusTypeDef HAL_TIM_PWM_Stop_DMA(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	关闭 PWM 模式和产生比较事件时的 DMA 请求
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-98 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.31 函数 HAL_TIM_IC_Init

描述了函数 HAL_TIM_IC_Init

表23-99 函数 HAL_TIM_IC_Init

函数名	HAL_TIM_IC_Init

函数原形	HAL_StatusTypeDef HAL_TIM_IC_Init(TIM_HandleTypeDef *htim)
功能描述	初始化输入捕获模式
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.32 函数 HAL_TIM_IC_DeInit

描述了函数 HAL_TIM_IC_DeInit

表23-100 函数 HAL_TIM_IC_DeInit

函数名	HAL_TIM_IC_DeInit
函数原形	HAL_StatusTypeDef HAL_TIM_IC_DeInit(TIM_HandleTypeDef *htim)
功能描述	将输入捕获模式配置设为缺省值
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.33 函数 HAL_TIM_IC_MspInit

描述了函数 HAL_TIM_IC_MspInit

表23-101 函数 HAL_TIM_IC_MspInit

函数名	HAL_TIM_IC_MspInit
函数原形	void HAL_TIM_IC_MspInit(TIM_HandleTypeDef *htim)
功能描述	初始化输入捕获模式相关的 MSP
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.34 函数 HAL_TIM_IC_MspDeInit

描述了函数 HAL_TIM_IC_MspDeInit

表23-102 函数 HAL_TIM_IC_MspDeInit

函数名	HAL_TIM_IC_MspDeInit
函数原形	void HAL_TIM_IC_MspDeInit(TIM_HandleTypeDef *htim)
功能描述	将输入捕获模式相关的 MSP 设为缺省值
输入参数	htim：TIM 句柄
输出参数	无
返回值	无

先决条件	无

23.2.35 函数 HAL_TIM_IC_Start

描述了函数 HAL_TIM_IC_Start

表23-103 函数 HAL_TIM_IC_Start

函数名	HAL_TIM_IC_Start
函数原形	HAL_StatusTypeDef HAL_TIM_IC_Start(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	开启输入捕获模式
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输入通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-104 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.36 函数 HAL_TIM_IC_Stop

描述了函数 HAL_TIM_IC_Stop

表23-105 函数 HAL_TIM_IC_Stop

函数名	HAL_TIM_IC_Stop
函数原形	HAL_StatusTypeDef HAL_TIM_IC_Stop(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	关闭输入捕获模式
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输入通道
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.37 函数 HAL_TIM_IC_Start_IT

描述了函数 HAL_TIM_IC_Start_IT

表23-106 函数 HAL_TIM_IC_Start_IT

函数名	HAL_TIM_IC_Start_IT

函数原形	HAL_StatusTypeDef HAL_TIM_IC_Start_IT(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	开启输入捕获模式和捕获中断
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输入通道
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.38 函数 HAL_TIM_IC_Stop_IT

描述了函数 HAL_TIM_IC_Stop_IT

表23-107 函数 HAL_TIM_IC_Stop_IT

函数名	HAL_TIM_IC_Stop_IT
函数原形	HAL_StatusTypeDef HAL_TIM_IC_Stop_IT(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	关闭输入捕获模式和捕获中断
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输入通道
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.39 函数 HAL_TIM_IC_Start_DMA

描述了函数 HAL_TIM_IC_Start_DMA

表23-108 函数 HAL_TIM_IC_Start_DMA

函数名	HAL_TIM_IC_Start_DMA
函数原形	HAL_StatusTypeDef HAL_TIM_IC_Start_DMA(TIM_HandleTypeDef *htim, uint32_t Channel, uint32_t *pData, uint16_t Length)
功能描述	开启输入捕获模式和产生捕获事件时 DMA 请求
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输入通道
输入参数 3	pData：数据缓冲区指针
输入参数 4	Length：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-109 Channel 可选参数

参数	描述

TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.40 函数 HAL_TIM_IC_Stop_DMA

描述了函数 HAL_TIM_IC_Stop_DMA

表23-110 函数 HAL_TIM_IC_Stop_DMA

函数名	HAL_TIM_IC_Stop_DMA
函数原形	HAL_StatusTypeDef HAL_TIM_IC_Stop_DMA(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	关闭输入捕获模式和产生捕获事件时 DMA 请求
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输入通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数:

表23-111 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.41 函数 HAL_TIM_OnePulse_Init

描述了函数 HAL_TIM_OnePulse_Init

表23-112 函数 HAL_TIM_OnePulse_Init

函数名	HAL_TIM_OnePulse_Init
函数原形	HAL_StatusTypeDef HAL_TIM_OnePulse_Init(TIM_HandleTypeDef *htim, uint32_t OnePulseMode)
功能描述	初始化单脉冲模式
输入参数 1	htim：TIM 句柄
输入参数 2	OnePulseMode：脉冲模式
输出参数	无
返回值	HAL 状态
先决条件	无

OnePulseMode 可选参数:

表23-113 OnePulseMode 可选参数

参数	描述
TIM_OPMODE_SINGLE	单次触发模式
TIM_OPMODE_REPETITIVE	循环触发模式

23.2.42 函数 HAL_TIM_OnePulse_DeInit

描述了函数 HAL_TIM_OnePulse_DeInit

表23-114 函数 HAL_TIM_OnePulse_DeInit

函数名	HAL_TIM_OnePulse_DeInit
函数原形	HAL_StatusTypeDef HAL_TIM_OnePulse_DeInit(TIM_HandleTypeDef *htim)
功能描述	将单脉冲模式配置设为缺省值
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.43 函数 HAL_TIM_OnePulse_MspInit

描述了函数 HAL_TIM_OnePulse_MspInit

表23-115 函数 HAL_TIM_OnePulse_MspInit

函数名	HAL_TIM_OnePulse_MspInit
函数原形	void HAL_TIM_OnePulse_MspInit(TIM_HandleTypeDef *htim)
功能描述	初始化单脉冲相关的 MSP
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.44 函数 HAL_TIM_OnePulse_MspDeInit

描述了函数 HAL_TIM_OnePulse_MspDeInit

表23-116 函数 HAL_TIM_OnePulse_MspDeInit

函数名	HAL_TIM_OnePulse_MspDeInit
函数原形	void HAL_TIM_OnePulse_MspDeInit(TIM_HandleTypeDef *htim)
功能描述	将单脉冲相关的 MSP 设为缺省值
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.45 函数 HAL_TIM_OnePulse_Start

描述了函数 HAL_TIM_OnePulse_Start

表23-117 函数 HAL_TIM_OnePulse_Start

函数名	HAL_TIM_OnePulse_Start
函数原形	HAL_StatusTypeDef HAL_TIM_OnePulse_Start(TIM_HandleTypeDef *htim, uint32_t OutputChannel)
功能描述	开启单脉冲模式
输入参数 1	htim：TIM 句柄
输入参数 2	OutputChannel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

OutputChannel 可选参数:

表23-118 OutputChannel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2

23.2.46 函数 HAL_TIM_OnePulse_Stop

描述了函数 HAL_TIM_OnePulse_Stop

表23-119 函数 HAL_TIM_OnePulse_Stop

函数名	HAL_TIM_OnePulse_Stop
函数原形	HAL_StatusTypeDef HAL_TIM_OnePulse_Stop(TIM_HandleTypeDef *htim, uint32_t OutputChannel)
功能描述	关闭单脉冲模式
输入参数 1	htim：TIM 句柄
输入参数 2	OutputChannel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

OutputChannel 可选参数:

表23-120 OutputChannel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2

23.2.47 函数 HAL_TIM_OnePulse_Start_IT

描述了函数 HAL_TIM_OnePulse_Start_IT

表23-121 函数 HAL_TIM_OnePulse_Start_IT

函数名	HAL_TIM_OnePulse_Start_IT
函数原形	HAL_StatusTypeDef HAL_TIM_OnePulse_Start_IT(TIM_HandleTypeDef *htim, uint32_t OutputChannel)
功能描述	开启单脉冲模式和捕获/比较中断
输入参数 1	htim：TIM 句柄
输入参数 2	OutputChannel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

OutputChannel 可选参数:

表23-122 OutputChannel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2

23.2.48 函数 HAL_TIM_OnePulse_Stop_IT

描述了函数 HAL_TIM_OnePulse_Stop_IT

表23-123 函数 HAL_TIM_OnePulse_Stop_IT

函数名	HAL_TIM_OnePulse_Stop_IT
函数原形	HAL_StatusTypeDef HAL_TIM_OnePulse_Stop_IT(TIM_HandleTypeDef *htim, uint32_t OutputChannel)
功能描述	关闭单脉冲模式和捕获/比较中断
输入参数 1	htim：TIM 句柄
输入参数 2	OutputChannel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

OutputChannel 可选参数:

表23-124 OutputChannel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2

23.2.49 函数 HAL_TIM_Encoder_Init

描述了函数 HAL_TIM_Encoder_Init

表23-125 函数 HAL_TIM_Encoder_Init

函数名	HAL_TIM_Encoder_Init

函数原形	HAL_StatusTypeDef HAL_TIM_Encoder_Init(TIM_HandleTypeDef *htim, TIM_Encoder_InitTypeDef *sConfig)
功能描述	初始化编码器接口模式
输入参数 1	htim：TIM 句柄
输入参数 2	sConfig：Encoder 初始化配置结构体
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.50 函数 HAL_TIM_Encoder_DeInit

描述了函数 HAL_TIM_Encoder_DeInit

表23-126 函数 HAL_TIM_Encoder_DeInit

函数名	HAL_TIM_Encoder_DeInit
函数原形	HAL_StatusTypeDef HAL_TIM_Encoder_DeInit(TIM_HandleTypeDef *htim)
功能描述	将编码器接口模式配置设为缺省值
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.51 函数 HAL_TIM_Encoder_MspInit

描述了函数 HAL_TIM_Encoder_MspInit

表23-127 函数 HAL_TIM_Encoder_MspInit

函数名	HAL_TIM_Encoder_MspInit
函数原形	void HAL_TIM_Encoder_MspInit(TIM_HandleTypeDef *htim)
功能描述	初始化编码器接口模式相关的 MSP
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.52 函数 HAL_TIM_Encoder_MspDeInit

描述了函数 HAL_TIM_Encoder_MspDeInit

表23-128 函数 HAL_TIM_Encoder_MspDeInit

函数名	HAL_TIM_Encoder_MspDeInit
函数原形	void HAL_TIM_Encoder_MspDeInit(TIM_HandleTypeDef *htim)
功能描述	将编码器接口模式相关的 MSP 设为缺省值
输入参数	htim：TIM 句柄
输出参数	无

返回值	无
先决条件	无

23.2.53 函数 HAL_TIM_Encoder_Start

描述了函数 HAL_TIM_Encoder_Start

表23-129 函数 HAL_TIM_Encoder_Start

函数名	HAL_TIM_Encoder_Start
函数原形	HAL_StatusTypeDef HAL_TIM_Encoder_Start(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	开启编码器接口模式
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输入通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-130 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_ALL	选择通道 1 和通道 2

23.2.54 函数 HAL_TIM_Encoder_Stop

描述了函数 HAL_TIM_Encoder_Stop

表23-131 函数 HAL_TIM_Encoder_Stop

函数名	HAL_TIM_Encoder_Stop
函数原形	HAL_StatusTypeDef HAL_TIM_Encoder_Stop(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	关闭编码器接口模式
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输入通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-132 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1

TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_ALL	选择通道 1 和通道 2

23.2.55 函数 HAL_TIM_Encoder_Start_IT

描述了函数 HAL_TIM_Encoder_Start_IT

表23-133 函数 HAL_TIM_Encoder_Start_IT

函数名	HAL_TIM_Encoder_Start_IT
函数原形	HAL_StatusTypeDef HAL_TIM_Encoder_Start_IT(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	开启编码器接口模式和捕获中断
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输入通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-134 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_ALL	选择通道 1 和通道 2

23.2.56 函数 HAL_TIM_Encoder_Stop_IT

描述了函数 HAL_TIM_Encoder_Stop_IT

表23-135 函数 HAL_TIM_Encoder_Stop_IT

函数名	HAL_TIM_Encoder_Stop_IT
函数原形	HAL_StatusTypeDef HAL_TIM_Encoder_Stop_IT(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	关闭编码器接口模式和捕获中断
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输入通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-136 Channel 可选参数

参数	描述

TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_ALL	选择通道 1 和通道 2

23.2.57 函数 HAL_TIM_Encoder_Start_DMA

描述了函数 HAL_TIM_Encoder_Start_DMA

表23-137 函数 HAL_TIM_Encoder_Start_DMA

函数名	HAL_TIM_Encoder_Start_DMA
函数原形	HAL_StatusTypeDef HAL_TIM_Encoder_Start_DMA(TIM_HandleTypeDef *htim, uint32_t Channel, uint32_t *pData1, uint32_t *pData2, uint16_t Length)
功能描述	开启编码器接口模式和产生捕获事件时 DMA 请求
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输入通道
输入参数 3	pData1：IC1 数据缓冲区地址
输入参数 4	pData2：IC2 数据缓冲区地址
输入参数 5	Length：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-138 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_ALL	选择通道 1 和通道 2

23.2.58 函数 HAL_TIM_Encoder_Stop_DMA

描述了函数 HAL_TIM_Encoder_Stop_DMA

表23-139 函数 HAL_TIM_Encoder_Stop_DMA

函数名	HAL_TIM_Encoder_Stop_DMA
函数原形	HAL_StatusTypeDef HAL_TIM_Encoder_Stop_DMA(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	关闭编码器接口模式和产生捕获事件时 DMA 请求
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输入通道
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.59 函数 HAL_TIM_IRQHandler

描述了函数 HAL_TIM_IRQHandler

表23-140 函数 HAL_TIM_IRQHandler

函数名	HAL_TIM_IRQHandler
函数原形	void HAL_TIM_IRQHandler(TIM_HandleTypeDef *htim)
功能描述	中断请求处理
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.60 函数 HAL_TIM_OC_ConfigChannel

描述了函数 HAL_TIM_OC_ConfigChannel

表23-141 函数 HAL_TIM_OC_ConfigChannel

函数名	HAL_TIM_OC_ConfigChannel
函数原形	HAL_StatusTypeDef HAL_TIM_OC_ConfigChannel(TIM_HandleTypeDef *htim, TIM_OC_InitTypeDef *sConfig, uint32_t Channel)
功能描述	输出比较模式通道配置
输入参数 1	htim：TIM 句柄
输入参数 2	sConfig：输出比较初始化配置结构体
输入参数 3	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数:

表23-142 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.61 函数 HAL_TIM_PWM_ConfigChannel

描述了函数 HAL_TIM_PWM_ConfigChannel

表23-143 函数 HAL_TIM_PWM_ConfigChannel

函数名	HAL_TIM_PWM_ConfigChannel
函数原形	HAL_StatusTypeDef HAL_TIM_PWM_ConfigChannel(TIM_HandleTypeDef *htim, TIM_OC_InitTypeDef *sConfig, uint32_t Channel)

功能描述	PWM 模式通道配置
输入参数 1	htim：TIM 句柄
输入参数 2	sConfig：PWM 初始化配置结构体
输入参数 3	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-144 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.62 函数 HAL_TIM_IC_ConfigChannel

描述了函数 HAL_TIM_IC_ConfigChannel

表23-145 函数 HAL_TIM_IC_ConfigChannel

函数名	HAL_TIM_IC_ConfigChannel
函数原形	HAL_StatusTypeDef HAL_TIM_IC_ConfigChannel(TIM_HandleTypeDef *htim, TIM_IC_InitTypeDef *sConfig, uint32_t Channel)
功能描述	输入捕获模式通道配置
输入参数 1	htim：TIM 句柄
输入参数 2	sConfig：输入捕获初始化配置结构体
输入参数 3	Channel：输入通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表23-146 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.63 函数 HAL_TIM_OnePulse_ConfigChannel

描述了函数 HAL_TIM_OnePulse_ConfigChannel

表23-147 函数 HAL_TIM_OnePulse_ConfigChannel

函数名	HAL_TIM_OnePulse_ConfigChannel
函数原形	HAL_StatusTypeDef HAL_TIM_OnePulse_ConfigChannel(TIM_HandleTypeDef *htim, TIM_OnePulse_InitTypeDef *sConfig, uint32_t OutputChannel,uint32_t InputChannel)
功能描述	单脉冲模式通道配置
输入参数 1	htim：TIM 句柄
输入参数 2	sConfig：输入捕获初始化配置结构体
输入参数 3	OutputChannel：输出通道
输入参数 4	InputChannel：输入通道
输出参数	无
返回值	HAL 状态
先决条件	无

OutputChannel 可选参数:

表23-148 OutputChannel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2

InputChannel 可选参数:

表23-149 InputChannel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2

23.2.64 函数 HAL_TIM_ConfigOCrefClear

描述了函数 HAL_TIM_ConfigOCrefClear

表23-150 函数 HAL_TIM_ConfigOCrefClear

函数名	HAL_TIM_ConfigOCrefClear
函数原形	HAL_StatusTypeDef HAL_TIM_ConfigOCrefClear(TIM_HandleTypeDef *htim, TIM_ClearInputConfigTypeDef *sClearInputConfig, uint32_t Channel)
功能描述	配置外部事件清除 OCREF 信号
输入参数 1	htim：TIM 句柄
输入参数 2	sClearInputConfig：外部事件清除 OCREF 信号初始化配置结构体
输入参数 3	Channel：输入通道
输出参数	无

返回值	HAL 状态
先决条件	无

InputChannel 可选参数：

表23-151 InputChannel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.65 函数 HAL_TIM_ConfigClockSource

描述了函数 HAL_TIM_ConfigClockSource

表23-152 函数 HAL_TIM_ConfigClockSource

函数名	HAL_TIM_ConfigClockSource
函数原形	HAL_StatusTypeDef HAL_TIM_ConfigClockSource(TIM_HandleTypeDef *htim, TIM_ClockConfigTypeDef *sClockSourceConfig)
功能描述	配置时钟源
输入参数 1	htim：TIM 句柄
输入参数 2	sClockSourceConfig：时钟源初始化配置结构体
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.66 函数 HAL_TIM_ConfigTI1Input

描述了函数 HAL_TIM_ConfigTI1Input

表23-153 函数 HAL_TIM_ConfigTI1Input

函数名	HAL_TIM_ConfigTI1Input
函数原形	HAL_StatusTypeDef HAL_TIM_ConfigTI1Input(TIM_HandleTypeDef *htim, uint32_t TI1_Selection)
功能描述	配置 TI1 的输入信号
输入参数 1	htim：TIM 句柄
输入参数 2	TI1_Selection：TI1 输入信号选择
输出参数	无
返回值	HAL 状态
先决条件	无

TI1_Selection 可选参数：

表23-154 TI1_Selection 可选参数

参数	描述

TIM_TI1SELECTION_CH1	CH1 管脚连到 TI1 输入
TIM_TI1SELECTION_XORCOMBINATION	CH1、CH2 和 CH3 管脚经异或后连接到 TI1 输入

23.2.67 函数 HAL_TIM_SlaveConfigSynchro

描述了函数 HAL_TIM_SlaveConfigSynchro

表23-155 函数 HAL_TIM_SlaveConfigSynchro

函数名	HAL_TIM_SlaveConfigSynchro
函数原形	HAL_StatusTypeDef HAL_TIM_SlaveConfigSynchro(TIM_HandleTypeDef *htim, TIM_SlaveConfigTypeDef *sSlaveConfig)
功能描述	配置从模式
输入参数 1	htim：TIM 句柄
输入参数 2	sSlaveConfig：从模式初始化配置结构体
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.68 函数 HAL_TIM_SlaveConfigSynchro_IT

描述了函数 HAL_TIM_SlaveConfigSynchro_IT

表23-156 函数 HAL_TIM_SlaveConfigSynchro_IT

函数名	HAL_TIM_SlaveConfigSynchro_IT
函数原形	HAL_StatusTypeDef HAL_TIM_SlaveConfigSynchro_IT(TIM_HandleTypeDef *htim, TIM_SlaveConfigTypeDef *sSlaveConfig)
功能描述	配置从模式并开启触发中断
输入参数 1	htim：TIM 句柄
输入参数 2	sSlaveConfig：从模式初始化配置结构体
输出参数	无
返回值	HAL 状态
先决条件	无

23.2.69 函数 HAL_TIM_DMABurst_WriteStart

描述了函数 HAL_TIM_DMABurst_WriteStart

表23-157 函数 HAL_TIM_DMABurst_WriteStart

函数名	HAL_TIM_DMABurst_WriteStart
函数原形	HAL_StatusTypeDef HAL_TIM_DMABurst_WriteStart(TIM_HandleTypeDef *htim, uint32_t BurstBaseAddress, uint32_t BurstRequestSrc, uint32_t *BurstBuffer, uint32_t BurstLength)
功能描述	使用 DMA 突发模式将数据写入到 TIM 寄存器
输入参数 1	htim：TIM 句柄
输入参数 2	BurstBaseAddress：写入数据的基地址
输入参数 3	BurstRequestSrc：DMA 请求触发源
输入参数 4	BurstBuffer：数据缓冲区指针

输入参数 5	BurstLength：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

BurstBaseAddress 可选参数：

表23-158 BurstBaseAddress 可选参数

参数	描述
TIM_DMABASE_CR1	CR1 寄存器
TIM_DMABASE_CR2	CR2 寄存器
TIM_DMABASE_SMCR	SMCR 寄存器
TIM_DMABASE_DIER	DIER 寄存器
TIM_DMABASE_SR	SR 寄存器
TIM_DMABASE_EGR	EGR 寄存器
TIM_DMABASE_CCMR1	CCMR1 寄存器
TIM_DMABASE_CCMR2	CCMR2 寄存器
TIM_DMABASE_CCER	CCER 寄存器
TIM_DMABASE_CNT	CNT 寄存器
TIM_DMABASE_PSC	PSC 寄存器
TIM_DMABASE_ARR	ARR 寄存器
TIM_DMABASE_RCR	RCR 寄存器
TIM_DMABASE_CCR1	CCR1 寄存器
TIM_DMABASE_CCR2	CCR2 寄存器
TIM_DMABASE_CCR3	CCR3 寄存器
TIM_DMABASE_CCR4	CCR4 寄存器

BurstRequestSrc 可选参数：

表23-159 BurstRequestSrc 可选参数

参数	描述
TIM_DMA_UPDATE	更新中断 DMA 请求
TIM_DMA_CC1	捕获/比较 1 DMA 请求
TIM_DMA_CC2	捕获/比较 2 DMA 请求
TIM_DMA_CC3	捕获/比较 3 DMA 请求
TIM_DMA_CC4	捕获/比较 4 DMA 请求
TIM_DMA_COM	换向 DMA 请求
TIM_DMA_TRIGGER	触发 DMA 请求

23.2.70 函数 HAL_TIM_DMABurst_MultiWriteStart

描述了函数 HAL_TIM_DMABurst_MultiWriteStart

表23-160 函数 HAL_TIM_DMABurst_MultiWriteStart

函数名	HAL_TIM_DMABurst_MultiWriteStart
函数原形	HAL_StatusTypeDef HAL_TIM_DMABurst_MultiWriteStart(TIM_HandleTypeDef *htim, uint32_t BurstBaseAddress, uint32_t BurstRequestSrc, uint32_t *BurstBuffer, uint32_t BurstLength,uint32_t DataLength)
功能描述	使用 DMA 突发模式将多个数据写入到 TIM 寄存器
输入参数 1	htim：TIM 句柄
输入参数 2	BurstBaseAddress：写入数据的基地址
输入参数 3	BurstRequestSrc：DMA 请求触发源
输入参数 4	BurstBuffer：数据缓冲区指针
输入参数 5	BurstLength：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

BurstBaseAddress 可选参数：

表23-161 BurstBaseAddress 可选参数

参数	描述
TIM_DMABASE_CR1	CR1 寄存器
TIM_DMABASE_CR2	CR2 寄存器
TIM_DMABASE_SMCR	SMCR 寄存器
TIM_DMABASE_DIER	DIER 寄存器
TIM_DMABASE_SR	SR 寄存器
TIM_DMABASE_EGR	EGR 寄存器
TIM_DMABASE_CCMR1	CCMR1 寄存器
TIM_DMABASE_CCMR2	CCMR2 寄存器
TIM_DMABASE_CCER	CCER 寄存器
TIM_DMABASE_CNT	CNT 寄存器
TIM_DMABASE_PSC	PSC 寄存器
TIM_DMABASE_ARR	ARR 寄存器
TIM_DMABASE_RCR	RCR 寄存器
TIM_DMABASE_CCR1	CCR1 寄存器
TIM_DMABASE_CCR2	CCR2 寄存器
TIM_DMABASE_CCR3	CCR3 寄存器
TIM_DMABASE_CCR4	CCR4 寄存器

BurstRequestSrc 可选参数：

表23-162 BurstRequestSrc 可选参数

参数	描述
TIM_DMA_UPDATE	更新中断 DMA 请求
TIM_DMA_CC1	捕获/比较 1 DMA 请求
TIM_DMA_CC2	捕获/比较 2 DMA 请求
TIM_DMA_CC3	捕获/比较 3 DMA 请求
TIM_DMA_CC4	捕获/比较 4 DMA 请求
TIM_DMA_COM	换向 DMA 请求
TIM_DMA_TRIGGER	触发 DMA 请求

23.2.71 函数 HAL_TIM_DMABurst_WriteStop

描述了函数 HAL_TIM_DMABurst_WriteStop

表23-163 函数 HAL_TIM_DMABurst_WriteStop

函数名	HAL_TIM_DMABurst_WriteStop
函数原形	HAL_StatusTypeDef HAL_TIM_DMABurst_WriteStop(TIM_HandleTypeDef *htim, uint32_t BurstRequestSrc)
功能描述	停止 DMA 突发模式写入数据
输入参数 1	htim：TIM 句柄
输入参数 2	BurstRequestSrc：DMA 请求触发源
输出参数	无
返回值	HAL 状态
先决条件	无

BurstRequestSrc 可选参数：

表23-164 BurstRequestSrc 可选参数

参数	描述
TIM_DMA_UPDATE	更新中断 DMA 请求
TIM_DMA_CC1	捕获/比较 1 DMA 请求
TIM_DMA_CC2	捕获/比较 2 DMA 请求
TIM_DMA_CC3	捕获/比较 3 DMA 请求
TIM_DMA_CC4	捕获/比较 4 DMA 请求
TIM_DMA_COM	换向 DMA 请求
TIM_DMA_TRIGGER	触发 DMA 请求

23.2.72 函数 HAL_TIM_DMABurst_ReadStart

描述了函数 HAL_TIM_DMABurst_ReadStart

表23-165 函数 HAL_TIM_DMABurst_ReadStart

函数名	HAL_TIM_DMABurst_ReadStart
函数原形	HAL_StatusTypeDef HAL_TIM_DMABurst_ReadStart(TIM_HandleTypeDef *htim, uint32_t BurstBaseAddress, uint32_t BurstRequestSrc, uint32_t *BurstBuffer, uint32_t BurstLength)
功能描述	使用 DMA 突发模式将数据从 TIM 读取到存储器
输入参数 1	htim：TIM 句柄
输入参数 2	BurstBaseAddress：读取数据的基地址
输入参数 3	BurstRequestSrc：DMA 请求触发源
输入参数 4	BurstBuffer：数据缓冲区指针
输入参数 5	BurstLength：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

BurstBaseAddress 可选参数：

表23-166 BurstBaseAddress 可选参数

参数	描述
TIM_DMABASE_CR1	CR1 寄存器
TIM_DMABASE_CR2	CR2 寄存器
TIM_DMABASE_SMCR	SMCR 寄存器
TIM_DMABASE_DIER	DIER 寄存器
TIM_DMABASE_SR	SR 寄存器
TIM_DMABASE_EGR	EGR 寄存器
TIM_DMABASE_CCMR1	CCMR1 寄存器
TIM_DMABASE_CCMR2	CCMR2 寄存器
TIM_DMABASE_CCER	CCER 寄存器
TIM_DMABASE_CNT	CNT 寄存器
TIM_DMABASE_PSC	PSC 寄存器
TIM_DMABASE_ARR	ARR 寄存器
TIM_DMABASE_RCR	RCR 寄存器
TIM_DMABASE_CCR1	CCR1 寄存器
TIM_DMABASE_CCR2	CCR2 寄存器
TIM_DMABASE_CCR3	CCR3 寄存器
TIM_DMABASE_CCR4	CCR4 寄存器

BurstRequestSrc 可选参数：

表23-167 BurstRequestSrc 可选参数

参数	描述
TIM_DMA_UPDATE	更新中断 DMA 请求
TIM_DMA_CC1	捕获/比较 1 DMA 请求
TIM_DMA_CC2	捕获/比较 2 DMA 请求
TIM_DMA_CC3	捕获/比较 3 DMA 请求
TIM_DMA_CC4	捕获/比较 4 DMA 请求
TIM_DMA_COM	换向 DMA 请求
TIM_DMA_TRIGGER	触发 DMA 请求

23.2.73 函数 HAL_TIM_DMABurst_MultiReadStart

描述了函数 HAL_TIM_DMABurst_MultiReadStart

表23-168 函数 HAL_TIM_DMABurst_MultiReadStart

函数名	HAL_TIM_DMABurst_MultiReadStart
函数原形	HAL_StatusTypeDef HAL_TIM_DMABurst_MultiReadStart(TIM_HandleTypeDef *htim, uint32_t BurstBaseAddress, uint32_t BurstRequestSrc, uint32_t *BurstBuffer, uint32_t BurstLength,uint32_t DataLength)
功能描述	使用 DMA 突发模式将大量数据从 TIM 读取到存储器
输入参数 1	htim：TIM 句柄
输入参数 2	BurstBaseAddress：读取数据的基地址
输入参数 3	BurstRequestSrc：DMA 请求触发源
输入参数 4	BurstBuffer：数据缓冲区指针
输入参数 5	BurstLength：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

BurstBaseAddress 可选参数:

表23-169 BurstBaseAddress 可选参数

参数	描述
TIM_DMABASE_CR1	CR1 寄存器
TIM_DMABASE_CR2	CR2 寄存器
TIM_DMABASE_SMCR	SMCR 寄存器
TIM_DMABASE_DIER	DIER 寄存器
TIM_DMABASE_SR	SR 寄存器
TIM_DMABASE_EGR	EGR 寄存器
TIM_DMABASE_CCMR1	CCMR1 寄存器
TIM_DMABASE_CCMR2	CCMR2 寄存器

TIM_DMABASE_CCER	CCER 寄存器
TIM_DMABASE_CNT	CNT 寄存器
TIM_DMABASE_PSC	PSC 寄存器
TIM_DMABASE_ARR	ARR 寄存器
TIM_DMABASE_RCR	RCR 寄存器
TIM_DMABASE_CCR1	CCR1 寄存器
TIM_DMABASE_CCR2	CCR2 寄存器
TIM_DMABASE_CCR3	CCR3 寄存器
TIM_DMABASE_CCR4	CCR4 寄存器

BurstRequestSrc 可选参数:

表23-170 BurstRequestSrc 可选参数

参数	描述
TIM_DMA_UPDATE	更新中断 DMA 请求
TIM_DMA_CC1	捕获/比较 1 DMA 请求
TIM_DMA_CC2	捕获/比较 2 DMA 请求
TIM_DMA_CC3	捕获/比较 3 DMA 请求
TIM_DMA_CC4	捕获/比较 4 DMA 请求
TIM_DMA_COM	换向 DMA 请求
TIM_DMA_TRIGGER	触发 DMA 请求

23.2.74 函数 HAL_TIM_DMABurst_ReadStop

描述了函数 HAL_TIM_DMABurst_ReadStop

表23-171 函数 HAL_TIM_DMABurst_ReadStop

函数名	HAL_TIM_DMABurst_ReadStop
函数原形	HAL_StatusTypeDef HAL_TIM_DMABurst_ReadStop(TIM_HandleTypeDef *htim, uint32_t BurstRequestSrc)
功能描述	停止 DMA 突发模式读取数据
输入参数 1	htim：TIM 句柄
输入参数 2	BurstRequestSrc：DMA 请求触发源
输出参数	无
返回值	HAL 状态
先决条件	无

BurstRequestSrc 可选参数:

表23-172 BurstRequestSrc 可选参数

参数	描述
TIM_DMA_UPDATE	更新中断 DMA 请求

TIM_DMA_CC1	捕获/比较 1 DMA 请求
TIM_DMA_CC2	捕获/比较 2 DMA 请求
TIM_DMA_CC3	捕获/比较 3 DMA 请求
TIM_DMA_CC4	捕获/比较 4 DMA 请求
TIM_DMA_COM	换向 DMA 请求
TIM_DMA_TRIGGER	触发 DMA 请求

23.2.75 函数 HAL_TIM_GenerateEvent

描述了函数 HAL_TIM_GenerateEvent

表23-173 函数 HAL_TIM_GenerateEvent

函数名	HAL_TIM_GenerateEvent
函数原形	HAL_StatusTypeDef HAL_TIM_GenerateEvent(TIM_HandleTypeDef *htim, uint32_t EventSource)
功能描述	软件产生一个事件
输入参数 1	htim：TIM 句柄
输入参数 2	EventSource：事件源
输出参数	无
返回值	HAL 状态
先决条件	无

EventSource 可选参数：

表23-174 EventSource 可选参数

参数	描述
TIM_EVENTSOURCE_UPDATE	更新事件
TIM_EVENTSOURCE_CC1	捕获/比较 1 事件
TIM_EVENTSOURCE_CC2	捕获/比较 2 事件
TIM_EVENTSOURCE_CC3	捕获/比较 3 事件
TIM_EVENTSOURCE_CC4	捕获/比较 4 事件
TIM_EVENTSOURCE_COM	换向 DMA 事件
TIM_EVENTSOURCE_TRIGGER	触发 DMA 事件
TIM_EVENTSOURCE_BREAK	刹车事件

23.2.76 函数 HAL_TIM_ReadCapturedValue

描述了函数 HAL_TIM_ReadCapturedValue

表23-175 函数 HAL_TIM_ReadCapturedValue

函数名	HAL_TIM_ReadCapturedValue
函数原形	uint32_t HAL_TIM_ReadCapturedValue(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	读取捕获/比较寄存器值

输入参数 1	htim：TIM 句柄
输入参数 2	Channel：通道
输出参数	无
返回值	捕获值
先决条件	无

InputChannel 可选参数:

表23-176 InputChannel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3
TIM_CHANNEL_4	选择通道 4

23.2.77 函数 HAL_TIM_PeriodElapsedCallback

描述了函数 HAL_TIM_PeriodElapsedCallback

表23-177 函数 HAL_TIM_PeriodElapsedCallback

函数名	HAL_TIM_PeriodElapsedCallback
函数原形	void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
功能描述	计数周期回调函数
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.78 函数 HAL_TIM_PeriodElapsedHalfCpltCallback

描述了函数 HAL_TIM_PeriodElapsedHalfCpltCallback

表23-178 函数 HAL_TIM_PeriodElapsedHalfCpltCallback

函数名	HAL_TIM_PeriodElapsedHalfCpltCallback
函数原形	void HAL_TIM_PeriodElapsedHalfCpltCallback(TIM_HandleTypeDef *htim)
功能描述	计数周期半完成回调函数
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.79 函数 HAL_TIM_OC_DelayElapsedCallback

描述了函数 HAL_TIM_OC_DelayElapsedCallback

表23-179 函数 HAL_TIM_OC_DelayElapsedCallback

函数名	HAL_TIM_OC_DelayElapsedCallback
函数原形	void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef *htim)
功能描述	输出比较回调函数
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.80 函数 HAL_TIM_IC_CaptureCallback

描述了函数 HAL_TIM_IC_CaptureCallback

表23-180 函数 HAL_TIM_IC_CaptureCallback

函数名	HAL_TIM_IC_CaptureCallback
函数原形	void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
功能描述	输入捕获回调函数
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.81 函数 HAL_TIM_IC_CaptureHalfCpltCallback

描述了函数 HAL_TIM_IC_CaptureHalfCpltCallback

表23-181 函数 HAL_TIM_IC_CaptureHalfCpltCallback

函数名	HAL_TIM_IC_CaptureHalfCpltCallback
函数原形	void HAL_TIM_IC_CaptureHalfCpltCallback(TIM_HandleTypeDef *htim)
功能描述	输入捕获半完成回调函数
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.82 函数 HAL_TIM_PWM_PulseFinishedCallback

描述了函数 HAL_TIM_PWM_PulseFinishedCallback

表23-182 函数 HAL_TIM_PWM_PulseFinishedCallback

函数名	HAL_TIM_PWM_PulseFinishedCallback
函数原形	void HAL_TIM_PWM_PulseFinishedCallback(TIM_HandleTypeDef *htim)
功能描述	PWM 脉冲回调函数
输入参数	htim：TIM 句柄

输出参数	无
返回值	无
先决条件	无

23.2.83 函数 HAL_TIM_PWM_PulseFinishedHalfCpltCallback

描述了函数 HAL_TIM_PWM_PulseFinishedHalfCpltCallback

表23-183 函数 HAL_TIM_PWM_PulseFinishedHalfCpltCallback

函数名	HAL_TIM_PWM_PulseFinishedHalfCpltCallback
函数原形	void HAL_TIM_PWM_PulseFinishedHalfCpltCallback(TIM_HandleTypeDef *htim)
功能描述	PWM 脉冲半完成回调函数
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.84 函数 HAL_TIM_TriggerCallback

描述了函数 HAL_TIM_TriggerCallback

表23-184 函数 HAL_TIM_TriggerCallback

函数名	HAL_TIM_TriggerCallback
函数原形	void HAL_TIM_TriggerCallback(TIM_HandleTypeDef *htim)
功能描述	霍尔触发检测回调函数
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.85 函数 HAL_TIM_TriggerHalfCpltCallback

描述了函数 HAL_TIM_TriggerHalfCpltCallback

表23-185 函数 HAL_TIM_TriggerHalfCpltCallback

函数名	HAL_TIM_TriggerHalfCpltCallback
函数原形	void HAL_TIM_TriggerHalfCpltCallback(TIM_HandleTypeDef *htim)
功能描述	霍尔触发检测半完成回调函数
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.86 函数 HAL_TIM_ErrorCallback

描述了函数 HAL_TIM_ErrorCallback

表23-186 函数 HAL_TIM_ErrorCallback

函数名	HAL_TIM_ErrorCallback
函数原形	void HAL_TIM_ErrorCallback(TIM_HandleTypeDef *htim)
功能描述	错误回调函数
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

23.2.87 函数 HAL_TIM_Base_GetState

描述了函数 HAL_TIM_Base_GetState

表23-187 函数 HAL_TIM_Base_GetState

函数名	HAL_TIM_Base_GetState
函数原形	HAL_TIM_StateTypeDef HAL_TIM_Base_GetState(TIM_HandleTypeDef *htim)
功能描述	获取时基单元状态
输入参数	htim：TIM 句柄
输出参数	无
返回值	TIM 状态
先决条件	无

23.2.88 函数 HAL_TIM_OC_GetState

描述了函数 HAL_TIM_OC_GetState

表23-188 函数 HAL_TIM_OC_GetState

函数名	HAL_TIM_OC_GetState
函数原形	HAL_TIM_StateTypeDef HAL_TIM_OC_GetState(TIM_HandleTypeDef *htim)
功能描述	获取输出比较模式状态
输入参数	htim：TIM 句柄
输出参数	无
返回值	TIM 状态
先决条件	无

23.2.89 函数 HAL_TIM_PWM_GetState

描述了函数 HAL_TIM_PWM_GetState

表23-189 函数 HAL_TIM_PWM_GetState

函数名	HAL_TIM_PWM_GetState
函数原形	HAL_TIM_StateTypeDef HAL_TIM_PWM_GetState(TIM_HandleTypeDef *htim)

功能描述	获取 PWM 模式状态
输入参数	htim：TIM 句柄
输出参数	无
返回值	TIM 状态
先决条件	无

23.2.90 函数 HAL_TIM_IC_GetState

描述了函数 HAL_TIM_IC_GetState

表23-190 函数 HAL_TIM_IC_GetState

函数名	HAL_TIM_IC_GetState
函数原形	HAL_TIM_StateTypeDef　HAL_TIM_IC_GetState(TIM_HandleTypeDef　*htim)
功能描述	获取输入捕获模式状态
输入参数	htim：TIM 句柄
输出参数	无
返回值	TIM 状态
先决条件	无

23.2.91 函数 HAL_TIM_OnePulse_GetState

描述了函数 HAL_TIM_OnePulse_GetState

表23-191 函数 HAL_TIM_OnePulse_GetState

函数名	HAL_TIM_OnePulse_GetState
函数原形	HAL_TIM_StateTypeDef HAL_TIM_OnePulse_GetState(TIM_HandleTypeDef *htim)
功能描述	获取单脉冲模式状态
输入参数	htim：TIM 句柄
输出参数	无
返回值	TIM 状态
先决条件	无

23.2.92 函数 HAL_TIM_Encoder_GetState

描述了函数 HAL_TIM_Encoder_GetState

表23-192 函数 HAL_TIM_Encoder_GetState

函数名	HAL_TIM_Encoder_GetState
函数原形	HAL_TIM_StateTypeDef HAL_TIM_Encoder_GetState(TIM_HandleTypeDef *htim)
功能描述	获取编码器接口模式状态
输入参数	htim：TIM 句柄
输出参数	无
返回值	TIM 状态

先决条件	无

24 HAL 定时器扩展驱动程序（TIM_Ex）

24.1 TIM_Ex 固件驱动寄存器结构

24.1.1 TIM_HallSensor_InitTypeDef

TIM_HallSensor_InitTypeDef，定义于文件"air001xx_hal_tim_ex.h"如下：

```
typedef struct
{
uint32_t IC1Polarity;
uint32_t IC1Prescaler;
uint32_t IC1Filter;
uint32_t Commutation_Delay;
} TIM_HallSensor_InitTypeDef;
```

字段说明：

表24-1 TIM_HallSensor_InitTypeDef 字段说明

字段	描述
IC1Polarity	输入捕获信号的极性配置
IC1Prescaler	输入的需要被捕获信号的分频系数
IC1Filter	输入的需要被捕获的信号的滤波器系数（0x0~0xF）
Commutation_Delay	加载到捕获/比较寄存器的值（0x0000~0xFFFF）

参数说明：

IC1Polarity 可选参数：

表24-2 IC1Polarity 可选参数

参数	描述
TIM_ICPOLARITY_RISING	不反相，上升沿触发捕获
TIM_ICPOLARITY_FALLING	反相，下降沿触发捕获
TIM_ICPOLARITY_BOTHEDGE	上升沿和下降沿都能触发捕获

IC1Prescaler 可选参数：

表24-3 IC1Prescaler 可选参数

参数	描述
TIM_ICPSC_DIV1	每个事件都触发捕获
TIM_ICPSC_DIV2	每 2 个事件触发一次捕获
TIM_ICPSC_DIV4	每 4 个事件触发一次捕获
TIM_ICPSC_DIV8	每 8 个事件触发一次捕获

24.2 TIM_Ex 固件库函数

表24-4 TIM_Ex 固件库函数说明

函数名	描述
HAL_TIMEx_HallSensor_Init	初始化霍尔传感器接口
HAL_TIMEx_HallSensor_DeInit	将霍尔传感器接口配置设为缺省值
HAL_TIMEx_HallSensor_MspInit	初始化霍尔传感器接口相关 MSP
HAL_TIMEx_HallSensor_MspDeInit	将霍尔传感器接口相关 MSP 设为缺省值
HAL_TIMEx_HallSensor_Start	开启霍尔传感器接口
HAL_TIMEx_HallSensor_Stop	关闭霍尔传感器接口
HAL_TIMEx_HallSensor_Start_IT	开启霍尔传感器接口和捕获中断
HAL_TIMEx_HallSensor_Stop_IT	关闭霍尔传感器接口和捕获中断
HAL_TIMEx_HallSensor_Start_DMA	开启霍尔传感器接口和产生捕获事件时的 DMA 请求
HAL_TIMEx_HallSensor_Stop_DMA	关闭霍尔传感器接口和产生捕获事件时的 DMA 请求
HAL_TIMEx_OCN_Start	开启输出比较模式的互补输出
HAL_TIMEx_OCN_Stop	关闭输出比较模式的互补输出
HAL_TIMEx_OCN_Start_IT	开启输出比较模式的互补输出和比较中断
HAL_TIMEx_OCN_Stop_IT	关闭输出比较模式的互补输出和比较中断
HAL_TIMEx_OCN_Start_DMA	开启输出比较模式的互补输出和产生比较事件时的 DMA 请求
HAL_TIMEx_OCN_Stop_DMA	关闭输出比较模式的互补输出和产生比较事件时的 DMA 请求
HAL_TIMEx_PWMN_Start	开启PWM 模式的互补输出
HAL_TIMEx_PWMN_Stop	关闭PWM 模式的互补输出
HAL_TIMEx_PWMN_Start_IT	开启PWM 模式的互补输出和比较中断
HAL_TIMEx_PWMN_Stop_IT	关闭PWM 模式的互补输出和比较中断
HAL_TIMEx_PWMN_Start_DMA	开启PWM模式的互补输出和产生比较事件时的DMA请求
HAL_TIMEx_PWMN_Stop_DMA	关闭PWM模式的互补输出和产生比较事件时的DMA请求
HAL_TIMEx_OnePulseN_Start	开启单脉冲模式的互补输出
HAL_TIMEx_OnePulseN_Stop	关闭单脉冲模式的互补输出
HAL_TIMEx_OnePulseN_Start_IT	开启单脉冲模式的互补输出和捕获/比较中断

HAL_TIMEx_OnePulseN_Stop_IT	关闭单脉冲模式的互补输出和捕获/比较中断
HAL_TIMEx_ConfigCommutEvent	配置换向事件
HAL_TIMEx_ConfigCommutEvent_IT	配置换向事件并开启 COM 中断
HAL_TIMEx_ConfigCommutEvent_DMA	配置换向事件并开启产生COM 事件时的DMA 请求
HAL_TIMEx_MasterConfigSynchronization	配置TIM 主模式
HAL_TIMEx_ConfigBreakDeadTime	配置死区设置、锁定等级、关闭状态、刹车功能、自动输出
HAL_TIMEx_RemapConfig	配置TIM14 TI1 重映射
HAL_TIMEx_CommutCallback	霍尔换向回调函数
HAL_TIMEx_CommutHalfCpltCallback	霍尔换向半完成回调函数
HAL_TIMEx_BreakCallback	刹车回调函数
HAL_TIMEx_HallSensor_GetState	获取霍尔传感器接口状态

24.2.1 函数 HAL_TIMEx_HallSensor_Init

描述了函数 HAL_TIMEx_HallSensor_Init

表24-5 函数 HAL_TIMEx_HallSensor_Init

函数名	HAL_TIMEx_HallSensor_Init
函数原形	HAL_StatusTypeDef HAL_TIMEx_HallSensor_Init(TIM_HandleTypeDef *htim, TIM_HallSensor_InitTypeDef *sConfig)
功能描述	初始化霍尔传感器接口
输入参数 1	htim：TIM 句柄
输入参数 2	sConfig：霍尔传感器接口初始化配置结构体
输出参数	无
返回值	HAL 状态
先决条件	无

24.2.2 函数 HAL_TIMEx_HallSensor_DeInit

描述了函数 HAL_TIMEx_HallSensor_DeInit

表24-6 函数 HAL_TIMEx_HallSensor_DeInit

函数名	HAL_TIMEx_HallSensor_DeInit
函数原形	HAL_StatusTypeDef HAL_TIMEx_HallSensor_DeInit(TIM_HandleTypeDef *htim)
功能描述	将霍尔传感器接口配置设为缺省值
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态

先决条件	无

24.2.3 函数 HAL_TIMEx_HallSensor_MspInit

描述了函数 HAL_TIMEx_HallSensor_MspInit

表24-7 函数 HAL_TIMEx_HallSensor_MspInit

函数名	HAL_TIMEx_HallSensor_MspInit
函数原形	void HAL_TIMEx_HallSensor_MspInit(TIM_HandleTypeDef *htim)
功能描述	初始化霍尔传感器接口相关 MSP
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

24.2.4 函数 HAL_TIMEx_HallSensor_MspDeInit

描述了函数 HAL_TIMEx_HallSensor_MspDeInit

表24-8 函数 HAL_TIMEx_HallSensor_MspDeInit

函数名	HAL_TIMEx_HallSensor_MspDeInit
函数原形	void HAL_TIMEx_HallSensor_MspDeInit(TIM_HandleTypeDef *htim)
功能描述	将霍尔传感器接口相关 MSP 设为缺省值
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

24.2.5 函数 HAL_TIMEx_HallSensor_Start

描述了函数 HAL_TIMEx_HallSensor_Start

表24-9 函数 HAL_TIMEx_HallSensor_Start

函数名	HAL_TIMEx_HallSensor_Start
函数原形	HAL_StatusTypeDef HAL_TIMEx_HallSensor_Start(TIM_HandleTypeDef *htim)
功能描述	开启霍尔传感器接口
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

24.2.6 函数 HAL_TIMEx_HallSensor_Stop

描述了函数 HAL_TIMEx_HallSensor_Stop

表24-10 函数 HAL_TIMEx_HallSensor_Stop

函数名	HAL_TIMEx_HallSensor_Stop
函数原形	HAL_StatusTypeDef HAL_TIMEx_HallSensor_Stop(TIM_HandleTypeDef *htim)
功能描述	关闭霍尔传感器接口
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

24.2.7 函数 HAL_TIMEx_HallSensor_Start_IT

描述了函数 HAL_TIMEx_HallSensor_Start_IT

表24-11 函数 HAL_TIMEx_HallSensor_Start_IT

函数名	HAL_TIMEx_HallSensor_Start_IT
函数原形	HAL_StatusTypeDef HAL_TIMEx_HallSensor_Start_IT(TIM_HandleTypeDef *htim)
功能描述	开启霍尔传感器接口和捕获中断
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

24.2.8 函数 HAL_TIMEx_HallSensor_Stop_IT

描述了函数 HAL_TIMEx_HallSensor_Stop_IT

表24-12 函数 HAL_TIMEx_HallSensor_Stop_IT

函数名	HAL_TIMEx_HallSensor_Stop_IT
函数原形	HAL_StatusTypeDef HAL_TIMEx_HallSensor_Stop_IT(TIM_HandleTypeDef *htim)
功能描述	关闭霍尔传感器接口和捕获中断
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

24.2.9 函数 HAL_TIMEx_HallSensor_Start_DMA

描述了函数 HAL_TIMEx_HallSensor_Start_DMA

表24-13 函数 HAL_TIMEx_HallSensor_Start_DMA

函数名	HAL_TIMEx_HallSensor_Start_DMA
函数原形	HAL_StatusTypeDef HAL_TIMEx_HallSensor_Start_DMA(TIM_HandleTypeDef *htim, uint32_t *pData, uint16_t Length)
功能描述	开启霍尔传感器接口和产生捕获事件时的 DMA 请求
输入参数 1	htim：TIM 句柄
输入参数 2	pData：数据缓冲区指针
输入参数 3	Length：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

24.2.10 函数 HAL_TIMEx_HallSensor_Stop_DMA

描述了函数 HAL_TIMEx_HallSensor_Stop_DMA

表24-14 函数 HAL_TIMEx_HallSensor_Stop_DMA

函数名	HAL_TIMEx_HallSensor_Stop_DMA
函数原形	HAL_StatusTypeDef HAL_TIMEx_HallSensor_Stop_DMA(TIM_HandleTypeDef *htim)
功能描述	关闭霍尔传感器接口和产生捕获事件时的 DMA 请求
输入参数	htim：TIM 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

24.2.11 函数 HAL_TIMEx_OCN_Start

描述了函数 HAL_TIMEx_OCN_Start

表24-15 函数 HAL_TIMEx_OCN_Start

函数名	HAL_TIMEx_OCN_Start
函数原形	HAL_StatusTypeDef HAL_TIMEx_OCN_Start(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	开启输出比较模式的互补输出
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表24-16 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3

24.2.12 函数 HAL_TIMEx_OCN_Stop

描述了函数 HAL_TIMEx_OCN_Stop

表24-17 函数 HAL_TIMEx_OCN_Stop

函数名	HAL_TIMEx_OCN_Stop
函数原形	HAL_StatusTypeDef HAL_TIMEx_OCN_Stop(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	关闭输出比较模式的互补输出
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数:

表24-18 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3

24.2.13 函数 HAL_TIMEx_OCN_Start_IT

描述了函数 HAL_TIMEx_OCN_Start_IT

表24-19 函数 HAL_TIMEx_OCN_Start_IT

函数名	HAL_TIMEx_OCN_Start_IT
函数原形	HAL_StatusTypeDef HAL_TIMEx_OCN_Start_IT(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	开启输出比较模式的互补输出和比较中断
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表24-20 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3

24.2.14 函数 HAL_TIMEx_OCN_Stop_IT

描述了函数 HAL_TIMEx_OCN_Stop_IT

表24-21 函数 HAL_TIMEx_OCN_Stop_IT

函数名	HAL_TIMEx_OCN_Stop_IT
函数原形	HAL_StatusTypeDef HAL_TIMEx_OCN_Stop_IT(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	关闭输出比较模式的互补输出和比较中断
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表24-22 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3

24.2.15 函数 HAL_TIMEx_OCN_Start_DMA

描述了函数 HAL_TIMEx_OCN_Start_DMA

表24-23 函数 HAL_TIMEx_OCN_Start_DMA

函数名	HAL_TIMEx_OCN_Start_DMA
函数原形	HAL_StatusTypeDef HAL_TIMEx_OCN_Start_DMA(TIM_HandleTypeDef *htim, uint32_t Channel, uint32_t *pData, uint16_t Length)
功能描述	开启输出比较模式的互补输出和产生比较事件时的 DMA 请求
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输入参数 3	pData：数据缓冲区指针
输入参数 4	Length：数据长度

输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表24-24 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3

24.2.16 函数 HAL_TIMEx_OCN_Stop_DMA

描述了函数 HAL_TIMEx_OCN_Stop_DMA

表24-25 函数 HAL_TIMEx_OCN_Stop_DMA

函数名	HAL_TIMEx_OCN_Stop_DMA
函数原形	HAL_StatusTypeDef HAL_TIMEx_OCN_Stop_DMA(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	关闭输出比较模式的互补输出和产生比较事件时的 DMA 请求
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表24-26 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3

24.2.17 函数 HAL_TIMEx_PWMN_Start

描述了函数 HAL_TIMEx_PWMN_Start

表24-27 函数 HAL_TIMEx_PWMN_Start

函数名	HAL_TIMEx_PWMN_Start
函数原形	HAL_StatusTypeDef HAL_TIMEx_PWMN_Start(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	开启 PWM 模式的互补输出

输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表24-28 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3

24.2.18 函数 HAL_TIMEx_PWMN_Stop

描述了函数 HAL_TIMEx_PWMN_Stop

表24-29 函数 HAL_TIMEx_PWMN_Stop

函数名	HAL_TIMEx_PWMN_Stop
函数原形	HAL_StatusTypeDef HAL_TIMEx_PWMN_Stop(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	关闭 PWM 模式的互补输出
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表24-30 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3

24.2.19 函数 HAL_TIMEx_PWMN_Start_IT

描述了函数 HAL_TIMEx_PWMN_Start_IT

表24-31 函数 HAL_TIMEx_PWMN_Start_IT

函数名	HAL_TIMEx_PWMN_Start_IT

函数原形	HAL_StatusTypeDef HAL_TIMEx_PWMN_Start_IT(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	开启 PWM 模式的互补输出和捕获/比较中断
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表24-32 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3

24.2.20 函数 HAL_TIMEx_PWMN_Stop_IT

描述了函数 HAL_TIMEx_PWMN_Stop_IT

表24-33 函数 HAL_TIMEx_PWMN_Stop_IT

函数名	HAL_TIMEx_PWMN_Stop_IT
函数原形	HAL_StatusTypeDef HAL_TIMEx_PWMN_Stop_IT(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	关闭 PWM 模式的互补输出和捕获/比较中断
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表24-34 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3

24.2.21 函数 HAL_TIMEx_PWMN_Start_DMA

描述了函数 HAL_TIMEx_PWMN_Start_DMA

表24-35 函数 HAL_TIMEx_PWMN_Start_DMA

函数名	HAL_TIMEx_PWMN_Start_DMA
函数原形	HAL_StatusTypeDef HAL_TIMEx_PWMN_Start_DMA(TIM_HandleTypeDef *htim, uint32_t Channel, uint32_t *pData, uint16_t Length)
功能描述	开启 PWM 模式的互补输出和产生捕获/比较事件时的 DMA 请求
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输入参数 3	pData：数据缓冲区指针
输入参数 4	Length：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表24-36 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2
TIM_CHANNEL_3	选择通道 3

24.2.22 函数 HAL_TIMEx_PWMN_Stop_DMA

描述了函数 HAL_TIMEx_PWMN_Stop_DMA

表24-37 函数 HAL_TIMEx_PWMN_Stop_DMA

函数名	HAL_TIMEx_PWMN_Stop_DMA
函数原形	HAL_StatusTypeDef HAL_TIMEx_PWMN_Stop_DMA(TIM_HandleTypeDef *htim, uint32_t Channel)
功能描述	关闭 PWM 模式的互补输出和产生捕获/比较事件时的 DMA 请求
输入参数 1	htim：TIM 句柄
输入参数 2	Channel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

Channel 可选参数：

表24-38 Channel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2

TIM_CHANNEL_3	选择通道 3

24.2.23 函数 HAL_TIMEx_OnePulseN_Start

描述了函数 HAL_TIMEx_OnePulseN_Start

表24-39 函数 HAL_TIMEx_OnePulseN_Start

函数名	HAL_TIMEx_OnePulseN_Start
函数原形	HAL_StatusTypeDef HAL_TIMEx_OnePulseN_Start(TIM_HandleTypeDef *htim, uint32_t OutputChannel)
功能描述	开启单脉冲模式的互补输出
输入参数 1	htim：TIM 句柄
输入参数 2	OutputChannel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

OutputChannel 可选参数：

表24-40 OutputChannel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2

24.2.24 函数 HAL_TIMEx_OnePulseN_Stop

描述了函数 HAL_TIMEx_OnePulseN_Stop

表24-41 函数 HAL_TIMEx_OnePulseN_Stop

函数名	HAL_TIMEx_OnePulseN_Stop
函数原形	HAL_StatusTypeDef HAL_TIMEx_OnePulseN_Stop(TIM_HandleTypeDef *htim, uint32_t OutputChannel)
功能描述	关闭单脉冲模式的互补输出
输入参数 1	htim：TIM 句柄
输入参数 2	OutputChannel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

OutputChannel 可选参数：

表24-42 OutputChannel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1

TIM_CHANNEL_2	选择通道 2

24.2.25 函数 HAL_TIMEx_OnePulseN_Start_IT

描述了函数 HAL_TIMEx_OnePulseN_Start_IT

表24-43 函数 HAL_TIMEx_OnePulseN_Start_IT

函数名	HAL_TIMEx_OnePulseN_Start_IT
函数原形	HAL_StatusTypeDef HAL_TIMEx_OnePulseN_Start_IT(TIM_HandleTypeDef *htim, uint32_t OutputChannel)
功能描述	开启单脉冲模式的互补输出和捕获/比较中断
输入参数 1	htim：TIM 句柄
输入参数 2	OutputChannel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

OutputChannel 可选参数：

表24-44 OutputChannel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1
TIM_CHANNEL_2	选择通道 2

24.2.26 函数 HAL_TIMEx_OnePulseN_Stop_IT

描述了函数 HAL_TIMEx_OnePulseN_Stop_IT

表24-45 函数 HAL_TIMEx_OnePulseN_Stop_IT

函数名	HAL_TIMEx_OnePulseN_Stop_IT
函数原形	HAL_StatusTypeDef HAL_TIMEx_OnePulseN_Stop_IT(TIM_HandleTypeDef *htim, uint32_t OutputChannel)
功能描述	关闭单脉冲模式的互补输出和捕获/比较中断
输入参数 1	htim：TIM 句柄
输入参数 2	OutputChannel：输出通道
输出参数	无
返回值	HAL 状态
先决条件	无

OutputChannel 可选参数：

表24-46 OutputChannel 可选参数

参数	描述
TIM_CHANNEL_1	选择通道 1

TIM_CHANNEL_2	选择通道 2

24.2.27 函数 HAL_TIMEx_ConfigCommutEvent

描述了函数 HAL_TIMEx_ConfigCommutEvent

表24-47 函数 HAL_TIMEx_ConfigCommutEvent

函数名	HAL_TIMEx_ConfigCommutEvent
函数原形	HAL_StatusTypeDef HAL_TIMEx_ConfigCommutEvent(TIM_HandleTypeDef *htim, uint32_t InputTrigger,uint32_t CommutationSource)
功能描述	配置换向事件
输入参数 1	htim：TIM 句柄
输入参数 2	InputTrigger：输入触发源
输入参数 3	CommutationSource：换向源
输出参数	无
返回值	HAL 状态
先决条件	无

InputTrigger 可选参数：

表24-48 InputTrigger 可选参数

参数	描述
TIM_TS_ITR0	内部触发 0
TIM_TS_ITR1	内部触发 1
TIM_TS_ITR2	内部触发 2
TIM_TS_ITR3	内部触发 3
TIM_TS_NONE	无触发源

CommutationSource 可选参数：

表24-49 CommutationSource 可选参数

参数	描述
TIM_COMMUTATION_TRGI	TIM 输入接口 TRGI 触发
TIM_COMMUTATION_SOFTWARE	软件设置 COMG 位触发

24.2.28 函数 HAL_TIMEx_ConfigCommutEvent_IT

描述了函数 HAL_TIMEx_ConfigCommutEvent_IT

表24-50 函数 HAL_TIMEx_ConfigCommutEvent_IT

函数名	HAL_TIMEx_ConfigCommutEvent_IT
函数原形	HAL_StatusTypeDef HAL_TIMEx_ConfigCommutEvent_IT(TIM_HandleTypeDef *htim, uint32_t InputTrigger,uint32_t CommutationSource)
功能描述	配置换向事件并开启 COM 中断

输入参数 1	htim：TIM 句柄
输入参数 2	InputTrigger：输入触发源
输入参数 3	CommutationSource：换向源
输出参数	无
返回值	HAL 状态
先决条件	无

InputTrigger 可选参数：

表24-51 InputTrigger 可选参数

参数	描述
TIM_TS_ITR0	（保留）
TIM_TS_ITR1	（保留）
TIM_TS_ITR2	TIM3 触发
TIM_TS_ITR3	TIM17 触发
TIM_TS_NONE	无触发源

CommutationSource 可选参数：

表24-52 CommutationSource 可选参数

参数	描述
TIM_COMMUTATION_TRGI	TIM 输入接口 TRGI 触发
TIM_COMMUTATION_SOFTWARE	软件设置 COMG 位触发

24.2.29 函数 HAL_TIMEx_ConfigCommutEvent_DMA

描述了函数 HAL_TIMEx_ConfigCommutEvent_DMA

表24-53 函数 HAL_TIMEx_ConfigCommutEvent_DMA

函数名	HAL_TIMEx_ConfigCommutEvent_DMA
函数原形	HAL_StatusTypeDef HAL_TIMEx_ConfigCommutEvent_DMA(TIM_HandleTypeDef *htim, uint32_t InputTrigger,uint32_t CommutationSource)
功能描述	配置换向事件并开启产生 COM 事件时的 DMA 请求
输入参数 1	htim：TIM 句柄
输入参数 2	InputTrigger：输入触发源
输入参数 3	CommutationSource：换向源
输出参数	无
返回值	HAL 状态
先决条件	无

InputTrigger 可选参数：

表24-54 InputTrigger 可选参数

参数	描述
TIM_TS_ITR0	（保留）
TIM_TS_ITR1	（保留）
TIM_TS_ITR2	TIM3 触发
TIM_TS_ITR3	TIM17 触发
TIM_TS_NONE	无触发源

CommutationSource 可选参数：

表24-55 CommutationSource 可选参数

参数	描述
TIM_COMMUTATION_TRGI	TIM 输入接口 TRGI 触发
TIM_COMMUTATION_SOFTWARE	软件设置 COMG 位触发

24.2.30 函数 HAL_TIMEx_MasterConfigSynchronization

描述了函数 HAL_TIMEx_MasterConfigSynchronization

表24-56 函数 HAL_TIMEx_MasterConfigSynchronization

函数名	HAL_TIMEx_MasterConfigSynchronization
函数原形	HAL_StatusTypeDef HAL_TIMEx_MasterConfigSynchronization(TIM_HandleTypeDef *htim,TIM_MasterConfigTypeDef *sMasterConfig)
功能描述	配置 TIM 主模式
输入参数 1	htim：TIM 句柄
输入参数 2	sMasterConfig：主模式初始化配置结构体
输出参数	无
返回值	HAL 状态
先决条件	无

24.2.31 函数 HAL_TIMEx_ConfigBreakDeadTime

描述了函数 HAL_TIMEx_ConfigBreakDeadTime

表24-57 函数 HAL_TIMEx_ConfigBreakDeadTime

函数名	HAL_TIMEx_ConfigBreakDeadTime
函数原形	HAL_StatusTypeDef HAL_TIMEx_ConfigBreakDeadTime(TIM_HandleTypeDef *htim,TIM_BreakDeadTimeConfigTypeDef *sBreakDeadTimeConfig)
功能描述	配置 BDTR（死区设置、锁定等级、关闭状态、刹车功能、自动输出）
输入参数 1	htim：TIM 句柄
输入参数 2	sBreakDeadTimeConfig：BDTR 初始化配置结构体
输出参数	无
返回值	HAL 状态

先决条件	无

24.2.32 函数 HAL_TIMEx_RemapConfig

描述了函数 HAL_TIMEx_RemapConfig

表24-58 函数 HAL_TIMEx_RemapConfig

函数名	HAL_TIMEx_RemapConfig
函数原形	HAL_StatusTypeDef HAL_TIMEx_RemapConfig(TIM_HandleTypeDef *htim, uint32_t Remap)
功能描述	配置 TIM14 TI1 重映射
输入参数 1	htim：TIM 句柄
输入参数 2	Remap：TI 重映射源
输出参数	无
返回值	HAL 状态
先决条件	无

CommutationSource 可选参数：

表24-59 CommutationSource 可选参数

参数	描述
TIM_TIM14_GPIO	TIM14 TI1 连接到 GPIO
TIM_TIM14_RTC	TIM14 TI1 连接到 RTC
TIM_TIM14_HSE	TIM14 TI1 连接到 HSE/32
TIM_TIM14_MCO	TIM14 TI1 连接到 MCO

24.2.33 函数 HAL_TIMEx_CommutCallback

描述了函数 HAL_TIMEx_CommutCallback

表24-60 函数 HAL_TIMEx_CommutCallback

函数名	HAL_TIMEx_CommutCallback
函数原形	void HAL_TIMEx_CommutCallback(TIM_HandleTypeDef *htim)
功能描述	霍尔传感器接口换向回调函数
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

24.2.34 函数 HAL_TIMEx_CommutHalfCpltCallback

描述了函数 HAL_TIMEx_CommutHalfCpltCallback

表24-61 函数 HAL_TIMEx_CommutHalfCpltCallback

函数名	HAL_TIMEx_CommutHalfCpltCallback

函数原形	void HAL_TIMEx_CommutHalfCpltCallback(TIM_HandleTypeDef *htim)
功能描述	霍尔传感器接口换向半完成回调函数
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

24.2.35 函数 HAL_TIMEx_BreakCallback

描述了函数 HAL_TIMEx_BreakCallback

表24-62 函数 HAL_TIMEx_BreakCallback

函数名	HAL_TIMEx_BreakCallback
函数原形	void HAL_TIMEx_BreakCallback(TIM_HandleTypeDef *htim)
功能描述	刹车回调函数
输入参数	htim：TIM 句柄
输出参数	无
返回值	无
先决条件	无

24.2.36 函数 HAL_TIM_StateTypeDef

描述了函数 HAL_TIM_StateTypeDef

表24-63 函数 HAL_TIM_StateTypeDef

函数名	HAL_TIM_StateTypeDef
函数原形	HAL_TIM_StateTypeDef HAL_TIMEx_HallSensor_GetState(TIM_HandleTypeDef *htim)
功能描述	获取霍尔传感器接口状态
输入参数	htim：TIM 句柄
输出参数	无
返回值	TIM 状态
先决条件	无

25 HAL 异步收发器通用驱动程序（UART）

25.1 UART 固件驱动寄存器结构

25.1.1 UART_InitTypeDef

UART_InitTypeDef，定义于文件”air001xx_hal_uart.h”如下：

```
typedef struct
{
uint32_t BaudRate;
uint32_t WordLength;
uint32_t StopBits;
uint32_t Parity;
uint32_t Mode;
uint32_t HwFlowCtl;
uint32_t OverSampling;
} UART_InitTypeDef;
```

字段说明：

表25-1 UART_InitTypeDef 字段说明

字段	描述
BaudRate	波特率
WordLength	数据长度
StopBits	停止位数
Parity	校验方式
Mode	工作模式
HwFlowCtl	硬件流控制
OverSampling	过采样率

参数说明：

WordLength 可选参数：

表25-2 WordLength 可选参数

参数	描述
UART_WORDLENGTH_8B	8 个数据位
UART_WORDLENGTH_9B	9 个数据位

StopBits 可选参数：

表25-3 StopBits 可选参数

参数	描述
UART_STOPBITS_1	1 个停止位
UART_STOPBITS_2	2 个停止位

Parity 可选参数：

表25-4 Parity 可选参数

参数	描述
UART_PARITY_NONE	无校验
UART_PARITY_EVEN	偶校验
UART_PARITY_ODD	奇校验

Mode 可选参数：

表25-5 Mode 可选参数

参数	描述
UART_MODE_RX	使能接收模式
UART_MODE_TX	使能发送模式
UART_MODE_TX_RX	使能接收和发送

HwFlowCtl 可选参数：

表25-6 HwFlowCtl 可选参数

参数	描述
UART_HWCONTROL_NONE	无硬件流控制
UART_HWCONTROL_RTS	使能 RTS 控制
UART_HWCONTROL_CTS	使能 CTS 控制
UART_HWCONTROL_RTS_CTS	使能 RTS 和 CTS 控制

OverSampling 可选参数：

表25-7 OverSampling 可选参数

参数	描述
UART_OVERSAMPLING_16	16 倍过采样倍率
UART_OVERSAMPLING_8	8 倍过采样倍率

25.1.2 UART_AdvFeatureInitTypeDef

UART_AdvFeatureInitTypeDef，定义于文件”air001xx_hal_uart.h”如下：

```
typedef struct
{
uint32_t AdvFeatureInit;
uint32_t AutoBaudRateEnable;
```

```
uint32_t AutoBaudRateMode;
} UART_AdvFeatureInitTypeDef;
```

字段说明：

表25-8 UART_AdvFeatureInitTypeDef 字段说明

字段	描述
AdvFeatureInit	UART 高级功能初始化类型
AutoBaudRateEnable	自动波特率使能控制
AutoBaudRateMode	自动波特率模式

参数说明：

AdvFeatureInit 可选参数：

表25-9 AdvFeatureInit 可选参数

参数	描述
UART_ADVFEATURE_NO_INIT	无高级功能
UART_ADVFEATURE_AUTOBAUDRATE_INIT	初始化自动波特率功能

AutoBaudRateEnable 可选参数：

表25-10 AutoBaudRateEnable 可选参数

参数	描述
UART_ADVFEATURE_AUTOBAUDRATE_DISABLE	关闭自动波特率
UART_ADVFEATURE_AUTOBAUDRATE_ENABLE	开启自动波特率

AutoBaudRateMode 可选参数：

表25-11 AutoBaudRateMode 可选参数

参数	描述
UART_ADVFEATURE_AUTOBAUDRATE_ONSTARTBIT	测量起始位的持续时间
UART_ADVFEATURE_AUTOBAUDRATE_ONFALLINGEDGE	测量起始位和第一个数据的持续时间

25.1.3 UART_HandleTypeDef

UART_HandleTypeDef，定义于文件”air001xx_hal_uart.h”如下：

```
typedef struct __UART_HandleTypeDef
{
USART_TypeDef *Instance;
UART_InitTypeDef Init;
UART_AdvFeatureInitTypeDef AdvancedInit;
uint8_t *pTxBuffPtr;
uint16_t TxXferSize;
```

```
_IO uint16_t TxXferCount;
uint8_t *pRxBuffPtr;
uint16_t RxXferSize;
_IO uint16_t RxXferCount;
DMA_HandleTypeDef *hdmatx;
DMA_HandleTypeDef *hdmarx;
HAL_LockTypeDef Lock;
__IO HAL_UART_StateTypeDef gState;
__IO HAL_UART_StateTypeDef RxState;
__IO uint32_t ErrorCode;
} UART_HandleTypeDef;
```

字段说明：

表25-12 UART_HandleTypeDef 字段说明

字段	描述
Instance	UART 基地址
Init	初始化参数结构体
AdvancedInit	UART 高级功能初始化结构体
pTxBuffPtr	发送数据缓冲区指针
TxXferSize	发送数据总量
TxXferCount	未发送的数据数量
pRxBuffPtr	接收数据缓冲区指针
RxXferSize	接收数据总量
RxXferCount	未接收的数据数量
hdmatx	DMA 发送句柄指针
hdmarx	DMA 接收句柄指针
Lock	HAL 锁
gState	UART 全局状态（与 Tx 状态也有关）
RxState	UART Rx 状态
ErrorCode	错误代码

25.2 UART 固件库函数

表25-13 UART 固件库函数说明

函数名	描述
HAL_UART_Init	初始化 UART

HAL_HalfDuplex_Init	初始化 UART 半双工模式
HAL_MultiProcessor_Init	初始化 UART 多机通信模式
HAL_UART_DeInit	将 UART 相关配置设为缺省值
HAL_UART_MspInit	初始化 UART 相关的 MSP
HAL_UART_MspDeInit	将 UART 相关的 MSP 设为缺省值
HAL_UART_Transmit	使用轮询的方式发送数据
HAL_UART_Receive	使用轮询的方式接收数据
HAL_UART_Transmit_IT	使用中断的方式发送数据
HAL_UART_Receive_IT	使用中断的方式接收数据
HAL_UART_Transmit_DMA	使用DMA 的方式发送数据
HAL_UART_Receive_DMA	使用DMA 的方式接收数据
HAL_UART_DMAPause	暂停DMA 传输
HAL_UART_DMAResume	恢复DMA 传输
HAL_UART_DMAStop	停止DMA 传输
HAL_UART_Abort	中止UART 传输
HAL_UART_AbortTransmit	中止UART 数据发送
HAL_UART_AbortReceive	中止UART 数据接收
HAL_UART_Abort_IT	中止UART 传输并关闭中断
HAL_UART_AbortTransmit_IT	中止UART 数据发送并关闭发送相关中断
HAL_UART_AbortReceive_IT	中止UART 数据接收并关闭接收相关中断
HAL_UART_IRQHandler	中断请求处理
HAL_UART_TxCpltCallback	发送完成回调函数
HAL_UART_TxHalfCpltCallback	发送半完成回调函数
HAL_UART_RxCpltCallback	接收完成回调函数
HAL_UART_RxHalfCpltCallback	接收半完成回调函数
HAL_UART_ErrorCallback	错误回调函数
HAL_UART_AbortCpltCallback	中止完成回调函数
HAL_UART_AbortTransmitCpltCallback	中止发送完成回调函数
HAL_UART_AbortReceiveCpltCallback	中止接收完成回调函数
HAL_UART_SendBreak	发送一个空白帧
HAL_MultiProcessor_EnterMuteMode	多机通信模式下进入静默模式

HAL_MultiProcessor_ExitMuteMode	多机通信模式下退出静默模式
HAL_HalfDuplex_EnableTransmitter	半双工模式下开启发送功能
HAL_HalfDuplex_EnableReceiver	半双工模式下开启接收功能
HAL_UART_GetState	获取 UART 状态
HAL_UART_GetError	获取错误代码

25.2.1 函数 HAL_UART_Init

描述了函数 HAL_UART_Init

表25-14 函数 HAL_UART_Init

函数名	HAL_UART_Init
函数原形	HAL_StatusTypeDef HAL_UART_Init(UART_HandleTypeDef *huart)
功能描述	初始化 UART
输入参数	huart：UART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.2 函数 HAL_HalfDuplex_Init

描述了函数 HAL_HalfDuplex_Init

表25-15 函数 HAL_HalfDuplex_Init

函数名	HAL_HalfDuplex_Init
函数原形	HAL_StatusTypeDef HAL_HalfDuplex_Init(UART_HandleTypeDef *huart)
功能描述	初始化 UART 半双工模式
输入参数	huart：UART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.3 函数 HAL_MultiProcessor_Init

描述了函数 HAL_MultiProcessor_Init

表25-16 函数 HAL_MultiProcessor_Init

函数名	HAL_MultiProcessor_Init
函数原形	HAL_StatusTypeDef HAL_MultiProcessor_Init(UART_HandleTypeDef *huart, uint8_t Address, uint32_t WakeUpMethod)
功能描述	初始化 UART 多机通信模式
输入参数	huart：UART 句柄

输出参数	无
返回值	HAL 状态
先决条件	无

25.2.4 函数 HAL_UART_DeInit

描述了函数 HAL_UART_DeInit

表25-17　函数 HAL_UART_DeInit

函数名	HAL_UART_DeInit
函数原形	HAL_StatusTypeDef　HAL_UART_DeInit(UART_HandleTypeDef　*huart)
功能描述	将 UART 配置设为缺省值
输入参数	huart：UART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.5 函数 HAL_UART_MspInit

描述了函数 HAL_UART_MspInit

表25-18　函数 HAL_UART_MspInit

函数名	HAL_UART_MspInit
函数原形	void HAL_UART_MspInit(UART_HandleTypeDef *huart)
功能描述	初始化 UART 相关的 MSP
输入参数	huart：UART 句柄
输出参数	无
返回值	无
先决条件	无

25.2.6 函数 HAL_UART_MspDeInit

描述了函数 HAL_UART_MspDeInit

表25-19　函数 HAL_UART_MspDeInit

函数名	HAL_UART_MspDeInit
函数原形	void　HAL_UART_MspDeInit(UART_HandleTypeDef　*huart)
功能描述	将 UART 相关的 MSP 设为缺省值
输入参数	huart：UART 句柄
输出参数	无
返回值	无
先决条件	无

25.2.7 函数 HAL_UART_Transmit

描述了函数 HAL_UART_Transmit

表25-20 函数 HAL_UART_Transmit

函数名	HAL_UART_Transmit
函数原形	HAL_StatusTypeDef HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)
功能描述	使用轮询的方式发送数据
输入参数 1	huart：UART 句柄
输入参数 2	pData：发送数据缓冲区指针
输入参数 3	Size：数据长度
输入参数 4	Timeout：超时时间
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.8 函数 HAL_UART_Receive

描述了函数 HAL_UART_Receive

表25-21 函数 HAL_UART_Receive

函数名	HAL_UART_Receive
函数原形	HAL_StatusTypeDef HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)
功能描述	使用轮询的方式接收数据
输入参数 1	huart：UART 句柄
输入参数 2	pData：接收数据缓冲区指针
输入参数 3	Size：数据长度
输入参数 4	Timeout：超时时间
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.9 函数 HAL_UART_Transmit_IT

描述了函数 HAL_UART_Transmit_IT

表25-22 函数 HAL_UART_Transmit_IT

函数名	HAL_UART_Transmit_IT
函数原形	HAL_StatusTypeDef HAL_UART_Transmit_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)
功能描述	使用中断的方式发送数据
输入参数 1	huart：UART 句柄

输入参数 2	pData：发送数据缓冲区指针
输入参数 3	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.10 函数 HAL_UART_Receive_IT

描述了函数 HAL_UART_Receive_IT

表25-23 函数 HAL_UART_Receive_IT

函数名	HAL_UART_Receive_IT
函数原形	HAL_StatusTypeDef HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)
功能描述	使用中断的方式接收数据
输入参数 1	huart：UART 句柄
输入参数 2	pData：接收数据缓冲区指针
输入参数 3	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.11 函数 HAL_UART_Transmit_DMA

描述了函数 HAL_UART_Transmit_DMA

表25-24 函数 HAL_UART_Transmit_DMA

函数名	HAL_UART_Transmit_DMA
函数原形	HAL_StatusTypeDef HAL_UART_Transmit_DMA(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)
功能描述	使用 DMA 的方式发送数据
输入参数 1	huart：UART 句柄
输入参数 2	pData：发送数据缓冲区指针
输入参数 3	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.12 函数 HAL_UART_Receive_DMA

描述了函数 HAL_UART_Receive_DMA

表25-25 函数 HAL_UART_Receive_DMA

函数名	HAL_UART_Receive_DMA

函数原形	HAL_StatusTypeDef HAL_UART_Receive_DMA(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)
功能描述	使用 DMA 的方式接收数据
输入参数 1	huart：UART 句柄
输入参数 2	pData：接收数据缓冲区指针
输入参数 3	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.13 函数 HAL_UART_DMAPause

描述了函数 HAL_UART_DMAPause

表25-26 函数 HAL_UART_DMAPause

函数名	HAL_UART_DMAPause
函数原形	HAL_StatusTypeDef HAL_UART_DMAPause(UART_HandleTypeDef *huart)
功能描述	暂停正在进行的 DMA 传输
输入参数	huart：UART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.14 函数 HAL_UART_DMAResume

描述了函数 HAL_UART_DMAResume

表25-27 函数 HAL_UART_DMAResume

函数名	HAL_UART_DMAResume
函数原形	HAL_StatusTypeDef HAL_UART_DMAResume(UART_HandleTypeDef *huart)
功能描述	恢复暂停的 DMA 传输
输入参数	huart：UART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.15 函数 HAL_UART_DMAStop

描述了函数 HAL_UART_DMAStop

表25-28 函数 HAL_UART_DMAStop

函数名	HAL_UART_DMAStop
函数原形	HAL_StatusTypeDef HAL_UART_DMAStop(UART_HandleTypeDef *huart)

功能描述	停止正在进行的 DMA 传输
输入参数	huart：UART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.16 函数 HAL_UART_Abort

描述了函数 HAL_UART_Abort

表25-29 函数 HAL_UART_Abort

函数名	HAL_UART_Abort
函数原形	HAL_StatusTypeDef HAL_UART_Abort(UART_HandleTypeDef *huart)
功能描述	中止 UART 传输
输入参数	huart：UART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.17 函数 HAL_UART_AbortTransmit

描述了函数 HAL_UART_AbortTransmit

表25-30 函数 HAL_UART_AbortTransmit

函数名	HAL_UART_AbortTransmit
函数原形	HAL_StatusTypeDef HAL_UART_AbortTransmit(UART_HandleTypeDef *huart)
功能描述	中止 UART 数据发送
输入参数	huart：UART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.18 函数 HAL_UART_AbortReceive

描述了函数 HAL_UART_AbortReceive

表25-31 函数 HAL_UART_AbortReceive

函数名	HAL_UART_AbortReceive
函数原形	HAL_StatusTypeDef HAL_UART_AbortReceive(UART_HandleTypeDef *huart)
功能描述	中止 UART 数据接收
输入参数	huart：UART 句柄
输出参数	无

返回值	HAL 状态
先决条件	无

25.2.19 函数 HAL_UART_Abort_IT

描述了函数 HAL_UART_Abort_IT

表25-32 函数 HAL_UART_Abort_IT

函数名	HAL_UART_Abort_IT
函数原形	HAL_StatusTypeDef HAL_UART_Abort_IT(UART_HandleTypeDef *huart)
功能描述	中止 UART 传输并关闭中断
输入参数	huart：UART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.20 函数 HAL_UART_AbortTransmit_IT

描述了函数 HAL_UART_AbortTransmit_IT

表25-33 函数 HAL_UART_AbortTransmit_IT

函数名	HAL_UART_AbortTransmit_IT
函数原形	HAL_StatusTypeDef HAL_UART_AbortTransmit_IT(UART_HandleTypeDef *huart)
功能描述	中止 UART 数据发送并关闭发送相关中断
输入参数	huart：UART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.21 函数 HAL_UART_AbortReceive_IT

描述了函数 HAL_UART_AbortReceive_IT

表25-34 函数 HAL_UART_AbortReceive_IT

函数名	HAL_UART_AbortReceive_IT
函数原形	HAL_StatusTypeDef HAL_UART_AbortReceive_IT(UART_HandleTypeDef *huart)
功能描述	中止 UART 数据接收并关闭接收相关中断
输入参数	huart：UART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.22 函数 HAL_UART_IRQHandler

描述了函数 HAL_UART_IRQHandler

表25-35　函数 HAL_UART_IRQHandler

函数名	HAL_UART_IRQHandler
函数原形	void　HAL_UART_IRQHandler(UART_HandleTypeDef　*huart)
功能描述	中断请求处理
输入参数	huart：UART 句柄
输出参数	无
返回值	无
先决条件	无

25.2.23 函数 HAL_UART_TxCpltCallback

描述了函数 HAL_UART_TxCpltCallback

表25-36　函数 HAL_UART_TxCpltCallback

函数名	HAL_UART_TxCpltCallback
函数原形	void　HAL_UART_TxCpltCallback(UART_HandleTypeDef　*huart)
功能描述	发送完成回调函数
输入参数	huart：UART 句柄
输出参数	无
返回值	无
先决条件	无

25.2.24 函数 HAL_UART_TxHalfCpltCallback

描述了函数 HAL_UART_TxHalfCpltCallback

表25-37　函数 HAL_UART_TxHalfCpltCallback

函数名	HAL_UART_TxHalfCpltCallback
函数原形	void　HAL_UART_TxHalfCpltCallback(UART_HandleTypeDef　*huart)
功能描述	发送半完成回调函数
输入参数	huart：UART 句柄
输出参数	无
返回值	无
先决条件	无

25.2.25 函数 HAL_UART_RxCpltCallback

描述了函数 HAL_UART_RxCpltCallback

表25-38 函数 HAL_UART_RxCpltCallback

函数名	HAL_UART_RxCpltCallback
函数原形	void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
功能描述	接收完成回调函数
输入参数	huart：UART 句柄
输出参数	无
返回值	无
先决条件	无

25.2.26 函数 HAL_UART_RxHalfCpltCallback

描述了函数 HAL_UART_RxHalfCpltCallback

表25-39 函数 HAL_UART_RxHalfCpltCallback

函数名	HAL_UART_RxHalfCpltCallback
函数原形	void HAL_UART_RxHalfCpltCallback(UART_HandleTypeDef *huart)
功能描述	接收半完成回调函数
输入参数	huart：UART 句柄
输出参数	无
返回值	无
先决条件	无

25.2.27 函数 HAL_UART_ErrorCallback

描述了函数 HAL_UART_ErrorCallback

表25-40 函数 HAL_UART_ErrorCallback

函数名	HAL_UART_ErrorCallback
函数原形	void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart)
功能描述	错误回调函数
输入参数	huart：UART 句柄
输出参数	无
返回值	无
先决条件	无

25.2.28 函数 HAL_UART_AbortCpltCallback

描述了函数 HAL_UART_AbortCpltCallback

表25-41 函数 HAL_UART_AbortCpltCallback

函数名	HAL_UART_AbortCpltCallback
函数原形	void HAL_UART_AbortCpltCallback(UART_HandleTypeDef *huart)

功能描述	中止数据传输回调函数
输入参数	huart：UART 句柄
输出参数	无
返回值	无
先决条件	无

25.2.29 函数 HAL_UART_AbortTransmitCpltCallback

描述了函数 HAL_UART_AbortTransmitCpltCallback

表25-42 函数 HAL_UART_AbortTransmitCpltCallback

函数名	HAL_UART_AbortTransmitCpltCallback
函数原形	void HAL_UART_AbortTransmitCpltCallback(UART_HandleTypeDef *huart)
功能描述	中止数据发送回调函数
输入参数	huart：UART 句柄
输出参数	无
返回值	无
先决条件	无

25.2.30 函数 HAL_UART_AbortReceiveCpltCallback

描述了函数 HAL_UART_AbortReceiveCpltCallback

表25-43 函数 HAL_UART_AbortReceiveCpltCallback

函数名	HAL_UART_AbortReceiveCpltCallback
函数原形	void HAL_UART_AbortReceiveCpltCallback(UART_HandleTypeDef *huart)
功能描述	中止数据接收回调函数
输入参数	huart：UART 句柄
输出参数	无
返回值	无
先决条件	无

25.2.31 函数 HAL_UART_SendBreak

描述了函数 HAL_UART_SendBreak

表25-44 函数 HAL_UART_SendBreak

函数名	HAL_UART_SendBreak
函数原形	HAL_StatusTypeDef HAL_UART_SendBreak(UART_HandleTypeDef *huart)
功能描述	发送一帧空白帧
输入参数	huart：UART 句柄
输出参数	无

返回值	HAL 状态
先决条件	无

25.2.32 函数 HAL_MultiProcessor_EnterMuteMode

描述了函数 HAL_MultiProcessor_EnterMuteMode

表25-45 函数 HAL_MultiProcessor_EnterMuteMode

函数名	HAL_MultiProcessor_EnterMuteMode
函数原形	HAL_StatusTypeDef HAL_MultiProcessor_EnterMuteMode(UART_HandleTypeDef *huart)
功能描述	多机通信模式下进入静默模式
输入参数	huart：UART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.33 函数 HAL_MultiProcessor_ExitMuteMode

描述了函数 HAL_MultiProcessor_ExitMuteMode

表25-46 函数 HAL_MultiProcessor_ExitMuteMode

函数名	HAL_MultiProcessor_ExitMuteMode
函数原形	HAL_StatusTypeDef HAL_MultiProcessor_ExitMuteMode(UART_HandleTypeDef *huart)
功能描述	多机通信模式下退出静默模式
输入参数	huart：UART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.34 函数 HAL_HalfDuplex_EnableTransmitter

描述了函数 HAL_HalfDuplex_EnableTransmitter

表25-47 函数 HAL_HalfDuplex_EnableTransmitter

函数名	HAL_HalfDuplex_EnableTransmitter
函数原形	HAL_StatusTypeDef HAL_HalfDuplex_EnableTransmitter(UART_HandleTypeDef *huart)
功能描述	半双工模式下开启发送功能
输入参数	huart：UART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.35 函数 HAL_HalfDuplex_EnableReceiver

描述了函数 HAL_HalfDuplex_EnableReceiver

表25-48 函数 HAL_HalfDuplex_EnableReceiver

函数名	HAL_HalfDuplex_EnableReceiver
函数原形	HAL_StatusTypeDef HAL_HalfDuplex_EnableReceiver(UART_HandleTypeDef *huart)
功能描述	半双工模式下开启接收功能
输入参数	huart：UART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

25.2.36 函数 HAL_UART_GetState

描述了函数 HAL_UART_GetState

表25-49 函数 HAL_UART_GetState

函数名	HAL_UART_GetState
函数原形	HAL_UART_StateTypeDef HAL_UART_GetState(UART_HandleTypeDef *huart)
功能描述	获取 UART 通信状态
输入参数	huart：UART 句柄
输出参数	无
返回值	UART 通信状态
先决条件	无

25.2.37 函数 HAL_UART_GetError

描述了函数 HAL_UART_GetError

表25-50 函数 HAL_UART_GetError

函数名	HAL_UART_GetError
函数原形	uint32_t HAL_UART_GetError(UART_HandleTypeDef *huart)
功能描述	获取错误代码
输入参数	huart：UART 句柄
输出参数	无
返回值	错误代码
先决条件	无

26 HAL 同步异步收发器通用驱动程序（USART）

USART 能够灵活地与外部设备进行全双工数据交换，满足外部设备对工业标准 NRZ 异步串行数据格式的要求。USART 利用分数波特率发生器提供宽范围的波特率选择。

USART 不仅支持同步单向通信和半双工单线通信，还支持多处理器通信。

26.1 USART 固件驱动寄存器结构

26.1.1 USART_InitTypeDef

USART_InitTypeDef，定义于文件"air001xx_hal_usart.h"如下：

```
typedef struct
{
uint32_t BaudRate;
uint32_t WordLength;
uint32_t StopBits;
uint32_t Parity;
uint32_t Mode;
uint32_t CLKPolarity;
uint32_t CLKPhase;
uint32_t CLKLastBit;
} USART_InitTypeDef;
```

字段说明：

表26-1 USART_InitTypeDef 字段说明

字段	描述
BaudRate	波特率
WordLength	数据长度
StopBits	停止位数
Parity	校验方式
Mode	工作模式
CLKPolarity	时钟极性
CLKPhase	时钟相位
CLKLastBit	是否输出最后一位数据时钟脉冲

参数说明：

WordLength 可选参数：

表26-2 WordLength 可选参数

参数	描述
USART_WORDLENGTH_8B	8 个数据位
USART_WORDLENGTH_9B	9 个数据位

StopBits 可选参数：

表26-3 StopBits 可选参数

参数	描述
USART_STOPBITS_1	1 个停止位
USART_STOPBITS_2	2 个停止位

Parity 可选参数：

表26-4 Parity 可选参数

参数	描述
USART_PARITY_NONE	无校验
USART_PARITY_EVEN	偶校验
USART_PARITY_ODD	奇校验

Mode 可选参数：

表26-5 Mode 可选参数

参数	描述
USART_MODE_RX	使能接收模式
USART_MODE_TX	使能发送模式
USART_MODE_TX_RX	使能接收和发送模式

CLKPolarity 可选参数：

表26-6 CLKPolarity 可选参数

参数	描述
USART_POLARITY_LOW	总线空闲时 CK 引脚上保持低电平
USART_POLARITY_HIGH	总线空闲时 CK 引脚上保持高电平

CLKPhase 可选参数：

表26-7 CLKPhase 可选参数

参数	描述
USART_PHASE_1EDGE	第一个时钟边沿采样
USART_PHASE_2EDGE	第二个时钟边沿采样

CLKLastBit 可选参数：

表26-8 CLKLastBit 可选参数

参数	描述
USART_LASTBIT_DISABLE	不发送最后一个字节的时钟脉冲
USART_LASTBIT_ENABLE	发送最后一个字节的时钟脉冲

26.1.2 USART_HandleTypeDef

USART_HandleTypeDef，定义于文件"air001xx_hal_usart.h"如下：

```
typedef struct __USART_HandleTypeDef
{
USART_TypeDef *Instance;
USART_InitTypeDef Init;
uint8_t *pTxBuffPtr;
uint16_t TxXferSize;
__IO uint16_t TxXferCount;
uint8_t *pRxBuffPtr;
uint16_t RxXferSize;
__IO uint16_t RxXferCount;
DMA_HandleTypeDef *hdmatx;
DMA_HandleTypeDef *hdmarx;
HAL_LockTypeDef Lock;
__IO HAL_USART_StateTypeDef State;
__IO uint32_t ErrorCode;
} USART_HandleTypeDef;
```

字段说明：

表26-9 USART_HandleTypeDef 字段说明

字段	描述
Instance	USART 基地址
Init	初始化参数结构体
pTxBuffPtr	发送数据缓冲区指针
TxXferSize	发送数据总量
TxXferCount	未发送的数据数量
pRxBuffPtr	接收数据缓冲区指针
RxXferSize	接收数据总量
RxXferCount	未接收的数据数量
hdmatx	DMA 发送句柄指针
hdmarx	DMA 接收句柄指针

Lock	HAL 锁
State	USART 通信状态
ErrorCode	错误代码

26.2 USART 固件库函数

表26-10 USART 固件库函数说明

函数名	描述
HAL_USART_Init	初始化 USART
HAL_USART_DeInit	将 USART 参数设为缺省值
HAL_USART_MspInit	初始化 USART 相关的 MSP
HAL_USART_MspDeInit	将 USART 相关的 MSP 设为缺省值
HAL_USART_Transmit	使用轮询的方式发送数据
HAL_USART_Receive	使用轮询的方式接收数据
HAL_USART_TransmitReceive	使用轮询的方式同时发送和接收数据
HAL_USART_Transmit_IT	使用中断的方式发送数据
HAL_USART_Receive_IT	使用中断的方式接收数据
HAL_USART_TransmitReceive_IT	使用中断的方式同时发送和接收数据
HAL_USART_Transmit_DMA	使用DMA 的方式发送数据
HAL_USART_Receive_DMA	使用DMA 的方式接收数据
HAL_USART_TransmitReceive_DMA	使用DMA 的方式同时发送和接收数据
HAL_USART_DMAPause	暂停正在进行的 DMA 传输
HAL_USART_DMAResume	恢复暂停的 DMA 传输
HAL_USART_DMAStop	停止DMA 传输
HAL_USART_Abort	中止USART 传输
HAL_USART_Abort_IT	中止USART 传输并关闭中断
HAL_USART_IRQHandler	中断请求处理
HAL_USART_TxCpltCallback	发送完成回调函数
HAL_USART_TxHalfCpltCallback	发送半完成回调函数
HAL_USART_RxCpltCallback	接收完成回调函数
HAL_USART_RxHalfCpltCallback	接收半完成回调函数
HAL_USART_TxRxCpltCallback	同时发送和接收完成回调函数
HAL_USART_ErrorCallback	错误回调函数

HAL_USART_AbortCpltCallback	中止完成回调函数
HAL_USART_GetState	获取 USART 状态
HAL_USART_GetError	获取错误代码

26.2.1 函数 HAL_USART_Init

描述了函数 HAL_USART_Init

表26-11 函数 HAL_USART_Init

函数名	HAL_USART_Init
函数原形	HAL_StatusTypeDef HAL_USART_Init(USART_HandleTypeDef * husart)
功能描述	初始化 USART
输入参数	husart：USART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

26.2.2 函数 HAL_USART_DeInit

描述了函数 HAL_USART_DeInit

表26-12 函数 HAL_USART_DeInit

函数名	HAL_USART_DeInit
函数原形	HAL_StatusTypeDef HAL_USART_DeInit(USART_HandleTypeDef *husart)
功能描述	将 USART 参数设为缺省值
输入参数	husart：USART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

26.2.3 函数 HAL_USART_MspInit

描述了函数 HAL_USART_MspInit

表26-13 函数 HAL_USART_MspInit

函数名	HAL_USART_MspInit
函数原形	void HAL_USART_MspInit(USART_HandleTypeDef *husart)
功能描述	初始化 USART 相关的 MSP
输入参数	husart：USART 句柄
输出参数	无
返回值	无
先决条件	无

26.2.4 函数 HAL_USART_MspDeInit

描述了函数 HAL_USART_MspDeInit

表26-14 函数 HAL_USART_MspDeInit

函数名	HAL_USART_MspDeInit
函数原形	void HAL_USART_MspDeInit(USART_HandleTypeDef *husart)
功能描述	将 USART 相关的 MSP 设为缺省值
输入参数	husart：USART 句柄
输出参数	无
返回值	无
先决条件	无

26.2.5 函数 HAL_USART_Transmit

描述了函数 HAL_USART_Transmit

表26-15 函数 HAL_USART_Transmit

函数名	HAL_USART_Transmit
函数原形	HAL_StatusTypeDef HAL_USART_Transmit(USART_HandleTypeDef *husart, uint8_t *pTxData, uint16_t Size, uint32_t Timeout)
功能描述	使用轮询的方式发送数据
输入参数 1	husart：USART 句柄
输入参数 2	pTxData：发送数据缓冲区指针
输入参数 3	Size：数据长度
输入参数 4	Timeout：超时时间
输出参数	无
返回值	HAL 状态
先决条件	无

26.2.6 函数 HAL_USART_Receive

描述了函数 HAL_USART_Receive

表26-16 函数 HAL_USART_Receive

函数名	HAL_USART_Receive
函数原形	HAL_StatusTypeDef HAL_USART_Receive(USART_HandleTypeDef *husart, uint8_t *pRxData, uint16_t Size, uint32_t Timeout)
功能描述	使用轮询的方式接收数据
输入参数 1	husart：USART 句柄
输入参数 2	pRxData：接收数据缓冲区指针
输入参数 3	Size：数据长度
输入参数 4	Timeout：超时时间

输出参数	无
返回值	HAL 状态
先决条件	无

26.2.7 函数 HAL_USART_TransmitReceive

描述了函数 HAL_USART_TransmitReceive

表26-17　函数 HAL_USART_TransmitReceive

函数名	HAL_USART_TransmitReceive
函数原形	HAL_StatusTypeDef HAL_USART_TransmitReceive(USART_HandleTypeDef *husart, uint8_t *pTxData, uint8_t *pRxData, uint16_t Size, uint32_t Timeout)
功能描述	使用轮询的方式同时发送和接收数据
输入参数 1	husart：USART 句柄
输入参数 2	pTxData：发送数据缓冲区指针
输入参数 3	pRxData：接收数据缓冲区指针
输入参数 4	Size：数据长度
输入参数 5	Timeout：超时时间
输出参数	无
返回值	HAL 状态
先决条件	无

26.2.8 函数 HAL_USART_Transmit_IT

描述了函数 HAL_USART_Transmit_IT

表26-18　函数 HAL_USART_Transmit_IT

函数名	HAL_USART_Transmit_IT
函数原形	HAL_StatusTypeDef HAL_USART_Transmit_IT(USART_HandleTypeDef *husart, uint8_t *pTxData, uint16_t Size)
功能描述	使用中断的方式发送数据
输入参数 1	husart：HAL 句柄
输入参数 2	pTxData：发送数据缓冲区指针
输入参数 3	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

26.2.9 函数 HAL_USART_Receive_IT

描述了函数 HAL_USART_Receive_IT

表26-19　函数 HAL_USART_Receive_IT

函数名	HAL_USART_Receive_IT

函数原形	HAL_StatusTypeDef HAL_USART_Receive_IT(USART_HandleTypeDef *husart, uint8_t *pRxData, uint16_t Size)
功能描述	使用中断的方式接收数据
输入参数 1	husart：HAL 句柄
输入参数 2	pRxData：接收数据缓冲区指针
输入参数 3	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

26.2.10 函数 HAL_USART_TransmitReceive_IT

描述了函数 HAL_USART_TransmitReceive_IT

表26-20 函数 HAL_USART_TransmitReceive_IT

函数名	HAL_USART_TransmitReceive_IT
函数原形	HAL_StatusTypeDef HAL_USART_TransmitReceive_IT(USART_HandleTypeDef *husart, uint8_t *pTxData, uint8_t *pRxData, uint16_t Size)
功能描述	使用中断的方式同时发送和接收数据
输入参数 1	husart：USART 句柄
输入参数 2	pTxData：发送数据缓冲区指针
输入参数 3	pRxData：接收数据缓冲区指针
输入参数 4	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

26.2.11 函数 HAL_USART_Transmit_DMA

描述了函数 HAL_USART_Transmit_DMA

表26-21 函数 HAL_USART_Transmit_DMA

函数名	HAL_USART_Transmit_DMA
函数原形	HAL_StatusTypeDef HAL_USART_Transmit_DMA(USART_HandleTypeDef *husart, uint8_t *pTxData, uint16_t Size)
功能描述	使用 DMA 的方式发送数据
输入参数 1	husart：HAL 句柄
输入参数 2	pTxData：发送数据缓冲区指针
输入参数 3	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

26.2.12 函数 HAL_USART_Receive_DMA

描述了函数 HAL_USART_Receive_DMA

表26-22 函数 HAL_USART_Receive_DMA

函数名	HAL_USART_Receive_DMA
函数原形	HAL_StatusTypeDef HAL_USART_Receive_DMA(USART_HandleTypeDef *husart, uint8_t *pRxData, uint16_t Size)
功能描述	使用 DMA 的方式接收数据
输入参数 1	husart：HAL 句柄
输入参数 2	pRxData：接收数据缓冲区指针
输入参数 3	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

26.2.13 函数 HAL_USART_TransmitReceive_DMA

描述了函数 HAL_USART_TransmitReceive_DMA

表26-23 函数 HAL_USART_TransmitReceive_DMA

函数名	HAL_USART_TransmitReceive_DMA
函数原形	HAL_StatusTypeDef HAL_USART_TransmitReceive_DMA(USART_HandleTypeDef *husart, uint8_t *pTxData, uint8_t *pRxData, uint16_t Size)
功能描述	使用 DMA 的方式同时发送和接收数据
输入参数 1	husart：USART 句柄
输入参数 2	pTxData：发送数据缓冲区指针
输入参数 3	pRxData：接收数据缓冲区指针
输入参数 4	Size：数据长度
输出参数	无
返回值	HAL 状态
先决条件	无

26.2.14 函数 HAL_USART_DMAPause

描述了函数 HAL_USART_DMAPause

表26-24 函数 HAL_USART_DMAPause

函数名	HAL_USART_DMAPause
函数原形	HAL_StatusTypeDef HAL_USART_DMAPause(USART_HandleTypeDef *husart)
功能描述	暂停正在进行的 DMA 传输
输入参数	husart：USART 句柄

输出参数	无
返回值	HAL 状态
先决条件	无

26.2.15 函数 HAL_USART_DMAResume

描述了函数 HAL_USART_DMAResume

表26-25 函数 HAL_USART_DMAResume

函数名	HAL_USART_DMAResume
函数原形	HAL_StatusTypeDef HAL_USART_DMAResume(USART_HandleTypeDef *husart)
功能描述	恢复暂停的 DMA 传输
输入参数	husart：USART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

26.2.16 函数 HAL_USART_DMAStop

描述了函数 HAL_USART_DMAStop

表26-26 函数 HAL_USART_DMAStop

函数名	HAL_USART_DMAStop
函数原形	HAL_StatusTypeDef HAL_USART_DMAStop(USART_HandleTypeDef *husart)
功能描述	停止 DMA 传输
输入参数	husart：USART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

26.2.17 函数 HAL_USART_Abort

描述了函数 HAL_USART_Abort

表26-27 函数 HAL_USART_Abort

函数名	HAL_USART_Abort
函数原形	HAL_StatusTypeDef HAL_USART_Abort(USART_HandleTypeDef *husart)
功能描述	中止 USART 传输
输入参数	husart：USART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

26.2.18 函数 HAL_USART_Abort_IT

描述了函数 HAL_USART_Abort_IT

表26-28 函数 HAL_USART_Abort_IT

函数名	HAL_USART_Abort_IT
函数原形	HAL_StatusTypeDef HAL_USART_Abort_IT(USART_HandleTypeDef *husart)
功能描述	中止 USART 传输并关闭中断
输入参数	husart：USART 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

26.2.19 函数 HAL_USART_IRQHandler

描述了函数 HAL_USART_IRQHandler

表26-29 函数 HAL_USART_IRQHandler

函数名	HAL_USART_IRQHandler
函数原形	void HAL_USART_IRQHandler(USART_HandleTypeDef *husart)
功能描述	中断请求处理
输入参数	husart：USART 句柄
输出参数	无
返回值	无
先决条件	无

26.2.20 函数 HAL_USART_TxCpltCallback

描述了函数 HAL_USART_TxCpltCallback

表26-30 函数 HAL_USART_TxCpltCallback

函数名	HAL_USART_TxCpltCallback
函数原形	void HAL_USART_TxCpltCallback(USART_HandleTypeDef *husart)
功能描述	发送完成回调函数
输入参数	husart：USART 句柄
输出参数	无
返回值	无
先决条件	无

26.2.21 函数 HAL_USART_TxHalfCpltCallback

描述了函数 HAL_USART_TxHalfCpltCallback

表26-31　函数 HAL_USART_TxHalfCpltCallback

函数名	HAL_USART_TxHalfCpltCallback
函数原形	void　HAL_USART_TxHalfCpltCallback(USART_HandleTypeDef　*husart)
功能描述	发送半完成回调函数
输入参数	husart：USART 句柄
输出参数	无
返回值	无
先决条件	无

26.2.22 函数 HAL_USART_RxCpltCallback

描述了函数 HAL_USART_RxCpltCallback

表26-32　函数 HAL_USART_RxCpltCallback

函数名	HAL_USART_RxCpltCallback
函数原形	void　HAL_USART_RxCpltCallback(USART_HandleTypeDef　*husart)
功能描述	接收完成回调函数
输入参数	husart：USART 句柄
输出参数	无
返回值	无
先决条件	无

26.2.23 函数 HAL_USART_RxHalfCpltCallback

描述了函数 HAL_USART_RxHalfCpltCallback

表26-33　函数 HAL_USART_RxHalfCpltCallback

函数名	HAL_USART_RxHalfCpltCallback
函数原形	void　HAL_USART_RxHalfCpltCallback(USART_HandleTypeDef　*husart)
功能描述	接收半完成回调函数
输入参数	husart：USART 句柄
输出参数	无
返回值	无
先决条件	无

26.2.24 函数 HAL_USART_TxRxCpltCallback

描述了函数 HAL_USART_TxRxCpltCallback

表26-34　函数 HAL_USART_TxRxCpltCallback

函数名	HAL_USART_TxRxCpltCallback
函数原形	void　HAL_USART_TxRxCpltCallback(USART_HandleTypeDef　*husart)

功能描述	同时发送和接收完成回调函数
输入参数	husart：USART 句柄
输出参数	无
返回值	无
先决条件	无

26.2.25 函数 HAL_USART_ErrorCallback

描述了函数 HAL_USART_ErrorCallback

表26-35 函数 HAL_USART_ErrorCallback

函数名	HAL_USART_ErrorCallback
函数原形	void HAL_USART_ErrorCallback(USART_HandleTypeDef *husart)
功能描述	错误回调函数
输入参数	husart：USART 句柄
输出参数	无
返回值	无
先决条件	无

26.2.26 函数 HAL_USART_AbortCpltCallback

描述了函数 HAL_USART_AbortCpltCallback

表26-36 函数 HAL_USART_AbortCpltCallback

函数名	HAL_USART_AbortCpltCallback
函数原形	void HAL_USART_AbortCpltCallback(USART_HandleTypeDef *husart)
功能描述	中止传输回调函数
输入参数	husart：USART 句柄
输出参数	无
返回值	无
先决条件	无

26.2.27 函数 HAL_USART_GetState

描述了函数 HAL_USART_GetState

表26-37 函数 HAL_USART_GetState

函数名	HAL_USART_GetState
函数原形	HAL_USART_StateTypeDef HAL_USART_GetState(USART_HandleTypeDef *husart)
功能描述	获取 USART 状态
输入参数	husart：USART 句柄
输出参数	无

返回值	USART 状态
先决条件	无

26.2.28 函数 HAL_USART_GetError

描述了函数 HAL_USART_GetError

表26-38　函数 HAL_USART_GetError

函数名	HAL_USART_GetError
函数原形	uint32_t HAL_USART_GetError(USART_HandleTypeDef *husart)
功能描述	获取错误代码
输入参数	husart：USART 句柄
输出参数	无
返回值	错误代码
先决条件	无

27 HAL 窗口看门狗通用驱动程序（WWDG）

系统窗口看门狗(WWDG)通常被用来监测由外部干扰或不可预见的逻辑条件造成的应用程序背离正常的运行序列而产生的软件故障。除非程序在 T6 位变成 0 前刷新递减计数器的值，否则看门狗电路在达到预置的时间周期时，会产生一个 MCU 复位。如果在递减计数器达到窗口寄存器值之前刷新控制寄存器中的 7 位递减计数器值，也会产生 MCU 复位。这意味着必须在限定的时间窗口内刷新计数器。

WWDG 时钟由 APB 时钟经预分频后提供，通过可配置的时间窗口来检测应用程序非正常的过迟或过早的操作。

WWDG 最适合那些要求看门狗在精确计时窗口起作用的应用程序。

27.1 WWDG 固件驱动寄存器结构

27.1.1 WWDG_InitTypeDef

WWDG_InitTypeDef，定义于文件"air001xx_hal_wwdg.h"如下：

```
typedef struct
{
uint32_t Prescaler;
uint32_t Window;
uint32_t Counter;
uint32_t EWIMode ;
} WWDG_InitTypeDef;
```

字段说明：

表27-1 WWDG_InitTypeDef

字段	描述
Prescaler	时钟预分频值
Window	窗口值（0x40~0x7F）
Counter	装载值（0x40~0x7F）
EWIMode	提前唤醒中断使能

参数说明：

Prescaler 可选参数：

表27-2 Prescaler 可选参数

参数	描述
WWDG_PRESCALER_1	WWDG 时钟 1 分频
WWDG_PRESCALER_2	WWDG 时钟 2 分频

WWDG_PRESCALER_4	WWDG 时钟 4 分频
WWDG_PRESCALER_8	WWDG 时钟 8 分频

EWIMode 可选参数：

表27-3 EWIMode 可选参数

参数	描述
WWDG_EWI_DISABLE	关闭提前唤醒中断功能
WWDG_EWI_ENABLE	开启提前唤醒中断功能

27.1.2 WWDG_HandleTypeDef

WWDG_HandleTypeDef，定义于文件"air001xx_hal_wwdg.h"如下：

```
typedef struct
{
WWDG_TypeDef *Instance;
WWDG_InitTypeDef Init;
} WWDG_HandleTypeDef;
```

字段说明：

表27-4 WWDG_HandleTypeDef 字段说明

字段	描述
Instance	WWDG 基地址
Init	初始化参数结构体

27.2 WWDG 固件库函数

表27-5 WWDG 固件库函数说明

函数名	描述
HAL_WWDG_Init	初始化WWDG
HAL_WWDG_MspInit	初始化WWDG 相关的 MSP
HAL_WWDG_Refresh	刷新计数器
HAL_WWDG_IRQHandler	中断请求处理
HAL_WWDG_EarlyWakeupCallback	提前唤醒回调函数

27.2.1 函数 HAL_WWDG_Init

描述了函数 HAL_WWDG_Init

表27-6 函数 HAL_WWDG_Init

函数名	HAL_WWDG_Init
函数原形	HAL_StatusTypeDef HAL_WWDG_Init(WWDG_HandleTypeDef *hwwdg)

功能描述	初始化WWDG
输入参数	hwwdg：WWDG 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

27.2.2 函数 HAL_WWDG_MspInit

描述了函数 HAL_WWDG_MspInit

表27-7 函数 HAL_WWDG_MspInit

函数名	HAL_WWDG_MspInit
函数原形	void HAL_WWDG_MspInit(WWDG_HandleTypeDef *hwwdg)
功能描述	初始化WWDG 相关的 MSP
输入参数	hwwdg：WWDG 句柄
输出参数	无
返回值	无
先决条件	无

27.2.3 函数 HAL_WWDG_Refresh

描述了函数 HAL_WWDG_Refresh

表27-8 函数 HAL_WWDG_Refresh

函数名	HAL_WWDG_Refresh
函数原形	HAL_StatusTypeDef HAL_WWDG_Refresh(WWDG_HandleTypeDef *hwwdg)
功能描述	刷新计数器
输入参数	hwwdg：WWDG 句柄
输出参数	无
返回值	HAL 状态
先决条件	无

27.2.4 函数 HAL_WWDG_IRQHandler

描述了函数 HAL_WWDG_IRQHandler

表27-9 函数 HAL_WWDG_IRQHandler

函数名	HAL_WWDG_IRQHandler
函数原形	void HAL_WWDG_IRQHandler(WWDG_HandleTypeDef *hwwdg)
功能描述	中断请求处理
输入参数	hwwdg：WWDG 句柄
输出参数	无

返回值	无
先决条件	无

27.2.5 函数 HAL_WWDG_EarlyWakeupCallback

描述了函数 HAL_WWDG_EarlyWakeupCallback

表27-10　函数 HAL_WWDG_EarlyWakeupCallback

函数名	HAL_WWDG_EarlyWakeupCallback
函数原形	void HAL_WWDG_EarlyWakeupCallback(WWDG_HandleTypeDef *hwwdg)
功能描述	提前唤醒中断回调函数
输入参数	hwwdg：WWDG 句柄
输出参数	无
返回值	无
先决条件	无

28 LL 模拟/数字转换器通用驱动程序（ADC）

12 位 ADC 是逐次逼近型模数转换器。它具有多达 12 个复用通道，可测量来自 10 个外部源和 2 个内部源的信号。

各种不同通道的 A/D 转换可在单次、连续、扫描或不连续采样模式下进行。ADC 的结果存储在一个左对齐或右对齐的 16 位数据寄存器中。

ADC 具有模拟看门狗特性，允许应用检测输入电压是否超过了用户自定义的阈值上限或下限。

ADC 实现了在低频率下运行，可获得极低的功耗。

28.1 ADC 固件驱动寄存器结构

28.1.1 LL_ADC_InitTypeDef

LL_ADC_InitTypeDef，定义于文件“air001xx_ll_adc.h”如下：

```
typedef struct
{
uint32_t Clock;
uint32_t Resolution;
uint32_t DataAlignment;
uint32_t LowPowerMode;
}LL_ADC_InitTypeDef;
```

字段说明：

表28-1 LL_ADC_InitTypeDef 字段说明

字段	描述
Clock	配置ADC 时钟源
Resolution	配置ADC 分辨率
DataAlignment	配置ADC 数据对齐方式
LowPowerMode	配置ADC 等待转换模式

参数说明：

Clock 可选参数：

表28-2 Clock 可选参数

参数	描述
LL_ADC_CLOCK_SYNC_PCLK_DIV1	选择 APB 作为时钟源，不分频
LL_ADC_CLOCK_SYNC_PCLK_DIV2	选择 APB 作为时钟源，2 分频

LL_ADC_CLOCK_SYNC_PCLK_DIV4	选择 APB 作为时钟源，4 分频
LL_ADC_CLOCK_SYNC_PCLK_DIV8	选择 APB 作为时钟源，8 分频
LL_ADC_CLOCK_SYNC_PCLK_DIV16	选择 APB 作为时钟源，16 分频
LL_ADC_CLOCK_SYNC_PCLK_DIV32	选择 APB 作为时钟源，32 分频
LL_ADC_CLOCK_SYNC_PCLK_DIV64	选择 APB 作为时钟源，64 分频
LL_ADC_CLOCK_ASYNC_HSI_DIV1	选择 HSI 作为时钟源，不分频
LL_ADC_CLOCK_ASYNC_HSI_DIV2	选择 HSI 作为时钟源，2 分频
LL_ADC_CLOCK_ASYNC_HSI_DIV4	选择 HSI 作为时钟源，4 分频
LL_ADC_CLOCK_ASYNC_HSI_DIV8	选择 HSI 作为时钟源，8 分频
LL_ADC_CLOCK_ASYNC_HSI_DIV16	选择 HSI 作为时钟源，16 分频
LL_ADC_CLOCK_ASYNC_HSI_DIV32	选择 HSI 作为时钟源，32 分频
LL_ADC_CLOCK_ASYNC_HSI_DIV64	选择 HSI 作为时钟源，64 分频

Resolution 可选参数：

表28-3 Resolution 可选参数

参数	描述
LL_ADC_RESOLUTION_12B	12 位分辨率
LL_ADC_RESOLUTION_10B	10 位分辨率
LL_ADC_RESOLUTION_8B	8 位分辨率
LL_ADC_RESOLUTION_6B	6 位分辨率

DataAlignment 可选参数：

表28-4 DataAlignment 可选参数

参数	描述
LL_ADC_DATA_ALIGN_RIGHT	右对齐
LL_ADC_DATA_ALIGN_LEFT	左对齐

LowPowerMode 可选参数：

表28-5 LowPowerMode 可选参数

参数	描述
LL_ADC_LP_MODE_NONE	等待模式关闭
LL_ADC_LP_AUTOWAIT	等待模式打开

28.1.2 LL_ADC_REG_InitTypeDef

LL_ADC_REG_InitTypeDef，定义于文件“air001xx_ll_adc.h”如下：

```
typedef struct
{
uint32_t TriggerSource;
```

```
uint32_t SequencerDiscont;
uint32_t ContinuousMode;
uint32_t DMATransfer;
uint32_t Overrun;
}LL_ADC_REG_InitTypeDef;
```

字段说明:

表29-1 LL_ADC_REG_InitTypeDef 字段说明

字段	描述
TriggerSource	配置ADC 触发源
SequencerDiscont	配置ADC 非连续模式
ContinuousMode	配置ADC 单次/连续转换模式
DMATransfer	配置ADC DMA 模式
Overrun	配置ADC 过载管理模式

参数说明

TriggerSource 可选参数:

表29-2 TriggerSource 可选参数

参数	描述
LL_ADC_REG_TRIG_SOFTWARE	软件触发
LL_ADC_REG_TRIG_EXT_TIM1_TRGO	TIM1_TRGO 触发
LL_ADC_REG_TRIG_EXT_TIM1_CH4	TIM1_CH4 触发
LL_ADC_REG_TRIG_EXT_TIM3_TRGO	TIM3_TRGO 触发

SequencerDiscont 可选参数:

表29-3 SequencerDiscont 可选参数

参数	描述
LL_ADC_REG_SEQ_DISCONT_DISABLE	关闭非连续模式
LL_ADC_REG_SEQ_DISCONT_1RANK	开启非连续模式

ContinousMode 可选参数:

表29-4 ContinousMode 可选参数

参数	描述
LL_ADC_REG_CONV_SINGLE	单次转换模式
LL_ADC_REG_CONV_CONTINUOUS	连续转换模式

DMATransfer 可选参数:

表29-5 DMATransfer 可选参数

参数	描述
LL_ADC_REG_DMA_TRANSFER_NONE	DMA 关闭
LL_ADC_REG_DMA_TRANSFER_LIMITED	DMA 单次模式
LL_ADC_REG_DMA_TRANSFER_UNLIMITED	DMA 循环模式

Overrun 可选参数:

表29-6 Overrun 可选参数

参数	描述
LL_ADC_REG_OVR_DATA_PRESERVED	保留原有数据
LL_ADC_REG_OVR_DATA_OVERWRITTEN	覆盖原有数据

28.2 ADC 固件库函数

表29-7 ADC 固件库函数说明

函数名	描述
LL_ADC_SetClock	设置ADC 时钟
LL_ADC_GetClock	获取ADC 时钟
LL_ADC_SetResolution	设置分辨率
LL_ADC_GetResolution	获取分辨率
LL_ADC_SetDataAlignment	设置对齐方式
LL_ADC_GetDataAlignment	获取对齐方式
LL_ADC_SetLowPowerMode	设置等待转换模式
LL_ADC_GetLowPowerMode	获取等待转换模式
LL_ADC_SetSamplingTimeCommonChannels	设置采样时间
LL_ADC_GetSamplingTimeCommonChannels	获取采样时间
LL_ADC_SetCommonPathInternalCh	设置ADC 内部通道
LL_ADC_SetCommonPathInternalCh	获取ADC 内部通道
LL_ADC_REG_SetTriggerSource	设置触发源
LL_ADC_REG_GetTriggerSource	获取触发源
LL_ADC_REG_IsTriggerSourceSWStart	检查是否软件触发
LL_ADC_REG_SetTriggerEdge	设置触发沿
LL_ADC_REG_GetTriggerEdge	获取触发沿
LL_ADC_REG_SetSequencerScanDirection	设置扫描方向
LL_ADC_REG_GetSequencerScanDirection	获取扫描方向

LL_ADC_REG_SetSequencerDiscont	设置非连续模式
LL_ADC_REG_GetSequencerDiscont	获取非连续模式
LL_ADC_REG_SetContinuousMode	设置单次/连续转换模式
LL_ADC_REG_GetContinuousMode	获取单次/连续转换模式
LL_ADC_REG_SetDMATransfer	设置DMA 模式
LL_ADC_REG_GetDMATransfer	获取DMA 模式
LL_ADC_REG_SetOverrun	设置过载管理模式
LL_ADC_REG_GetOverrun	获取过载管理模式
LL_ADC_REG_SetSequencerChannels	设置ADC 通道
LL_ADC_REG_SetSequencerChAdd	添加ADC 通道
LL_ADC_REG_SetSequencerChRem	删除ADC 通道
LL_ADC_SetAnalogWDMonitChannels	设置AWD 通道
LL_ADC_GetAnalogWDMonitChannels	获取AWD 通道
LL_ADC_ConfigAnalogWDThresholds	配置AWD 高阈值和低阈值
LL_ADC_SetAnalogWDThresholds	设置AWD 高阈值或低阈值
LL_ADC_GetAnalogWDThresholds	获取AWD 高阈值或低阈值
LL_ADC_StartCalibration	开始校准
LL_ADC_IsCalibrationOnGoing	是否校准中
LL_ADC_Enable	使能ADC
LL_ADC_Disable	禁用ADC
LL_ADC_IsEnabled	ADC 使能状态
LL_ADC_Reset	复位ADC
LL_ADC_REG_StartConversion	开始转换
LL_ADC_REG_StopConversion	停止转换
LL_ADC_REG_IsConversionOnGoing	是否转换中
LL_ADC_REG_IsStopConversionOnGoing	是否停止转换中
LL_ADC_REG_ReadConversionData32	获取ADC 的 32 位转换结果
LL_ADC_REG_ReadConversionData12	获取ADC 的 12 位转换结果
LL_ADC_REG_ReadConversionData10	获取ADC 的 10 位转换结果
LL_ADC_REG_ReadConversionData8	获取ADC 的 8 位转换结果
LL_ADC_REG_ReadConversionData6	获取ADC 的 6 位转换结果

LL_ADC_IsActiveFlag_EOC	检查EOC 标志是否置位
LL_ADC_IsActiveFlag_EOS	检查EOS 标志是否置位
LL_ADC_IsActiveFlag_OVR	检查 OVR 标志是否置位
LL_ADC_IsActiveFlag_EOSMP	检查EOSMP 标志是否置位
LL_ADC_IsActiveFlag_AWD	检查AWD 标志是否置位
LL_ADC_ClearFlag_EOC	清除EOC 标志位
LL_ADC_ClearFlag_EOS	清除EOS 标志位
LL_ADC_ClearFlag_OVR	清除 OVR 标志位
LL_ADC_ClearFlag_EOSMP	清除EOSMP 标志位
LL_ADC_ClearFlag_AWD	清除AWD 标志位
LL_ADC_EnableIT_EOC	使能EOC 中断
LL_ADC_EnableIT_EOS	使能EOS 中断
LL_ADC_EnableIT_OVR	使能 OVR 中断
LL_ADC_EnableIT_EOSMP	使能EOSMP 中断
LL_ADC_EnableIT_AWD	使能AWD 中断
LL_ADC_DisableIT_EOC	禁用EOC 中断
LL_ADC_DisableIT_EOS	禁用EOS 中断
LL_ADC_DisableIT_OVR	禁用 OVR 中断
LL_ADC_DisableIT_EOSMP	禁用EOSMP 中断
LL_ADC_DisableIT_AWD	禁用AWD 中断
LL_ADC_IsEnableIT_EOC	检查EOC 中断是否使能
LL_ADC_IsEnableIT_EOS	检查EOS 中断是否使能
LL_ADC_IsEnableIT_OVR	检查 OVR 中断是否使能
LL_ADC_IsEnableIT_EOSMP	检查EOSMP 中断是否使能
LL_ADC_IsEnableIT_AWD	检查AWD 中断是否使能
LL_ADC_SetCalibrationSamplingTime	设置ADC 校准采样时间
LL_ADC_GetCalibrationSamplingTime	获取ADC 校准采样时间
LL_ADC_SetCalibrationMode	设置ADC 校准模式
LL_ADC_GetCalibrationMode	获取ADC 校准模式
LL_ADC_GetCalibrationStatus	获取ADC 校准状态
LL_ADC_ClearCalibrationStatus	清空ADC 校准状态

LL_ADC_Init	初始化 ADC
LL_ADC_REG_Init	初始化 ADC REG
LL_ADC_DeInit	将 ADC 配置重设为缺省值
LL_ADC_CommonDeInit	将通用 ADC 配置重设为缺省值
__LL_ADC_COMMNON_INSTANCE	设置通用ADC
__LL_ADC_CONVERT_DATA_RESOLUTION	ADC 分辨率转换
__LL_ADC_CALC_DATA_TO_VOLTAGE	ADC 数据转电压
__LL_ADC_CALC_VREFANALOG_VOLTAGE	ADC 内部参考计算模拟参考电压值
__LL_ADC_CALC_TEMPERATURE	ADC 计算温度值
__LL_ADC_CALC_TEMPERATURE_TYP_PARAMS	ADC 通过传感器参数计算温度值

28.2.1 函数 LL_ADC_SetClock

描述了函数 LL_ADC_SetClock

表29-8 函数 LL_ADC_SetClock

函数名	LL_ADC_SetClock
函数原形	__STATIC_INLINE void LL_ADC_SetClock(ADC_TypeDef *ADCx,uint32_t ClockSource)
功能描述	设置 ADC 的时钟
输入参数 1	ADCx：ADC 实例
输入参数 2	ClockSource：时钟源
输出参数	无
返回值	无
先决条件	无

ClockSource 可选参数：

表29-9 ClockSource 可选参数

参数	描述
LL_ADC_CLOCK_SYNC_PCLK_DIV1	选择 APB 作为时钟源，不分频
LL_ADC_CLOCK_SYNC_PCLK_DIV2	选择 APB 作为时钟源，2 分频
LL_ADC_CLOCK_SYNC_PCLK_DIV4	选择 APB 作为时钟源，4 分频
LL_ADC_CLOCK_SYNC_PCLK_DIV8	选择 APB 作为时钟源，8 分频
LL_ADC_CLOCK_SYNC_PCLK_DIV16	选择 APB 作为时钟源，16 分频
LL_ADC_CLOCK_SYNC_PCLK_DIV32	选择 APB 作为时钟源，32 分频
LL_ADC_CLOCK_SYNC_PCLK_DIV64	选择 APB 作为时钟源，64 分频
LL_ADC_CLOCK_ASYNC_HSI_DIV1	选择 HSI 作为时钟源，不分频
LL_ADC_CLOCK_ASYNC_HSI_DIV2	选择 HSI 作为时钟源，2 分频

LL_ADC_CLOCK_ASYNC_HSI_DIV4	选择 HSI 作为时钟源，4 分频
LL_ADC_CLOCK_ASYNC_HSI_DIV8	选择 HSI 作为时钟源，8 分频
LL_ADC_CLOCK_ASYNC_HSI_DIV16	选择 HSI 作为时钟源，16 分频
LL_ADC_CLOCK_ASYNC_HSI_DIV32	选择 HSI 作为时钟源，32 分频
LL_ADC_CLOCK_ASYNC_HSI_DIV64	选择 HSI 作为时钟源，64 分频

28.2.2 函数 LL_ADC_GetClock

描述了函数 LL_ADC_GetClock

表29-10 函数 LL_ADC_GetClock

函数名	LL_ADC_GetClock
函数原形	__STATIC_INLINE uint32_t LL_ADC_GetClock(ADC_TypeDef *ADCx)
功能描述	获取 ADC 的时钟
输入参数	ADCx：ADC 实例
输出参数	无
返回值	ClockSource：时钟源
先决条件	无

28.2.3 函数 LL_ADC_SetResolution

描述了函数 LL_ADC_SetResolution

表29-11 函数 LL_ADC_SetResolution

函数名	LL_ADC_SetResolution
函数原形	__STATIC_INLINE void LL_ADC_SetResolution(ADC_TypeDef *ADCx,uint32_t Resolution)
功能描述	设置 ADC 的分辨率
输入参数 1	ADCx：ADC 实例
输入参数 2	Resolution：分辨率
输出参数	无
返回值	无
先决条件	无

Resolution 可选参数：

表29-12 Resolution 可选参数

参数	描述
LL_ADC_RESOLUTION_12B	12 位分辨率
LL_ADC_RESOLUTION_10B	10 位分辨率
LL_ADC_RESOLUTION_8B	8 位分辨率
LL_ADC_RESOLUTION_6B	6 位分辨率

28.2.4 函数 LL_ADC_GetResolution

描述了函数 LL_ADC_GetResolution

表29-13 函数 LL_ADC_GetResolution

函数名	LL_ADC_GetResolution
函数原形	__STATIC_INLINE uint32_t LL_ADC_GetResolution(ADC_TypeDef *ADCx)
功能描述	获取 ADC 的分辨率
输入参数	ADCx：ADC 实例
输出参数	无
返回值	Resolution：分辨率
先决条件	无

28.2.5 函数 LL_ADC_SetDataAlignment

描述了函数 LL_ADC_SetDataAlignment

表29-14 函数 LL_ADC_SetDataAlignment

函数名	LL_ADC_SetDataAlignment
函数原形	__STATIC_INLINE void LL_ADC_SetDataAlignment(ADC_TypeDef *ADCx,uint32_t DataAlignment)
功能描述	设置 ADC 的对齐方式
输入参数 1	ADCx：ADC 实例
输入参数 2	DataAlignment：对齐方式
输出参数	无
返回值	无
先决条件	无

DataAlignment 可选参数:

表29-15 DataAlignment 可选参数

参数	描述
LL_ADC_DATA_ALIGN_RIGHT	右对齐
LL_ADC_DATA_ALIGN_LEFT	左对齐

28.2.6 函数 LL_ADC_GetDataAlignment

描述了函数 LL_ADC_GetDataAlignment

表29-16 函数 LL_ADC_GetDataAlignment

函数名	LL_ADC_GetDataAlignment
函数原形	__STATIC_INLINE uint32_t LL_ADC_GetDataAlignment(ADC_TypeDef *ADCx)
功能描述	获取 ADC 的对齐方式
输入参数	ADCx：ADC 实例

输出参数	无
返回值	DataAlignment：对齐方式
先决条件	无

28.2.7 函数 LL_ADC_SetLowPowerMode

描述了函数 LL_ADC_SetLowPowerMode

表29-17 函数 LL_ADC_SetLowPowerMode

函数名	LL_ADC_SetLowPowerMode
函数原形	__STATIC_INLINE void LL_ADC_SetLowPowerMode(ADC_TypeDef *ADCx,uint32_t LowPowerMode)
功能描述	设置 ADC 的等待转换模式
输入参数 1	ADCx：ADC 实例
输入参数 2	LowPowerMode：等待模式
输出参数	无
返回值	无
先决条件	无

LowPowerMode 可选参数:

表29-18 LowPowerMode 可选参数

参数	描述
LL_ADC_LP_MODE_NONE	等待模式关闭
LL_ADC_LP_AUTOWAIT	等待模式打开

28.2.8 函数 LL_ADC_GetLowPowerMode

描述了函数 LL_ADC_GetLowPowerMode

表29-19 函数 LL_ADC_GetLowPowerMode

函数名	LL_ADC_GetLowPowerMode
函数原形	__STATIC_INLINE uint32_t LL_ADC_GetLowPowerMode(ADC_TypeDef *ADCx)
功能描述	获取 ADC 的等待转换模式
输入参数	ADCx：ADC 实例
输出参数	无
返回值	LowPowerMode：等待模式
先决条件	无

28.2.9 函数 LL_ADC_SetSamplingTimeCommonChannels

描述了函数 LL_ADC_SetSamplingTimeCommonChannels

表29-20 函数 LL_ADC_SetSamplingTimeCommonChannels

函数名	LL_ADC_SetClock

函数原形	__STATIC_INLINE void LL_ADC_SetSamplingTimeCommonChannels (ADC_TypeDef *ADCx,uint32_t SamplingTime)
功能描述	设置 ADC 的采样时间
输入参数 1	ADCx：ADC 实例
输入参数 2	SamplingTime：采样时间
输出参数	无
返回值	无
先决条件	无

SamplingTime 可选参数:

表29-21 SamplingTime 可选参数

参数	描述
LL_ADC_SAMPLINGTIME_3CYCLES_5	3.5ADC 时钟周期
LL_ADC_SAMPLINGTIME_5CYCLES_5	5.5ADC 时钟周期
LL_ADC_SAMPLINGTIME_7CYCLES_5	7.5ADC 时钟周期
LL_ADC_SAMPLINGTIME_13CYCLES_5	13.5ADC 时钟周期
LL_ADC_SAMPLINGTIME_28CYCLES_5	28.5ADC 时钟周期
LL_ADC_SAMPLINGTIME_41CYCLES_5	41.5ADC 时钟周期
LL_ADC_SAMPLINGTIME_71CYCLES_5	71.5ADC 时钟周期
LL_ADC_SAMPLINGTIME_239CYCLES_5	239.5ADC 时钟周期

28.2.10 函数 LL_ADC_GetSamplingTimeCommonChannels

描述了函数 LL_ADC_GetSamplingTimeCommonChannels

表29-22 函数 LL_ADC_GetSamplingTimeCommonChannels

函数名	LL_ADC_GetSamplingTimeCommonChannels
函数原形	__STATIC_INLINE uint32_t LL_ADC_GetSamplingCommonChannels (ADC_TypeDef *ADCx)
功能描述	获取 ADC 的采样时间
输入参数	ADCx：ADC 实例
输出参数	无
返回值	SamplingTime：采样时间
先决条件	无

28.2.11 函数 LL_ADC_SetCommonPathInternalCh

描述了函数 LL_ADC_SetCommonPathInternalCh

表29-23 函数 LL_ADC_SetCommonPathInternalCh

函数名	LL_ADC_SetCommpnPathInternalCh
函数原形	__STATIC_INLINE void LL_ADC_SetCommonPathInternalCh(ADC_TypeDef *ADCx,uint32_t PathInternal)

功能描述	设置 ADC 的内部转换通道
输入参数 1	ADCx：ADC 实例
输入参数 2	PathInternal：内部通道
输出参数	无
返回值	无
先决条件	无

PathInternal 可选参数：

表29-24 PathInternal 可选参数

参数	描述
LL_ADC_PATH_INTERNAL_NONE	无
LL_ADC_PATH_INTERNAL_VREFINT	内部基准电压
LL_ADC_PATH_INTERNAL_TEMPSENSOR	内部温度传感器

28.2.12 函数 LL_ADC_GetCommonPathInternalCh

描述了函数 LL_ADC_GetCommonPathInternalCh

表29-25 函数 LL_ADC_GetCommonPathInternalCh

函数名	LL_ADC_GetCommonPathInternalCh
函数原形	__STATIC_INLINE uint32_t LL_ADC_GetCommonPathInternalCh (ADC_TypeDef *ADCx)
功能描述	获取 ADC 的内部通道
输入参数	ADCx：ADC 实例
输出参数	无
返回值	PathInternal：内部通道
先决条件	无

28.2.13 函数 LL_ADC_REG_SetTriggerSource

描述了函数 LL_ADC_REG_SetTriggerSource

表29-26 函数 LL_ADC_REG_SetTriggerSource

函数名	LL_ADC_REG_SetTriggerSource
函数原形	__STATIC_INLINE void LL_ADC_REG_SetTriggerSource(ADC_TypeDef *ADCx,uint32_t TriggerSource)
功能描述	设置 ADC 的触发源
输入参数 1	ADCx：ADC 实例
输入参数 2	TriggerSource：触发源
输出参数	无
返回值	无
先决条件	无

TriggerSource 可选参数:

表29-27 TriggerSource 可选参数

参数	描述
LL_ADC_REG_TRIG_SOFTWARE	软件触发
LL_ADC_REG_TRIG_EXT_TIM1_TRGO	TIM1_TRGO 触发
LL_ADC_REG_TRIG_EXT_TIM1_CH4	TIM1_CH4 触发
LL_ADC_REG_TRIG_EXT_TIM3_TRGO	TIM3_TRGO 触发

28.2.14 函数 LL_ADC_REG_GetTriggerSource

描述了函数 LL_ADC_REG_GetTriggerSource

表29-28 函数 LL_ADC_REG_GetTriggerSource

函数名	LL_ADC_REG_GetTriggerSource
函数原形	__STATIC_INLINE uint32_t LL_ADC_REG_GetTriggerSource(ADC_TypeDef *ADCx)
功能描述	获取 ADC 的触发源
输入参数	ADCx：ADC 实例
输出参数	无
返回值	TriggerSource：触发源
先决条件	无

28.2.15 函数 LL_ADC_REG_IsTriggerSourceSWStart

描述了函数 LL_ADC_REG_IsTriggerSourceSWStart

表29-29 函数 LL_ADC_REG_IsTriggerSourceSWStart

函数名	LL_ADC_REG_IsTriggerSourceSWStart
函数原形	__STATIC_INLINE uint32_t LL_ADC_REG_IsTriggerSourceSWStart (ADC_TypeDef *ADCx)
功能描述	是否是软件触发
输入参数	ADCx：ADC 实例
输出参数	无
返回值	0：外部触发 1：软件触发
先决条件	无

28.2.16 函数 LL_ADC_REG_SetTriggerEdge

描述了函数 LL_ADC_REG_SetTriggerEdge

表29-30 函数 LL_ADC_REG_SetTriggerEdge

函数名	LL_ADC_SetTriggerEdge
函数原形	__STATIC_INLINE void LL_ADC_REG_SetTriggerEdge(ADC_TypeDef *ADCx,uint32_t ExternalTriggerEdge)
功能描述	设置 ADC 的触发沿

输入参数 1	ADCx：ADC 实例
输入参数 2	ExternalTriggerEdge：外部触发沿
输出参数	无
返回值	无
先决条件	无

ExternalTriggerEdge 可选参数：

表29-31 ExternalTriggerEdge 可选参数

参数	描述
LL_ADC_REG_TRIG_EXT_RISING	上升沿触发
LL_ADC_REG_TRIG_EXT_FALLING	下降沿触发
LL_ADC_REG_TRIG_EXT_RISINGFALLING	上升沿和下降沿触发

28.2.17 函数 LL_ADC_REG_GetTriggerEdge

描述了函数 LL_ADC_REG_GetTriggerEdge

表29-32 函数 LL_ADC_REG_GetTriggerEdge

函数名	LL_ADC_REG_GetTriggerEdge
函数原形	__STATIC_INLINE uint32_t LL_ADC_REG_GetTriggerEdge(ADC_TypeDef *ADCx)
功能描述	获取 ADC 的触发沿
输入参数	ADCx：ADC 实例
输出参数	无
返回值	ExternalTriggerEdge：外部触发沿
先决条件	无

28.2.18 函数 LL_ADC_REG_SetSequencerScanDirection

描述了函数 LL_ADC_REG_SetSequencerScanDirection

表29-33 函数 LL_ADC_REG_SetSequencerScanDirection

函数名	LL_ADC_SetSequencerScanDirection
函数原形	__STATIC_INLINE void LL_ADC_SetSequencerScanDirection(ADC_TypeDef *ADCx,uint32_t ScanDirection)
功能描述	设置 ADC 的扫描方向
输入参数 1	ADCx：ADC 实例
输入参数 2	ScanDirection：扫描方向
输出参数	无
返回值	无
先决条件	无

ScanDirection 可选参数：

表29-34 ScanDirection 可选参数

参数	描述
LL_ADC_REG_SEQ_SCAN_DIR_FORWARD	向上扫描
LL_ADC_REG_SEQ_SCAN_DIR_BACKWARD	向下扫描

28.2.19 函数 LL_ADC_REG_GetSequencerScanDirection

描述了函数 LL_ADC_REG_GetSequencerScanDirection

表29-35 函数 LL_ADC_REG_GetSequencerScanDirection

函数名	LL_ADC_REG_GetSequencerScanDirection
函数原形	__STATIC_INLINE uint32_t LL_ADC_REG_GetSequencerScanDirection (ADC_TypeDef *ADCx)
功能描述	获取 ADC 的扫描方向
输入参数	ADCx：ADC 实例
输出参数	无
返回值	ScanDirection：扫描方向
先决条件	无

28.2.20 函数 LL_ADC_REG_SetSequencerDiscont

描述了函数 LL_ADC_REG_SetSequencerDiscont

表29-36 函数 LL_ADC_REG_SetSequencerDiscont

函数名	LL_ADC_REG_SetSequencerDiscont
函数原形	__STATIC_INLINE void LL_ADC_REG_SetSequencerDiscont(ADC_TypeDef *ADCx,uint32_t SeqDiscont)
功能描述	设置 ADC 的非连续模式
输入参数 1	ADCx：ADC 实例
输入参数 2	SeqDiscont：非连续模式
输出参数	无
返回值	无
先决条件	无

SeqDiscont 可选参数：

表29-37 SeqDiscont 可选参数

参数	描述
LL_ADC_REG_SEQ_DISCONT_DISABLE	关闭非连续模式
LL_ADC_REG_SEQ_DISCONT_1RANK	开启非连续模式

28.2.21 函数 LL_ADC_REG_GetSequencerDiscont

描述了函数 LL_ADC_REG_GetSequencerDiscont

表29-38 函数 LL_ADC_REG_GetSequencerDiscont

函数名	LL_ADC_REG_GetSequencerDiscont
函数原形	__STATIC_INLINE uint32_t LL_ADC_REG_GetSequencerDiscont (ADC_TypeDef *ADCx)
功能描述	获取 ADC 的非连续模式
输入参数	ADCx：ADC 实例
输出参数	无
返回值	SeqDiscont：非连续模式
先决条件	无

28.2.22 函数 LL_ADC_REG_SetContinuousMode

描述了函数 LL_ADC_REG_SetContinuousMode

表29-39 函数 LL_ADC_REG_SetContinuousMode

函数名	LL_ADC_REG_SetContinuousMode
函数原形	__STATIC_INLINE void LL_ADC_REG_SetContinousMode(ADC_TypeDef *ADCx,uint32_t Continuous)
功能描述	设置 ADC 的单次/连续转换模式
输入参数 1	ADCx：ADC 实例
输入参数 2	Continuous：单次/连续转换模式
输出参数	无
返回值	无
先决条件	无

Continuous 可选参数:

表29-40 Continuous 可选参数

参数	描述
LL_ADC_REG_CONV_SINGLE	单次转换模式
LL_ADC_REG_CONV_CONTINUOUS	连续转换模式

28.2.23 函数 LL_ADC_REG_GetContinuousMode

描述了函数 LL_ADC_REG_GetContinuousMode

表29-41 函数 LL_ADC_REG_GetContinuousMode

函数名	LL_ADC_REG_GetContinuousMode
函数原形	__STATIC_INLINE uint32_t LL_ADC_REG_GetContinuousMode(ADC_TypeDef *ADCx)
功能描述	获取 ADC 的单次/连续转换模式

输入参数	ADCx：ADC 实例
输出参数	无
返回值	Continuous：单次/连续转换模式
先决条件	无

28.2.24 函数 LL_ADC_REG_SetDMATransfer

描述了函数 LL_ADC_REG_SetDMATransfer

表29-42 函数 LL_ADC_REG_SetDMATransfer

函数名	LL_ADC_REG_SetDMATransfer
函数原形	__STATIC_INLINE void LL_ADC_REG_SetDMATransfer(ADC_TypeDef *ADCx,uint32_t DMATransfer)
功能描述	设置 ADC 的 DMA 传输模式
输入参数 1	ADCx：ADC 实例
输入参数 2	DMATransfer：DMA 传输模式
输出参数	无
返回值	无
先决条件	无

DMATransfer 可选参数：

表29-43 DMATransfer 可选参数

参数	描述
LL_ADC_REG_DMA_TRANSFER_NONE	DMA 模式关闭
LL_ADC_REG_DMA_TRANSFER_LIMITED	DMA 单次模式
LL_ADC_REG_DMA_TRANSFER_UNLIMITED	DMA 循环模式

28.2.25 函数 LL_ADC_REG_GetDMATransfer

描述了函数 LL_ADC_REG_GetDMATransfer

表29-44 函数 LL_ADC_REG_GetDMATransfer

函数名	LL_ADC_REG_GetDMATransfer
函数原形	__STATIC_INLINE uint32_t LL_ADC_REG_GetDMATransfer(ADC_TypeDef *ADCx)
功能描述	获取 ADC 的 DMA 传输模式
输入参数	ADCx：ADC 实例
输出参数	无
返回值	DMATransfer：DMA 传输模式
先决条件	无

28.2.26 函数 LL_ADC_REG_SetOverrun

描述了函数 LL_ADC_REG_SetOverrun

表29-45 函数 LL_ADC_REG_SetOverrun

函数名	LL_ADC_REG_SetOverrun
函数原形	__STATIC_INLINE void LL_ADC_REG_SetOverrun(ADC_TypeDef *ADCx,uint32_t Overrun)
功能描述	设置 ADC 的过载管理模式
输入参数 1	ADCx：ADC 实例
输入参数 2	Overun：过载管理模式
输出参数	无
返回值	无
先决条件	无

Overrun 可选参数:

表29-46 Overrun 可选参数

参数	描述
LL_ADC_REG_OVR_DATA_PRESERVED	保留原有数据
LL_ADC_REG_OVR_DATA_OVERWRITTEN	覆盖原有数据

28.2.27 函数 LL_ADC_REG_GetOverrun

描述了函数 LL_ADC_REG_GetOverrun

表29-47 函数 LL_ADC_REG_GetOverrun

函数名	LL_ADC_REG_GetOverrun
函数原形	__STATIC_INLINE uint32_t LL_ADC_REG_GetOverrun(ADC_TypeDef *ADCx)
功能描述	获取 ADC 的过载管理模式
输入参数	ADCx：ADC 实例
输出参数	无
返回值	Overrun：过载管理模式
先决条件	无

28.2.28 函数 LL_ADC_REG_SetSequencerChannels

描述了函数 LL_ADC_REG_SetSequencerChannels

表29-48 函数 LL_ADC_REG_SetSequencerChannels

函数名	LL_ADC_REG_SetSequencerChannels
函数原形	__STATIC_INLINE void LL_ADC_REG_SetSequencerChannels(ADC_TypeDef *ADCx,uint32_t Channel)
功能描述	设置 ADC 的通道

输入参数 1	ADCx：ADC 实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数:

表29-49 Channel 可选参数

参数	描述
LL_ADC_CHANNEL_0	通道 0
LL_ADC_CHANNEL_1	通道 1
LL_ADC_CHANNEL_2	通道 2
LL_ADC_CHANNEL_3	通道 3
LL_ADC_CHANNEL_4	通道 4
LL_ADC_CHANNEL_5	通道 5
LL_ADC_CHANNEL_6	通道 6
LL_ADC_CHANNEL_7	通道 7
LL_ADC_CHANNEL_8	通道 8
LL_ADC_CHANNEL_9	通道 9
LL_ADC_CHANNEL_11	通道 11
LL_ADC_CHANNEL_12	通道 12
LL_ADC_CHANNEL_VREFINT	通道 VREFINT
LL_ADC_CHANNEL_TEMPSENSOR	通道 TEMPSENSOR

28.2.29 函数 LL_ADC_REG_SetSequencerChAdd

描述了函数 LL_ADC_REG_SetSequencerChAdd

表29-50 函数 LL_ADC_REG_SetSequencerChAdd

函数名	LL_ADC_REG_SetSequencerChAdd
函数原形	__STATIC_INLINE uint32_t LL_ADC_REG_SetSequencerChAdd(ADC_TypeDef *ADCx,uint32_t Channel)
功能描述	添加 ADC 通道
输入参数 1	ADCx：ADC 实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表29-51 Channel 可选参数

参数	描述
LL_ADC_CHANNEL_0	通道 0
LL_ADC_CHANNEL_1	通道 1
LL_ADC_CHANNEL_2	通道 2
LL_ADC_CHANNEL_3	通道 3
LL_ADC_CHANNEL_4	通道 4
LL_ADC_CHANNEL_5	通道 5
LL_ADC_CHANNEL_6	通道 6
LL_ADC_CHANNEL_7	通道 7
LL_ADC_CHANNEL_8	通道 8
LL_ADC_CHANNEL_9	通道 9
LL_ADC_CHANNEL_11	通道 11
LL_ADC_CHANNEL_12	通道 12
LL_ADC_CHANNEL_VREFINT	通道 VREFINT
LL_ADC_CHANNEL_TEMPSENSOR	通道 TEMPSENSOR

28.2.30 函数 LL_ADC_REG_SetSequencerChRem

描述了函数 LL_ADC_REG_SetSequencerChRem

表29-52 函数 LL_ADC_REG_SetSequencerChRem

函数名	LL_ADC_REG_SetSequencerChRem
函数原形	__STATIC_INLINE uint32_t LL_ADC_REG_SetSequencerChRem (ADC_TypeDef *ADCx,uint32_t Channel)
功能描述	删除 ADC 通道
输入参数 1	ADCx：ADC 实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表29-53 Channel 可选参数

参数	描述
LL_ADC_CHANNEL_0	通道 0
LL_ADC_CHANNEL_1	通道 1
LL_ADC_CHANNEL_2	通道 2

LL_ADC_CHANNEL_3	通道 3
LL_ADC_CHANNEL_4	通道 4
LL_ADC_CHANNEL_5	通道 5
LL_ADC_CHANNEL_6	通道 6
LL_ADC_CHANNEL_7	通道 7
LL_ADC_CHANNEL_8	通道 8
LL_ADC_CHANNEL_9	通道 9
LL_ADC_CHANNEL_11	通道 11
LL_ADC_CHANNEL_12	通道 12
LL_ADC_CHANNEL_VREFINT	通道 VREFINT
LL_ADC_CHANNEL_TEMPSENSOR	通道 TEMPSENSOR

28.2.31 函数 LL_ADC_SetAnalogWDMonitChannels

描述了函数 LL_ADC_SetAnalogWDMonitChannels

表29-54 函数 LL_ADC_SetAnalogWDMonitChannels

函数名	LL_ADC_SetAnalogWDMonitChannels
函数原形	__STATIC_INLINE void LL_ADC_SetAnalogWDMonitChannels (ADC_TypeDef *ADCx,uint32_t AWDChannelGroup)
功能描述	设置 ADC 的 AWD 通道
输入参数 1	ADCx：ADC 实例
输入参数 2	AWDChannelGroup：AWD 通道
输出参数	无
返回值	无
先决条件	无

AWDChannelGroup 可选参数：

表29-55 AWDChannelGroup 可选参数

参数	描述
LL_ADC_AWD_DISABLE	关闭 AWD
LL_ADC_ALL_CHANNELS_REG	使能全部通道 AWD
LL_ADC_ALL_CHANNELS_0_REG	使能第 0 个通道 AWD
LL_ADC_ALL_CHANNELS_1_REG	使能第 1 个通道 AWD
LL_ADC_ALL_CHANNELS_2_REG	使能第 2 个通道 AWD
LL_ADC_ALL_CHANNELS_3_REG	使能第 3 个通道 AWD
LL_ADC_ALL_CHANNELS_4_REG	使能第 4 个通道 AWD
LL_ADC_ALL_CHANNELS_5_REG	使能第 5 个通道 AWD
LL_ADC_ALL_CHANNELS_6_REG	使能第 6 个通道 AWD

LL_ADC_ALL_CHANNELS_7_REG	使能第 7 个通道 AWD
LL_ADC_ALL_CHANNELS_8_REG	使能第 8 个通道 AWD
LL_ADC_ALL_CHANNELS_9_REG	使能第 9 个通道 AWD
LL_ADC_ALL_CHANNELS_11_REG	使能第 11 个通道 AWD
LL_ADC_ALL_CHANNELS_12_REG	使能第 12 个通道 AWD
LL_ADC_ALL_CH_VREFINT_REG	使能 VREFINT 通道 AWD
LL_ADC_ALL_CH_TEMPSENSOR_REG	使能 TEMPSENSOR 通道 AWD

28.2.32 函数 LL_ADC_GetAnalogWDMonitChannels

描述了函数 LL_ADC_GetAnalogWDMonitChannels

表29-56 函数 LL_ADC_GetAnalogWDMonitChannels

函数名	LL_ADC_GetAnalogWDMonitChannels
函数原形	__STATIC_INLINE uint32_t LL_ADC_GetAnalogWDMonitChannels (ADC_TypeDef *ADCx)
功能描述	获取 ADC 的 AWD 通道
输入参数	ADCx：ADC 实例
输出参数	无
返回值	AWDChannelGroup：AWD 通道
先决条件	无

28.2.33 函数 LL_ADC_ConfigAnalogWDThresholds

描述了函数 LL_ADC_ConfigAnalogWDThresholds

表29-57 函数 LL_ADC_REG_ ConfigAnalogWDThresholds

函数名	LL_ADC_ConfigAnalogWDThresholds
函数原形	__STATIC_INLINE void LL_ADC_ConfigAnalogWDThresholds (ADC_TypeDef *ADCx,uint32_t AWDThresholdHighValue, uint32_t AWDThresholdLowValue)
功能描述	配置 AWD 的高阈值和低阈值
输入参数 1	ADCx：ADC 实例
输入参数 2	AWDThresholdHighValue：高阈值
输入参数 3	AWDThresholdLowValue：低阈值
输出参数	无
返回值	无
先决条件	无

28.2.34 函数 LL_ADC_SetAnalogWDThresholds

描述了函数 LL_ADC_REG_SetAnalogWDThresholds

表29-58 函数 LL_ADC_REG_SetAnalogWDThresholds

函数名	LL_ADC_SetAnalogWDThresholds

函数原形	__STATIC_INLINE void LL_ADC_SetAnalogWDThresholds(ADC_TypeDef *ADCx,uint32_t AWDThresholdsHighLow,uint32_t AWDThresholdValue)
功能描述	设置 AWD 的高阈值或低阈值
输入参数 1	ADCx：ADC 实例
输入参数 2	AWDThresholdsHighLow：高阈值或低阈值
输入参数 3	AWDThresholdValue：高阈值或低阈值的数据
输出参数	无
返回值	无
先决条件	无

AWDThresholdsHighLow 可选参数:

表29-59 AWDThresholdsHighLow 可选参数

参数	描述
LL_ADC_AWD_THRESHOLD_HIGH	设置高阈值
LL_ADC_AWD_THRESHOLD_LOW	设置低阈值

28.2.35 函数 LL_ADC_GetAnalogWDThresholds

描述了函数 LL_ADC_GetAnalogWDThresholds

表29-60 函数 LL_ADC_GetAnalogWDThresholds

函数名	LL_ADC_GetAnalogWDThresholds
函数原形	__STATIC_INLINE uint32_t LL_ADC_GetAnalogWDThresholds(ADC_TypeDef *ADCx, uint32_t AWDThresholdsHighLow)
功能描述	获取 AWD 的高阈值或低阈值
输入参数	ADCx：ADC 实例
输出参数	无
返回值	高/低阈值的数据
先决条件	无

AWDThresholdsHighLow 可选参数:

表29-61 AWDThresholdsHighLow 可选参数

参数	描述
LL_ADC_AWD_THRESHOLD_HIGH	获取高阈值
LL_ADC_AWD_THRESHOLD_LOW	获取低阈值

28.2.36 函数 LL_ADC_StartCalibration

描述了函数 LL_ADC_StartCalibration

表29-62 函数 LL_ADC_StartCalibraion

函数名	LL_ADC_StartCalibration

函数原形	__STATIC_INLINE void LL_ADC_StartCalibration(ADC_TypeDef *ADCx)
功能描述	ADC 开始校准
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.37 函数 LL_ADC_IsCalibrationOnGoing

描述了函数 LL_ADC_IsCalibrationOnGoing

表29-63 函数 LL_ADC_IsCalibrationOnGoing

函数名	LL_ADC_REG_IsCalibrationOnGoing
函数原形	__STATIC_INLINE uint32_t LL_ADC_IsCalibrationOnGoing(ADC_TypeDef *ADCx)
功能描述	检查 ADC 是否正在校准中
输入参数	ADCx：ADC 实例
输出参数	无
返回值	0：未正在校准 1：正在校准
先决条件	无

28.2.38 函数 LL_ADC_Enable

描述了函数 LL_ADC_Enable

表29-64 函数 LL_ADC_Enable

函数名	LL_ADC_Enable
函数原形	__STATIC_INLINE void LL_ADC_Enable(ADC_TypeDef *ADCx)
功能描述	使能 ADC
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.39 函数 LL_ADC_Disable

描述了函数 LL_ADC_Disable

表29-65 函数 LL_ADC_Disable

函数名	LL_ADC_Disable
函数原形	ErrorStatus LL_ADC_Disable(ADC_TypeDef *ADCx)
功能描述	禁用 ADC
输入参数	ADCx：ADC 实例

输出参数	无
返回值	0：Disable 不成功 1：Disable 成功
先决条件	无

28.2.40 函数 LL_ADC_IsEnabled

描述了函数 LL_ADC_IsEnabled

表29-66 函数 LL_ADC_IsEnabled

函数名	LL_ADC_IsEnabled
函数原形	__STATIC_INLINE uint32_t LL_ADC_IsEnabled(ADC_TypeDef *ADCx)
功能描述	检查 ADC 是否使能
输入参数	ADCx：ADC 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

28.2.41 函数 LL_ADC_Reset

描述了函数 LL_ADC_Reset

表29-67 函数 LL_ADC_Reset

函数名	LL_ADC_Reset
函数原形	__STATIC_INLINE uint32_t LL_ADC_Reset(ADC_TypeDef *ADCx)
功能描述	复位 ADC
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.42 函数 LL_ADC_REG_StartConversion

描述了函数 LL_ADC_REG_StartConversion

表29-68 函数 LL_ADC_REG_StartConversion

函数名	LL_ADC_REG_StartConversion
函数原形	__STATIC_INLINE void LL_ADC_REG_StartConversion(ADC_TypeDef *ADCx)
功能描述	开始 ADC 转换
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.43 函数 LL_ADC_REG_StopConversion

描述了函数 LL_ADC_REG_StopConversion

表29-69 函数 LL_ADC_REG_StopConversion

函数名	LL_ADC_REG_StopConversion
函数原形	__STATIC_INLINE void LL_ADC_REG_StopConversion(ADC_TypeDef *ADCx)
功能描述	停止 ADC 转换
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.44 函数 LL_ADC_REG_IsConversionOnGoing

描述了函数 LL_ADC_REG_IsConversionOnGoing

表29-70 函数 LL_ADC_REG_IsConversionOnGoing

函数名	LL_ADC_REG_IsConversionOnGoing
函数原形	__STATIC_INLINE uint32_t LL_ADC_REG_IsConversionOnGoing (ADC_TypeDef *ADCx)
功能描述	ADC 是否正在转换
输入参数	ADCx：ADC 实例
输出参数	无
返回值	0：未正在转换 1：正在转换
先决条件	无

28.2.45 函数 LL_ADC_REG_IsStopConversionOnGoing

描述了函数 LL_ADC_REG_IsStopConversionOnGoing

表29-71 函数 LL_ADC_REG_IsStopConversionOnGoing

函数名	LL_ADC_REG_IsStopConversionOnGoing
函数原形	__STATIC_INLINE uint32_t LL_ADC_REG_IsStopConversionOnGoing (ADC_TypeDef *ADCx)
功能描述	ADC 是否正在停止转换
输入参数	ADCx：ADC 实例
输出参数	无
返回值	0：未正在停止转换 1：正在停止转换
先决条件	无

28.2.46 函数 LL_ADC_REG_ReadConversionData32

描述了函数 LL_ADC_REG_ReadConversionData32

表29-72 函数 LL_ADC_REG_ReadConversionData32

函数名	LL_ADC_REG_ReadConversionData32
函数原形	__STATIC_INLINE uint32_t LL_ADC_REG_ ReadConversionData32 (ADC_TypeDef *ADCx)
功能描述	读取 ADC 的 32 位数据
输入参数	ADCx：ADC 实例
输出参数	无
返回值	ADC 转换结果
先决条件	无

28.2.47 函数 LL_ADC_REG_ReadConversionData12

描述了函数 LL_ADC_REG_ReadConversionData12

表29-73 函数 LL_ADC_REG_ReadConversionData12

函数名	LL_ADC_REG_ReadConversionData12
函数原形	__STATIC_INLINE uint16_t LL_ADC_REG_ReadConversionData12 (ADC_TypeDef *ADCx)
功能描述	读取 ADC 的 12 位数据
输入参数	ADCx：ADC 实例
输出参数	无
返回值	ADC 转换结果
先决条件	无

28.2.48 函数 LL_ADC_REG_ReadConversionData10

描述了函数 LL_ADC_REG_ReadConversionData10

表29-74 函数 LL_ADC_REG_ReadConversionData10

函数名	LL_ADC_REG_ReadConversionData10
函数原形	__STATIC_INLINE uint16_t LL_ADC_REG_ReadConversionData10 (ADC_TypeDef *ADCx)
功能描述	读取 ADC 的 10 位数据
输入参数	ADCx：ADC 实例
输出参数	无
返回值	ADC 转换结果
先决条件	无

28.2.49 函数 LL_ADC_REG_ReadConversionData8

描述了函数 LL_ADC_REG_ReadConversionData8

表29-75 函数 LL_ADC_REG_ReadConversionData8

函数名	LL_ADC_REG_ReadConversionData8
函数原形	__STATIC_INLINE uint8_t LL_ADC_REG_ReadConversionData8

	(ADC_TypeDef *ADCx)
功能描述	读取 ADC 的 8 位数据
输入参数	ADCx：ADC 实例
输出参数	无
返回值	ADC 转换结果
先决条件	无

28.2.50 函数 LL_ADC_REG_ReadConversionData6

描述了函数 LL_ADC_REG_ReadConversionData6

表29-76 函数 LL_ADC_REG_ReadConversionData6

函数名	LL_ADC_REG_ReadConversionData6
函数原形	__STATIC_INLINE uint8_t LL_ADC_REG_ReadConversionData6 (ADC_TypeDef *ADCx)
功能描述	读取 ADC 的 6 位数据
输入参数	ADCx：ADC 实例
输出参数	无
返回值	ADC 转换结果
先决条件	无

28.2.51 函数 LL_ADC_IsActiveFlag_EOC

描述了函数 LL_ADC_IsActiveFlag_EOC

表29-77 函数 LL_ADC_IsActiveFlag_EOC

函数名	LL_ADC_IsActiveFlag_EOC
函数原形	__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_EOC(ADC_TypeDef *ADCx)
功能描述	检查 EOC 标志位
输入参数	ADCx：ADC 实例
输出参数	无
返回值	0：未置位 1：置位
先决条件	无

28.2.52 函数 LL_ADC_IsActiveFlag_EOS

描述了函数 LL_ADC_IsActiveFlag_EOS

表29-78 函数 LL_ADC_IsActiveFlag_EOS

函数名	LL_ADC_IsActiveFlag_EOS
函数原形	__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_EOS(ADC_TypeDef *ADCx)
功能描述	检查 EOS 标志位
输入参数	ADCx：ADC 实例

输出参数	无
返回值	0：未置位 1：置位
先决条件	无

28.2.53 函数 LL_ADC_IsActiveFlag_OVR

描述了函数 LL_ADC_IsActiveFlag_OVR

表29-79 函数 LL_ADC_IsActiveFlag_OVR

函数名	LL_ADC_IsActiveFlag_OVR
函数原形	__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_OVR(ADC_TypeDef *ADCx)
功能描述	检查 OVR 标志位
输入参数	ADCx：ADC 实例
输出参数	无
返回值	0：未置位 1：置位
先决条件	无

28.2.54 函数 LL_ADC_IsActiveFlag_EOSMP

描述了函数 LL_ADC_IsActiveFlag_EOSMP

表29-80 函数 LL_ADC_IsActiveFlag_EOSMP

函数名	LL_ADC_IsActiveFlag_EOSMP
函数原形	__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_EOSMP(ADC_TypeDef *ADCx)
功能描述	检查 EOSMP 标志位
输入参数	ADCx：ADC 实例
输出参数	无
返回值	0：未置位 1：置位
先决条件	无

28.2.55 函数 LL_ADC_IsActiveFlag_AWD

描述了函数 LL_ADC_IsActiveFlag_AWD

表29-81 函数 LL_ADC_IsActiveFlag_AWD

函数名	LL_ADC_IsActiveFlag_AWD
函数原形	__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_AWD(ADC_TypeDef *ADCx)
功能描述	检查 AWD 标志位
输入参数	ADCx：ADC 实例
输出参数	无
返回值	0：未置位 1：置位
先决条件	无

28.2.56 函数 LL_ADC_ClearFlag_EOC

描述了函数 LL_ADC_ClearFlag_EOC

表29-82 函数 LL_ADC_ClearFlag_EOC

函数名	LL_ADC_ClearFlag_EOC
函数原形	__STATIC_INLINE void LL_ADC_ClearFlag_EOC(ADC_TypeDef *ADCx)
功能描述	清除 EOC 标志位
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.57 函数 LL_ADC_ClearFlag_EOS

描述了函数 LL_ADC_ClearFlag_EOS

表29-83 函数 LL_ADC_ClearFlag_EOS

函数名	LL_ADC_ClearFlag_EOS
函数原形	__STATIC_INLINE void LL_ADC_ClearFlag_EOS(ADC_TypeDef *ADCx)
功能描述	清除 EOS 标志位
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.58 函数 LL_ADC_ClearFlag_OVR

描述了函数 LL_ADC_ClearFlag_OVR

表29-84 函数 LL_ADC_ClearFlag_OVR

函数名	LL_ADC_ClearFlag_OVR
函数原形	__STATIC_INLINE void LL_ADC_ClearFlag_OVR(ADC_TypeDef *ADCx)
功能描述	清除 OVR 标志位
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.59 函数 LL_ADC_ClearFlag_EOSMP

描述了函数 LL_ADC_ClearFlag_EOSMP

表29-85 函数 LL_ADC_ClearFlag_EOSMP

函数名	LL_ADC_ClearFlag_EOSMP
函数原形	__STATIC_INLINE void LL_ADC_ClearFlag_EOSMP(ADC_TypeDef *ADCx)
功能描述	清除 EOSMP 标志位
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.60 函数 LL_ADC_ClearFlag_AWD

描述了函数 LL_ADC_ClearFlag_AWD

表29-86　函数 LL_ADC_ClearFlag_AWD

函数名	LL_ADC_ClearFlag_AWD
函数原形	__STATIC_INLINE void LL_ADC_ClearFlag_AWD(ADC_TypeDef *ADCx)
功能描述	清除 AWD 标志位
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.61 函数 LL_ADC_EnableIT_EOC

描述了函数 LL_ADC_EnableIT_EOC

表29-87　函数 LL_ADC_EnableIT_EOC

函数名	LL_ADC_EnableIT_EOC
函数原形	__STATIC_INLINE void LL_ADC_EnableIT_EOC(ADC_TypeDef *ADCx)
功能描述	使能 EOC 中断
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.62 函数 LL_ADC_EnableIT_EOS

描述了函数 LL_ADC_EnableIT_EOS

表29-88　函数 LL_ADC_EnableIT_EOS

函数名	LL_ADC_EnableIT_EOS
函数原形	__STATIC_INLINE void LL_ADC_EnableIT_EOS(ADC_TypeDef *ADCx)
功能描述	使能 EOS 中断

输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.63 函数 LL_ADC_EnableIT_OVR

描述了函数 LL_ADC_EnableIT_OVR

表29-89 函数 LL_ADC_EnableIT_OVR

函数名	LL_ADC_EnableIT_OVR
函数原形	__STATIC_INLINE void LL_ADC_EnableIT_OVR(ADC_TypeDef *ADCx)
功能描述	使能 OVR 中断
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.64 函数 LL_ADC_EnableIT_EOSMP

描述了函数 LL_ADC_EnableIT_EOSMP

表29-90 函数 LL_ADC_EnableIT_EOSMP

函数名	LL_ADC_EnableIT_EOSMP
函数原形	__STATIC_INLINE void LL_ADC_EnableIT_EOSMP(ADC_TypeDef *ADCx)
功能描述	使能 EOSMP 中断
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.65 函数 LL_ADC_EnableIT_AWD

描述了函数 LL_ADC_EnableIT_AWD

表29-91 函数 LL_ADC_EnableIT_AWD

函数名	LL_ADC_EnableIT_AWD
函数原形	__STATIC_INLINE void LL_ADC_EnableIT_AWD(ADC_TypeDef *ADCx)
功能描述	使能 AWD 中断
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无

先决条件	无

28.2.66 函数 LL_ADC_DisableIT_EOC

描述了函数 LL_ADC_DisableIT_EOC

表29-92 函数 LL_ADC_DisableIT_EOC

函数名	LL_ADC_DisableIT_EOC
函数原形	__STATIC_INLINE void LL_ADC_DisableIT_EOC(ADC_TypeDef *ADCx)
功能描述	禁用 EOC 中断
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.67 函数 LL_ADC_DisableIT_EOS

描述了函数 LL_ADC_DisableIT_EOS

表29-93 函数 LL_ADC_DisableIT_EOS

函数名	LL_ADC_DisableIT_EOS
函数原形	__STATIC_INLINE void LL_ADC_DisableIT_EOS(ADC_TypeDef *ADCx)
功能描述	禁用 EOS 中断
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.68 函数 LL_ADC_DisableIT_OVR

描述了函数 LL_ADC_DisableIT_OVR

表29-94 函数 LL_ADC_DisableIT_OVR

函数名	LL_ADC_DisableIT_OVR
函数原形	__STATIC_INLINE void LL_ADC_DisableIT_OVR(ADC_TypeDef *ADCx)
功能描述	禁用 OVR 中断
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.69 函数 LL_ADC_DisableIT_EOSMP

描述了函数 LL_ADC_DisableIT_EOSMP

表29-95 函数 LL_ADC_DisableIT_EOSMP

函数名	LL_ADC_DisableIT_EOSMP
函数原形	__STATIC_INLINE void LL_ADC_DisableIT_EOSMP(ADC_TypeDef *ADCx)
功能描述	禁用 EOSMP 中断
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.70 函数 LL_ADC_DisableIT_AWD

描述了函数 LL_ADC_DisableIT_AWD

表29-96 函数 LL_ADC_DisableIT_AWD

函数名	LL_ADC_DisableIT_AWD
函数原形	__STATIC_INLINE void LL_ADC_DisableIT_AWD(ADC_TypeDef *ADCx)
功能描述	禁用 AWD 中断
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.71 函数 LL_ADC_IsEnabledIT_EOC

描述了函数 LL_ADC_IsEnabledIT_EOC

表29-97 函数 LL_ADC_IsEnabledIT_EOC

函数名	LL_ADC_IsEnabledIT_EOC
函数原形	__STATIC_INLINE uint32_t LL_ADC_IsEnabledIT_EOC(ADC_TypeDef *ADCx)
功能描述	检查 EOC 中断是否使能
输入参数	ADCx：ADC 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

28.2.72 函数 LL_ADC_IsEnabledIT_EOS

描述了函数 LL_ADC_IsEnabledIT_EOS

表29-98 函数 LL_ADC_IsEnabledIT_EOS

函数名	LL_ADC_IsEnabledIT_EOS
函数原形	__STATIC_INLINE uint32_t LL_ADC_IsEnabledIT_EOS(ADC_TypeDef *ADCx)

功能描述	检查 EOS 中断是否使能
输入参数	ADCx：ADC 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

28.2.73 函数 LL_ADC_IsEnabledIT_OVR

描述了函数 LL_ADC_IsEnabledIT_OVR

表29-99　函数 LL_ADC_IsEnabledIT_OVR

函数名	LL_ADC_IsEnabledIT_OVR
函数原形	__STATIC_INLINE uint32_t LL_ADC_IsEnabledIT_OVR(ADC_TypeDef *ADCx)
功能描述	检查 OVR 中断是否使能
输入参数	ADCx：ADC 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

28.2.74 函数 LL_ADC_IsEnabledIT_EOSMP

描述了函数 LL_ADC_IsEnabledIT_EOSMP

表29-100 函数 LL_ADC_IsEnabledIT_EOSMP

函数名	LL_ADC_IsEnabledIT_EOSMP
函数原形	__STATIC_INLINE uint32_t LL_ADC_IsEnabledIT_EOSMP(ADC_TypeDef *ADCx)
功能描述	检查 EOSMP 中断是否使能
输入参数	ADCx：ADC 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

28.2.75 函数 LL_ADC_IsEnabledIT_AWD

描述了函数 LL_ADC_IsEnabledIT_AWD

表29-101 函数 LL_ADC_IsEnabledIT_AWD

函数名	LL_ADC_IsEnabledIT_AWD
函数原形	__STATIC_INLINE uint32_t LL_ADC_IsEnabledIT_AWD(ADC_TypeDef *ADCx)
功能描述	检查 AWD 中断是否使能
输入参数	ADCx：ADC 实例
输出参数	无

返回值	状态位（1 或 0）
先决条件	无

28.2.76 函数 LL_ADC_SetCalibrationSamplingTime

描述了函数 LL_ADC_SetCalibrationSamplingTime

表29-102 函数 LL_ADC_SetCalibrationSamplingTime

函数名	LL_ADC_SetCalibrationSamplingTime
函数原形	__STATIC_INLINE void LL_ADC_SetCalibrationSamplingTime(ADC_TypeDef *ADCx,uint32_t CalibrationSamplingTime)
功能描述	设置 ADC 的校准采样时间
输入参数 1	ADCx：ADC 实例
输入参数 2	CalibraitonSamplingTime：校准采样时间
输出参数	无
返回值	无
先决条件	无

CalibrationSamplingTime 可选参数:

表29-103 CalibrationSamplingTime 可选参数

参数	描述
LL_ADC_CAL_SAMPLINGTIME_2CYCLES	2ADC 时钟
LL_ADC_CAL_SAMPLINGTIME_4CYCLES	4ADC 时钟
LL_ADC_CAL_SAMPLINGTIME_8CYCLES	8ADC 时钟
LL_ADC_CAL_SAMPLINGTIME_1CYCLE	1ADC 时钟

28.2.77 函数 LL_ADC_GetCalibraitonSampingTime

描述了函数 LL_ADC_GetCalibraitonSamplingTime

表29-104 函数 LL_ADC_GetCalibrationSamplingTime

函数名	LL_ADC_GetCalibrationSamplingTime
函数原形	__STATIC_INLINE uint32_t LL_ADC_GetCalibraitonSampingTime (ADC_TypeDef *ADCx)
功能描述	获取 ADC 的校准采样时间
输入参数	ADCx：ADC 实例
输出参数	无
返回值	CalibraitonSamplingTime：校准采样时间
先决条件	无

28.2.78 函数 LL_ADC_SetCalibrationMode

描述了函数 LL_ADC_SetCalibrationMode

表29-105 函数 LL_ADC_SetCalibrationMode

函数名	LL_ADC_SetCalibrationMode
函数原形	__STATIC_INLINE void LL_ADC_SetCalibrationMode(ADC_TypeDef *ADCx,uint32_t CalibrationMode)
功能描述	设置 ADC 的校准模式
输入参数 1	ADCx：ADC 实例
输入参数 2	CalibraitonMode：校准模式
输出参数	无
返回值	无
先决条件	无

CalibrationMode 可选参数:

表29-106 CalibrationMode 可选参数

参数	描述
LL_ADC_CAL_MODE_OFFSET	只校准 OFFSET
LL_ADC_CAL_MODE_OFFSETANDLINEARITY	校准 OFFSET 和 LINERARITY

28.2.79 函数 LL_ADC_GetCalibraitonMode

描述了函数 LL_ADC_GetCalibraitonMode

表29-107 函数 LL_ADC_GetCalibrationMode

函数名	LL_ADC_GetCalibrationMode
函数原形	__STATIC_INLINE uint32_t LL_ADC_GetCalibraitonMode (ADC_TypeDef *ADCx)
功能描述	获取 ADC 的校准模式
输入参数	ADCx：ADC 实例
输出参数	无
返回值	CalibraitonMode：校准模式
先决条件	无

28.2.80 函数 LL_ADC_GetCalibraitonStatus

描述了函数 LL_ADC_GetCalibraitonStatus

表29-108 函数 LL_ADC_GetCalibrationStatus

函数名	LL_ADC_GetCalibrationStaus
函数原形	__STATIC_INLINE uint32_t LL_ADC_GetCalibraitonStatus (ADC_TypeDef *ADCx)
功能描述	获取 ADC 的校准状态
输入参数	ADCx：ADC 实例
输出参数	无
返回值	CalibraitonStatus：校准状态

先决条件	无

CalibrationStatus 可选参数:

表29-109 CalibrationStatus 可选参数

参数	描述
LL_ADC_CAL_STATUS_SUCCESS	校准成功
LL_ADC_CAL_STATUS_FAIL	校准失败
LL_ADC_CAL_STATUS_ONGOING	正在校准
LL_ADC_CAL_STATUS_INVALID	无效状态

28.2.81 函数 LL_ADC_ClearCalibrationStatus

描述了函数 LL_ADC_ClearCalibrationStatus

表29-110 函数 LL_ADC_ClearCalibrationStatus

函数名	LL_ADC_ClearCaibrationStatus
函数原形	__STATIC_INLINE void LL_ADC_ClearCalibrationStatus (ADC_TypeDef *ADCx)
功能描述	清空 ADC 的校准状态
输入参数	ADCx：ADC 实例
输出参数	无
返回值	无
先决条件	无

28.2.82 函数 LL_ADC_Init

描述了函数 LL_ADC_Init

表29-111 函数 LL_ADC_Init

函数名	LL_ADC_Init
函数原形	ErrorStatus LL_ADC_Init(ADC_TypeDef *ADCx,LL_ADC_InitTypeDef *ADC_InitStruct)
功能描述	初始化 ADC
输入参数 1	ADCx：ADC 实例
输入参数 2	ADC_InitStrut：指向 ADC_InitTypeDef 的指针
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

28.2.83 函数 LL_ADC_REG_Init

描述了函数 LL_ADC_REG_Init

表29-112 函数 LL_ADC_REG_Init

函数名	LL_ADC_REG_Init

函数原形	ErrorStatus LL_ADC_Init(ADC_TypeDef *ADCx,LL_ADC_REG_InitTypeDef *ADC_REG_InitStruct)
功能描述	初始化 ADC REG
输入参数 1	ADCx：ADC 实例
输入参数 2	ADC_REG_InitStrut：指向 ADC_REG_InitTypeDef 的指针
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

28.2.84 函数 LL_ADC_DeInit

描述了函数 LL_ADC_DeInit

表29-113 函数 LL_ADC_DeInit

函数名	LL_ADC_DeInit
函数原形	ErrorStatus LL_ADC_DeInit(ADC_TypeDef *ADCx)
功能描述	将 ADC 配置重设为缺省值
输入参数	ADCx：ADC 实例
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

28.2.85 函数 LL_ADC_CommonDeInit

描述了函数 LL_ADC_CommonDeInit

表29-114 函数 LL_ADC_CommonDeInit

函数名	LL_ADC_CommonDeInit
函数原形	ErrorStatus LL_ADC_CommonDeInit(ADC_Common_TypeDef *ADCxy_COMMON)
功能描述	将 ADC 通用配置重设为缺省值
输入参数	ADCxy_COMMON：通用 ADC 实例
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

28.2.86 函数__LL_ADC_COMMON_INSTANCE

描述了函数__LL_ADC_COMMNON_INSTANCE

表29-115 函数__LL_ADC_COMMON_INSTANCE

函数名	__LL_ADC_COMMON_INSTANCE
函数原形	#define __LL_ADC_COMMON_INSTANCE(__ADCx__)
功能描述	设置通用 ADC

输入参数 1	__ADCx__：ADC 实例
输出参数	无
返回值	通用 ADC1
先决条件	无

28.2.87 函数__LL_ADC_CONVERT_DATA_RESOLUTION

描述了函数__LL_ADC_CONVERT_DATA_RESOLUTION

表29-116 函数 LL_ADC_CONVERT_DATA_RESOLUTION

函数名	__LL_ADC_CONVERT_DATA_RESOLUTION
函数原形	#define __LL_ADC_CONVERT_DATA_RESOLUTION (__DATA__,__ADC_RESOLUTION_CURRENT__,__ADC_RESOLUTION_TARGET__)
功能描述	分辨率转换
输入参数 1	__DATA__: ADC 数据
输入参数 2	__ADC_RESOLUTION_CURRENT__：ADC 的当前分辨率
输入参数 3	__ADC_RESOLUTION_TARGET__：ADC 的目标分辨率
输出参数	无
返回值	ADC 数据
先决条件	无

28.2.88 函数__LL_ADC_CALC_DATA_TO_VOLTAGE

描述了函数__LL_ADC_CALC_DATA_TO_VOLTAGE

表29-117 函数__LL_ADC_CALC_DATA_TO_VOLTAGE

函数名	__LL_ADC_CALC_DATA_TO_VOLTAGE
函数原形	#define __LL_ADC_CALC_DATA_TO_VOLTAGE (__VREFANANLOG_VOLTAGE__,__ADC_DATA__,__ADC_RESOLUTION__)
功能描述	计算电压值
输入参数 1	__VREFANALOG_VOLTAGE__：参考电压的值
输入参数 2	__ADC_DATA__：ADC 的数据
输入参数 3	__ADC_RESOLUTION__：ADC 分辨率
输出参数	无
返回值	电压值
先决条件	无

28.2.89 函数__LL_ADC_CALC_VREFANALOG_VOLTAGE

描述了函数__LL_ADC_CALC_VREFANALOG_VOLTAGE

表29-118 函数__LL_ADC_CALC_VREFANALOG_VOLTAGE

函数名	__LL_ADC_CALC_VREFANALOG_VOLTAGE

函数原形	#define __LL_ADC_CALC_VREFANALOG_VOLTAGE (__VREFINT_ADC_DATA__, __ADC_RESOLUTION__)
功能描述	内部参考电压换算 VCC
输入参数 1	__VREFINT_ADC_DATA__：内部参考电压的数据
输入参数 2	__ADC_RESOLUTION__：ADC 分辨率
输出参数	无
返回值	VCC
先决条件	无

28.2.90 函数__LL_ADC_CALC_TEMPERATURE

描述了函数__LL_ADC_CALC_TEMPERATURE

表29-119 函数__LL_ADC_CALC_TEMPERATURE

函数名	__LL_ADC_CALC_TEMPERATURE
函数原形	#define __LL_ADC_CALC_TEMPERATURE (__VREFANALOG_VOLTAGE__, __TEMPSENSOR_ADC_DATA,__ADC_RESOLUTION__)
功能描述	计算温度值
输入参数 1	__VREFANALOG_VOLTAGE __：参考电压
输入参数 2	__TEMPSENSOR_ADC_DATA__：ADC 的 Tempsensor 数据
输入参数 3	__ADC_RESOLUTION__：ADC 分辨率
输出参数	无
返回值	温度值
先决条件	无

28.2.91 函数__LL_ADC_CALC_TEMPERATURE_TYP_PARAMS

描述了函数__LL_ADC_CALC_TEMPERATURE_TYP_PARAMS

表29-120 函数__LL_ADC_CALC_TEMPERATURE_TYP_PARAMS

函数名	__LL_ADC_CALC_TEMPERATURE_TYP_PARAMS
函数原形	#define LL_ADC_CALC_TEMPERATURE_TYP_PARAMS (__TEMPSENSOR_TYP_AVGSLOPE__, __TEMPSENSOR_TYP_CALX_V_, TEMPSENSOR_CALX_TEMP__, __VREFANLOG_VOLTAGE__, __TEMPSENSOR_ADC_DATA__, __ADC_RESOLUTION__)
功能描述	根据 Tempsensor 参数计算温度值
输入参数 1	__TEMPSENSOR_TYP_AVGSLOPE__：Tempsensor 温度斜率
输入参数 2	__TEMPSENSOR_TYP_CALX_V__： Tempsensor 校准的电压值
输入参数 3	__TEMPSENSOR_CALX_TEMP__：Tempsensor 校准的温度值
输入参数 4	__VREFANLOG_VOLTAGE__：参考电压
输入参数 5	__TEMPSENSOR_ADC_DATA__：ADC Tempsensor 数据
输入参数 6	__ADC_RESOLUTION__：ADC 分辨率

输出参数	无
返回值	温度值
先决条件	无

29 LL BUS 通用驱动程序（BUS）

BUS 包含对 AHB、APB 总线时钟的配置。

29.1 BUS 固件库函数

表30-1 BUS 固件库函数说明

函数名	描述
LL_AHB1_GRP1_EnableClock	使能AHB1 外设时钟
LL_AHB1_GRP1_IsEnabledClock	检查AHB1 外设时钟是否使能
LL_AHB1_GRP1_DisableClock	禁用AHB1 外设时钟
LL_AHB1_GRP1_ForceReset	强制AHB1 外设复位
LL_AHB1_GRP1_ReleaseReset	释放AHB1 外设复位
LL_APB1_GRP1_EnableClock	使能APB1 GRP1 外设时钟
LL_APB1_GRP1_IsEnabledClock	检查APB1 GRP1 外设时钟是否使能
LL_APB1_GRP1_DisableClock	禁用APB1 GRP1 外设时钟
LL_APB1_GRP1_ForceReset	强制APB1 GRP1 外设复位
LL_APB1_GRP1_ReleaseReset	释放APB1 GRP1 外设复位
LL_APB1_GRP2_EnableClock	使能APB1 GRP2 外设时钟
LL_APB1_GRP2_IsEnabledClock	检查APB1 GRP2 外设时钟是否使能
LL_APB1_GRP2_DisableClock	禁用APB1 GRP2 外设时钟
LL_APB1_GRP2_ForceReset	强制APB1 GRP2 外设复位
LL_APB1_GRP2_ReleaseReset	释放APB1 GRP2 外设复位
LL_IOP_GRP1_EnableClock	使能 GPIO 外设时钟
LL_IOP_GRP1_IsEnabledClock	检查 GPIO 外设时钟是否使能
LL_IOP_GRP1_DisableClock	禁用 GPIO 外设时钟
LL_IOP_GRP1_ForceReset	强制 GPIO 外设复位
LL_IOP_GRP1_ReleaseReset	释放 GPIO 外设复位

29.1.1 函数 LL_AHB1_GRP1_EnableClock

描述了函数 LL_AHB1_GRP1_EnableClock

表30-2 函数 LL_AHB1_GRP1_EnableClock

函数名	LL_AHB1_GRP1_EnableClock
函数原形	__STATIC_INLINE void LL_AHB1_GRP1_EnableClock(uint32_t Periphs)

功能描述	使能 AHB1 外设时钟
输入参数	Periphs：外设模块
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表30-3 Periphs 可选参数

参数	描述
LL_AHB1_GRP1_PERIPH_DMA1	DMA1 模块
LL_AHB1_GRP1_PERIPH_FLASH	FLASH 模块
LL_AHB1_GRP1_PERIPH_SRAM	SRAM 模块
LL_AHB1_GRP1_PERIPH_CRC	CRC 模块

29.1.2 函数 LL_AHB1_GRP1_IsEnabledClock

描述了函数 LL_AHB1_GRP1_IsEnabledClock

表30-4 函数 LL_AHB1_GRP1_IsEnabledClock

函数名	LL_AHB1_GRP1_IsEnabledClock
函数原形	__STATIC_INLINE uint32_t LL_AHB1_GRP1_IsEnabledClock(uint32_t Periphs)
功能描述	检查 AHB1 外设时钟是否使能
输入参数	Periphs：外设模块
输出参数	无
返回值	外设时钟状态
先决条件	无

Periphs 可选参数：

表30-5 Periphs 可选参数

参数	描述
LL_AHB1_GRP1_PERIPH_DMA1	DMA1 模块
LL_AHB1_GRP1_PERIPH_FLASH	FLASH 模块
LL_AHB1_GRP1_PERIPH_SRAM	SRAM 模块
LL_AHB1_GRP1_PERIPH_CRC	CRC 模块

29.1.3 函数 LL_AHB1_GRP1_DisableClock

描述了函数 LL_AHB1_GRP1_DisableClock

表30-6 函数 LL_AHB1_GRP1_DisableClock

函数名	LL_AHB1_GRP1_DisableClock

函数原形	__STATIC_INLINE void LL_AHB1_GRP1_DisableClock(uint32_t Periphs)
功能描述	禁用 AHB1 外设时钟
输入参数	Periphs：外设模块
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表30-7 Periphs 可选参数

参数	描述
LL_AHB1_GRP1_PERIPH_DMA1	DMA1 模块
LL_AHB1_GRP1_PERIPH_FLASH	FLASH 模块
LL_AHB1_GRP1_PERIPH_SRAM	SRAM 模块
LL_AHB1_GRP1_PERIPH_CRC	CRC 模块

29.1.4 函数 LL_AHB1_GRP1_ForceReset

描述了函数 LL_AHB1_GRP1_ForceReset

表30-8 函数 LL_AHB1_GRP1_ForceReset

函数名	LL_AHB1_GRP1_ForceReset
函数原形	__STATIC_INLINE void LL_AHB1_GRP1_ForceReset(uint32_t Periphs)
功能描述	强制 AHB1 外设复位
输入参数	Periphs：外设模块
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表30-9 Periphs 可选参数

参数	描述
LL_AHB1_GRP1_PERIPH_ALL	DMA1 和 CRC 模块
LL_AHB1_GRP1_PERIPH_DMA1	DMA1 模块
LL_AHB1_GRP1_PERIPH_CRC	CRC 模块

29.1.5 函数 LL_AHB1_GRP1_ReleaseReset

描述了函数 LL_AHB1_GRP1_ReleaseReset

表30-10 函数 LL_AHB1_GRP1_ReleaseReset

函数名	LL_AHB1_GRP1_ReleaseReset

函数原形	__STATIC_INLINE void LL_AHB1_GRP1_ReleaseReset(uint32_t Periphs)
功能描述	释放 AHB1 外设复位
输入参数	Periphs：外设模块
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表30-11 Periphs 可选参数

参数	描述
LL_AHB1_GRP1_PERIPH_ALL	DMA1 和 CRC 模块
LL_AHB1_GRP1_PERIPH_DMA1	DMA1 模块
LL_AHB1_GRP1_PERIPH_CRC	CRC 模块

29.1.6 函数 LL_APB1_GRP1_EnableClock

描述了函数 LL_APB1_GRP1_EnableClock

表30-12 函数 LL_APB1_GRP1_EnableClock

函数名	LL_APB1_GRP1_EnableClock
函数原形	__STATIC_INLINE void LL_APB1_GRP1_EnableClock(uint32_t Periphs)
功能描述	使能 APB1 GRP1 外设时钟
输入参数	Periphs：外设模块
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表30-13 Periphs 可选参数

参数	描述
LL_APB1_GRP1_PERIPH_TIM3	TIM3 模块
LL_APB1_GRP1_PERIPH_RTC	RTC 模块
LL_APB1_GRP1_PERIPH_WWDG	WWDG 模块
LL_APB1_GRP1_PERIPH_SPI2	SPI2 模块
LL_APB1_GRP1_PERIPH_USART2	USART2 模块
LL_APB1_GRP1_PERIPH_I2C1	I2C1 模块
LL_APB1_GRP1_PERIPH_DBGMCU	DBGMCU 模块
LL_APB1_GRP1_PERIPH_PWR	PWR 模块
LL_APB1_GRP1_PERIPH_LPTIM1	LPTIM1 模块

29.1.7 函数 LL_APB1_GRP1_IsEnabledClock

描述了函数 LL_APB1_GRP1_IsEnabledClock

表30-14 函数 LL_APB1_GRP1_IsEnabledClock

函数名	LL_APB1_GRP1_IsEnabledClock
函数原形	__STATIC_INLINE uint32_t LL_APB1_GRP1_IsEnabledClock(uint32_t Periphs)
功能描述	检查 APB1 GRP1 外设时钟是否使能
输入参数	Periphs：外设模块
输出参数	无
返回值	外设时钟状态
先决条件	无

Periphs 可选参数：

表30-15 Periphs 可选参数

参数	描述
LL_APB1_GRP1_PERIPH_TIM3	TIM3 模块
LL_APB1_GRP1_PERIPH_RTC	RTC 模块
LL_APB1_GRP1_PERIPH_WWDG	WWDG 模块
LL_APB1_GRP1_PERIPH_SPI2	SPI2 模块
LL_APB1_GRP1_PERIPH_USART2	USART2 模块
LL_APB1_GRP1_PERIPH_I2C1	I2C1 模块
LL_APB1_GRP1_PERIPH_DBGMCU	DBGMCU 模块
LL_APB1_GRP1_PERIPH_PWR	PWR 模块
LL_APB1_GRP1_PERIPH_LPTIM1	LPTIM1 模块

29.1.8 函数 LL_APB1_GRP1_DisableClock

描述了函数 LL_APB1_GRP1_DisableClock

表30-16 函数 LL_APB1_GRP1_DisableClock

函数名	LL_APB1_GRP1_DisableClock
函数原形	__STATIC_INLINE void LL_APB1_GRP1_DisableClock(uint32_t Periphs)
功能描述	禁用 APB1 GRP1 外设时钟
输入参数	Periphs：外设模块
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表30-17 Periphs 可选参数

参数	描述
LL_APB1_GRP1_PERIPH_TIM3	TIM3 模块
LL_APB1_GRP1_PERIPH_RTC	RTC 模块
LL_APB1_GRP1_PERIPH_WWDG	WWDG 模块
LL_APB1_GRP1_PERIPH_SPI2	SPI2 模块
LL_APB1_GRP1_PERIPH_USART2	USART2 模块
LL_APB1_GRP1_PERIPH_I2C1	I2C1 模块
LL_APB1_GRP1_PERIPH_DBGMCU	DBGMCU 模块
LL_APB1_GRP1_PERIPH_PWR	PWR 模块
LL_APB1_GRP1_PERIPH_LPTIM1	LPTIM1 模块

29.1.9 函数 LL_APB1_GRP1_ForceReset

描述了函数 LL_APB1_GRP1_ForceReset

表30-18 函数 LL_APB1_GRP1_ForceReset

函数名	LL_APB1_GRP1_ForceReset
函数原形	__STATIC_INLINE void LL_APB1_GRP1_ForceReset(uint32_t Periphs)
功能描述	强制 APB1 GRP1 外设复位
输入参数	Periphs：外设模块
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表30-19 Periphs 可选参数

参数	描述
LL_APB1_GRP1_PERIPH_TIM3	TIM3 模块
LL_APB1_GRP1_PERIPH_RTC	RTC 模块
LL_APB1_GRP1_PERIPH_WWDG	WWDG 模块
LL_APB1_GRP1_PERIPH_SPI2	SPI2 模块
LL_APB1_GRP1_PERIPH_USART2	USART2 模块
LL_APB1_GRP1_PERIPH_I2C1	I2C1 模块
LL_APB1_GRP1_PERIPH_DBGMCU	DBGMCU 模块
LL_APB1_GRP1_PERIPH_PWR	PWR 模块
LL_APB1_GRP1_PERIPH_LPTIM1	LPTIM1 模块

29.1.10 函数 LL_APB1_GRP1_ReleaseReset

描述了函数 LL_APB1_GRP1_ReleaseReset

表30-20 函数 LL_APB1_GRP1_ReleaseReset

函数名	LL_APB1_GRP1_ReleaseReset
函数原形	__STATIC_INLINE void LL_APB1_GRP1_ReleaseReset(uint32_t Periphs)
功能描述	释放 APB1 GRP1 外设复位
输入参数	Periphs：外设模块
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表30-21 Periphs 可选参数

参数	描述
LL_APB1_GRP1_PERIPH_TIM3	TIM3 模块
LL_APB1_GRP1_PERIPH_RTC	RTC 模块
LL_APB1_GRP1_PERIPH_WWDG	WWDG 模块
LL_APB1_GRP1_PERIPH_SPI2	SPI2 模块
LL_APB1_GRP1_PERIPH_USART2	USART2 模块
LL_APB1_GRP1_PERIPH_I2C1	I2C1 模块
LL_APB1_GRP1_PERIPH_DBGMCU	DBGMCU 模块
LL_APB1_GRP1_PERIPH_PWR	PWR 模块
LL_APB1_GRP1_PERIPH_LPTIM1	LPTIM1 模块

29.1.11 函数 LL_APB1_GRP2_EnableClock

描述了函数 LL_APB1_GRP2_EnableClock

表30-22 函数 LL_APB1_GRP2_EnableClock

函数名	LL_APB1_GRP2_EnableClock
函数原形	__STATIC_INLINE void LL_APB1_GRP2_EnableClock(uint32_t Periphs)
功能描述	使能 APB1 GRP2 外设时钟
输入参数	Periphs：外设模块
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表30-23 Periphs 可选参数

参数	描述
LL_APB1_GRP2_PERIPH_SYSCFG	SYSCFG 模块
LL_APB1_GRP2_PERIPH_TIM1	TIM1 模块
LL_APB1_GRP2_PERIPH_SPI1	SPI1 模块
LL_APB1_GRP2_PERIPH_USART1	USART1 模块
LL_APB1_GRP2_PERIPH_TIM14	TIM14 模块
LL_APB1_GRP2_PERIPH_TIM16	TIM16 模块
LL_APB1_GRP2_PERIPH_TIM17	TIM17 模块
LL_APB1_GRP2_PERIPH_ADC1	ADC1 模块
LL_APB1_GRP2_PERIPH_COMP1	COMP1 模块
LL_APB1_GRP2_PERIPH_COMP2	COMP2 模块
LL_APB1_GRP2_PERIPH_LED	LED 模块

29.1.12 函数 LL_APB1_GRP2_IsEnabledClock

描述了函数 LL_APB1_GRP2_IsEnabledClock

表30-24 函数 LL_APB1_GRP2_IsEnabledClock

函数名	LL_APB1_GRP2_IsEnabledClock
函数原形	__STATIC_INLINE uint32_t LL_APB1_GRP2_IsEnabledClock(uint32_t Periphs)
功能描述	检查 APB1 GRP2 外设时钟是否使能
输入参数	Periphs：外设模块
输出参数	无
返回值	外设时钟状态
先决条件	无

Periphs 可选参数：

表30-25 Periphs 可选参数

参数	描述
LL_APB1_GRP2_PERIPH_SYSCFG	SYSCFG 模块
LL_APB1_GRP2_PERIPH_TIM1	TIM1 模块
LL_APB1_GRP2_PERIPH_SPI1	SPI1 模块
LL_APB1_GRP2_PERIPH_USART1	USART1 模块
LL_APB1_GRP2_PERIPH_TIM14	TIM14 模块
LL_APB1_GRP2_PERIPH_TIM16	TIM16 模块
LL_APB1_GRP2_PERIPH_TIM17	TIM17 模块
LL_APB1_GRP2_PERIPH_ADC1	ADC1 模块

LL_APB1_GRP2_PERIPH_COMP1	COMP1 模块
LL_APB1_GRP2_PERIPH_COMP2	COMP2 模块
LL_APB1_GRP2_PERIPH_LED	LED 模块

29.1.13 函数 LL_APB1_GRP2_DisableClock

描述了函数 LL_APB1_GRP2_DisableClock

表30-26 函数 LL_APB1_GRP2_DisableClock

函数名	LL_APB1_GRP2_DisableClock
函数原形	__STATIC_INLINE void LL_APB1_GRP2_DisableClock(uint32_t Periphs)
功能描述	禁用 APB1 GRP2 外设时钟
输入参数	Periphs：外设模块
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表30-27 Periphs 可选参数

参数	描述
LL_APB1_GRP2_PERIPH_SYSCFG	SYSCFG 模块
LL_APB1_GRP2_PERIPH_TIM1	TIM1 模块
LL_APB1_GRP2_PERIPH_SPI1	SPI1 模块
LL_APB1_GRP2_PERIPH_USART1	USART1 模块
LL_APB1_GRP2_PERIPH_TIM14	TIM14 模块
LL_APB1_GRP2_PERIPH_TIM16	TIM16 模块
LL_APB1_GRP2_PERIPH_TIM17	TIM17 模块
LL_APB1_GRP2_PERIPH_ADC1	ADC1 模块
LL_APB1_GRP2_PERIPH_COMP1	COMP1 模块
LL_APB1_GRP2_PERIPH_COMP2	COMP2 模块
LL_APB1_GRP2_PERIPH_LED	LED 模块

29.1.14 函数 LL_APB1_GRP2_ForceReset

描述了函数 LL_APB1_GRP2_ForceReset

表30-28 函数 LL_APB1_GRP2_ForceReset

函数名	LL_APB1_GRP2_ForceReset
函数原形	__STATIC_INLINE void LL_APB1_GRP2_ForceReset(uint32_t Periphs)
功能描述	强制 APB1 GRP2 外设复位
输入参数	Periphs：外设模块

输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表30-29 Periphs 可选参数

参数	描述
LL_APB1_GRP2_PERIPH_SYSCFG	SYSCFG 模块
LL_APB1_GRP2_PERIPH_TIM1	TIM1 模块
LL_APB1_GRP2_PERIPH_SPI1	SPI1 模块
LL_APB1_GRP2_PERIPH_USART1	USART1 模块
LL_APB1_GRP2_PERIPH_TIM14	TIM14 模块
LL_APB1_GRP2_PERIPH_TIM16	TIM16 模块
LL_APB1_GRP2_PERIPH_TIM17	TIM17 模块
LL_APB1_GRP2_PERIPH_ADC1	ADC1 模块
LL_APB1_GRP2_PERIPH_COMP1	COMP1 模块
LL_APB1_GRP2_PERIPH_COMP2	COMP2 模块
LL_APB1_GRP2_PERIPH_LED	LED 模块

29.1.15 函数 LL_APB1_GRP2_ReleaseReset

描述了函数 LL_APB1_GRP2_ReleaseReset

表30-30 函数 LL_APB1_GRP2_ReleaseReset

函数名	LL_APB1_GRP2_ReleaseReset
函数原形	__STATIC_INLINE void LL_APB1_GRP2_ReleaseReset(uint32_t Periphs)
功能描述	释放 APB1 GRP2 外设复位
输入参数	Periphs：外设模块
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表30-31 Periphs 可选参数

参数	描述
LL_APB1_GRP2_PERIPH_SYSCFG	SYSCFG 模块
LL_APB1_GRP2_PERIPH_TIM1	TIM1 模块
LL_APB1_GRP2_PERIPH_SPI1	SPI1 模块
LL_APB1_GRP2_PERIPH_USART1	USART1 模块

LL_APB1_GRP2_PERIPH_TIM14	TIM14 模块
LL_APB1_GRP2_PERIPH_TIM16	TIM16 模块
LL_APB1_GRP2_PERIPH_TIM17	TIM17 模块
LL_APB1_GRP2_PERIPH_ADC1	ADC1 模块
LL_APB1_GRP2_PERIPH_COMP1	COMP1 模块
LL_APB1_GRP2_PERIPH_COMP2	COMP2 模块
LL_APB1_GRP2_PERIPH_LED	LED 模块

29.1.16 函数 LL_IOP_GRP1_EnableClock

描述了函数 LL_IOP_GRP1_EnableClock

表30-32 函数 LL_IOP_GRP1_EnableClock

函数名	LL_IOP_GRP1_EnableClock
函数原形	__STATIC_INLINE void LL_IOP_GRP1_EnableClock(uint32_t Periphs)
功能描述	使能 GPIO 外设时钟
输入参数	Periphs：外设模块
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表30-33 Periphs 可选参数

参数	描述
LL_IOP_GRP1_PERIPH_GPIOA	GPIO 端口 A
LL_IOP_GRP1_PERIPH_GPIOB	GPIO 端口 B
LL_IOP_GRP1_PERIPH_GPIOF	GPIO 端口 F

29.1.17 函数 LL_IOP_GRP1_IsEnabledClock

描述了函数 LL_IOP_GRP1_IsEnabledClock

表30-34 函数 LL_IOP_GRP1_IsEnabledClock

函数名	LL_IOP_GRP1_IsEnabledClock
函数原形	__STATIC_INLINE uint32_t LL_IOP_GRP1_IsEnabledClock(uint32_t Periphs)
功能描述	检查 GPIO 外设时钟是否使能
输入参数	Periphs：外设模块
输出参数	无
返回值	外设时钟状态
先决条件	无

Periphs 可选参数：

表30-35 Periphs 可选参数

参数	描述
LL_IOP_GRP1_PERIPH_GPIOA	GPIO 端口 A
LL_IOP_GRP1_PERIPH_GPIOB	GPIO 端口 B
LL_IOP_GRP1_PERIPH_GPIOF	GPIO 端口 F

29.1.18 函数 LL_IOP_GRP1_DisableClock

描述了函数 LL_IOP_GRP1_DisableClock

表30-36 函数 LL_IOP_GRP1_DisableClock

函数名	LL_IOP_GRP1_DisableClock
函数原形	__STATIC_INLINE void LL_IOP_GRP1_DisableClock(uint32_t Periphs)
功能描述	禁用 GPIO 外设时钟
输入参数	Periphs：外设模块
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表30-37 Periphs 可选参数

参数	描述
LL_IOP_GRP1_PERIPH_GPIOA	GPIO 端口 A
LL_IOP_GRP1_PERIPH_GPIOB	GPIO 端口 B
LL_IOP_GRP1_PERIPH_GPIOF	GPIO 端口 F

29.1.19 函数 LL_IOP_GRP1_ForceReset

描述了函数 LL_IOP_GRP1_ForceReset

表30-38 函数 LL_IOP_GRP1_ForceReset

函数名	LL_IOP_GRP1_ForceReset
函数原形	__STATIC_INLINE void LL_IOP_GRP1_ForceReset(uint32_t Periphs)
功能描述	强制 GPIO 外设复位
输入参数	Periphs：外设模块
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表30-39 Periphs 可选参数

参数	描述
LL_IOP_GRP1_PERIPH_GPIOA	GPIO 端口 A
LL_IOP_GRP1_PERIPH_GPIOB	GPIO 端口 B
LL_IOP_GRP1_PERIPH_GPIOF	GPIO 端口 F

29.1.20 函数 LL_IOP_GRP1_ReleaseReset

描述了函数 LL_IOP_GRP1_ReleaseReset

表30-40 函数 LL_IOP_GRP1_ReleaseReset

函数名	LL_IOP_GRP1_ReleaseReset
函数原形	__STATIC_INLINE void LL_IOP_GRP1_ReleaseReset(uint32_t Periphs)
功能描述	释放 GPIO 外设复位
输入参数	Periphs：外设模块
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表30-41 Periphs 可选参数

参数	描述
LL_IOP_GRP1_PERIPH_GPIOA	GPIO 端口 A
LL_IOP_GRP1_PERIPH_GPIOB	GPIO 端口 B
LL_IOP_GRP1_PERIPH_GPIOF	GPIO 端口 F

30 LL 比较器通用驱动程序（COMP）

芯片内集成了 2 个通用比较器（general purpose comparators）COMP，分别是 COMP1 和 COMP2。这两个模块可以作为单独的模块，也可以与TIM 组合在一起使用。

比较器可以实现如下功能：

- 被模拟信号触发，产生低功耗模式唤醒功能
- 模拟信号调节
- 当与来自TIM 的 PWM 输出连接时，构成逐周期电流控制环路。

30.1 COMP 固件驱动寄存器结构

30.1.1 LL_COMP_InitTypeDef

LL_COMP_InitTypeDef，定义于文件“air001xx_ll_comp.h”如下：

```
typedef struct
{
uint32_t PowerMode;
uint32_t InputPlus;
uint32_t InputMinus;
uint32_t InputHysteresis;
uint32_t OutputPolarity;
uint32_t DigitalFilter;
}LL_COMP_InitTypeDef;
```

字段说明：

表31-1 LL_COMP_InitTypeDef 字段说明

字段	描述
PowerMode	配置COMP 功耗模式
InputPlus	配置COMP Plus 端输入
InputMinus	配置COMP Minus 端输入
InputHysteresis	配置COMP 迟滞
OutputPolarity	配置COMP 输出极性
DigitalFilter	配置COMP 数字滤波

参数说明：

PowerMode 可选参数：

表31-2 PowerMode 可选参数

参数	描述
LL_COMP_POWERMODE_HIGHSPEED	高速模式
LL_COMP_POWERMODE_MEDIUMSPEED	中等速度模式

InputPlus 可选参数:

表31-3 InpuPlus 可选参数

参数	描述
LL_COMP_INPUT_PLUS_IO1	Plus 端输入为 IO1（COMP1：PB8，COMP2：PB4）
LL_COMP_INPUT_PLUS_IO2	Plus 端输入为 IO2（COMP1：PB2，COMP2：PB6）
LL_COMP_INPUT_PLUS_IO3	Plus 端输入为 IO3（COMP1：PA1，COMP2：PA3）
LL_COMP_INPUT_PLUS_IO4	Plus 端输入为 IO4（COMP1：无，COMP2：PF3）

InputMinus 可选参数:

表31-4 InputMinus 可选参数

参数	描述
COMP_INPUT_MINUS_1_4VREFINT	Minus 端输入为 1/4VREFINT
COMP_INPUT_MINUS_1_2VREFINT	Minus 端输入为 1/2VREFINT
COMP_INPUT_MINUS_3_4VREFINT	Minus 端输入为 3/4VREFINT
COMP_INPUT_MINUS_VREFINT	Minus 端输入为 VREFINT
COMP_INPUT_MINUS_VCC	Minus 端输入为 VCC
COMP_INPUT_MINUS_TS	Minus 端输入为温度传感器
COMP_INPUT_MINUS_IO1	Minus 端输入为 IO1（COMP1：PB1，COMP2：PB3）
COMP_INPUT_MINUS_IO2	Minus 端输入为 IO2（COMP1：无，COMP2：PB7）
COMP_INPUT_MINUS_IO3	Minus 端输入为 IO3（COMP1：PA0，COMP2：PA2）

InputHysteresis 可选参数:

表31-5 InputHysteresis 可选参数

参数	描述
LL_COMP_HYSTERESIS_DISABLE	迟滞关闭
LL_COMP_HYSTERESIS_ENABLE	迟滞打开

OutputPolarity 可选参数:

表31-6 OutputPolarity 可选参数

参数	描述
LL_COMP_OUTPUTPOL_NONINVERTED	极性不反向
LL_COMP_OUTPUTPOL_INVERTED	极性反向

30.2 COMP 固件库函数

表31-7 COMP 固件库函数说明

函数名	描述
__LL_COMP_COMMON_INSTANCE	设置COMP 通用部分
LL_COMP_SetCommonWindowMode	设置COMP 窗口模式
LL_COMP_GetCommonWindowMode	获取COMP 窗口模式
LL_COMP_SetPowerMode	设置COMP 功耗模式
LL_COMP_GetPowerMode	获取COMP 功耗模式
LL_COMP_ConfigInputs	设置COMP Minus 端和 Plus 端输入
LL_COMP_SetInputPlus	设置COMP Plus 端输入
LL_COMP_GetInputPlus	获取COMP Plus 端输入
LL_COMP_SetInputMinus	设置COMP Minus 端输入
LL_COMP_GetInputMinus	获取COMP Minus 端输入
LL_COMP_SetInputHysteresis	设置COMP 迟滞
LL_COMP_GetInputHysteresis	获取COMP 迟滞
LL_COMP_EnableScaler	使能COMP Scaler
LL_COMP_DisableScaler	禁用COMP Scaler
LL_COMP_IsEnabledScaler	检查COMP Scaler 是否使能
LL_COMP_Enable	使能COMP
LL_COMP_Disable	禁用COMP
LL_COMP_IsEnabled	检查COMP 是否使能
LL_COMP_Lock	锁定COMP
LL_COMP_IsLocked	检查COMP 锁定状态
LL_COMP_ReadOuputLevel	读取COMP 的输出状态
LL_COMP_EnableDigitalFilter	使能COMP 数字滤波
LL_COMP_DisableDigitalFilter	禁用COMP 数字滤波
LL_COMP_IsEnabledDigitalFilter	COMP 数字滤波的使能状态
LL_COMP_SetDigitalFilter	设置COMP 的数字滤波值
LL_COMP_GetDigitalFilter	获取COMP 的数字滤波值
LL_COMP_Init	COMP 初始化
LL_COMP_DeInit	将 COMP 配置重设为缺省值

30.2.1 函数__LL_COMP_COMMON_INSTANCE

描述了函数__LL_COMP_COMMON_INSTANCE

表31-8 函数__LL_COMP_COMMON_INSTANCE

函数名	__LL_COMP_COMMON_INSTANCE
函数原形	#define __LL_COMP_COMMON_INSTANCE (__COMPx__)
功能描述	设置通用 COMP
输入参数	COMPx：COMP 实例
输出参数	无
返回值	COMP12_COMMON：通用 COMP
先决条件	无

30.2.2 函数 LL_COMP_SetCommonWindowMode

描述了函数 LL_COMP_SetCommonWindowMode

表31-9 函数 LL_COMP_SetCommonWindowMode

函数名	LL_COMP_SetCommonWindowMode
函数原形	__STATIC_INLINE void LL_COMP_SetCommonWindowMode (COMP_Common_TypeDef *COMPxy_COMMON,uint32_t WindowMode)
功能描述	设置 COMP 的窗口模式
输入参数 1	COMPxy_COMMON：COMP_COMMON 实例
输入参数 2	WindowMode：窗口模式
输出参数	无
返回值	无
先决条件	无

WindowMode 可选参数:

表31-10 WindowMode 可选参数

参数	描述
LL_COMP_WINDOWMODE_DISABLE	禁用窗口模式
LL_COMP_WINDOWMODE_COMP1_INPUT_PLUS_COMMON	比较器 1 的 Plus 端作为公共端
LL_COMP_WINDOWMODE_COMP2_INPUT_PLUS_COMMON	比较器 2 的 Plus 端作为公共端

30.2.3 函数 LL_COMP_GetCommonWindowMode

描述了函数 LL_COMP_GetCommonWindowMode

表31-11 函数 LL_COMP_GetCommonWindowMode

函数名	LL_COMP_GetCommonWindowMode
函数原形	__STATIC_INLINE uint32_t LL_COMP_GetCommonWindowMode (COMP_Common_TypeDef *COMPxy_COMMON)

功能描述	获取 COMP 的窗口模式
输入参数	COMPxy_COMMON：COMP_COMMON 实例
输出参数	无
返回值	WindowMode：窗口模式
先决条件	无

30.2.4 函数 LL_COMP_SetPowerMode

描述了函数 LL_COMP_SetPowerMode

表31-12 函数 LL_COMP_SetPowerMode

函数名	LL_COMP_SetPowerMode
函数原形	__STATIC_INLINE void LL_COMP_SetPowerMode(COMP_TypeDef *COMPx,uint32_t PowerMode)
功能描述	设置 COMP 的功耗模式
输入参数 1	COMPx：COMP 实例
输入参数 2	PowerMode：功耗模式
输出参数	无
返回值	无
先决条件	无

PowerMode 可选参数:

表31-13 PowerMode 可选参数

参数	描述
LL_COMP_POWERMODE_HIGHSPEED	高速模式
LL_COMP_POWERMODE_MEDIUMSPEED	中等速度模式

30.2.5 函数 LL_COMP_GetPowerMode

描述了函数 LL_COMP_GetPowerMode

表31-14 函数 LL_COMP_GetPowerMode

函数名	LL_COMP_GetPowerMode
函数原形	__STATIC_INLINE uint32_t LL_COMP_GetPowerMode(COMP_TypeDef *COMPx)
功能描述	获取 COMP 的功耗模式
输入参数	COMPx：COMP 实例
输出参数	无
返回值	PowerMode：功耗模式
先决条件	无

30.2.6 函数 LL_COMP_ConfigInputs

描述了函数 LL_COMP_ConfigInputs

表31-15 函数 LL_COMP_ConfigInputs

函数名	LL_COMP_ConfigInputs
函数原形	__STATIC_INLINE void LL_COMP_ConfigInputs(COMP_TypeDef *COMPx,uint32_t InputMinus,uint32_t InputPlus)
功能描述	设置 COMP 的 Minus 端和 Plus 端输入
输入参数 1	COMPx：COMP 实例
输入参数 2	InputMinus：Minus 端输入
输入参数 3	InputPlus：Plus 端输入
输出参数	无
返回值	无
先决条件	无

InputMinus 可选参数:

表31-16 InputMinus 可选参数

参数	描述
COMP_INPUT_MINUS_1_4VREFINT	Minus 端输入为 1/4VREFINT
COMP_INPUT_MINUS_1_2VREFINT	Minus 端输入为 1/2VREFINT
COMP_INPUT_MINUS_3_4VREFINT	Minus 端输入为 3/4VREFINT
COMP_INPUT_MINUS_VREFINT	Minus 端输入为 VREFINT
COMP_INPUT_MINUS_VCC	Minus 端输入为 VCC
COMP_INPUT_MINUS_TS	Minus 端输入为 TS
COMP_INPUT_MINUS_IO1	Minus 端输入为 IO1（COMP1：PB1，COMP2：PB3）
COMP_INPUT_MINUS_IO2	Minus 端输入为 IO2（COMP1：无，COMP2：PB7）
COMP_INPUT_MINUS_IO3	Minus 端输入为 IO3（COMP1：PA0，COMP2：PA2）

InputPlus 可选参数:

表31-17 InputPlus 可选参数

参数	描述
LL_COMP_INPUT_PLUS_IO1	Plus 端输入为 IO1（COMP1：PB8，COMP2：PB4）
LL_COMP_INPUT_PLUS_IO2	Plus 端输入为 IO2（COMP1：PB2，COMP2：PB6）
LL_COMP_INPUT_PLUS_IO3	Plus 端输入为 IO3（COMP1：PA1，COMP2：PA3）
LL_COMP_INPUT_PLUS_IO4	Plus 端输入为 IO4（COMP1：无，COMP2：PF3）

30.2.7 函数 LL_COMP_SetInputPlus

描述了函数 LL_COMP_SetInputPlus

表31-18 函数 LL_COMP_SetInputPlus

函数名	LL_COMP_SetInputPlus
函数原形	__STATIC_INLINE void LL_COMP_SetInputPlus(COMP_TypeDef *COMPx,uint32_t InputPlus)
功能描述	设置 COMP 的 Plus 端输入
输入参数 1	COMPx：COMP 实例
输入参数 2	InputPlus：Plus 端输入
输出参数	无
返回值	无
先决条件	无

InputPlus 可选参数：

表31-19 InputPlus 可选参数

参数	描述
LL_COMP_INPUT_PLUS_IO1	Plus 端输入为 IO1（COMP1：PB8，COMP2：PB4）
LL_COMP_INPUT_PLUS_IO2	Plus 端输入为 IO2（COMP1：PB2，COMP2：PB6）
LL_COMP_INPUT_PLUS_IO3	Plus 端输入为 IO3（COMP1：PA1，COMP2：PA3）
LL_COMP_INPUT_PLUS_IO4	Plus 端输入为 IO4（COMP1：无，COMP2：PF3）

30.2.8 函数 LL_COMP_GetInputPlus

描述了函数 LL_COMP_GetInputPlus

表31-20 函数 LL_COMP_GetInputPlus

函数名	LL_COMP_GetInputPlus
函数原形	__STATIC_INLINE uint32_t LL_COMP_GetInputPlus (COMP_TypeDef *COMPx)
功能描述	获取 COMP 的 Plus 端输入
输入参数 1	COMPx：COMP 实例
输出参数	无
返回值	InputPlus：Plus 端输入
先决条件	无

30.2.9 函数 LL_COMP_SetInputMinus

描述了函数 LL_COMP_SetInputMinus

表31-21 函数 LL_COMP_SetInputMinus

函数名	LL_COMP_SetInputMinus
函数原形	__STATIC_INLINE void LL_COMP_SetInputMinus(COMP_TypeDef *COMPx,uint32_t InputMinus)
功能描述	设置 COMP 的 Minus 端输入
输入参数 1	COMPx：COMP 实例

输入参数 2	InputMinus：Minus 端输入
输出参数	无
返回值	无
先决条件	无

InputMinus 可选参数:

表31-22 InputMinus 可选参数

参数	描述
COMP_INPUT_MINUS_1_4VREFINT	Minus 端输入为 1/4VREFINT
COMP_INPUT_MINUS_1_2VREFINT	Minus 端输入为 1/2VREFINT
COMP_INPUT_MINUS_3_4VREFINT	Minus 端输入为 3/4VREFINT
COMP_INPUT_MINUS_VREFINT	Minus 端输入为 VREFINT
COMP_INPUT_MINUS_VCC	Minus 端输入为 VCC
COMP_INPUT_MINUS_TS	Minus 端输入为 TS
COMP_INPUT_MINUS_IO1	Minus 端输入为 IO1（COMP1：PB1，COMP2：PB3）
COMP_INPUT_MINUS_IO2	Minus 端输入为 IO2（COMP1：无，COMP2：PB7）
COMP_INPUT_MINUS_IO3	Minus 端输入为 IO3（COMP1：PA0，COMP2：PA2）

30.2.10 函数 LL_COMP_GetInputMinus

描述了函数 LL_COMP_GetInputMinus

表31-23 函数 LL_COMP_GetInputMinus

函数名	LL_COMP_GetInputMinus
函数原形	__STATIC_INLINE uint32_t LL_COMP_GetInputMinus(COMP_TypeDef *COMPx)
功能描述	获取 COMP 的 Minus 端输入
输入参数	COMPx：COMP 实例
输出参数	无
返回值	InputMinus：Minus 端输入
先决条件	无

30.2.11 函数 LL_COMP_SetInputHysteresis

描述了函数 LL_COMP_SetInputHysteresis

表31-24 函数 LL_COMP_SetInputHysteresis

函数名	LL_COMP_SetInputHysteresis
函数原形	__STATIC_INLINE void LL_COMP_SetInputHysteresis(COMP_TypeDef *COMPx,uint32_t InputHysteresis)
功能描述	设置 COMP 的迟滞功能
输入参数 1	COMPx：COMP 实例

输入参数 2	InputHysteresis：迟滞功能
输出参数	无
返回值	无
先决条件	无

InputHysteresis 可选参数:

表31-25 InputHysteresis 可选参数

参数	描述
LL_COMP_HYSTERESIS_DISABLE	迟滞功能关闭
LL_COMP_HYSTERESIS_ENABLE	迟滞功能打开

30.2.12 函数 LL_COMP_GetInputHysteresis

描述了函数 LL_COMP_GetInputHysteresis

表31-26 函数 LL_COMP_GetInputHysteresis

函数名	LL_COMP_GetInputHysteresis
函数原形	__STATIC_INLINE uint32_t LL_COMP_GetInputHysteresis (COMP_TypeDef *COMPx)
功能描述	获取 COMP 的迟滞功能状态
输入参数	COMPx：COMP 实例
输出参数	无
返回值	Hysteresis：迟滞功能
先决条件	无

30.2.13 函数 LL_COMP_SetOutputPolarity

描述了函数 LL_COMP_SetOutputPolarity

表31-27 函数 LL_COMP_SetOutputPolarity

函数名	LL_COMP_SetOutputPolarity
函数原形	__STATIC_INLINE void LL_COMP_SetOutputPolarity(COMP_TypeDef *COMPx,uint32_t OutputPolarity)
功能描述	设置 COMP 的输出极性
输入参数 1	COMPx：COMP 实例
输入参数 2	OutputPolarity：输出极性
输出参数	无
返回值	无
先决条件	无

OutputPolarity 可选参数:

表31-28 OutputPolarity 可选参数

参数	描述
LL_COMP_OUTPUTPOL_NONINVERTED	极性不反向
LL_COMP_OUTPUTPOL_INVERTED	极性反向

30.2.14 函数 LL_COMP_GetOutputPolarity

描述了函数 LL_COMP_GetOutputPolarity

表31-29 函数 LL_COMP_GetOutputPolarity

函数名	LL_COMP_GetOutputPolarity
函数原形	__STATIC_INLINE uint32_t LL_COMP_GetOutputPolarity (COMP_TypeDef *COMPx)
功能描述	获取 COMP 的极性状态
输入参数	COMPx：COMP 实例
输出参数	无
返回值	OutputPolarity：输出极性
先决条件	无

30.2.15 函数 LL_COMP_EnableScaler

描述了函数 LL_COMP_EnableScaler

表31-30 函数 LL_COMP_EnableScaler

函数名	LL_COMP_EnableScaler
函数原形	__STATIC_INLINE void LL_COMP_EnableScaler(COMP_TypeDef *COMPx)
功能描述	使能 Scaler
输入参数	COMPx：COMP 实例
输出参数	无
返回值	无
先决条件	无

30.2.16 函数 LL_COMP_DisableScaler

描述了函数 LL_COM_DisableScaler

表31-31 函数 LL_COMP_DisableScaler

函数名	LL_COMP_DisableScaler
函数原形	__STATIC_INLINE uint32_t LL_COMP_DisableScaler (COMP_TypeDef *COMPx)
功能描述	禁用 Scaler
输入参数	COMPx：COMP 实例
输出参数	无
返回值	无

先决条件	无

30.2.17 函数 LL_COMP_IsEnabledScaler

描述了函数 LL_COM_IsEnabledScaler

表31-32 函数 LL_COMP_IsEnabledScaler

函数名	LL_COMP_IsEnabledScaler
函数原形	__STATIC_INLINE uint32_t LL_COMP_IsEnabledScaler(COMP_TypeDef *COMPx)
功能描述	检查 COMP Scaler 是否使能
输入参数	COMPx：COMP 实例
输出参数	无
返回值	无
先决条件	无

30.2.18 函数 LL_COMP_Enable

描述了函数 LL_COMP_Enable

表31-33 函数 LL_COMP_Enable

函数名	LL_COMP_Enable
函数原形	__STATIC_INLINE void LL_COMP_Enable(COMP_TypeDef *COMPx)
功能描述	使能 COMP
输入参数	COMPx：COMP 实例
输出参数	无
返回值	无
先决条件	无

30.2.19 函数 LL_COMP_Disable

描述了函数 LL_COM_Disable

表31-34 函数 LL_COMP_Disable

函数名	LL_COMP_Disable
函数原形	__STATIC_INLINE void LL_COMP_Disable (COMP_TypeDef *COMPx)
功能描述	禁用 COMP
输入参数	COMPx：COMP 实例
输出参数	无
返回值	无
先决条件	无

30.2.20 函数 LL_COMP_IsEnabled

描述了函数 LL_COM_IsEnabled

表31-35 函数 LL_COMP_IsEnabled

函数名	LL_COMP_IsEnabled
函数原形	__STATIC_INLINE uint32_t LL_COMP_IsEnabled (COMP_TypeDef *COMPx)
功能描述	检查 COMP 是否使能
输入参数	COMPx：COMP 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

30.2.21 函数 LL_COMP_Lock

描述了函数 LL_COM_Lock

表31-36 函数 LL_COMP_Lock

函数名	LL_COMP_Lock
函数原形	__STATIC_INLINE void LL_COMP_Lock (COMP_TypeDef *COMPx)
功能描述	锁定 COMP
输入参数	COMPx：COMP 实例
输出参数	无
返回值	无
先决条件	无

30.2.22 函数 LL_COMP_IsLocked

描述了函数 LL_COM_IsLocked

表31-37 函数 LL_COMP_IsLocked

函数名	LL_COMP_IsLocked
函数原形	__STATIC_INLINE uint32_t LL_COMP_IsLocked (COMP_TypeDef *COMPx)
功能描述	检查 COMP 锁定状态
输入参数	COMPx：COMP 实例
输出参数	无
返回值	0：未锁定 1：锁定
先决条件	无

30.2.23 函数 LL_COMP_ReadOutputLevel

描述了函数 LL_COM_ReadOutputLevel

表31-38 函数 LL_COMP_ReadOutputLevel

函数名	LL_COMP_ ReadOutputLevel
函数原形	__STATIC_INLINE uint32_t LL_COMP_ReadOutputLevel(COMP_TypeDef *COMPx)
功能描述	COMP 输出状态
输入参数	COMPx：COMP 实例
输出参数	无
返回值	0：输出低电平 1：输出高电平
先决条件	无

30.2.24 函数 LL_COMP_EnableDigitalFilter

描述了函数 LL_COMP_EnableDigitalFilter

表31-39 函数 LL_COMP_EnableDigitalFilter

函数名	LL_COMP_EnableDigitalFilter
函数原形	__STATIC_INLINE void LL_COMP_EnableDigitalFilter(COMP_TypeDef *COMPx)
功能描述	使能 COMP 数字滤波
输入参数	COMPx：COMP 实例
输出参数	无
返回值	无
先决条件	无

30.2.25 函数 LL_COMP_DisableDigitalFilter

描述了函数 LL_COM_DisableDigitalFilter

表31-40 函数 LL_COMP_DisableDigitalFilter

函数名	LL_COMP_DisableDigitalFilter
函数原形	__STATIC_INLINE void LL_COMP_DisableDigitalFilter (COMP_TypeDef *COMPx)
功能描述	禁用 COMP 数字滤波
输入参数	COMPx：COMP 实例
输出参数	无
返回值	无
先决条件	无

30.2.26 函数 LL_COMP_IsEnabledDigitalFilter

描述了函数 LL_COM_IsEnabledDigitalFilter

表31-41 函数 LL_COMP_IsEnabledDigitalFilter

函数名	LL_COMP_IsEnabledDigitalFilter
函数原形	__STATIC_INLINE uint32_t LL_COMP_IsEnabledDigitalFilter(COMP_TypeDef *COMPx)

功能描述	COMP 数字滤波使能状态
输入参数	COMPx：COMP 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

30.2.27 函数 LL_COMP_SetDigitalFilter

描述了函数 LL_COMP_SetDigitalFilter

表31-42 函数 LL_COMP_SetDigitalFilter

函数名	LL_COMP_SetDigitalFilter
函数原形	__STATIC_INLINE void LL_COMP_SetDigitalFilter(COMP_TypeDef *COMPx,uint32_t FLTCNTValue)
功能描述	设置 COMP 的数字滤波值
输入参数 1	COMPx：COMP 实例
输入参数 2	FLTCNTVlaue：数字滤波值
输出参数	无
返回值	无
先决条件	无

30.2.28 函数 LL_COMP_GetDigitalFilter

描述了函数 LL_ADC_GetDigitalFilter

表31-43 函数 LL_ADC_GetDigitalFilter

函数名	LL_COMP_GetDigitalFilter
函数原形	__STATIC_INLINE uint32_t LL_COMP_GetDigitalFilter (COMP_TypeDef *COMPx)
功能描述	获取 COMP 数字滤波值
输入参数	COMPx：COMP 实例
输出参数	无
返回值	数字滤波值
先决条件	无

30.2.29 函数 LL_COMP_Init

描述了函数 LL_COMP_Init

表31-44 函数 LL_COMP_Init

函数名	LL_COMP_Init
函数原形	ErrorStatus LL_COMP_Init(COMP_TypeDef *COMPx, LL_COMP_InitTypeDef *COMP_InitStruct)
功能描述	初始化 COMP
输入参数 1	COMPx：COMP 实例

输入参数 2	COMP_InitStruct：指向 COMP_InitTypeDef 的指针
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

30.2.30 函数 LL_COMP_Deinit

描述了函数 LL_COMP_Deinit

表31-45 函数 LL_COMP_Deint

函数名	LL_COMP_Deint
函数原形	ErrorStatus LL_COMP_DeInit(COMP_TypeDef *COMPx)
功能描述	将 COMP 配置重设为缺省值
输入参数	COMPx：COMP 实例
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

31 LL Cortex 通用驱动程序（CORTEX）

Cortex 包含对SysTick 定时器、SCB 系统控制的配置，以及获取CPU ID。

31.1 GPIO 固件库函数

表32-1 GPIO 固件库函数说明

函数名	描述
LL_SYSTICK_IsActiveCounterFlag	检查SysTick 定时器的计数是否减至 0
LL_SYSTICK_SetClkSource	配置SysTick 定时器的时钟源
LL_SYSTICK_GetClkSource	获取SysTick 定时器的时钟源
LL_SYSTICK_EnableIT	使能SysTick 定时器中断
LL_SYSTICK_DisableIT	禁用SysTick 定时器中断
LL_SYSTICK_IsEnabledIT	检查SysTick 定时器中断是否使能
LL_LPM_EnableSleep	配置处理器进入普通休眠模式
LL_LPM_EnableDeepSleep	配置处理器进入深度休眠模式
LL_LPM_EnableSleepOnExit	当退出异常处理并返回程序线程时，使能处理器自动进入休眠模式(WFI)
LL_LPM_DisableSleepOnExit	当退出异常处理并返回程序线程时，禁止处理器自动进入休眠模式(WFI)
LL_LPM_EnableEventOnPend	配置所有事件和中断都能唤醒处理器
LL_LPM_DisableEventOnPend	配置仅已使能事件和中断才能唤醒处理器
LL_CPUID_GetImplementer	获取CPU ID 寄存器的制造者 ID(bit[31:24])
LL_CPUID_GetVariant	获取CPU ID 寄存器的变量区域(bit[23:20])
LL_CPUID_GetArchitecture	获取CPU ID 寄存器的常量区域(bit[19:16])
LL_CPUID_GetParNo	获取CPU ID 寄存器的器件编号(bit[15:4])
LL_CPUID_GetRevision	获取CPU ID 寄存器的修订号(bit[3:0])

31.1.1 函数 LL_SYSTICK_IsActiveCounterFlag

描述了函数 LL_SYSTICK_IsActiveCounterFlag

表32-2 函数 LL_SYSTICK_IsActiveCounterFlag

函数名	LL_SYSTICK_IsActiveCounterFlag
函数原形	__STATIC_INLINE uint32_t LL_SYSTICK_IsActiveCounterFlag(void)
功能描述	检查 SysTick 定时器的计数是否减至 0

输入参数	无
输出参数	无
返回值	COUNTFLAG: 1/0
先决条件	无

31.1.2 函数 LL_SYSTICK_SetClkSource

描述了函数 LL_SYSTICK_SetClkSource

表32-3 函数 LL_SYSTICK_SetClkSource

函数名	LL_SYSTICK_SetClkSource
函数原形	__STATIC_INLINE void LL_SYSTICK_SetClkSource(uint32_t Source)
功能描述	配置 SysTick 定时器的时钟源
输入参数 1	Source：时钟源
输出参数	无
返回值	无
先决条件	无

Source 可选参数:

表32-4 Source 可选参数

参数	描述
LL_SYSTICK_CLKSOURCE_HCLK_DIV8	SysTick 使用参考时钟频率（内核时钟）
LL_SYSTICK_CLKSOURCE_HCLK	SysTick 使用内核时钟

31.1.3 函数 LL_SYSTICK_GetClkSource

描述了函数 LL_SYSTICK_GetClkSource

表32-5 函数 LL_SYSTICK_GetClkSource

函数名	LL_SYSTICK_GetClkSource
函数原形	__STATIC_INLINE uint32_t LL_SYSTICK_GetClkSource(void)
功能描述	获取 SysTick 定时器的时钟源
输入参数	无
输出参数	无
返回值	SysTick 定时器的时钟源
先决条件	无

31.1.4 函数 LL_SYSTICK_EnableIT

描述了函数 LL_SYSTICK_EnableIT

表32-6 函数 LL_SYSTICK_EnableIT

函数名	LL_SYSTICK_EnableIT

函数原形	__STATIC_INLINE void LL_SYSTICK_EnableIT(void)
功能描述	使能 SysTick 定时器中断
输入参数	无
输出参数	无
返回值	无
先决条件	无

31.1.5 函数 LL_SYSTICK_DisableIT

描述了函数 LL_SYSTICK_DisableIT

表32-7 函数 LL_SYSTICK_DisableIT

函数名	LL_SYSTICK_DisableIT
函数原形	__STATIC_INLINE void LL_SYSTICK_DisableIT(void)
功能描述	禁用 SysTick 定时器中断
输入参数	无
输出参数	无
返回值	无
先决条件	无

31.1.6 函数 LL_SYSTICK_IsEnabledIT

描述了函数 LL_SYSTICK_IsEnabledIT

表32-8 函数 LL_SYSTICK_IsEnabledIT

函数名	LL_SYSTICK_IsEnabledIT
函数原形	__STATIC_INLINE uint32_t LL_SYSTICK_IsEnabledIT(void)
功能描述	检查 SysTick 中断是否使能
输入参数	无
输出参数	无
返回值	SysTick 中断使能状态
先决条件	无

31.1.7 函数 LL_LPM_EnableSleep

描述了函数 LL_LPM_EnableSleep

表32-9 函数 LL_LPM_EnableSleep

函数名	LL_LPM_EnableSleep
函数原形	__STATIC_INLINE void LL_LPM_EnableSleep(void)
功能描述	配置处理器进入普通休眠模式
输入参数	无

输出参数	无
返回值	无
先决条件	无

31.1.8 函数 LL_LPM_EnableDeepSleep

描述了函数 LL_LPM_EnableDeepSleep

表32-10 函数 LL_LPM_EnableDeepSleep

函数名	LL_LPM_EnableDeepSleep
函数原形	__STATIC_INLINE void LL_LPM_EnableDeepSleep(void)
功能描述	配置处理器进入深度休眠模式
输入参数	无
输出参数	无
返回值	无
先决条件	无

31.1.9 函数 LL_LPM_EnableSleepOnExit

描述了函数 LL_LPM_EnableSleepOnExit

表32-11 函数 LL_LPM_EnableSleepOnExit

函数名	LL_LPM_EnableSleepOnExit
函数原形	__STATIC_INLINE void LL_LPM_EnableSleepOnExit(void)
功能描述	当退出异常处理并返回程序线程时，使能处理器自动进入休眠模式(WFI)
输入参数	无
输出参数	无
返回值	无
先决条件	无

31.1.10 函数 LL_LPM_DisableSleepOnExit

描述了函数 LL_LPM_DisableSleepOnExit

表32-12 函数 LL_LPM_DisableSleepOnExit

函数名	LL_LPM_DisableSleepOnExit
函数原形	__STATIC_INLINE void LL_LPM_DisableSleepOnExit(void)
功能描述	当退出异常处理并返回程序线程时，禁止处理器自动进入休眠模式(WFI)
输入参数	无
输出参数	无
返回值	无
先决条件	无

31.1.11 函数 LL_LPM_EnableEventOnPend

描述了函数 LL_LPM_EnableEventOnPend

表32-13 函数 LL_LPM_EnableEventOnPend

函数名	LL_LPM_EnableEventOnPend
函数原形	__STATIC_INLINE void LL_LPM_EnableEventOnPend(void)
功能描述	配置所有事件和中断都能唤醒处理器
输入参数	无
输出参数	无
返回值	无

31.1.12 函数 LL_LPM_DisableEventOnPend

描述了函数 LL_LPM_DisableEventOnPend

表32-14 函数 LL_LPM_DisableEventOnPend

函数名	LL_LPM_DisableEventOnPend
函数原形	__STATIC_INLINE void LL_LPM_DisableEventOnPend(void)
功能描述	配置仅已使能事件和中断才能唤醒处理器
输入参数	无
输出参数	无
返回值	无
先决条件	无

31.1.13 函数 LL_CPUID_GetImplementer

描述了函数 LL_CPUID_GetImplementer

表32-15 函数 LL_CPUID_GetImplementer

函数名	LL_CPUID_GetImplementer
函数原形	__STATIC_INLINE uint32_t LL_CPUID_GetImplementer(void)
功能描述	获取 CPU ID 寄存器的制造者 ID(bit[31:24])
输入参数	无
输出参数	无
返回值	CPU ID 寄存器的制造者 ID
先决条件	无

31.1.14 函数 LL_CPUID_GetVariant

描述了函数 LL_CPUID_GetVariant

表32-16 函数 LL_CPUID_GetVariant

函数名	LL_CPUID_GetVariant
函数原形	__STATIC_INLINE uint32_t LL_CPUID_GetVariant(void)
功能描述	获取 CPU ID 寄存器的变量区域(bit[23:20])
输入参数	无
输出参数	无
返回值	CPU ID 寄存器的变量区域
先决条件	无

31.1.15 函数 LL_CPUID_GetArchitecture

描述了函数 LL_CPUID_GetArchitecture

表32-17 函数 LL_CPUID_GetArchitecture

函数名	LL_CPUID_GetArchitecture
函数原形	__STATIC_INLINE uint32_t LL_CPUID_GetArchitecture(void)
功能描述	获取 CPU ID 寄存器的常量区域(bit[19:16])
输入参数	无
输出参数	无
返回值	CPU ID 寄存器的常量区域
先决条件	无

31.1.16 函数 LL_CPUID_GetParNo

描述了函数 LL_CPUID_GetParNo

表32-18 函数 LL_CPUID_GetParNo

函数名	LL_CPUID_GetParNo
函数原形	__STATIC_INLINE uint32_t LL_CPUID_GetParNo(void)
功能描述	获取 CPU ID 寄存器的器件编号(bit[15:4])
输入参数	无
输出参数	无
返回值	CPU ID 寄存器的器件编号
先决条件	无

31.1.17 函数 LL_CPUID_GetRevision

描述了函数 LL_CPUID_GetRevision

表32-19 函数 LL_CPUID_GetRevision

函数名	LL_CPUID_GetRevision
函数原形	__STATIC_INLINE uint32_t LL_CPUID_GetRevision(void)
功能描述	获取 CPU ID 寄存器的修订号(bit[3:0])

输入参数	无
输出参数	无
返回值	CPU ID 寄存器的修订号
先决条件	无

32 LL 循环冗余校验通用驱动程序（CRC）

根据生成多项式，CRC 计算单元将输入的 32 位数据，运算产生一个 CRC 结果。

32.1 CRC 固件库函数

表33-1 CRC 固件库函数说明

函数名	描述
LL_CRC_DeInit	将 CRC 配置重设为缺省值
LL_CRC_ResetCRCCalculationUnit	重置 CRC 计算单元
LL_CRC_FeedData32	将给定的 32 位数据写入 CRC 计算器
LL_CRC_ReadData32	返回当前 CRC 计算结果
LL_CRC_Read_IDR	返回存储在 IDR 中的数据
LL_CRC_Write_IDR	将数据存储在 IDR 中

32.1.1 函数 LL_CRC_DeInit

描述了函数 LL_CRC_DeInit

表33-2 函数 LL_CRC_DeInit

函数名	LL_CRC_DeInit
函数原形	ErrorStatus LL_CRC_DeInit(CRC_TypeDef *CRCx)
功能描述	将 CRC 配置重设为缺省值
输入参数	CRCx：CRC 实例
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

32.1.2 函数 LL_CRC_ResetCRCCalculationUnit

描述了函数 LL_CRC_ResetCRCCalculationUnit

表33-3 函数 LL_CRC_ResetCRCCalculationUnit

函数名	LL_CRC_ResetCRCCalculationUnit
函数原形	__STATIC_INLINE void LL_CRC_ResetCRCCalculationUnit(CRC_TypeDef *CRCx)
功能描述	重置 CRC 计算单元
输入参数	CRCx：CRC 实例
输出参数	无
返回值	无
先决条件	无

32.1.3 函数 LL_CRC_FeedData32

描述了函数 LL_CRC_FeedData32

表33-4 函数 LL_CRC_FeedData32

函数名	LL_CRC_FeedData32
函数原形	__STATIC_INLINE void LL_CRC_FeedData32(CRC_TypeDef *CRCx, uint32_t InData)
功能描述	将给定的 32 位数据写入 CRC 计算器
输入参数 1	CRCx：CRC 实例
输入参数 2	InData：数据输入
输出参数	无
返回值	无
先决条件	无

32.1.4 函数 LL_CRC_ReadData32

描述了函数 LL_CRC_ReadData32

表33-5 函数 LL_CRC_ReadData32

函数名	LL_CRC_ReadData32
函数原形	__STATIC_INLINE uint32_t LL_CRC_ReadData32(CRC_TypeDef *CRCx)
功能描述	返回当前 CRC 计算结果
输入参数	CRCx：CRC 实例
输出参数	无
返回值	CRC 计算结果
先决条件	无

32.1.5 函数 LL_CRC_Read_IDR

描述了函数 LL_CRC_Read_IDR

表33-6 函数 LL_CRC_Read_IDR

函数名	LL_CRC_Read_IDR
函数原形	__STATIC_INLINE uint32_t LL_CRC_Read_IDR(CRC_TypeDef *CRCx)
功能描述	返回存储在 IDR 中的数据
输入参数	CRCx：CRC 实例
输出参数	无
返回值	IDR 中的数据
先决条件	无

32.1.6 函数 LL_CRC_Write_IDR

描述了函数 LL_CRC_Write_IDR

表33-7 函数 LL_CRC_Write_IDR

函数名	LL_CRC_Write_IDR
函数原形	__STATIC_INLINE void LL_CRC_Write_IDR(CRC_TypeDef *CRCx, uint32_t InData)
功能描述	将数据存储在 IDR 中
输入参数 1	CRCx：CRC 实例
输入参数 2	InData：数据输入
输出参数	无
返回值	无
先决条件	无

33 LL DMA 控制器通用驱动程序（DMA）

直接存储器存取(DMA)用来提供在外设和存储器之间或者存储器和存储器之间的高速数据传输。无须CPU 干预，数据可以通过DMA 快速地移动，节省了CPU 的资源,进行其他操作。

DMA 控制器有 3 条 DMA 通道，每条通道负责管理来自 1 个或者多个外设对存储器访问的请求。DMA控制器包括处理DMA 请求的仲裁器，仲裁器用于处理各个DMA 请求的优先级。

33.1 DMA 固件驱动寄存器结构

33.1.1 LL_DMA_InitTypeDef

LL_DMA_InitTypeDef，定义于文件“air001xx_ll_dma.h”如下：

```
typedef struct
{
uint32_t PeriphOrM2MSrcAddress;
uint32_t MemoryOrM2MDstAddress;
uint32_t Direction;
uint32_t Mode;
uint32_t PeriphOrM2MSrcIncMode;
uint32_t MemoryOrM2MDstIncMode;
uint32_t PeriphOrM2MSrcDataSize;
uint32_t MemoryOrM2MDstDataSize;
uint32_t NbData;
uint32_t Priority;
} LL_DMA_InitTypeDef;
```

字段说明：

表34-1 LL_DMA_InitTypeDef 字段说明

字段	描述
PeriphOrM2MSrcAddress	外设基地址，在存储器到存储器模式下为源基地址
MemoryOrM2MDstAddress	存储器基地址，在存储器到存储器模式下为目标基地址
Direction	传输方向
Mode	传输模式
PeriphOrM2MSrcIncMode	外设地址增量
MemoryOrM2MDstIncMode	存储器地址增量
PeriphOrM2MSrcDataSize	外设数据对齐大小
MemoryOrM2MDstDataSize	存储器数据对齐大小

NbData	需要传输的数据量
Priority	通道优先级

参数说明:

Direction 可选参数:

表34-2 Direction 可选参数

参数	描述
LL_DMA_DIRECTION_PERIPH_TO_MEMORY	从外设到存储器
LL_DMA_DIRECTION_MEMORY_TO_PERIPH	从存储器到外设
LL_DMA_DIRECTION_MEMORY_TO_MEMORY	从存储器到存储器

Mode 可选参数:

表34-3 Mode 可选参数

参数	描述
LL_DMA_MODE_NORMAL	普通模式
LL_DMA_MODE_CIRCULAR	循环模式

PeriphOrM2MSrcIncMode 可选参数:

表34-4 PeriphOrM2MSrcIncMode 可选参数

参数	描述
LL_DMA_PERIPH_INCREMENT	开启外设地址自增
LL_DMA_PERIPH_NOINCREMENT	禁用外设地址自增

MemoryOrM2MDstIncMode 可选参数:

表34-5 MemoryOrM2MDstIncMode 可选参数

参数	描述
LL_DMA_MEMORY_INCREMENT	开启存储器地址自增
LL_DMA_MEMORY_NOINCREMENT	禁用存储器地址自增

PeriphOrM2MSrcDataSize 可选参数:

表34-6 PeriphOrM2MSrcDataSize 可选参数

参数	描述
LL_DMA_PDATAALIGN_BYTE	字节对齐
LL_DMA_PDATAALIGN_HALFWORD	半字对齐
LL_DMA_PDATAALIGN_WORD	字对齐

MemoryOrM2MDstDataSize 可选参数:

表34-7 MemoryOrM2MDstDataSize 可选参数

参数	描述
LL_DMA_MDATAALIGN_BYTE	字节对齐
LL_DMA_MDATAALIGN_HALFWORD	半字对齐
LL_DMA_MDATAALIGN_WORD	字对齐

Priority 可选参数：

表34-8 Priority 可选参数

参数	描述
LL_DMA_PRIORITY_LOW	优先级低
LL_DMA_PRIORITY_MEDIUM	优先级中
LL_DMA_PRIORITY_HIGH	优先级高
LL_DMA_PRIORITY_VERYHIGH	优先级非常高

33.2 DMA 固件库函数

表34-9 DMA 固件库函数说明

函数名	描述
LL_DMA_EnableChannel	使能指定通道
LL_DMA_DisableChannel	禁用指定通道
LL_DMA_IsEnabledChannel	检查通道是否使能
LL_DMA_ConfigTransfer	配置DMA 传输参数
LL_DMA_SetDataTransferDirection	设置数据传输方向
LL_DMA_GetDataTransferDirection	获取数据传输方向
LL_DMA_SetMode	设置传输模式
LL_DMA_GetMode	获取传输模式
LL_DMA_SetPeriphIncMode	设置外设地址增量模式
LL_DMA_GetPeriphIncMode	获取外设地址增量模式
LL_DMA_SetMemoryIncMode	设置存储器地址增量模式
LL_DMA_GetMemoryIncMode	获取存储器地址增量模式
LL_DMA_SetPeriphSize	设置外设数据对齐大小
LL_DMA_GetPeriphSize	获取外设数据对齐大小
LL_DMA_SetMemorySize	设置存储器数据对齐大小
LL_DMA_GetMemorySize	获取存储器数据对齐大小
LL_DMA_SetChannelPriorityLevel	设置指定通道优先级

LL_DMA_GetChannelPriorityLevel	获取指定通道优先级
LL_DMA_SetDataLength	设置发送数据长度
LL_DMA_GetDataLength	获取发送数据长度
LL_DMA_ConfigAddresses	配置传输源地址和目标地址
LL_DMA_SetMemoryAddress	设置存储器地址
LL_DMA_SetPeriphAddress	设置外设地址
LL_DMA_GetMemoryAddress	获取存储器地址
LL_DMA_GetPeriphAddress	获取外设地址
LL_DMA_SetM2MSrcAddress	设置存储器到存储器模式下源地址
LL_DMA_SetM2MDstAddress	设置存储器到存储器模式下的目标地址
LL_DMA_GetM2MSrcAddress	获取存储器到存储器模式下源地址
LL_DMA_GetM2MDstAddress	获取存储器到存储器模式下的目标地址
LL_DMA_IsActiveFlag_GI1	检查 GI1 是否置位
LL_DMA_IsActiveFlag_GI2	检查 GI2 是否置位
LL_DMA_IsActiveFlag_GI3	检查 GI3 是否置位
LL_DMA_IsActiveFlag_TC1	检查TC1 是否置位
LL_DMA_IsActiveFlag_TC2	检查TC2 是否置位
LL_DMA_IsActiveFlag_TC3	检查TC3 是否置位
LL_DMA_IsActiveFlag_HT1	检查HT1 是否置位
LL_DMA_IsActiveFlag_HT2	检查HT2 是否置位
LL_DMA_IsActiveFlag_HT3	检查HT3 是否置位
LL_DMA_IsActiveFlag_TE1	检查TE1 是否置位
LL_DMA_IsActiveFlag_TE2	检查TE2 是否置位
LL_DMA_IsActiveFlag_TE3	检查TE3 是否置位
LL_DMA_ClearFlag_GI1	清除 GI1 标志位
LL_DMA_ClearFlag_GI2	清除 GI2 标志位
LL_DMA_ClearFlag_GI3	清除 GI3 标志位
LL_DMA_ClearFlag_TC1	清除TC1 标志位
LL_DMA_ClearFlag_TC2	清除TC2 标志位
LL_DMA_ClearFlag_TC3	清除TC3 标志位
LL_DMA_ClearFlag_HT1	清除HT1 标志位

LL_DMA_ClearFlag_HT2	清除HT2 标志位
LL_DMA_ClearFlag_HT3	清除HT3 标志位
LL_DMA_ClearFlag_TE1	清除TE1 标志位
LL_DMA_ClearFlag_TE2	清除TE2 标志位
LL_DMA_ClearFlag_TE3	清除TE3 标志位
LL_DMA_EnableIT_TC	使能TC 中断
LL_DMA_EnableIT_HT	使能 HT 中断
LL_DMA_EnableIT_TE	使能TE 中断
LL_DMA_DisableIT_TC	禁用TC 中断
LL_DMA_DisableIT_HT	禁用 HT 中断
LL_DMA_DisableIT_TE	禁用TE 中断
LL_DMA_IsEnabledIT_TC	检查TC 中断是否使能
LL_DMA_IsEnabledIT_HT	检查 HT 中断是否使能
LL_DMA_IsEnabledIT_TE	检查TE 中断是否使能
LL_DMA_Init	初始化 DMA 配置
LL_DMA_DeInit	将 DMA 相关配置重设为缺省值
LL_DMA_StructInit	初始化 DMA 结构体为默认值

33.2.1 函数 LL_DMA_EnableChannel

描述了函数 LL_DMA_EnableChannel

表34-10　函数 LL_DMA_EnableChannel

函数名	LL_DMA_EnableChannel
函数原形	__STATIC_INLINE void LL_DMA_EnableChannel(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	使能指定通道
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数:

表34-11　Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1

LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.2 函数 LL_DMA_DisableChannel

描述了函数 LL_DMA_DisableChannel

表34-12 函数 LL_DMA_DisableChannel

函数名	LL_DMA_DisableChannel
函数原形	__STATIC_INLINE void LL_DMA_DisableChannel(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	禁用指定通道
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表34-13 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.3 函数 LL_DMA_IsEnabledChannel

描述了函数 LL_DMA_IsEnabledChannel

表34-14 函数 LL_DMA_IsEnabledChannel

函数名	LL_DMA_IsEnabledChannel
函数原形	__STATIC_INLINE uint32_t LL_DMA_IsEnabledChannel(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	检查通道是否使能
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

Channel 可选参数：

表34-15 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.4 函数 LL_DMA_ConfigTransfer

描述了函数 LL_DMA_ConfigTransfer

表34-16 函数 LL_DMA_ConfigTransfer

函数名	LL_DMA_ConfigTransfer
函数原形	__STATIC_INLINE void LL_DMA_ConfigTransfer(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t Configuration)
功能描述	配置 DMA 传输参数
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输入参数 3	Configuration：配置参数
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表34-17 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

Configuration 可选参数：

表34-18 Configuration 可选参数

参数	描述
LL_DMA_DIRECTION_PERIPH_TO_MEMORY	传输方向从外设到存储器
LL_DMA_DIRECTION_MEMORY_TO_PERIPH	传输方向从存储器到外设
LL_DMA_DIRECTION_MEMORY_TO_MEMORY	传输方向从存储器到存储器
LL_DMA_MODE_NORMAL	普通模式
LL_DMA_MODE_CIRCULAR	循环模式
LL_DMA_PERIPH_INCREMENT	使能外设地址自增
LL_DMA_PERIPH_NOINCREMENT	禁用外设地址自增
LL_DMA_MEMORY_INCREMENT	使能存储器地址自增

LL_DMA_MEMORY_NOINCREMENT	禁用存储器地址自增
LL_DMA_PDATAALIGN_BYTE	外设数据大小按字节对齐
LL_DMA_PDATAALIGN_HALFWORD	外设数据大小按半字对齐
LL_DMA_PDATAALIGN_WORD	外设数据大小按字对齐
LL_DMA_MDATAALIGN_BYTE	存储器数据大小按字节对齐
LL_DMA_MDATAALIGN_HALFWORD	存储器数据大小按半字对齐
LL_DMA_MDATAALIGN_WORD	存储器数据大小按字对齐
LL_DMA_PRIORITY_LOW	通道优先级低
LL_DMA_PRIORITY_MEDIUM	通道优先级中
LL_DMA_PRIORITY_HIGH	通道优先级高
LL_DMA_PRIORITY_VERYHIGH	通道优先级很高

33.2.5 函数 LL_DMA_SetDataTransferDirection

描述了函数 LL_DMA_SetDataTransferDirection

表34-19 函数 LL_DMA_SetDataTransferDirection

函数名	LL_DMA_SetDataTransferDirection
函数原形	__STATIC_INLINE void LL_DMA_SetDataTransferDirection(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t Direction)
功能描述	设置数据传输方向
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输入参数 3	Direction：传输方向
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表34-20 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

Direction 可选参数：

表34-21 Direction 可选参数

参数	描述
LL_DMA_DIRECTION_PERIPH_TO_MEMORY	从外设到存储器

LL_DMA_DIRECTION_MEMORY_TO_PERIPH	从存储器到外设
LL_DMA_DIRECTION_MEMORY_TO_MEMORY	从存储器到存储器

33.2.6 函数 LL_DMA_GetDataTransferDirection

描述了函数 LL_DMA_GetDataTransferDirection

表34-22 函数 LL_DMA_GetDataTransferDirection

函数名	LL_DMA_GetDataTransferDirection
函数原形	__STATIC_INLINE uint32_t LL_DMA_GetDataTransferDirection(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	获取数据传输方向
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表34-23 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.7 函数 LL_DMA_SetMode

描述了函数 LL_DMA_SetMode

表34-24 函数 LL_DMA_SetMode

函数名	LL_DMA_SetMode
函数原形	__STATIC_INLINE void LL_DMA_SetMode(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t Mode)
功能描述	设置传输模式
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输入参数 3	Mode：模式
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表34-25 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

Mode 可选参数:

表34-26 Mode 可选参数

参数	描述
LL_DMA_MODE_NORMAL	普通模式
LL_DMA_MODE_CIRCULAR	循环模式

33.2.8 函数 LL_DMA_GetMode

描述了函数 LL_DMA_GetMode

表34-27 函数 LL_DMA_GetMode

函数名	LL_DMA_GetMode
函数原形	__STATIC_INLINE uint32_t LL_DMA_GetMode(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	获取传输模式
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数:

表34-28 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.9 函数 LL_DMA_SetPeriphIncMode

描述了函数 LL_DMA_SetPeriphIncMode

表34-29 函数 LL_DMA_SetPeriphIncMode

函数名	LL_DMA_SetPeriphIncMode
函数原形	__STATIC_INLINE void LL_DMA_SetPeriphIncMode(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t PeriphOrM2MSrcIncMode)
功能描述	设置外设地址增量模式

输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输入参数 3	PeriphOrM2MsrcIncMode：增量模式
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表34-30 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

PeriphOrM2MsrcIncMode 可选参数：

表34-31 PeriphOrM2MsrcIncMode 可选参数

参数	描述
LL_DMA_PERIPH_INCREMENT	开启地址自增
LL_DMA_PERIPH_NOINCREMENT	禁用地址自增

33.2.10 函数 LL_DMA_GetPeriphIncMode

描述了函数 LL_DMA_GetPeriphIncMode

表34-32 函数 LL_DMA_GetPeriphIncMode

函数名	LL_DMA_GetPeriphIncMode
函数原形	__STATIC_INLINE uint32_t LL_DMA_GetPeriphIncMode(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	获取外设地址增量模式
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	外设增量模式
先决条件	无

33.2.11 函数 LL_DMA_SetMemoryIncMode

描述了函数 LL_DMA_SetMemoryIncMode

表34-33 函数 LL_DMA_SetMemoryIncMode

函数名	LL_DMA_SetMemoryIncMode
函数原形	__STATIC_INLINE void LL_DMA_SetMemoryIncMode(DMA_TypeDef *DMAx,

	uint32_t Channel, uint32_t MemoryOrM2MDstIncMode)
功能描述	设置存储器地址增量模式
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输入参数 3	MemoryOrM2MdstIncMode：存储器地址增量模式
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表34-34 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

MemoryOrM2MdstIncMode 可选参数：

表34-35 MemoryOrM2MdstIncMode 可选参数

参数	描述
LL_DMA_MEMORY_INCREMENT	使能存储器地址自增
LL_DMA_MEMORY_NOINCREMENT	禁用存储器地址自增

33.2.12 函数 LL_DMA_GetMemoryIncMode

描述了函数 LL_DMA_GetMemoryIncMode

表34-36 函数 LL_DMA_GetMemoryIncMode

函数名	LL_DMA_GetMemoryIncMode
函数原形	__STATIC_INLINE uint32_t LL_DMA_GetMemoryIncMode(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	获取存储器地址增量模式
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	存储器地址增量模式
先决条件	无

33.2.13 函数 LL_DMA_SetPeriphSize

描述了函数 LL_DMA_SetPeriphSize

表34-37 函数 LL_DMA_SetPeriphSize

函数名	LL_DMA_SetPeriphSize
函数原形	__STATIC_INLINE void LL_DMA_SetPeriphSize(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t PeriphOrM2MSrcDataSize)
功能描述	设置外设数据对齐大小
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输入参数 3	PeriphOrM2MsrcDataSize：外设数据对齐大小
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表34-38 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

PeriphOrM2MsrcDataSize 可选参数：

表34-39 PeriphOrM2MsrcDataSize 可选参数

参数	描述
LL_DMA_PDATAALIGN_BYTE	按字节对齐
LL_DMA_PDATAALIGN_HALFWORD	按半字对齐
LL_DMA_PDATAALIGN_WORD	按字对齐

33.2.14 函数 LL_DMA_GetPeriphSize

描述了函数 LL_DMA_GetPeriphSize

表34-40 函数 LL_DMA_GetPeriphSize

函数名	LL_DMA_GetPeriphSize
函数原形	__STATIC_INLINE uint32_t LL_DMA_GetPeriphSize(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	获取外设数据对齐大小
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	外设数据对齐大小
先决条件	无

Channel 可选参数：

表34-41 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.15 函数 LL_DMA_SetMemorySize

描述了函数 LL_DMA_SetMemorySize

表34-42 函数 LL_DMA_SetMemorySize

函数名	LL_DMA_SetMemorySize
函数原形	__STATIC_INLINE void LL_DMA_SetMemorySize(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t MemoryOrM2MDstDataSize)
功能描述	设置存储器数据对齐大小
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输入参数 3	MemoryOrM2MdstDataSize：存储器数据对齐大小
输出参数	无
返回值	无
先决条件	无

Channel 可选参数:

表34-43 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

MemoryOrM2MdstDataSize 可选参数:

表34-44 MemoryOrM2MdstDataSize 可选参数

参数	描述
LL_DMA_MDATAALIGN_BYTE	按字节对齐
LL_DMA_MDATAALIGN_HALFWORD	按半字对齐
LL_DMA_MDATAALIGN_WORD	按字对齐

33.2.16 函数 LL_DMA_GetMemorySize

描述了函数 LL_DMA_GetMemorySize

表34-45 函数 LL_DMA_GetMemorySize

函数名	LL_DMA_GetMemorySize

函数原形	__STATIC_INLINE uint32_t LL_DMA_GetMemorySize(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	获取存储器数据对齐大小
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	存储器数据对齐大小
先决条件	无

Channel 可选参数：

表34-46 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.17 函数 LL_DMA_SetChannelPriorityLevel

描述了函数 LL_DMA_SetChannelPriorityLevel

表34-47 函数 LL_DMA_SetChannelPriorityLevel

函数名	LL_DMA_SetChannelPriorityLevel
函数原形	__STATIC_INLINE void LL_DMA_SetChannelPriorityLevel(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t Priority)
功能描述	设置指定通道优先级
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输入参数 3	Priority：优先级
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表34-48 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

Priority 可选参数：

表34-49 Priority 可选参数

参数	描述
LL_DMA_PRIORITY_LOW	优先级低
LL_DMA_PRIORITY_MEDIUM	优先级中
LL_DMA_PRIORITY_HIGH	优先级高
LL_DMA_PRIORITY_VERYHIGH	优先级最高

33.2.18 函数 LL_DMA_GetChannelPriorityLevel

描述了函数 LL_DMA_GetChannelPriorityLevel

表34-50 函数 LL_DMA_GetChannelPriorityLevel

函数名	LL_DMA_GetChannelPriorityLevel
函数原形	__STATIC_INLINE uint32_t LL_DMA_GetChannelPriorityLevel(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	获取指定通道优先级
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	优先级
先决条件	无

Channel 可选参数:

表34-51 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.19 函数 LL_DMA_SetDataLength

描述了函数 LL_DMA_SetDataLength

表34-52 函数 LL_DMA_SetDataLength

函数名	LL_DMA_SetDataLength
函数原形	__STATIC_INLINE void LL_DMA_SetDataLength(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t NbData)
功能描述	设置发送数据长度
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输入参数 3	NbData：数据长度（0x0000~0xFFFF）
输出参数	无

返回值	无
先决条件	无

Channel 可选参数：

表34-53 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.20 函数 LL_DMA_GetDataLength

描述了函数 LL_DMA_GetDataLength

表34-54 函数 LL_DMA_GetDataLength

函数名	LL_DMA_GetDataLength
函数原形	__STATIC_INLINE uint32_t LL_DMA_GetDataLength(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	获取发送数据长度
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	发送数据长度
先决条件	无

Channel 可选参数：

表34-55 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.21 函数 LL_DMA_ConfigAddresses

描述了函数 LL_DMA_ConfigAddresses

表34-56 函数 LL_DMA_ConfigAddresses

函数名	LL_DMA_ConfigAddresses
函数原形	__STATIC_INLINE void LL_DMA_ConfigAddresses(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t SrcAddress, uint32_t DstAddress, uint32_t Direction)
功能描述	配置传输源地址和目标地址
输入参数 1	DMAx：DMA 实例

输入参数 2	Channel：通道
输入参数 3	SrcAddress：源地址
输入参数 4	DstAddress：目标地址
输入参数 5	Direction：传输方向
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表34-57 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.22 函数 LL_DMA_SetMemoryAddress

描述了函数 LL_DMA_SetMemoryAddress

表34-58 函数 LL_DMA_SetMemoryAddress

函数名	LL_DMA_SetMemoryAddress
函数原形	__STATIC_INLINE void LL_DMA_SetMemoryAddress(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t MemoryAddress)
功能描述	设置存储器地址
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输入参数 3	MemoryAddress：存储器地址
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表34-59 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.23 函数 LL_DMA_SetPeriphAddress

描述了函数 LL_DMA_SetPeriphAddress

表34-60 函数 LL_DMA_SetPeriphAddress

函数名	LL_DMA_SetPeriphAddress
函数原形	__STATIC_INLINE void LL_DMA_SetPeriphAddress(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t PeriphAddress)
功能描述	设置外设地址
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输入参数 3	PeriphAddress：外设地址
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表34-61 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.24 函数 LL_DMA_GetMemoryAddress

描述了函数 LL_DMA_GetMemoryAddress

表34-62 函数 LL_DMA_GetMemoryAddress

函数名	LL_DMA_GetMemoryAddress
函数原形	__STATIC_INLINE uint32_t LL_DMA_GetMemoryAddress(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	获取存储器地址
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	存储器地址
先决条件	无

Channel 可选参数：

表34-63 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.25 函数 LL_DMA_GetPeriphAddress

描述了函数 LL_DMA_GetPeriphAddress

表34-64 函数 LL_DMA_GetPeriphAddress

函数名	LL_DMA_GetPeriphAddress
函数原形	__STATIC_INLINE uint32_t LL_DMA_GetPeriphAddress(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	获取外设地址
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	外设地址
先决条件	无

Channel 可选参数：

表34-65 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.26 函数 LL_DMA_SetM2MSrcAddress

描述了函数 LL_DMA_SetM2MSrcAddress

表34-66 函数 LL_DMA_SetM2MSrcAddress

函数名	LL_DMA_SetM2MSrcAddress
函数原形	__STATIC_INLINE void LL_DMA_SetM2MSrcAddress(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t MemoryAddress)
功能描述	设置存储器到存储器模式下源地址
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输入参数 2	MemoryAddress：存储器源地址
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表34-67 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1

LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.27 函数 LL_DMA_SetM2MDstAddress

描述了函数 LL_DMA_SetM2MDstAddress

表34-68 函数 LL_DMA_SetM2MDstAddress

函数名	LL_DMA_SetM2MDstAddress
函数原形	__STATIC_INLINE void LL_DMA_SetM2MDstAddress(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t MemoryAddress)
功能描述	设置存储器到存储器模式下的目标地址
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输入参数 3	MemoryAddress：存储器目标地址
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表34-69 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.28 函数 LL_DMA_GetM2MSrcAddress

描述了函数 LL_DMA_GetM2MSrcAddress

表34-70 函数 LL_DMA_GetM2MSrcAddress

函数名	LL_DMA_GetM2MSrcAddress
函数原形	__STATIC_INLINE uint32_t LL_DMA_GetM2MSrcAddress(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	获取存储器到存储器模式下源地址
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	存储器源地址
先决条件	无

Channel 可选参数：

表34-71 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.29 函数 LL_DMA_GetM2MDstAddress

描述了函数 LL_DMA_GetM2MDstAddress

表34-72 函数 LL_DMA_GetM2MDstAddress

函数名	LL_DMA_GetM2MDstAddress
函数原形	__STATIC_INLINE uint32_t LL_DMA_GetM2MDstAddress(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	获取存储器到存储器模式下的目标地址
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	存储器目标地址
先决条件	无

Channel 可选参数:

表34-73 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.30 函数 LL_DMA_IsActiveFlag_GI1

描述了函数 LL_DMA_IsActiveFlag_GI1

表34-74 函数 LL_DMA_IsActiveFlag_GI1

函数名	LL_DMA_IsActiveFlag_GI1
函数原形	__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_GI1(DMA_TypeDef *DMAx)
功能描述	检查 GI1 是否置位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

33.2.31 函数 LL_DMA_IsActiveFlag_GI2

描述了函数 LL_DMA_IsActiveFlag_GI2

表34-75 函数 LL_DMA_IsActiveFlag_GI2

函数名	LL_DMA_IsActiveFlag_GI2
函数原形	__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_GI2(DMA_TypeDef *DMAx)
功能描述	检查 GI2 是否置位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

33.2.32 函数 LL_DMA_IsActiveFlag_GI3

描述了函数 LL_DMA_IsActiveFlag_GI3

表34-76 函数 LL_DMA_IsActiveFlag_GI3

函数名	LL_DMA_IsActiveFlag_GI3
函数原形	__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_GI3(DMA_TypeDef *DMAx)
功能描述	检查 GI3 是否置位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

33.2.33 函数 LL_DMA_IsActiveFlag_TC1

描述了函数 LL_DMA_IsActiveFlag_TC1

表34-77 函数 LL_DMA_IsActiveFlag_TC1

函数名	LL_DMA_IsActiveFlag_TC1
函数原形	__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TC1(DMA_TypeDef *DMAx)
功能描述	检查 TC1 是否置位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

33.2.34 函数 LL_DMA_IsActiveFlag_TC2

描述了函数 LL_DMA_IsActiveFlag_TC2

表34-78 函数 LL_DMA_IsActiveFlag_TC2

函数名	LL_DMA_IsActiveFlag_TC2
函数原形	__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TC2(DMA_TypeDef *DMAx)
功能描述	检查 TC2 是否置位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

33.2.35 函数 LL_DMA_IsActiveFlag_TC3

描述了函数 LL_DMA_IsActiveFlag_TC3

表34-79 函数 LL_DMA_IsActiveFlag_TC3

函数名	LL_DMA_IsActiveFlag_TC3
函数原形	__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TC3(DMA_TypeDef *DMAx)
功能描述	检查 TC3 是否置位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

33.2.36 函数 LL_DMA_IsActiveFlag_HT1

描述了函数 LL_DMA_IsActiveFlag_HT1

表34-80 函数 LL_DMA_IsActiveFlag_HT1

函数名	LL_DMA_IsActiveFlag_HT1
函数原形	__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_HT1(DMA_TypeDef *DMAx)
功能描述	检查 HT1 是否置位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

33.2.37 函数 LL_DMA_IsActiveFlag_HT2

描述了函数 LL_DMA_IsActiveFlag_HT2

表34-81 函数 LL_DMA_IsActiveFlag_HT2

函数名	LL_DMA_IsActiveFlag_HT2
函数原形	__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_HT2(DMA_TypeDef *DMAx)
功能描述	检查 HT2 是否置位

输入参数	DMAx：DMA 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

33.2.38 函数 LL_DMA_IsActiveFlag_HT3

描述了函数 LL_DMA_IsActiveFlag_HT3

表34-82 函数 LL_DMA_IsActiveFlag_HT3

函数名	LL_DMA_IsActiveFlag_HT3
函数原形	__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_HT3(DMA_TypeDef *DMAx)
功能描述	检查 HT3 是否置位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

33.2.39 函数 LL_DMA_IsActiveFlag_TE1

描述了函数 LL_DMA_IsActiveFlag_TE1

表34-83 函数 LL_DMA_IsActiveFlag_TE1

函数名	LL_DMA_IsActiveFlag_TE1
函数原形	__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TE1(DMA_TypeDef *DMAx)
功能描述	检查 TE1 是否置位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

33.2.40 函数 LL_DMA_IsActiveFlag_TE2

描述了函数 LL_DMA_IsActiveFlag_TE2

表34-84 函数 LL_DMA_IsActiveFlag_TE2

函数名	LL_DMA_IsActiveFlag_TE2
函数原形	__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TE2(DMA_TypeDef *DMAx)
功能描述	检查 TE2 是否置位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	状态位（1 或 0）

先决条件	无

33.2.41 函数 LL_DMA_IsActiveFlag_TE3

描述了函数 LL_DMA_IsActiveFlag_TE3

表34-85 函数 LL_DMA_IsActiveFlag_TE3

函数名	LL_DMA_IsActiveFlag_TE3
函数原形	__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TE3(DMA_TypeDef *DMAx)
功能描述	检查 TE3 是否置位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

33.2.42 函数 LL_DMA_ClearFlag_GI1

描述了函数 LL_DMA_ClearFlag_GI1

表34-86 函数 LL_DMA_ClearFlag_GI1

函数名	LL_DMA_ClearFlag_GI1
函数原形	__STATIC_INLINE void LL_DMA_ClearFlag_GI1(DMA_TypeDef *DMAx)
功能描述	清除 GI1 标志位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	无
先决条件	无

33.2.43 函数 LL_DMA_ClearFlag_GI2

描述了函数 LL_DMA_ClearFlag_GI2

表34-87 函数 LL_DMA_ClearFlag_GI2

函数名	LL_DMA_ClearFlag_GI2
函数原形	__STATIC_INLINE void LL_DMA_ClearFlag_GI2(DMA_TypeDef *DMAx)
功能描述	清除 GI2 标志位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	无
先决条件	无

33.2.44 函数 LL_DMA_ClearFlag_GI3

描述了函数 LL_DMA_ClearFlag_GI3

表34-88 函数 LL_DMA_ClearFlag_GI3

函数名	LL_DMA_ClearFlag_GI3
函数原形	__STATIC_INLINE void LL_DMA_ClearFlag_GI3(DMA_TypeDef *DMAx)
功能描述	清除 GI3 标志位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	无
先决条件	无

33.2.45 函数 LL_DMA_ClearFlag_TC1

描述了函数 LL_DMA_ClearFlag_TC1

表34-89 函数 LL_DMA_ClearFlag_TC1

函数名	LL_DMA_ClearFlag_TC1
函数原形	__STATIC_INLINE void LL_DMA_ClearFlag_TC1(DMA_TypeDef *DMAx)
功能描述	清除 TC1 标志位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	无
先决条件	无

33.2.46 函数 LL_DMA_ClearFlag_TC2

描述了函数 LL_DMA_ClearFlag_TC2

表34-90 函数 LL_DMA_ClearFlag_TC2

函数名	LL_DMA_ClearFlag_TC2
函数原形	__STATIC_INLINE void LL_DMA_ClearFlag_TC2(DMA_TypeDef *DMAx)
功能描述	清除 TC2 标志位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	无
先决条件	无

33.2.47 函数 LL_DMA_ClearFlag_TC3

描述了函数 LL_DMA_ClearFlag_TC3

表34-91 函数 LL_DMA_ClearFlag_TC3

函数名	LL_DMA_ClearFlag_TC3
函数原形	__STATIC_INLINE void LL_DMA_ClearFlag_TC3(DMA_TypeDef *DMAx)

功能描述	清除 TC3 标志位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	无
先决条件	无

33.2.48 函数 LL_DMA_ClearFlag_HT1

描述了函数 LL_DMA_ClearFlag_HT1

表34-92 函数 LL_DMA_ClearFlag_HT1

函数名	LL_DMA_ClearFlag_HT1
函数原形	__STATIC_INLINE void LL_DMA_ClearFlag_HT1(DMA_TypeDef *DMAx)
功能描述	清除 HT1 标志位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	无
先决条件	无

33.2.49 函数 LL_DMA_ClearFlag_HT2

描述了函数 LL_DMA_ClearFlag_HT2

表34-93 函数 LL_DMA_ClearFlag_HT2

函数名	LL_DMA_ClearFlag_HT2
函数原形	__STATIC_INLINE void LL_DMA_ClearFlag_HT2(DMA_TypeDef *DMAx)
功能描述	清除 HT2 标志位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	无
先决条件	无

33.2.50 函数 LL_DMA_ClearFlag_HT3

描述了函数 LL_DMA_ClearFlag_HT3

表34-94 函数 LL_DMA_ClearFlag_HT3

函数名	LL_DMA_ClearFlag_HT3
函数原形	__STATIC_INLINE void LL_DMA_ClearFlag_HT3(DMA_TypeDef *DMAx)
功能描述	清除 HT3 标志位
输入参数	DMAx：DMA 实例
输出参数	无

返回值	无
先决条件	无

33.2.51 函数 LL_DMA_ClearFlag_TE1

描述了函数 LL_DMA_ClearFlag_TE1

表34-95 函数 LL_DMA_ClearFlag_TE1

函数名	LL_DMA_ClearFlag_TE1
函数原形	__STATIC_INLINE void LL_DMA_ClearFlag_TE1(DMA_TypeDef *DMAx)
功能描述	清除 TE1 标志位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	无
先决条件	无

33.2.52 函数 LL_DMA_ClearFlag_TE2

描述了函数 LL_DMA_ClearFlag_TE2

表34-96 函数 LL_DMA_ClearFlag_TE2

函数名	LL_DMA_ClearFlag_TE2
函数原形	__STATIC_INLINE void LL_DMA_ClearFlag_TE2(DMA_TypeDef *DMAx)
功能描述	清除 TE2 标志位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	无
先决条件	无

33.2.53 函数 LL_DMA_ClearFlag_TE3

描述了函数 LL_DMA_ClearFlag_TE3

表34-97 函数 LL_DMA_ClearFlag_TE3

函数名	LL_DMA_ClearFlag_TE3
函数原形	__STATIC_INLINE void LL_DMA_ClearFlag_TE3(DMA_TypeDef *DMAx)
功能描述	清除 TE3 标志位
输入参数	DMAx：DMA 实例
输出参数	无
返回值	无
先决条件	无

33.2.54 函数 LL_DMA_EnableIT_TC

描述了函数 LL_DMA_EnableIT_TC

表34-98 函数 LL_DMA_EnableIT_TC

函数名	LL_DMA_EnableIT_TC
函数原形	__STATIC_INLINE void LL_DMA_EnableIT_TC(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	使能 TC 中断
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数:

表34-99 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.55 函数 LL_DMA_EnableIT_HT

描述了函数 LL_DMA_EnableIT_HT

表34-100 函数 LL_DMA_EnableIT_HT

函数名	LL_DMA_EnableIT_HT
函数原形	__STATIC_INLINE void LL_DMA_EnableIT_HT(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	使能 HT 中断
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数:

表34-101 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2

LL_DMA_CHANNEL_3	DMA 通道 3

33.2.56 函数 LL_DMA_EnableIT_TE

描述了函数 LL_DMA_EnableIT_TE

表34-102 函数 LL_DMA_EnableIT_TE

函数名	LL_DMA_EnableIT_TE
函数原形	__STATIC_INLINE void LL_DMA_EnableIT_TE(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	使能 TE 中断
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数:

表34-103 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.57 函数 LL_DMA_DisableIT_TC

描述了函数 LL_DMA_DisableIT_TC

表34-104 函数 LL_DMA_DisableIT_TC

函数名	LL_DMA_DisableIT_TC
函数原形	__STATIC_INLINE void LL_DMA_DisableIT_TC(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	禁用 TC 中断
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数:

表34-105 Channel 可选参数

参数	描述

LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.58 函数 LL_DMA_DisableIT_HT

描述了函数 LL_DMA_DisableIT_HT

表34-106 函数 LL_DMA_DisableIT_HT

函数名	LL_DMA_DisableIT_HT
函数原形	__STATIC_INLINE void LL_DMA_DisableIT_HT(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	禁用 HT 中断
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表34-107 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.59 函数 LL_DMA_DisableIT_TE

描述了函数 LL_DMA_DisableIT_TE

表34-108 函数 LL_DMA_DisableIT_TE

函数名	LL_DMA_DisableIT_TE
函数原形	__STATIC_INLINE void LL_DMA_DisableIT_TE(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	禁用 TE 中断
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表34-109 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.60 函数 LL_DMA_IsEnabledIT_TC

描述了函数 LL_DMA_IsEnabledIT_TC

表34-110 函数 LL_DMA_IsEnabledIT_TC

函数名	LL_DMA_IsEnabledIT_TC
函数原形	__STATIC_INLINE uint32_t LL_DMA_IsEnabledIT_TC(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	检查 TC 中断是否使能
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

Channel 可选参数:

表34-111 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.61 函数 LL_DMA_IsEnabledIT_HT

描述了函数 LL_DMA_IsEnabledIT_HT

表34-112 函数 LL_DMA_IsEnabledIT_HT

函数名	LL_DMA_IsEnabledIT_HT
函数原形	__STATIC_INLINE uint32_t LL_DMA_IsEnabledIT_HT(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	检查 HT 中断是否使能
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

Channel 可选参数：

表34-113 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.62 函数 LL_DMA_IsEnabledIT_TE

描述了函数 LL_DMA_IsEnabledIT_TE

表34-114 函数 LL_DMA_IsEnabledIT_TE

函数名	LL_DMA_IsEnabledIT_TE
函数原形	__STATIC_INLINE uint32_t LL_DMA_IsEnabledIT_TE(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	检查 TE 中断是否使能
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

Channel 可选参数：

表34-115 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.63 函数 LL_DMA_Init

描述了函数 LL_DMA_Init

表34-116 函数 LL_DMA_Init

函数名	LL_DMA_Init
函数原形	uint32_t LL_DMA_Init(DMA_TypeDef *DMAx, uint32_t Channel, LL_DMA_InitTypeDef *DMA_InitStruct)
功能描述	初始化 DMA 配置
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输入参数 3	DMA_InitStruct：DMA 初始化结构体
输出参数	无

返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

Channel 可选参数：

表34-117 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.64 函数 LL_DMA_DeInit

描述了函数 LL_DMA_DeInit

表34-118 函数 LL_DMA_DeInit

函数名	LL_DMA_DeInit
函数原形	uint32_t LL_DMA_DeInit(DMA_TypeDef *DMAx, uint32_t Channel)
功能描述	将 DMA 相关配置重设为缺省值
输入参数 1	DMAx：DMA 实例
输入参数 2	Channel：通道
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

Channel 可选参数：

表34-119 Channel 可选参数

参数	描述
LL_DMA_CHANNEL_1	DMA 通道 1
LL_DMA_CHANNEL_2	DMA 通道 2
LL_DMA_CHANNEL_3	DMA 通道 3

33.2.65 函数 LL_DMA_StructInit

描述了函数 LL_DMA_StructInit

表34-120 函数 LL_DMA_StructInit

函数名	LL_DMA_StructInit
函数原形	void LL_DMA_StructInit(LL_DMA_InitTypeDef *DMA_InitStruct)
功能描述	初始化 DMA 结构体为默认值
输入参数	DMA_InitStruct：DMA 初始化结构体
输出参数	无

返回值	无
先决条件	无

34 LL 外部中断/事件控制器通用驱动程序（EXTI）

外部中断/事件控制器管理 21 个中断输入（19 个可配置(configurable)中断和 2 个直接(direct)中断）。可配置中断能够选择触发端口、触发边沿和触发模式（中断/事件），直接中断直接由指定外设触发。

34.1 EXTI 寄存器结构

34.1.1 LL_EXTI_InitTypeDef

LL_EXTI_InitTypeDef，定义于文件“air001xx_ll_exti.h”如下：

```
typedef struct
{
uint32_t Line;
FunctionalState LineCommand;
uint8_t Mode;
uint8_t Trigger;
} LL_EXTI_InitTypeDef;
```

字段说明：

表35-1 LL_EXTI_InitTypeDef 字段说明

字段	描述
Line	配置外部的中断线
LineCommand	配置外部中断是否使能
Mode	配置内核处理外部中断的模式（中断/事件/无）
Trigger	配置外部中断触发边沿

参数说明：

Line 可选参数：

表35-2 Line 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线

LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

LineCommand 可选参数:

表35-3 LineCommand 可选参数

参数	描述
ENABLE	外部中断使能
DISABLE	外部中断禁用

Mode 可选参数:

表35-4 Mode 可选参数

参数	描述
LL_EXTI_MODE_IT	外部触发中断
LL_EXTI_MODE_EVENT	外部触发事件
LL_EXTI_MODE_IT_EVENT	外部触发中断和事件

Trigger 可选参数:

表35-5 Trigger 可选参数

参数	描述
LL_EXTI_TRIGGER_NONE	不触发中断
LL_EXTI_TRIGGER_RISING	上升沿触发中断
LL_EXTI_TRIGGER_FALLING	下降沿触发中断
LL_EXTI_TRIGGER_RISING_FALLING	上升沿和下降沿触发中断

34.2 EXTI 固件库函数

表35-6 UTILS 固件库函数说明

函数名	描述
LL_EXTI_EnableIT	使能指定EXTI 线路的外部触发中断
LL_EXTI_DisableIT	禁用指定 EXTI 线路的外部触发中断
LL_EXTI_IsEnabledIT	检查指定 EXTI 线路的外部触发中断是否使能
LL_EXTI_EnableEvent	使能指定EXTI 线路的外部触发事件
LL_EXTI_DisableEvent	禁用指定 EXTI 线路的外部触发事件
LL_EXTI_IsEnabledEvent	检查指定 EXTI 线路的外部触发事件是否使能
LL_EXTI_EnableRisingTrig	使能指定EXTI 线路的上升沿触发配置
LL_EXTI_DisableRisingTrig	禁用指定 EXTI 线路的上升沿触发配置
LL_EXTI_IsEnabledRisingTrig	检查指定 EXTI 线路的上升沿触发配置是否使能
LL_EXTI_EnableFallingTrig	使能指定EXTI 线路的下降沿触发配置
LL_EXTI_DisableFallingTrig	禁用指定 EXTI 线路的下降沿触发配置
LL_EXTI_IsEnabledFallingTrig	检查指定 EXTI 线路的下降沿触发配置是否使能
LL_EXTI_GenerateSWI	在指定线路上产生软件中断
LL_EXTI_IsActiveFlag	检查指定 EXTI 线路的挂起位
LL_EXTI_ReadFlag	获取指定EXTI 线路的挂起位
LL_EXTI_ClearFlag	清除指定EXTI 线路的挂起位
LL_EXTI_SetEXTISource	配置指定EXTI 线路的外部中断选择源
LL_EXTI_GetEXTISource	获取指定EXTI 线路的外部中断选择源
LL_EXTI_Init	初始化外部中断
LL_EXTI_DeInit	将 EXTI 配置设为缺省值
LL_EXTI_StructInit	初始化 EXTI_InitStruct 结构体

34.2.1 函数 LL_EXTI_EnableIT

描述了函数 LL_EXTI_EnableIT

表35-7 函数 LL_EXTI_EnableIT

函数名	LL_EXTI_EnableIT
函数原形	__STATIC_INLINE void LL_EXTI_EnableIT(uint32_t ExtiLine)
功能描述	使能指定 EXTI 线路的外部触发中断
输入参数	ExtiLine：EXTI 线路

输出参数	无
返回值	无
先决条件	无

ExtiLine 可选参数:

表35-8 ExtiLine 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线
LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

34.2.2 函数 LL_EXTI_DisableIT

描述了函数 LL_EXTI_DisableIT

表35-9 函数 LL_EXTI_DisableIT

函数名	LL_EXTI_DisableIT
函数原形	__STATIC_INLINE void LL_EXTI_DisableIT(uint32_t ExtiLine)
功能描述	禁用指定 EXTI 线路的外部触发中断
输入参数	ExtiLine：EXTI 线路

输出参数	无
返回值	无
先决条件	无

ExtiLine 可选参数:

表35-10 ExtiLine 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线
LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

34.2.3 函数 LL_EXTI_IsEnabledIT

描述了函数 LL_EXTI_IsEnabledIT

表35-11 函数 LL_EXTI_IsEnabledIT

函数名	LL_EXTI_IsEnabledIT
函数原形	__STATIC_INLINE uint32_t LL_EXTI_IsEnabledIT(uint32_t ExtiLine)
功能描述	检查指定 EXTI 线路的外部触发中断是否使能
输入参数	ExtiLine：EXTI 线路

输出参数	无
返回值	状态（1 或 0）
先决条件	无

ExtiLine 可选参数：

表35-12　ExtiLine 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线
LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

34.2.4 函数 LL_EXTI_EnableEvent

描述了函数 LL_EXTI_EnableEvent

表35-13　函数 LL_EXTI_EnableEvent

函数名	LL_EXTI_EnableEvent
函数原形	__STATIC_INLINE void LL_EXTI_EnableEvent(uint32_t ExtiLine)
功能描述	使能指定 EXTI 线路的外部触发事件
输入参数	ExtiLine：EXTI 线路

输出参数	无
返回值	无
先决条件	无

ExtiLine 可选参数:

表35-14 ExtiLine 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线
LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

34.2.5 函数 LL_EXTI_DisableEvent

描述了函数 LL_EXTI_DisableEvent

表35-15 函数 LL_EXTI_DisableEvent

函数名	LL_EXTI_DisableEvent
函数原形	__STATIC_INLINE void LL_EXTI_DisableEvent(uint32_t ExtiLine)
功能描述	禁用指定 EXTI 线路的外部触发事件
输入参数	ExtiLine：EXTI 线路

输出参数	无
返回值	无
先决条件	无

ExtiLine 可选参数:

表35-16 ExtiLine 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线
LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

34.2.6 函数 LL_EXTI_IsEnabledEvent

描述了函数 LL_EXTI_IsEnabledEvent

表35-17 函数 LL_EXTI_IsEnabledEvent

函数名	LL_EXTI_IsEnabledEvent
函数原形	__STATIC_INLINE uint32_t LL_EXTI_IsEnabledEvent(uint32_t ExtiLine)
功能描述	检查指定 EXTI 线路的外部触发事件是否使能
输入参数	ExtiLine：EXTI 线路

输出参数	无
返回值	状态（1 或 0）
先决条件	无

ExtiLine 可选参数：

表35-18　ExtiLine 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线
LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

34.2.7 函数 LL_EXTI_EnableRisingTrig

描述了函数 LL_EXTI_EnableRisingTrig

表35-19　函数 LL_EXTI_EnableRisingTrig

函数名	LL_EXTI_EnableRisingTrig
函数原形	__STATIC_INLINE void LL_EXTI_EnableRisingTrig(uint32_t ExtiLine)
功能描述	使能指定 EXTI 线路的上升沿触发配置
输入参数	ExtiLine：EXTI 线路

输出参数	无
返回值	无
先决条件	无

ExtiLine 可选参数:

表35-20 ExtiLine 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线
LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

34.2.8 函数 LL_EXTI_DisableRisingTrig

描述了函数 LL_EXTI_DisableRisingTrig

表35-21 函数 LL_EXTI_DisableRisingTrig

函数名	LL_EXTI_DisableRisingTrig
函数原形	__STATIC_INLINE void LL_EXTI_DisableRisingTrig(uint32_t ExtiLine)
功能描述	禁用指定 EXTI 线路的上升沿触发配置
输入参数	ExtiLine：EXTI 线路

输出参数	无
返回值	无
先决条件	无

ExtiLine 可选参数:

表35-22 ExtiLine 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线
LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

34.2.9 函数 LL_EXTI_IsEnabledRisingTrig

描述了函数 LL_EXTI_IsEnabledRisingTrig

表35-23 函数 LL_EXTI_IsEnabledRisingTrig

函数名	LL_EXTI_IsEnabledRisingTrig
函数原形	__STATIC_INLINE uint32_t LL_EXTI_IsEnabledRisingTrig(uint32_t ExtiLine)
功能描述	检查指定 EXTI 线路的上升沿触发配置是否使能
输入参数	ExtiLine：EXTI 线路

输出参数	无
返回值	状态（1 或 0）
先决条件	无

ExtiLine 可选参数：

表35-24　ExtiLine 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线
LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

34.2.10 函数 LL_EXTI_EnableFallingTrig

描述了函数 LL_EXTI_EnableFallingTrig

表35-25　函数 LL_EXTI_EnableFallingTrig

函数名	LL_EXTI_EnableFallingTrig
函数原形	__STATIC_INLINE void LL_EXTI_EnableFallingTrig(uint32_t ExtiLine)
功能描述	使能指定 EXTI 线路的下降沿触发配置
输入参数	ExtiLine：EXTI 线路

输出参数	无
返回值	无
先决条件	无

ExtiLine 可选参数:

表35-26 ExtiLine 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线
LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

34.2.11 函数 LL_EXTI_DisableFallingTrig

描述了函数 LL_EXTI_DisableFallingTrig

表35-27 函数 LL_EXTI_DisableFallingTrig

函数名	LL_EXTI_DisableFallingTrig
函数原形	__STATIC_INLINE void LL_EXTI_DisableFallingTrig(uint32_t ExtiLine)
功能描述	禁用指定 EXTI 线路的下降沿触发配置
输入参数	ExtiLine：EXTI 线路

输出参数	无
返回值	无
先决条件	无

ExtiLine 可选参数:

表35-28 ExtiLine 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线
LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

34.2.12 函数 LL_EXTI_IsEnabledFallingTrig

描述了函数 LL_EXTI_IsEnabledFallingTrig

表35-29 函数 LL_EXTI_IsEnabledFallingTrig

函数名	LL_EXTI_IsEnabledFallingTrig
函数原形	__STATIC_INLINE uint32_t LL_EXTI_IsEnabledFallingTrig(uint32_t ExtiLine)
功能描述	检查指定 EXTI 线路的下降沿触发配置是否使能
输入参数	ExtiLine：EXTI 线路

输出参数	无
返回值	状态位（1 或 0）
先决条件	无

ExtiLine 可选参数：

表35-30　ExtiLine 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线
LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

34.2.13 函数 LL_EXTI_GenerateSWI

描述了函数 LL_EXTI_GenerateSWI

表35-31　函数 LL_EXTI_GenerateSWI

函数名	LL_EXTI_GenerateSWI
函数原形	__STATIC_INLINE void LL_EXTI_GenerateSWI(uint32_t ExtiLine)
功能描述	在指定线路上产生软件中断
输入参数	ExtiLine：EXTI 线路

输出参数	无
返回值	无
先决条件	无

ExtiLine 可选参数:

表35-32 ExtiLine 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线
LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

34.2.14 函数 LL_EXTI_IsActiveFlag

描述了函数 LL_EXTI_IsActiveFlag

表35-33 函数 LL_EXTI_IsActiveFlag

函数名	LL_EXTI_IsActiveFlag
函数原形	__STATIC_INLINE uint32_t LL_EXTI_IsActiveFlag(uint32_t ExtiLine)
功能描述	检查指定 EXTI 线路的挂起位
输入参数	ExtiLine：EXTI 线路

输出参数	无
返回值	状态位（1 或 0）
先决条件	无

ExtiLine 可选参数：

表35-34　ExtiLine 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线
LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

34.2.15 函数 LL_EXTI_ReadFlag

描述了函数 LL_EXTI_ReadFlag

表35-35　函数 LL_EXTI_ReadFlag

函数名	LL_EXTI_ReadFlag
函数原形	__STATIC_INLINE uint32_t LL_EXTI_ReadFlag(uint32_t ExtiLine)
功能描述	获取指定 EXTI 线路的挂起位
输入参数	ExtiLine：EXTI 线路

输出参数	无
返回值	指定 EXTI 线路的挂起位
先决条件	无

ExtiLine 可选参数:

表35-36 ExtiLine 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线
LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

34.2.16 函数 LL_EXTI_ClearFlag

描述了函数 LL_EXTI_ClearFlag

表35-37 函数 LL_EXTI_ClearFlag

函数名	LL_EXTI_ClearFlag
函数原形	__STATIC_INLINE void LL_EXTI_ClearFlag(uint32_t ExtiLine)
功能描述	清除指定 EXTI 线路的挂起位
输入参数	ExtiLine：EXTI 线路

输出参数	无
返回值	无
先决条件	无

ExtiLine 可选参数:

表35-38 ExtiLine 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线
LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

34.2.17 函数 LL_EXTI_SetEXTISource

描述了函数 LL_EXTI_SetEXTISource

表35-39 函数 LL_EXTI_SetEXTISource

函数名	LL_EXTI_SetEXTISource
函数原形	__STATIC_INLINE void LL_EXTI_SetEXTISource(uint32_t Port, uint32_t Line)
功能描述	配置指定 EXTI 线路的外部中断选择源
输入参数 1	Port：GPIO 端口

输入参数 2	Line：EXTI 线路
输出参数	无
返回值	无
先决条件	无

Port 可选参数：

表35-40 Port 可选参数

参数	描述
LL_EXTI_CONFIG_PORTA	GPIO 端口 A
LL_EXTI_CONFIG_PORTB	GPIO 端口 B
LL_EXTI_CONFIG_PORTF	GPIO 端口 F

Line 可选参数：

表35-41 Line 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线
LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

34.2.18 函数 LL_EXTI_GetEXTISource

描述了函数 LL_EXTI_GetEXTISource

表35-42 函数 LL_EXTI_GetEXTISource

函数名	LL_EXTI_GetEXTISource
函数原形	__STATIC_INLINE void LL_EXTI_GetEXTISource(uint32_t Line)
功能描述	获取指定 EXTI 线路的外部中断选择源
输入参数	Line：EXTI 线路
输出参数	无
返回值	GPIO 端口
先决条件	无

Line 可选参数：

表35-43 Line 可选参数

参数	描述
LL_EXTI_LINE_0	0 号外部中断线
LL_EXTI_LINE_1	1 号外部中断线
LL_EXTI_LINE_2	2 号外部中断线
LL_EXTI_LINE_3	3 号外部中断线
LL_EXTI_LINE_4	4 号外部中断线
LL_EXTI_LINE_5	5 号外部中断线
LL_EXTI_LINE_6	6 号外部中断线
LL_EXTI_LINE_7	7 号外部中断线
LL_EXTI_LINE_8	8 号外部中断线
LL_EXTI_LINE_9	9 号外部中断线
LL_EXTI_LINE_10	10 号外部中断线
LL_EXTI_LINE_11	11 号外部中断线
LL_EXTI_LINE_12	12 号外部中断线
LL_EXTI_LINE_13	13 号外部中断线
LL_EXTI_LINE_14	14 号外部中断线
LL_EXTI_LINE_15	15 号外部中断线
LL_EXTI_LINE_16	16 号外部中断线
LL_EXTI_LINE_17	17 号外部中断线
LL_EXTI_LINE_18	18 号外部中断线
LL_EXTI_LINE_19	19 号外部中断线
LL_EXTI_LINE_29	29 号外部中断线

34.2.19 函数 LL_EXTI_Init

描述了函数 LL_EXTI_Init

表35-44 函数 LL_EXTI_Init

函数名	LL_EXTI_Init
函数原形	uint32_t LL_EXTI_Init(LL_EXTI_InitTypeDef *EXTI_InitStruct)
功能描述	初始化外部中断
输入参数	EXTI_InitStruct：初始化参数结构体
输出参数	无
返回值	函数执行结果（0 或其它值）
先决条件	无

34.2.20 函数 LL_EXTI_DeInit

描述了函数 LL_EXTI_DeInit

表35-45 函数 LL_EXTI_DeInit

函数名	LL_EXTI_DeInit
函数原形	uint32_t LL_EXTI_DeInit(void)
功能描述	将 EXTI 配置设为缺省值
输入参数	无
输出参数	无
返回值	0
先决条件	无

34.2.21 函数 LL_EXTI_StructInit

描述了函数 LL_EXTI_StructInit

表35-46 函数 LL_EXTI_StructInit

函数名	LL_EXTI_StructInit
函数原形	void LL_EXTI_StructInit(LL_EXTI_InitTypeDef *EXTI_InitStruct)
功能描述	初始化 EXTI_InitStruct 结构体
输入参数	EXTI_InitStruct：初始化参数结构体
输出参数	无
返回值	无
先决条件	无

35 LL 通用输入/输出通用驱动程序（GPIO）

每个 GPIO 端口有：

4 个 32 位配置寄存器(GPIOx_MODER,GPIOx_OTYPER,GPIOx_OSPEEDR, GPIOx_PUPDR)

2 个 32 位数据寄存器 (GPIOx_IDR 和 GPIOx_ODR)

1 个 32 位置位/复位寄存器(GPIOx_BSRR)

1 个 32 位锁定寄存器(GPIOx_LCKR)

2 个复用功能选择寄存器(GPIOx_AFRH 和 GPIOx_AFRL)。

35.1 GPIO 寄存器结构

35.1.1 LL_GPIO_InitTypeDef

LL_GPIO_InitTypeDef，定义于文件“air001xx_ll_gpio.h”如下：

```
typedef struct
{
uint32_t Pin;
uint32_t Mode;
uint32_t Speed;
uint32_t OutputType;
uint32_t Pull;
uint32_t Alternate;
} LL_GPIO_InitTypeDef;
```

字段说明：

表36-1 LL_GPIO_InitTypeDef 字段说明

字段	描述
Pin	选择需要配置的引脚
Mode	配置指定引脚的模式
Speed	配置指定引脚的速度
OutputType	配置指定引脚的输出类型
Pull	配置指定引脚上拉或下拉
Alternate	配置指定引脚需要连接到的外设

参数说明：

Pin 可选参数：

表36-2 Pin 可选参数

参数	描述

LL_GPIO_PIN_0	引脚 0
LL_GPIO_PIN_1	引脚 1
LL_GPIO_PIN_2	引脚 2
LL_GPIO_PIN_3	引脚 3
LL_GPIO_PIN_4	引脚 4
LL_GPIO_PIN_5	引脚 5
LL_GPIO_PIN_6	引脚 6
LL_GPIO_PIN_7	引脚 7
LL_GPIO_PIN_8	引脚 8
LL_GPIO_PIN_9	引脚 9
LL_GPIO_PIN_10	引脚 10
LL_GPIO_PIN_11	引脚 11
LL_GPIO_PIN_12	引脚 12
LL_GPIO_PIN_13	引脚 13
LL_GPIO_PIN_14	引脚 14
LL_GPIO_PIN_15	引脚 15
LL_GPIO_PIN_ALL	全部引脚

Mode 可选参数：

表36-3 Mode 可选参数

参数	描述
LL_GPIO_MODE_INPUT	输入模式
LL_GPIO_MODE_OUTPUT	输出模式
LL_GPIO_MODE_ALTERNATE	复用模式
LL_GPIO_MODE_ANALOG	模拟模式

Speed 可选参数：

表36-4 Speed 可选参数

参数	描述
LL_GPIO_SPEED_FREQ_LOW	低速度模式
LL_GPIO_SPEED_FREQ_MEDIUM	中速度模式
LL_GPIO_SPEED_FREQ_HIGH	高速度模式
LL_GPIO_SPEED_FREQ_VERY_HIGH	最高速度模式

OutputType 可选参数：

表36-5 OutputType 可选参数

参数	描述

LL_GPIO_OUTPUT_PUSHPULL	推挽输出
LL_GPIO_OUTPUT_OPENDRAIN	开漏输出

Pull 可选参数:

表36-6 Pull 可选参数

参数	描述
LL_GPIO_PULL_NO	无上拉或下拉
LL_GPIO_PULL_UP	引脚上拉
LL_GPIO_PULL_DOWN	引脚下拉

Alternate 可选参数:

表36-7 Alternate 可选参数

参数	描述
LL_GPIO_AF_0	复用模式 AF0
LL_GPIO_AF_1	复用模式 AF1
LL_GPIO_AF_2	复用模式 AF2
LL_GPIO_AF_3	复用模式 AF3
LL_GPIO_AF_4	复用模式 AF4
LL_GPIO_AF_5	复用模式 AF5
LL_GPIO_AF_6	复用模式 AF6
LL_GPIO_AF_7	复用模式 AF7
LL_GPIO_AF_8	复用模式 AF8
LL_GPIO_AF_9	复用模式 AF9
LL_GPIO_AF_10	复用模式 AF10
LL_GPIO_AF_11	复用模式 AF11
LL_GPIO_AF_12	复用模式 AF12
LL_GPIO_AF_13	复用模式 AF13
LL_GPIO_AF_14	复用模式 AF14
LL_GPIO_AF_15	复用模式 AF15
LL_GPIO_AF0_SWJ	复用模式 AF0：连接到 SWDIO
LL_GPIO_AF0_SPI1	复用模式 AF0：连接到 SPI1
LL_GPIO_AF0_SPI2	复用模式 AF0：连接到 SPI2
LL_GPIO_AF0_TIM14	复用模式 AF0：连接到 TIM14
LL_GPIO_AF0_USART1	复用模式 AF0：连接到 USART1
LL_GPIO_AF0_USART2	复用模式 AF0：连接到 USART2
LL_GPIO_AF1_IR	复用模式 AF1：连接到 IRTIM
LL_GPIO_AF1_SPI2	复用模式 AF1：连接到 SPI2

LL_GPIO_AF1_TIM1	复用模式 AF1：连接到 TIM1
LL_GPIO_AF1_TIM3	复用模式 AF1：连接到 TIM3
LL_GPIO_AF1_USART1	复用模式 AF1：连接到 USART1
LL_GPIO_AF1_USART2	复用模式 AF1：连接到 USART2
LL_GPIO_AF2_SPI2	复用模式 AF2：连接到 SPI2
LL_GPIO_AF2_TIM2	复用模式 AF2：连接到 TIM2
LL_GPIO_AF2_TIM14	复用模式 AF2：连接到 TIM14
LL_GPIO_AF2_TIM16	复用模式 AF2：连接到 TIM16
LL_GPIO_AF2_TIM17	复用模式 AF2：连接到 TIM17
LL_GPIO_AF3_LED	复用模式 AF3：连接到 LED
LL_GPIO_AF3_USART1	复用模式 AF3：连接到 USART1
LL_GPIO_AF3_USART2	复用模式 AF3：连接到 USART2
LL_GPIO_AF3_SPI2	复用模式 AF3：连接到 SPI2
LL_GPIO_AF4_TIM14	复用模式 AF4：连接到 TIM14
LL_GPIO_AF4_USART2	复用模式 AF4：连接到 USART2
LL_GPIO_AF5_LPTIM	复用模式 AF5：连接到 LPTIM
LL_GPIO_AF5_USART2	复用模式 AF5：连接到 USART2
LL_GPIO_AF5_TIM16	复用模式 AF5：连接到 TIM16
LL_GPIO_AF5_TIM17	复用模式 AF5：连接到 TIM17
LL_GPIO_AF5_EVENTOUT	复用模式 AF5：连接到 EVENT OUT
LL_GPIO_AF5_MCO	复用模式 AF5：连接到 MCO
LL_GPIO_AF6_I2C	复用模式 AF6：连接到 I2C
LL_GPIO_AF6_LED	复用模式 AF6：连接到 LED
LL_GPIO_AF6_MCO	复用模式 AF6：连接到 MCO
LL_GPIO_AF6_EVENTOUT	复用模式 AF6：连接到 EVENT OUT
LL_GPIO_AF7_EVENTOUT	复用模式 AF7：连接到 EVENT OUT
LL_GPIO_AF7_COMP1	复用模式 AF7：连接到 COMP1
LL_GPIO_AF7_COMP2	复用模式 AF7：连接到 COMP2
LL_GPIO_AF8_USART1	复用模式 AF8：连接到 USART1
LL_GPIO_AF9_USART2	复用模式 AF9：连接到 USART2
LL_GPIO_AF10_SPI1	复用模式 AF10：连接到 SPI1
LL_GPIO_AF11_SPI2	复用模式 AF11：连接到 SPI2
LL_GPIO_AF12_I2C	复用模式 AF12：连接到 I2C
LL_GPIO_AF13_TIM1	复用模式 AF13：连接到 TIM1
LL_GPIO_AF13_TIM3	复用模式 AF13：连接到 TIM3
LL_GPIO_AF13_TIM14	复用模式 AF13：连接到 TIM14

LL_GPIO_AF13_TIM17	复用模式 AF13：连接到 TIM17
LL_GPIO_AF14_TIM1	复用模式 AF14：连接到 TIM1
LL_GPIO_AF15_RTCOUT	复用模式 AF15：连接到 RTC OUT
LL_GPIO_AF15_MCO	复用模式 AF15：连接到 MCO
LL_GPIO_AF15_IR	复用模式 AF15：连接到 IRTIM

35.2 GPIO 固件库函数

表36-8 GPIO 固件库函数说明

函数名	描述
LL_GPIO_SetPinMode	配置指定引脚的模式
LL_GPIO_GetPinMode	获取指定引脚的模式
LL_GPIO_SetPinOutputType	配置指定引脚的输出类型
LL_GPIO_GetPinOutputType	获取指定引脚的输出类型
LL_GPIO_SetPinSpeed	配置指定引脚的速度
LL_GPIO_GetPinSpeed	获取指定引脚的速度
LL_GPIO_SetPinPull	配置指定引脚上拉或下拉
LL_GPIO_GetPinPull	获取指定引脚上拉或下拉
LL_GPIO_SetAFPin_0_7	配置指定引脚(0~7)的复用功能
LL_GPIO_GetAFPin_0_7	获取指定引脚(0~7)的复用功能
LL_GPIO_SetAFPin_8_15	配置指定引脚(8~15)的复用功能
LL_GPIO_GetAFPin_8_15	获取指定引脚(8~15)的复用功能
LL_GPIO_LockPin	锁定指定引脚的配置
LL_GPIO_IsPinLocked	检查指定引脚的配置是否锁定
LL_GPIO_IsAnyPinLocked	检查是否已锁定至少 1 个引脚的配置
LL_GPIO_ReadInputPort	获取指定端口的输入数据
LL_GPIO_IsInputPinSet	检查指定引脚是否输入高电平
LL_GPIO_WriteOutputPort	配置指定端口的输出数据
LL_GPIO_ReadOutputPort	获取指定端口的输出数据
LL_GPIO_IsOutputPinSet	检查指定引脚是否输出高电平
LL_GPIO_SetOutputPin	配置指定引脚输出高电平
LL_GPIO_ResetOutputPin	配置指定引脚输出低电平
LL_GPIO_TogglePin	配置指定引脚输出翻转

LL_GPIO_DeInit	将 GPIO 配置重设为缺省值
LL_GPIO_Init	初始化 GPIO
LL_GPIO_StructInit	初始化 GPIO_InitStruct 结构体

35.2.1 函数 LL_GPIO_SetPinMode

描述了函数 LL_GPIO_SetPinMode

表36-9 函数 LL_GPIO_SetPinMode

函数名	LL_GPIO_SetPinMode
函数原形	__STATIC_INLINE void LL_GPIO_SetPinMode(GPIO_TypeDef *GPIOx, uint32_t Pin, uint32_t Mode)
功能描述	配置指定引脚的模式
输入参数 1	GPIOx：GPIO 端口
输入参数 2	Pin：GPIO 引脚
输入参数 3	Mode：GPIO 模式
输出参数	无
返回值	无
先决条件	无

GPIOx 可选参数：

表36-10 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

Pin 可选参数：

表36-11 Pin 可选参数

参数	描述
LL_GPIO_PIN_0	引脚 0
LL_GPIO_PIN_1	引脚 1
LL_GPIO_PIN_2	引脚 2
LL_GPIO_PIN_3	引脚 3
LL_GPIO_PIN_4	引脚 4
LL_GPIO_PIN_5	引脚 5
LL_GPIO_PIN_6	引脚 6
LL_GPIO_PIN_7	引脚 7
LL_GPIO_PIN_8	引脚 8

LL_GPIO_PIN_9	引脚 9
LL_GPIO_PIN_10	引脚 10
LL_GPIO_PIN_11	引脚 11
LL_GPIO_PIN_12	引脚 12
LL_GPIO_PIN_13	引脚 13
LL_GPIO_PIN_14	引脚 14
LL_GPIO_PIN_15	引脚 15

Mode 可选参数：

表36-12　Mode 可选参数

参数	描述
LL_GPIO_MODE_INPUT	输入模式
LL_GPIO_MODE_OUTPUT	输出模式
LL_GPIO_MODE_ALTERNATE	复用模式
LL_GPIO_MODE_ANALOG	模拟模式

35.2.2 函数 LL_GPIO_GetPinMode

描述了函数 LL_GPIO_GetPinMode

表36-13　函数 LL_GPIO_GetPinMode

函数名	LL_GPIO_GetPinMode
函数原形	__STATIC_INLINE uint32_t LL_GPIO_GetPinMode(GPIO_TypeDef *GPIOx, uint32_t Pin)
功能描述	获取指定引脚的模式
输入参数 1	GPIOx：GPIO 端口
输入参数 2	Pin：GPIO 引脚
输出参数	无
返回值	端口模式
先决条件	无

GPIOx 可选参数：

表36-14　GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

Pin 可选参数：

表36-15 Pin 可选参数

参数	描述
LL_GPIO_PIN_0	引脚 0
LL_GPIO_PIN_1	引脚 1
LL_GPIO_PIN_2	引脚 2
LL_GPIO_PIN_3	引脚 3
LL_GPIO_PIN_4	引脚 4
LL_GPIO_PIN_5	引脚 5
LL_GPIO_PIN_6	引脚 6
LL_GPIO_PIN_7	引脚 7
LL_GPIO_PIN_8	引脚 8
LL_GPIO_PIN_9	引脚 9
LL_GPIO_PIN_10	引脚 10
LL_GPIO_PIN_11	引脚 11
LL_GPIO_PIN_12	引脚 12
LL_GPIO_PIN_13	引脚 13
LL_GPIO_PIN_14	引脚 14
LL_GPIO_PIN_15	引脚 15

35.2.3 函数 LL_GPIO_SetPinOutputType

描述了函数 LL_GPIO_SetPinOutputType

表36-16 函数 LL_GPIO_SetPinOutputType

函数名	LL_GPIO_SetPinOutputType
函数原形	__STATIC_INLINE void LL_GPIO_SetPinOutputType(GPIO_TypeDef *GPIOx, uint32_t PinMask, uint32_t OutputType)
功能描述	配置指定引脚的输出类型
输入参数 1	GPIOx：GPIO 端口
输入参数 2	PinMask：GPIO 引脚
输入参数 3	OutputType：输出类型
输出参数	无
返回值	无
先决条件	无

GPIOx 可选参数：

表36-17 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A

GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

PinMask 可选参数：

表36-18 PinMask 可选参数

参数	描述
LL_GPIO_PIN_0	引脚 0
LL_GPIO_PIN_1	引脚 1
LL_GPIO_PIN_2	引脚 2
LL_GPIO_PIN_3	引脚 3
LL_GPIO_PIN_4	引脚 4
LL_GPIO_PIN_5	引脚 5
LL_GPIO_PIN_6	引脚 6
LL_GPIO_PIN_7	引脚 7
LL_GPIO_PIN_8	引脚 8
LL_GPIO_PIN_9	引脚 9
LL_GPIO_PIN_10	引脚 10
LL_GPIO_PIN_11	引脚 11
LL_GPIO_PIN_12	引脚 12
LL_GPIO_PIN_13	引脚 13
LL_GPIO_PIN_14	引脚 14
LL_GPIO_PIN_15	引脚 15
LL_GPIO_PIN_ALL	全部引脚

OutputType 可选参数：

表36-19 OutputType 可选参数

参数	描述
LL_GPIO_OUTPUT_PUSHPULL	推挽输出
LL_GPIO_OUTPUT_OPENDRAIN	开漏输出

35.2.4 函数 LL_GPIO_GetPinOutputType

描述了函数 LL_GPIO_GetPinOutputType

表36-20 函数 LL_GPIO_GetPinOutputType

函数名	LL_GPIO_GetPinOutputType
函数原形	__STATIC_INLINE uint32_t LL_GPIO_GetPinOutputType(GPIO_TypeDef *GPIOx, uint32_t Pin)
功能描述	获取指定引脚的输出类型

输入参数 1	GPIOx：GPIO 端口
输入参数 2	Pin：GPIO 引脚
输出参数	无
返回值	输出类型
先决条件	无

GPIOx 可选参数：

表36-21 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

Pin 可选参数：

表36-22 Pin 可选参数

参数	描述
LL_GPIO_PIN_0	引脚 0
LL_GPIO_PIN_1	引脚 1
LL_GPIO_PIN_2	引脚 2
LL_GPIO_PIN_3	引脚 3
LL_GPIO_PIN_4	引脚 4
LL_GPIO_PIN_5	引脚 5
LL_GPIO_PIN_6	引脚 6
LL_GPIO_PIN_7	引脚 7
LL_GPIO_PIN_8	引脚 8
LL_GPIO_PIN_9	引脚 9
LL_GPIO_PIN_10	引脚 10
LL_GPIO_PIN_11	引脚 11
LL_GPIO_PIN_12	引脚 12
LL_GPIO_PIN_13	引脚 13
LL_GPIO_PIN_14	引脚 14
LL_GPIO_PIN_15	引脚 15

35.2.5 函数 LL_GPIO_SetPinSpeed

描述了函数 LL_GPIO_SetPinSpeed

表36-23 函数 LL_GPIO_SetPinSpeed

函数名	LL_GPIO_SetPinSpeed

函数原形	__STATIC_INLINE void LL_GPIO_SetPinSpeed(GPIO_TypeDef *GPIOx, uint32_t Pin, uint32_t Speed)
功能描述	配置指定引脚的速度
输入参数 1	GPIOx：GPIO 端口
输入参数 2	Pin：GPIO 引脚
输入参数 3	Speed：速度
输出参数	无
返回值	无
先决条件	无

GPIOx 可选参数：

表36-24 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

Pin 可选参数：

表36-25 Pin 可选参数

参数	描述
LL_GPIO_PIN_0	引脚 0
LL_GPIO_PIN_1	引脚 1
LL_GPIO_PIN_2	引脚 2
LL_GPIO_PIN_3	引脚 3
LL_GPIO_PIN_4	引脚 4
LL_GPIO_PIN_5	引脚 5
LL_GPIO_PIN_6	引脚 6
LL_GPIO_PIN_7	引脚 7
LL_GPIO_PIN_8	引脚 8
LL_GPIO_PIN_9	引脚 9
LL_GPIO_PIN_10	引脚 10
LL_GPIO_PIN_11	引脚 11
LL_GPIO_PIN_12	引脚 12
LL_GPIO_PIN_13	引脚 13
LL_GPIO_PIN_14	引脚 14
LL_GPIO_PIN_15	引脚 15

Speed 可选参数：

表36-26 Speed 可选参数

参数	描述
LL_GPIO_SPEED_FREQ_LOW	低速度模式
LL_GPIO_SPEED_FREQ_MEDIUM	中速度模式
LL_GPIO_SPEED_FREQ_HIGH	高速度模式
LL_GPIO_SPEED_FREQ_VERY_HIGH	最高速度模式

35.2.6 函数 LL_GPIO_GetPinSpeed

描述了函数 LL_GPIO_GetPinSpeed

表36-27 函数 LL_GPIO_GetPinSpeed

函数名	LL_GPIO_GetPinSpeed
函数原形	__STATIC_INLINE uint32_t LL_GPIO_GetPinSpeed(GPIO_TypeDef *GPIOx, uint32_t Pin)
功能描述	获取指定引脚的速度
输入参数 1	GPIOx：GPIO 端口
输入参数 2	Pin：GPIO 引脚
输出参数	无
返回值	速度
先决条件	无

GPIOx 可选参数：

表36-28 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

Pin 可选参数：

表36-29 Pin 可选参数

参数	描述
LL_GPIO_PIN_0	引脚 0
LL_GPIO_PIN_1	引脚 1
LL_GPIO_PIN_2	引脚 2
LL_GPIO_PIN_3	引脚 3
LL_GPIO_PIN_4	引脚 4
LL_GPIO_PIN_5	引脚 5
LL_GPIO_PIN_6	引脚 6
LL_GPIO_PIN_7	引脚 7

LL_GPIO_PIN_8	引脚 8
LL_GPIO_PIN_9	引脚 9
LL_GPIO_PIN_10	引脚 10
LL_GPIO_PIN_11	引脚 11
LL_GPIO_PIN_12	引脚 12
LL_GPIO_PIN_13	引脚 13
LL_GPIO_PIN_14	引脚 14
LL_GPIO_PIN_15	引脚 15

35.2.7 函数 LL_GPIO_SetPinPull

描述了函数 LL_GPIO_SetPinPull

表36-30 函数 LL_GPIO_SetPinPull

函数名	LL_GPIO_SetPinPull
函数原形	__STATIC_INLINE void LL_GPIO_SetPinPull(GPIO_TypeDef *GPIOx, uint32_t Pin, uint32_t Pull)
功能描述	配置指定引脚上拉或下拉
输入参数 1	GPIOx：GPIO 端口
输入参数 2	Pin：GPIO 引脚
输入参数 3	Pull：上下拉
输出参数	无
返回值	无
先决条件	无

GPIOx 可选参数：

表36-31 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

Pin 可选参数：

表36-32 Pin 可选参数

参数	描述
LL_GPIO_PIN_0	引脚 0
LL_GPIO_PIN_1	引脚 1
LL_GPIO_PIN_2	引脚 2
LL_GPIO_PIN_3	引脚 3

LL_GPIO_PIN_4	引脚 4
LL_GPIO_PIN_5	引脚 5
LL_GPIO_PIN_6	引脚 6
LL_GPIO_PIN_7	引脚 7
LL_GPIO_PIN_8	引脚 8
LL_GPIO_PIN_9	引脚 9
LL_GPIO_PIN_10	引脚 10
LL_GPIO_PIN_11	引脚 11
LL_GPIO_PIN_12	引脚 12
LL_GPIO_PIN_13	引脚 13
LL_GPIO_PIN_14	引脚 14
LL_GPIO_PIN_15	引脚 15

Pull 可选参数：

表36-33 Pull 可选参数

参数	描述
LL_GPIO_PULL_NO	无上拉或下拉
LL_GPIO_PULL_UP	引脚上拉
LL_GPIO_PULL_DOWN	引脚下拉

35.2.8 函数 LL_GPIO_GetPinPull

描述了函数 LL_GPIO_GetPinPull

表36-34 函数 LL_GPIO_GetPinPull

函数名	LL_GPIO_GetPinPull
函数原形	__STATIC_INLINE uint32_t LL_GPIO_GetPinPull(GPIO_TypeDef *GPIOx, uint32_t Pin)
功能描述	获取指定引脚上拉或下拉
输入参数 1	GPIOx：GPIO 端口
输入参数 2	Pin：GPIO 引脚
输出参数	无
返回值	上拉或下拉
先决条件	无

GPIOx 可选参数：

表36-35 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A

GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

Pin 可选参数:

表36-36 Pin 可选参数

参数	描述
LL_GPIO_PIN_0	引脚 0
LL_GPIO_PIN_1	引脚 1
LL_GPIO_PIN_2	引脚 2
LL_GPIO_PIN_3	引脚 3
LL_GPIO_PIN_4	引脚 4
LL_GPIO_PIN_5	引脚 5
LL_GPIO_PIN_6	引脚 6
LL_GPIO_PIN_7	引脚 7
LL_GPIO_PIN_8	引脚 8
LL_GPIO_PIN_9	引脚 9
LL_GPIO_PIN_10	引脚 10
LL_GPIO_PIN_11	引脚 11
LL_GPIO_PIN_12	引脚 12
LL_GPIO_PIN_13	引脚 13
LL_GPIO_PIN_14	引脚 14
LL_GPIO_PIN_15	引脚 15

35.2.9 函数 LL_GPIO_SetAFPin_0_7

描述了函数 LL_GPIO_SetAFPin_0_7

表36-37 函数 LL_GPIO_SetAFPin_0_7

函数名	LL_GPIO_SetAFPin_0_7
函数原形	__STATIC_INLINE void LL_GPIO_SetAFPin_0_7(GPIO_TypeDef *GPIOx, uint32_t Pin, uint32_t Alternate)
功能描述	配置指定引脚(0~7)的复用功能
输入参数 1	GPIOx：GPIO 端口
输入参数 2	Pin：GPIO 引脚
输入参数 3	Alternate：复用功能
输出参数	无
返回值	无
先决条件	无

GPIOx 可选参数：

表36-38 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

Pin 可选参数：

表36-39 Pin 可选参数

参数	描述
LL_GPIO_PIN_0	引脚 0
LL_GPIO_PIN_1	引脚 1
LL_GPIO_PIN_2	引脚 2
LL_GPIO_PIN_3	引脚 3
LL_GPIO_PIN_4	引脚 4
LL_GPIO_PIN_5	引脚 5
LL_GPIO_PIN_6	引脚 6
LL_GPIO_PIN_7	引脚 7

Alternate 可选参数：

表36-40 Alternate 可选参数

参数	描述
LL_GPIO_AF_0	复用模式 AF0
LL_GPIO_AF_1	复用模式 AF1
LL_GPIO_AF_2	复用模式 AF2
LL_GPIO_AF_3	复用模式 AF3
LL_GPIO_AF_4	复用模式 AF4
LL_GPIO_AF_5	复用模式 AF5
LL_GPIO_AF_6	复用模式 AF6
LL_GPIO_AF_7	复用模式 AF7
LL_GPIO_AF_8	复用模式 AF8
LL_GPIO_AF_9	复用模式 AF9
LL_GPIO_AF_10	复用模式 AF10
LL_GPIO_AF_11	复用模式 AF11
LL_GPIO_AF_12	复用模式 AF12
LL_GPIO_AF_13	复用模式 AF13
LL_GPIO_AF_14	复用模式 AF14

LL_GPIO_AF_15	复用模式 AF15
LL_GPIO_AF0_SWJ	复用模式 AF0：连接到 SWDIO
LL_GPIO_AF0_SPI1	复用模式 AF0：连接到 SPI1
LL_GPIO_AF0_SPI2	复用模式 AF0：连接到 SPI2
LL_GPIO_AF0_TIM14	复用模式 AF0：连接到 TIM14
LL_GPIO_AF0_USART1	复用模式 AF0：连接到 USART1
LL_GPIO_AF0_USART2	复用模式 AF0：连接到 USART2
LL_GPIO_AF1_IR	复用模式 AF1：连接到 IRTIM
LL_GPIO_AF1_SPI2	复用模式 AF1：连接到 SPI2
LL_GPIO_AF1_TIM1	复用模式 AF1：连接到 TIM1
LL_GPIO_AF1_TIM3	复用模式 AF1：连接到 TIM3
LL_GPIO_AF1_USART1	复用模式 AF1：连接到 USART1
LL_GPIO_AF1_USART2	复用模式 AF1：连接到 USART2
LL_GPIO_AF2_SPI2	复用模式 AF2：连接到 SPI2
LL_GPIO_AF2_TIM2	复用模式 AF2：连接到 TIM2
LL_GPIO_AF2_TIM14	复用模式 AF2：连接到 TIM14
LL_GPIO_AF2_TIM16	复用模式 AF2：连接到 TIM16
LL_GPIO_AF2_TIM17	复用模式 AF2：连接到 TIM17
LL_GPIO_AF3_LED	复用模式 AF3：连接到 LED
LL_GPIO_AF3_USART1	复用模式 AF3：连接到 USART1
LL_GPIO_AF3_USART2	复用模式 AF3：连接到 USART2
LL_GPIO_AF3_SPI2	复用模式 AF3：连接到 SPI2
LL_GPIO_AF4_TIM14	复用模式 AF4：连接到 TIM14
LL_GPIO_AF4_USART2	复用模式 AF4：连接到 USART2
LL_GPIO_AF5_LPTIM	复用模式 AF5：连接到 LPTIM
LL_GPIO_AF5_USART2	复用模式 AF5：连接到 USART2
LL_GPIO_AF5_TIM16	复用模式 AF5：连接到 TIM16
LL_GPIO_AF5_TIM17	复用模式 AF5：连接到 TIM17
LL_GPIO_AF5_EVENTOUT	复用模式 AF5：连接到 EVENT OUT
LL_GPIO_AF5_MCO	复用模式 AF5：连接到 MCO
LL_GPIO_AF6_I2C	复用模式 AF6：连接到 I2C
LL_GPIO_AF6_LED	复用模式 AF6：连接到 LED
LL_GPIO_AF6_MCO	复用模式 AF6：连接到 MCO
LL_GPIO_AF6_EVENTOUT	复用模式 AF6：连接到 EVENT OUT
LL_GPIO_AF7_EVENTOUT	复用模式 AF7：连接到 EVENT OUT
LL_GPIO_AF7_COMP1	复用模式 AF7：连接到 COMP1

LL_GPIO_AF7_COMP2	复用模式 AF7：连接到 COMP2
LL_GPIO_AF8_USART1	复用模式 AF8：连接到 USART1
LL_GPIO_AF9_USART2	复用模式 AF9：连接到 USART2
LL_GPIO_AF10_SPI1	复用模式 AF10：连接到 SPI1
LL_GPIO_AF11_SPI2	复用模式 AF11：连接到 SPI2
LL_GPIO_AF12_I2C	复用模式 AF12：连接到 I2C
LL_GPIO_AF13_TIM1	复用模式 AF13：连接到 TIM1
LL_GPIO_AF13_TIM3	复用模式 AF13：连接到 TIM3
LL_GPIO_AF13_TIM14	复用模式 AF13：连接到 TIM14
LL_GPIO_AF13_TIM17	复用模式 AF13：连接到 TIM17
LL_GPIO_AF14_TIM1	复用模式 AF14：连接到 TIM1
LL_GPIO_AF15_RTCOUT	复用模式 AF15：连接到 RTC OUT
LL_GPIO_AF15_MCO	复用模式 AF15：连接到 MCO
LL_GPIO_AF15_IR	复用模式 AF15：连接到 IRTIM

35.2.10 函数 LL_GPIO_GetAFPin_0_7

描述了函数 LL_GPIO_GetAFPin_0_7

表36-41 函数 LL_GPIO_GetAFPin_0_7

函数名	LL_GPIO_GetAFPin_0_7
函数原形	__STATIC_INLINE uint32_t LL_GPIO_GetAFPin_0_7(GPIO_TypeDef *GPIOx, uint32_t Pin)
功能描述	获取指定引脚（0~7）的复用功能
输入参数 1	GPIOx：GPIO 端口
输入参数 2	Pin：GPIO 引脚
输出参数	无
返回值	复用功能
先决条件	无

GPIOx 可选参数：

表36-42 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

Pin 可选参数：

表36-43 Pin 可选参数

参数	描述
LL_GPIO_PIN_0	引脚 0
LL_GPIO_PIN_1	引脚 1
LL_GPIO_PIN_2	引脚 2
LL_GPIO_PIN_3	引脚 3
LL_GPIO_PIN_4	引脚 4
LL_GPIO_PIN_5	引脚 5
LL_GPIO_PIN_6	引脚 6
LL_GPIO_PIN_7	引脚 7

35.2.11 函数 LL_GPIO_SetAFPin_8_15

描述了函数 LL_GPIO_SetAFPin_8_15

表36-44 函数 LL_GPIO_SetAFPin_8_15

函数名	LL_GPIO_SetAFPin_8_15
函数原形	__STATIC_INLINE void LL_GPIO_SetAFPin_8_15(GPIO_TypeDef *GPIOx, uint32_t Pin, uint32_t Alternate)
功能描述	配置指定引脚(8~15)的复用功能
输入参数 1	GPIOx：GPIO 端口
输入参数 2	Pin：GPIO 引脚
输入参数 3	Alternate：复用功能
输出参数	无
返回值	无
先决条件	无

GPIOx 可选参数:

表36-45 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

Pin 可选参数:

表36-46 Pin 可选参数

参数	描述
LL_GPIO_PIN_8	引脚 8
LL_GPIO_PIN_9	引脚 9
LL_GPIO_PIN_10	引脚 10

LL_GPIO_PIN_11	引脚 11
LL_GPIO_PIN_12	引脚 12
LL_GPIO_PIN_13	引脚 13
LL_GPIO_PIN_14	引脚 14
LL_GPIO_PIN_15	引脚 15

Alternate 可选参数：

表36-47 Alternate 可选参数

参数	描述
LL_GPIO_AF_0	复用模式 AF0
LL_GPIO_AF_1	复用模式 AF1
LL_GPIO_AF_2	复用模式 AF2
LL_GPIO_AF_3	复用模式 AF3
LL_GPIO_AF_4	复用模式 AF4
LL_GPIO_AF_5	复用模式 AF5
LL_GPIO_AF_6	复用模式 AF6
LL_GPIO_AF_7	复用模式 AF7
LL_GPIO_AF_8	复用模式 AF8
LL_GPIO_AF_9	复用模式 AF9
LL_GPIO_AF_10	复用模式 AF10
LL_GPIO_AF_11	复用模式 AF11
LL_GPIO_AF_12	复用模式 AF12
LL_GPIO_AF_13	复用模式 AF13
LL_GPIO_AF_14	复用模式 AF14
LL_GPIO_AF_15	复用模式 AF15
LL_GPIO_AF0_SWJ	复用模式 AF0：连接到 SWDIO
LL_GPIO_AF0_SPI1	复用模式 AF0：连接到 SPI1
LL_GPIO_AF0_SPI2	复用模式 AF0：连接到 SPI2
LL_GPIO_AF0_TIM14	复用模式 AF0：连接到 TIM14
LL_GPIO_AF0_USART1	复用模式 AF0：连接到 USART1
LL_GPIO_AF0_USART2	复用模式 AF0：连接到 USART2
LL_GPIO_AF1_IR	复用模式 AF1：连接到 IRTIM
LL_GPIO_AF1_SPI2	复用模式 AF1：连接到 SPI2
LL_GPIO_AF1_TIM1	复用模式 AF1：连接到 TIM1
LL_GPIO_AF1_TIM3	复用模式 AF1：连接到 TIM3
LL_GPIO_AF1_USART1	复用模式 AF1：连接到 USART1

LL_GPIO_AF1_USART2	复用模式 AF1：连接到 USART2
LL_GPIO_AF2_SPI2	复用模式 AF2：连接到 SPI2
LL_GPIO_AF2_TIM2	复用模式 AF2：连接到 TIM2
LL_GPIO_AF2_TIM14	复用模式 AF2：连接到 TIM14
LL_GPIO_AF2_TIM16	复用模式 AF2：连接到 TIM16
LL_GPIO_AF2_TIM17	复用模式 AF2：连接到 TIM17
LL_GPIO_AF3_LED	复用模式 AF3：连接到 LED
LL_GPIO_AF3_USART1	复用模式 AF3：连接到 USART1
LL_GPIO_AF3_USART2	复用模式 AF3：连接到 USART2
LL_GPIO_AF3_SPI2	复用模式 AF3：连接到 SPI2
LL_GPIO_AF4_TIM14	复用模式 AF4：连接到 TIM14
LL_GPIO_AF4_USART2	复用模式 AF4：连接到 USART2
LL_GPIO_AF5_LPTIM	复用模式 AF5：连接到 LPTIM
LL_GPIO_AF5_USART2	复用模式 AF5：连接到 USART2
LL_GPIO_AF5_TIM16	复用模式 AF5：连接到 TIM16
LL_GPIO_AF5_TIM17	复用模式 AF5：连接到 TIM17
LL_GPIO_AF5_EVENTOUT	复用模式 AF5：连接到 EVENT OUT
LL_GPIO_AF5_MCO	复用模式 AF5：连接到 MCO
LL_GPIO_AF6_I2C	复用模式 AF6：连接到 I2C
LL_GPIO_AF6_LED	复用模式 AF6：连接到 LED
LL_GPIO_AF6_MCO	复用模式 AF6：连接到 MCO
LL_GPIO_AF6_EVENTOUT	复用模式 AF6：连接到 EVENT OUT
LL_GPIO_AF7_EVENTOUT	复用模式 AF7：连接到 EVENT OUT
LL_GPIO_AF7_COMP1	复用模式 AF7：连接到 COMP1
LL_GPIO_AF7_COMP2	复用模式 AF7：连接到 COMP2
LL_GPIO_AF8_USART1	复用模式 AF8：连接到 USART1
LL_GPIO_AF9_USART2	复用模式 AF9：连接到 USART2
LL_GPIO_AF10_SPI1	复用模式 AF10：连接到 SPI1
LL_GPIO_AF11_SPI2	复用模式 AF11：连接到 SPI2
LL_GPIO_AF12_I2C	复用模式 AF12：连接到 I2C
LL_GPIO_AF13_TIM1	复用模式 AF13：连接到 TIM1
LL_GPIO_AF13_TIM3	复用模式 AF13：连接到 TIM3
LL_GPIO_AF13_TIM14	复用模式 AF13：连接到 TIM14
LL_GPIO_AF13_TIM17	复用模式 AF13：连接到 TIM17
LL_GPIO_AF14_TIM1	复用模式 AF14：连接到 TIM1
LL_GPIO_AF15_RTCOUT	复用模式 AF15：连接到 RTC OUT

LL_GPIO_AF15_MCO	复用模式 AF15：连接到 MCO
LL_GPIO_AF15_IR	复用模式 AF15：连接到 IRTIM

35.2.12 函数 LL_GPIO_GetAFPin_8_15

描述了函数 LL_GPIO_GetAFPin_8_15

表36-48 函数 LL_GPIO_GetAFPin_8_15

函数名	LL_GPIO_GetAFPin_8_15
函数原形	__STATIC_INLINE uint32_t LL_GPIO_GetAFPin_8_15(GPIO_TypeDef *GPIOx, uint32_t Pin)
功能描述	获取指定引脚（8~15）的复用功能
输入参数 1	GPIOx：GPIO 端口
输入参数 2	Pin：GPIO 引脚
输出参数	无
返回值	复用功能
先决条件	无

GPIOx 可选参数：

表36-49 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

Pin 可选参数：

表36-50 Pin 可选参数

参数	描述
LL_GPIO_PIN_8	引脚 8
LL_GPIO_PIN_9	引脚 9
LL_GPIO_PIN_10	引脚 10
LL_GPIO_PIN_11	引脚 11
LL_GPIO_PIN_12	引脚 12
LL_GPIO_PIN_13	引脚 13
LL_GPIO_PIN_14	引脚 14
LL_GPIO_PIN_15	引脚 15

35.2.13 函数 LL_GPIO_LockPin

描述了函数 LL_GPIO_LockPin

表36-51 函数 LL_GPIO_LockPin

函数名	LL_GPIO_LockPin
函数原形	__STATIC_INLINE void LL_GPIO_LockPin(GPIO_TypeDef *GPIOx, uint32_t PinMask)
功能描述	锁定指定引脚的配置
输入参数 1	GPIOx：GPIO 端口
输入参数 2	PinMask：GPIO 引脚
输出参数	无
返回值	无
先决条件	无

GPIOx 可选参数：

表36-52 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

Pin 可选参数：

表36-53 Pin 可选参数

参数	描述
LL_GPIO_PIN_0	引脚 0
LL_GPIO_PIN_1	引脚 1
LL_GPIO_PIN_2	引脚 2
LL_GPIO_PIN_3	引脚 3
LL_GPIO_PIN_4	引脚 4
LL_GPIO_PIN_5	引脚 5
LL_GPIO_PIN_6	引脚 6
LL_GPIO_PIN_7	引脚 7
LL_GPIO_PIN_8	引脚 8
LL_GPIO_PIN_9	引脚 9
LL_GPIO_PIN_10	引脚 10
LL_GPIO_PIN_11	引脚 11
LL_GPIO_PIN_12	引脚 12
LL_GPIO_PIN_13	引脚 13
LL_GPIO_PIN_14	引脚 14
LL_GPIO_PIN_15	引脚 15

35.2.14 函数 LL_GPIO_IsPinLocked

描述了函数 LL_GPIO_IsPinLocked

表36-54 函数 LL_GPIO_IsPinLocked

函数名	LL_GPIO_IsPinLocked
函数原形	__STATIC_INLINE uint32_t LL_GPIO_IsPinLocked(GPIO_TypeDef *GPIOx, uint32_t PinMask)
功能描述	检查指定引脚的配置是否锁定
输入参数 1	GPIOx：GPIO 端口
输入参数 2	PinMask：GPIO 引脚
输出参数	无
返回值	锁定状态
先决条件	无

GPIOx 可选参数：

表36-55 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

Pin 可选参数：

表36-56 Pin 可选参数

参数	描述
LL_GPIO_PIN_0	引脚 0
LL_GPIO_PIN_1	引脚 1
LL_GPIO_PIN_2	引脚 2
LL_GPIO_PIN_3	引脚 3
LL_GPIO_PIN_4	引脚 4
LL_GPIO_PIN_5	引脚 5
LL_GPIO_PIN_6	引脚 6
LL_GPIO_PIN_7	引脚 7
LL_GPIO_PIN_8	引脚 8
LL_GPIO_PIN_9	引脚 9
LL_GPIO_PIN_10	引脚 10
LL_GPIO_PIN_11	引脚 11
LL_GPIO_PIN_12	引脚 12
LL_GPIO_PIN_13	引脚 13

LL_GPIO_PIN_14	引脚 14
LL_GPIO_PIN_15	引脚 15

35.2.15 函数 LL_GPIO_IsAnyPinLocked

描述了函数 LL_GPIO_IsAnyPinLocked

表36-57 函数 LL_GPIO_IsAnyPinLocked

函数名	LL_GPIO_IsAnyPinLocked
函数原形	__STATIC_INLINE uint32_t LL_GPIO_IsAnyPinLocked(GPIO_TypeDef *GPIOx)
功能描述	检查是否已锁定至少一个引脚的配置
输入参数	GPIOx：GPIO 端口
输出参数	无
返回值	锁定状态
先决条件	无

GPIOx 可选参数：

表36-58 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

35.2.16 函数 LL_GPIO_ReadInputPort

描述了函数 LL_GPIO_ReadInputPort

表36-59 函数 LL_GPIO_ReadInputPort

函数名	LL_GPIO_ReadInputPort
函数原形	__STATIC_INLINE uint32_t LL_GPIO_ReadInputPort(GPIO_TypeDef *GPIOx)
功能描述	获取指定端口的输入数据
输入参数	GPIOx：GPIO 端口
输出参数	无
返回值	输入数据
先决条件	无

GPIOx 可选参数：

表36-60 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B

GPIOF	GPIO 端口 F

35.2.17 函数 LL_GPIO_IsInputPinSet

描述了函数 LL_GPIO_IsInputPinSet

表36-61 函数 LL_GPIO_IsInputPinSet

函数名	LL_GPIO_IsInputPinSet
函数原形	__STATIC_INLINE uint32_t LL_GPIO_IsInputPinSet(GPIO_TypeDef *GPIOx, uint32_t PinMask)
功能描述	检查指定引脚是否输入高电平
输入参数 1	GPIOx：GPIO 端口
输入参数 2	PinMask：GPIO 引脚
输出参数	无
返回值	电平状态
先决条件	无

GPIOx 可选参数：

表36-62 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

Pin 可选参数：

表36-63 Pin 可选参数

参数	描述
LL_GPIO_PIN_0	引脚 0
LL_GPIO_PIN_1	引脚 1
LL_GPIO_PIN_2	引脚 2
LL_GPIO_PIN_3	引脚 3
LL_GPIO_PIN_4	引脚 4
LL_GPIO_PIN_5	引脚 5
LL_GPIO_PIN_6	引脚 6
LL_GPIO_PIN_7	引脚 7
LL_GPIO_PIN_8	引脚 8
LL_GPIO_PIN_9	引脚 9
LL_GPIO_PIN_10	引脚 10
LL_GPIO_PIN_11	引脚 11

LL_GPIO_PIN_12	引脚 12
LL_GPIO_PIN_13	引脚 13
LL_GPIO_PIN_14	引脚 14
LL_GPIO_PIN_15	引脚 15

35.2.18 函数 LL_GPIO_WriteOutputPort

描述了函数 LL_GPIO_WriteOutputPort

表36-64 函数 LL_GPIO_WriteOutputPort

函数名	LL_GPIO_WriteOutputPort
函数原形	__STATIC_INLINE void LL_GPIO_WriteOutputPort(GPIO_TypeDef *GPIOx, uint32_t PortValue)
功能描述	配置指定端口的输出数据
输入参数 1	GPIOx：GPIO 端口
输入参数 2	PortValue：输出数据
输出参数	无
返回值	无
先决条件	无

GPIOx 可选参数：

表36-65 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

35.2.19 函数 LL_GPIO_ReadOutputPort

描述了函数 LL_GPIO_ReadOutputPort

表36-66 函数 LL_GPIO_ReadOutputPort

函数名	LL_GPIO_ReadOutputPort
函数原形	__STATIC_INLINE uint32_t LL_GPIO_ReadOutputPort(GPIO_TypeDef *GPIOx)
功能描述	获取指定端口的输出数据
输入参数	GPIOx：GPIO 端口
输出参数	无
返回值	输出数据
先决条件	无

GPIOx 可选参数：

表36-67 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

35.2.20 函数 LL_GPIO_IsOutputPinSet

描述了函数 LL_GPIO_IsOutputPinSet

表36-68 函数 LL_GPIO_IsOutputPinSet

函数名	LL_GPIO_IsOutputPinSet
函数原形	__STATIC_INLINE uint32_t LL_GPIO_IsOutputPinSet(GPIO_TypeDef *GPIOx, uint32_t PinMask)
功能描述	检查指定引脚是否输出高电平
输入参数 1	GPIOx：GPIO 端口
输入参数 2	PinMask：GPIO 引脚
输出参数	无
返回值	电平状态
先决条件	无

GPIOx 可选参数:

表36-69 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

Pin 可选参数:

表36-70 Pin 可选参数

参数	描述
LL_GPIO_PIN_0	引脚 0
LL_GPIO_PIN_1	引脚 1
LL_GPIO_PIN_2	引脚 2
LL_GPIO_PIN_3	引脚 3
LL_GPIO_PIN_4	引脚 4
LL_GPIO_PIN_5	引脚 5
LL_GPIO_PIN_6	引脚 6
LL_GPIO_PIN_7	引脚 7
LL_GPIO_PIN_8	引脚 8

LL_GPIO_PIN_9	引脚 9
LL_GPIO_PIN_10	引脚 10
LL_GPIO_PIN_11	引脚 11
LL_GPIO_PIN_12	引脚 12
LL_GPIO_PIN_13	引脚 13
LL_GPIO_PIN_14	引脚 14
LL_GPIO_PIN_15	引脚 15

35.2.21 函数 LL_GPIO_SetOutputPin

描述了函数 LL_GPIO_SetOutputPin

表36-71 函数 LL_GPIO_SetOutputPin

函数名	LL_GPIO_SetOutputPin
函数原形	__STATIC_INLINE void LL_GPIO_SetOutputPin(GPIO_TypeDef *GPIOx, uint32_t PinMask)
功能描述	配置指定引脚输出高电平
输入参数 1	GPIOx：GPIO 端口
输入参数 2	PinMask：GPIO 引脚
输出参数	无
返回值	无
先决条件	无

GPIOx 可选参数：

表36-72 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

Pin 可选参数：

表36-73 Pin 可选参数

参数	描述
LL_GPIO_PIN_0	引脚 0
LL_GPIO_PIN_1	引脚 1
LL_GPIO_PIN_2	引脚 2
LL_GPIO_PIN_3	引脚 3
LL_GPIO_PIN_4	引脚 4
LL_GPIO_PIN_5	引脚 5

LL_GPIO_PIN_6	引脚 6
LL_GPIO_PIN_7	引脚 7
LL_GPIO_PIN_8	引脚 8
LL_GPIO_PIN_9	引脚 9
LL_GPIO_PIN_10	引脚 10
LL_GPIO_PIN_11	引脚 11
LL_GPIO_PIN_12	引脚 12
LL_GPIO_PIN_13	引脚 13
LL_GPIO_PIN_14	引脚 14
LL_GPIO_PIN_15	引脚 15

35.2.22 函数 LL_GPIO_ResetOutputPin

描述了函数 LL_GPIO_ResetOutputPin

表36-74 函数 LL_GPIO_ResetOutputPin

函数名	LL_GPIO_ResetOutputPin
函数原形	__STATIC_INLINE void LL_GPIO_ResetOutputPin(GPIO_TypeDef *GPIOx, uint32_t PinMask)
功能描述	配置指定引脚输出低电平
输入参数 1	GPIOx：GPIO 端口
输入参数 2	PinMask：GPIO 引脚
输出参数	无
返回值	无
先决条件	无

GPIOx 可选参数：

表36-75 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

Pin 可选参数：

表36-76 Pin 可选参数

参数	描述
LL_GPIO_PIN_0	引脚 0
LL_GPIO_PIN_1	引脚 1
LL_GPIO_PIN_2	引脚 2

LL_GPIO_PIN_3	引脚 3
LL_GPIO_PIN_4	引脚 4
LL_GPIO_PIN_5	引脚 5
LL_GPIO_PIN_6	引脚 6
LL_GPIO_PIN_7	引脚 7
LL_GPIO_PIN_8	引脚 8
LL_GPIO_PIN_9	引脚 9
LL_GPIO_PIN_10	引脚 10
LL_GPIO_PIN_11	引脚 11
LL_GPIO_PIN_12	引脚 12
LL_GPIO_PIN_13	引脚 13
LL_GPIO_PIN_14	引脚 14
LL_GPIO_PIN_15	引脚 15

35.2.23 函数 LL_GPIO_TogglePin

描述了函数 LL_GPIO_TogglePin

表36-77 函数 LL_GPIO_TogglePin

函数名	LL_GPIO_TogglePin
函数原形	__STATIC_INLINE void LL_GPIO_TogglePin(GPIO_TypeDef *GPIOx, uint32_t PinMask)
功能描述	配置指定引脚输出翻转
输入参数 1	GPIOx：GPIO 端口
输入参数 2	PinMask：GPIO 引脚
输出参数	无
返回值	无
先决条件	无

GPIOx 可选参数：

表36-78 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

Pin 可选参数：

表36-79 Pin 可选参数

参数	描述

LL_GPIO_PIN_0	引脚 0
LL_GPIO_PIN_1	引脚 1
LL_GPIO_PIN_2	引脚 2
LL_GPIO_PIN_3	引脚 3
LL_GPIO_PIN_4	引脚 4
LL_GPIO_PIN_5	引脚 5
LL_GPIO_PIN_6	引脚 6
LL_GPIO_PIN_7	引脚 7
LL_GPIO_PIN_8	引脚 8
LL_GPIO_PIN_9	引脚 9
LL_GPIO_PIN_10	引脚 10
LL_GPIO_PIN_11	引脚 11
LL_GPIO_PIN_12	引脚 12
LL_GPIO_PIN_13	引脚 13
LL_GPIO_PIN_14	引脚 14
LL_GPIO_PIN_15	引脚 15

35.2.24 函数 LL_GPIO_DeInit

描述了函数 LL_GPIO_DeInit

表36-80　函数 LL_GPIO_DeInit

函数名	LL_GPIO_DeInit
函数原形	ErrorStatus LL_GPIO_DeInit(GPIO_TypeDef *GPIOx)
功能描述	将 GPIOx 配置重设为缺省值
输入参数	GPIOx：GPIO 端口
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

GPIOx 可选参数:

表36-81　GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

35.2.25 函数 LL_GPIO_Init

描述了函数 LL_GPIO_Init

表36-82 函数 LL_GPIO_Init

函数名	LL_GPIO_Init
函数原形	ErrorStatus LL_GPIO_Init(GPIO_TypeDef *GPIOx, LL_GPIO_InitTypeDef *GPIO_InitStruct)
功能描述	初始化指定 GPIO 引脚
输入参数 1	GPIOx：GPIO 端口
输入参数 2	GPIO_InitStruct：初始化参数结构体
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

GPIOx 可选参数：

表36-83 GPIOx 可选参数

参数	描述
GPIOA	GPIO 端口 A
GPIOB	GPIO 端口 B
GPIOF	GPIO 端口 F

35.2.26 函数 LL_GPIO_StructInit

描述了函数 LL_GPIO_StructInit

表36-84 函数 LL_GPIO_StructInit

函数名	LL_GPIO_StructInit
函数原形	void LL_GPIO_StructInit(LL_GPIO_InitTypeDef *GPIO_InitStruct)
功能描述	初始化 GPIO_InitStruct 结构体
输入参数	GPIO_InitStruct：初始化参数结构体
输出参数	无
返回值	无
先决条件	无

36 LL 内部集成电路总线通用驱动程序（I2C）

I2C（内部集成电路）总线接口处理微控制器与串行 I2C 总线间的通信。它提供多主模式功能，可以控制所有 I2C 总线特定的序列、协议、仲裁和时序。它支持标准模式 (Sm)、快速模式 (Fm)。

36.1 I2C 固件驱动寄存器结构

36.1.1 LL_I2C_InitTypeDef

```
typedef struct
{
uint32_t ClockSpeed;
uint32_t DutyCycle;
uint32_t OwnAddress1;
  uint32_t TypeAcknowledge;
} LL_I2C_InitTypeDef;
```

字段说明：

表37-1 LL_I2C_InitTypeDef 字段说明

字段	描述
ClockSpeed	指定时钟频率（最高 400khz）
DutyCycle	指定 I2C 快速模式占空比
OwnAddress1	指定设备自己的地址
TypeAcknowledge	指定是否使能应答

参数说明：

DutyCycle 可选参数：

表37-2 DutyCycle 可选参数

参数	描述
LL_I2C_DUTYCYCLE_2	I2C 快速模式 Tlow/Thigh = 2
LL_I2C_DUTYCYCLE_16_9	I2C fast mode Tlow/Thigh = 16/9

TypeAcknowledge 可选参数：

表37-3 TypeAcknowledge 可选参数

参数	描述
LL_I2C_ACK	使能应答
LL_I2C_NACK	禁用应答

36.2 I2C 固件库函数

表37-4 I2C 固件库函数说明

函数名	描述
LL_I2C_Enable	使能 I2C
LL_I2C_Disable	禁用 I2C
LL_I2C_IsEnabled	检查 I2C 是否使能
LL_I2C_EnableDMAReq_TX	使能DMA 发送
LL_I2C_DisableDMAReq_TX	禁用DMA 发送
LL_I2C_IsEnabledDMAReq_TX	检查是否使能DMA 发送
LL_I2C_EnableDMAReq_RX	使能DMA 接收
LL_I2C_DisableDMAReq_RX	禁用DMA 接收
LL_I2C_IsEnabledDMAReq_RX	检查是否使能DMA 接收
LL_I2C_DMA_GetRegAddr	获取数据寄存器地址
LL_I2C_EnableClockStretching	使能时钟延长
LL_I2C_DisableClockStretching	禁用时钟延长
LL_I2C_IsEnabledClockStretching	检查是否使能时钟延长
LL_I2C_EnableGeneralCall	使能广播
LL_I2C_DisableGeneralCall	禁用广播
LL_I2C_IsEnabledGeneralCall	检查是否使能广播
LL_I2C_SetOwnAddress1	设置自身地址
LL_I2C_SetPeriphClock	设置外设时钟频率
LL_I2C_GetPeriphClock	获取外设时钟频率
LL_I2C_SetDutyCycle	设置占空比（仅快速模式）
LL_I2C_GetDutyCycle	获取占空比（仅快速模式）
LL_I2C_SetClockSpeedMode	设置时钟速度模式
LL_I2C_GetClockSpeedMode	获取时钟速度模式
LL_I2C_SetRiseTime	设置SCL、SDA 上升时间
LL_I2C_GetRiseTime	获取SCL、SDA 上升时间
LL_I2C_SetClockPeriod	设置SCL 高低周期
LL_I2C_GetClockPeriod	获取SCL 高低周期
LL_I2C_ConfigSpeed	设置SCL 速度

LL_I2C_EnableIT_TX	使能发送空中断
LL_I2C_DisableIT_TX	禁用发送空中断
LL_I2C_IsEnabledIT_TX	检查是否使能发送空中断
LL_I2C_EnableIT_RX	使能接收数据寄存器非空中断
LL_I2C_DisableIT_RX	禁用接收数据寄存器非空中断
LL_I2C_IsEnabledIT_RX	检查是否使能接收数据寄存器非空中断
LL_I2C_EnableIT_EVT	使能事件中断
LL_I2C_DisableIT_EVT	禁用事件中断
LL_I2C_IsEnabledIT_EVT	检查是否使能事件中断
LL_I2C_EnableIT_BUF	使能BUF 中断
LL_I2C_DisableIT_BUF	禁用BUF 中断
LL_I2C_IsEnabledIT_BUF	检查是否使能BUF 中断
LL_I2C_EnableIT_ERR	使能错误中断
LL_I2C_DisableIT_ERR	禁用错误中断
LL_I2C_IsEnabledIT_ERR	检查是否使能错误中断
LL_I2C_IsActiveFlag_TXE	检查发送数据寄存器空标志是否置位
LL_I2C_IsActiveFlag_BTF	检查字节传输完成标志是否置位
LL_I2C_IsActiveFlag_RXNE	检查接收数据寄存器非空标志是否置位
LL_I2C_IsActiveFlag_SB	检查起始位标志是否置位
LL_I2C_IsActiveFlag_ADDR	检查地址标志是否置位
LL_I2C_IsActiveFlag_AF	检查应答失败标志是否置位
LL_I2C_IsActiveFlag_STOP	检查停止标志是否置位
LL_I2C_IsActiveFlag_BERR	检查总线错误标志是否置位
LL_I2C_IsActiveFlag_ARLO	检查仲裁丢失标志是否置位
LL_I2C_IsActiveFlag_OVR	检查过载/欠载标志是否置位
LL_I2C_IsActiveSMBusFlag_PECERR	检查PEC 错误标志是否置位
LL_I2C_IsActiveFlag_BUSY	检查总线忙标志是否置位
LL_I2C_IsActiveFlag_GENCALL	检查收到广播地址标志是否置位
LL_I2C_IsActiveFlag_MSL	检查是否是主机模式
LL_I2C_ClearFlag_ADDR	清除地址标志
LL_I2C_ClearFlag_AF	清除应答失败标志

LL_I2C_ClearFlag_STOP	清除停止标志
LL_I2C_ClearFlag_BERR	我那个车总线错误标志
LL_I2C_ClearFlag_ARLO	清除仲裁丢失标志
LL_I2C_ClearFlag_OVR	清除过载/欠载标志
LL_I2C_ClearSMBusFlag_PECERR	清除PEC 错误标志
LL_I2C_EnableReset	使能 I2C 复位
LL_I2C_DisableReset	禁用 I2C 复位
LL_I2C_IsResetEnabled	检查 I2C 复位是否使能
LL_I2C_AcknowledgeNextData	设置应答/非应答
LL_I2C_GenerateStartCondition	产生起始条件
LL_I2C_GenerateStopCondition	产生停止条件
LL_I2C_EnableBitPOS	使能POS
LL_I2C_DisableBitPOS	禁用POS
LL_I2C_IsEnabledBitPOS	检查是否使能POS
LL_I2C_GetTransferDirection	获取传输方向
LL_I2C_EnableLastDMA	使能DMA 最后一次传输
LL_I2C_DisableLastDMA	禁用DMA 最后一次传输
LL_I2C_IsEnabledLastDMA	是否使能DMA 最后一次传输
LL_I2C_ReceiveData8	接收 8 位数据
LL_I2C_TransmitData8	发送 8 位数据
LL_I2C_Init	初始化 I2C
LL_I2C_DeInit	将 I2C 配置重设为缺省值
LL_I2C_StructInit	将结构体 LL_I2C_InitTypeDef 字段设置为默认值

36.2.1 函数 LL_I2C_Enable

描述了函数 LL_I2C_Enable

表37-5 函数 LL_I2C_Enable

函数名	LL_I2C_Enable
函数原形	__STATIC_INLINE void LL_I2C_Enable(I2C_TypeDef *I2Cx)
功能描述	使能 I2C
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无

先决条件	无

36.2.2 函数 LL_I2C_Disable

描述了函数 LL_I2C_Disable

表37-6 函数 LL_I2C_Disable

函数名	LL_I2C_Disable
函数原形	__STATIC_INLINE void LL_I2C_Disable (I2C_TypeDef *I2Cx)
功能描述	禁用 I2C
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.3 函数 LL_I2C_IsEnabled

描述了函数 LL_I2C_IsEnabled

表37-7 函数 LL_I2C_IsEnabled

函数名	LL_I2C_IsEnabled
函数原形	__STATIC_INLINE void LL_I2C_IsEnabled (I2C_TypeDef *I2Cx)
功能描述	检查是否使能 I2C
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.4 函数 LL_I2C_EnableDMAReq_TX

描述了函数 LL_I2C_EnableDMAReq_TX

表37-8 函数 LL_I2C_EnableDMAReq_TX

函数名	LL_I2C_EnableDMAReq_TX
函数原形	__STATIC_INLINE void LL_I2C_EnableDMAReq_TX (I2C_TypeDef *I2Cx)
功能描述	使能 DMA 发送
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.5 函数 LL_I2C_DisableDMAReq_TX

描述了函数 LL_I2C_DisableDMAReq_TX

表37-9 函数 LL_I2C_DisableDMAReq_TX

函数名	LL_I2C_DisableDMAReq_TX
函数原形	__STATIC_INLINE void LL_I2C_DisableDMAReq_TX (I2C_TypeDef *I2Cx)
功能描述	禁用 DMA 发送
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.6 函数 LL_I2C_IsEnabledDMAReq_TX

描述了函数 LL_I2C_IsEnabledDMAReq_TX

表37-10 函数 LL_I2C_IsEnabledDMAReq_TX

函数名	LL_I2C_IsEnabledDMAReq_TX
函数原形	__STATIC_INLINE void LL_I2C_IsEnabledDMAReq_TX (I2C_TypeDef *I2Cx)
功能描述	检查是否使能 DMA 发送
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.7 函数 LL_I2C_EnableDMAReq_RX

描述了函数 LL_I2C_EnableDMAReq_RX

表37-11 函数 LL_I2C_EnableDMAReq_RX

函数名	LL_I2C_EnableDMAReq_RX
函数原形	__STATIC_INLINE void LL_I2C_EnableDMAReq_RX (I2C_TypeDef *I2Cx)
功能描述	使能 DMA 接收
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.8 函数 LL_I2C_DisableDMAReq_RX

描述了函数 LL_I2C_DisableDMAReq_RX

表37-12 函数 LL_I2C_DisableDMAReq_RX

函数名	LL_I2C_DisableDMAReq_RX
函数原形	__STATIC_INLINE void LL_I2C_DisableDMAReq_RX (I2C_TypeDef *I2Cx)

功能描述	禁用 DMA 接收
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.9 函数 LL_I2C_IsEnabledDMAReq_RX

描述了函数 LL_I2C_IsEnabledDMAReq_RX

表37-13 函数 LL_I2C_IsEnabledDMAReq_RX

函数名	LL_I2C_IsEnabledDMAReq_RX
函数原形	__STATIC_INLINE void LL_I2C_IsEnabledDMAReq_RX (I2C_TypeDef *I2Cx)
功能描述	检查是否使能 DMA 接收
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.10 函数 LL_I2C_DMA_GetRegAddr

描述了函数 LL_I2C_DMA_GetRegAddr

表37-14 函数 LL_I2C_DMA_GetRegAddr

函数名	LL_I2C_DMA_GetRegAddr
函数原形	__STATIC_INLINE void LL_I2C_DMA_GetRegAddr (I2C_TypeDef *I2Cx)
功能描述	获取 I2C 数据寄存器地址
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	I2C 数据寄存器地址
先决条件	无

36.2.11 函数 LL_I2C_EnableClockStretching

描述了函数 LL_I2C_EnableClockStretching

表37-15 函数 LL_I2C_EnableClockStretching

函数名	LL_I2C_EnableClockStretching
函数原形	__STATIC_INLINE void LL_I2C_EnableClockStretching (I2C_TypeDef *I2Cx)
功能描述	使能时钟延长
输入参数	I2Cx：I2C 实例
输出参数	无

返回值	无
先决条件	无

36.2.12 函数 LL_I2C_DisableClockStretching

描述了函数 LL_I2C_DisableClockStretching

表37-16 函数 LL_I2C_DisableClockStretching

函数名	LL_I2C_DisableClockStretching
函数原形	__STATIC_INLINE void LL_I2C_DisableClockStretching (I2C_TypeDef *I2Cx)
功能描述	禁用时钟延长
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.13 函数 LL_I2C_IsEnabledClockStretching

描述了函数 LL_I2C_IsEnabledClockStretching

表37-17 函数 LL_I2C_IsEnabledClockStretching

函数名	LL_I2C_IsEnabledClockStretching
函数原形	__STATIC_INLINE void LL_I2C_IsEnabledClockStretching (I2C_TypeDef *I2Cx)
功能描述	检查是否使能时钟延长
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.14 函数 LL_I2C_EnableGeneralCall

描述了函数 LL_I2C_EnableGeneralCall

表37-18 函数 LL_I2C_EnableGeneralCall

函数名	LL_I2C_EnableGeneralCall
函数原形	__STATIC_INLINE void LL_I2C_EnableGeneralCall (I2C_TypeDef *I2Cx)
功能描述	使能广播
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.15 函数 LL_I2C_DisableGeneralCall

描述了函数 LL_I2C_DisableGeneralCall

表37-19 函数 LL_I2C_DisableGeneralCall

函数名	LL_I2C_DisableGeneralCall
函数原形	__STATIC_INLINE void LL_I2C_DisableGeneralCall (I2C_TypeDef *I2Cx)
功能描述	禁用广播
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.16 函数 LL_I2C_IsEnabledGeneralCall

描述了函数 LL_I2C_IsEnabledGeneralCall

表37-20 函数 LL_I2C_IsEnabledGeneralCall

函数名	LL_I2C_IsEnabledGeneralCall
函数原形	__STATIC_INLINE void LL_I2C_IsEnabledGeneralCall (I2C_TypeDef *I2Cx)
功能描述	检查是否使能广播
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.17 函数 LL_I2C_SetOwnAddress1

描述了函数 LL_I2C_SetOwnAddress1

表37-21 函数 LL_I2C_SetOwnAddress1

函数名	LL_I2C_SetOwnAddress1
函数原形	__STATIC_INLINE void LL_I2C_SetOwnAddress1(I2C_TypeDef *I2Cx, uint32_t OwnAddress1, uint32_t OwnAddrSize)
功能描述	设置自身地址
输入参数 1	I2Cx：I2C 实例
输入参数 2	OwnAddress1：I2C 自身地址
输入参数 3	OwnAddrSize：未被使用的参数，传递 0
输出参数	无
返回值	无
先决条件	无

36.2.18 函数 LL_I2C_SetPeriphClock

描述了函数 LL_I2C_SetPeriphClock

表37-22 函数 LL_I2C_SetPeriphClock

函数名	LL_I2C_SetPeriphClock
函数原形	__STATIC_INLINE void LL_I2C_SetPeriphClock(I2C_TypeDef *I2Cx, uint32_t PeriphClock)
功能描述	设置外设时钟频率
输入参数 1	I2Cx：I2C 实例
输入参数 2	PeriphClock：外设时钟频率
输出参数	无
返回值	无
先决条件	无

36.2.19 函数 LL_I2C_GetPeriphClock

描述了函数 LL_I2C_GetPeriphClock

表37-23 函数 LL_I2C_GetPeriphClock

函数名	LL_I2C_GetPeriphClock
函数原形	__STATIC_INLINE uint32_t LL_I2C_GetPeriphClock(I2C_TypeDef *I2Cx)
功能描述	获取外设时钟频率
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	外设时钟频率
先决条件	无

36.2.20 函数 LL_I2C_SetDutyCycle

描述了函数 LL_I2C_SetDutyCycle

表37-24 函数 LL_I2C_SetDutyCycle

函数名	LL_I2C_SetDutyCycle
函数原形	__STATIC_INLINE void LL_I2C_SetDutyCycle(I2C_TypeDef *I2Cx, uint32_t DutyCycle)
功能描述	设置占空比（仅快速模式）
输入参数 1	I2Cx：I2C 实例
输入参数 2	DutyCycle：占空比
输出参数	无
返回值	无
先决条件	无

DutyCycle 可选参数：

表37-25 DutyCycle 可选参数

参数	描述
LL_I2C_DUTYCYCLE_2	快速模式下：T low /T high =2
LL_I2C_DUTYCYCLE_16_9	快速模式下：T low /T high =16/9

36.2.21 函数 LL_I2C_GetDutyCycle

描述了函数 LL_I2C_GetDutyCycle

表37-26 函数 LL_I2C_GetDutyCycle

函数名	LL_I2C_GetDutyCycle
函数原形	__STATIC_INLINE uint32_t LL_I2C_GetDutyCycle (I2C_TypeDef *I2Cx)
功能描述	获取占空比
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	占空比
先决条件	无

36.2.22 函数 LL_I2C_SetClockSpeedMode

描述了函数 LL_I2C_SetClockSpeedMode

表37-27 函数 LL_I2C_SetClockSpeedMode

函数名	LL_I2C_SetClockSpeedMode
函数原形	__STATIC_INLINE void LL_I2C_SetClockSpeedMode (I2C_TypeDef *I2Cx, uint32_t ClockSpeedMode)
功能描述	设置主机速度模式
输入参数 1	I2Cx：I2C 实例
输入参数 2	ClockSpeedMode：速度模式
输出参数	无
返回值	无
先决条件	无

ClockSpeedMode 可选参数：

表37-28 ClockSpeedMode 可选参数

参数	描述
LL_I2C_CLOCK_SPEED_STANDARD_MODE	正常模式
LL_I2C_CLOCK_SPEED_FAST_MODE	快速模式

36.2.23 函数 LL_I2C_GetClockSpeedMode

描述了函数 LL_I2C_GetClockSpeedMode

表37-29 函数 LL_I2C_GetClockSpeedMode

函数名	LL_I2C_GetClockSpeedMode
函数原形	__STATIC_INLINE uint32_t LL_I2C_GetClockSpeedMode (I2C_TypeDef *I2Cx)
功能描述	获取主机速度模式
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	主机速度模式
先决条件	无

36.2.24 函数 LL_I2C_SetRiseTime

描述了函数 LL_I2C_SetRiseTime

表37-30 函数 LL_I2C_SetRiseTime

函数名	LL_I2C_SetRiseTime
函数原形	__STATIC_INLINE void LL_I2C_SetRiseTime(I2C_TypeDef *I2Cx, uint32_t RiseTime)
功能描述	设置 SCL, SDA 上升时间
输入参数 1	I2Cx：I2C 实例
输入参数 2	RiseTime：SCL, SDA 上升时间
输出参数	无
返回值	无
先决条件	无

36.2.25 函数 LL_I2C_GetRiseTime

描述了函数 LL_I2C_GetRiseTime

表37-31 函数 LL_I2C_GetRiseTime

函数名	LL_I2C_GetRiseTime
函数原形	__STATIC_INLINE uint32_t LL_I2C_GetRiseTime (I2C_TypeDef *I2Cx)
功能描述	获取 SCL, SDA 上升时间
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	SCL, SDA 上升时间
先决条件	无

36.2.26 函数 LL_I2C_SetClockPeriod

描述了函数 LL_I2C_SetClockPeriod

表37-32 函数 LL_I2C_SetClockPeriod

函数名	LL_I2C_SetClockPeriod

函数原形	__STATIC_INLINE void LL_I2C_SetClockPeriod(I2C_TypeDef *I2Cx, uint32_t ClockPeriod)
功能描述	设置 SCL 高低周期
输入参数 1	I2Cx：I2C 实例
输入参数 2	ClockPeriod：SCL 高低周期
输出参数	无
返回值	无
先决条件	无

36.2.27 函数 LL_I2C_GetClockPeriod

描述了函数 LL_I2C_GetClockPeriod

表37-33 函数 LL_I2C_GetClockPeriod

函数名	LL_I2C_GetClockPeriod
函数原形	__STATIC_INLINE uint32_t LL_I2C_GetClockPeriod (I2C_TypeDef *I2Cx)
功能描述	获取 SCL 高低周期
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	SCL 高低周期
先决条件	无

36.2.28 函数 LL_I2C_ConfigSpeed

描述了函数 LL_I2C_ConfigSpeed

表37-34 函数 LL_I2C_ConfigSpeed

函数名	LL_I2C_ConfigSpeed
函数原形	__STATIC_INLINE void LL_I2C_ConfigSpeed(I2C_TypeDef *I2Cx, uint32_t PeriphClock, uint32_t ClockSpeed, uint32_t DutyCycle)
功能描述	设置 SCL 速度
输入参数 1	I2Cx：I2C 实例
输入参数 2	PeriphClock：外设时钟频率
输入参数 3	ClockSpeed：SCL 速度
输入参数 4	DutyCycle：占空比
输出参数	无
返回值	无
先决条件	无

DutyCycle 可选参数：

表37-35 DutyCycle 可选参数

参数	描述

LL_I2C_DUTYCYCLE_2	快速模式下：T low /T high =2
LL_I2C_DUTYCYCLE_16_9	快速模式下：T low /T high =16/9

36.2.29 函数 LL_I2C_EnableIT_TX

描述了函数 LL_I2C_EnableIT_TX

表37-36 函数 LL_I2C_EnableIT_TX

函数名	LL_I2C_EnableIT_TX
函数原形	__STATIC_INLINE void LL_I2C_EnableIT_TX (I2C_TypeDef *I2Cx)
功能描述	使能发送数据寄存器空中断
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.30 函数 LL_I2C_DisableIT_TX

描述了函数 LL_I2C_DisableIT_TX

表37-37 函数 LL_I2C_DisableIT_TX

函数名	LL_I2C_DisableIT_TX
函数原形	__STATIC_INLINE void LL_I2C_DisableIT_TX (I2C_TypeDef *I2Cx)
功能描述	禁用发送数据寄存器空中断
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.31 函数 LL_I2C_IsEnabledIT_TX

描述了函数 LL_I2C_IsEnabledIT_TX

表37-38 函数 LL_I2C_IsEnabledIT_TX

函数名	LL_I2C_IsEnabledIT_TX
函数原形	__STATIC_INLINE void LL_I2C_IsEnabledIT_TX (I2C_TypeDef *I2Cx)
功能描述	检查是否使能发送数据寄存器空中断
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.32 函数 LL_I2C_EnableIT_RX

描述了函数 LL_I2C_EnableIT_RX

表37-39 函数 LL_I2C_EnableIT_RX

函数名	LL_I2C_EnableIT_RX
函数原形	__STATIC_INLINE void LL_I2C_EnableIT_RX (I2C_TypeDef *I2Cx)
功能描述	使能接收数据寄存器非空中断
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.33 函数 LL_I2C_DisableIT_RX

描述了函数 LL_I2C_DisableIT_RX

表37-40 函数 LL_I2C_DisableIT_RX

函数名	LL_I2C_DisableIT_RX
函数原形	__STATIC_INLINE void LL_I2C_DisableIT_RX (I2C_TypeDef *I2Cx)
功能描述	禁用接收数据寄存器非空中断
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.34 函数 LL_I2C_IsEnabledIT_RX

描述了函数 LL_I2C_IsEnabledIT_RX

表37-41 函数 LL_I2C_IsEnabledIT_RX

函数名	LL_I2C_IsEnabledIT_RX
函数原形	__STATIC_INLINE void LL_I2C_IsEnabledIT_RX (I2C_TypeDef *I2Cx)
功能描述	检查是否使能接收数据寄存器非空中断
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.35 函数 LL_I2C_EnableIT_EVT

描述了函数 LL_I2C_EnableIT_EVT

表37-42 函数 LL_I2C_EnableIT_EVT

函数名	LL_I2C_EnableIT_EVT
函数原形	__STATIC_INLINE void LL_I2C_EnableIT_EVT (I2C_TypeDef *I2Cx)
功能描述	使能事件中断
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.36 函数 LL_I2C_DisableIT_EVT

描述了函数 LL_I2C_DisableIT_EVT

表37-43 函数 LL_I2C_DisableIT_EVT

函数名	LL_I2C_DisableIT_EVT
函数原形	__STATIC_INLINE void LL_I2C_DisableIT_EVT (I2C_TypeDef *I2Cx)
功能描述	禁用事件中断
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.37 函数 LL_I2C_IsEnabledIT_EVT

描述了函数 LL_I2C_IsEnabledIT_EVT

表37-44 函数 LL_I2C_IsEnabledIT_EVT

函数名	LL_I2C_IsEnabledIT_EVT
函数原形	__STATIC_INLINE void LL_I2C_IsEnabledIT_EVT (I2C_TypeDef *I2Cx)
功能描述	检查是否使能事件中断
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.38 函数 LL_I2C_EnableIT_BUF

描述了函数 LL_I2C_EnableIT_BUF

表37-45 函数 LL_I2C_EnableIT_BUF

函数名	LL_I2C_EnableIT_BUF
函数原形	__STATIC_INLINE void LL_I2C_EnableIT_BUF (I2C_TypeDef *I2Cx)
功能描述	使能 BUF 中断

输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.39 函数 LL_I2C_DisableIT_BUF

描述了函数 LL_I2C_DisableIT_BUF

表37-46 函数 LL_I2C_DisableIT_BUF

函数名	LL_I2C_DisableIT_BUF
函数原形	__STATIC_INLINE void LL_I2C_DisableIT_BUF (I2C_TypeDef *I2Cx)
功能描述	禁用 BUF 中断
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.40 函数 LL_I2C_IsEnabledIT_BUF

描述了函数 LL_I2C_IsEnabledIT_BUF

表37-47 函数 LL_I2C_IsEnabledIT_BUF

函数名	LL_I2C_IsEnabledIT_BUF
函数原形	__STATIC_INLINE void LL_I2C_IsEnabledIT_BUF (I2C_TypeDef *I2Cx)
功能描述	检查是否使能 BUF 中断
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.41 函数 LL_I2C_EnableIT_ERR

描述了函数 LL_I2C_EnableIT_ERR

表37-48 函数 LL_I2C_EnableIT_ERR

函数名	LL_I2C_EnableIT_ERR
函数原形	__STATIC_INLINE void LL_I2C_EnableIT_ERR (I2C_TypeDef *I2Cx)
功能描述	使能错误中断
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无

先决条件	无

36.2.42 函数 LL_I2C_DisableIT_ERR

描述了函数 LL_I2C_DisableIT_ERR

表37-49 函数 LL_I2C_DisableIT_ERR

函数名	LL_I2C_DisableIT_ERR
函数原形	__STATIC_INLINE void LL_I2C_DisableIT_ERR (I2C_TypeDef *I2Cx)
功能描述	禁用错误中断
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.43 函数 LL_I2C_IsEnabledIT_ERR

描述了函数 LL_I2C_IsEnabledIT_ERR

表37-50 函数 LL_I2C_IsEnabledIT_ERR

函数名	LL_I2C_IsEnabledIT_ERR
函数原形	__STATIC_INLINE void LL_I2C_IsEnabledIT_ERR (I2C_TypeDef *I2Cx)
功能描述	检查是否使能错误中断
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.44 函数 LL_I2C_IsActiveFlag_TXE

描述了函数 LL_I2C_IsActiveFlag_TXE

表37-51 函数 LL_I2C_IsActiveFlag_TXE

函数名	LL_I2C_IsActiveFlag_TXE
函数原形	__STATIC_INLINE void LL_I2C_IsActiveFlag_TXE (I2C_TypeDef *I2Cx)
功能描述	检查是否发送数据寄存器空标志置位
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.45 函数 LL_I2C_IsActiveFlag_BTF

描述了函数 LL_I2C_IsActiveFlag_BTF

表37-52 函数 LL_I2C_IsActiveFlag_BTF

函数名	LL_I2C_IsActiveFlag_BTF
函数原形	__STATIC_INLINE void LL_I2C_IsActiveFlag_BTF (I2C_TypeDef *I2Cx)
功能描述	检查是否字节传输完成标志置位
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.46 函数 LL_I2C_IsActiveFlag_RXNE

描述了函数 LL_I2C_IsActiveFlag_RXNE

表37-53 函数 LL_I2C_IsActiveFlag_RXNE

函数名	LL_I2C_IsActiveFlag_RXNE
函数原形	__STATIC_INLINE void LL_I2C_IsActiveFlag_RXNE (I2C_TypeDef *I2Cx)
功能描述	检查是否接收数据寄存器非空标志置位
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.47 函数 LL_I2C_IsActiveFlag_SB

描述了函数 LL_I2C_IsActiveFlag_SB

表37-54 函数 LL_I2C_IsActiveFlag_SB

函数名	LL_I2C_IsActiveFlag_SB
函数原形	__STATIC_INLINE void LL_I2C_IsActiveFlag_SB (I2C_TypeDef *I2Cx)
功能描述	检查是否起始位标志置位
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.48 函数 LL_I2C_IsActiveFlag_ADDR

描述了函数 LL_I2C_IsActiveFlag_ADDR

表37-55 函数 LL_I2C_IsActiveFlag_ADDR

函数名	LL_I2C_IsActiveFlag_ADDR
函数原形	__STATIC_INLINE void LL_I2C_IsActiveFlag_ADDR (I2C_TypeDef *I2Cx)

功能描述	检查是否地址标志置位
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.49 函数 LL_I2C_IsActiveFlag_AF

描述了函数 LL_I2C_IsActiveFlag_AF

表37-56 函数 LL_I2C_IsActiveFlag_AF

函数名	LL_I2C_IsActiveFlag_AF
函数原形	__STATIC_INLINE void LL_I2C_IsActiveFlag_AF (I2C_TypeDef *I2Cx)
功能描述	检查是否应答失败标志置位
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.50 函数 LL_I2C_IsActiveFlag_STOP

描述了函数 LL_I2C_IsActiveFlag_STOP

表37-57 函数 LL_I2C_IsActiveFlag_STOP

函数名	LL_I2C_IsActiveFlag_STOP
函数原形	__STATIC_INLINE void LL_I2C_IsActiveFlag_STOP (I2C_TypeDef *I2Cx)
功能描述	检查是否 STOP 标志置位
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.51 函数 LL_I2C_IsActiveFlag_BERR

描述了函数 LL_I2C_IsActiveFlag_BERR

表37-58 函数 LL_I2C_IsActiveFlag_BERR

函数名	LL_I2C_IsActiveFlag_BERR
函数原形	__STATIC_INLINE void LL_I2C_IsActiveFlag_BERR (I2C_TypeDef *I2Cx)
功能描述	检查是否总线错误标志置位
输入参数	I2Cx：I2C 实例
输出参数	无

返回值	状态位（1 或 0）
先决条件	无

36.2.52 函数 LL_I2C_IsActiveFlag_ARLO

描述了函数 LL_I2C_IsActiveFlag_ARLO

表37-59 函数 LL_I2C_IsActiveFlag_ARLO

函数名	LL_I2C_IsActiveFlag_ARLO
函数原形	__STATIC_INLINE void LL_I2C_IsActiveFlag_ARLO (I2C_TypeDef *I2Cx)
功能描述	检查是否仲裁丢失标志置位
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.53 函数 LL_I2C_IsActiveFlag_OVR

描述了函数 LL_I2C_IsActiveFlag_OVR

表37-60 函数 LL_I2C_IsActiveFlag_OVR

函数名	LL_I2C_IsActiveFlag_OVR
函数原形	__STATIC_INLINE void LL_I2C_IsActiveFlag_OVR (I2C_TypeDef *I2Cx)
功能描述	检查是否超载/签载标志置位
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.54 函数 LL_I2C_IsActiveSMBusFlag_PECERR

描述了函数 LL_I2C_IsActiveSMBusFlag_PECERR

表37-61 函数 LL_I2C_IsActiveSMBusFlag_PECERR

函数名	LL_I2C_IsActiveSMBusFlag_PECERR
函数原形	__STATIC_INLINE void LL_I2C_IsActiveSMBusFlag_PECERR (I2C_TypeDef *I2Cx)
功能描述	检查是否 PEC 错位标志置位
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.55 函数 LL_I2C_IsActiveFlag_BUSY

描述了函数 LL_I2C_IsActiveFlag_BUSY

表37-62 函数 LL_I2C_IsActiveFlag_BUSY

函数名	LL_I2C_IsActiveFlag_BUSY
函数原形	__STATIC_INLINE void LL_I2C_IsActiveFlag_BUSY (I2C_TypeDef *I2Cx)
功能描述	检查是否总线忙标志置位
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.56 函数 LL_I2C_IsActiveFlag_GENCALL

描述了函数 LL_I2C_IsActiveFlag_GENCALL

表37-63 函数 LL_I2C_IsActiveFlag_GENCALL

函数名	LL_I2C_IsActiveFlag_GENCALL
函数原形	__STATIC_INLINE void LL_I2C_IsActiveFlag_GENCALL (I2C_TypeDef *I2Cx)
功能描述	检查是否接收到广播地址（从机模式）
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.57 函数 LL_I2C_IsActiveFlag_MSL

描述了函数 LL_I2C_IsActiveFlag_MSL

表37-64 函数 LL_I2C_IsActiveFlag_MSL

函数名	LL_I2C_IsActiveFlag_MSL
函数原形	__STATIC_INLINE void LL_I2C_IsActiveFlag_MSL (I2C_TypeDef *I2Cx)
功能描述	检查是否是主机模式
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.58 函数 LL_I2C_ClearFlag_ADDR

描述了函数 LL_I2C_ClearFlag_ADDR

表37-65 函数 LL_I2C_ClearFlag_ADDR

函数名	LL_I2C_ClearFlag_ADDR
函数原形	__STATIC_INLINE void LL_I2C_ClearFlag_ADDR (I2C_TypeDef *I2Cx)
功能描述	清除地址标志
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.59 函数 LL_I2C_ClearFlag_AF

描述了函数 LL_I2C_ClearFlag_AF

表37-66　函数 LL_I2C_ClearFlag_AF

函数名	LL_I2C_ClearFlag_AF
函数原形	__STATIC_INLINE void LL_I2C_ClearFlag_AF (I2C_TypeDef *I2Cx)
功能描述	清除应答失败标志
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.60 函数 LL_I2C_ClearFlag_STOP

描述了函数 LL_I2C_ClearFlag_STOP

表37-67　函数 LL_I2C_ClearFlag_STOP

函数名	LL_I2C_ClearFlag_STOP
函数原形	__STATIC_INLINE void LL_I2C_ClearFlag_STOP (I2C_TypeDef *I2Cx)
功能描述	清除 stop 标志
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.61 函数 LL_I2C_ClearFlag_BERR

描述了函数 LL_I2C_ClearFlag_BERR

表37-68　函数 LL_I2C_ClearFlag_BERR

函数名	LL_I2C_ClearFlag_BERR
函数原形	__STATIC_INLINE void LL_I2C_ClearFlag_BERR (I2C_TypeDef *I2Cx)
功能描述	清除总线错误标志

输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.62 函数 LL_I2C_ClearFlag_ARLO

描述了函数 LL_I2C_ClearFlag_ARLO

表37-69 函数 LL_I2C_ClearFlag_ARLO

函数名	LL_I2C_ClearFlag_ARLO
函数原形	__STATIC_INLINE void LL_I2C_ClearFlag_ARLO (I2C_TypeDef *I2Cx)
功能描述	清除仲裁丢失标志
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.63 函数 LL_I2C_ClearFlag_OVR

描述了函数 LL_I2C_ClearFlag_OVR

表37-70 函数 LL_I2C_ClearFlag_OVR

函数名	LL_I2C_ClearFlag_OVR
函数原形	__STATIC_INLINE void LL_I2C_ClearFlag_OVR (I2C_TypeDef *I2Cx)
功能描述	清除超载/欠载标志
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.64 函数 LL_I2C_ClearSMBusFlag_PECERR

描述了函数 LL_I2C_ClearSMBusFlag_PECERR

表37-71 函数 LL_I2C_ClearSMBusFlag_PECERR

函数名	LL_I2C_ClearSMBusFlag_PECERR
函数原形	__STATIC_INLINE void LL_I2C_ClearSMBusFlag_PECERR (I2C_TypeDef *I2Cx)
功能描述	清除 PEC 错误标志
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无

先决条件	无

36.2.65 函数 LL_I2C_EnableReset

描述了函数 LL_I2C_EnableReset

表37-72 函数 LL_I2C_EnableReset

函数名	LL_I2C_EnableReset
函数原形	__STATIC_INLINE void LL_I2C_EnableReset (I2C_TypeDef *I2Cx)
功能描述	使能 I2C 复位
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.66 函数 LL_I2C_DisableReset

描述了函数 LL_I2C_DisableReset

表37-73 函数 LL_I2C_DisableReset

函数名	LL_I2C_DisableReset
函数原形	__STATIC_INLINE void LL_I2C_DisableReset (I2C_TypeDef *I2Cx)
功能描述	禁用 I2C 复位
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.67 函数 LL_I2C_IsResetEnabled

描述了函数 LL_I2C_IsResetEnabled

表37-74 函数 LL_I2C_IsResetEnabled

函数名	LL_I2C_IsResetEnabled
函数原形	__STATIC_INLINE void LL_I2C_IsResetEnabled (I2C_TypeDef *I2Cx)
功能描述	检查是否使能 I2C 复位
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.68 函数 LL_I2C_AcknowledgeNextData

描述了函数 LL_I2C_AcknowledgeNextData

表37-75 函数 LL_I2C_AcknowledgeNextData

函数名	LL_I2C_AcknowledgeNextData
函数原形	__STATIC_INLINE void LL_I2C_AcknowledgeNextData(I2C_TypeDef *I2Cx, uint32_t TypeAcknowledge)
功能描述	设置应答/非应答
输入参数 1	I2Cx：I2C 实例
输入参数 2	TypeAcknowledge：应答/非应答
输出参数	无
返回值	无
先决条件	无

TypeAcknowledge 可选参数：

表37-76 TypeAcknowledge 可选参数

参数	描述
LL_I2C_ACK	设置应答
LL_I2C_NACK	设置非应答

36.2.69 函数 LL_I2C_GenerateStartCondition

描述了函数 LL_I2C_GenerateStartCondition

表37-77 函数 LL_I2C_GenerateStartCondition

函数名	LL_I2C_GenerateStartCondition
函数原形	__STATIC_INLINE void LL_I2C_GenerateStartCondition (I2C_TypeDef *I2Cx)
功能描述	产生起始条件
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.70 函数 LL_I2C_GenerateStopCondition

描述了函数 LL_I2C_GenerateStopCondition

表37-78 函数 LL_I2C_GenerateStopCondition

函数名	LL_I2C_GenerateStopCondition
函数原形	__STATIC_INLINE void LL_I2C_GenerateStopCondition (I2C_TypeDef *I2Cx)
功能描述	产生停止条件
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无

先决条件	无

36.2.71 函数 LL_I2C_EnableBitPOS

描述了函数 LL_I2C_EnableBitPOS

表37-79 函数 LL_I2C_EnableBitPOS

函数名	LL_I2C_EnableBitPOS
函数原形	__STATIC_INLINE void LL_I2C_EnableBitPOS (I2C_TypeDef *I2Cx)
功能描述	使能 POS
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.72 函数 LL_I2C_DisableBitPOS

描述了函数 LL_I2C_DisableBitPOS

表37-80 函数 LL_I2C_DisableBitPOS

函数名	LL_I2C_DisableBitPOS
函数原形	__STATIC_INLINE void LL_I2C_DisableBitPOS (I2C_TypeDef *I2Cx)
功能描述	禁用 POS
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.73 函数 LL_I2C_IsEnabledBitPOS

描述了函数 LL_I2C_IsEnabledBitPOS

表37-81 函数 LL_I2C_IsEnabledBitPOS

函数名	LL_I2C_IsEnabledBitPOS
函数原形	__STATIC_INLINE void LL_I2C_IsEnabledBitPOS (I2C_TypeDef *I2Cx)
功能描述	检查是否使能 POS
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.74 函数 LL_I2C_GetTransferDirection

描述了函数 LL_I2C_GetTransferDirection

表37-82 函数 LL_I2C_GetTransferDirection

函数名	LL_I2C_GetTransferDirection
函数原形	__STATIC_INLINE void LL_I2C_GetTransferDirection (I2C_TypeDef *I2Cx)
功能描述	获取传输方向
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	传输方向
先决条件	无

36.2.75 函数 LL_I2C_EnableLastDMA

描述了函数 LL_I2C_EnableLastDMA

表37-83 函数 LL_I2C_EnableLastDMA

函数名	LL_I2C_EnableLastDMA
函数原形	__STATIC_INLINE void LL_I2C_EnableLastDMA (I2C_TypeDef *I2Cx)
功能描述	使能 DMA 最后一次传输
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.76 函数 LL_I2C_DisableLastDMA

描述了函数 LL_I2C_DisableLastDMA

表37-84 函数 LL_I2C_DisableLastDMA

函数名	LL_I2C_DisableLastDMA
函数原形	__STATIC_INLINE void LL_I2C_DisableLastDMA (I2C_TypeDef *I2Cx)
功能描述	禁用 DMA 最后一次传输
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	无
先决条件	无

36.2.77 函数 LL_I2C_IsEnabledLastDMA

描述了函数 LL_I2C_IsEnabledLastDMA

表37-85 函数 LL_I2C_IsEnabledLastDMA

函数名	LL_I2C_IsEnabledLastDMA
函数原形	__STATIC_INLINE void LL_I2C_IsEnabledLastDMA (I2C_TypeDef *I2Cx)

功能描述	检查是否使能 DMA 最后一次传输
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

36.2.78 函数 LL_I2C_ReceiveData8

描述了函数 LL_I2C_ReceiveData8

表37-86 函数 LL_I2C_ReceiveData8

函数名	LL_I2C_ReceiveData8
函数原形	__STATIC_INLINE uint8_t LL_I2C_ReceiveData8(I2C_TypeDef *I2Cx)
功能描述	接收 8 位数据
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	8 位数据
先决条件	无

36.2.79 函数 LL_I2C_TransmitData8

描述了函数 LL_I2C_TransmitData8

表37-87 函数 LL_I2C_TransmitData8

函数名	LL_I2C_TransmitData8
函数原形	__STATIC_INLINE void LL_I2C_TransmitData8(I2C_TypeDef *I2Cx, uint8_t Data)
功能描述	发送 8 位数据
输入参数 1	I2Cx：I2C 实例
输入参数 2	Data：8 位数据
输出参数	无
返回值	无
先决条件	无

36.2.80 函数 LL_I2C_Init

描述了函数 LL_I2C_Init

表37-88 函数 LL_I2C_Init

函数名	LL_I2C_Init
函数原形	uint32_t LL_I2C_Init(I2C_TypeDef *I2Cx, LL_I2C_InitTypeDef *I2C_InitStruct)
功能描述	初始化 I2C
输入参数 1	I2Cx：I2C 实例

输入参数 2	I2C_InitStruct：指向结构体 LL_I2C_InitTypeDef 的指针（保护 I2C 的配置信息）
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

36.2.81 函数 LL_I2C_DeInit

描述了函数 LL_I2C_DeInit

表37-89 函数 LL_I2C_DeInit

函数名	LL_I2C_DeInit
函数原形	uint32_t LL_I2C_DeInit(I2C_TypeDef *I2Cx)
功能描述	将 I2C 配置重设为缺省值
输入参数	I2Cx：I2C 实例
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

36.2.82 函数 LL_I2C_StructInit

描述了函数 LL_I2C_StructInit

表37-90 函数 LL_I2C_StructInit

函数名	LL_I2C_StructInit
函数原形	void LL_I2C_StructInit(LL_I2C_InitTypeDef *I2C_InitStruct)
功能描述	将结构体 LL_I2C_InitTypeDef 字段设置为默认值
输入参数	无
输出参数	I2C_InitStruct：指向结构体 LL_I2C_InitTypeDef 的指针（保护 I2C 的配置信息）
返回值	无
先决条件	无

37 LL 独立看门狗通用驱动程序（IWDG）

芯片内集成了一个 Independent watchdog（简称 IWDG），该模块具有安全性高、定时准确及使用灵活的优点。此独立看门狗外设可检测并解决由软件错误导致的故障，并在计数器达到给定的超时值时触发系统复位。

独立看门狗(IWDG)由其专用低速时钟(LSI)驱动，因此即便在主时钟发生故障时仍然保持工作状态。

IWDG 最适合应用于需要看门狗作为一个在主程序之外，能够完全独立工作，并且对时间精度要求较低的应用。

37.1 IWDG 固件库函数

表38-1 CRC 固件库函数说明

函数名	描述
LL_IWDG_Enable	启动独立看门狗
LL_IWDG_ReloadCounter	重新加载 IWDG 计数器
LL_IWDG_EnableWriteAccess	启用对 IWDG_PR、IWDG_RLR 和 IWDG_WINR 寄存器的写操作
LL_IWDG_DisableWriteAccess	禁用对 IWDG_PR、IWDG_RLR 和 IWDG_WINR 寄存器的写操作
LL_IWDG_SetPrescaler	设置预分频器
LL_IWDG_GetPrescaler	返回当前 IWDG 预分频器值
LL_IWDG_SetReloadCounter	设定 IWDG 计数器的重载值
LL_IWDG_GetReloadCounter	获取 IWDG 计数器的重载值
LL_IWDG_IsActiveFlag_PVU	检查预分频器值更新标志是否激活
LL_IWDG_IsActiveFlag_RVU	检查重载值更新标志是否激活
LL_IWDG_IsReady	检查是否所有的预分频器重新、重载值重新和窗口值更新标志都被重置

37.1.1 函数 LL_IWDG_Enable

描述了函数 LL_IWDG_Enable

表38-2 函数 LL_IWDG_Enable

函数名	LL_IWDG_Enable
函数原形	__STATIC_INLINE void LL_IWDG_Enable(IWDG_TypeDef *IWDGx)
功能描述	启动独立看门狗
输入参数	IWDGx：IWDG 实例

输出参数	无
返回值	无
先决条件	无

37.1.2 函数 LL_IWDG_ReloadCounter

描述了函数 LL_IWDG_ReloadCounter

表38-3 函数 LL_IWDG_ReloadCounter

函数名	LL_IWDG_ReloadCounter
函数原形	__STATIC_INLINE void LL_IWDG_ReloadCounter(IWDG_TypeDef *IWDGx)
功能描述	重新加载 IWDG 计数器
输入参数	IWDGx：IWDG 实例
输出参数	无
返回值	状态值
先决条件	无

37.1.3 函数 LL_IWDG_EnableWriteAccess

描述了函数 LL_IWDG_EnableWriteAccess

表38-4 函数 LL_IWDG_EnableWriteAccess

函数名	LL_IWDG_EnableWriteAccess
函数原形	__STATIC_INLINE void LL_IWDG_EnableWriteAccess(IWDG_TypeDef *IWDGx)
功能描述	启用对 IWDG_PR、IWDG_RLR 和 IWDG_WINR 寄存器的写操作
输入参数	IWDGx：IWDG 实例
输出参数	无
返回值	无
先决条件	无

37.1.4 函数 LL_IWDG_DisableWriteAccess

描述了函数 LL_IWDG_DisableWriteAccess

表38-5 函数 LL_IWDG_DisableWriteAccess

函数名	LL_IWDG_DisableWriteAccess
函数原形	__STATIC_INLINE void LL_IWDG_DisableWriteAccess(IWDG_TypeDef *IWDGx)
功能描述	禁用对 IWDG_PR、IWDG_RLR 和 IWDG_WINR 寄存器的写操作
输入参数	IWDGx：IWDG 实例
输出参数	无
返回值	无
先决条件	无

37.1.5 函数 LL_IWDG_SetPrescaler

描述了函数 LL_IWDG_SetPrescaler

表38-6 函数 LL_IWDG_SetPrescaler

函数名	LL_IWDG_SetPrescaler
函数原形	__STATIC_INLINE void LL_IWDG_SetPrescaler(IWDG_TypeDef *IWDGx, uint32_t Prescaler)
功能描述	设置预分频器
输入参数 1	IWDGx：IWDG 实例
输入参数 2	Prescaler：预分频值
输出参数	无
返回值	无
先决条件	无

37.1.6 函数 LL_IWDG_GetPrescaler

描述了函数 LL_IWDG_GetPrescaler

表38-7 函数 LL_IWDG_GetPrescaler

函数名	LL_IWDG_GetPrescaler
函数原形	__STATIC_INLINE uint32_t LL_IWDG_GetPrescaler(IWDG_TypeDef *IWDGx)
功能描述	返回当前 IWDG 预分频器值
输入参数	IWDGx：IWDG 实例
输出参数	无
返回值	IWDG 预分频器值
先决条件	无

37.1.7 函数 LL_IWDG_SetReloadCounter

描述了函数 LL_IWDG_SetReloadCounter

表38-8 函数 LL_IWDG_SetReloadCounter

函数名	LL_IWDG_SetReloadCounter
函数原形	__STATIC_INLINE void LL_IWDG_SetReloadCounter(IWDG_TypeDef *IWDGx, uint32_t Counter)
功能描述	设定 IWDG 计数器的重载值
输入参数 1	IWDGx：IWDG 实例
输入参数 2	Window ： IWDG 窗口值
输出参数	无
返回值	无
先决条件	无

37.1.8 函数 LL_IWDG_GetReloadCounter

描述了函数 LL_IWDG_GetReloadCounter

表38-9 函数 LL_IWDG_GetReloadCounter

函数名	LL_IWDG_GetReloadCounter
函数原形	__STATIC_INLINE uint32_t LL_IWDG_GetReloadCounter(IWDG_TypeDef *IWDGx)
功能描述	获取 IWDG 计数器的重载值
输入参数	IWDGx：IWDG 实例
输出参数	无
返回值	IWDG 计数器的重载值
先决条件	无

37.1.9 函数 LL_IWDG_IsActiveFlag_PVU

描述了函数 LL_IWDG_IsActiveFlag_PVU

表38-10 函数 LL_IWDG_IsActiveFlag_PVU

函数名	LL_IWDG_IsActiveFlag_PVU
函数原形	__STATIC_INLINE uint32_t LL_IWDG_IsActiveFlag_PVU(IWDG_TypeDef *IWDGx)
功能描述	检查预分频器值更新标志是否激活
输入参数	IWDGx：IWDG 实例
输出参数	无
返回值	状态值
先决条件	无

37.1.10 函数 LL_IWDG_IsActiveFlag_RVU

描述了函数 LL_IWDG_IsActiveFlag_RVU

表38-11 函数 LL_IWDG_IsActiveFlag_RVU

函数名	LL_IWDG_IsActiveFlag_RVU
函数原形	__STATIC_INLINE uint32_t LL_IWDG_IsActiveFlag_RVU(IWDG_TypeDef *IWDGx)
功能描述	检查重载值更新标志是否激活
输入参数	IWDGx：IWDG 实例
输出参数	无
返回值	状态值
先决条件	无

37.1.11 函数 LL_IWDG_IsReady

描述了函数 LL_IWDG_IsReady

表38-12 函数 LL_IWDG_IsReady

函数名	LL_IWDG_IsReady
函数原形	__STATIC_INLINE uint32_t LL_IWDG_IsReady(IWDG_TypeDef *IWDGx)
功能描述	检查是否所有的预分频器重新、重载值重新和窗口值更新标志都被重置
输入参数	IWDGx：IWDG 实例
输出参数	无
返回值	状态值
先决条件	无

38 LL 数码管控制器通用驱动程序（LED）

本芯片支持 1~4 个 8 段式共阴极 LED 数码管的控制功能。该控制器通过 4 个支持超大灌电流（80mA/60mA/40mA/20mA）的管脚输出，对应点亮 4 个 7 段式数码管，同一时间只点亮一个数字。

38.1 LED 固件驱动寄存器结构

38.1.1 LL_LED_InitTypeDef

LL_LED_InitTypeDef，定义于文档“air001xx_ll_led.h”如下：

```
typedef struct
{
uint32_t ComDrive;
uint32_t Prescaler;
uint32_t ComSelect;
uint32_t LightTime;
uint32_t DeadTime;
} LL_LED_InitTypeDef;
```

字段说明：

表39-1 LL_LED_InitTypeDef 字段说明

字段	描述
ComDrive	LED COM 驱动能力
Prescaler	LED 驱动时钟预分频值
ComSelect	打开COM 数量
LightTime	点亮 LED 时间（0x01~0xFF）
DeadTime	熄灭 LED 时间（0x01~0xFF）

参数说明：

ComDrive 可选参数：

表39-2 ComDrive 可选参数

参数	描述
LL_LED_COMDRIVE_LOW	弱驱动能力
LL_LED_COMDRIVE_HIGH	强驱动能力

ComSelect 可选参数：

表39-3 ComSelect 可选参数

参数	描述

LL_LED_COMSELECT_1COM	打开 1 个 COM
LL_LED_COMSELECT_2COM	打开 2 个 COM
LL_LED_COMSELECT_3COM	打开 3 个 COM
LL_LED_COMSELECT_4COM	打开 4 个 COM

38.2 LED 固件库函数

表39-4 LED 固件库函数说明

函数名	描述
LL_LED_SetComDrive	设置COM 驱动能力
LL_LED_GetComDrive	获取COM 驱动能力
LL_LED_EnableIT	使能 LED 中断
LL_LED_DisableIT	禁用 LED 中断
LL_LED_IsEnabledIT	检查 LED 中断是否启用
LL_LED_SetComNum	设置COM 口开启数量
LL_LED_GetComNum	获取COM 口开启数量
LL_LED_Enable	使能 LED
LL_LED_Disable	禁用 LED
LL_LED_IsEnabled	检查 LED 是否使能
LL_LED_SetPrescaler	设置预分频值
LL_LED_GetPrescaler	获取预分频值
LL_LED_SetLightAndDeadTime	设置点亮时间和熄灭时间
LL_LED_SetLightTime	设置点亮时间
LL_LED_SetDeadTime	设置熄灭时间
LL_LED_GetLightTime	获取点亮时间
LL_LED_GetDeadTime	获取熄灭时间
LL_LED_SetDisplayValue	设置 LED 显示值
LL_LED_GetDisplayValue	获取 LED 显示值
LL_LED_IsActiveFlag_IT	检查 IT 标志位是否置位
LL_LED_ClearFlag_IT	清除 IT 标志位
LL_LED_DeInit	将 LED 配置重设为缺省值
LL_LED_Init	初始化 LED
LL_LED_StructInit	设置 LED 初始化结构体为默认值

38.2.1 函数 LL_LED_SetComDrive

描述了函数 LL_LED_SetComDrive

表39-5 函数 LL_LED_SetComDrive

函数名	LL_LED_SetComDrive
函数原形	__STATIC_INLINE void LL_LED_SetComDrive(LED_TypeDef *LEDx, uint32_t ComDrive)
功能描述	设置 COM 驱动能力
输入参数 1	LEDx：LED 实例
输入参数 2	ComDrive：COM 驱动能力
输出参数	无
返回值	无
先决条件	无

ComDrive 可选参数:

表39-6 ComDrive 可选参数

参数	描述
LL_LED_COMDRIVE_LOW	弱驱动能力
LL_LED_COMDRIVE_HIGH	强驱动能力

38.2.2 函数 LL_LED_GetComDrive

描述了函数 LL_LED_GetComDrive

表39-7 函数 LL_LED_GetComDrive

函数名	LL_LED_GetComDrive
函数原形	__STATIC_INLINE uint32_t LL_LED_GetComDrive(LED_TypeDef *LEDx)
功能描述	获取 COM 驱动能力
输入参数	LEDx：LED 实例
输出参数	无
返回值	COM 驱动能力
先决条件	无

38.2.3 函数 LL_LED_EnableIT

描述了函数 LL_LED_EnableIT

表39-8 函数 LL_LED_EnableIT

函数名	LL_LED_EnableIT
函数原形	__STATIC_INLINE void LL_LED_EnableIT(LED_TypeDef *LEDx)
功能描述	使能 LED 中断
输入参数	LEDx：LED 实例

输出参数	无
返回值	无
先决条件	无

38.2.4 函数 LL_LED_DisableIT

描述了函数 LL_LED_DisableIT

表39-9 函数 LL_LED_DisableIT

函数名	LL_LED_DisableIT
函数原形	__STATIC_INLINE void LL_LED_DisableIT(LED_TypeDef *LEDx)
功能描述	禁用 LED 中断
输入参数	LEDx：LED 实例
输出参数	无
返回值	无
先决条件	无

38.2.5 函数 LL_LED_IsEnabledIT

描述了函数 LL_LED_IsEnabledIT

表39-10 函数 LL_LED_IsEnabledIT

函数名	LL_LED_IsEnabledIT
函数原形	__STATIC_INLINE uint32_t LL_LED_IsEnabledIT(LED_TypeDef *LEDx)
功能描述	检查 LED 中断是否启用
输入参数	LEDx：LED 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

38.2.6 函数 LL_LED_SetComNum

描述了函数 LL_LED_SetComNum

表39-11 函数 LL_LED_SetComNum

函数名	LL_LED_SetComNum
函数原形	__STATIC_INLINE void LL_LED_SetComNum(LED_TypeDef *LEDx, uint32_t ComNum)
功能描述	设置 COM 口开启数量
输入参数 1	LEDx：LED 实例
输入参数 2	ComNum：COM 口开启数量
输出参数	无
返回值	无

先决条件	无

ComSelect 可选参数：

表39-12　ComSelect 可选参数

参数	描述
LL_LED_COMSELECT_1COM	打开 1 个 COM
LL_LED_COMSELECT_2COM	打开 2 个 COM
LL_LED_COMSELECT_3COM	打开 3 个 COM
LL_LED_COMSELECT_4COM	打开 4 个 COM

38.2.7 函数 LL_LED_GetComNum

描述了函数 LL_LED_GetComNum

表39-13　函数 LL_LED_GetComNum

函数名	LL_LED_GetComNum
函数原形	__STATIC_INLINE uint32_t LL_LED_GetComNum(LED_TypeDef *LEDx)
功能描述	获取 COM 口开启数量
输入参数	LEDx：LED 实例
输出参数	无
返回值	COM 口开启数量
先决条件	无

38.2.8 函数 LL_LED_Enable

描述了函数 LL_LED_Enable

表39-14　函数 LL_LED_Enable

函数名	LL_LED_Enable
函数原形	__STATIC_INLINE void LL_LED_Enable(LED_TypeDef *LEDx)
功能描述	使能 LED
输入参数	LEDx：LED 实例
输出参数	无
返回值	无
先决条件	无

38.2.9 函数 LL_LED_Disable

描述了函数 LL_LED_Disable

表39-15　函数 LL_LED_Disable

函数名	LL_LED_Disable
函数原形	__STATIC_INLINE void LL_LED_Disable(LED_TypeDef *LEDx)

功能描述	禁用 LED
输入参数	LEDx：LED 实例
输出参数	无
返回值	无
先决条件	无

38.2.10 函数 LL_LED_IsEnabled

描述了函数 LL_LED_IsEnabled

表39-16 函数 LL_LED_IsEnabled

函数名	LL_LED_IsEnabled
函数原形	__STATIC_INLINE uint32_t LL_LED_IsEnabled(LED_TypeDef *LEDx)
功能描述	检查 LED 是否使能
输入参数	LEDx：LED 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

38.2.11 函数 LL_LED_SetPrescaler

描述了函数 LL_LED_SetPrescaler

表39-17 函数 LL_LED_SetPrescaler

函数名	LL_LED_SetPrescaler
函数原形	__STATIC_INLINE void LL_LED_SetPrescaler(LED_TypeDef *LEDx, uint32_t Prescaler)
功能描述	设置预分频值
输入参数 1	LEDx：LED 实例
输入参数 2	Prescaler：预分频值
输出参数	无
返回值	无
先决条件	无

38.2.12 函数 LL_LED_GetPrescaler

描述了函数 LL_LED_GetPrescaler

表39-18 函数 LL_LED_GetPrescaler

函数名	LL_LED_GetPrescaler
函数原形	__STATIC_INLINE uint32_t LL_LED_GetPrescaler(LED_TypeDef *LEDx)
功能描述	获取预分频值
输入参数	LEDx：LED 实例

输出参数	无
返回值	预分频值
先决条件	无

38.2.13 函数 LL_LED_SetLightAndDeadTime

描述了函数 LL_LED_SetLightAndDeadTime

表39-19 函数 LL_LED_SetLightAndDeadTime

函数名	LL_LED_SetLightAndDeadTime
函数原形	__STATIC_INLINE void LL_LED_SetLightAndDeadTime(LED_TypeDef *LEDx, uint32_t LightTime, uint32_t DeadTime)
功能描述	设置点亮时间和熄灭时间
输入参数 1	LEDx：LED 实例
输入参数 2	LightTime：LED 点亮时间（0x01~0xFF）
输入参数 3	DeadTime：LED 熄灭时间（0x01~0xFF）
输出参数	无
返回值	无
先决条件	无

38.2.14 函数 LL_LED_SetLightTime

描述了函数 LL_LED_SetLightTime

表39-20 函数 LL_LED_SetLightTime

函数名	LL_LED_SetLightTime
函数原形	__STATIC_INLINE void LL_LED_SetLightTime(LED_TypeDef *LEDx, uint32_t LightTime)
功能描述	设置点亮时间
输入参数 1	LEDx：LED 实例
输入参数 2	LightTime：LED 点亮时间（0x01~0xFF）
输出参数	无
返回值	无
先决条件	无

38.2.15 函数 LL_LED_SetDeadTime

描述了函数 LL_LED_SetDeadTime

表39-21 函数 LL_LED_SetDeadTime

函数名	LL_LED_SetDeadTime
函数原形	__STATIC_INLINE void LL_LED_SetDeadTime(LED_TypeDef *LEDx, uint32_t DeadTime)
功能描述	设置熄灭时间
输入参数 1	LEDx：LED 实例

输入参数 2	DeadTime：LED 熄灭时间（0x01~0xFF）
输出参数	无
返回值	无
先决条件	无

38.2.16 函数 LL_LED_GetLightTime

描述了函数 LL_LED_GetLightTime

表39-22 函数 LL_LED_GetLightTime

函数名	LL_LED_GetLightTime
函数原形	__STATIC_INLINE uint32_t LL_LED_GetLightTime(LED_TypeDef *LEDx)
功能描述	获取点亮时间
输入参数	LEDx：LED 实例
输出参数	无
返回值	LED 点亮时间
先决条件	无

38.2.17 函数 LL_LED_GetDeadTime

描述了函数 LL_LED_GetDeadTime

表39-23 函数 LL_LED_GetDeadTime

函数名	LL_LED_GetDeadTime
函数原形	__STATIC_INLINE uint32_t LL_LED_GetDeadTime(LED_TypeDef *LEDx)
功能描述	获取熄灭时间
输入参数	LEDx：LED 实例
输出参数	无
返回值	LED 熄灭时间
先决条件	无

38.2.18 函数 LL_LED_SetDisplayValue

描述了函数 LL_LED_SetDisplayValue

表39-24 函数 LL_LED_SetDisplayValue

函数名	LL_LED_SetDisplayValue
函数原形	__STATIC_INLINE void LL_LED_SetDisplayValue(LED_TypeDef *LEDx,uint32_t comCh, uint32_t data)
功能描述	设置 LED 显示值
输入参数 1	LEDx：LED 实例
输入参数 2	comCh：COM 通道
输入参数 3	Data：显示值

输出参数	无
返回值	无
先决条件	无

comCh 可选参数：

表39-25　comCh 可选参数

参数	描述
LL_LED_COM0	COM 通道 0
LL_LED_COM1	COM 通道 1
LL_LED_COM2	COM 通道 2
LL_LED_COM3	COM 通道 3

data 可选参数：

表39-26　data 可选参数

参数	描述
LL_LED_DISP_NONE	不显示
LL_LED_DISP_FULL	显示所有 LED 管
LL_LED_DISP_0	显示“0”
LL_LED_DISP_1	显示“1”
LL_LED_DISP_2	显示“2”
LL_LED_DISP_3	显示“3”
LL_LED_DISP_4	显示“4”
LL_LED_DISP_5	显示“5”
LL_LED_DISP_6	显示“6”
LL_LED_DISP_7	显示“7”
LL_LED_DISP_8	显示“8”
LL_LED_DISP_9	显示“9”
LL_LED_DISP_A	显示“A”
LL_LED_DISP_B	显示“B”
LL_LED_DISP_C	显示“C”
LL_LED_DISP_D	显示“D”
LL_LED_DISP_E	显示“E”
LL_LED_DISP_F	显示“F”
LL_LED_DISP_H	显示“H”
LL_LED_DISP_P	显示“P”
LL_LED_DISP_U	显示“U”
LL_LED_DISP_DOT	显示“.”

38.2.19 函数 LL_LED_GetDisplayValue

描述了函数 LL_LED_GetDisplayValue

表39-27 函数 LL_LED_GetDisplayValue

函数名	LL_LED_GetDisplayValue
函数原形	__STATIC_INLINE uint32_t LL_LED_GetDisplayValue(LED_TypeDef *LEDx, uint32_t comCh)
功能描述	获取 LED 显示值
输入参数	LEDx：LED 实例
输出参数	无
返回值	显示值
先决条件	无

comCh 可选参数:

表39-28 comCh 可选参数

参数	描述
LL_LED_COM0	COM 通道 0
LL_LED_COM1	COM 通道 1
LL_LED_COM2	COM 通道 2
LL_LED_COM3	COM 通道 3

38.2.20 函数 LL_LED_IsActiveFlag_IT

描述了函数 LL_LED_IsActiveFlag_IT

表39-29 函数 LL_LED_IsActiveFlag_IT

函数名	LL_LED_IsActiveFlag_IT
函数原形	__STATIC_INLINE uint32_t LL_LED_IsActiveFlag_IT(LED_TypeDef *LEDx)
功能描述	检查 IT 标志位是否置位
输入参数	LEDx：LED 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

38.2.21 函数 LL_LED_ClearFlag_IT

描述了函数 LL_LED_ClearFlag_IT

表39-30 函数 LL_LED_ClearFlag_IT

函数名	LL_LED_ClearFlag_IT
函数原形	__STATIC_INLINE void LL_LED_ClearFlag_IT(LED_TypeDef *LEDx)
功能描述	清除 IT 标志位

输入参数	LEDx：LED 实例
输出参数	无
返回值	无
先决条件	无

38.2.22 函数 LL_LED_DeInit

描述了函数 LL_LED_DeInit

表39-31 函数 LL_LED_DeInit

函数名	LL_LED_DeInit
函数原形	ErrorStatus LL_LED_DeInit(LED_TypeDef *LEDx)
功能描述	将 LED 配置重设为缺省值
输入参数	LEDx：LED 实例
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

38.2.23 函数 LL_LED_Init

描述了函数 LL_LED_Init

表39-32 函数 LL_LED_Init

函数名	LL_LED_Init
函数原形	ErrorStatus LL_LED_Init(LED_TypeDef *LEDx, LL_LED_InitTypeDef *LED_InitStruct)
功能描述	初始化 LED
输入参数 1	LEDx：LED 实例
输入参数 2	LED_InitStruct：LED 初始化结构体
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

38.2.24 函数 LL_LED_StructInit

描述了函数 LL_LED_StructInit

表39-33 函数 LL_LED_StructInit

函数名	LL_LED_StructInit
函数原形	void LL_LED_StructInit(LL_LED_InitTypeDef *LED_InitStruct)
功能描述	设置 LED 初始化结构体为默认值
输入参数	LED_InitStruct：LED 初始化结构体
输出参数	无

返回值	无
先决条件	无

39 LL 低功耗定时器通用驱动程序（LPTIM）

LPTIM 是一款 16 位定时器。LPTIM 将系统从低功耗模式中唤醒的能力使得它适合于实现低功耗应用。

LPTIM 引入了一种灵活的时钟方案，可提供所需的功能和性能，同时将功耗降至最低。

39.1 LPTIM 固件驱动寄存器结构

39.1.1 LL_LPTIM_InitTypeDef

LL_LPTIM_InitTypeDef，定于文件“air001xx_ll_lptim.h”如下：

```
typedef struct
{
uint32_t Prescaler;
uint32_t UpdateMode;
} LL_LPTIM_InitTypeDef;
```

字段说明：

表40-1 LL_LPTIM_InitTypeDef 字段说明

字段	描述
Prescaler	时钟预分频值
UpdateMode	重装值更新模式

参数说明：

Prescaler 可选参数：

表40-2 Prescaler 可选参数

参数	描述
LL_LPTIM_PRESCALER_DIV1	不分频
LL_LPTIM_PRESCALER_DIV2	2 分频
LL_LPTIM_PRESCALER_DIV4	4 分频
LL_LPTIM_PRESCALER_DIV8	8 分频
LL_LPTIM_PRESCALER_DIV16	16 分频
LL_LPTIM_PRESCALER_DIV32	32 分频
LL_LPTIM_PRESCALER_DIV64	64 分频
LL_LPTIM_PRESCALER_DIV128	128 分频

UpdateMode 可选参数：

表40-3 UpdateMode 可选参数

参数	描述

LL_LPTIM_UPDATE_MODE_IMMEDIATE	立即更新重装载值
LL_LPTIM_UPDATE_MODE_ENDOFPERIOD	当前周期结束后更新重装载值

39.2 LPTIM 固件库函数

表40-4 LPTIM 固件库函数说明

函数名	描述
LL_LPTIM_Enable	使能 LPTIM
LL_LPTIM_Disable	禁用 LPTIM
LL_LPTIM_IsEnabled	检查 LPTIM 是否使能
LL_LPTIM_StartCounter	开始计数
LL_LPTIM_EnableResetAfterRead	使能读取后复位
LL_LPTIM_DisableResetAfterRead	禁用读取后复位
LL_LPTIM_IsEnabledResetAfterRead	检查是否开启读取后复位
LL_LPTIM_SetUpdateMode	设置更新模式
LL_LPTIM_GetUpdateMode	获取更新模式
LL_LPTIM_SetAutoReload	设置自动重装载值
LL_LPTIM_GetAutoReload	获取自动重装载值
LL_LPTIM_GetCounter	获取计数值
LL_LPTIM_SetPrescaler	设置预分频值
LL_LPTIM_GetPrescaler	获取预分频值
LL_LPTIM_ClearFLAG_ARRM	清除ARRM 标志位
LL_LPTIM_IsActiveFlag_ARRM	检查ARRM 标志位是否置位
LL_LPTIM_EnableIT_ARRM	使能ARRM 中断
LL_LPTIM_DisableIT_ARRM	禁用ARRM 中断
LL_LPTIM_IsEnabledIT_ARRM	检查ARRM 中断是否开启
LL_LPTIM_DeInit	将 LPTIM 相关配置重设为缺省值
LL_LPTIM_StructInit	设置 LPTIM 初始化结构体为默认值
LL_LPTIM_Init	初始化 LPTIM

39.2.1 函数 LL_LPTIM_Enable

描述了函数 LL_LPTIM_Enable

表40-5 函数 LL_LPTIM_Enable

函数名	LL_LPTIM_Enable

函数原形	__STATIC_INLINE void LL_LPTIM_Enable(LPTIM_TypeDef *LPTIMx)
功能描述	使能 LPTIM
输入参数	LPTIMx：LPTIM 实例
输出参数	无
返回值	无
先决条件	无

39.2.2 函数 LL_LPTIM_Disable

描述了函数 LL_LPTIM_Disable

表40-6 函数 LL_LPTIM_Disable

函数名	LL_LPTIM_Disable
函数原形	__STATIC_INLINE void LL_LPTIM_Disable(LPTIM_TypeDef *LPTIMx)
功能描述	禁用 LPTIM
输入参数	LPTIMx：LPTIM 实例
输出参数	无
返回值	无
先决条件	无

39.2.3 函数 LL_LPTIM_IsEnabled

描述了函数 LL_LPTIM_IsEnabled

表40-7 函数 LL_LPTIM_IsEnabled

函数名	LL_LPTIM_IsEnabled
函数原形	__STATIC_INLINE uint32_t LL_LPTIM_IsEnabled(LPTIM_TypeDef *LPTIMx)
功能描述	检查 LPTIM 是否使能
输入参数	LPTIMx：LPTIM 实例
输出参数	无
返回值	无
先决条件	无

39.2.4 函数 LL_LPTIM_StartCounter

描述了函数 LL_LPTIM_StartCounter

表40-8 函数 LL_LPTIM_StartCounter

函数名	LL_LPTIM_StartCounter
函数原形	__STATIC_INLINE void LL_LPTIM_StartCounter(LPTIM_TypeDef *LPTIMx, uint32_t OperatingMode)
功能描述	开始计数
输入参数 1	LPTIMx：LPTIM 实例

输入参数 2	OperatingMode：计数模式
输出参数	无
返回值	无
先决条件	无

OperatingMode 可选参数:

表40-9 OperatingMode 可选参数

参数	描述
LL_LPTIM_OPERATING_MODE_ONESHOT	单次模式

39.2.5 函数 LL_LPTIM_EnableResetAfterRead

描述了函数 LL_LPTIM_EnableResetAfterRead

表40-10 函数 LL_LPTIM_EnableResetAfterRead

函数名	LL_LPTIM_EnableResetAfterRead
函数原形	__STATIC_INLINE void LL_LPTIM_EnableResetAfterRead(LPTIM_TypeDef *LPTIMx)
功能描述	使能读取后复位
输入参数 1	LPTIMx：LPTIM 实例
输出参数	无
返回值	无
先决条件	无

39.2.6 函数 LL_LPTIM_DisableResetAfterRead

描述了函数 LL_LPTIM_DisableResetAfterRead

表40-11 函数 LL_LPTIM_DisableResetAfterRead

函数名	LL_LPTIM_DisableResetAfterRead
函数原形	__STATIC_INLINE void LL_LPTIM_DisableResetAfterRead(LPTIM_TypeDef *LPTIMx)
功能描述	禁用读取后复位
输入参数	LPTIMx：LPTIM 实例
输出参数	无
返回值	无
先决条件	无

39.2.7 函数 LL_LPTIM_IsEnabledResetAfterRead

描述了函数 LL_LPTIM_IsEnabledResetAfterRead

表40-12 函数 LL_LPTIM_IsEnabledResetAfterRead

函数名	LL_LPTIM_IsEnabledResetAfterRead

函数原形	__STATIC_INLINE uint32_t LL_LPTIM_IsEnabledResetAfterRead(LPTIM_TypeDef *LPTIMx)
功能描述	检查是否开启读取后复位
输入参数	LPTIMx：LPTIM 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

39.2.8 函数 LL_LPTIM_SetUpdateMode

描述了函数 LL_LPTIM_SetUpdateMode

表40-13 函数 LL_LPTIM_SetUpdateMode

函数名	LL_LPTIM_SetUpdateMode
函数原形	__STATIC_INLINE void LL_LPTIM_SetUpdateMode(LPTIM_TypeDef *LPTIMx, uint32_t UpdateMode)
功能描述	设置更新模式
输入参数 1	LPTIMx：LPTIM 实例
输入参数 2	UpdateMode：更新模式
输出参数	无
返回值	无
先决条件	无

UpdateMode 可选参数：

表40-14 UpdateMode 可选参数

参数	描述
LL_LPTIM_UPDATE_MODE_IMMEDIATE	立即更新重装载值
LL_LPTIM_UPDATE_MODE_ENDOFPERIOD	当前周期结束后更新重装载值

39.2.9 函数 LL_LPTIM_GetUpdateMode

描述了函数 LL_LPTIM_GetUpdateMode

表40-15 函数 LL_LPTIM_GetUpdateMode

函数名	LL_LPTIM_GetUpdateMode
函数原形	__STATIC_INLINE uint32_t LL_LPTIM_GetUpdateMode(LPTIM_TypeDef *LPTIMx)
功能描述	获取更新模式
输入参数	LPTIMx：LPTIM 实例
输出参数	无
返回值	更新模式
先决条件	无

39.2.10 函数 LL_LPTIM_SetAutoReload

描述了函数 LL_LPTIM_SetAutoReload

表40-16 函数 LL_LPTIM_SetAutoReload

函数名	LL_LPTIM_SetAutoReload
函数原形	__STATIC_INLINE void LL_LPTIM_SetAutoReload(LPTIM_TypeDef *LPTIMx, uint32_t AutoReload)
功能描述	设置自动重装载值
输入参数 1	LPTIMx：LPTIM 实例
输入参数 2	AutoReload：自动重装载值
输出参数	无
返回值	无
先决条件	无

39.2.11 函数 LL_LPTIM_GetAutoReload

描述了函数 LL_LPTIM_GetAutoReload

表40-17 函数 LL_LPTIM_GetAutoReload

函数名	LL_LPTIM_GetAutoReload
函数原形	__STATIC_INLINE uint32_t LL_LPTIM_GetAutoReload(LPTIM_TypeDef *LPTIMx)
功能描述	获取自动重装载值
输入参数	LPTIMx：LPTIM 实例
输出参数	无
返回值	自动重装载值
先决条件	无

39.2.12 函数 LL_LPTIM_GetCounter

描述了函数 LL_LPTIM_GetCounter

表40-18 函数 LL_LPTIM_GetCounter

函数名	LL_LPTIM_GetCounter
函数原形	__STATIC_INLINE uint32_t LL_LPTIM_GetCounter(LPTIM_TypeDef *LPTIMx)
功能描述	获取计数值
输入参数	LPTIMx：LPTIM 实例
输出参数	无
返回值	当前计数值
先决条件	无

39.2.13 函数 LL_LPTIM_SetPrescaler

描述了函数 LL_LPTIM_SetPrescaler

表40-19 函数 LL_LPTIM_SetPrescaler

函数名	LL_LPTIM_SetPrescaler
函数原形	__STATIC_INLINE void LL_LPTIM_SetPrescaler(LPTIM_TypeDef *LPTIMx, uint32_t Prescaler)
功能描述	设置预分频值
输入参数	LPTIMx：LPTIM 实例
输入参数 2	Prescaler：预分频值
输出参数	无
返回值	无
先决条件	无

Prescaler 可选参数:

表40-20 Prescaler 可选参数

参数	描述
LL_LPTIM_PRESCALER_DIV1	不分频
LL_LPTIM_PRESCALER_DIV2	2 分频
LL_LPTIM_PRESCALER_DIV4	4 分频
LL_LPTIM_PRESCALER_DIV8	8 分频
LL_LPTIM_PRESCALER_DIV16	16 分频
LL_LPTIM_PRESCALER_DIV32	32 分频
LL_LPTIM_PRESCALER_DIV64	64 分频
LL_LPTIM_PRESCALER_DIV128	128 分频

39.2.14 函数 LL_LPTIM_GetPrescaler

描述了函数 LL_LPTIM_GetPrescaler

表40-21 函数 LL_LPTIM_GetPrescaler

函数名	LL_LPTIM_GetPrescaler
函数原形	__STATIC_INLINE uint32_t LL_LPTIM_GetPrescaler(LPTIM_TypeDef *LPTIMx)
功能描述	获取预分频值
输入参数	LPTIMx：LPTIM 实例
输出参数	无
返回值	预分频值
先决条件	无

39.2.15 函数 LL_LPTIM_ClearFLAG_ARRM

描述了函数 LL_LPTIM_ClearFLAG_ARRM

表40-22 函数 LL_LPTIM_ClearFLAG_ARRM

函数名	LL_LPTIM_ClearFLAG_ARRM
函数原形	__STATIC_INLINE void LL_LPTIM_ClearFLAG_ARRM(LPTIM_TypeDef *LPTIMx)
功能描述	清除 ARRM 标志位
输入参数	LPTIMx：LPTIM 实例
输出参数	无
返回值	无
先决条件	无

39.2.16 函数 LL_LPTIM_IsActiveFlag_ARRM

描述了函数 LL_LPTIM_IsActiveFlag_ARRM

表40-23 函数 LL_LPTIM_IsActiveFlag_ARRM

函数名	LL_LPTIM_IsActiveFlag_ARRM
函数原形	__STATIC_INLINE uint32_t LL_LPTIM_IsActiveFlag_ARRM(LPTIM_TypeDef *LPTIMx)
功能描述	检查 ARRM 标志位是否置位
输入参数	LPTIMx：LPTIM 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

39.2.17 函数 LL_LPTIM_EnableIT_ARRM

描述了函数 LL_LPTIM_EnableIT_ARRM

表40-24 函数 LL_LPTIM_EnableIT_ARRM

函数名	LL_LPTIM_EnableIT_ARRM
函数原形	__STATIC_INLINE void LL_LPTIM_EnableIT_ARRM(LPTIM_TypeDef *LPTIMx)
功能描述	使能 ARRM 中断
输入参数	LPTIMx：LPTIM 实例
输出参数	无
返回值	无
先决条件	无

39.2.18 函数 LL_LPTIM_DisableIT_ARRM

描述了函数 LL_LPTIM_DisableIT_ARRM

表40-25 函数 LL_LPTIM_DisableIT_ARRM

函数名	LL_LPTIM_DisableIT_ARRM
函数原形	__STATIC_INLINE void LL_LPTIM_DisableIT_ARRM(LPTIM_TypeDef *LPTIMx)
功能描述	禁用 ARRM 中断
输入参数	LPTIMx：LPTIM 实例
输出参数	无
返回值	无
先决条件	无

39.2.19 函数 LL_LPTIM_IsEnabledIT_ARRM

描述了函数 LL_LPTIM_IsEnabledIT_ARRM

表40-26 函数 LL_LPTIM_IsEnabledIT_ARRM

函数名	LL_LPTIM_IsEnabledIT_ARRM
函数原形	__STATIC_INLINE uint32_t LL_LPTIM_IsEnabledIT_ARRM(LPTIM_TypeDef *LPTIMx)
功能描述	检查 ARRM 中断是否开启
输入参数	LPTIMx：LPTIM 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

39.2.20 函数 LL_LPTIM_DeInit

描述了函数 LL_LPTIM_DeInit

表40-27 函数 LL_LPTIM_DeInit

函数名	LL_LPTIM_DeInit
函数原形	ErrorStatus LL_LPTIM_DeInit(LPTIM_TypeDef *LPTIMx)
功能描述	将 LPTIM 相关配置重设为缺省值
输入参数	LPTIMx：LPTIM 实例
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

39.2.21 函数 LL_LPTIM_StructInit

描述了函数 LL_LPTIM_StructInit

表40-28 函数 LL_LPTIM_StructInit

函数名	LL_LPTIM_StructInit
函数原形	void LL_LPTIM_StructInit(LL_LPTIM_InitTypeDef *LPTIM_InitStruct)

功能描述	设置 LPTIM 初始化结构体为默认值
输入参数	LPTIM_InitStruct：初始化结构体
输出参数	无
返回值	无
先决条件	无

39.2.22 函数 LL_LPTIM_Init

描述了函数 LL_LPTIM_Init

表40-29　函数 LL_LPTIM_Init

函数名	LL_LPTIM_Init
函数原形	ErrorStatus LL_LPTIM_Init(LPTIM_TypeDef *LPTIMx, LL_LPTIM_InitTypeDef *LPTIM_InitStruct)
功能描述	初始化 LPTIM
输入参数 1	LPTIMx：LPTIM 实例
输入参数 2	LPTIM_InitStruct：初始化结构体
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

40 LL 电源功耗控制通用驱动程序（PWR）

芯片拥有两种低功耗模式Sleep Mode、 Stop Mode，可以配置通过事件或中断唤醒。

40.1 PWR 固件库函数

表40-30 UTILS 固件库函数说明

函数名	描述
LL_PWR_SetRegulVoltageScaling	配置STOP 模式 VDD 电压
LL_PWR_GetRegulVoltageScaling	获取STOP 模式 VDD 电压
LL_PWR_EnableLowPowerRunMode	配置STOP 模式使用 LPR 电压调节器
LL_PWR_DisableLowPowerRunMode	配置STOP 模式使用 MR 电压调节器
LL_PWR_IsEnabledLowPowerRunMode	检查STOP 模式是否使用LPR 电压调节器
LL_PWR_EnterLowPowerRunMode	同 LL_PWR_EnableLowPowerRunMode
LL_PWR_ExitLowPowerRunMode	同 LL_PWR_DisableLowPowerRunMode
LL_PWR_EnableBkUpAccess	允许访问 RTC
LL_PWR_DisableBkUpAccess	禁止访问 RTC
LL_PWR_IsEnabledBkUpAccess	检查 RTC 是否允许访问
LL_PWR_SetWakeUpHSIOnMode	配置STOP 唤醒后HSI 打开时间
LL_PWR_GetWakeUpHSIOnMode	获取STOP 唤醒后HSI 打开时间
LL_PWR_SetSramRetentionVolt	配置STOP 模式下SRAM retention 电压
LL_PWR_GetSramRetentionVolt	获取STOP 模式下SRAM retention 电压
LL_PWR_SetWakeUpFlashDelay	配置STOP 唤醒、HSI 稳定后FLASH 操作等待时间
LL_PWR_GetWakeUpFlashDelay	获取STOP 唤醒、HSI 稳定后FLASH 操作等待时间
LL_PWR_SetWakeUpLPToVRReadyTime	配置STOP 唤醒后 LP-VR 切换至 Main-VR 的时间
LL_PWR_GetWakeUpLPToVRReadyTime	获取STOP 唤醒后 LP-VR 切换至 Main-VR 的时间
LL_PWR_SetBiasCurrents	配置 MR 偏置电流选择和偏置电流寄存器的值
LL_PWR_GetBiasCurrentsLoadSource	获取 MR 偏置电流选择
LL_PWR_GetBiasCRValue	获取 MR 偏置电流寄存器的值
LL_PWR_SetPVDLevel	配置PVD（电压检测）阈值
LL_PWR_GetPVDLevel	获取PVD（电压检测）阈值
LL_PWR_EnablePVD	使能PVD
LL_PWR_DisablePVD	禁用PVD

LL_PWR_IsEnabledPVD	检查PVD 是否使能
LL_PWR_SetPVDSource	PVD(电压检测)电源选择
LL_PWR_GetPVDSource	获取PVD(电压检测)输入源
LL_PWR_EnablePVDFilter	使能PVD 数字滤波功能
LL_PWR_DisablePVDFilter	禁用PVD 数字滤波功能
LL_PWR_IsEnabledPVDFilter	检查PVD 数字滤波功能是否使能
LL_PWR_SetPVDFilter	配置PVD 数字滤波时间
LL_PWR_GetPVDFilter	获取PVD 数字滤波时间
LL_PWR_IsActiveFlag_PVDO	返回PVD 检测结果（高于或低于阈值）
LL_PWR_DeInit	将 PWR 配置重设为缺省值

40.1.1 函数 LL_PWR_SetRegulVoltageScaling

描述了函数 LL_PWR_SetRegulVoltageScaling

表40-31 函数 LL_PWR_SetRegulVoltageScaling

函数名	LL_PWR_SetRegulVoltageScaling
函数原形	__STATIC_INLINE void LL_PWR_SetRegulVoltageScaling(uint32_t VoltageScaling)
功能描述	配置 STOP 模式 VDD 电压
输入参数	VoltageScaling：VDD 电压
输出参数	无
返回值	无
先决条件	无

VoltageScaling 可选参数:

表40-32 VoltageScaling 可选参数

参数	描述
LL_PWR_REGU_VOLTAGE_SCALE1	进入 stop 模式后，VDD=1.2V
LL_PWR_REGU_VOLTAGE_SCALE2	进入 stop 模式后，VDD=1.0V

40.1.2 函数 LL_PWR_GetRegulVoltageScaling

描述了函数 LL_PWR_GetRegulVoltageScaling

表40-33 函数 LL_PWR_GetRegulVoltageScaling

函数名	LL_PWR_GetRegulVoltageScaling
函数原形	__STATIC_INLINE uint32_t LL_PWR_GetRegulVoltageScaling(void)
功能描述	获取 STOP 模式 VDD 电压
输入参数	无

输出参数	无
返回值	STOP 模式电压
先决条件	无

40.1.3 函数 LL_PWR_EnableLowPowerRunMode

描述了函数 LL_PWR_EnableLowPowerRunMode

表40-34 函数 LL_PWR_EnableLowPowerRunMode

函数名	LL_PWR_EnableLowPowerRunMode
函数原形	__STATIC_INLINE void LL_PWR_EnableLowPowerRunMode(void)
功能描述	配置 STOP 模式使用 LPR 电压调节器
输入参数	无
输出参数	无
返回值	无
先决条件	无

40.1.4 函数 LL_PWR_DisableLowPowerRunMode

描述了函数 LL_PWR_DisableLowPowerRunMode

表40-35 函数 LL_PWR_DisableLowPowerRunMode

函数名	LL_PWR_DisableLowPowerRunMode
函数原形	__STATIC_INLINE void LL_PWR_DisableLowPowerRunMode(void)
功能描述	配置 STOP 模式使用 MR 电压调节器
输入参数	无
输出参数	无
返回值	无
先决条件	无

40.1.5 函数 LL_PWR_IsEnabledLowPowerRunMode

描述了函数 LL_PWR_IsEnabledLowPowerRunMode

表40-36 函数 LL_PWR_IsEnabledLowPowerRunMode

函数名	LL_PWR_IsEnabledLowPowerRunMode
函数原形	__STATIC_INLINE uint32_t LL_PWR_IsEnabledLowPowerRunMode(void)
功能描述	检查 STOP 模式是否使用 LPR 电压调节器
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

40.1.6 函数 LL_PWR_EnterLowPowerRunMode

描述了函数 LL_PWR_EnterLowPowerRunMode

表40-37 函数 LL_PWR_EnterLowPowerRunMode

函数名	LL_PWR_EnterLowPowerRunMode
函数原形	__STATIC_INLINE void LL_PWR_EnterLowPowerRunMode(void)
功能描述	配置 STOP 模式使用 LPR 电压调节器
输入参数	无
输出参数	无
返回值	无
先决条件	无

40.1.7 函数 LL_PWR_ExitLowPowerRunMode

描述了函数 LL_PWR_ExitLowPowerRunMode

表40-38 函数 LL_PWR_ExitLowPowerRunMode

函数名	LL_PWR_ExitLowPowerRunMode
函数原形	__STATIC_INLINE void LL_PWR_ExitLowPowerRunMode(void)
功能描述	配置 STOP 模式使用 MR 电压调节器
输入参数	无
输出参数	无
返回值	无
先决条件	无

40.1.8 函数 LL_PWR_EnableBkUpAccess

描述了函数 LL_PWR_EnableBkUpAccess

表40-39 函数 LL_PWR_EnableBkUpAccess

函数名	LL_PWR_EnableBkUpAccess
函数原形	__STATIC_INLINE void LL_PWR_EnableBkUpAccess(void)
功能描述	允许访问 RTC
输入参数	无
输出参数	无
返回值	无
先决条件	无

40.1.9 函数 LL_PWR_DisableBkUpAccess

描述了函数 LL_PWR_DisableBkUpAccess

表40-40 函数 LL_PWR_DisableBkUpAccess

函数名	LL_PWR_DisableBkUpAccess
函数原形	__STATIC_INLINE void LL_PWR_DisableBkUpAccess(void)
功能描述	禁止访问 RTC
输入参数	无
输出参数	无
返回值	无
先决条件	无

40.1.10 函数 LL_PWR_IsEnabledBkUpAccess

描述了函数 LL_PWR_IsEnabledBkUpAccess

表40-41 函数 LL_PWR_IsEnabledBkUpAccess

函数名	LL_PWR_IsEnabledBkUpAccess
函数原形	__STATIC_INLINE uint32_t LL_PWR_IsEnabledBkUpAccess(void)
功能描述	检查是否允许访问 RTC
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

40.1.11 函数 LL_PWR_SetWakeUpHSIOnMode

描述了函数 LL_PWR_SetWakeUpHSIOnMode

表40-42 函数 LL_PWR_SetWakeUpHSIOnMode

函数名	LL_PWR_SetWakeUpHSIOnMode
函数原形	__STATIC_INLINE void LL_PWR_SetWakeUpHSIOnMode(void)
功能描述	配置 STOP 唤醒后 HSI 打开时间
输入参数	无
输出参数	无
返回值	无
先决条件	无

40.1.12 函数 LL_PWR_GetWakeUpHSIOnMode

描述了函数 LL_PWR_GetWakeUpHSIOnMode

表40-43 函数 LL_PWR_GetWakeUpHSIOnMode

函数名	LL_PWR_GetWakeUpHSIOnMode
函数原形	__STATIC_INLINE void LL_PWR_GetWakeUpHSIOnMode(void)
功能描述	获取 STOP 唤醒后 HSI 打开时间

输入参数	无
输出参数	无
返回值	无
先决条件	无

40.1.13 函数 LL_PWR_SetSramRetentionVolt

描述了函数 LL_PWR_SetSramRetentionVolt

表40-44 函数 LL_PWR_SetSramRetentionVolt

函数名	LL_PWR_SetSramRetentionVolt
函数原形	__STATIC_INLINE void LL_PWR_SetSramRetentionVolt (void)
功能描述	配置 STOP 模式下 SRAM retention 电压
输入参数	无
输出参数	无
返回值	无
先决条件	无

40.1.14 函数 LL_PWR_GetSramRetentionVolt

描述了函数 LL_PWR_GetSramRetentionVolt

表40-45 函数 LL_PWR_GetSramRetentionVolt

函数名	LL_PWR_GetSramRetentionVolt
函数原形	__STATIC_INLINE void LL_PWR_GetSramRetentionVolt(void)
功能描述	获取 STOP 模式下 SRAM retention 电压
输入参数	无
输出参数	无
返回值	无
先决条件	无

40.1.15 函数 LL_PWR_SetWakeUpFlashDelay

描述了函数 LL_PWR_SetWakeUpFlashDelay

表40-46 函数 LL_PWR_SetWakeUpFlashDelay

函数名	LL_PWR_SetWakeUpFlashDelay
函数原形	__STATIC_INLINE void LL_PWR_SetWakeUpFlashDelay(void)
功能描述	配置 STOP 唤醒、HSI 稳定后 FLASH 操作等待时间
输入参数	无
输出参数	无
返回值	无

先决条件	无

40.1.16 函数 LL_PWR_GetWakeUpFlashDelay

描述了函数 LL_PWR_GetWakeUpFlashDelay

表40-47 函数 LL_PWR_GetWakeUpFlashDelay

函数名	LL_PWR_GetWakeUpFlashDelay
函数原形	__STATIC_INLINE void LL_PWR_GetWakeUpFlashDelay(void)
功能描述	获取 STOP 唤醒、HSI 稳定后 FLASH 操作等待时间
输入参数	无
输出参数	无
返回值	无
先决条件	无

40.1.17 函数 LL_PWR_SetWakeUpLPToVRReadyTime

描述了函数 LL_PWR_SetWakeUpLPToVRReadyTime

表40-48 函数 LL_PWR_SetWakeUpLPToVRReadyTime

函数名	LL_PWR_SetWakeUpLPToVRReadyTime
函数原形	__STATIC_INLINE void LL_PWR_SetWakeUpLPToVRReadyTime (void)
功能描述	配置 STOP 唤醒后 LP-VR 切换至 Main-VR 的时间
输入参数	无
输出参数	无
返回值	无
先决条件	无

40.1.18 函数 LL_PWR_GetWakeUpLPToVRReadyTime

描述了函数 LL_PWR_GetWakeUpFlashDelay

表40-49 函数 LL_PWR_GetWakeUpFlashDelay

函数名	LL_PWR_GetWakeUpFlashDelay
函数原形	__STATIC_INLINE void LL_PWR_GetWakeUpFlashDelay(void)
功能描述	获取 STOP 唤醒后 LP-VR 切换至 Main-VR 的时间
输入参数	无
输出参数	无
返回值	无
先决条件	无

40.1.19 函数 LL_PWR_SetBiasCurrents

描述了函数 LL_PWR_SetBiasCurrents

表40-50　函数 LL_PWR_SetBiasCurrents

函数名	LL_PWR_SetBiasCurrents
函数原形	__STATIC_INLINE void LL_PWR_SetBiasCurrents(uint32_t BiasCurSel, uint32_t BiasCur)
功能描述	配置 MR 偏置电流选择和偏置电流寄存器的值
输入参数 1	BiasCurSel：MR 偏置电流选择
输入参数 2	BiasCur：配置偏置电流 BIAS_CR 寄存器的值
输出参数	无
返回值	无
先决条件	无

BiasCurSel 可选参数:

表40-51　BiasCurSel 可选参数

参数	描述
LL_PWR_BIAS_CURRENTS_FROM_FACTORY_BYTES	MR 偏置电流选择来自 Factory config.bytes 区的加载
LL_PWR_BIAS_CURRENTS_FROM_BIAS_CR	MR 偏置电流选择来自 BIAS_CR 寄存器

40.1.20 函数 LL_PWR_GetBiasCurrentsLoadSource

描述了函数 LL_PWR_GetBiasCurrentsLoadSource

表40-52　函数 LL_PWR_GetBiasCurrentsLoadSource

函数名	LL_PWR_GetBiasCurrentsLoadSource
函数原形	__STATIC_INLINE uint32_t LL_PWR_GetBiasCurrentsLoadSource(void)
功能描述	获取 MR 偏置电流选择
输入参数	无
输出参数	无
返回值	无
先决条件	无

40.1.21 函数 LL_PWR_GetBiasCRValue

描述了函数 LL_PWR_GetBiasCRValue

表40-53　函数 LL_PWR_GetBiasCRValue

函数名	LL_PWR_GetBiasCRValue
函数原形	__STATIC_INLINE uint32_t LL_PWR_GetBiasCRValue(void)
功能描述	获取 MR 偏置电流寄存器的值
输入参数	无
输出参数	无
返回值	MR 偏置电流寄存器的值

先决条件	无

40.1.22 函数 LL_PWR_SetPVDLevel

描述了函数 LL_PWR_SetPVDLevel

表40-54 函数 LL_PWR_SetPVDLevel

函数名	LL_PWR_SetPVDLevel
函数原形	__STATIC_INLINE void LL_PWR_SetPVDLevel(uint32_t PVDLevel)
功能描述	配置 PVD 检测阈值
输入参数	PVDLevel：PVD 检测阈值
输出参数	无
返回值	无
先决条件	无

PVDLevel 可选参数：

表40-55 PVDLevel 可选参数

参数	描述
LL_PWR_PVDLEVEL_0	PVD 检测等级 0 上升沿检测阈值 1.8V，下降沿相应减少 0.1V
LL_PWR_PVDLEVEL_1	PVD 检测等级 1 上升沿检测阈值 2.0V，下降沿相应减少 0.1V
LL_PWR_PVDLEVEL_2	PVD 检测等级 2 上升沿检测阈值 2.2V，下降沿相应减少 0.1V
LL_PWR_PVDLEVEL_3	PVD 检测等级 3 上升沿检测阈值 2.4V，下降沿相应减少 0.1V
LL_PWR_PVDLEVEL_4	PVD 检测等级 4 上升沿检测阈值 2.6V，下降沿相应减少 0.1V
LL_PWR_PVDLEVEL_5	PVD 检测等级 5 上升沿检测阈值 2.8V，下降沿相应减少 0.1V
LL_PWR_PVDLEVEL_6	PVD 检测等级 6 上升沿检测阈值 3.0V，下降沿相应减少 0.1V
LL_PWR_PVDLEVEL_7	PVD 检测等级 7 上升沿检测阈值 3.2V，下降沿相应减少 0.1V

40.1.23 函数 LL_PWR_GetPVDLevel

描述了函数 LL_PWR_GetPVDLevel

表40-56 函数 LL_PWR_GetPVDLevel

函数名	LL_PWR_GetPVDLevel

函数原形	__STATIC_INLINE uint32_t LL_PWR_GetPVDLevel(void)
功能描述	获取 PVD 检测阈值
输入参数	无
输出参数	无
返回值	PVD 检测阈值
先决条件	无

40.1.24 函数 LL_PWR_EnablePVD

描述了函数 LL_PWR_EnablePVD

表40-57　函数 LL_PWR_EnablePVD

函数名	LL_PWR_EnablePVD
函数原形	__STATIC_INLINE void LL_PWR_EnablePVD(void)
功能描述	使能 PVD
输入参数	无
输出参数	无
返回值	无
先决条件	无

40.1.25 函数 LL_PWR_DisablePVD

描述了函数 LL_PWR_DisablePVD

表40-58　函数 LL_PWR_DisablePVD

函数名	LL_PWR_DisablePVD
函数原形	__STATIC_INLINE void LL_PWR_DisablePVD(void)
功能描述	禁用 PVD
输入参数	无
输出参数	无
返回值	无
先决条件	无

40.1.26 函数 LL_PWR_IsEnabledPVD

描述了函数 LL_PWR_IsEnabledPVD

表40-59　函数 LL_PWR_IsEnabledPVD

函数名	LL_PWR_IsEnabledPVD
函数原形	__STATIC_INLINE uint32_t LL_PWR_IsEnabledPVD(void)
功能描述	检查 PVD 是否使能
输入参数	无

输出参数	无
返回值	状态位（1 或 0）
先决条件	无

40.1.27 函数 LL_PWR_SetPVDSource

描述了函数 LL_PWR_SetPVDSource

表40-60 函数 LL_PWR_SetPVDSource

函数名	LL_PWR_SetPVDSource
函数原形	__STATIC_INLINE void LL_PWR_SetPVDSource(uint32_t PVDSrc)
功能描述	PVD(电压检测)电源选择
输入参数	PVDSrc：电源选择
输出参数	无
返回值	无
先决条件	无

PVDSrc 可选参数:

表40-61 PVDSrc 可选参数

参数	描述
LL_PWR_PVD_SOURCE_VCC	检测 VCC
LL_PWR_PVD_SOURCE_PB7	检测 PB7

40.1.28 函数 LL_PWR_GetPVDSource

描述了函数 LL_PWR_GetPVDSource

表40-62 函数 LL_PWR_GetPVDSource

函数名	LL_PWR_GetPVDSource
函数原形	__STATIC_INLINE uint32_t LL_PWR_GetPVDSource(void)
功能描述	获取 PVD(电压检测)输入源
输入参数	无
输出参数	无
返回值	输入源
先决条件	无

40.1.29 函数 LL_PWR_EnablePVDFilter

描述了函数 LL_PWR_EnablePVDFilter

表40-63 函数 LL_PWR_EnablePVDFilter

函数名	LL_PWR_EnablePVDFilter
函数原形	__STATIC_INLINE void LL_PWR_EnablePVDFilter(void)

功能描述	使能 PVD 数字滤波功能
输入参数	无
输出参数	无
返回值	无
先决条件	无

40.1.30 函数 LL_PWR_DisablePVDFilter

描述了函数 LL_PWR_DisablePVDFilter

表40-64 函数 LL_PWR_DisablePVDFilter

函数名	LL_PWR_DisablePVDFilter
函数原形	__STATIC_INLINE void LL_PWR_DisablePVDFilter(void)
功能描述	禁用 PVD 数字滤波功能
输入参数	无
输出参数	无
返回值	无
先决条件	无

40.1.31 函数 LL_PWR_IsEnabledPVDFilter

描述了函数 LL_PWR_IsEnabledPVDFilter

表40-65 函数 LL_PWR_IsEnabledPVDFilter

函数名	LL_PWR_IsEnabledPVDFilter
函数原形	__STATIC_INLINE uint32_t LL_PWR_IsEnabledPVDFilter (void)
功能描述	检查 PVD 数字滤波功能是否使能
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

40.1.32 函数 LL_PWR_SetPVDFilter

描述了函数 LL_PWR_SetPVDFilter

表40-66 函数 LL_PWR_SetPVDFilter

函数名	LL_PWR_SetPVDFilter
函数原形	__STATIC_INLINE void LL_PWR_SetPVDFilter(uint32_t PVDFilter)
功能描述	配置 PVD 数字滤波时间
输入参数	PVDFilter：PVD 数字滤波时间
输出参数	无

返回值	无
先决条件	无

PVDFilter 可选参数:

表40-67 PVDFilter 可选参数

参数	描述
LL_PWR_PVD_FILTER_1CLOCK	滤波时间大约为 30us（1 个 LSI 或者 LSE 时钟）
LL_PWR_PVD_FILTER_2CLOCK	滤波时间大约为 60us（2 个 LSI 或者 LSE 时钟）
LL_PWR_PVD_FILTER_4CLOCK	滤波时间大约为 120us（4 个 LSI 或者 LSE 时钟）
LL_PWR_PVD_FILTER_16CLOCK	滤波时间大约为 480us（16 个 LSI 或者 LSE 时钟）
LL_PWR_PVD_FILTER_64CLOCK	滤波时间大约为 1.92ms（64 个 LSI 或者 LSE 时钟）
LL_PWR_PVD_FILTER_128CLOCK	滤波时间大约为 3.8ms（128 个 LSI 或者 LSE 时钟）
LL_PWR_PVD_FILTER_1024CLOCK	滤波时间大约为 30.7ms（1024 个 LSI 或者 LSE 时钟）

40.1.33 函数 LL_PWR_GetPVDFilter

描述了函数 LL_PWR_GetPVDFilter

表40-68 函数 LL_PWR_GetPVDFilter

函数名	LL_PWR_GetPVDFilter
函数原形	__STATIC_INLINE uint32_t LL_PWR_GetPVDFilter(void)
功能描述	获取 PVD 数字滤波时间
输入参数	无
输出参数	无
返回值	PVD 数字滤波时间
先决条件	无

40.1.34 函数 LL_PWR_IsActiveFlag_PVDO

描述了函数 LL_PWR_IsActiveFlag_PVDO

表40-69 函数 LL_PWR_IsActiveFlag_PVDO

函数名	LL_PWR_IsActiveFlag_PVDO
函数原形	__STATIC_INLINE uint32_t LL_PWR_IsActiveFlag_PVDO(void)
功能描述	返回 PVD 检测结果高于还是低于阈值
输入参数	无
输出参数	无
返回值	PVD 检测结果（1 或 0）
先决条件	无

40.1.35 函数 LL_PWR_DeInit

描述了函数 LL_PWR_DeInit

表40-70　函数 LL_PWR_DeInit

函数名	LL_PWR_DeInit
函数原形	__STATIC_INLINE void LL_PWR_DeInit(void)
功能描述	将 PWR 配置重设为缺省值
输入参数	无
输出参数	无
返回值	无
先决条件	无

41 LL 复位和时钟通用驱动程序（RCC）

复位一共有三种类型的复位，分别为系统复位、电源复位和 RTC 域复位。

芯片提供以下时钟源，可以产生主时钟：

- 外部高速时钟 HSE
- 外部低速时钟 LSE
- 内部高速时钟 HIS
- 内部低速时钟 LSI
- PLL

41.1 RCC 固件驱动寄存器结构

41.1.1 LL_RCC_ClocksTypeDef

LL_RCC_ClocksTypeDef，定义于文件“air001xx_ll_rcc.h”如下：

```
typedef struct
{
uint32_t SYSCLK_Frequency;
uint32_t HCLK_Frequency;
uint32_t PCLK1_Frequency;
} LL_RCC_ClocksTypeDef;
```

字段说明：

表41-1 LL_RCC_ClocksTypeDef 字段说明

字段	描述
SYSCLK_Frequency	系统时钟频率
HCLK_Frequency	AHB 时钟频率
PCLK1_Frequency	APB 时钟频率

41.2 RCC 固件库函数

表41-2 ADC 固件库函数说明

函数名	描述
LL_RCC_HSE_EnableCSS	使能HSE 时钟安全系统
LL_RCC_HSE_EnableBypass	使能HSE 外部振荡器
LL_RCC_HSE_DisableBypass	禁用 HSE 外部振荡器
LL_RCC_HSE_Enable	使能HSE
LL_RCC_HSE_Disable	禁用HSE

LL_RCC_HSE_IsReady	检查HSE 是否准备就绪
LL_RCC_HSE_SetFreqRegion	设置当前HSE 振荡器频率
LL_RCC_HSI_Enable	使能HSI
LL_RCC_HSI_Disable	禁用HSI
LL_RCC_HSI_IsReady	检查HSI 是否准备就绪
LL_RCC_HSI_SetCalibTrimming	设置HSI 校准微调值
LL_RCC_HSI_GetCalibTrimming	获取HSI 校准微调值
LL_RCC_HSI_SetCalibFreq	设置 HSI 校准频率
LL_RCC_HSI_GetFreq	获取 HSI 校准频率
LL_RCC_LSE_Enable	使能 LSE
LL_RCC_LSE_Disable	禁用 LSE
LL_RCC_LSE_EnableBypass	使能 LSE 外部振荡器
LL_RCC_LSE_DisableBypass	禁用 LSE 外部振荡器
LL_RCC_LSE_SetDriveCapability	设置 LSE 驱动能力
LL_RCC_LSE_GetDriveCapability	获取 LSE 驱动能力
LL_RCC_LSE_EnableCSS	使能 LSE 时钟安全系统
LL_RCC_LSE_DisableCSS	禁用 LSE 时钟安全系统
LL_RCC_LSE_IsReady	检查 LSE 是否准备就绪
LL_RCC_LSE_IsCSSDetected	检查 LSE 是否失效
LL_RCC_LSI_Enable	使能 LSI
LL_RCC_LSI_Disable	禁用 LSI
LL_RCC_LSI_IsReady	检查 LSI 是否准备就绪
LL_RCC_LSCO_Enable	使能 LSC
LL_RCC_LSCO_Disable	禁用 LSC
LL_RCC_LSCO_SetSource	设置 LSC 源
LL_RCC_LSCO_GetSource	获取 LSC 源
LL_RCC_SetSysClkSource	设置系统时钟源
LL_RCC_GetSysClkSource	获取系统时钟源
LL_RCC_SetAHBPrescaler	设置AHB 分频
LL_RCC_SetAPB1Prescaler	设置APB 分频
LL_RCC_SetHSIDiv	设置HSI 时钟分频

LL_RCC_GetAHBPrescaler	获取AHB 时钟分频值
LL_RCC_GetAPB1Prescaler	获取APB1 时钟分频值
LL_RCC_GetHSIDiv	获取HSI 时钟分频值
LL_RCC_ConfigMCO	配置 MCO
LL_RCC_GetMCOClockSource	获取 MCO 时钟源
LL_RCC_GetMCODiv	获取 MCO 分频
LL_RCC_SetPVDClockSource	设置PVD 时钟源
LL_RCC_GetPVDClockSource	获取PVD 时钟源
LL_RCC_SetCOMPClockSource	设置COMP 时钟源
LL_RCC_GetCOMPClockSource	获取COMP 时钟源
LL_RCC_SetLPTIMClockSource	设置 LPTIM 时钟源
LL_RCC_GetLPTIMClockSource	获取 LPTIM 时钟源
LL_RCC_SetRTCClockSource	设置 RTC 时钟源
LL_RCC_GetRTCClockSource	获取 RTC 时钟源
LL_RCC_EnableRTC	使能 RTC
LL_RCC_DisableRTC	禁用 RTC
LL_RCC_IsEnabledRTC	检查 RTC 是否是使能
LL_RCC_ForceBackupDomainReset	强制备份域寄存器复位
LL_RCC_ReleaseBackupDomainReset	释放备份域寄存器复位
LL_RCC_PLL_Enable	使能PLL
LL_RCC_PLL_Disable	禁用PLL
LL_RCC_PLL_IsReady	检查PLL 是否准备就绪
LL_RCC_PLL_SetMainSource	设置PLL 时钟源
LL_RCC_PLL_GetMainSource	获取PLL 时钟源
LL_RCC_ClearFlag_LSIRDY	清除 LSIRDY 标记
LL_RCC_ClearFlag_LSERDY	清除 LSERDY 标记
LL_RCC_ClearFlag_HSIRDY	清除HSIRDY 标记
LL_RCC_ClearFlag_HSERDY	清除HSERDY 标记
LL_RCC_ClearFlag_PLLRDY	清除PLLRDY 标记
LL_RCC_ClearFlag_HSECSS	清除HSE CSS 中断标记
LL_RCC_ClearFlag_LSECSS	清除 LSE CSS 中断标记

LL_RCC_IsActiveFlag_LSIRDY	检查 LSIRDY 标记是否置位
LL_RCC_IsActiveFlag_LSERDY	检查 LSERDY 标记是否置位
LL_RCC_IsActiveFlag_HSIRDY	检查HSIRDY 标记是否置位
LL_RCC_IsActiveFlag_HSERDY	检查HSERDY 标记是否置位
LL_RCC_IsActiveFlag_PLLRDY	检查PLLRDY 标记是否置位
LL_RCC_IsActiveFlag_HSECSS	检查HSE CSS 中断标记是否置位
LL_RCC_IsActiveFlag_LSECSS	检查 LSE CSS 中断标记是否置位
LL_RCC_IsActiveFlag_IWDGRST	检查 IWDGRST 中断标记是否置位
LL_RCC_IsActiveFlag_OBLRST	检查 OBLRST 中断标记是否置位
LL_RCC_IsActiveFlag_PINRST	检查PINRST 中断标记是否置位
LL_RCC_IsActiveFlag_SFTRST	检查SFTRST 中断标记是否置位
LL_RCC_IsActiveFlag_WWDGRST	检查WWDGRST 中断标记是否置位
LL_RCC_IsActiveFlag_PWRRST	检查PWRRST 中断标记是否置位
LL_RCC_ClearResetFlags	清除复位标记
LL_RCC_EnableNRSTFilter	使能NRST 滤波
LL_RCC_DisableNRSTFilter	禁用NRST 滤波
LL_RCC_IsEnableNRSTFilter	检查NRST 滤波是否开启
LL_RCC_EnableIT_LSIRDY	使能 LSIRDY 中断
LL_RCC_EnableIT_LSERDY	使能 LSERDY 中断
LL_RCC_EnableIT_HSIRDY	使能HSIRDY 中断
LL_RCC_EnableIT_HSERDY	使能HSERDY 中断
LL_RCC_EnableIT_PLLRDY	使能PLLRDY 中断
LL_RCC_DisableIT_LSIRDY	禁用 LSIRDY 中断
LL_RCC_DisableIT_LSERDY	禁用 LSERDY 中断
LL_RCC_DisableIT_HSIRDY	禁用HSIRDY 中断
LL_RCC_DisableIT_HSERDY	禁用HSERDY 中断
LL_RCC_DisableIT_PLLRDY	禁用PLLRDY 中断
LL_RCC_IsEnabledIT_LSIRDY	检查 LSIRDY 中断标志是否置位
LL_RCC_IsEnabledIT_LSERDY	检查 LSERDY 中断标志是否置位
LL_RCC_IsEnabledIT_HSIRDY	检查HSIRDY 中断标志是否置位
LL_RCC_IsEnabledIT_HSERDY	检查HSERDY 中断标志是否置位

LL_RCC_IsEnabledIT_PLLRDY	检查PLLRDY 中断标志是否置位
LL_RCC_GetSystemClocksFreq	获取系统时钟频率
LL_RCC_GetMCOClockFreq	获取 MCO 时钟频率
LL_RCC_GetLSCClockFreq	获取 LSC 时钟频率
LL_RCC_GetPVDClockFreq	获取PVD 时钟频率
LL_RCC_GetCOMPClockFreq	获取COMP 时钟频率
LL_RCC_GetLPTIMClockFreq	获取 LPTIM 时钟频率
LL_RCC_GetRTCClockFreq	获取 RTC 时钟频率

41.2.1 函数 LL_RCC_HSE_EnableCSS

描述了函数 LL_RCC_HSE_EnableCSS

表41-3 函数 LL_RCC_HSE_EnableCSS

函数名	LL_RCC_HSE_EnableCSS
函数原形	__STATIC_INLINE void LL_RCC_HSE_EnableCSS(void)
功能描述	使能 HSE 时钟安全系统
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.2 函数 LL_RCC_HSE_EnableBypass

描述了函数 LL_RCC_HSE_EnableBypass

表41-4 函数 LL_RCC_HSE_EnableBypass

函数名	LL_RCC_HSE_EnableBypass
函数原形	__STATIC_INLINE void LL_RCC_HSE_EnableBypass(void)
功能描述	使能 HSE 外部振荡器
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.3 函数 LL_RCC_HSE_DisableBypass

描述了函数 LL_RCC_HSE_DisableBypass

表41-5 函数 LL_RCC_HSE_DisableBypass

函数名	LL_RCC_HSE_DisableBypass

函数原形	__STATIC_INLINE void LL_RCC_HSE_DisableBypass(void)
功能描述	禁用 HSE 外部振荡器
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.4 函数 LL_RCC_HSE_Enable

描述了函数 LL_RCC_HSE_Enable

表41-6 函数 LL_RCC_HSE_Enable

函数名	LL_RCC_HSE_Enable
函数原形	__STATIC_INLINE void LL_RCC_HSE_Enable(void)
功能描述	使能 HSE
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.5 函数 LL_RCC_HSE_Disable

描述了函数 LL_RCC_HSE_Disable

表41-7 函数 LL_RCC_HSE_Disable

函数名	LL_RCC_HSE_Disable
函数原形	__STATIC_INLINE void LL_RCC_HSE_Disable(void)
功能描述	禁用 HSE
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.6 函数 LL_RCC_HSE_IsReady

描述了函数 LL_RCC_HSE_IsReady

表41-8 函数 LL_RCC_HSE_IsReady

函数名	LL_RCC_HSE_IsReady
函数原形	__STATIC_INLINE uint32_t LL_RCC_HSE_IsReady(void)
功能描述	检查 HSE 是否准备就绪
输入参数	无

输出参数	无
返回值	无
先决条件	无

41.2.7 函数 LL_RCC_HSE_SetFreqRegion

描述了函数 LL_RCC_HSE_SetFreqRegion

表41-9 函数 LL_RCC_HSE_SetFreqRegion

函数名	LL_RCC_HSE_SetFreqRegion
函数原形	__STATIC_INLINE void LL_RCC_HSE_SetFreqRegion(uint32_t HSEFreq)
功能描述	设置当前 HSE 振荡器频率
输入参数	HSEFreq：HSE 振荡器频率
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

HSEFreq 可选参数：

表41-10 HSEFreq 可选参数

参数	描述
LL_RCC_HSE_STOP	禁用 HSE
LL_RCC_HSE_4_8MHz	HSE 振荡器频率为 4~8MHz
LL_RCC_HSE_8_16MHz	HSE 振荡器频率为 8~16MHz
LL_RCC_HSE_16_32MHz	HSE 振荡器频率为 16~32MHz

41.2.8 函数 LL_RCC_HSI_Enable

描述了函数 LL_RCC_HSI_Enable

表41-11 函数 LL_RCC_HSI_Enable

函数名	LL_RCC_HSI_Enable
函数原形	__STATIC_INLINE void LL_RCC_HSI_Enable(void)
功能描述	使能 HSI
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.9 函数 LL_RCC_HSI_Disable

描述了函数 LL_RCC_HSI_Disable

表41-12 函数 LL_RCC_HSI_Disable

函数名	LL_RCC_HSI_Disable
函数原形	__STATIC_INLINE void LL_RCC_HSI_Disable(void)
功能描述	禁用 HSI
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.10 函数 LL_RCC_HSI_IsReady

描述了函数 LL_RCC_HSI_IsReady

表41-13 函数 LL_RCC_HSI_IsReady

函数名	LL_RCC_HSI_IsReady
函数原形	__STATIC_INLINE uint32_t LL_RCC_HSI_IsReady(void)
功能描述	检查 HSI 是否准备就绪
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.11 函数 LL_RCC_HSI_SetCalibTrimming

描述了函数 LL_RCC_HSI_SetCalibTrimming

表41-14 函数 LL_RCC_HSI_SetCalibTrimming

函数名	LL_RCC_HSI_SetCalibTrimming
函数原形	__STATIC_INLINE void LL_RCC_HSI_SetCalibTrimming(uint32_t Value)
功能描述	设置 HSI 校准微调值
输入参数	Value：校准微调值（0x0000~0x1FFF）
输出参数	无
返回值	无
先决条件	无

41.2.12 函数 LL_RCC_HSI_GetCalibTrimming

描述了函数 LL_RCC_HSI_GetCalibTrimming

表41-15 函数 LL_RCC_HSI_GetCalibTrimming

函数名	LL_RCC_HSI_GetCalibTrimming
函数原形	__STATIC_INLINE uint32_t LL_RCC_HSI_GetCalibTrimming(void)
功能描述	获取 HSI 校准微调值

输入参数	无
输出参数	无
返回值	校准微调值
先决条件	无

41.2.13 函数 LL_RCC_HSI_SetCalibFreq

描述了函数 LL_RCC_HSI_SetCalibFreq

表41-16 函数 LL_RCC_HSI_SetCalibFreq

函数名	LL_RCC_HSI_SetCalibFreq
函数原形	__STATIC_INLINE void LL_RCC_HSI_SetCalibFreq(uint32_t Value)
功能描述	设置 HSI 校准频率
输入参数	Value：HSI 校准频率
输出参数	无
返回值	无
先决条件	无

Value 可选参数：

表41-17 Value 可选参数

参数	描述
LL_RCC_HSICALIBRATION_4MHz	校准频率 4MHz
LL_RCC_HSICALIBRATION_8MHz	校准频率 8MHz
LL_RCC_HSICALIBRATION_16MHz	校准频率 16MHz
LL_RCC_HSICALIBRATION_22p12MHz	校准频率 22.12MHz
LL_RCC_HSICALIBRATION_24MHz	校准频率 24MHz

41.2.14 函数 LL_RCC_HSI_GetFreq

描述了函数 LL_RCC_HSI_GetFreq

表41-18 函数 LL_RCC_HSI_GetFreq

函数名	LL_RCC_HSI_GetFreq
函数原形	__STATIC_INLINE uint32_t LL_RCC_HSI_GetFreq(void)
功能描述	获取 HSI 校准频率
输入参数	无
输出参数	无
返回值	HSI 校准频率
先决条件	无

41.2.15 函数 LL_RCC_LSE_Enable

描述了函数 LL_RCC_LSE_Enable

表41-19 函数 LL_RCC_LSE_Enable

函数名	LL_RCC_LSE_Enable
函数原形	__STATIC_INLINE void LL_RCC_LSE_Enable(void)
功能描述	使能 LSE
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.16 函数 LL_RCC_LSE_Disable

描述了函数 LL_RCC_LSE_Disable

表41-20 函数 LL_RCC_LSE_Disable

函数名	LL_RCC_LSE_Disable
函数原形	__STATIC_INLINE void LL_RCC_LSE_Disable(void)
功能描述	禁用 LSE
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.17 函数 LL_RCC_LSE_EnableBypass

描述了函数 LL_RCC_LSE_EnableBypass

表41-21 函数 LL_RCC_LSE_EnableBypass

函数名	LL_RCC_LSE_EnableBypass
函数原形	__STATIC_INLINE void LL_RCC_LSE_EnableBypass(void)
功能描述	使能 LSE 外部振荡器
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.18 函数 LL_RCC_LSE_DisableBypass

描述了函数 LL_RCC_LSE_DisableBypass

表41-22 函数 LL_RCC_LSE_DisableBypass

函数名	LL_RCC_LSE_DisableBypass
函数原形	__STATIC_INLINE void LL_RCC_LSE_DisableBypass(void)
功能描述	禁用 LSE 外部振荡器
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.19 函数 LL_RCC_LSE_SetDriveCapability

描述了函数 LL_RCC_LSE_SetDriveCapability

表41-23 函数 LL_RCC_LSE_SetDriveCapability

函数名	LL_RCC_LSE_SetDriveCapability
函数原形	__STATIC_INLINE void LL_RCC_LSE_SetDriveCapability(uint32_t LSEDrive)
功能描述	设置 LSE 驱动能力
输入参数	LSEDrive：驱动能力
输出参数	无
返回值	无
先决条件	无

LSEDrive 可选参数:

表41-24 LSEDrive 可选参数

参数	描述
LL_RCC_LSEDRIVE_CLOSE	关闭 LSE 驱动
LL_RCC_LSEDRIVE_LOW	低驱动能力
LL_RCC_LSEDRIVE_MEDIUM	中驱动能力
LL_RCC_LSEDRIVE_HIGH	高驱动能力

41.2.20 函数 LL_RCC_LSE_GetDriveCapability

描述了函数 LL_RCC_LSE_GetDriveCapability

表41-25 函数 LL_RCC_LSE_GetDriveCapability

函数名	LL_RCC_LSE_GetDriveCapability
函数原形	__STATIC_INLINE uint32_t LL_RCC_LSE_GetDriveCapability(void)
功能描述	获取 LSE 驱动能力
输入参数	无
输出参数	无
返回值	LSE 驱动能力
先决条件	无

41.2.21 函数 LL_RCC_LSE_EnableCSS

描述了函数 LL_RCC_LSE_EnableCSS

表41-26 函数 LL_RCC_LSE_EnableCSS

函数名	LL_RCC_LSE_EnableCSS
函数原形	__STATIC_INLINE void LL_RCC_LSE_EnableCSS(void)
功能描述	使能 LSE 时钟安全系统
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.22 函数 LL_RCC_LSE_DisableCSS

描述了函数 LL_RCC_LSE_DisableCSS

表41-27 函数 LL_RCC_LSE_DisableCSS

函数名	LL_RCC_LSE_DisableCSS
函数原形	__STATIC_INLINE void LL_RCC_LSE_DisableCSS(void)
功能描述	禁用 LSE 时钟安全系统
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.23 函数 LL_RCC_LSE_IsReady

描述了函数 LL_RCC_LSE_IsReady

表41-28 函数 LL_RCC_LSE_IsReady

函数名	LL_RCC_LSE_IsReady
函数原形	__STATIC_INLINE uint32_t LL_RCC_LSE_IsReady(void)
功能描述	检查 LSE 是否准备就绪
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.24 函数 LL_RCC_LSE_IsCSSDetected

描述了函数 LL_RCC_LSE_IsCSSDetected

表41-29 函数 LL_RCC_LSE_IsCSSDetected

函数名	LL_RCC_LSE_IsCSSDetected
函数原形	__STATIC_INLINE uint32_t LL_RCC_LSE_IsCSSDetected(void)
功能描述	检查 LSE 是否失效
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.25 函数 LL_RCC_LSI_Enable

描述了函数 LL_RCC_LSI_Enable

表41-30 函数 LL_RCC_LSI_Enable

函数名	LL_RCC_LSI_Enable
函数原形	__STATIC_INLINE void LL_RCC_LSI_Enable(void)
功能描述	使能 LSI
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.26 函数 LL_RCC_LSI_Disable

描述了函数 LL_RCC_LSI_Disable

表41-31 函数 LL_RCC_LSI_Disable

函数名	LL_RCC_LSI_Disable
函数原形	__STATIC_INLINE void LL_RCC_LSI_Disable(void)
功能描述	禁用 LSI
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.27 函数 LL_RCC_LSI_IsReady

描述了函数 LL_RCC_LSI_IsReady

表41-32 函数 LL_RCC_LSI_IsReady

函数名	LL_RCC_LSI_IsReady
函数原形	__STATIC_INLINE uint32_t LL_RCC_LSI_IsReady(void)
功能描述	检查 LSI 是否准备就绪

输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.28 函数 LL_RCC_LSCO_Enable

描述了函数 LL_RCC_LSCO_Enable

表41-33 函数 LL_RCC_LSCO_Enable

函数名	LL_RCC_LSCO_Enable
函数原形	__STATIC_INLINE void LL_RCC_LSCO_Enable(void)
功能描述	使能 LSC
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.29 函数 LL_RCC_LSCO_Disable

描述了函数 LL_RCC_LSCO_Disable

表41-34 函数 LL_RCC_LSCO_Disable

函数名	LL_RCC_LSCO_Disable
函数原形	__STATIC_INLINE void LL_RCC_LSCO_Disable(void)
功能描述	禁用 LSC
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.30 函数 LL_RCC_LSCO_SetSource

描述了函数 LL_RCC_LSCO_SetSource

表41-35 函数 LL_RCC_LSCO_SetSource

函数名	LL_RCC_LSCO_SetSource
函数原形	__STATIC_INLINE void LL_RCC_LSCO_SetSource(uint32_t Source)
功能描述	设置 LSC 源
输入参数	Source：LSC 源
输出参数	无
返回值	无

先决条件	无

Source 可选参数:

表41-36 Source 可选参数

参数	描述
LL_RCC_LSCO_CLKSOURCE_LSI	将 LSI 作为 LSC 源
LL_RCC_LSCO_CLKSOURCE_LSE	将 LSE 作为 LSC 源

41.2.31 函数 LL_RCC_LSCO_GetSource

描述了函数 LL_RCC_LSCO_GetSource

表41-37 函数 LL_RCC_LSCO_GetSource

函数名	LL_RCC_LSCO_GetSource
函数原形	__STATIC_INLINE uint32_t LL_RCC_LSCO_GetSource(void)
功能描述	获取 LSC 源
输入参数	无
输出参数	无
返回值	LSC 源
先决条件	无

41.2.32 函数 LL_RCC_SetSysClkSource

描述了函数 LL_RCC_SetSysClkSource

表41-38 函数 LL_RCC_SetSysClkSource

函数名	LL_RCC_SetSysClkSource
函数原形	__STATIC_INLINE void LL_RCC_SetSysClkSource(uint32_t Source)
功能描述	设置系统时钟源
输入参数	Source：时钟源
输出参数	无
返回值	无
先决条件	无

Source 可选参数:

表41-39 Source 可选参数

参数	描述
LL_RCC_SYS_CLKSOURCE_HSISYS	HSI 作为系统时钟源
LL_RCC_SYS_CLKSOURCE_HSE	HSE 作为系统时钟源
LL_RCC_SYS_CLKSOURCE_PLL	PLL 作为系统时钟源
LL_RCC_SYS_CLKSOURCE_LSI	LSI 作为系统时钟源

LL_RCC_SYS_CLKSOURCE_LSE	LSE 作为系统时钟源

41.2.33 函数 LL_RCC_GetSysClkSource

描述了函数 LL_RCC_GetSysClkSource

表41-40 函数 LL_RCC_GetSysClkSource

函数名	LL_RCC_GetSysClkSource
函数原形	__STATIC_INLINE uint32_t LL_RCC_GetSysClkSource(void)
功能描述	获取系统时钟源
输入参数	无
输出参数	无
返回值	时钟源
先决条件	无

41.2.34 函数 LL_RCC_SetAHBPrescaler

描述了函数 LL_RCC_SetAHBPrescaler

表41-41 函数 LL_RCC_SetAHBPrescaler

函数名	LL_RCC_SetAHBPrescaler
函数原形	__STATIC_INLINE void LL_RCC_SetAHBPrescaler(uint32_t Prescaler)
功能描述	设置 AHB 预分频值
输入参数	Prescaler：预分频值
输出参数	无
返回值	无
先决条件	无

Prescaler 可选参数：

表41-42 Prescaler 可选参数

参数	描述
LL_RCC_SYSCLK_DIV_1	不分频
LL_RCC_SYSCLK_DIV_2	2 分频
LL_RCC_SYSCLK_DIV_4	4 分频
LL_RCC_SYSCLK_DIV_8	8 分频
LL_RCC_SYSCLK_DIV_16	16 分频
LL_RCC_SYSCLK_DIV_64	64 分频
LL_RCC_SYSCLK_DIV_128	128 分频
LL_RCC_SYSCLK_DIV_256	256 分频
LL_RCC_SYSCLK_DIV_512	512 分频

41.2.35 函数 LL_RCC_SetAPB1Prescaler

描述了函数 LL_RCC_SetAPB1Prescaler

表41-43 函数 LL_RCC_SetAPB1Prescaler

函数名	LL_RCC_SetAPB1Prescaler
函数原形	__STATIC_INLINE void LL_RCC_SetAPB1Prescaler(uint32_t Prescaler)
功能描述	设置 APB 预分频值
输入参数	Prescaler：预分频值
输出参数	无
返回值	无
先决条件	无

Prescaler 可选参数：

表41-44 Prescaler 可选参数

参数	描述
LL_RCC_APB1_DIV_1	不分频
LL_RCC_APB1_DIV_2	2 分频
LL_RCC_APB1_DIV_4	4 分频
LL_RCC_APB1_DIV_8	8 分频
LL_RCC_APB1_DIV_16	16 分频

41.2.36 函数 LL_RCC_SetHSIDiv

描述了函数 LL_RCC_SetHSIDiv

表41-45 函数 LL_RCC_SetHSIDiv

函数名	LL_RCC_SetHSIDiv
函数原形	__STATIC_INLINE void LL_RCC_SetHSIDiv(uint32_t HSIDiv)
功能描述	设置 HSI 时钟分频
输入参数	HSIDiv：分频值
输出参数	无
返回值	无
先决条件	无

HSIDiv 可选参数：

表41-46 HSIDiv 可选参数

参数	描述
LL_RCC_HSI_DIV_1	不分频
LL_RCC_HSI_DIV_2	2 分频

LL_RCC_HSI_DIV_4	4 分频
LL_RCC_HSI_DIV_8	8 分频
LL_RCC_HSI_DIV_16	16 分频
LL_RCC_HSI_DIV_32	32 分频
LL_RCC_HSI_DIV_64	64 分频
LL_RCC_HSI_DIV_128	128 分频

41.2.37 函数 LL_RCC_GetAHBPrescaler

描述了函数 LL_RCC_GetAHBPrescaler

表41-47 函数 LL_RCC_GetAHBPrescaler

函数名	LL_RCC_GetAHBPrescaler
函数原形	__STATIC_INLINE uint32_t LL_RCC_GetAHBPrescaler(void)
功能描述	获取 AHB 时钟预分频值
输入参数	无
输出参数	AHB 时钟预分频值
返回值	无
先决条件	无

41.2.38 函数 LL_RCC_GetAPB1Prescaler

描述了函数 LL_RCC_GetAPB1Prescaler

表41-48 函数 LL_RCC_GetAPB1Prescaler

函数名	LL_RCC_GetAPB1Prescaler
函数原形	__STATIC_INLINE uint32_t LL_RCC_GetAPB1Prescaler(void)
功能描述	获取 APB1 时钟预分频值
输入参数	无
输出参数	无
返回值	APB1 时钟预分频值
先决条件	无

41.2.39 函数 LL_RCC_GetHSIDiv

描述了函数 LL_RCC_GetHSIDiv

表41-49 函数 LL_RCC_GetHSIDiv

函数名	LL_RCC_GetHSIDiv
函数原形	__STATIC_INLINE uint32_t LL_RCC_GetHSIDiv(void)
功能描述	获取 HSI 时钟预分频值
输入参数	无

输出参数	无
返回值	HSI 时钟预分频值
先决条件	无

41.2.40 函数 LL_RCC_ConfigMCO

描述了函数 LL_RCC_ConfigMCO

表41-50 函数 LL_RCC_ConfigMCO

函数名	LL_RCC_ConfigMCO
函数原形	__STATIC_INLINE void LL_RCC_ConfigMCO(uint32_t MCOxSource, uint32_t MCOxPrescaler)
功能描述	配置 MCO
输入参数 1	MCOxSource：MCO 输出源
输入参数 2	MCOxPrescaler：MCO 输出源预分频值
输出参数	无
返回值	无
先决条件	无

MCOxSource 可选参数：

表41-51 MCOxSource 可选参数

参数	描述
LL_RCC_MCO1SOURCE_NOCLOCK	无时钟输出
LL_RCC_MCO1SOURCE_SYSCLK	输出系统时钟
LL_RCC_MCO1SOURCE_HSI	输出 HSI
LL_RCC_MCO1SOURCE_HSE	输出 HSE
LL_RCC_MCO1SOURCE_PLLCLK	输出 PLL
LL_RCC_MCO1SOURCE_LSI	输出 LSI
LL_RCC_MCO1SOURCE_LSE	输出 LSE

MCOxPrescaler 可选参数：

表41-52 MCOxPrescaler 可选参数

参数	描述
LL_RCC_MCO1_DIV_1	不分频
LL_RCC_MCO1_DIV_2	2 分频
LL_RCC_MCO1_DIV_4	4 分频
LL_RCC_MCO1_DIV_8	8 分频
LL_RCC_MCO1_DIV_16	16 分频
LL_RCC_MCO1_DIV_32	32 分频

LL_RCC_MCO1_DIV_64	64 分频
LL_RCC_MCO1_DIV_128	128 分频

41.2.41 函数 LL_RCC_GetMCOClockSource

描述了函数 LL_RCC_GetMCOClockSource

表41-53 函数 LL_RCC_GetMCOClockSource

函数名	LL_RCC_GetMCOClockSource
函数原形	__STATIC_INLINE uint32_t LL_RCC_GetMCOClockSource(uint32_t MCOx)
功能描述	获取 MCO 时钟源
输入参数	MCOx：MCO 端口
输出参数	无
返回值	MCO 输出源
先决条件	无

MCOx 可选参数：

表41-54 MCOx 可选参数

参数	描述
LL_RCC_MCO1_CLKSOURCE	MCO1 时钟

41.2.42 函数 LL_RCC_GetMCODiv

描述了函数 LL_RCC_GetMCODiv

表41-55 函数 LL_RCC_GetMCODiv

函数名	LL_RCC_GetMCODiv
函数原形	__STATIC_INLINE uint32_t LL_RCC_GetMCODiv(uint32_t MCOx)
功能描述	获取 MCO 分频
输入参数	MCOx：MCO 端口
输出参数	无
返回值	MCO 分频值
先决条件	无

MCOx 可选参数：

表41-56 MCOx 可选参数

参数	描述
LL_RCC_MCO1_CLKSOURCE	MCO1 时钟

41.2.43 函数 LL_RCC_SetPVDClockSource

描述了函数 LL_RCC_SetPVDClockSource

表41-57 函数 LL_RCC_SetPVDClockSource

函数名	LL_RCC_SetPVDClockSource
函数原形	__STATIC_INLINE void LL_RCC_SetPVDClockSource(uint32_t PVDSource)
功能描述	设置 PVD 时钟源
输入参数	PVDSource：PVD 时钟源
输出参数	无
返回值	无
先决条件	无

PVDSource 可选参数:

表41-58 PVDSource 可选参数

参数	描述
LL_RCC_PVD_CLKSOURCE_PCLK1	APB 时钟
LL_RCC_PVD_CLKSOURCE_LSC	LSC 时钟

41.2.44 函数 LL_RCC_GetPVDClockSource

描述了函数 LL_RCC_GetPVDClockSource

表41-59 函数 LL_RCC_GetPVDClockSource

函数名	LL_RCC_GetPVDClockSource
函数原形	__STATIC_INLINE uint32_t LL_RCC_GetPVDClockSource(void)
功能描述	获取 PVD 时钟源
输入参数	无
输出参数	无
返回值	PVD 时钟源
先决条件	无

41.2.45 函数 LL_RCC_SetCOMPClockSource

描述了函数 LL_RCC_SetCOMPClockSource

表41-60 函数 LL_RCC_SetCOMPClockSource

函数名	LL_RCC_SetCOMPClockSource
函数原形	__STATIC_INLINE void LL_RCC_SetCOMPClockSource(uint32_t COMPxSource)
功能描述	设置 COMP 时钟源
输入参数	COMPxSource：COMP 时钟源
输出参数	无
返回值	无
先决条件	无

COMPxSource 可选参数:

表41-61 COMPxSource 可选参数

参数	描述
LL_RCC_COMP1_CLKSOURCE_PCLK1	APB 时钟作为 COMP1 时钟源
LL_RCC_COMP1_CLKSOURCE_LSC	LSC 时钟作为 COMP1 时钟源
LL_RCC_COMP2_CLKSOURCE_PCLK1	APB 时钟作为 COMP2 时钟源
LL_RCC_COMP2_CLKSOURCE_LSC	LSC 时钟作为 COMP2 时钟源

41.2.46 函数 LL_RCC_GetCOMPClockSource

描述了函数 LL_RCC_GetCOMPClockSource

表41-62 函数 LL_RCC_GetCOMPClockSource

函数名	LL_RCC_GetCOMPClockSource
函数原形	__STATIC_INLINE uint32_t LL_RCC_GetCOMPClockSource(uint32_t COMPx)
功能描述	获取 COMP 时钟源
输入参数	COMPx：比较器
输出参数	无
返回值	比较器时钟源
先决条件	无

COMPx 可选参数:

表41-63 COMPx 可选参数

参数	描述
LL_RCC_COMP1_CLKSOURCE	COMP1 时钟源
LL_RCC_COMP2_CLKSOURCE	COMP2 时钟源

41.2.47 函数 LL_RCC_SetLPTIMClockSource

描述了函数 LL_RCC_SetLPTIMClockSource

表41-64 函数 LL_RCC_SetLPTIMClockSource

函数名	LL_RCC_SetLPTIMClockSource
函数原形	__STATIC_INLINE void LL_RCC_SetLPTIMClockSource(uint32_t LPTIMxSource)
功能描述	设置 LPTIM 时钟源
输入参数	LPTIMxSource：LPTIM 时钟源
输出参数	无
返回值	无
先决条件	无

LPTIMxSource 可选参数:

表41-65 LPTIMxSource 可选参数

参数	描述
LL_RCC_LPTIM1_CLKSOURCE_PCLK1	APB 时钟
LL_RCC_LPTIM1_CLKSOURCE_LSI	LSI 时钟
LL_RCC_LPTIM1_CLKSOURCE_NONE	无时钟
LL_RCC_LPTIM1_CLKSOURCE_LSE	LSE 时钟

41.2.48 函数 LL_RCC_GetLPTIMClockSource

描述了函数 LL_RCC_GetLPTIMClockSource

表41-66 函数 LL_RCC_GetLPTIMClockSource

函数名	LL_RCC_GetLPTIMClockSource
函数原形	__STATIC_INLINE uint32_t LL_RCC_GetLPTIMClockSource(uint32_t LPTIMx)
功能描述	获取 LPTIM 时钟源
输入参数	LPTIMx：LPTIM
输出参数	无
返回值	LPTIM 时钟源
先决条件	无

LPTIMx 可选参数:

表41-67 LPTIMx 可选参数

参数	描述
LL_RCC_LPTIM1_CLKSOURCE	LPTIM1 时钟源

41.2.49 函数 LL_RCC_SetRTCClockSource

描述了函数 LL_RCC_SetRTCClockSource

表41-68 函数 LL_RCC_SetRTCClockSource

函数名	LL_RCC_SetRTCClockSource
函数原形	__STATIC_INLINE void LL_RCC_SetRTCClockSource(uint32_t Source)
功能描述	获取 RTC 时钟源
输入参数	Source：时钟源
输出参数	无
返回值	无
先决条件	无

Source 可选参数:

表41-69 Source 可选参数

参数	描述

LL_RCC_RTC_CLKSOURCE_NONE	无时钟
LL_RCC_RTC_CLKSOURCE_LSE	LSE 时钟
LL_RCC_RTC_CLKSOURCE_LSI	LSI 时钟
LL_RCC_RTC_CLKSOURCE_HSE_DIV128	HSE 时钟 128 分频

41.2.50 函数 LL_RCC_GetRTCClockSource

描述了函数 LL_RCC_GetRTCClockSource

表41-70 函数 LL_RCC_GetRTCClockSource

函数名	LL_RCC_GetRTCClockSource
函数原形	__STATIC_INLINE uint32_t LL_RCC_GetRTCClockSource(void)
功能描述	获取 RTC 时钟源
输入参数	无
输出参数	无
返回值	RTC 时钟源
先决条件	无

41.2.51 函数 LL_RCC_EnableRTC

描述了函数 LL_RCC_EnableRTC

表41-71 函数 LL_RCC_EnableRTC

函数名	LL_RCC_EnableRTC
函数原形	__STATIC_INLINE void LL_RCC_EnableRTC(void)
功能描述	使能 RTC
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.52 函数 LL_RCC_DisableRTC

描述了函数 LL_RCC_DisableRTC

表41-72 函数 LL_RCC_DisableRTC

函数名	LL_RCC_DisableRTC
函数原形	__STATIC_INLINE void LL_RCC_DisableRTC(void)
功能描述	禁用 RTC
输入参数	无
输出参数	无
返回值	无

先决条件	无

41.2.53 函数 LL_RCC_IsEnabledRTC

描述了函数 LL_RCC_IsEnabledRTC

表41-73 函数 LL_RCC_IsEnabledRTC

函数名	LL_RCC_IsEnabledRTC
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsEnabledRTC(void)
功能描述	检查 RTC 是否是使能
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.54 函数 LL_RCC_ForceBackupDomainReset

描述了函数 LL_RCC_ForceBackupDomainReset

表41-74 函数 LL_RCC_ForceBackupDomainReset

函数名	LL_RCC_ForceBackupDomainReset
函数原形	__STATIC_INLINE void LL_RCC_ForceBackupDomainReset(void)
功能描述	强制备份域寄存器复位
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.55 函数 LL_RCC_ReleaseBackupDomainReset

描述了函数 LL_RCC_ReleaseBackupDomainReset

表41-75 函数 LL_RCC_ReleaseBackupDomainReset

函数名	LL_RCC_ReleaseBackupDomainReset
函数原形	__STATIC_INLINE void LL_RCC_ReleaseBackupDomainReset(void)
功能描述	释放备份域寄存器复位
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.56 函数 LL_RCC_PLL_Enable

描述了函数 LL_RCC_PLL_Enable

表41-76 函数 LL_RCC_PLL_Enable

函数名	LL_RCC_PLL_Enable
函数原形	__STATIC_INLINE void LL_RCC_PLL_Enable(void)
功能描述	使能 PLL
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.57 函数 LL_RCC_PLL_Disable

描述了函数 LL_RCC_PLL_Disable

表41-77 函数 LL_RCC_PLL_Disable

函数名	LL_RCC_PLL_Disable
函数原形	__STATIC_INLINE void LL_RCC_PLL_Disable(void)
功能描述	禁用 PLL
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.58 函数 LL_RCC_PLL_IsReady

描述了函数 LL_RCC_PLL_IsReady

表41-78 函数 LL_RCC_PLL_IsReady

函数名	LL_RCC_PLL_IsReady
函数原形	__STATIC_INLINE uint32_t LL_RCC_PLL_IsReady(void)
功能描述	检查 PLL 是否准备就绪
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.59 函数 LL_RCC_PLL_SetMainSource

描述了函数 LL_RCC_PLL_SetMainSource

表41-79 函数 LL_RCC_PLL_SetMainSource

函数名	LL_RCC_PLL_SetMainSource
函数原形	__STATIC_INLINE void LL_RCC_PLL_SetMainSource(uint32_t PLLSource)

功能描述	设置 PLL 时钟源
输入参数	PLLSource：PLL 时钟源
输出参数	无
返回值	无
先决条件	无

PLLSource 可选参数:

表41-80 PLLSource 可选参数

参数	描述
LL_RCC_PLLSOURCE_HSI	HIS 时钟
LL_RCC_PLLSOURCE_HSE	HSE 时钟

41.2.60 函数 LL_RCC_PLL_GetMainSource

描述了函数 LL_RCC_PLL_GetMainSource

表41-81 函数 LL_RCC_PLL_GetMainSource

函数名	LL_RCC_PLL_GetMainSource
函数原形	__STATIC_INLINE uint32_t LL_RCC_PLL_GetMainSource(void)
功能描述	获取 PLL 时钟源
输入参数	无
输出参数	无
返回值	PLL 时钟源
先决条件	无

41.2.61 函数 LL_RCC_ClearFlag_LSIRDY

描述了函数 LL_RCC_ClearFlag_LSIRDY

表41-82 函数 LL_RCC_ClearFlag_LSIRDY

函数名	LL_RCC_ClearFlag_LSIRDY
函数原形	__STATIC_INLINE void LL_RCC_ClearFlag_LSIRDY(void)
功能描述	清除 LSIRDY 标记
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.62 函数 LL_RCC_ClearFlag_LSERDY

描述了函数 LL_RCC_ClearFlag_LSERDY

表41-83 函数 LL_RCC_ClearFlag_LSERDY

函数名	LL_RCC_ClearFlag_LSERDY
函数原形	__STATIC_INLINE void LL_RCC_ClearFlag_LSERDY(void)
功能描述	清除 LSERDY 标记
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.63 函数 LL_RCC_ClearFlag_HSIRDY

描述了函数 LL_RCC_ClearFlag_HSIRDY

表41-84 函数 LL_RCC_ClearFlag_HSIRDY

函数名	LL_RCC_ClearFlag_HSIRDY
函数原形	__STATIC_INLINE void LL_RCC_ClearFlag_HSIRDY(void)
功能描述	清除 HSIRDY 标记
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.64 函数 LL_RCC_ClearFlag_HSERDY

描述了函数 LL_RCC_ClearFlag_HSERDY

表41-85 函数 LL_RCC_ClearFlag_HSERDY

函数名	LL_RCC_ClearFlag_HSERDY
函数原形	__STATIC_INLINE void LL_RCC_ClearFlag_HSERDY(void)
功能描述	清除 HSERDY 标记
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.65 函数 LL_RCC_ClearFlag_PLLRDY

描述了函数 LL_RCC_ClearFlag_PLLRDY

表41-86 函数 LL_RCC_ClearFlag_PLLRDY

函数名	LL_RCC_ClearFlag_PLLRDY
函数原形	__STATIC_INLINE void LL_RCC_ClearFlag_PLLRDY(void)
功能描述	清除 PLLRDY 标记

输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.66 函数 LL_RCC_ClearFlag_HSECSS

描述了函数 LL_RCC_ClearFlag_HSECSS

表41-87 函数 LL_RCC_ClearFlag_HSECSS

函数名	LL_RCC_ClearFlag_HSECSS
函数原形	__STATIC_INLINE void LL_RCC_ClearFlag_HSECSS(void)
功能描述	清除 HSE CSS 中断标记
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.67 函数 LL_RCC_ClearFlag_LSECSS

描述了函数 LL_RCC_ClearFlag_LSECSS

表41-88 函数 LL_RCC_ClearFlag_LSECSS

函数名	LL_RCC_ClearFlag_LSECSS
函数原形	__STATIC_INLINE void LL_RCC_ClearFlag_LSECSS(void)
功能描述	清除 LSE CSS 中断标记
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.68 函数 LL_RCC_IsActiveFlag_LSIRDY

描述了函数 LL_RCC_IsActiveFlag_LSIRDY

表41-89 函数 LL_RCC_IsActiveFlag_LSIRDY

函数名	LL_RCC_IsActiveFlag_LSIRDY
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_LSIRDY(void)
功能描述	检查 LSIRDY 标记是否置位
输入参数	无
输出参数	无
返回值	状态位（1 或 0）

先决条件	无

41.2.69 函数 LL_RCC_IsActiveFlag_LSERDY

描述了函数 LL_RCC_IsActiveFlag_LSERDY

表41-90 函数 LL_RCC_IsActiveFlag_LSERDY

函数名	LL_RCC_IsActiveFlag_LSERDY
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_LSERDY(void)
功能描述	检查 LSERDY 标记是否置位
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.70 函数 LL_RCC_IsActiveFlag_HSIRDY

描述了函数 LL_RCC_IsActiveFlag_HSIRDY

表41-91 函数 LL_RCC_IsActiveFlag_HSIRDY

函数名	LL_RCC_IsActiveFlag_HSIRDY
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_HSIRDY(void)
功能描述	检查 HSIRDY 标记是否置位
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.71 函数 LL_RCC_IsActiveFlag_HSERDY

描述了函数 LL_RCC_IsActiveFlag_HSERDY

表41-92 函数 LL_RCC_IsActiveFlag_HSERDY

函数名	LL_RCC_IsActiveFlag_HSERDY
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_HSERDY(void)
功能描述	检查 HSERDY 标记是否置位
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.72 函数 LL_RCC_IsActiveFlag_PLLRDY

描述了函数 LL_RCC_IsActiveFlag_PLLRDY

表41-93 函数 LL_RCC_IsActiveFlag_PLLRDY

函数名	LL_RCC_IsActiveFlag_PLLRDY
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_PLLRDY(void)
功能描述	检查 PLLRDY 标记是否置位
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.73 函数 LL_RCC_IsActiveFlag_HSECSS

描述了函数 LL_RCC_IsActiveFlag_HSECSS

表41-94 函数 LL_RCC_IsActiveFlag_HSECSS

函数名	LL_RCC_IsActiveFlag_HSECSS
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_HSECSS (void)
功能描述	检查 HSE CSS 中断标记是否置位
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.74 函数 LL_RCC_IsActiveFlag_LSECSS

描述了函数 LL_RCC_IsActiveFlag_LSECSS

表41-95 函数 LL_RCC_IsActiveFlag_LSECSS

函数名	LL_RCC_IsActiveFlag_LSECSS
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_LSECSS(void)
功能描述	检查 LSE CSS 中断标记是否置位
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.75 函数 LL_RCC_IsActiveFlag_IWDGRST

描述了函数 LL_RCC_IsActiveFlag_IWDGRST

表41-96 函数 LL_RCC_IsActiveFlag_IWDGRST

函数名	LL_RCC_IsActiveFlag_IWDGRST
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_IWDGRST(void)

功能描述	检查 IWDGRST 中断标记是否置位
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.76 函数 LL_RCC_IsActiveFlag_OBLRST

描述了函数 LL_RCC_IsActiveFlag_OBLRST

表41-97 函数 LL_RCC_IsActiveFlag_OBLRST

函数名	LL_RCC_IsActiveFlag_OBLRST
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_OBLRST(void)
功能描述	检查 OBLRST 中断标记是否置位
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.77 函数 LL_RCC_IsActiveFlag_PINRST

描述了函数 LL_RCC_IsActiveFlag_PINRST

表41-98 函数 LL_RCC_IsActiveFlag_PINRST

函数名	LL_RCC_IsActiveFlag_PINRST
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_PINRST(void)
功能描述	检查 PINRST 中断标记是否置位
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.78 函数 LL_RCC_IsActiveFlag_SFTRST

描述了函数 LL_RCC_IsActiveFlag_SFTRST

表41-99 函数 LL_RCC_IsActiveFlag_SFTRST

函数名	LL_RCC_IsActiveFlag_SFTRST
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_SFTRST(void)
功能描述	检查 SFTRST 中断标记是否置位
输入参数	无
输出参数	无

返回值	状态位（1 或 0）
先决条件	无

41.2.79 函数 LL_RCC_IsActiveFlag_WWDGRST

描述了函数 LL_RCC_IsActiveFlag_WWDGRST

表41-100 函数 LL_RCC_IsActiveFlag_WWDGRST

函数名	LL_RCC_IsActiveFlag_WWDGRST
函数原形	LL_RCC_IsActiveFlag_WWDGRST
功能描述	检查WWDGRST 中断标记是否置位
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.80 函数 LL_RCC_IsActiveFlag_PWRRST

描述了函数 LL_RCC_IsActiveFlag_PWRRST

表41-101 函数 LL_RCC_IsActiveFlag_PWRRST

函数名	LL_RCC_IsActiveFlag_PWRRST
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_PWRRST(void)
功能描述	检查 PWRRST 中断标记是否置位
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.81 函数 LL_RCC_ClearResetFlags

描述了函数 LL_RCC_ClearResetFlags

表41-102 函数 LL_RCC_ClearResetFlags

函数名	LL_RCC_ClearResetFlags
函数原形	__STATIC_INLINE void LL_RCC_ClearResetFlags(void)
功能描述	清除复位标记
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.82 函数 LL_RCC_EnableNRSTFilter

描述了函数 LL_RCC_EnableNRSTFilter

表41-103 函数 LL_RCC_EnableNRSTFilter

函数名	LL_RCC_EnableNRSTFilter
函数原形	__STATIC_INLINE void LL_RCC_EnableNRSTFilter(void)
功能描述	使能 NRST 滤波
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.83 函数 LL_RCC_DisableNRSTFilter

描述了函数 LL_RCC_DisableNRSTFilter

表41-104 函数 LL_RCC_DisableNRSTFilter

函数名	LL_RCC_DisableNRSTFilter
函数原形	__STATIC_INLINE void LL_RCC_DisableNRSTFilter(void)
功能描述	禁用 NRST 滤波
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.84 函数 LL_RCC_IsEnableNRSTFilter

描述了函数 LL_RCC_IsEnableNRSTFilter

表41-105 函数 LL_RCC_IsEnableNRSTFilter

函数名	LL_RCC_IsEnableNRSTFilter
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsEnableNRSTFilter(void)
功能描述	检查 NRST 滤波是否开启
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.85 函数 LL_RCC_EnableIT_LSIRDY

描述了函数 LL_RCC_EnableIT_LSIRDY

表41-106 函数 LL_RCC_EnableIT_LSIRDY

函数名	LL_RCC_EnableIT_LSIRDY
函数原形	__STATIC_INLINE void LL_RCC_EnableIT_LSIRDY(void)
功能描述	使能 LSIRDY 中断
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.86 函数 LL_RCC_EnableIT_LSERDY

描述了函数 LL_RCC_EnableIT_LSERDY

表41-107 函数 LL_RCC_EnableIT_LSERDY

函数名	LL_RCC_EnableIT_LSERDY
函数原形	__STATIC_INLINE void LL_RCC_EnableIT_LSERDY(void)
功能描述	使能 LSERDY 中断
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.87 函数 LL_RCC_EnableIT_HSIRDY

描述了函数 LL_RCC_EnableIT_HSIRDY

表41-108 函数 LL_RCC_EnableIT_HSIRDY

函数名	LL_RCC_EnableIT_HSIRDY
函数原形	__STATIC_INLINE void LL_RCC_EnableIT_HSIRDY(void)
功能描述	使能 HSIRDY 中断
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.88 函数 LL_RCC_EnableIT_HSERDY

描述了函数 LL_RCC_EnableIT_HSERDY

表41-109 函数 LL_RCC_EnableIT_HSERDY

函数名	LL_RCC_EnableIT_HSERDY
函数原形	__STATIC_INLINE void LL_RCC_EnableIT_HSERDY(void)
功能描述	使能 HSERDY 中断

输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.89 函数 LL_RCC_EnableIT_PLLRDY

描述了函数 LL_RCC_EnableIT_PLLRDY

表41-110 函数 LL_RCC_EnableIT_PLLRDY

函数名	LL_RCC_EnableIT_PLLRDY
函数原形	__STATIC_INLINE void LL_RCC_EnableIT_PLLRDY(void)
功能描述	使能 PLLRDY 中断
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.90 函数 LL_RCC_DisableIT_LSIRDY

描述了函数 LL_RCC_DisableIT_LSIRDY

表41-111 函数 LL_RCC_DisableIT_LSIRDY

函数名	LL_RCC_DisableIT_LSIRDY
函数原形	__STATIC_INLINE void LL_RCC_DisableIT_LSIRDY(void)
功能描述	禁用 LSIRDY 中断
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.91 函数 LL_RCC_DisableIT_LSERDY

描述了函数 LL_RCC_DisableIT_LSERDY

表41-112 函数 LL_RCC_DisableIT_LSERDY

函数名	LL_RCC_DisableIT_LSERDY
函数原形	__STATIC_INLINE void LL_RCC_DisableIT_LSERDY(void)
功能描述	禁用 LSERDY 中断
输入参数	无
输出参数	无
返回值	无

先决条件	无

41.2.92 函数 LL_RCC_DisableIT_HSIRDY

描述了函数 LL_RCC_DisableIT_HSIRDY

表41-113 函数 LL_RCC_DisableIT_HSIRDY

函数名	LL_RCC_DisableIT_HSIRDY
函数原形	__STATIC_INLINE void LL_RCC_DisableIT_HSIRDY(void)
功能描述	禁用 HSIRDY 中断
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.93 函数 LL_RCC_DisableIT_HSERDY

描述了函数 LL_RCC_DisableIT_HSERDY

表41-114 函数 LL_RCC_DisableIT_HSERDY

函数名	LL_RCC_DisableIT_HSERDY
函数原形	__STATIC_INLINE void LL_RCC_DisableIT_HSERDY(void)
功能描述	禁用 HSERDY 中断
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.94 函数 LL_RCC_DisableIT_PLLRDY

描述了函数 LL_RCC_DisableIT_PLLRDY

表41-115 函数 LL_RCC_DisableIT_PLLRDY

函数名	LL_RCC_DisableIT_PLLRDY
函数原形	__STATIC_INLINE void LL_RCC_DisableIT_PLLRDY(void)
功能描述	禁用 PLLRDY 中断
输入参数	无
输出参数	无
返回值	无
先决条件	无

41.2.95 函数 LL_RCC_IsEnabledIT_LSIRDY

描述了函数 LL_RCC_IsEnabledIT_LSIRDY

表41-116 函数 LL_RCC_IsEnabledIT_LSIRDY

函数名	LL_RCC_IsEnabledIT_LSIRDY
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_LSIRDY(void)
功能描述	检查 LSIRDY 中断标志是否置位
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.96 函数 LL_RCC_IsEnabledIT_LSERDY

描述了函数 LL_RCC_IsEnabledIT_LSERDY

表41-117 函数 LL_RCC_IsEnabledIT_LSERDY

函数名	LL_RCC_IsEnabledIT_LSERDY
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_LSERDY(void)
功能描述	检查 LSERDY 中断标志是否置位
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.97 函数 LL_RCC_IsEnabledIT_HSIRDY

描述了函数 LL_RCC_IsEnabledIT_HSIRDY

表41-118 函数 LL_RCC_IsEnabledIT_HSIRDY

函数名	LL_RCC_IsEnabledIT_HSIRDY
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_HSIRDY(void)
功能描述	检查 HSIRDY 中断标志是否置位
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.98 函数 LL_RCC_IsEnabledIT_HSERDY

描述了函数 LL_RCC_IsEnabledIT_HSERDY

表41-119 函数 LL_RCC_IsEnabledIT_HSERDY

函数名	LL_RCC_IsEnabledIT_HSERDY
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_HSERDY(void)

功能描述	检查 HSERDY 中断标志是否置位
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.99 函数 LL_RCC_IsEnabledIT_PLLRDY

描述了函数 LL_RCC_IsEnabledIT_PLLRDY

表41-120 函数 LL_RCC_IsEnabledIT_PLLRDY

函数名	LL_RCC_IsEnabledIT_PLLRDY
函数原形	__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_PLLRDY(void)
功能描述	检查 PLLRDY 中断标志是否置位
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

41.2.100 函数 LL_RCC_GetSystemClocksFreq

描述了函数 LL_RCC_GetSystemClocksFreq

表41-121 函数 LL_RCC_GetSystemClocksFreq

函数名	LL_RCC_GetSystemClocksFreq
函数原形	void LL_RCC_GetSystemClocksFreq(LL_RCC_ClocksTypeDef *RCC_Clocks)
功能描述	检查 PLLRDY 中断标志是否置位
输入参数	RCC_Clocks：RCC 时钟结构体
输出参数	无
返回值	系统时钟频率
先决条件	无

41.2.101 函数 LL_RCC_GetMCOClockFreq

描述了函数 LL_RCC_GetMCOClockFreq

表41-122 函数 LL_RCC_GetMCOClockFreq

函数名	LL_RCC_GetMCOClockFreq
函数原形	uint32_t LL_RCC_GetMCOClockFreq(uint32_t MCOx)
功能描述	获取 MCO 时钟频率
输入参数	MCOx：
输出参数	无

返回值	MCO 时钟频率
先决条件	无

41.2.102 函数 LL_RCC_GetLSCClockFreq

描述了函数 LL_RCC_GetLSCClockFreq

表41-123 函数 LL_RCC_GetLSCClockFreq

函数名	LL_RCC_GetLSCClockFreq
函数原形	uint32_t LL_RCC_GetLSCClockFreq(void)
功能描述	获取 LSC 时钟频率
输入参数	无
输出参数	无
返回值	LSC 时钟频率
先决条件	无

41.2.103 函数 LL_RCC_GetPVDClockFreq

描述了函数 LL_RCC_GetPVDClockFreq

表41-124 函数 LL_RCC_GetPVDClockFreq

函数名	LL_RCC_GetPVDClockFreq
函数原形	uint32_t LL_RCC_GetPVDClockFreq(void);
功能描述	获取 PVD 时钟频率
输入参数	无
输出参数	无
返回值	PVD 时钟频率
先决条件	无

41.2.104 函数 LL_RCC_GetCOMPClockFreq

描述了函数 LL_RCC_GetCOMPClockFreq

表41-125 函数 LL_RCC_GetCOMPClockFreq

函数名	LL_RCC_GetCOMPClockFreq
函数原形	uint32_t LL_RCC_GetCOMPClockFreq(uint32_t COMPx)
功能描述	获取 COMP 时钟频率
输入参数	COMPx：COMP 实例
输出参数	无
返回值	COMP 时钟频率
先决条件	无

COMPx 可选参数:

表41-126 COMPx 可选参数

参数	描述
LL_RCC_COMP1_CLKSOURCE	COMP1 时钟源
LL_RCC_COMP2_CLKSOURCE	COMP2 时钟源

41.2.105 函数 LL_RCC_GetLPTIMClockFreq

描述了函数 LL_RCC_GetLPTIMClockFreq

表41-127 函数 LL_RCC_GetLPTIMClockFreq

函数名	LL_RCC_GetLPTIMClockFreq
函数原形	uint32_t LL_RCC_GetLPTIMClockFreq(uint32_t LPTIMx);
功能描述	获取 LPTIM 时钟频率
输入参数	LPTIMx：LPTIM
输出参数	无
返回值	LPTIM 时钟频率
先决条件	无

LPTIMx 可选参数:

表41-128 LPTIMx 可选参数

参数	描述
LL_RCC_LPTIM1_CLKSOURCE	LPTIM1 时钟源

41.2.106 函数 LL_RCC_GetRTCClockFreq

描述了函数 LL_RCC_GetRTCClockFreq

表41-129 函数 LL_RCC_GetRTCClockFreq

函数名	LL_RCC_GetRTCClockFreq
函数原形	uint32_t LL_RCC_GetRTCClockFreq(void);
功能描述	获取 RTC 时钟频率
输入参数	无
输出参数	无
返回值	RTC 时钟频率
先决条件	无

42 LL 实时时钟通用驱动程序（RTC）

实时时钟（real time clock）是一个独立的定时器。RTC 模块拥有一组连续计数的计数器，在相应软件配置下，可提供时钟日历的功能。修改计数器的值可以重新设置系统当前的时间和日期。RTC 模块和时钟配置系统处于后备区域，即在系统复位后，RTC 的设置和时间维持不变。

42.1 RTC 固件驱动寄存器结构

42.1.1 LL_RTC_InitTypeDef

LL_RTC_InitTypeDef，定义于文件“air001xx_ll_rtc.h”如下：

```
typedef struct
{
uint32_t AsynchPrescaler;
uint32_t OutPutSource;
} LL_RTC_InitTypeDef;
```

字段说明：

表42-1 LL_RTC_InitTypeDef 字段说明

字段	描述
AsynchPrescaler	RTC 预分频器值，0x00~0xFFFFF
OutPutSource	RTC 输出源

参数说明：

OutPutSource 可选参数：

表42-2 OutPutSource 可选参数

参数	描述
LL_RTC_CALIB_OUTPUT_NONE	禁用输出
LL_RTC_CALIB_OUTPUT_RTCCLOCK	输出源是 RTC clock 的 64 分频
LL_RTC_CALIB_OUTPUT_ALARM	输出源是闹钟脉冲信号
LL_RTC_CALIB_OUTPUT_SECOND	输出源是秒脉冲信号

42.1.2 LL_RTC_TimeTypeDef

LL_RTC_TimeTypeDef，定义于文件“air001xx_ll_rtc.h”如下：

```
typedef struct
{
uint8_t Hours;
uint8_t Minutes;
uint8_t Seconds;
```

```
} LL_RTC_TimeTypeDef;
```

字段说明：

表42-3 LL_RTC_TimeTypeDef 字段说明

字段	描述
Hours	设置 RTC 时间：时
Minutes	设置 RTC 时间：分
Seconds	设置 RTC 时间：秒

42.1.3 LL_RTC_AlarmTypeDef

```
typedef struct
{
LL_RTC_TimeTypeDef AlarmTime;
} LL_RTC_AlarmTypeDef
```

字段说明：

表42-4 LL_RTC_AlarmTypeDef 字段说明

字段	描述
AlarmTime	闹钟时间

42.2 RTC 固件库函数

表42-5 RTC 固件库函数说明

函数名	描述
LL_RTC_SetAsynchPrescaler	设置预分频因子
LL_RTC_GetDivider	获取预分频因子
LL_RTC_SetOutputSource	设置输出源
LL_RTC_GetOutPutSource	获取输出源
LL_RTC_EnableWriteProtection	使能 RTC 寄存器的写保护
LL_RTC_DisableWriteProtection	屏蔽 RTC 寄存器的写保护
LL_RTC_TIME_Set	设置时间
LL_RTC_TIME_Get	获取时间
LL_RTC_ALARM_Set	设置闹钟
LL_RTC_CAL_SetCoarseDigital	设置 RTC 校准值
LL_RTC_CAL_GetCoarseDigital	获取 RTC 校准值
LL_RTC_IsActiveFlag_ALR	检查闹钟标志是否置位
LL_RTC_ClearFlag_ALR	清除闹钟标志

LL_RTC_IsActiveFlag_RS	检查寄存器同步标志是否置位
LL_RTC_ClearFlag_RS	清除寄存器同步标志
LL_RTC_IsActiveFlag_OW	检查计数器溢出标志是否置位
LL_RTC_ClearFlag_OW	清除计数器溢出标志
LL_RTC_IsActiveFlag_SEC	检查秒标志是否置位
LL_RTC_ClearFlag_SEC	清除秒标志
LL_RTC_IsActiveFlag_RTOF	检查 RTC 操作完成标志是否置位
LL_RTC_EnableIT_ALR	使能闹钟中断
LL_RTC_DisableIT_ALR	屏蔽闹钟中断
LL_RTC_IsEnabledIT_ALR	检查是否使能闹钟中断
LL_RTC_EnableIT_SEC	使能秒中断
LL_RTC_DisableIT_SEC	禁用秒中断
LL_RTC_IsEnabledIT_SEC	检查是否使能秒中断
LL_RTC_EnableIT_OW	使能计数器溢出中断
LL_RTC_DisableIT_OW	禁用计数器溢出中断
LL_RTC_IsEnabledIT_OW	检查是否使能计数器溢出中断
LL_RTC_DeInit	将 RTC 寄存器配置为缺省值
LL_RTC_Init	RTC 初始化函数
LL_RTC_StructInit	将结构体 LL_RTC_InitTypeDef 的字段设置为默认值。
LL_RTC_TIME_Init	初始化 RTC 的当前时间
LL_RTC_TIME_StructInit	将结构体 LL_RTC_TimeTypeDef 的字段设置为默认值
LL_RTC_ALARM_Init	初始化 RTC 的闹钟时间
LL_RTC_ALARM_StructInit	将结构体 LL_RTC_AlarmTypeDef 的字段设置为默认值
LL_RTC_EnterInitMode	进入 RTC 初始化模式
LL_RTC_ExitInitMode	退出 RTC 初始化模式
LL_RTC_WaitForSynchro	等待 RTC 寄存器与 RTC APB 时钟同步
LL_RTC_TIME_SetCounter	设置时间计数器
LL_RTC_ALARM_SetCounter	设置闹钟计数器

42.2.1 函数 LL_RTC_SetAsynchPrescaler

描述了函数 LL_RTC_SetAsynchPrescaler

表42-6 函数 LL_RTC_SetAsynchPrescaler

函数名	LL_RTC_SetAsynchPrescaler
函数原形	__STATIC_INLINE void LL_RTC_SetAsynchPrescaler(RTC_TypeDef *RTCx, uint32_t AsynchPrescaler)
功能描述	设置预分频因子
输入参数 1	RTCx：RTC 实例
输入参数 2	AsynchPrescaler：预分频因子（0-0xFFFFF）
输出参数	无
返回值	无
先决条件	无

42.2.2 函数 LL_RTC_GetDivider

描述了函数 LL_RTC_GetDivider

表42-7 函数 LL_RTC_GetDivider

函数名	LL_RTC_GetDivider
函数原形	__STATIC_INLINE uint32_t LL_RTC_GetDivider(RTC_TypeDef *RTCx)
功能描述	获取预分频因子
输入参数	RTCx：RTC 实例
输出参数	无
返回值	预分频因子（0-0xFFFFF）
先决条件	无

42.2.3 函数 LL_RTC_SetOutputSource

描述了函数 LL_RTC_SetOutputSource

表42-8 函数 LL_RTC_SetOutputSource

函数名	LL_RTC_SetOutputSource
函数原形	__STATIC_INLINE void LL_RTC_SetOutputSource(RTC_TypeDef *RTCx, uint32_t OutputSource)
功能描述	设置 RTC 的输出源
输入参数 1	RTCx：RTC 实例
输入参数 2	OutputSource：输出源
输出参数	无
返回值	无
先决条件	无

OutputSource 可选参数：

表42-9 OutputSource 可选参数

参数	描述
LL_RTC_CALIB_OUTPUT_NONE	无输出源

LL_RTC_CALIB_OUTPUT_RTCCLOCK	输出源是 RTC clock 的 64 分频
LL_RTC_CALIB_OUTPUT_ALARM	输出源是闹钟脉冲信号
LL_RTC_CALIB_OUTPUT_SECOND	输出源是秒脉冲信号

42.2.4 函数 LL_RTC_GetOutPutSource

描述了函数 LL_RTC_GetOutPutSource

表42-10 函数 LL_RTC_GetOutPutSource

函数名	LL_RTC_GetOutPutSource
函数原形	__STATIC_INLINE uint32_t LL_RTC_GetOutPutSource(RTC_TypeDef *RTCx)
功能描述	获取输出源
输入参数	RTCx：RTC 实例
输出参数	无
返回值	输出源
先决条件	无

42.2.5 函数 LL_RTC_EnableWriteProtection

描述了函数 LL_RTC_EnableWriteProtection

表42-11 函数 LL_RTC_EnableWriteProtection

函数名	LL_RTC_EnableWriteProtection
函数原形	__STATIC_INLINE void LL_RTC_EnableWriteProtection(RTC_TypeDef *RTCx)
功能描述	使能写保护
输入参数	RTCx：RTC 实例
输出参数	无
返回值	无
先决条件	无

42.2.6 函数 LL_RTC_DisableWriteProtection

描述了函数 LL_RTC_DisableWriteProtection

表42-12 函数 LL_RTC_DisableWriteProtection

函数名	LL_RTC_DisableWriteProtection
函数原形	__STATIC_INLINE void LL_RTC_DisableWriteProtection(RTC_TypeDef *RTCx)
功能描述	屏蔽写保护
输入参数	RTCx：RTC 实例
输出参数	无
返回值	无
先决条件	无

42.2.7 函数 LL_RTC_TIME_Set

描述了函数 LL_RTC_TIME_Set

表42-13 函数 LL_RTC_TIME_Set

函数名	LL_RTC_TIME_Set
函数原形	__STATIC_INLINE void LL_RTC_TIME_Set(RTC_TypeDef *RTCx, uint32_t TimeCounter)
功能描述	设置时间计数器
输入参数 1	RTCx：RTC 实例
输入参数 2	TimeCounter：时间计数器值（0x00-0xFFFFFFFF）
输出参数	无
返回值	无
先决条件	无

42.2.8 函数 LL_RTC_TIME_Get

描述了函数 LL_RTC_TIME_Get

表42-14 函数 LL_RTC_TIME_Get

函数名	LL_RTC_TIME_Get
函数原形	__STATIC_INLINE uint32_t LL_RTC_TIME_Get(RTC_TypeDef *RTCx)
功能描述	获取时间计数器值
输入参数	RTCx：RTC 实例
输出参数	无
返回值	时间计数器值（0x00-0xFFFFFFFF）
先决条件	无

42.2.9 函数 LL_RTC_ALARM_Set

描述了函数 LL_RTC_ALARM_Set

表42-15 函数 LL_RTC_ALARM_Set

函数名	LL_RTC_ALARM_Set
函数原形	__STATIC_INLINE void LL_RTC_ALARM_Set(RTC_TypeDef *RTCx, uint32_t AlarmCounter)
功能描述	设置闹钟计数器
输入参数 1	RTCx：RTC 实例
输入参数 2	AlarmCounter：闹钟计数器值（0x00-0xFFFFFFFF）
输出参数	无
返回值	无
先决条件	无

42.2.10 函数 LL_RTC_CAL_SetCoarseDigital

描述了函数 LL_RTC_CAL_SetCoarseDigital

表42-16 函数 LL_RTC_CAL_SetCoarseDigital

函数名	LL_RTC_CAL_SetCoarseDigital
函数原形	LL_RTC_CAL_SetCoarseDigital(RTC_TypeDef *RTCx, uint32_t Value)
功能描述	设置 RTC 校准值
输入参数 1	RTCx：RTC 实例
输入参数 2	Value：校准值
输出参数	无
返回值	无
先决条件	无

42.2.11 函数 LL_RTC_CAL_GetCoarseDigital

描述了函数 LL_RTC_CAL_GetCoarseDigital

表42-17 函数 LL_RTC_CAL_GetCoarseDigital

函数名	LL_RTC_CAL_GetCoarseDigital
函数原形	__STATIC_INLINE uint32_t LL_RTC_CAL_GetCoarseDigital(RTC_TypeDef *RTCx)
功能描述	获取 RTC 校准值
输入参数	RTCx：RTC 实例
输出参数	无
返回值	校准值
先决条件	无

42.2.12 函数 LL_RTC_IsActiveFlag_ALR

描述了函数 LL_RTC_IsActiveFlag_ALR

表42-18 函数 LL_RTC_IsActiveFlag_ALR

函数名	LL_RTC_IsActiveFlag_ALR
函数原形	__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_ALR(RTC_TypeDef *RTCx)
功能描述	检查闹钟标志是否置位
输入参数	RTCx：RTC 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

42.2.13 函数 LL_RTC_ClearFlag_ALR

描述了函数 LL_RTC_ClearFlag_ALR

表42-19 函数 LL_RTC_ClearFlag_ALR

函数名	LL_RTC_ClearFlag_ALR
函数原形	__STATIC_INLINE void LL_RTC_ClearFlag_ALR(RTC_TypeDef *RTCx)
功能描述	清除闹钟标志
输入参数	RTCx：RTC 实例
输出参数	无
返回值	无
先决条件	无

42.2.14 函数 LL_RTC_IsActiveFlag_RS

描述了函数 LL_RTC_IsActiveFlag_RS

表42-20 函数 LL_RTC_IsActiveFlag_RS

函数名	LL_RTC_IsActiveFlag_RS
函数原形	__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_RS(RTC_TypeDef *RTCx)
功能描述	检查寄存器同步标志是否置位
输入参数	RTCx：RTC 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

42.2.15 函数 LL_RTC_ClearFlag_RS

描述了函数 LL_RTC_ClearFlag_RS

表42-21 函数 LL_RTC_ClearFlag_RS

函数名	LL_RTC_ClearFlag_RS
函数原形	__STATIC_INLINE void LL_RTC_ClearFlag_RS(RTC_TypeDef *RTCx)
功能描述	清除寄存器同步标志
输入参数	RTCx：RTC 实例
输出参数	无
返回值	无
先决条件	无

42.2.16 函数 LL_RTC_IsActiveFlag_OW

描述了函数 LL_RTC_IsActiveFlag_OW

表42-22 函数 LL_RTC_IsActiveFlag_OW

函数名	LL_RTC_IsActiveFlag_OW
函数原形	__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_OW(RTC_TypeDef *RTCx)

功能描述	检查溢出标志是否置位
输入参数	RTCx：RTC 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

42.2.17 函数 LL_RTC_ClearFlag_OW

描述了函数 LL_RTC_ClearFlag_OW

表42-23 函数 LL_RTC_ClearFlag_OW

函数名	LL_RTC_ClearFlag_OW
函数原形	__STATIC_INLINE void LL_RTC_ClearFlag_OW(RTC_TypeDef *RTCx)
功能描述	清除溢出标志
输入参数	RTCx：RTC 实例
输出参数	无
返回值	无
先决条件	无

42.2.18 函数 LL_RTC_IsActiveFlag_SEC

描述了函数 LL_RTC_IsActiveFlag_SEC

表42-24 函数 LL_RTC_IsActiveFlag_SEC

函数名	LL_RTC_IsActiveFlag_SEC
函数原形	__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_SEC(RTC_TypeDef *RTCx)
功能描述	检查秒标志是否置位
输入参数	RTCx：RTC 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

42.2.19 函数 LL_RTC_ClearFlag_SEC

描述了函数 LL_RTC_ClearFlag_SEC

表42-25 函数 LL_RTC_ClearFlag_SEC

函数名	LL_RTC_ClearFlag_SEC
函数原形	__STATIC_INLINE void LL_RTC_ClearFlag_SEC(RTC_TypeDef *RTCx)
功能描述	清除秒标志
输入参数	RTCx：RTC 实例
输出参数	无

返回值	无
先决条件	无

42.2.20 函数 LL_RTC_IsActiveFlag_RTOF

描述了函数 LL_RTC_IsActiveFlag_RTOF

表42-26 函数 LL_RTC_IsActiveFlag_RTOF

函数名	LL_RTC_IsActiveFlag_RTOF
函数原形	__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_RTOF(RTC_TypeDef *RTCx)
功能描述	检查 RTC 操作完成标志是否置位
输入参数	RTCx：RTC 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

42.2.21 函数 LL_RTC_EnableIT_ALR

描述了函数 LL_RTC_EnableIT_ALR

表42-27 函数 LL_RTC_EnableIT_ALR

函数名	LL_RTC_EnableIT_ALR
函数原形	__STATIC_INLINE void LL_RTC_EnableIT_ALR(RTC_TypeDef *RTCx)
功能描述	使能闹钟中断
输入参数	RTCx：RTC 实例
输出参数	无
返回值	无
先决条件	无

42.2.22 函数 LL_RTC_DisableIT_ALR

描述了函数 LL_RTC_DisableIT_ALR

表42-28 函数 LL_RTC_DisableIT_ALR

函数名	LL_RTC_DisableIT_ALR
函数原形	__STATIC_INLINE void LL_RTC_DisableIT_ALR(RTC_TypeDef *RTCx)
功能描述	禁用闹钟中断
输入参数	RTCx：RTC 实例
输出参数	无
返回值	无
先决条件	无

42.2.23 函数 LL_RTC_IsEnabledIT_ALR

描述了函数 LL_RTC_IsEnabledIT_ALR

表42-29 函数 LL_RTC_IsEnabledIT_ALR

函数名	LL_RTC_IsEnabledIT_ALR
函数原形	__STATIC_INLINE uint32_t LL_RTC_IsEnabledIT_ALR(RTC_TypeDef *RTCx)
功能描述	检查是否使能闹钟中断
输入参数	RTCx：RTC 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

42.2.24 函数 LL_RTC_EnableIT_SEC

描述了函数 LL_RTC_EnableIT_SEC

表42-30 函数 LL_RTC_EnableIT_SEC

函数名	LL_RTC_EnableIT_SEC
函数原形	__STATIC_INLINE void LL_RTC_EnableIT_SEC(RTC_TypeDef *RTCx)
功能描述	使能秒中断
输入参数	RTCx：RTC 实例
输出参数	无
返回值	无
先决条件	无

42.2.25 函数 LL_RTC_DisableIT_SEC

描述了函数 LL_RTC_DisableIT_SEC

表42-31 函数 LL_RTC_DisableIT_SEC

函数名	LL_RTC_DisableIT_SEC
函数原形	__STATIC_INLINE void LL_RTC_DisableIT_SEC(RTC_TypeDef *RTCx)
功能描述	禁用秒中断
输入参数	RTCx：RTC 实例
输出参数	无
返回值	无
先决条件	无

42.2.26 函数 LL_RTC_IsEnabledIT_SEC

描述了函数 LL_RTC_IsEnabledIT_SEC

表42-32 函数 LL_RTC_IsEnabledIT_SEC

函数名	LL_RTC_IsEnabledIT_SEC
函数原形	__STATIC_INLINE uint32_t LL_RTC_IsEnabledIT_SEC(RTC_TypeDef *RTCx)
功能描述	检查是否使能秒中断
输入参数	RTCx：RTC 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

42.2.27 函数 LL_RTC_EnableIT_OW

描述了函数 LL_RTC_EnableIT_OW

表42-33 函数 LL_RTC_EnableIT_OW

函数名	LL_RTC_EnableIT_OW
函数原形	__STATIC_INLINE void LL_RTC_EnableIT_OW(RTC_TypeDef *RTCx)
功能描述	使能溢出中断
输入参数	RTCx：RTC 实例
输出参数	无
返回值	无
先决条件	无

42.2.28 函数 LL_RTC_DisableIT_OW

描述了函数 LL_RTC_DisableIT_OW

表42-34 函数 LL_RTC_DisableIT_OW

函数名	LL_RTC_DisableIT_OW
函数原形	__STATIC_INLINE void LL_RTC_DisableIT_OW(RTC_TypeDef *RTCx)
功能描述	禁用溢出中断
输入参数	RTCx：RTC 实例
输出参数	无
返回值	无
先决条件	无

42.2.29 函数 LL_RTC_IsEnabledIT_OW

描述了函数 LL_RTC_IsEnabledIT_OW

表42-35 函数 LL_RTC_IsEnabledIT_OW

函数名	LL_RTC_IsEnabledIT_OW
函数原形	__STATIC_INLINE uint32_t LL_RTC_IsEnabledIT_OW (RTC_TypeDef *RTCx)
功能描述	检查是否使能秒中断

输入参数	RTCx：RTC 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

42.2.30 函数 LL_RTC_DeInit

描述了函数 LL_RTC_DeInit

表42-36 函数 LL_RTC_DeInit

函数名	LL_RTC_DeInit
函数原形	ErrorStatus LL_RTC_DeInit(RTC_TypeDef *RTCx)
功能描述	将 RTC 寄存器配置为默认复位值
输入参数	RTCx：RTC 实例
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

42.2.31 函数 LL_RTC_Init

描述了函数 LL_RTC_Init

表42-37 函数 LL_RTC_Init

函数名	LL_RTC_Init
函数原形	ErrorStatus LL_RTC_Init(RTC_TypeDef *RTCx, LL_RTC_InitTypeDef *RTC_InitStruct)
功能描述	RTC 初始化函数
输入参数 1	RTCx：RTC 实例
输入参数 2	RTC_InitStruct：指向包含 RTC 外设配置信息的 LL_RTC_InitTypeDef 结构体的指针
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

42.2.32 函数 LL_RTC_StructInit

描述了函数 LL_RTC_StructInit

表42-38 函数 LL_RTC_StructInit

函数名	LL_RTC_StructInit
函数原形	void LL_RTC_StructInit(LL_RTC_InitTypeDef *RTC_InitStruct)
功能描述	将结构体 LL_RTC_InitTypeDef 的字段设置为默认值
输入参数	无

输出参数	RTC_InitStruct：指向将被初始化的 LL_RTC_InitTypeDef 结构的指针
返回值	无
先决条件	无

42.2.33 函数 LL_RTC_TIME_Init

描述了函数 LL_RTC_TIME_Init

表42-39 函数 LL_RTC_TIME_Init

函数名	LL_RTC_TIME_Init
函数原形	ErrorStatus LL_RTC_TIME_Init(RTC_TypeDef *RTCx, uint32_t RTC_Format, LL_RTC_TimeTypeDef *RTC_TimeStruct)
功能描述	设置 RTC 的当前时间
输入参数 1	RTCx：RTC 实例
输入参数 2	RTC_Format：RTC 时间格式，值可以是 LL_RTC_FORMAT_BIN、LL_RTC_FORMAT_BCD
输出参数	RTC_TimeStruct：指向包含 RTC 时间配置信息的 LL_RTC_TimeTypeDef 结构体的指针
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

42.2.34 函数 LL_RTC_TIME_StructInit

描述了函数 LL_RTC_TIME_StructInit

表42-40 函数 LL_RTC_TIME_StructInit

函数名	LL_RTC_TIME_StructInit
函数原形	void LL_RTC_TIME_StructInit(LL_RTC_TimeTypeDef *RTC_TimeStruct)
功能描述	将结构体 LL_RTC_TimeTypeDef 的字段设置为默认值
输入参数	无
输出参数	RTC_TimeStruct：指向将被初始化的 LL_RTC_TimeTypeDef 结构的指针
返回值	无
先决条件	无

42.2.35 函数 LL_RTC_ALARM_Init

描述了函数 LL_RTC_ALARM_Init

表42-41 函数 LL_RTC_ALARM_Init

函数名	LL_RTC_ALARM_Init
函数原形	ErrorStatus LL_RTC_ALARM_Init(RTC_TypeDef *RTCx, uint32_t RTC_Format, LL_RTC_AlarmTypeDef *RTC_AlarmStruct)
功能描述	设置 RTC 的闹钟时间
输入参数 1	RTCx：RTC 实例

输入参数 2	RTC_Format：RTC 时间格式，值可以是 LL_RTC_FORMAT_BIN、LL_RTC_FORMAT_BCD
输出参数	RTC_AlarmStruct：指向包含 RTC 时间配置信息的 LL_RTC_AlarmTypeDef 结构体的指针
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

42.2.36 函数 LL_RTC_ALARM_StructInit

描述了函数 LL_RTC_ALARM_StructInit

表42-42 函数 LL_RTC_ALARM_StructInit

函数名	LL_RTC_ALARM_StructInit
函数原形	void LL_RTC_ALARM_StructInit(LL_RTC_AlarmTypeDef *RTC_AlarmStruct)
功能描述	将结构体 LL_RTC_AlarmTypeDef 的字段设置为默认值
输入参数	无
输出参数	RTC_AlarmStruct：指向将被初始化的 LL_RTC_AlarmTypeDef 结构的指针
返回值	无
先决条件	无

42.2.37 函数 LL_RTC_EnterInitMode

描述了函数 LL_RTC_EnterInitMode

表42-43 函数 LL_RTC_EnterInitMode

函数名	LL_RTC_EnterInitMode
函数原形	ErrorStatus LL_RTC_EnterInitMode(RTC_TypeDef *RTCx)
功能描述	进入 RTC 的初始化模式
输入参数	RTCx：RTC 实例
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

42.2.38 函数 LL_RTC_ExitInitMode

描述了函数 LL_RTC_ExitInitMode

表42-44 函数 LL_RTC_ExitInitMode

函数名	LL_RTC_ExitInitMode
函数原形	ErrorStatus LL_RTC_ExitInitMode(RTC_TypeDef *RTCx)
功能描述	退出 RTC 的初始化模式
输入参数	RTCx：RTC 实例
输出参数	无

返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

42.2.39 函数 LL_RTC_WaitForSynchro

描述了函数 LL_RTC_WaitForSynchro

表42-45 函数 LL_RTC_WaitForSynchro

函数名	LL_RTC_WaitForSynchro
函数原形	ErrorStatus LL_RTC_WaitForSynchro(RTC_TypeDef *RTCx)
功能描述	等待 RTC 寄存器与 RTC APB 时钟同步
输入参数	RTCx：RTC 实例
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

42.2.40 函数 LL_RTC_TIME_SetCounter

描述了函数 LL_RTC_TIME_SetCounter

表42-46 函数 LL_RTC_TIME_SetCounter

函数名	LL_RTC_TIME_SetCounter
函数原形	ErrorStatus LL_RTC_TIME_SetCounter(RTC_TypeDef *RTCx, uint32_t TimeCounter)
功能描述	设置 RTC 时间计数器值
输入参数 1	RTCx：RTC 实例
输入参数 2	TimeCounter：RTC 时间计数器值（0-0xFFFFFFFF）
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

42.2.41 函数 LL_RTC_ALARM_SetCounter

描述了函数 LL_RTC_ALARM_SetCounter

表42-47 函数 LL_RTC_ALARM_SetCounter

函数名	LL_RTC_ALARM_SetCounter
函数原形	ErrorStatus LL_RTC_ALARM_SetCounter(RTC_TypeDef *RTCx, uint32_t AlarmCounter)
功能描述	设置 RTC 闹钟计数器值
输入参数 1	RTCx：RTC 实例
输入参数 2	AlarmCounter：RTC 闹钟计数器值（0-0xFFFFFFFF）
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）

先决条件	无

43 LL 串行外设接口通用驱动程序（SPI）

串行外设接口(SPI)允许芯片与外部设备以半双工、全双工、单工同步的串行方式通信。此接口可以被配置成主模式/从模式，作为主模式时，能够为外部从设备提供通信时钟(SCK)。接口还能以多主配置方式工作。

43.1 SPI 固件驱动寄存器结构

43.1.1 LL_SPI_InitTypeDef

LL_SPI_InitTypeDef，定义于文件”air001xx_ll_spi.h”如下：

```
typedef struct
{
uint32_t TransferDirection;
uint32_t Mode;
uint32_t DataWidth;
uint32_t ClockPolarity;
uint32_t ClockPhase;
uint32_t NSS;
uint32_t BaudRate;
uint32_t BitOrder;
uint32_t SlaveSpeedMode;
} LL_SPI_InitTypeDef;
```

字段说明：

表43-1 LL_SPI_InitTypeDef 字段说明

字段	描述
TransferDirection	指定 SPI 单向或双向数据模式
Mode	指定 SPI 模式（主/从）
DataWidth	指定 SPI 数据宽度
ClockPolarity	指定时钟极性
ClockPhase	指定时钟相位
NSS	指定 NSS 信号是由硬件（NSS 引脚）还是由软件使用 SSI 位管理。
BaudRate	指定波特率分频值
BitOrder	指定数据传输是从 MSB 位还是 LSB 位开始
SlaveSpeedMode	指定从机速度模式

参数说明：

TransferDirection 可选参数：

表43-2 TransferDirection 可选参数

参数	描述
LL_SPI_FULL_DUPLEX	全双工模式
LL_SPI_SIMPLEX_RX	单工接收模式
LL_SPI_HALF_DUPLEX_RX	半双工接收模式
LL_SPI_HALF_DUPLEX_TX	半双工发送模式

Mode 可选参数：

表43-3 Mode 可选参数

参数	描述
LL_SPI_MODE_MASTER	主机模式
LL_SPI_MODE_SLAVE	从机模式

DataWidth 可选参数：

表43-4 DataWidth 可选参数

参数	描述
LL_SPI_DATAWIDTH_8BIT	8 位数据宽度
LL_SPI_DATAWIDTH_16BIT	16 位数据宽度

ClockPolarity 可选参数：

表43-5 ClockPolarity 可选参数

参数	描述
LL_SPI_POLARITY_LOW	空闲时时钟为低电平
LL_SPI_POLARITY_HIGH	空闲时时钟为高电平

ClockPhase 可选参数：

表43-6 ClockPhase 可选参数

参数	描述
LL_SPI_PHASE_1EDGE	数据采样从第一个时钟边沿开始
LL_SPI_PHASE_2EDGE	数据采样从第二个时钟边沿开始

NSS 可选参数：

表43-7 NSS 可选参数

参数	描述
LL_SPI_NSS_SOFT	NSS 软件管理， NSS 引脚未使用且空闲
LL_SPI_NSS_HARD_INPUT	NSS 引脚用于输入，仅在主模式下使用
LL_SPI_NSS_HARD_OUTPUT	NSS 引脚用于输出， 仅在从模式下用作片选

BaudRate 可选参数：

表43-8 BaudRate 可选参数

参数	描述
LL_SPI_BAUDRATEPRESCALER_DIV2	BaudRate = fPCLK/2
LL_SPI_BAUDRATEPRESCALER_DIV4	BaudRate = fPCLK/4
LL_SPI_BAUDRATEPRESCALER_DIV8	BaudRate = fPCLK/8
LL_SPI_BAUDRATEPRESCALER_DIV16	BaudRate = fPCLK/16
LL_SPI_BAUDRATEPRESCALER_DIV32	BaudRate = fPCLK/32
LL_SPI_BAUDRATEPRESCALER_DIV64	BaudRate = fPCLK/64
LL_SPI_BAUDRATEPRESCALER_DIV128	BaudRate = fPCLK/128
LL_SPI_BAUDRATEPRESCALER_DIV256	BaudRate = fPCLK/256

BitOrder 可选参数：

表43-9 BitOrder 可选参数

参数	描述
LL_SPI_LSB_FIRST	最低有效位优先
LL_SPI_MSB_FIRST	最高有效为优先

SlaveSpeedMode 可选参数：

表43-10 SlaveSpeedMode 可选参数

参数	描述
LL_SPI_SLAVE_SPEED_NORMAL	从机正常速度模式
LL_SPI_SLAVE_SPEED_FAST	从机快速模式

43.2 SPI 固件库函数

表43-11 SPI 固件库函数说明

函数名	描述
LL_SPI_Enable	使能SPI
LL_SPI_Disable	禁用SPI
LL_SPI_IsEnabled	检查SPI 是否使能
LL_SPI_SetMode	设置SPI 运行模式（主\从模式）
LL_SPI_GetMode	获取SPI 的运行模式（主\从模式）
LL_SPI_SetClockPhase	设置时钟相位
LL_SPI_GetClockPhase	获取时钟相位
LL_SPI_SetClockPolarity	设置时钟极性
LL_SPI_GetClockPolarity	获取时钟极性
LL_SPI_SetBaudRatePrescaler	设置波特率分频因子

LL_SPI_GetBaudRatePrescaler	获取波特率分频因子
LL_SPI_SetTransferBitOrder	设置传输位序
LL_SPI_GetTransferBitOrder	获取传输位序
LL_SPI_SetTransferDirection	设置传输方向
LL_SPI_GetTransferDirection	获取传输方向
LL_SPI_SetDataWidth	设置数据宽度
LL_SPI_GetDataWidth	获取数据宽度
LL_SPI_SetRxFIFOThreshold	设置接收阈值
LL_SPI_GetRxFIFOThreshold	获取接收阈值
LL_SPI_SetNSSMode	设置NSS 模式
LL_SPI_GetNSSMode	获取NSS 模式
LL_SPI_IsActiveFlag_RXNE	检查接收数据寄存器非空标志是否置位
LL_SPI_IsActiveFlag_TXE	检查发送数据寄存器空标志是否置位
LL_SPI_IsActiveFlag_MODF	检查 MODF 标志是否置位
LL_SPI_IsActiveFlag_OVR	检查溢出标志是否置位
LL_SPI_IsActiveFlag_BSY	检查忙标志是否置位
LL_SPI_GetRxFIFOLevel	获取FIFO 接收 level
LL_SPI_GetTxFIFOLevel	获取FIFO 发送 level
LL_SPI_ClearFlag_MODF	清除 MODF 标志
LL_SPI_ClearFlag_OVR	清除溢出标志
LL_SPI_EnableIT_ERR	使能错误中断
LL_SPI_EnableIT_RXNE	使能接收数据寄存器非空中断
LL_SPI_EnableIT_TXE	使能发送数据寄存器空中断
LL_SPI_DisableIT_ERR	禁用错误中断
LL_SPI_DisableIT_RXNE	禁用接收数据寄存器非空中断
LL_SPI_DisableIT_TXE	禁用发送数据寄存器空中断
LL_SPI_IsEnabledIT_ERR	检查是否使能错误中断
LL_SPI_IsEnabledIT_RXNE	检查是否使能接收数据寄存器非空中断
LL_SPI_IsEnabledIT_TXE	检查是否使能发送数据寄存器空中断
LL_SPI_EnableDMAReq_RX	使能DMA 接收
LL_SPI_DisableDMAReq_RX	禁用DMA 接收

LL_SPI_IsEnabledDMAReq_RX	检查是否使能DMA 接收
LL_SPI_EnableDMAReq_TX	使能DMA 发送
LL_SPI_DisableDMAReq_TX	禁用DMA 发送
LL_SPI_IsEnabledDMAReq_TX	检查是否使能DMA 发送
LL_SPI_SetDMAParity_RX	设置DMA 接收的数据总量奇偶性
LL_SPI_GetDMAParity_RX	获取DMA 接收的数据总量奇偶性
LL_SPI_SetDMAParity_TX	设置DMA 发送的数据总量奇偶性
LL_SPI_GetDMAParity_TX	获取DMA 发送的数据总量奇偶性
LL_SPI_DMA_GetRegAddr	获取数据寄存器地址
LL_SPI_ReceiveData8	从数据寄存器读 8 位数据
LL_SPI_ReceiveData16	从数据寄存器读 16 位数据
LL_SPI_TransmitData8	向数据寄存器写 8 位数据
LL_SPI_TransmitData16	向数据寄存器写 16 位数据
LL_SPI_SetSlaveSpeedMode	设置从机的速度模式
LL_SPI_GetSlaveSpeedMode	获取从机的速度模式
LL_SPI_DeInit	将 SPI 配置重设为缺省值
LL_SPI_Init	SPI 初始化
LL_SPI_StructInit	将 SPI 初始化结构体设置为默认值

43.2.1 函数 LL_SPI_Enable

描述了函数 LL_SPI_Enable

表43-12 函数 LL_SPI_Enable

函数名	LL_SPI_Enable
函数原形	_STATIC_INLINE void LL_SPI_Enable(SPI_TypeDef *SPIx)
功能描述	使能 SPI
输入参数	SPIx：SPI 实例
输出参数	无
返回值	无
先决条件	无

43.2.2 函数 LL_SPI_Disable

描述了函数 LL_SPI_Disable

表43-13 函数 LL_SPI_Disable

函数名	LL_SPI_Disable

函数原形	_STATIC_INLINE void LL_SPI_Disable (SPI_TypeDef *SPIx)
功能描述	禁用 SPI
输入参数	SPIx：SPI 实例
输出参数	无
返回值	无
先决条件	无

43.2.3 函数 LL_SPI_IsEnabled

描述了函数 LL_SPI_IsEnabled

表43-14 函数 LL_SPI_IsEnabled

函数名	LL_SPI_Enable
函数原形	_STATIC_INLINE void LL_SPI_IsEnabled (SPI_TypeDef *SPIx)
功能描述	检查 SPI 是否使能
输入参数	SPIx：SPI 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

43.2.4 函数 LL_SPI_SetMode

描述了函数 LL_SPI_SetMode

表43-15 函数 LL_SPI_SetMode

函数名	LL_SPI_SetMode
函数原形	__STATIC_INLINE void LL_SPI_SetMode(SPI_TypeDef *SPIx, uint32_t Mode)
功能描述	设置主/从模式
输入参数 1	SPIx：SPI 实例
输入参数 2	Mode：主/从模式
输出参数	无
返回值	无
先决条件	无

Mode 可选参数：

表43-16 Mode 可选参数

参数	描述
LL_SPI_MODE_MASTER	主机模式
LL_SPI_MODE_SLAVE	从机模式

43.2.5 函数 LL_SPI_GetMode

描述了函数 LL_SPI_GetMode

表43-17 函数 LL_SPI_GetMode

函数名	LL_SPI_GetMode
函数原形	_STATIC_INLINE void LL_SPI_GetMode (SPI_TypeDef *SPIx)
功能描述	获取主/从模式
输入参数	SPIx：SPI 实例
输出参数	无
返回值	主从模式
先决条件	无

43.2.6 函数 LL_SPI_SetClockPhase

描述了函数 LL_SPI_SetClockPhase

表43-18 函数 LL_SPI_SetClockPhase

函数名	LL_SPI_SetClockPhase
函数原形	__STATIC_INLINE void LL_SPI_SetClockPhase(SPI_TypeDef *SPIx, uint32_t ClockPhase)
功能描述	设置时钟相位
输入参数 1	SPIx：SPI 实例
输入参数 2	ClockPhase：时钟相位
输出参数	无
返回值	无
先决条件	无

ClockPhase 可选参数：

表43-19 ClockPhase 可选参数

参数	描述
LL_SPI_PHASE_1EDGE	数据采样从第一个时钟边沿开始
LL_SPI_PHASE_2EDGE	数据采样从第二个时钟边沿开始

43.2.7 函数 LL_SPI_GetClockPhase

描述了函数 LL_SPI_GetClockPhase

表43-20 函数 LL_SPI_GetClockPhase

函数名	LL_SPI_GetClockPhase
函数原形	_STATIC_INLINE void LL_SPI_GetClockPhase (SPI_TypeDef *SPIx)
功能描述	获取时钟相位
输入参数	SPIx：SPI 实例

输出参数	无
返回值	时钟相位
先决条件	无

43.2.8 函数 LL_SPI_SetClockPolarity

描述了函数 LL_SPI_SetClockPolarity

表43-21 函数 LL_SPI_SetClockPolarity

函数名	LL_SPI_SetClockPolarity
函数原形	__STATIC_INLINE void LL_SPI_SetClockPolarity(SPI_TypeDef *SPIx, uint32_t ClockPolarity)
功能描述	设置时钟极性
输入参数 1	SPIx：SPI 实例
输入参数 2	ClockPolarity：时钟极性
输出参数	无
返回值	无
先决条件	无

ClockPolarity 可选参数:

表43-22 ClockPolarity 可选参数

参数	描述
LL_SPI_POLARITY_LOW	空闲状态时，SCK 保持低电平
LL_SPI_POLARITY_HIGH	空闲状态时，SCK 保持高电平

43.2.9 函数 LL_SPI_GetClockPolarity

描述了函数 LL_SPI_GetClockPolarity

表43-23 函数 LL_SPI_GetClockPolarity

函数名	LL_SPI_GetClockPolarity
函数原形	_STATIC_INLINE void LL_SPI_GetClockPolarity (SPI_TypeDef *SPIx)
功能描述	获取时钟极性
输入参数	SPIx：SPI 实例
输出参数	无
返回值	时钟极性
先决条件	无

43.2.10 函数 LL_SPI_SetBaudRatePrescaler

描述了函数 LL_SPI_SetBaudRatePrescaler

表43-24 函数 LL_SPI_SetBaudRatePrescaler

函数名	LL_SPI_SetBaudRatePrescaler

函数原形	__STATIC_INLINE void LL_SPI_SetBaudRatePrescaler(SPI_TypeDef *SPIx, uint32_t BaudRate)
功能描述	设置波特率分频因子
输入参数 1	SPIx：SPI 实例
输入参数 2	BaudRate：波特率分频因子
输出参数	无
返回值	无
先决条件	无

BaudRate 可选参数：

表43-25 BaudRate 可选参数

参数	描述
LL_SPI_BAUDRATEPRESCALER_DIV2	fPCLK/2
LL_SPI_BAUDRATEPRESCALER_DIV4	fPCLK/4
LL_SPI_BAUDRATEPRESCALER_DIV8	fPCLK/8
LL_SPI_BAUDRATEPRESCALER_DIV16	fPCLK/16
LL_SPI_BAUDRATEPRESCALER_DIV32	fPCLK/32
LL_SPI_BAUDRATEPRESCALER_DIV64	fPCLK/64
LL_SPI_BAUDRATEPRESCALER_DIV128	fPCLK/128
LL_SPI_BAUDRATEPRESCALER_DIV256	fPCLK/256

43.2.11 函数 LL_SPI_GetBaudRatePrescaler

描述了函数 LL_SPI_GetBaudRatePrescaler

表43-26 函数 LL_SPI_GetBaudRatePrescaler

函数名	LL_SPI_GetBaudRatePrescaler
函数原形	__STATIC_INLINE uint32_t LL_SPI_GetBaudRatePrescaler(SPI_TypeDef *SPIx)
功能描述	获取波特率分频因子
输入参数	SPIx：SPI 实例
输出参数	无
返回值	波特率分频因子
先决条件	无

43.2.12 函数 LL_SPI_SetTransferBitOrder

描述了函数 LL_SPI_SetTransferBitOrder

表43-27 函数 LL_SPI_SetTransferBitOrder

函数名	LL_SPI_SetTransferBitOrder
函数原形	__STATIC_INLINE void LL_SPI_SetTransferBitOrder(SPI_TypeDef *SPIx, uint32_t BitOrder)
功能描述	设置传输位序
输入参数 1	SPIx：SPI 实例

输入参数 2	BitOrder：传输位序
输出参数	无
返回值	无
先决条件	无

BitOrder 可选参数：

表43-28 BitOrder 可选参数

参数	描述
LL_SPI_LSB_FIRST	最低位有效位优先
LL_SPI_MSB_FIRST	最高有效位优先

43.2.13 函数 LL_SPI_GetTransferBitOrder

描述了函数 LL_SPI_GetTransferBitOrder

表43-29 函数 LL_SPI_GetTransferBitOrder

函数名	LL_SPI_GetTransferBitOrder
函数原形	__STATIC_INLINE uint32_t LL_SPI_GetTransferBitOrder (SPI_TypeDef *SPIx)
功能描述	获取传输位序
输入参数	SPIx：SPI 实例
输出参数	无
返回值	传输位序
先决条件	无

43.2.14 函数 LL_SPI_SetTransferDirection

描述了函数 LL_SPI_SetTransferDirection

表43-30 函数 LL_SPI_SetTransferDirection

函数名	LL_SPI_SetTransferDirection
函数原形	__STATIC_INLINE void LL_SPI_SetTransferDirection(SPI_TypeDef *SPIx, uint32_t TransferDirection)
功能描述	设置传输方向
输入参数 1	SPIx：SPI 实例
输入参数 2	TransferDirection：传输方向
输出参数	无
返回值	无
先决条件	无

TransferDirection 可选参数：

表43-31 TransferDirection 可选参数

参数	描述

LL_SPI_FULL_DUPLEX	全双工
LL_SPI_SIMPLEX_RX	单工接收
LL_SPI_HALF_DUPLEX_RX	半双工接收
LL_SPI_HALF_DUPLEX_TX	半双工发送

43.2.15 函数 LL_SPI_GetTransferDirection

描述了函数 LL_SPI_GetTransferDirection

表43-32 函数 LL_SPI_GetTransferDirection

函数名	LL_SPI_GetTransferDirection
函数原形	__STATIC_INLINE uint32_t LL_SPI_GetTransferDirection (SPI_TypeDef *SPIx)
功能描述	获取传输方向
输入参数	SPIx：SPI 实例
输出参数	无
返回值	传输方向
先决条件	无

43.2.16 函数 LL_SPI_SetDataWidth

描述了函数 LL_SPI_SetDataWidth

表43-33 函数 LL_SPI_SetDataWidth

函数名	LL_SPI_SetDataWidth
函数原形	__STATIC_INLINE void LL_SPI_SetDataWidth(SPI_TypeDef *SPIx, uint32_t DataWidth)
功能描述	设置数据宽度
输入参数 1	SPIx：SPI 实例
输入参数 2	DataWidth：数据宽度
输出参数	无
返回值	无
先决条件	无

DataWidth 可选参数:

表43-34 DataWidth 可选参数

参数	描述
LL_SPI_DATAWIDTH_8BIT	数据宽度占 8 位
LL_SPI_DATAWIDTH_16BIT	数据宽度占 16 位

43.2.17 函数 LL_SPI_GetDataWidth

描述了函数 LL_SPI_GetDataWidth

表43-35 函数 LL_SPI_GetDataWidth

函数名	LL_SPI_GetDataWidth
函数原形	__STATIC_INLINE uint32_t LL_SPI_GetDataWidth (SPI_TypeDef *SPIx)
功能描述	获取数据宽度
输入参数	SPIx：SPI 实例
输出参数	无
返回值	数据宽度
先决条件	无

43.2.18 函数 LL_SPI_SetRxFIFOThreshold

描述了函数 LL_SPI_SetRxFIFOThreshold

表43-36 函数 LL_SPI_SetRxFIFOThreshold

函数名	LL_SPI_SetRxFIFOThreshold
函数原形	__STATIC_INLINE void LL_SPI_SetRxFIFOThreshold(SPI_TypeDef *SPIx, uint32_t Threshold)
功能描述	设置接收阈值
输入参数 1	SPIx：SPI 实例
输入参数 2	Threshold：接收阈值
输出参数	无
返回值	无
先决条件	无

Threshold 可选参数：

表43-37 Threshold 可选参数

参数	描述
LL_SPI_RX_FIFO_TH_HALF	接收阈值是 16 位
LL_SPI_RX_FIFO_TH_QUARTER	接收阈值是 8 位

43.2.19 函数 LL_SPI_GetRxFIFOThreshold

描述了函数 LL_SPI_GetRxFIFOThreshold

表43-38 函数 LL_SPI_GetRxFIFOThreshold

函数名	LL_SPI_GetRxFIFOThreshold
函数原形	__STATIC_INLINE uint32_t LL_SPI_GetRxFIFOThreshold(SPI_TypeDef *SPIx)
功能描述	获取接收阈值
输入参数	SPIx：SPI 实例
输出参数	无
返回值	接收阈值
先决条件	无

43.2.20 函数 LL_SPI_SetNSSMode

描述了函数 LL_SPI_SetNSSMode

表43-39 函数 LL_SPI_SetNSSMode

函数名	LL_SPI_SetNSSMode
函数原形	__STATIC_INLINE void LL_SPI_SetNSSMode(SPI_TypeDef *SPIx, uint32_t NSS)
功能描述	设置 NSS 模式
输入参数 1	SPIx：SPI 实例
输入参数 2	NSS：NSS 模式
输出参数	无
返回值	无
先决条件	无

NSS 可选参数:

表43-40 NSS 可选参数

参数	描述
LL_SPI_NSS_SOFT	NSS 内部管理，NSS 引脚未使用且空闲
LL_SPI_NSS_HARD_INPUT	NSS 引脚用于输入，仅在主模式下使用
LL_SPI_NSS_HARD_OUTPUT	NSS 引脚用于输出，仅在从模式下用作片选

43.2.21 函数 LL_SPI_GetNSSMode

描述了函数 LL_SPI_GetNSSMode

表43-41 函数 LL_SPI_GetNSSMode

函数名	LL_SPI_GetNSSMode
函数原形	__STATIC_INLINE uint32_t LL_SPI_GetNSSMode (SPI_TypeDef *SPIx)
功能描述	获取 NSS 模式
输入参数	SPIx：SPI 实例
输出参数	无
返回值	NSS 模式
先决条件	无

43.2.22 函数 LL_SPI_IsActiveFlag_RXNE

描述了函数 LL_SPI_IsActiveFlag_RXNE

表43-42 函数 LL_SPI_IsActiveFlag_RXNE

函数名	LL_SPI_IsActiveFlag_RXNE
函数原形	__STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_RXNE(SPI_TypeDef *SPIx)
功能描述	检查接收数据寄存器非空标志是否置位

输入参数 1	SPIx：SPI 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

43.2.23 函数 LL_SPI_IsActiveFlag_TXE

描述了函数 LL_SPI_IsActiveFlag_TXE

表43-43 函数 LL_SPI_IsActiveFlag_TXE

函数名	LL_SPI_IsActiveFlag_TXE
函数原形	__STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_TXE (SPI_TypeDef *SPIx)
功能描述	检查发送数据寄存器空标志是否置位
输入参数	SPIx：SPI 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

43.2.24 函数 LL_SPI_IsActiveFlag_MODF

描述了函数 LL_SPI_IsActiveFlag_MODF

表43-44 函数 LL_SPI_IsActiveFlag_MODF

函数名	LL_SPI_IsActiveFlag_MODF
函数原形	__STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_MODF (SPI_TypeDef *SPIx)
功能描述	检查 MODF 标志是否置位
输入参数	SPIx：SPI 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

43.2.25 函数 LL_SPI_IsActiveFlag_OVR

描述了函数 LL_SPI_IsActiveFlag_OVR

表43-45 函数 LL_SPI_IsActiveFlag_OVR

函数名	LL_SPI_IsActiveFlag_OVR
函数原形	__STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_OVR (SPI_TypeDef *SPIx)
功能描述	检查溢出标志是否置位
输入参数	SPIx：SPI 实例
输出参数	无
返回值	状态位（1 或 0）

先决条件	无

43.2.26 函数 LL_SPI_IsActiveFlag_BSY

描述了函数 LL_SPI_IsActiveFlag_BSY

表43-46 函数 LL_SPI_IsActiveFlag_BSY

函数名	LL_SPI_IsActiveFlag_BSY
函数原形	__STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_BSY (SPI_TypeDef *SPIx)
功能描述	检查总线忙标志是否置位
输入参数	SPIx：SPI 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

43.2.27 函数 LL_SPI_GetRxFIFOLevel

描述了函数 LL_SPI_GetRxFIFOLevel

表43-47 函数 LL_SPI_GetRxFIFOLevel

函数名	LL_SPI_GetRxFIFOLevel
函数原形	__STATIC_INLINE uint32_t LL_SPI_GetRxFIFOLevel (SPI_TypeDef *SPIx)
功能描述	获取 FIFO 接收level
输入参数	SPIx：SPI 实例
输出参数	无
返回值	FIFO 接收 level
先决条件	无

43.2.28 函数 LL_SPI_GetTxFIFOLevel

描述了函数 LL_SPI_GetTxFIFOLevel

表43-48 函数 LL_SPI_GetTxFIFOLevel

函数名	LL_SPI_GetTxFIFOLevel
函数原形	__STATIC_INLINE uint32_t LL_SPI_GetTxFIFOLevel (SPI_TypeDef *SPIx)
功能描述	获取 FIFO 发送level
输入参数	SPIx：SPI 实例
输出参数	无
返回值	FIFO 发送 level
先决条件	无

43.2.29 函数 LL_SPI_ClearFlag_MODF

描述了函数 LL_SPI_ClearFlag_MODF

表43-49 函数 LL_SPI_ClearFlag_MODF

函数名	LL_SPI_ClearFlag_MODF
函数原形	__STATIC_INLINE void LL_SPI_ClearFlag_MODF(SPI_TypeDef *SPIx)
功能描述	清除 MODF 标志
输入参数	SPIx：SPI 实例
输出参数	无
返回值	无
先决条件	无

43.2.30 函数 LL_SPI_ClearFlag_OVR

描述了函数 LL_SPI_ClearFlag_OVR

表43-50 函数 LL_SPI_ClearFlag_OVR

函数名	LL_SPI_ClearFlag_OVR
函数原形	__STATIC_INLINE void LL_SPI_ClearFlag_OVR (SPI_TypeDef *SPIx)
功能描述	清除溢出标志
输入参数	SPIx：SPI 实例
输出参数	无
返回值	无
先决条件	无

43.2.31 函数 LL_SPI_EnableIT_ERR

描述了函数 LL_SPI_EnableIT_ERR

表43-51 函数 LL_SPI_EnableIT_ERR

函数名	LL_SPI_EnableIT_ERR
函数原形	_STATIC_INLINE void LL_SPI_EnableIT_ERR (SPI_TypeDef *SPIx)
功能描述	使能错误中断
输入参数	SPIx：SPI 实例
输出参数	无
返回值	无
先决条件	无

43.2.32 函数 LL_SPI_EnableIT_RXNE

描述了函数 LL_SPI_EnableIT_RXNE

表43-52 函数 LL_SPI_EnableIT_RXNE

函数名	LL_SPI_EnableIT_RXNE
函数原形	_STATIC_INLINE void LL_SPI_EnableIT_RXNE (SPI_TypeDef *SPIx)

功能描述	使能接收数据寄存器非空中断
输入参数	SPIx：SPI 实例
输出参数	无
返回值	无
先决条件	无

43.2.33 函数 LL_SPI_EnableIT_TXE

描述了函数 LL_SPI_EnableIT_TXE

表43-53 函数 LL_SPI_EnableIT_TXE

函数名	LL_SPI_EnableIT_TXE
函数原形	_STATIC_INLINE void LL_SPI_EnableIT_TXE (SPI_TypeDef *SPIx)
功能描述	使能发送数据寄存器空中断
输入参数	SPIx：SPI 实例
输出参数	无
返回值	无
先决条件	无

43.2.34 函数 LL_SPI_DisableIT_ERR

描述了函数 LL_SPI_DisableIT_ERR

表43-54 函数 LL_SPI_DisableIT_ERR

函数名	LL_SPI_DisableIT_ERR
函数原形	_STATIC_INLINE void LL_SPI_DisableIT_ERR (SPI_TypeDef *SPIx)
功能描述	禁用错误中断
输入参数	SPIx：SPI 实例
输出参数	无
返回值	无
先决条件	无

43.2.35 函数 LL_SPI_DisableIT_RXNE

描述了函数 LL_SPI_DisableIT_RXNE

表43-55 函数 LL_SPI_DisableIT_RXNE

函数名	LL_SPI_DisableIT_RXNE
函数原形	_STATIC_INLINE void LL_SPI_DisableIT_RXNE (SPI_TypeDef *SPIx)
功能描述	禁用接收数据寄存器非空中断
输入参数	SPIx：SPI 实例
输出参数	无

返回值	无
先决条件	无

43.2.36 函数 LL_SPI_DisableIT_TXE

描述了函数 LL_SPI_DisableIT_TXE

表43-56 函数 LL_SPI_DisableIT_TXE

函数名	LL_SPI_DisableIT_TXE
函数原形	_STATIC_INLINE void LL_SPI_DisableIT_TXE (SPI_TypeDef *SPIx)
功能描述	禁用发送数据寄存器空中断
输入参数	SPIx：SPI 实例
输出参数	无
返回值	无
先决条件	无

43.2.37 函数 LL_SPI_IsEnabledIT_ERR

描述了函数 LL_SPI_IsEnabledIT_ERR

表43-57 函数 LL_SPI_IsEnabledIT_ERR

函数名	LL_SPI_IsEnabledIT_ERR
函数原形	_STATIC_INLINE void LL_SPI_IsEnabledIT_ERR (SPI_TypeDef *SPIx)
功能描述	检查是否使能错误中断
输入参数	SPIx：SPI 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

43.2.38 函数 LL_SPI_IsEnabledIT_RXNE

描述了函数 LL_SPI_IsEnabledIT_RXNE

表43-58 函数 LL_SPI_IsEnabledIT_RXNE

函数名	LL_SPI_IsEnabledIT_RXNE
函数原形	_STATIC_INLINE void LL_SPI_IsEnabledIT_RXNE (SPI_TypeDef *SPIx)
功能描述	检查是否使能接收数据寄存器非空中断
输入参数	SPIx：SPI 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

43.2.39 函数 LL_SPI_IsEnabledIT_TXE

描述了函数 LL_SPI_IsEnabledIT_TXE

表43-59 函数 LL_SPI_IsEnabledIT_TXE

函数名	LL_SPI_IsEnabledIT_TXE
函数原形	_STATIC_INLINE void LL_SPI_IsEnabledIT_TXE (SPI_TypeDef *SPIx)
功能描述	检查是否使能发送数据寄存器空中断
输入参数	SPIx：SPI 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

43.2.40 函数 LL_SPI_EnableDMAReq_RX

描述了函数 LL_SPI_EnableDMAReq_RX

表43-60 函数 LL_SPI_EnableDMAReq_RX

函数名	LL_SPI_EnableDMAReq_RX
函数原形	_STATIC_INLINE void LL_SPI_EnableDMAReq_RX (SPI_TypeDef *SPIx)
功能描述	使能 DMA 接收
输入参数	SPIx：SPI 实例
输出参数	无
返回值	无
先决条件	无

43.2.41 函数 LL_SPI_DisableDMAReq_RX

描述了函数 LL_SPI_DisableDMAReq_RX

表43-61 函数 LL_SPI_DisableDMAReq_RX

函数名	LL_SPI_DisableDMAReq_RX
函数原形	_STATIC_INLINE void LL_SPI_DisableDMAReq_RX (SPI_TypeDef *SPIx)
功能描述	禁用 DMA 接收
输入参数	SPIx：SPI 实例
输出参数	无
返回值	无
先决条件	无

43.2.42 函数 LL_SPI_IsEnabledDMAReq_RX

描述了函数 LL_SPI_IsEnabledDMAReq_RX

表43-62 函数 LL_SPI_IsEnabledDMAReq_RX

函数名	LL_SPI_IsEnabledDMAReq_RX
函数原形	_STATIC_INLINE void LL_SPI_IsEnabledDMAReq_RX (SPI_TypeDef *SPIx)
功能描述	检查是否使能 DMA 接收
输入参数	SPIx：SPI 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

43.2.43 函数 LL_SPI_EnableDMAReq_TX

描述了函数 LL_SPI_EnableDMAReq_TX

表43-63 函数 LL_SPI_EnableDMAReq_TX

函数名	LL_SPI_EnableDMAReq_TX
函数原形	_STATIC_INLINE void LL_SPI_EnableDMAReq_TX (SPI_TypeDef *SPIx)
功能描述	使能 DMA 发送
输入参数	SPIx：SPI 实例
输出参数	无
返回值	无
先决条件	无

43.2.44 函数 LL_SPI_DisableDMAReq_TX

描述了函数 LL_SPI_DisableDMAReq_TX

表43-64 函数 LL_SPI_DisableDMAReq_TX

函数名	LL_SPI_DisableDMAReq_TX
函数原形	_STATIC_INLINE void LL_SPI_DisableDMAReq_TX (SPI_TypeDef *SPIx)
功能描述	禁用 DMA 发送
输入参数	SPIx：SPI 实例
输出参数	无
返回值	无
先决条件	无

43.2.45 函数 LL_SPI_IsEnabledDMAReq_TX

描述了函数 LL_SPI_IsEnabledDMAReq_TX

表43-65 函数 LL_SPI_IsEnabledDMAReq_TX

函数名	LL_SPI_IsEnabledDMAReq_TX
函数原形	_STATIC_INLINE void LL_SPI_IsEnabledDMAReq_TX (SPI_TypeDef *SPIx)
功能描述	检查是否使能 DMA 发送

输入参数	SPIx：SPI 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

43.2.46 函数 LL_SPI_SetDMAParity_RX

描述了函数 LL_SPI_SetDMAParity_RX

表43-66 函数 LL_SPI_SetDMAParity_RX

函数名	LL_SPI_SetDMAParity_RX
函数原形	__STATIC_INLINE void LL_SPI_SetDMAParity_RX(SPI_TypeDef *SPIx, uint32_t Parity)
功能描述	设置 DMA 接收的数据总量奇偶性
输入参数 1	SPIx：SPI 实例
输入参数 2	Parity：DMA 接收的数据总量奇偶性
输出参数	无
返回值	无
先决条件	无

Parity 可选参数：

表43-67 Parity 可选参数

参数	描述
LL_SPI_DMA_PARITY_ODD	DMA 接收的数据总量为奇数
LL_SPI_DMA_PARITY_EVEN	DMA 接收的数据总量为偶数

43.2.47 函数 LL_SPI_GetDMAParity_RX

描述了函数 LL_SPI_GetDMAParity_RX

表43-68 函数 LL_SPI_GetDMAParity_RX

函数名	LL_SPI_GetDMAParity_RX
函数原形	__STATIC_INLINE void LL_SPI_GetDMAParity_RX (SPI_TypeDef *SPIx)
功能描述	获取 DMA 接收的数据总量奇偶性
输入参数	SPIx：SPI 实例
输出参数	无
返回值	DMA 接收的数据总量奇偶性
先决条件	无

43.2.48 函数 LL_SPI_SetDMAParity_TX

描述了函数 LL_SPI_SetDMAParity_TX

表43-69 函数 LL_SPI_SetDMAParity_TX

函数名	LL_SPI_SetDMAParity_TX
函数原形	__STATIC_INLINE void LL_SPI_SetDMAParity_TX (SPI_TypeDef *SPIx, uint32_t Parity)
功能描述	设置 DMA 发送的数据总量奇偶性
输入参数 1	SPIx：SPI 实例
输入参数 2	Parity：DMA 发送的数据总量奇偶性
输出参数	无
返回值	无
先决条件	无

Parity 可选参数：

表43-70 Parity 可选参数

参数	描述
LL_SPI_DMA_PARITY_ODD	DMA 接收的数据总量为奇数
LL_SPI_DMA_PARITY_EVEN	DMA 接收的数据总量为偶数

43.2.49 函数 LL_SPI_GetDMAParity_TX

描述了函数 LL_SPI_GetDMAParity_TX

表43-71 函数 LL_SPI_GetDMAParity_TX

函数名	LL_SPI_GetDMAParity_TX
函数原形	__STATIC_INLINE void LL_SPI_GetDMAParity_TX (SPI_TypeDef *SPIx)
功能描述	获取 DMA 发送的数据总量奇偶性
输入参数	SPIx：SPI 实例
输出参数	无
返回值	DMA 发送的数据总量奇偶性
先决条件	无

43.2.50 函数 LL_SPI_DMA_GetRegAddr

描述了函数 LL_SPI_DMA_GetRegAddr

表43-72 函数 LL_SPI_DMA_GetRegAddr

函数名	LL_SPI_DMA_GetRegAddr
函数原形	__STATIC_INLINE void LL_SPI_DMA_GetRegAddr (SPI_TypeDef *SPIx)
功能描述	获取数据寄存器地址
输入参数	SPIx：SPI 实例
输出参数	无
返回值	数据寄存器地址
先决条件	无

43.2.51 函数 LL_SPI_ReceiveData8

描述了函数 LL_SPI_ReceiveData8

表43-73 函数 LL_SPI_ReceiveData8

函数名	LL_SPI_ReceiveData8
函数原形	__STATIC_INLINE void LL_SPI_ReceiveData8 (SPI_TypeDef *SPIx)
功能描述	接收 8 位数据
输入参数	SPIx：SPI 实例
输出参数	无
返回值	无
先决条件	无

43.2.52 函数 LL_SPI_ReceiveData16

描述了函数 LL_SPI_ReceiveData16

表43-74 函数 LL_SPI_ReceiveData16

函数名	LL_SPI_ReceiveData16
函数原形	__STATIC_INLINE void LL_SPI_ReceiveData16 (SPI_TypeDef *SPIx)
功能描述	接收 16 位数据
输入参数	SPIx：SPI 实例
输出参数	无
返回值	无
先决条件	无

43.2.53 函数 LL_SPI_TransmitData8

描述了函数 LL_SPI_TransmitData8

表43-75 函数 LL_SPI_TransmitData8

函数名	LL_SPI_TransmitData8
函数原形	__STATIC_INLINE void LL_SPI_TransmitData8(SPI_TypeDef *SPIx, uint8_t TxData)
功能描述	发送 8 位数据
输入参数 1	SPIx：SPI 实例
输入参数 2	TxData：8 位数据
输出参数	无
返回值	无
先决条件	无

43.2.54 函数 LL_SPI_TransmitData16

描述了函数 LL_SPI_TransmitData16

表43-76 函数 LL_SPI_TransmitData16

函数名	LL_SPI_TransmitData16
函数原形	__STATIC_INLINE void LL_SPI_TransmitData16 (SPI_TypeDef *SPIx, uint16_t TxData)
功能描述	发送 16 位数据
输入参数 1	SPIx：SPI 实例
输入参数 2	TxData：16 位数据
输出参数	无
返回值	无
先决条件	无

43.2.55 函数 LL_SPI_SetSlaveSpeedMode

描述了函数 LL_SPI_SetSlaveSpeedMode

表43-77 函数 LL_SPI_SetSlaveSpeedMode

函数名	LL_SPI_SetSlaveSpeedMode
函数原形	__STATIC_INLINE void LL_SPI_SetSlaveSpeedMode(SPI_TypeDef *SPIx, uint32_t SlaveSpeedMode)
功能描述	设置从机速度模式
输入参数 1	SPIx：SPI 实例
输入参数 2	SlaveSpeedMode：从机速度模式
输出参数	无
返回值	无
先决条件	无

SlaveSpeedMode 可选参数:

表43-78 SlaveSpeedMode 可选参数

参数	描述
LL_SPI_SLAVE_SPEED_NORMAL	从机正常速度模式
LL_SPI_SLAVE_SPEED_FAST	从机快速模式

43.2.56 函数 LL_SPI_GetSlaveSpeedMode

描述了函数 LL_SPI_GetSlaveSpeedMode

表43-79 函数 LL_SPI_GetSlaveSpeedMode

函数名	LL_SPI_GetSlaveSpeedMode
函数原形	__STATIC_INLINE void LL_SPI_GetSlaveSpeedMode (SPI_TypeDef *SPIx)
功能描述	获取从机速度模式
输入参数	SPIx：SPI 实例
输出参数	无

返回值	从机速度模式
先决条件	无

43.2.57 函数 LL_SPI_DeInit

描述了函数 LL_SPI_DeInit

表43-80 函数 LL_SPI_DeInit

函数名	LL_SPI_DeInit
函数原形	__STATIC_INLINE void LL_SPI_DeInit (SPI_TypeDef *SPIx)
功能描述	取消 SPI 初始化
输入参数	SPIx：SPI 实例
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

43.2.58 函数 LL_SPI_Init

描述了函数 LL_SPI_Init

表43-81 函数 LL_SPI_Init

函数名	LL_SPI_Init
函数原形	ErrorStatus LL_SPI_Init(SPI_TypeDef *SPIx, LL_SPI_InitTypeDef *SPI_InitStruct)
功能描述	SPI 初始化
输入参数 1	SPIx：SPI 实例
输入参数 2	SPI_InitStruct：指向 LL_SPI_InitTypeDef 结构的指针，包含SPI 的配置信息
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

43.2.59 函数 LL_SPI_StructInit

描述了函数 LL_SPI_StructInit

表43-82 函数 LL_SPI_StructInit

函数名	LL_SPI_StructInit
函数原形	void LL_SPI_StructInit(LL_SPI_InitTypeDef *SPI_InitStruct)
功能描述	设置结构体 LL_SPI_InitTypeDef 的字段为默认值
输入参数	无
输出参数	SPI_InitStruct：指向 LL_SPI_InitTypeDef 结构体的指针
返回值	无
先决条件	无

44 LL SYSTEM 通用驱动程序（SYSTEM）

SYSTEM 涵盖了SYSCFG、FLASH、DUGMCU 等模块配置。

SYSCFG（系统配置控制器）主要目的是：（1）使能或者禁用在某些 IO pin 上的 I2C fast Mode Plus；（2）Remap 某些 DMA 的触发源到不同的 DMA 通道；（3）Remap 位于代码区间开始区域的存储器；（4）管理连接到 GPIO 的外部中断；（5） 管理鲁棒性特性。

FLASH（闪存存储器）主要目的：设置、获取 flash 延迟。

DUGMCU（MCU 调试模块）主要目的：读取设备 ID、设备版本号，stop 模式下使能或禁用调试模块，冻结或解冻外设。

44.1 SYSTEM 固件库函数

表44-1 SYSTEM 固件库函数说明

函数名	描述
LL_SYSCFG_SetRemapMemory	将内存映射到地址 0x00000000
LL_SYSCFG_GetRemapMemory	获取地址 0x00000000 的内存映射
LL_SYSCFG_EnableI2CAnalogFilter	使能 I2C 相关 IO 的模拟滤波使能控制
LL_SYSCFG_DisableI2CAnalogFilter	禁用 I2C 相关 IO 的模拟滤波使能控制
LL_SYSCFG_IsEnabledI2CAnalogFilter	检查是否使能 I2C 相关 IO 的模拟滤波使能控制
LL_SYSCFG_EnableTIMBreakInputs	使能COMPx 为 TIMx 刹车输入
LL_SYSCFG_DisableTIMBreakInputs	禁用COMPx 为 TIMx 刹车输入
LL_SYSCFG_IsEnabledTIMBreakInputs	检查是否使能 COMPx 为 TIMx 刹车输入
LL_SYSCFG_SetTIM1ETRSource	设置TIMER1 ETR 输入源
LL_SYSCFG_GetTIM1ETRSource	获取TIMER1 ETR 输入源
LL_SYSCFG_SetDMARemap_CH1	设置DMA 通道 1 的请求映像
LL_SYSCFG_GetDMARemap_CH1	获取DMA 通道 1 的请求映像
LL_SYSCFG_SetDMAResponseSpeed_CH1	设置DMA 通道 1 的响应速度
LL_SYSCFG_GetDMAResponseSpeed_CH1	获取DMA 通道 1 的响应速度
LL_SYSCFG_SetDMARemap_CH2	设置DMA 通道 2 的请求映像
LL_SYSCFG_GetDMARemap_CH2	获取DMA 通道 2 的请求映像
LL_SYSCFG_SetDMAResponseSpeed_CH2	设置DMA 通道 2 的响应速度
LL_SYSCFG_GetDMAResponseSpeed_CH2	获取DMA 通道 2 的响应速度

LL_SYSCFG_SetDMARemap_CH3	设置DMA 通道 3 的请求映像
LL_SYSCFG_GetDMARemap_CH3	获取DMA 通道 3 的请求映像
LL_SYSCFG_SetDMAResponseSpeed_CH3	设置DMA 通道 3 的响应速度
LL_SYSCFG_GetDMAResponseSpeed_CH3	获取DMA 通道 3 的响应速度
LL_FLASH_SetLatency	设置flash 延迟
LL_FLASH_GetLatency	获取flash 延迟
LL_DBGMCU_GetDeviceID	获取设备 ID
LL_DBGMCU_GetRevisionID	获取版本 ID
LL_DBGMCU_EnableDBGStopMode	在 STOP 模式下使能调试模块
LL_DBGMCU_DisableDBGStopMode	在 STOP 模式下禁用调试模块
LL_DBGMCU_IsEnabledDBGStopMode	在 STOP 模式下，检查是否使能调试模块
LL_DBGMCU_APB1_GRP1_FreezePeriph	冻结APB1 外设（group1 外设）
LL_DBGMCU_APB1_GRP1_UnFreezePeriph	解冻APB1 外设（group1 外设）
LL_DBGMCU_APB1_GRP1_IsFreezePeriph	检查是否冻结 APB1 外设（group1 外设）
LL_DBGMCU_APB1_GRP2_FreezePeriph	冻结APB1 外设（group2 外设）
LL_DBGMCU_APB1_GRP2_UnFreezePeriph	解冻APB1 外设（group2 外设）
LL_DBGMCU_APB1_GRP2_IsFreezePeriph	检查是否冻结 APB1 外设（group2 外设）

44.1.1 函数 LL_SYSCFG_SetRemapMemory

描述了函数 LL_SYSCFG_SetRemapMemory

表44-2 函数 LL_SYSCFG_SetRemapMemory

函数名	LL_SYSCFG_SetRemapMemory
函数原形	__STATIC_INLINE void LL_SYSCFG_SetRemapMemory(uint32_t Memory)
功能描述	将内存映射到地址 0x00000000
输入参数	Memory：待映射的内存
输出参数	无
返回值	无
先决条件	无

Memory 可选参数:

表44-3 Memory 可选参数

参数	描述
LL_SYSCFG_REMAP_FLASH	Main flash 内存映射到地址 0x00000000
LL_SYSCFG_REMAP_SYSTEMFLASH	System flash 内存映射到地址 0x00000000

LL_SYSCFG_REMAP_SRAM	SRAM 内存映射到地址 0x00000000

44.1.2 函数 LL_SYSCFG_GetRemapMemory

描述了函数 LL_SYSCFG_GetRemapMemory

表44-4 函数 LL_SYSCFG_GetRemapMemory

函数名	LL_SYSCFG_GetRemapMemory
函数原形	__STATIC_INLINE uint32_t LL_SYSCFG_GetRemapMemory(void)
功能描述	获取地址 0x00000000 的内存映射
输入参数	无
输出参数	无
返回值	地址 0x00000000 的内存映射
先决条件	无

44.1.3 函数 LL_SYSCFG_EnableI2CAnalogFilter

描述了函数 LL_SYSCFG_EnableI2CAnalogFilter

表44-5 函数 LL_SYSCFG_EnableI2CAnalogFilter

函数名	LL_SYSCFG_EnableI2CAnalogFilter
函数原形	__STATIC_INLINE void LL_SYSCFG_EnableI2CAnalogFilter(uint32_t I2CAnalogFilter)
功能描述	使能 I2C 相关 IO 的模拟过滤
输入参数	I2CAnalogFilter：I2C 相关 IO
输出参数	无
返回值	无
先决条件	无

I2CAnalogFilter 可选参数:

表44-6 I2CAnalogFilter 可选参数

参数	描述
LL_SYSCFG_I2C_ANF_PA2	I2C 对应的 PA2 引脚
LL_SYSCFG_I2C_ANF_PA3	I2C 对应的 PA3 引脚
LL_SYSCFG_I2C_ANF_PA7	I2C 对应的 PA7 引脚
LL_SYSCFG_I2C_ANF_PA8	I2C 对应的 PA8 引脚
LL_SYSCFG_I2C_ANF_PA9	I2C 对应的 PA9 引脚
LL_SYSCFG_I2C_ANF_PA10	I2C 对应的 PA10 引脚
LL_SYSCFG_I2C_ANF_PA11	I2C 对应的 PA11 引脚
LL_SYSCFG_I2C_ANF_PA12	I2C 对应的 PA12 引脚
LL_SYSCFG_I2C_ANF_PB6	I2C 对应的 PB6 引脚

LL_SYSCFG_I2C_ANF_PB7	I2C 对应的 PB7 引脚
LL_SYSCFG_I2C_ANF_PB8	I2C 对应的 PB8 引脚
LL_SYSCFG_I2C_ANF_PF0	I2C 对应的 PF0 引脚
LL_SYSCFG_I2C_ANF_PF1	I2C 对应的 PF1 引脚

44.1.4 函数 LL_SYSCFG_DisableI2CAnalogFilter

描述了函数 LL_SYSCFG_DisableI2CAnalogFilter

表44-7 函数 LL_SYSCFG_DisableI2CAnalogFilter

函数名	LL_SYSCFG_DisableI2CAnalogFilter
函数原形	__STATIC_INLINE void LL_SYSCFG_DisableI2CAnalogFilter (uint32_t I2CAnalogFilter)
功能描述	禁用 I2C 相关 IO 的模拟过滤
输入参数	I2CAnalogFilter：I2C 相关 IO
输出参数	无
返回值	无
先决条件	无

I2CAnalogFilter 可选参数:

表44-8 I2CAnalogFilter 可选参数

参数	描述
LL_SYSCFG_I2C_ANF_PA2	I2C 对应的 PA2 引脚
LL_SYSCFG_I2C_ANF_PA3	I2C 对应的 PA3 引脚
LL_SYSCFG_I2C_ANF_PA7	I2C 对应的 PA7 引脚
LL_SYSCFG_I2C_ANF_PA8	I2C 对应的 PA8 引脚
LL_SYSCFG_I2C_ANF_PA9	I2C 对应的 PA9 引脚
LL_SYSCFG_I2C_ANF_PA10	I2C 对应的 PA10 引脚
LL_SYSCFG_I2C_ANF_PA11	I2C 对应的 PA11 引脚
LL_SYSCFG_I2C_ANF_PA12	I2C 对应的 PA12 引脚
LL_SYSCFG_I2C_ANF_PB6	I2C 对应的 PB6 引脚
LL_SYSCFG_I2C_ANF_PB7	I2C 对应的 PB7 引脚
LL_SYSCFG_I2C_ANF_PB8	I2C 对应的 PB8 引脚
LL_SYSCFG_I2C_ANF_PF0	I2C 对应的 PF0 引脚
LL_SYSCFG_I2C_ANF_PF1	I2C 对应的 PF1 引脚

44.1.5 函数 LL_SYSCFG_IsEnabledI2CAnalogFilter

描述了函数 LL_SYSCFG_IsEnabledI2CAnalogFilter

表44-9 函数 LL_SYSCFG_IsEnabledI2CAnalogFilter

函数名	LL_SYSCFG_IsEnabledI2CAnalogFilter
函数原形	__STATIC_INLINE uint32_t LL_SYSCFG_IsEnabledI2CAnalogFilter(uint32_t I2CAnalogFilter)
功能描述	检查是否使能 I2C 相关 IO 的模拟过滤
输入参数	I2CAnalogFilter：I2C 相关 IO
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

I2CAnalogFilter 可选参数:

表44-10 I2CAnalogFilter 可选参数

参数	描述
LL_SYSCFG_I2C_ANF_PA2	I2C 对应的 PA2 引脚
LL_SYSCFG_I2C_ANF_PA3	I2C 对应的 PA3 引脚
LL_SYSCFG_I2C_ANF_PA7	I2C 对应的 PA7 引脚
LL_SYSCFG_I2C_ANF_PA8	I2C 对应的 PA8 引脚
LL_SYSCFG_I2C_ANF_PA9	I2C 对应的 PA9 引脚
LL_SYSCFG_I2C_ANF_PA10	I2C 对应的 PA10 引脚
LL_SYSCFG_I2C_ANF_PA11	I2C 对应的 PA11 引脚
LL_SYSCFG_I2C_ANF_PA12	I2C 对应的 PA12 引脚
LL_SYSCFG_I2C_ANF_PB6	I2C 对应的 PB6 引脚
LL_SYSCFG_I2C_ANF_PB7	I2C 对应的 PB7 引脚
LL_SYSCFG_I2C_ANF_PB8	I2C 对应的 PB8 引脚
LL_SYSCFG_I2C_ANF_PF0	I2C 对应的 PF0 引脚
LL_SYSCFG_I2C_ANF_PF1	I2C 对应的 PF1 引脚

44.1.6 函数 LL_SYSCFG_EnableTIMBreakInputs

描述了函数 LL_SYSCFG_EnableTIMBreakInputs

表44-11 函数 LL_SYSCFG_EnableTIMBreakInputs

函数名	LL_SYSCFG_EnableTIMBreakInputs
函数原形	__STATIC_INLINE void LL_SYSCFG_EnableTIMBreakInputs(uint32_t TIMBreakInputs)
功能描述	使能 COMPx 作为 TIMx 刹车输入
输入参数	TIMBreakInputs：刹车输入项
输出参数	无
返回值	无
先决条件	无

TIMBreakInputs 可选参数:

表44-12 TIMBreakInputs 可选参数

参数	描述
LL_SYSCFG_TIMBREAK_LOCKUP_TO_ALL	LOCKUP(hardfault)输出给 TIM1/TIM16/TIM17 的刹车输入
LL_SYSCFG_TIMBREAK_PVD_TO_ALL	PVD 中断与 TIM1/TIM16/TIM17 的刹车输入连接
LL_SYSCFG_TIMBREAK_COMP1_TO_TIM1	COMP1 输出作为 TIM1 刹车输入
LL_SYSCFG_TIMBREAK_COMP2_TO_TIM1	COMP2 输出作为 TIM1 刹车输入
LL_SYSCFG_TIMBREAK_COMP1_TO_TIM16	COMP1 输出作为 TIM16 刹车输入
LL_SYSCFG_TIMBREAK_COMP2_TO_TIM16	COMP2 输出作为 TIM16 刹车输入
LL_SYSCFG_TIMBREAK_COMP1_TO_TIM17	COMP1 输出作为 TIM17 刹车输入
LL_SYSCFG_TIMBREAK_COMP2_TO_TIM17	COMP2 输出作为 TIM17 刹车输入

44.1.7 函数 LL_SYSCFG_DisableTIMBreakInputs

描述了函数 LL_SYSCFG_DisableTIMBreakInputs

表44-13 函数 LL_SYSCFG_DisableTIMBreakInputs

函数名	LL_SYSCFG_DisableTIMBreakInputs
函数原形	__STATIC_INLINE void LL_SYSCFG_DisableTIMBreakInputs(uint32_t TIMBreakInputs)
功能描述	禁用 COMPx 作为 TIMx 刹车输入
输入参数	TIMBreakInputs：刹车输入项
输出参数	无
返回值	无
先决条件	无

TIMBreakInputs 可选参数:

表44-14 TIMBreakInputs 可选参数

参数	描述
LL_SYSCFG_TIMBREAK_LOCKUP_TO_ALL	LOCKUP(hardfault)输出给 TIM1/TIM16/TIM17 的刹车输入
LL_SYSCFG_TIMBREAK_PVD_TO_ALL	PVD 中断与 TIM1/TIM16/TIM17 的刹车输入连接
LL_SYSCFG_TIMBREAK_COMP1_TO_TIM1	COMP1 输出作为 TIM1 刹车输入
LL_SYSCFG_TIMBREAK_COMP2_TO_TIM1	COMP2 输出作为 TIM1 刹车输入
LL_SYSCFG_TIMBREAK_COMP1_TO_TIM16	COMP1 输出作为 TIM16 刹车输入
LL_SYSCFG_TIMBREAK_COMP2_TO_TIM16	COMP2 输出作为 TIM16 刹车输入
LL_SYSCFG_TIMBREAK_COMP1_TO_TIM17	COMP1 输出作为 TIM17 刹车输入
LL_SYSCFG_TIMBREAK_COMP2_TO_TIM17	COMP2 输出作为 TIM17 刹车输入

44.1.8 函数 LL_SYSCFG_IsEnabledTIMBreakInputs

描述了函数 LL_SYSCFG_IsEnabledTIMBreakInputs

表44-15 函数 LL_SYSCFG_IsEnabledTIMBreakInputs

函数名	LL_SYSCFG_IsEnabledTIMBreakInputs
函数原形	__STATIC_INLINE void LL_SYSCFG_IsEnabledTIMBreakInputs (uint32_t TIMBreakInputs)
功能描述	检查是否使能 COMPx 作为 TIMx 刹车输入
输入参数	TIMBreakInputs：刹车输入项
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

TIMBreakInputs 可选参数:

表44-16 TIMBreakInputs 可选参数

参数	描述
LL_SYSCFG_TIMBREAK_LOCKUP_TO_ALL	LOCKUP(hardfault)输出给 TIM1/TIM16/TIM17 的刹车输入
LL_SYSCFG_TIMBREAK_PVD_TO_ALL	PVD 中断与 TIM1/TIM16/TIM17 的刹车输入连接
LL_SYSCFG_TIMBREAK_COMP1_TO_TIM1	COMP1 输出作为 TIM1 刹车输入
LL_SYSCFG_TIMBREAK_COMP2_TO_TIM1	COMP2 输出作为 TIM1 刹车输入
LL_SYSCFG_TIMBREAK_COMP1_TO_TIM16	COMP1 输出作为 TIM16 刹车输入
LL_SYSCFG_TIMBREAK_COMP2_TO_TIM16	COMP2 输出作为 TIM16 刹车输入
LL_SYSCFG_TIMBREAK_COMP1_TO_TIM17	COMP1 输出作为 TIM17 刹车输入
LL_SYSCFG_TIMBREAK_COMP2_TO_TIM17	COMP2 输出作为 TIM17 刹车输入

44.1.9 函数 LL_SYSCFG_SetTIM1ETRSource

描述了函数 LL_SYSCFG_SetTIM1ETRSource

表44-17 函数 LL_SYSCFG_SetTIM1ETRSource

函数名	LL_SYSCFG_SetTIM1ETRSource
函数原形	__STATIC_INLINE void LL_SYSCFG_SetTIM1ETRSource(uint32_t source)
功能描述	设置 TIMER1 ETR 输入源
输入参数	source：TIMER1 ETR 输入源
输出参数	无
返回值	无
先决条件	无

source 可选参数:

表44-18 source 可选参数

参数	描述
LL_SYSCFG_ETR_SRC_TIM1_GPIO	TIMER1 ETR 来源于 GPIO
LL_SYSCFG_ETR_SRC_TIM1_COMP1	TIMER1 ETR 来源于 COMP1
LL_SYSCFG_ETR_SRC_TIM1_COMP2	TIMER1 ETR 来源于 COMP2
LL_SYSCFG_ETR_SRC_TIM1_ADC	TIMER1 ETR 来源于 ADC

44.1.10 函数 LL_SYSCFG_GetTIM1ETRSource

描述了函数 LL_SYSCFG_GetTIM1ETRSource

表44-19 函数 LL_SYSCFG_GetTIM1ETRSource

函数名	LL_SYSCFG_GetTIM1ETRSource
函数原形	__STATIC_INLINE uint32_t LL_SYSCFG_GetTIM1ETRSource(void)
功能描述	获取 TIMER1 ETR 输入源
输入参数	无
输出参数	无
返回值	TIMER1 ETR 输入源
先决条件	无

44.1.11 函数 LL_SYSCFG_SetDMARemap_CH1

描述了函数 LL_SYSCFG_SetDMARemap_CH1

表44-20 函数 LL_SYSCFG_SetDMARemap_CH1

函数名	LL_SYSCFG_SetDMARemap_CH1
函数原形	__STATIC_INLINE void LL_SYSCFG_SetDMARemap_CH1(uint32_t Requset)
功能描述	设置 DMA 通道 1 的请求映像
输入参数 1	Requset：请求映像
输出参数	无
返回值	无
先决条件	无

Requset 可选参数：

表44-21 Requset 可选参数

参数	描述
LL_SYSCFG_DMA_MAP_ADC	请求映像为 ADC
LL_SYSCFG_DMA_MAP_SPI1_TX	请求映像为 SPI1_TX
LL_SYSCFG_DMA_MAP_SPI1_RX	请求映像为 SPI1_RX
LL_SYSCFG_DMA_MAP_SPI2_TX	请求映像为 SPI2_TX
LL_SYSCFG_DMA_MAP_SPI2_RX	请求映像为 SPI2_RX

LL_SYSCFG_DMA_MAP_USART1_TX	请求映像为 USART1_TX
LL_SYSCFG_DMA_MAP_USART1_RX	请求映像为 USART1_RX
LL_SYSCFG_DMA_MAP_USART2_TX	请求映像为 USART2_TX
LL_SYSCFG_DMA_MAP_USART2_RX	请求映像为 USART2_RX
LL_SYSCFG_DMA_MAP_I2C_TX	请求映像为 I2C_TX
LL_SYSCFG_DMA_MAP_I2C_RX	请求映像为 I2C_RX
LL_SYSCFG_DMA_MAP_TIM1_CH1	请求映像为 TIM1_CH1
LL_SYSCFG_DMA_MAP_TIM1_CH2	请求映像为 TIM1_CH2
LL_SYSCFG_DMA_MAP_TIM1_CH3	请求映像为 TIM1_CH3
LL_SYSCFG_DMA_MAP_TIM1_CH4	请求映像为 TIM1_CH4
LL_SYSCFG_DMA_MAP_TIM1_COM	请求映像为 TIM1_COM
LL_SYSCFG_DMA_MAP_TIM1_UP	请求映像为 TIM1_UP
LL_SYSCFG_DMA_MAP_TIM1_TRIG	请求映像为 TIM1_TRIG
LL_SYSCFG_DMA_MAP_TIM3_CH1	请求映像为 TIM3_CH1
LL_SYSCFG_DMA_MAP_TIM3_CH3	请求映像为 TIM3_CH3
LL_SYSCFG_DMA_MAP_TIM3_CH4	请求映像为 TIM3_CH4
LL_SYSCFG_DMA_MAP_TIM3_TRG	请求映像为 TIM3_TRG
LL_SYSCFG_DMA_MAP_TIM3_UP	请求映像为 TIM3_UP
LL_SYSCFG_DMA_MAP_TIM16_CH1	请求映像为 TIM16_CH1
LL_SYSCFG_DMA_MAP_TIM16_UP	请求映像为 TIM16_UP
LL_SYSCFG_DMA_MAP_TIM17_CH1	请求映像为 TIM17_CH1
LL_SYSCFG_DMA_MAP_TIM17_UP	请求映像为 TIM17_UP

44.1.12 函数 LL_SYSCFG_GetDMARemap_CH1

描述了函数 LL_SYSCFG_GetDMARemap_CH1

表44-22 函数 LL_SYSCFG_GetDMARemap_CH1

函数名	LL_SYSCFG_GetDMARemap_CH1
函数原形	__STATIC_INLINE uint32_t LL_SYSCFG_GetDMARemap_CH1(void)
功能描述	获取 DMA 通道 1 的请求映像
输入参数	无
输出参数	无
返回值	DMA 通道 1 的请求映像
先决条件	无

44.1.13 函数 LL_SYSCFG_SetDMAResponseSpeed_CH1

描述了函数 LL_SYSCFG_SetDMAResponseSpeed_CH1

表44-23 函数 LL_SYSCFG_SetDMAResponseSpeed_CH1

函数名	LL_SYSCFG_SetDMAResponseSpeed_CH1
函数原形	__STATIC_INLINE void LL_SYSCFG_SetDMAResponseSpeed_CH1(uint32_t ResponseSpeed)
功能描述	设置 DMA 通道 1 的响应速度
输入参数	ResponseSpeed：响应速度
输出参数	无
返回值	无
先决条件	无

ResponseSpeed 可选参数:

表44-24 ResponseSpeed 可选参数

参数	描述
LL_SYSCFG_DMA_ACKLVL_NORM	普通速度响应
LL_SYSCFG_DMA_ACKLVL_FAST	快速响应

44.1.14 函数 LL_SYSCFG_GetDMAResponseSpeed_CH1

描述了函数 LL_SYSCFG_GetDMAResponseSpeed_CH1

表44-25 函数 LL_SYSCFG_GetDMAResponseSpeed_CH1

函数名	LL_SYSCFG_GetDMAResponseSpeed_CH1
函数原形	__STATIC_INLINE uint32_t LL_SYSCFG_GetDMAResponseSpeed_CH1 (void)
功能描述	获取 DMA 通道 1 的响应速度
输入参数	无
输出参数	无
返回值	DMA 通道 1 的响应速度
先决条件	无

44.1.15 函数 LL_SYSCFG_SetDMARemap_CH2

描述了函数 LL_SYSCFG_SetDMARemap_CH2

表44-26 函数 LL_SYSCFG_SetDMARemap_CH2

函数名	LL_SYSCFG_SetDMARemap_CH2
函数原形	__STATIC_INLINE void LL_SYSCFG_SetDMARemap_CH2(uint32_t Requset)
功能描述	设置 DMA 通道 2 的请求映像
输入参数	Requset：请求映像
输出参数	无
返回值	无
先决条件	无

Requset 可选参数:

表44-27 Requset 可选参数

参数	描述
LL_SYSCFG_DMA_MAP_ADC	请求映像为 ADC
LL_SYSCFG_DMA_MAP_SPI1_TX	请求映像为 SPI1_TX
LL_SYSCFG_DMA_MAP_SPI1_RX	请求映像为 SPI1_RX
LL_SYSCFG_DMA_MAP_SPI2_TX	请求映像为 SPI2_TX
LL_SYSCFG_DMA_MAP_SPI2_RX	请求映像为 SPI2_RX
LL_SYSCFG_DMA_MAP_USART1_TX	请求映像为 USART1_TX
LL_SYSCFG_DMA_MAP_USART1_RX	请求映像为 USART1_RX
LL_SYSCFG_DMA_MAP_USART2_TX	请求映像为 USART2_TX
LL_SYSCFG_DMA_MAP_USART2_RX	请求映像为 USART2_RX
LL_SYSCFG_DMA_MAP_I2C_TX	请求映像为 I2C_TX
LL_SYSCFG_DMA_MAP_I2C_RX	请求映像为 I2C_RX
LL_SYSCFG_DMA_MAP_TIM1_CH1	请求映像为 TIM1_CH1
LL_SYSCFG_DMA_MAP_TIM1_CH2	请求映像为 TIM1_CH2
LL_SYSCFG_DMA_MAP_TIM1_CH3	请求映像为 TIM1_CH3
LL_SYSCFG_DMA_MAP_TIM1_CH4	请求映像为 TIM1_CH4
LL_SYSCFG_DMA_MAP_TIM1_COM	请求映像为 TIM1_COM
LL_SYSCFG_DMA_MAP_TIM1_UP	请求映像为 TIM1_UP
LL_SYSCFG_DMA_MAP_TIM1_TRIG	请求映像为 TIM1_TRIG
LL_SYSCFG_DMA_MAP_TIM3_CH1	请求映像为 TIM3_CH1
LL_SYSCFG_DMA_MAP_TIM3_CH3	请求映像为 TIM3_CH3
LL_SYSCFG_DMA_MAP_TIM3_CH4	请求映像为 TIM3_CH4
LL_SYSCFG_DMA_MAP_TIM3_TRG	请求映像为 TIM3_TRG
LL_SYSCFG_DMA_MAP_TIM3_UP	请求映像为 TIM3_UP
LL_SYSCFG_DMA_MAP_TIM16_CH1	请求映像为 TIM16_CH1
LL_SYSCFG_DMA_MAP_TIM16_UP	请求映像为 TIM16_UP
LL_SYSCFG_DMA_MAP_TIM17_CH1	请求映像为 TIM17_CH1
LL_SYSCFG_DMA_MAP_TIM17_UP	请求映像为 TIM17_UP

44.1.16 函数 LL_SYSCFG_GetDMARemap_CH2

描述了函数 LL_SYSCFG_GetDMARemap_CH2

表44-28 函数 LL_SYSCFG_GetDMARemap_CH2

函数名	LL_SYSCFG_GetDMARemap_CH2
函数原形	__STATIC_INLINE uint32_t LL_SYSCFG_GetDMARemap_CH2(void)

功能描述	获取 DMA 通道 2 的请求映像
输入参数	无
输出参数	无
返回值	DMA 通道 2 的请求映像
先决条件	无

44.1.17 函数 LL_SYSCFG_SetDMAResponseSpeed_CH2

描述了函数 LL_SYSCFG_SetDMAResponseSpeed_CH2

表44-29 函数 LL_SYSCFG_SetDMAResponseSpeed_CH2

函数名	LL_SYSCFG_SetDMAResponseSpeed_CH2
函数原形	__STATIC_INLINE void LL_SYSCFG_SetDMAResponseSpeed_CH2(uint32_t ResponseSpeed)
功能描述	设置 DMA 通道 2 的响应速度
输入参数	ResponseSpeed：响应速度
输出参数	无
返回值	无
先决条件	无

ResponseSpeed 可选参数:

表44-30 ResponseSpeed 可选参数

参数	描述
LL_SYSCFG_DMA_ACKLVL_NORM	普通速度响应
LL_SYSCFG_DMA_ACKLVL_FAST	快速响应

44.1.18 函数 LL_SYSCFG_GetDMAResponseSpeed_CH2

描述了函数 LL_SYSCFG_GetDMAResponseSpeed_CH2

表44-31 函数 LL_SYSCFG_GetDMAResponseSpeed_CH2

函数名	LL_SYSCFG_GetDMAResponseSpeed_CH2
函数原形	__STATIC_INLINE uint32_t LL_SYSCFG_GetDMAResponseSpeed_CH2(void)
功能描述	获取 DMA 通道 2 的响应速度
输入参数	无
输出参数	无
返回值	DMA 通道 2 的响应速度
先决条件	无

44.1.19 函数 LL_SYSCFG_SetDMARemap_CH3

描述了函数 LL_SYSCFG_SetDMARemap_CH3

表44-32 函数 LL_SYSCFG_SetDMARemap_CH3

函数名	LL_SYSCFG_SetDMARemap_CH3
函数原形	__STATIC_INLINE void LL_SYSCFG_SetDMARemap_CH3(uint32_t Requset)
功能描述	设置 DMA 通道 3 的请求映像
输入参数	Requset：请求映像
输出参数	无
返回值	无
先决条件	无

Requset 可选参数：

表44-33 Requset 可选参数

参数	描述
LL_SYSCFG_DMA_MAP_ADC	请求映像为 ADC
LL_SYSCFG_DMA_MAP_SPI1_TX	请求映像为 SPI1_TX
LL_SYSCFG_DMA_MAP_SPI1_RX	请求映像为 SPI1_RX
LL_SYSCFG_DMA_MAP_SPI2_TX	请求映像为 SPI2_TX
LL_SYSCFG_DMA_MAP_SPI2_RX	请求映像为 SPI2_RX
LL_SYSCFG_DMA_MAP_USART1_TX	请求映像为 USART1_TX
LL_SYSCFG_DMA_MAP_USART1_RX	请求映像为 USART1_RX
LL_SYSCFG_DMA_MAP_USART2_TX	请求映像为 USART2_TX
LL_SYSCFG_DMA_MAP_USART2_RX	请求映像为 USART2_RX
LL_SYSCFG_DMA_MAP_I2C_TX	请求映像为 I2C_TX
LL_SYSCFG_DMA_MAP_I2C_RX	请求映像为 I2C_RX
LL_SYSCFG_DMA_MAP_TIM1_CH1	请求映像为 TIM1_CH1
LL_SYSCFG_DMA_MAP_TIM1_CH2	请求映像为 TIM1_CH2
LL_SYSCFG_DMA_MAP_TIM1_CH3	请求映像为 TIM1_CH3
LL_SYSCFG_DMA_MAP_TIM1_CH4	请求映像为 TIM1_CH4
LL_SYSCFG_DMA_MAP_TIM1_COM	请求映像为 TIM1_COM
LL_SYSCFG_DMA_MAP_TIM1_UP	请求映像为 TIM1_UP
LL_SYSCFG_DMA_MAP_TIM1_TRIG	请求映像为 TIM1_TRIG
LL_SYSCFG_DMA_MAP_TIM3_CH1	请求映像为 TIM3_CH1
LL_SYSCFG_DMA_MAP_TIM3_CH3	请求映像为 TIM3_CH3
LL_SYSCFG_DMA_MAP_TIM3_CH4	请求映像为 TIM3_CH4
LL_SYSCFG_DMA_MAP_TIM3_TRG	请求映像为 TIM3_TRG
LL_SYSCFG_DMA_MAP_TIM3_UP	请求映像为 TIM3_UP
LL_SYSCFG_DMA_MAP_TIM16_CH1	请求映像为 TIM16_CH1
LL_SYSCFG_DMA_MAP_TIM16_UP	请求映像为 TIM16_UP

LL_SYSCFG_DMA_MAP_TIM17_CH1	请求映像为 TIM17_CH1
LL_SYSCFG_DMA_MAP_TIM17_UP	请求映像为 TIM17_UP

44.1.20 函数 LL_SYSCFG_GetDMARemap_CH3

描述了函数 LL_SYSCFG_GetDMARemap_CH3

表44-34 函数 LL_SYSCFG_GetDMARemap_CH3

函数名	LL_SYSCFG_GetDMARemap_CH3
函数原形	__STATIC_INLINE uint32_t LL_SYSCFG_GetDMARemap_CH3(void)
功能描述	获取 DMA 通道 3 的请求映像
输入参数	无
输出参数	无
返回值	DMA 通道 3 的请求映像
先决条件	无

44.1.21 函数 LL_SYSCFG_SetDMAResponseSpeed_CH3

描述了函数 LL_SYSCFG_SetDMAResponseSpeed_CH3

表44-35 函数 LL_SYSCFG_SetDMAResponseSpeed_CH3

函数名	LL_SYSCFG_SetDMAResponseSpeed_CH3
函数原形	__STATIC_INLINE void LL_SYSCFG_SetDMAResponseSpeed_CH3(uint32_t ResponseSpeed)
功能描述	设置 DMA 通道 3 的响应速度
输入参数	ResponseSpeed：响应速度
输出参数	无
返回值	无
先决条件	无

ResponseSpeed 可选参数:

表44-36 ResponseSpeed 可选参数

参数	描述
LL_SYSCFG_DMA_ACKLVL_NORM	普通速度响应
LL_SYSCFG_DMA_ACKLVL_FAST	快速响应

44.1.22 函数 LL_SYSCFG_GetDMAResponseSpeed_CH3

描述了函数 LL_SYSCFG_GetDMAResponseSpeed_CH3

表44-37 函数 LL_SYSCFG_GetDMAResponseSpeed_CH3

函数名	LL_SYSCFG_GetDMAResponseSpeed_CH3
函数原形	__STATIC_INLINE uint32_t LL_SYSCFG_GetDMAResponseSpeed_CH3(void)
功能描述	获取 DMA 通道 3 的响应速度

输入参数	无
输出参数	无
返回值	DMA 通道 3 的响应速度
先决条件	无

44.1.23 函数 LL_FLASH_SetLatency

描述了函数 LL_FLASH_SetLatency

表44-38 函数 LL_FLASH_SetLatency

函数名	LL_FLASH_SetLatency
函数原形	__STATIC_INLINE void LL_FLASH_SetLatency(uint32_t Latency)
功能描述	设置 flash 延迟
输入参数	Latency：flash 延迟
输出参数	无
返回值	无
先决条件	无

Latency 可选参数：

表44-39 Latency 可选参数

参数	描述
LL_FLASH_LATENCY_0	FLASH 零延迟周期
LL_FLASH_LATENCY_1	FLASH 一个延迟周期

44.1.24 函数 LL_FLASH_GetLatency

描述了函数 LL_FLASH_GetLatency

表44-40 函数 LL_FLASH_GetLatency

函数名	LL_FLASH_GetLatency
函数原形	__STATIC_INLINE uint32_t LL_FLASH_GetLatency(void)
功能描述	获取 flash 延迟
输入参数	无
输出参数	无
返回值	flash 延迟
先决条件	无

44.1.25 函数 LL_DBGMCU_GetDeviceID

描述了函数 LL_DBGMCU_GetDeviceID

表44-41 函数 LL_DBGMCU_GetDeviceID

函数名	LL_DBGMCU_GetDeviceID

函数原形	__STATIC_INLINE uint32_t LL_DBGMCU_GetDeviceID (void)
功能描述	获取设备 ID
输入参数	无
输出参数	无
返回值	设备 ID
先决条件	无

44.1.26 函数 LL_DBGMCU_GetRevisionID

描述了函数 LL_DBGMCU_GetRevisionID

表44-42 函数 LL_DBGMCU_GetRevisionID

函数名	LL_DBGMCU_GetRevisionID
函数原形	__STATIC_INLINE uint32_t LL_DBGMCU_GetRevisionID (void)
功能描述	获取版本 ID
输入参数	无
输出参数	无
返回值	设备 ID
先决条件	无

44.1.27 函数 LL_DBGMCU_EnableDBGStopMode

描述了函数 LL_DBGMCU_EnableDBGStopMode

表44-43 函数 LL_DBGMCU_EnableDBGStopMode

函数名	LL_DBGMCU_EnableDBGStopMode
函数原形	__STATIC_INLINE void LL_DBGMCU_EnableDBGStopMode(void)
功能描述	在 STOP 模式下，使能调试模块
输入参数	无
输出参数	无
返回值	无
先决条件	无

44.1.28 函数 LL_DBGMCU_DisableDBGStopMode

描述了函数 LL_DBGMCU_DisableDBGStopMode

表44-44 函数 LL_DBGMCU_DisableDBGStopMode

函数名	LL_DBGMCU_DisableDBGStopMode
函数原形	__STATIC_INLINE void LL_DBGMCU_DisableDBGStopMode (void)
功能描述	在 STOP 模式下，禁用调试模块
输入参数	无

输出参数	无
返回值	无
先决条件	无

44.1.29 函数 LL_DBGMCU_IsEnabledDBGStopMode

描述了函数 LL_DBGMCU_IsEnabledDBGStopMode

表44-45 函数 LL_DBGMCU_IsEnabledDBGStopMode

函数名	LL_DBGMCU_IsEnabledDBGStopMode
函数原形	__STATIC_INLINE uint32_t LL_DBGMCU_IsEnabledDBGStopMode(void)
功能描述	在 STOP 模式下，检查是否使能调试模块
输入参数	无
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

44.1.30 函数 LL_DBGMCU_APB1_GRP1_FreezePeriph

描述了函数 LL_DBGMCU_APB1_GRP1_FreezePeriph

表44-46 函数 LL_DBGMCU_APB1_GRP1_FreezePeriph

函数名	LL_DBGMCU_APB1_GRP1_FreezePeriph
函数原形	__STATIC_INLINE void LL_DBGMCU_APB1_GRP1_FreezePeriph(uint32_t Periphs)
功能描述	冻结 APB1 外设（group1 外设）
输入参数	Periphs：APB1 外设
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表44-47 Periphs 可选参数

参数	描述
LL_DBGMCU_APB1_GRP1_TIM3_STOP	外设 TIM3
LL_DBGMCU_APB1_GRP1_RTC_STOP	外设 RTC
LL_DBGMCU_APB1_GRP1_WWDG_STOP	外设WWDG
LL_DBGMCU_APB1_GRP1_IWDG_STOP	外设 IWDG
LL_DBGMCU_APB1_GRP1_LPTIM1_STOP	外设 LPTIM1

44.1.31 函数 LL_DBGMCU_APB1_GRP1_UnFreezePeriph

描述了函数 LL_DBGMCU_APB1_GRP1_UnFreezePeriph

表44-48 函数 LL_DBGMCU_APB1_GRP1_UnFreezePeriph

函数名	LL_DBGMCU_APB1_GRP1_UnFreezePeriph
函数原形	__STATIC_INLINE void LL_DBGMCU_APB1_GRP1_UnFreezePeriph (uint32_t Periphs)
功能描述	解冻 APB1 外设（group1 外设）
输入参数	Periphs：APB1 外设
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表44-49 Periphs 可选参数

参数	描述
LL_DBGMCU_APB1_GRP1_TIM3_STOP	外设 TIM3
LL_DBGMCU_APB1_GRP1_RTC_STOP	外设 RTC
LL_DBGMCU_APB1_GRP1_WWDG_STOP	外设WWDG
LL_DBGMCU_APB1_GRP1_IWDG_STOP	外设 IWDG
LL_DBGMCU_APB1_GRP1_LPTIM1_STOP	外设 LPTIM1

44.1.32 函数 LL_DBGMCU_APB1_GRP1_IsFreezePeriph

描述了函数 LL_DBGMCU_APB1_GRP1_IsFreezePeriph

表44-50 函数 LL_DBGMCU_APB1_GRP1_IsFreezePeriph

函数名	LL_DBGMCU_APB1_GRP1_IsFreezePeriph
函数原形	__STATIC_INLINE uint32_t LL_DBGMCU_APB1_GRP1_IsFreezePeriph(uint32_t Periphs)
功能描述	检查是否冻结 APB1 外设（group1 外设）
输入参数	Periphs：APB1 外设
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

Periphs 可选参数：

表44-51 Periphs 可选参数

参数	描述
LL_DBGMCU_APB1_GRP1_TIM3_STOP	外设 TIM3
LL_DBGMCU_APB1_GRP1_RTC_STOP	外设 RTC
LL_DBGMCU_APB1_GRP1_WWDG_STOP	外设WWDG
LL_DBGMCU_APB1_GRP1_IWDG_STOP	外设 IWDG

LL_DBGMCU_APB1_GRP1_LPTIM1_STOP	外设 LPTIM1

44.1.33 函数 LL_DBGMCU_APB1_GRP2_FreezePeriph

描述了函数 LL_DBGMCU_APB1_GRP2_FreezePeriph

表44-52 函数 LL_DBGMCU_APB1_GRP2_FreezePeriph

函数名	LL_DBGMCU_APB1_GRP2_FreezePeriph
函数原形	__STATIC_INLINE void LL_DBGMCU_APB1_GRP2_FreezePeriph(uint32_t Periphs)
功能描述	冻结 APB1 外设（group2 外设）
输入参数	Periphs：APB1 外设
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表44-53 Periphs 可选参数

参数	描述
LL_DBGMCU_APB1_GRP2_TIM1_STOP	外设 TIM1
LL_DBGMCU_APB1_GRP2_TIM14_STOP	外设 TIM14
LL_DBGMCU_APB1_GRP2_TIM16_STOP	外设 TIM16
LL_DBGMCU_APB1_GRP2_TIM17_STOP	外设 TIM17

44.1.34 函数 LL_DBGMCU_APB1_GRP2_UnFreezePeriph

描述了函数 LL_DBGMCU_APB1_GRP2_UnFreezePeriph

表44-54 函数 LL_DBGMCU_APB1_GRP2_UnFreezePeriph

函数名	LL_DBGMCU_APB1_GRP2_UnFreezePeriph
函数原形	__STATIC_INLINE void LL_DBGMCU_APB1_GRP2_UnFreezePeriph (uint32_t Periphs)
功能描述	解冻 APB1 外设（group2 外设）
输入参数	Periphs：APB1 外设
输出参数	无
返回值	无
先决条件	无

Periphs 可选参数：

表44-55 Periphs 可选参数

参数	描述
LL_DBGMCU_APB1_GRP2_TIM1_STOP	外设 TIM1

LL_DBGMCU_APB1_GRP2_TIM14_STOP	外设 TIM14
LL_DBGMCU_APB1_GRP2_TIM16_STOP	外设 TIM16
LL_DBGMCU_APB1_GRP2_TIM17_STOP	外设 TIM17

44.1.35 函数 LL_DBGMCU_APB1_GRP2_IsFreezePeriph

描述了函数 LL_DBGMCU_APB1_GRP2_IsFreezePeriph

表44-56 函数 LL_DBGMCU_APB1_GRP2_IsFreezePeriph

函数名	LL_DBGMCU_APB1_GRP2_IsFreezePeriph
函数原形	__STATIC_INLINE uint32_t LL_DBGMCU_APB1_GRP2_IsFreezePeriph(uint32_t Periphs)
功能描述	检查是否冻结 APB1 外设（group2 外设）
输入参数	Periphs：APB1 外设
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

Periphs 可选参数：

表44-57 Periphs 可选参数

参数	描述
LL_DBGMCU_APB1_GRP2_TIM1_STOP	外设 TIM1
LL_DBGMCU_APB1_GRP2_TIM14_STOP	外设 TIM14
LL_DBGMCU_APB1_GRP2_TIM16_STOP	外设 TIM16
LL_DBGMCU_APB1_GRP2_TIM17_STOP	外设 TIM17

45 LL 定时器通用驱动程序（TIM）

TIM 包含一个 16 位自动重载计数器，该计数器由可编程预分频器驱动。它可以被用作各种场景，包括：输入信号（输入捕获）的脉冲长度测量，或者产生输出波形（输出比较、输出 PWM、带死区插入的互补 PWM）。

使用定时器预分频器和 RCC 时钟控制器预分频器，可将脉冲宽度和波形周期从几微秒调制到几毫秒。

高级控制定时器(TIM1)和通用定时器(TIMy)彼此完全独立，不共享任何资源。

45.1 TIM 固件驱动寄存器结构

45.1.1 LL_TIM_InitTypeDef

LL_TIM_InitTypeDef，定义于文件"air001xx_ll_tim.h"如下：

```
typedef struct
{
uint16_t Prescaler;
uint32_t CounterMode;
uint32_t Autoreload;
uint32_t ClockDivision;
uint8_t RepetitionCounter;
} LL_TIM_InitTypeDef;
```

字段说明：

表45-1 LL_TIM_InitTypeDef 字段说明

字段	描述
Prescaler	预分频值（0x0000~0xFFFF）
CounterMode	计数模式
Autoreload	自动重装载值（0x0000~0xFFFF）
ClockDivision	死区发生器和数字滤波器频率
RepetitionCounter	重复计数值（0x00~0xFF）

参数说明：

CounterMode 可选参数：

表45-2 CounterMode 可选参数

参数	描述
LL_TIM_COUNTERMODE_UP	向上计数
LL_TIM_COUNTERMODE_DOWN	向下计数

LL_TIM_COUNTERMODE_CENTER_UP	中央对齐模式，仅向上计数触发比较中断标志
LL_TIM_COUNTERMODE_CENTER_DOWN	中央对齐模式，仅向下计数触发比较中断标志
LL_TIM_COUNTERMODE_CENTER_UP_DOWN	中央对齐模式，向上向下计数都能触发比较中断标志

ClockDivision 可选参数：

表45-3 ClockDivision 可选参数

参数	描述
LL_TIM_CLOCKDIVISION_DIV1	定时器时钟不分频
LL_TIM_CLOCKDIVISION_DIV2	定时器时钟 2 分频
LL_TIM_CLOCKDIVISION_DIV4	定时器时钟 4 分频

45.1.2 LL_TIM_OC_InitTypeDef

LL_TIM_OC_InitTypeDef，定义于文件"air001xx_ll_tim.h"如下：

```
typedef struct
{
uint32_t OCMode;
uint32_t OCState;
uint32_t OCNState;
uint32_t CompareValue;
uint32_t OCPolarity;
uint32_t OCNPolarity;
uint32_t OCIdleState;
uint32_t OCNIdleState;
} LL_TIM_OC_InitTypeDef;
```

字段说明：

表45-4 LL_TIM_OC_InitTypeDef 字段说明

字段	描述
OCMode	输出模式
OCState	输出状态
OCNState	互补输出状态
CompareValue	比较值（0x0000~0xFFFF）
OCPolarity	输出有效极性
OCNPolarity	互补输出有效极性
OCIdleState	输出通道空闲状态极性
OCNIdleState	互补输出通道空闲状态极性

参数说明:

OCMode 可选参数:

表45-5 OCMode 可选参数

参数	描述
LL_TIM_OCMODE_FROZEN	冻结，比较值不起作用
LL_TIM_OCMODE_ACTIVE	比较值匹配时配置通道为有效电平
LL_TIM_OCMODE_INACTIVE	比较值匹配时配置通道为无效电平
LL_TIM_OCMODE_TOGGLE	比较值匹配时翻转当前电平
LL_TIM_OCMODE_FORCED_INACTIVE	强制输出无效电平
LL_TIM_OCMODE_FORCED_ACTIVE	强制输出有效电平
LL_TIM_OCMODE_PWM1	PWM 模式 1
LL_TIM_OCMODE_PWM2	PWM 模式 2

OCState 可选参数:

表45-6 OCState 可选参数

参数	描述
LL_TIM_OCSTATE_DISABLE	关闭信号输出
LL_TIM_OCSTATE_ENABLE	开启信号输出

OCNState 可选参数:

表45-7 OCNState 可选参数

参数	描述
LL_TIM_OCSTATE_DISABLE	关闭信号输出
LL_TIM_OCSTATE_ENABLE	开启信号输出

OCPolarity 可选参数:

表45-8 OCPolarity 可选参数

参数	描述
LL_TIM_OCPOLARITY_HIGH	高电平为有效极性
LL_TIM_OCPOLARITY_LOW	低电平为有效极性

OCNPolarity 可选参数:

表45-9 OCNPolarity 可选参数

参数	描述
LL_TIM_OCPOLARITY_HIGH	高电平为有效极性
LL_TIM_OCPOLARITY_LOW	低电平为有效极性

OCIdleState 可选参数：

表45-10 OCIdleState 可选参数

参数	描述
LL_TIM_OCIDLESTATE_LOW	死区后保持低电平
LL_TIM_OCIDLESTATE_HIGH	死区后保持高电平

OCNIdleState 可选参数：

表45-11 OCNIdleState 可选参数

参数	描述
LL_TIM_OCIDLESTATE_LOW	死区后保持低电平
LL_TIM_OCIDLESTATE_HIGH	死区后保持高电平

45.1.3 LL_TIM_IC_InitTypeDef

LL_TIM_IC_InitTypeDef，定义于文件"air001xx_ll_tim.h"如下：

```
typedef struct
{
uint32_t ICPolarity;
uint32_t ICActiveInput;
uint32_t ICPrescaler;
uint32_t ICFilter;
} LL_TIM_IC_InitTypeDef;
```

字段说明：

字段	描述
ICPolarity	输入捕获极性
ICActiveInput	输入有效映射
ICPrescaler	输入信号预分频值
ICFilter	输入信号滤波值

参数说明：

ICPolarity 可选参数：

表45-12 ICPolarity 可选参数

参数	描述
LL_TIM_IC_POLARITY_RISING	上升沿捕获
LL_TIM_IC_POLARITY_FALLING	下降沿捕获
LL_TIM_IC_POLARITY_BOTHEDGE	双边沿捕获

ICActiveInput 可选参数:

表45-13 ICActiveInput 可选参数

参数	描述
LL_TIM_ACTIVEINPUT_DIRECTTI	输入模式，ICx 映射在 Tix 上
LL_TIM_ACTIVEINPUT_INDIRECTTI	输入模式，ICx 映射在 Tiy 上
LL_TIM_ACTIVEINPUT_TRC	输入模式，ICx 映射在 TRC 上

注意：当 *x=1* 时 *y=2*，当 *x=2* 时 *y=1*，当 *x=3* 时，*y=4*，当 *x=4* 时 *y=3*。

ICPrescaler 可选参数:

表45-14 ICPrescaler 可选参数

参数	描述
LL_TIM_ICPSC_DIV1	输入信号不分频
LL_TIM_ICPSC_DIV2	输入信号 2 分频
LL_TIM_ICPSC_DIV4	输入信号 4 分频
LL_TIM_ICPSC_DIV8	输入信号 8 分频

ICFilter 可选参数:

表45-15 ICFilter 可选参数

参数	描述
LL_TIM_IC_FILTER_FDIV1	无滤波
LL_TIM_IC_FILTER_FDIV1_N2	采样频率=定时器时钟频率，采样 2 次
LL_TIM_IC_FILTER_FDIV1_N4	采样频率=定时器时钟频率，采样 4 次
LL_TIM_IC_FILTER_FDIV1_N8	采样频率=定时器时钟频率，采样 8 次
LL_TIM_IC_FILTER_FDIV2_N6	采样频率=死区发生器时钟频率 2 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV2_N8	采样频率=死区发生器时钟频率 2 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV4_N6	采样频率=死区发生器时钟频率 4 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV4_N8	采样频率=死区发生器时钟频率 4 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV8_N6	采样频率=死区发生器时钟频率 8 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV8_N8	采样频率=死区发生器时钟频率 8 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV16_N5	采样频率=死区发生器时钟频率 16 分频，采样 5 次
LL_TIM_IC_FILTER_FDIV16_N6	采样频率=死区发生器时钟频率 16 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV16_N8	采样频率=死区发生器时钟频率 16 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV32_N5	采样频率=死区发生器时钟频率 32 分频，采样 5 次
LL_TIM_IC_FILTER_FDIV32_N6	采样频率=死区发生器时钟频率 32 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV32_N8	采样频率=死区发生器时钟频率 32 分频，采样 8 次

45.1.4 LL_TIM_ENCODER_InitTypeDef

LL_TIM_ENCODER_InitTypeDef，定义于文件"air001xx_ll_tim.h"如下：

```
typedef struct
{
uint32_t EncoderMode;
uint32_t IC1Polarity;
uint32_t IC1ActiveInput;
uint32_t IC1Prescaler;
uint32_t IC1Filter;
uint32_t IC2Polarity;
uint32_t IC2ActiveInput;
uint32_t IC2Prescaler;
uint32_t IC2Filter;
} LL_TIM_ENCODER_InitTypeDef;
```

字段说明：

表45-16 LL_TIM_ENCODER_InitTypeDef 字段说明

字段	描述
EncoderMode	编码器接口模式
IC1Polarity	输入通道 1 有效边沿
IC1ActiveInput	输入通道 1 有效映射
IC1Prescaler	输入通道 1 预分频值
IC1Filter	输入通道 1 数字滤波值
IC2Polarity	输入通道 2 有效边沿
IC2ActiveInput	输入通道 2 有效映射
IC2Prescaler	输入通道 2 预分频值
IC2Filter	输入通道 2 数字滤波值

参数说明：

EncoderMode 可选参数：

表45-17 EncoderMode 可选参数

参数	描述
LL_TIM_ENCODERMODE_X2_TI1	根据 TI1FP2 的电平，计数器在 TI2FP1 的边沿向上/下计数
LL_TIM_ENCODERMODE_X2_TI2	根据 TI1FP1 的电平，计数器在 TI2FP2 的边沿向上/下计数
LL_TIM_ENCODERMODE_X4_TI12	TI1FP1 和 TI1FP2 互相根据对方的电平同时向上/下计数

IC1Polarity 可选参数：

表45-18 IC1Polarity 可选参数

参数	描述
LL_TIM_IC_POLARITY_RISING	上升沿有效
LL_TIM_IC_POLARITY_FALLING	下降沿有效
LL_TIM_IC_POLARITY_BOTHEDGE	双边沿有效

IC1ActiveInput 可选参数：

表45-19 IC1ActiveInput 可选参数

参数	描述
LL_TIM_ACTIVEINPUT_DIRECTTI	输入模式，ICx 映射在 Tix 上
LL_TIM_ACTIVEINPUT_INDIRECTTI	输入模式，ICx 映射在 Tiy 上
LL_TIM_ACTIVEINPUT_TRC	输入模式，ICx 映射在 TRC 上

注意：当 *x=1* 时 *y=2*，当 *x=2* 时 *y=1*，当 *x=3* 时，*y=4*，当 *x=4* 时 *y=3*。

IC1Prescaler 可选参数：

表45-20 IC1Prescaler 可选参数

参数	描述
LL_TIM_ICPSC_DIV1	输入信号不分频
LL_TIM_ICPSC_DIV2	输入信号 2 分频
LL_TIM_ICPSC_DIV4	输入信号 4 分频
LL_TIM_ICPSC_DIV8	输入信号 8 分频

IC1Filter 可选参数：

表45-21 IC1Filter 可选参数

参数	描述
LL_TIM_IC_FILTER_FDIV1	无滤波
LL_TIM_IC_FILTER_FDIV1_N2	采样频率=定时器时钟频率，采样 2 次
LL_TIM_IC_FILTER_FDIV1_N4	采样频率=定时器时钟频率，采样 4 次
LL_TIM_IC_FILTER_FDIV1_N8	采样频率=定时器时钟频率，采样 8 次
LL_TIM_IC_FILTER_FDIV2_N6	采样频率=死区发生器时钟频率 2 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV2_N8	采样频率=死区发生器时钟频率 2 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV4_N6	采样频率=死区发生器时钟频率 4 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV4_N8	采样频率=死区发生器时钟频率 4 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV8_N6	采样频率=死区发生器时钟频率 8 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV8_N8	采样频率=死区发生器时钟频率 8 分频，采样 8 次

LL_TIM_IC_FILTER_FDIV16_N5	采样频率=死区发生器时钟频率 16 分频，采样 5 次
LL_TIM_IC_FILTER_FDIV16_N6	采样频率=死区发生器时钟频率 16 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV16_N8	采样频率=死区发生器时钟频率 16 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV32_N5	采样频率=死区发生器时钟频率 32 分频，采样 5 次
LL_TIM_IC_FILTER_FDIV32_N6	采样频率=死区发生器时钟频率 32 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV32_N8	采样频率=死区发生器时钟频率 32 分频，采样 8 次

IC2Polarity 可选参数：

表45-22 IC2Polarity 可选参数

参数	描述
LL_TIM_IC_POLARITY_RISING	上升沿有效
LL_TIM_IC_POLARITY_FALLING	下降沿有效
LL_TIM_IC_POLARITY_BOTHEDGE	双边沿有效

IC2ActiveInput 可选参数：

表45-23 IC2ActiveInput 可选参数

参数	描述
LL_TIM_ACTIVEINPUT_DIRECTTI	输入模式，ICx 映射在 Tix 上
LL_TIM_ACTIVEINPUT_INDIRECTTI	输入模式，ICx 映射在 Tiy 上
LL_TIM_ACTIVEINPUT_TRC	输入模式，ICx 映射在 TRC 上

注意：当 *x=1* 时 *y=2*，当 *x=2* 时 *y=1*，当 *x=3* 时，*y=4*，当 *x=4* 时 *y=3*。

IC2Prescaler 可选参数：

表45-24 IC2Prescaler 可选参数

参数	描述
LL_TIM_ICPSC_DIV1	输入信号不分频
LL_TIM_ICPSC_DIV2	输入信号 2 分频
LL_TIM_ICPSC_DIV4	输入信号 4 分频
LL_TIM_ICPSC_DIV8	输入信号 8 分频

IC2Filter 可选参数：

表45-25 IC2Filter 可选参数

参数	描述
LL_TIM_IC_FILTER_FDIV1	无滤波
LL_TIM_IC_FILTER_FDIV1_N2	采样频率=定时器时钟频率，采样 2 次
LL_TIM_IC_FILTER_FDIV1_N4	采样频率=定时器时钟频率，采样 4 次
LL_TIM_IC_FILTER_FDIV1_N8	采样频率=定时器时钟频率，采样 8 次

LL_TIM_IC_FILTER_FDIV2_N6	采样频率=死区发生器时钟频率 2 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV2_N8	采样频率=死区发生器时钟频率 2 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV4_N6	采样频率=死区发生器时钟频率 4 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV4_N8	采样频率=死区发生器时钟频率 4 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV8_N6	采样频率=死区发生器时钟频率 8 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV8_N8	采样频率=死区发生器时钟频率 8 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV16_N5	采样频率=死区发生器时钟频率 16 分频，采样 5 次
LL_TIM_IC_FILTER_FDIV16_N6	采样频率=死区发生器时钟频率 16 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV16_N8	采样频率=死区发生器时钟频率 16 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV32_N5	采样频率=死区发生器时钟频率 32 分频，采样 5 次
LL_TIM_IC_FILTER_FDIV32_N6	采样频率=死区发生器时钟频率 32 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV32_N8	采样频率=死区发生器时钟频率 32 分频，采样 8 次

45.1.5 LL_TIM_HALLSENSOR_InitTypeDef

LL_TIM_HALLSENSOR_InitTypeDef，定义于文件"air001xx_ll_tim.h"如下：

```
typedef struct
{
uint32_t IC1Polarity;
uint32_t IC1Prescaler;
uint32_t IC1Filter;
uint32_t CommutationDelay;
} LL_TIM_HALLSENSOR_InitTypeDef;
```

字段说明：

表45-26 LL_TIM_HALLSENSOR_InitTypeDef 字段说明

字段	描述
IC1Polarity	输入捕获极性
IC1Prescaler	输入信号预分频值
IC1Filter	输入信号滤波值
CommutationDelay	换向信号延迟（0x0000~0xFFFF）

参数说明：

IC1Polarity 可选参数：

表45-27 IC1Polarity 可选参数

参数	描述
LL_TIM_IC_POLARITY_RISING	上升沿捕获
LL_TIM_IC_POLARITY_FALLING	下降沿捕获

LL_TIM_IC_POLARITY_BOTHEDGE	双边沿捕获

IC1Prescaler 可选参数：

表45-28 IC1Prescaler 可选参数

参数	描述
LL_TIM_ICPSC_DIV1	输入信号不分频
LL_TIM_ICPSC_DIV2	输入信号 2 分频
LL_TIM_ICPSC_DIV4	输入信号 4 分频
LL_TIM_ICPSC_DIV8	输入信号 8 分频

IC1Filter 可选参数：

表45-29 IC1Filter 可选参数

参数	描述
LL_TIM_IC_FILTER_FDIV1	无滤波
LL_TIM_IC_FILTER_FDIV1_N2	采样频率=定时器时钟频率，采样 2 次
LL_TIM_IC_FILTER_FDIV1_N4	采样频率=定时器时钟频率，采样 4 次
LL_TIM_IC_FILTER_FDIV1_N8	采样频率=定时器时钟频率，采样 8 次
LL_TIM_IC_FILTER_FDIV2_N6	采样频率=死区发生器时钟频率 2 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV2_N8	采样频率=死区发生器时钟频率 2 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV4_N6	采样频率=死区发生器时钟频率 4 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV4_N8	采样频率=死区发生器时钟频率 4 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV8_N6	采样频率=死区发生器时钟频率 8 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV8_N8	采样频率=死区发生器时钟频率 8 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV16_N5	采样频率=死区发生器时钟频率 16 分频，采样 5 次
LL_TIM_IC_FILTER_FDIV16_N6	采样频率=死区发生器时钟频率 16 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV16_N8	采样频率=死区发生器时钟频率 16 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV32_N5	采样频率=死区发生器时钟频率 32 分频，采样 5 次
LL_TIM_IC_FILTER_FDIV32_N6	采样频率=死区发生器时钟频率 32 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV32_N8	采样频率=死区发生器时钟频率 32 分频，采样 8 次

45.1.6 LL_TIM_BDTR_InitTypeDef

LL_TIM_BDTR_InitTypeDef，定义于文件"air001xx_ll_tim.h"如下：

```
typedef struct
{
uint32_t OSSRState;
uint32_t OSSIState;
uint32_t LockLevel;
uint8_t DeadTime;
```

```
uint16_t BreakState;
uint32_t BreakPolarity;
uint32_t AutomaticOutput;
} LL_TIM_BDTR_InitTypeDef;
```

字段说明：

表45-30 LL_TIM_BDTR_InitTypeDef 字段说明

字段	描述
OSSRState	运行模式下“关闭状态”选择
OSSIState	空闲模式下“关闭状态”选择
LockLevel	寄存器锁定级别
DeadTime	死区时间
BreakState	刹车使能
BreakPolarity	刹车输入极性
AutomaticOutput	自动输出使能

OSSRState 可选参数：

表45-31 OSSRState 可选参数

参数	描述
LL_TIM_OSSR_DISABLE	“关闭状态”下输出被禁用
LL_TIM_OSSR_ENABLE	“关闭状态”下如果输出使能，输出无效电平

OSSIState 可选参数：

表45-32 OSSIState 可选参数

参数	描述
LL_TIM_OSSI_DISABLE	“关闭状态”下输出被禁用
LL_TIM_OSSI_ENABLE	“关闭状态”下如果输出使能，输出空闲电平

LockLevel 可选参数：

表45-33 LockLevel 可选参数

参数	描述
LL_TIM_LOCKLEVEL_OFF	无锁定
LL_TIM_LOCKLEVEL_1	锁定等级 1
LL_TIM_LOCKLEVEL_2	锁定等级 2
LL_TIM_LOCKLEVEL_3	锁定等级 3

BreakState 可选参数：

表45-34 BreakState 可选参数

参数	描述
LL_TIM_BREAK_DISABLE	禁用刹车
LL_TIM_BREAK_ENABLE	使能刹车

BreakPolarity 可选参数：

表45-35 BreakPolarity 可选参数

参数	描述
LL_TIM_BREAK_POLARITY_LOW	刹车低电平有效
LL_TIM_BREAK_POLARITY_HIGH	刹车高电平有效

AutomaticOutput 可选参数：

表45-36 AutomaticOutput 可选参数

参数	描述
LL_TIM_AUTOMATICOUTPUT_DISABLE	禁用自动输出
LL_TIM_AUTOMATICOUTPUT_ENABLE	使能自动输出

45.2 TIM 固件库函数

函数名	描述
LL_TIM_EnableCounter	开启计数器
LL_TIM_DisableCounter	关闭计数器
LL_TIM_IsEnabledCounter	检查是否开启计数器
LL_TIM_EnableUpdateEvent	允许更新事件产生
LL_TIM_DisableUpdateEvent	禁止更新事件产生
LL_TIM_IsEnabledUpdateEvent	检查是否允许更新事件产生
LL_TIM_SetUpdateSource	设置更新请求源
LL_TIM_GetUpdateSource	获取更新请求源
LL_TIM_SetOnePulseMode	设置单脉冲模式
LL_TIM_GetOnePulseMode	获取单脉冲模式
LL_TIM_SetCounterMode	设置计数模式
LL_TIM_GetCounterMode	获取计数模式
LL_TIM_EnableARRPreload	使能自动重装载预装载
LL_TIM_DisableARRPreload	禁用自动重装载预装载
LL_TIM_IsEnabledARRPreload	检查是否使能自动重装载预装载

LL_TIM_SetClockDivision	设置死区发生器、数字滤波器时钟分频
LL_TIM_GetClockDivision	获取死区发生器、数字滤波器时钟分频
LL_TIM_SetCounter	设置计数值
LL_TIM_GetCounter	获取计数值
LL_TIM_GetDirection	获取计数方向
LL_TIM_SetPrescaler	设置预分频值
LL_TIM_GetPrescaler	获取预分频值
LL_TIM_SetAutoReload	设置自动重装载值
LL_TIM_GetAutoReload	获取自动重装载值
LL_TIM_SetRepetitionCounter	设置重复计数值
LL_TIM_GetRepetitionCounter	获取重复计数值
LL_TIM_CC_EnablePreload	使能捕获/比较预装载
LL_TIM_CC_DisablePreload	禁用捕获/比较预装载
LL_TIM_CC_SetUpdate	设置捕获/比较预装载更新源
LL_TIM_CC_SetDMAReqTrigger	设置捕获/比较 DMA 请求触发源
LL_TIM_CC_GetDMAReqTrigger	获取捕获/比较 DMA 请求触发源
LL_TIM_CC_SetLockLevel	设置锁定级别
LL_TIM_CC_EnableChannel	开启指定通道
LL_TIM_CC_DisableChannel	关闭指定通道
LL_TIM_CC_IsEnabledChannel	检查指定通道是否开启
LL_TIM_OC_ConfigOutput	配置通道为输出
LL_TIM_OC_SetMode	设置输出模式
LL_TIM_OC_GetMode	获取输出模式
LL_TIM_OC_SetPolarity	设置输出有效极性
LL_TIM_OC_GetPolarity	获取输出有效极性
LL_TIM_OC_SetIdleState	设置空闲状态电平
LL_TIM_OC_GetIdleState	获取空闲状态电平配置
LL_TIM_OC_EnableFast	开启快速模式
LL_TIM_OC_DisableFast	禁用快速模式
LL_TIM_OC_IsEnabledFast	检查是否使能快速模式
LL_TIM_OC_EnablePreload	使能指定通道预装载

LL_TIM_OC_DisablePreload	禁用指定通道预装载
LL_TIM_OC_IsEnabledPreload	检查指定通道是否开启预装载
LL_TIM_OC_EnableClear	使能外部事件清除通道输出功能
LL_TIM_OC_DisableClear	禁用外部事件清除输出功能
LL_TIM_OC_IsEnabledClear	检查是否使能外部事件清除输出功能
LL_TIM_OC_SetDeadTime	设置死区时间
LL_TIM_OC_SetCompareCH1	设置通道 1 比较值
LL_TIM_OC_SetCompareCH2	设置通道 2 比较值
LL_TIM_OC_SetCompareCH3	设置通道 3 比较值
LL_TIM_OC_SetCompareCH4	设置通道 4 比较值
LL_TIM_OC_GetCompareCH1	获取通道 1 比较值
LL_TIM_OC_GetCompareCH2	获取通道 2 比较值
LL_TIM_OC_GetCompareCH3	获取通道 3 比较值
LL_TIM_OC_GetCompareCH4	获取通道 4 比较值
LL_TIM_IC_Config	配置通道为输入
LL_TIM_IC_SetActiveInput	设置有效输入
LL_TIM_IC_GetActiveInput	获取有效输入
LL_TIM_IC_SetPrescaler	设置输入信号预分频值
LL_TIM_IC_GetPrescaler	获取输入信号预分频值
LL_TIM_IC_SetFilter	设置数字滤波器
LL_TIM_IC_GetFilter	获取滤波值
LL_TIM_IC_SetPolarity	设置输入有效边沿
LL_TIM_IC_GetPolarity	获取输入有效边沿
LL_TIM_IC_EnableXORCombination	使能 XOR 输入
LL_TIM_IC_DisableXORCombination	禁用 XOR 输入
LL_TIM_IC_IsEnabledXORCombination	检查是否开启 XOR 输入
LL_TIM_IC_GetCaptureCH1	获取通道 1 的捕获值
LL_TIM_IC_GetCaptureCH2	获取通道 2 的捕获值
LL_TIM_IC_GetCaptureCH3	获取通道 3 的捕获值
LL_TIM_IC_GetCaptureCH4	获取通道 4 的捕获值
LL_TIM_EnableExternalClock	使能外部时钟

LL_TIM_DisableExternalClock	禁用外部时钟
LL_TIM_IsEnabledExternalClock	检查是否开启外部时钟
LL_TIM_SetClockSource	设置外部时钟源
LL_TIM_SetEncoderMode	设置编码器接口模式
LL_TIM_SetTriggerOutput	设置主定时器的同步触发输出信号
LL_TIM_SetSlaveMode	设置从定时器的同步方式
LL_TIM_SetTriggerInput	设置同步计数器的触发输入
LL_TIM_EnableMasterSlaveMode	使能主从模式
LL_TIM_DisableMasterSlaveMode	禁用主从模式
LL_TIM_IsEnabledMasterSlaveMode	检查是否开启主从模式
LL_TIM_ConfigETR	配置外部触发输入
LL_TIM_EnableBRK	使能刹车输入
LL_TIM_DisableBRK	禁用刹车输入
LL_TIM_ConfigBRK	配置刹车输入
LL_TIM_SetOffStates	设置运行和空闲模式下“关闭状态”
LL_TIM_EnableAutomaticOutput	开启自动输出
LL_TIM_DisableAutomaticOutput	禁用自动输出
LL_TIM_IsEnabledAutomaticOutput	检查自动输出是否开启
LL_TIM_EnableAllOutputs	使能主输出
LL_TIM_DisableAllOutputs	禁用主输出
LL_TIM_IsEnabledAllOutputs	检查主输出是否开启
LL_TIM_ConfigDMABurst	配置 DMA 突发模式
LL_TIM_SetOCRefClearInputSource	设置 OCREF 清除输入源
LL_TIM_ClearFlag_UPDATE	清除更新中断标志
LL_TIM_IsActiveFlag_UPDATE	检查是否产生更新中断
LL_TIM_ClearFlag_CC1	清除捕获/比较 1 中断标志
LL_TIM_IsActiveFlag_CC1	检查是否产生捕获/比较 1 中断
LL_TIM_ClearFlag_CC2	清除捕获/比较 2 中断标志
LL_TIM_IsActiveFlag_CC2	检查是否产生捕获/比较 2 中断
LL_TIM_ClearFlag_CC3	清除捕获/比较 3 中断标志
LL_TIM_IsActiveFlag_CC3	检查是否产生捕获/比较 3 中断

LL_TIM_ClearFlag_CC4	清除捕获/比较 4 中断标志
LL_TIM_IsActiveFlag_CC4	检查是否产生捕获/比较 4 中断
LL_TIM_ClearFlag_COM	清除换向中断标志
LL_TIM_IsActiveFlag_COM	检查是否产生换向中断
LL_TIM_ClearFlag_TRIG	清除触发器中断标志
LL_TIM_IsActiveFlag_TRIG	检查是否产生触发器中断
LL_TIM_ClearFlag_BRK	清除刹车中断标志
LL_TIM_IsActiveFlag_BRK	检查是否产生刹车中断
LL_TIM_ClearFlag_CC1OVR	清除捕获/比较 1 过捕获中断标志
LL_TIM_IsActiveFlag_CC1OVR	检查是否产生捕获/比较 1 过捕获中断
LL_TIM_ClearFlag_CC2OVR	清除捕获/比较 2 过捕获中断标志
LL_TIM_IsActiveFlag_CC2OVR	检查是否产生捕获/比较 2 过捕获中断
LL_TIM_ClearFlag_CC3OVR	清除捕获/比较 3 过捕获中断标志
LL_TIM_IsActiveFlag_CC3OVR	检查是否产生捕获/比较 3 过捕获中断
LL_TIM_ClearFlag_CC4OVR	清除捕获/比较 4 过捕获中断标志
LL_TIM_IsActiveFlag_CC4OVR	检查是否产生捕获/比较 4 过捕获中断
LL_TIM_EnableIT_UPDATE	使能更新中断
LL_TIM_DisableIT_UPDATE	禁用更新中断
LL_TIM_IsEnabledIT_UPDATE	检查是否开启更新中断
LL_TIM_EnableIT_CC1	使能捕获/比较 1 中断
LL_TIM_DisableIT_CC1	禁用捕获/比较 1 中断
LL_TIM_IsEnabledIT_CC1	检查是否开启捕获/比较 1 中断
LL_TIM_EnableIT_CC2	使能捕获/比较 2 中断
LL_TIM_DisableIT_CC2	禁用捕获/比较 2 中断
LL_TIM_IsEnabledIT_CC2	检查是否开启捕获/比较 2 中断
LL_TIM_EnableIT_CC3	使能捕获/比较 3 中断
LL_TIM_DisableIT_CC3	禁用捕获/比较 3 中断
LL_TIM_IsEnabledIT_CC3	检查是否开启捕获/比较 3 中断
LL_TIM_EnableIT_CC4	使能捕获/比较 4 中断
LL_TIM_DisableIT_CC4	禁用捕获/比较 4 中断
LL_TIM_IsEnabledIT_CC4	检查是否开启捕获/比较 4 中断

LL_TIM_EnableIT_COM	使能换向中断
LL_TIM_DisableIT_COM	禁用换向中断
LL_TIM_IsEnabledIT_COM	检查是否开启换向中断
LL_TIM_EnableIT_TRIG	使能触发器中断
LL_TIM_DisableIT_TRIG	禁用触发器中断
LL_TIM_IsEnabledIT_TRIG	检查是否开启触发器中断
LL_TIM_EnableIT_BRK	使能刹车中断
LL_TIM_DisableIT_BRK	禁用刹车中断
LL_TIM_IsEnabledIT_BRK	检查是否开启刹车中断
LL_TIM_EnableDMAReq_UPDATE	使能更新事件DMA 请求
LL_TIM_DisableDMAReq_UPDATE	禁用更新事件DMA 请求
LL_TIM_IsEnabledDMAReq_UPDATE	检查是否开启更新事件 DMA 请求
LL_TIM_EnableDMAReq_CC1	使能捕获/比较 1 事件DMA 请求
LL_TIM_DisableDMAReq_CC1	禁用捕获/比较 1 事件DMA 请求
LL_TIM_IsEnabledDMAReq_CC1	检查是否开启捕获/比较 1 事件DMA 请求
LL_TIM_EnableDMAReq_CC2	使能捕获/比较 2 事件DMA 请求
LL_TIM_DisableDMAReq_CC2	禁用捕获/比较 2 事件DMA 请求
LL_TIM_IsEnabledDMAReq_CC2	检查是否开启捕获/比较 2 事件DMA 请求
LL_TIM_EnableDMAReq_CC3	使能捕获/比较 3 事件DMA 请求
LL_TIM_DisableDMAReq_CC3	禁用捕获/比较 3 事件DMA 请求
LL_TIM_IsEnabledDMAReq_CC3	检查是否开启捕获/比较 3 事件DMA 请求
LL_TIM_EnableDMAReq_CC4	使能捕获/比较 4 事件DMA 请求
LL_TIM_DisableDMAReq_CC4	禁用捕获/比较 4 事件DMA 请求
LL_TIM_IsEnabledDMAReq_CC4	检查是否开启捕获/比较 4 事件DMA 请求
LL_TIM_EnableDMAReq_COM	使能换向事件DMA 请求
LL_TIM_DisableDMAReq_COM	禁用换向事件DMA 请求
LL_TIM_IsEnabledDMAReq_COM	检查是否开启换向事件 DMA 请求
LL_TIM_EnableDMAReq_TRIG	使能触发器事件 DMA 请求
LL_TIM_DisableDMAReq_TRIG	禁用触发器事件 DMA 请求
LL_TIM_IsEnabledDMAReq_TRIG	检查是否开启触发器事件DMA 请求
LL_TIM_GenerateEvent_UPDATE	产生更新事件

LL_TIM_GenerateEvent_CC1	产生捕获/比较 1 事件
LL_TIM_GenerateEvent_CC2	产生捕获/比较 2 事件
LL_TIM_GenerateEvent_CC3	产生捕获/比较 3 事件
LL_TIM_GenerateEvent_CC4	产生捕获/比较 4 事件
LL_TIM_GenerateEvent_COM	产生换向事件
LL_TIM_GenerateEvent_TRIG	产生触发器事件
LL_TIM_GenerateEvent_BRK	产生刹车事件
LL_TIM_DeInit	将 TIMx 寄存器重设为缺省值
LL_TIM_StructInit	将时基结构体配置为默认值
LL_TIM_Init	初始化时基配置
LL_TIM_OC_StructInit	将输出比较结构体配置为默认值
LL_TIM_OC_Init	初始化输出比较配置
LL_TIM_IC_StructInit	将输入捕获结构体配置为默认值
LL_TIM_IC_Init	初始化输入捕获配置
LL_TIM_ENCODER_StructInit	将编码器结构体配置为默认值
LL_TIM_ENCODER_Init	初始化编码器配置
LL_TIM_HALLSENSOR_StructInit	将霍尔传感器结构体配置为默认值
LL_TIM_HALLSENSOR_Init	初始化霍尔传感器配置
LL_TIM_BDTR_StructInit	将 BDTR 结构体配置为默认值
LL_TIM_BDTR_Init	初始化 BDTR 配置

45.2.1 函数 LL_TIM_EnableCounter

描述了函数 LL_TIM_EnableCounter

表45-37　函数 LL_TIM_EnableCounter

函数名	LL_TIM_EnableCounter
函数原形	__STATIC_INLINE void LL_TIM_EnableCounter(TIM_TypeDef *TIMx)
功能描述	开启计数器
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.2 函数 LL_TIM_DisableCounter

描述了函数 LL_TIM_DisableCounter

表45-38 函数 LL_TIM_DisableCounter

函数名	LL_TIM_DisableCounter
函数原形	__STATIC_INLINE void LL_TIM_DisableCounter(TIM_TypeDef *TIMx)
功能描述	关闭计数器
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.3 函数 LL_TIM_IsEnabledCounter

描述了函数 LL_TIM_IsEnabledCounter

表45-39 函数 LL_TIM_IsEnabledCounter

函数名	LL_TIM_IsEnabledCounter
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledCounter(TIM_TypeDef *TIMx)
功能描述	检查是否开启计数器
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.4 函数 LL_TIM_EnableUpdateEvent

描述了函数 LL_TIM_EnableUpdateEvent

表45-40 函数 LL_TIM_EnableUpdateEvent

函数名	LL_TIM_EnableUpdateEvent
函数原形	__STATIC_INLINE void LL_TIM_EnableUpdateEvent(TIM_TypeDef *TIMx)
功能描述	允许更新事件产生
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.5 函数 LL_TIM_DisableUpdateEvent

描述了函数 LL_TIM_DisableUpdateEvent

表45-41 函数 LL_TIM_DisableUpdateEvent

函数名	LL_TIM_DisableUpdateEvent
函数原形	__STATIC_INLINE void LL_TIM_DisableUpdateEvent(TIM_TypeDef *TIMx)
功能描述	禁止更新事件产生
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.6 函数 LL_TIM_IsEnabledUpdateEvent

描述了函数 LL_TIM_IsEnabledUpdateEvent

表45-42 函数 LL_TIM_IsEnabledUpdateEvent

函数名	LL_TIM_IsEnabledUpdateEvent
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledUpdateEvent(TIM_TypeDef *TIMx)
功能描述	检查是否允许更新事件产生
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.7 函数 LL_TIM_SetUpdateSource

描述了函数 LL_TIM_SetUpdateSource

表45-43 函数 LL_TIM_SetUpdateSource

函数名	LL_TIM_SetUpdateSource
函数原形	__STATIC_INLINE void LL_TIM_SetUpdateSource(TIM_TypeDef *TIMx, uint32_t UpdateSource)
功能描述	设置更新请求源
输入参数 1	TIMx：定时器实例
输入参数 2	UpdateSource：更新请求源
输出参数	无
返回值	无
先决条件	无

UpdateSource 可选参数:

表45-44 UpdateSource 可选参数

参数	描述
LL_TIM_UPDATESOURCE_REGULAR	计数器溢出/下溢、设置 UG 位和从模式控制器产生更新
LL_TIM_UPDATESOURCE_COUNTER	仅计数器溢出/下溢

45.2.8 函数 LL_TIM_GetUpdateSource

描述了函数 LL_TIM_GetUpdateSource

表45-45 函数 LL_TIM_GetUpdateSource

函数名	LL_TIM_GetUpdateSource
函数原形	__STATIC_INLINE uint32_t LL_TIM_GetUpdateSource(TIM_TypeDef *TIMx)
功能描述	获取更新请求源
输入参数	TIMx：定时器实例
输出参数	无
返回值	更新请求源
先决条件	无

45.2.9 函数 LL_TIM_SetOnePulseMode

描述了函数 LL_TIM_SetOnePulseMode

表45-46 函数 LL_TIM_SetOnePulseMode

函数名	LL_TIM_SetOnePulseMode
函数原形	__STATIC_INLINE void LL_TIM_SetOnePulseMode(TIM_TypeDef *TIMx, uint32_t OnePulseMode)
功能描述	设置单脉冲模式
输入参数 1	TIMx：定时器实例
输入参数 2	OnePulseMode：单脉冲模式
输出参数	无
返回值	无
先决条件	无

OnePulseMode 可选参数:

表45-47 OnePulseMode 可选参数

参数	描述
LL_TIM_ONEPULSEMODE_SINGLE	计数器在下一次更新事件时不停止计数
LL_TIM_ONEPULSEMODE_REPETITIVE	计数器在下一次更新事件时停止计数

45.2.10 函数 LL_TIM_GetOnePulseMode

描述了函数 LL_TIM_GetOnePulseMode

表45-48 函数 LL_TIM_GetOnePulseMode

函数名	LL_TIM_GetOnePulseMode
函数原形	__STATIC_INLINE uint32_t LL_TIM_GetOnePulseMode(TIM_TypeDef *TIMx)
功能描述	获取单脉冲模式
输入参数	TIMx：定时器实例

输出参数	无
返回值	单脉冲模式
先决条件	无

45.2.11 函数 LL_TIM_SetCounterMode

描述了函数 LL_TIM_SetCounterMode

表45-49 函数 LL_TIM_SetCounterMode

函数名	LL_TIM_SetCounterMode
函数原形	__STATIC_INLINE void LL_TIM_SetCounterMode(TIM_TypeDef *TIMx, uint32_t CounterMode)
功能描述	设置计数模式
输入参数 1	TIMx：定时器实例
输入参数 2	CounterMode：计数模式
输出参数	无
返回值	无
先决条件	无

CounterMode 可选参数:

表45-50 CounterMode 可选参数

参数	描述
LL_TIM_COUNTERMODE_UP	向上计数
LL_TIM_COUNTERMODE_DOWN	向下计数
LL_TIM_COUNTERMODE_CENTER_UP	中央对齐模式，仅向上计数触发比较中断标志
LL_TIM_COUNTERMODE_CENTER_DOWN	中央对齐模式，仅向下计数触发比较中断标志
LL_TIM_COUNTERMODE_CENTER_UP_DOWN	中央对齐模式，向上向下计数都能触发比较中断标志

45.2.12 函数 LL_TIM_GetCounterMode

描述了函数 LL_TIM_GetCounterMode

表45-51 函数 LL_TIM_GetCounterMode

函数名	LL_TIM_GetCounterMode
函数原形	__STATIC_INLINE uint32_t LL_TIM_GetCounterMode(TIM_TypeDef *TIMx)
功能描述	获取计数模式
输入参数	TIMx：定时器实例
输出参数	无
返回值	计数模式
先决条件	无

45.2.13 函数 LL_TIM_EnableARRPreload

描述了函数 LL_TIM_EnableARRPreload

表45-52 函数 LL_TIM_EnableARRPreload

函数名	LL_TIM_EnableARRPreload
函数原形	__STATIC_INLINE void LL_TIM_EnableARRPreload(TIM_TypeDef *TIMx)
功能描述	使能自动重装载预装载
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.14 函数 LL_TIM_DisableARRPreload

描述了函数 LL_TIM_DisableARRPreload

表45-53 函数 LL_TIM_DisableARRPreload

函数名	LL_TIM_DisableARRPreload
函数原形	__STATIC_INLINE void LL_TIM_DisableARRPreload(TIM_TypeDef *TIMx)
功能描述	禁用自动重装载预装载
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.15 函数 LL_TIM_IsEnabledARRPreload

描述了函数 LL_TIM_IsEnabledARRPreload

表45-54 函数 LL_TIM_IsEnabledARRPreload

函数名	LL_TIM_IsEnabledARRPreload
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledARRPreload(TIM_TypeDef *TIMx)
功能描述	检查是否使能自动重装载预装载
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.16 函数 LL_TIM_SetClockDivision

描述了函数 LL_TIM_SetClockDivision

表45-55 函数 LL_TIM_SetClockDivision

函数名	LL_TIM_SetClockDivision
函数原形	__STATIC_INLINE void LL_TIM_SetClockDivision(TIM_TypeDef *TIMx, uint32_t ClockDivision)
功能描述	设置死区发生器、数字滤波器时钟分频
输入参数 1	TIMx：定时器实例
输入参数 2	ClockDivision：分频值
输出参数	无
返回值	无
先决条件	无

ClockDivision 可选参数:

表45-56 ClockDivision 可选参数

参数	描述
LL_TIM_CLOCKDIVISION_DIV1	不分频
LL_TIM_CLOCKDIVISION_DIV2	二分频
LL_TIM_CLOCKDIVISION_DIV4	四分频

45.2.17 函数 LL_TIM_GetClockDivision

描述了函数 LL_TIM_GetClockDivision

表45-57 函数 LL_TIM_GetClockDivision

函数名	LL_TIM_GetClockDivision
函数原形	__STATIC_INLINE uint32_t LL_TIM_GetClockDivision(TIM_TypeDef *TIMx)
功能描述	获取死区发生器、数字滤波器时钟分频
输入参数	TIMx：定时器实例
输出参数	无
返回值	分频值
先决条件	无

45.2.18 函数 LL_TIM_SetCounter

描述了函数 LL_TIM_SetCounter

表45-58 函数 LL_TIM_SetCounter

函数名	LL_TIM_SetCounter
函数原形	__STATIC_INLINE void LL_TIM_SetCounter(TIM_TypeDef *TIMx, uint32_t Counter)
功能描述	设置计数值
输入参数 1	TIMx：定时器实例
输入参数 2	Counter：计数值
输出参数	无

返回值	无
先决条件	无

45.2.19 函数 LL_TIM_GetCounter

描述了函数 LL_TIM_GetCounter

表45-59 函数 LL_TIM_GetCounter

函数名	LL_TIM_GetCounter
函数原形	__STATIC_INLINE uint32_t LL_TIM_GetCounter(TIM_TypeDef *TIMx)
功能描述	获取计数值
输入参数	TIMx：定时器实例
输出参数	无
返回值	计数值
先决条件	无

45.2.20 函数 LL_TIM_GetDirection

描述了函数 LL_TIM_GetDirection

表45-60 函数 LL_TIM_GetDirection

函数名	LL_TIM_GetDirection
函数原形	__STATIC_INLINE uint32_t LL_TIM_GetDirection(TIM_TypeDef *TIMx)
功能描述	获取计数方向
输入参数	TIMx：定时器实例
输出参数	无
返回值	计数方向
先决条件	无

45.2.21 函数 LL_TIM_SetPrescaler

描述了函数 LL_TIM_SetPrescaler

表45-61 函数 LL_TIM_SetPrescaler

函数名	LL_TIM_SetPrescaler
函数原形	__STATIC_INLINE void LL_TIM_SetPrescaler(TIM_TypeDef *TIMx, uint32_t Prescaler)
功能描述	设置预分频值
输入参数	TIMx：定时器实例
输出参数	Prescaler：预分频值
返回值	无
先决条件	无

45.2.22 函数 LL_TIM_GetPrescaler

描述了函数 LL_TIM_GetPrescaler

表45-62 函数 LL_TIM_GetPrescaler

函数名	LL_TIM_GetPrescaler
函数原形	__STATIC_INLINE uint32_t LL_TIM_GetPrescaler(TIM_TypeDef *TIMx)
功能描述	获取预分频值
输入参数	TIMx：定时器实例
输出参数	无
返回值	预分频值
先决条件	无

45.2.23 函数 LL_TIM_SetAutoReload

描述了函数 LL_TIM_SetAutoReload

表45-63 函数 LL_TIM_SetAutoReload

函数名	LL_TIM_SetAutoReload
函数原形	__STATIC_INLINE void LL_TIM_SetAutoReload(TIM_TypeDef *TIMx, uint32_t AutoReload)
功能描述	设置自动重装载值
输入参数 1	TIMx：定时器实例
输入参数 2	AutoReload：自动重装载值
输出参数	无
返回值	无
先决条件	无

45.2.24 函数 LL_TIM_GetAutoReload

描述了函数 LL_TIM_GetAutoReload

表45-64 函数 LL_TIM_GetAutoReload

函数名	LL_TIM_GetAutoReload
函数原形	__STATIC_INLINE uint32_t LL_TIM_GetAutoReload(TIM_TypeDef *TIMx)
功能描述	获取自动重装载值
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.25 函数 LL_TIM_SetRepetitionCounter

描述了函数 LL_TIM_SetRepetitionCounter

表45-65 函数 LL_TIM_SetRepetitionCounter

函数名	LL_TIM_SetRepetitionCounter
函数原形	__STATIC_INLINE void LL_TIM_SetRepetitionCounter(TIM_TypeDef *TIMx, uint32_t RepetitionCounter)
功能描述	设置重复计数值
输入参数 1	TIMx：定时器实例
输入参数 2	RepetitionCounter：重复计数值
输出参数	无
返回值	无
先决条件	TIMx：定时器实例

45.2.26 函数 LL_TIM_GetRepetitionCounter

描述了函数 LL_TIM_GetRepetitionCounter

表45-66 函数 LL_TIM_GetRepetitionCounter

函数名	LL_TIM_GetRepetitionCounter
函数原形	__STATIC_INLINE uint32_t LL_TIM_GetRepetitionCounter(TIM_TypeDef *TIMx)
功能描述	获取重复计数值
输入参数	TIMx：定时器实例
输出参数	无
返回值	重复计数值
先决条件	无

45.2.27 函数 LL_TIM_CC_EnablePreload

描述了函数 LL_TIM_CC_EnablePreload

表45-67 函数 LL_TIM_CC_EnablePreload

函数名	LL_TIM_CC_EnablePreload
函数原形	__STATIC_INLINE void LL_TIM_CC_EnablePreload(TIM_TypeDef *TIMx)
功能描述	使能捕获/比较预装载
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.28 函数 LL_TIM_CC_DisablePreload

描述了函数 LL_TIM_CC_DisablePreload

表45-68 函数 LL_TIM_CC_DisablePreload

函数名	LL_TIM_CC_DisablePreload

函数原形	__STATIC_INLINE void LL_TIM_CC_DisablePreload(TIM_TypeDef *TIMx)
功能描述	禁用捕获/比较预装载
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.29 函数 LL_TIM_CC_SetUpdate

描述了函数 LL_TIM_CC_SetUpdate

表45-69 函数 LL_TIM_CC_SetUpdate

函数名	LL_TIM_CC_SetUpdate
函数原形	__STATIC_INLINE void LL_TIM_CC_SetUpdate(TIM_TypeDef *TIMx, uint32_t CCUpdateSource)
功能描述	设置捕获/比较预装载更新源
输入参数 1	TIMx：定时器实例
输入参数 2	CCUpdateSource：更新源
输出参数	无
返回值	无
先决条件	无

CCUpdateSource 可选参数：

表45-70 CCUpdateSource 可选参数

参数	描述
LL_TIM_CCUPDATESOURCE_COMG_ONLY	仅换向事件
LL_TIM_CCUPDATESOURCE_COMG_AND_TRGI	换向和触发器事件

45.2.30 函数 LL_TIM_CC_SetDMAReqTrigger

描述了函数 LL_TIM_CC_SetDMAReqTrigger

表45-71 函数 LL_TIM_CC_SetDMAReqTrigger

函数名	LL_TIM_CC_SetDMAReqTrigger
函数原形	__STATIC_INLINE void LL_TIM_CC_SetDMAReqTrigger(TIM_TypeDef *TIMx, uint32_t DMAReqTrigger)
功能描述	设置捕获/比较 DMA 请求触发源
输入参数 1	TIMx：定时器实例
输入参数 2	DMAReqTrigger：触发源
输出参数	无
返回值	无
先决条件	无

DMAReqTrigger 可选参数：

表45-72 DMAReqTrigger 可选参数

参数	描述
LL_TIM_CCDMAREQUEST_CC	发生捕获/比较事件时产生捕获/比较的 DMA 请求
LL_TIM_CCDMAREQUEST_UPDATE	发生更新事件时产生捕获/比较的 DMA 请求

45.2.31 函数 LL_TIM_CC_GetDMAReqTrigger

描述了函数 LL_TIM_CC_GetDMAReqTrigger

表45-73 函数 LL_TIM_CC_GetDMAReqTrigger

函数名	LL_TIM_CC_GetDMAReqTrigger
函数原形	__STATIC_INLINE uint32_t LL_TIM_CC_GetDMAReqTrigger(TIM_TypeDef *TIMx)
功能描述	获取捕获/比较 DMA 请求触发源
输入参数	TIMx：定时器实例
输出参数	无
返回值	触发源
先决条件	无

45.2.32 函数 LL_TIM_CC_SetLockLevel

描述了函数 LL_TIM_CC_SetLockLevel

表45-74 函数 LL_TIM_CC_SetLockLevel

函数名	LL_TIM_CC_SetLockLevel
函数原形	__STATIC_INLINE void LL_TIM_CC_SetLockLevel(TIM_TypeDef *TIMx, uint32_t LockLevel)
功能描述	设置寄存器锁定级别
输入参数 1	TIMx：定时器实例
输入参数 2	LockLevel：锁定级别
输出参数	无
返回值	无
先决条件	无

LockLevel 可选参数：

表45-75 LockLevel 可选参数

参数	描述
LL_TIM_LOCKLEVEL_OFF	无锁定
LL_TIM_LOCKLEVEL_1	锁定等级 1
LL_TIM_LOCKLEVEL_2	锁定等级 2
LL_TIM_LOCKLEVEL_3	锁定等级 3

45.2.33 函数 LL_TIM_CC_EnableChannel

描述了函数 LL_TIM_CC_EnableChannel

表45-76 函数 LL_TIM_CC_EnableChannel

函数名	LL_TIM_CC_EnableChannel
函数原形	__STATIC_INLINE void LL_TIM_CC_EnableChannel(TIM_TypeDef *TIMx, uint32_t Channels)
功能描述	开启指定通道
输入参数 1	TIMx：定时器实例
输入参数 2	Channels：通道
输出参数	无
返回值	无
先决条件	无

Channels 可选参数:

表45-77 Channels 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH1N	通道 1 互补通道
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH2N	通道 2 互补通道
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH3N	通道 3 互补通道
LL_TIM_CHANNEL_CH4	通道 4

45.2.34 函数 LL_TIM_CC_DisableChannel

描述了函数 LL_TIM_CC_DisableChannel

表45-78 函数 LL_TIM_CC_DisableChannel

函数名	LL_TIM_CC_DisableChannel
函数原形	__STATIC_INLINE void LL_TIM_CC_DisableChannel(TIM_TypeDef *TIMx, uint32_t Channels)
功能描述	关闭指定通道
输入参数 1	TIMx：定时器实例
输入参数 2	Channels：通道
输出参数	无
返回值	无
先决条件	无

Channels 可选参数:

表45-79 Channels 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH1N	通道 1 互补通道
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH2N	通道 2 互补通道
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH3N	通道 3 互补通道
LL_TIM_CHANNEL_CH4	通道 4

45.2.35 函数 LL_TIM_CC_IsEnabledChannel

描述了函数 LL_TIM_CC_IsEnabledChannel

表45-80 函数 LL_TIM_CC_IsEnabledChannel

函数名	LL_TIM_CC_IsEnabledChannel
函数原形	__STATIC_INLINE uint32_t LL_TIM_CC_IsEnabledChannel(TIM_TypeDef *TIMx, uint32_t Channels)
功能描述	检查指定通道是否开启
输入参数 1	TIMx：定时器实例
输入参数 2	Channels：通道
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

Channels 可选参数:

表45-81 Channels 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH1N	通道 1 互补通道
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH2N	通道 2 互补通道
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH3N	通道 3 互补通道
LL_TIM_CHANNEL_CH4	通道 4

45.2.36 函数 LL_TIM_OC_ConfigOutput

描述了函数 LL_TIM_OC_ConfigOutput

表45-82 函数 LL_TIM_OC_ConfigOutput

函数名	LL_TIM_OC_ConfigOutput
函数原形	__STATIC_INLINE void LL_TIM_OC_ConfigOutput(TIM_TypeDef *TIMx, uint32_t Channel, uint32_t Configuration)
功能描述	配置通道为输出
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输入参数 3	Configuration：配置参数
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表45-83 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

Configuration 可选参数：

表45-84 Configuration 可选参数

参数	描述
LL_TIM_OCPOLARITY_HIGH	有效电平为高电平
LL_TIM_OCPOLARITY_LOW	有效电平为低电平
LL_TIM_OCIDLESTATE_LOW	无效电平为低电平
LL_TIM_OCIDLESTATE_HIGH	无效电平为高电平

45.2.37 函数 LL_TIM_OC_SetMode

描述了函数 LL_TIM_OC_SetMode

表45-85 函数 LL_TIM_OC_SetMode

函数名	LL_TIM_OC_SetMode
函数原形	__STATIC_INLINE void LL_TIM_OC_SetMode(TIM_TypeDef *TIMx, uint32_t Channel, uint32_t Mode)
功能描述	设置输出模式
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输入参数 3	Mode：模式
输出参数	无

返回值	无
先决条件	无

Channel 可选参数：

表45-86 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

Mode 可选参数：

表45-87 Mode 可选参数

参数	描述
LL_TIM_OCMODE_FROZEN	冻结，比较值不起作用
LL_TIM_OCMODE_ACTIVE	匹配时配置通道为有效电平
LL_TIM_OCMODE_INACTIVE	匹配时配置通道为无效电平
LL_TIM_OCMODE_TOGGLE	匹配时翻转当前电平
LL_TIM_OCMODE_FORCED_INACTIVE	强制为无效电平
LL_TIM_OCMODE_FORCED_ACTIVE	强制为有效电平
LL_TIM_OCMODE_PWM1	PWM 模式 1
LL_TIM_OCMODE_PWM2	PWM 模式 2

45.2.38 函数 LL_TIM_OC_GetMode

描述了函数 LL_TIM_OC_GetMode

表45-88 函数 LL_TIM_OC_GetMode

函数名	LL_TIM_OC_GetMode
函数原形	__STATIC_INLINE uint32_t LL_TIM_OC_GetMode(TIM_TypeDef *TIMx, uint32_t Channel)
功能描述	获取输出模式
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输出参数	无
返回值	输出模式
先决条件	无

Channel 可选参数：

表45-89 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

45.2.39 函数 LL_TIM_OC_SetPolarity

描述了函数 LL_TIM_OC_SetPolarity

表45-90 函数 LL_TIM_OC_SetPolarity

函数名	LL_TIM_OC_SetPolarity
函数原形	__STATIC_INLINE void LL_TIM_OC_SetPolarity(TIM_TypeDef *TIMx, uint32_t Channel, uint32_t Polarity)
功能描述	设置输出有效极性
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输入参数 3	Polarity：极性
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表45-91 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

Polarity 可选参数：

表45-92 Polarity 可选参数

参数	描述
LL_TIM_OCPOLARITY_HIGH	有效电平为高电平
LL_TIM_OCPOLARITY_LOW	有效电平为低电平

45.2.40 函数 LL_TIM_OC_GetPolarity

描述了函数 LL_TIM_OC_GetPolarity

表45-93 函数 LL_TIM_OC_GetPolarity

函数名	LL_TIM_OC_GetPolarity
函数原形	__STATIC_INLINE uint32_t LL_TIM_OC_GetPolarity(TIM_TypeDef *TIMx, uint32_t Channel)
功能描述	获取输出有效极性
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数:

表45-94 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

45.2.41 函数 LL_TIM_OC_SetIdleState

描述了函数 LL_TIM_OC_SetIdleState

表45-95 函数 LL_TIM_OC_SetIdleState

函数名	LL_TIM_OC_SetIdleState
函数原形	__STATIC_INLINE void LL_TIM_OC_SetIdleState(TIM_TypeDef *TIMx, uint32_t Channel, uint32_t IdleState)
功能描述	设置空闲状态电平
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输入参数 3	IdleState：空闲状态
输出参数	无
返回值	无
先决条件	无

Channels 可选参数:

表45-96 Channels 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3

LL_TIM_CHANNEL_CH4	通道 4

IdleState 可选参数:

表45-97 IdleState 可选参数

参数	描述
LL_TIM_OCIDLESTATE_LOW	空闲电平设置为低电平
LL_TIM_OCIDLESTATE_HIGH	空闲电平设置为高电平

45.2.42 函数 LL_TIM_OC_GetIdleState

描述了函数 LL_TIM_OC_GetIdleState

表45-98 函数 LL_TIM_OC_GetIdleState

函数名	LL_TIM_OC_GetIdleState
函数原形	__STATIC_INLINE uint32_t LL_TIM_OC_GetIdleState(TIM_TypeDef *TIMx, uint32_t Channel)
功能描述	获取空闲状态电平配置
输入参数 1	TIMx：定时器实例
输入参数 2	Channels：通道
输出参数	无
返回值	空闲状态
先决条件	无

Channel 可选参数:

表45-99 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

45.2.43 函数 LL_TIM_OC_EnableFast

描述了函数 LL_TIM_OC_EnableFast

表45-100 函数 LL_TIM_OC_EnableFast

函数名	LL_TIM_OC_EnableFast
函数原形	__STATIC_INLINE void LL_TIM_OC_EnableFast(TIM_TypeDef *TIMx, uint32_t Channel)
功能描述	开启快速模式
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道

输出参数	无
返回值	空闲状态
先决条件	无

Channel 可选参数：

表45-101 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

45.2.44 函数 LL_TIM_OC_DisableFast

描述了函数 LL_TIM_OC_DisableFast

表45-102 函数 LL_TIM_OC_DisableFast

函数名	LL_TIM_OC_DisableFast
函数原形	__STATIC_INLINE void LL_TIM_OC_DisableFast(TIM_TypeDef *TIMx, uint32_t Channel)
功能描述	禁用快速模式
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表45-103 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

45.2.45 函数 LL_TIM_OC_IsEnabledFast

描述了函数 LL_TIM_OC_IsEnabledFast

表45-104 函数 LL_TIM_OC_IsEnabledFast

函数名	LL_TIM_OC_IsEnabledFast
函数原形	__STATIC_INLINE uint32_t LL_TIM_OC_IsEnabledFast(TIM_TypeDef *TIMx,

	uint32_t Channel)
功能描述	检查是否使能快速模式
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

Channel 可选参数：

表45-105 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

45.2.46 函数 LL_TIM_OC_EnablePreload

描述了函数 LL_TIM_OC_EnablePreload

表45-106 函数 LL_TIM_OC_EnablePreload

函数名	LL_TIM_OC_EnablePreload
函数原形	__STATIC_INLINE void LL_TIM_OC_EnablePreload(TIM_TypeDef *TIMx, uint32_t Channel)
功能描述	使能指定通道预装载
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表45-107 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

45.2.47 函数 LL_TIM_OC_DisablePreload

描述了函数 LL_TIM_OC_DisablePreload

表45-108 函数 LL_TIM_OC_DisablePreload

函数名	LL_TIM_OC_DisablePreload
函数原形	__STATIC_INLINE void LL_TIM_OC_DisablePreload(TIM_TypeDef *TIMx, uint32_t Channel)
功能描述	禁用指定通道预装载
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表45-109 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

45.2.48 函数 LL_TIM_OC_IsEnabledPreload

描述了函数 LL_TIM_OC_IsEnabledPreload

表45-110 函数 LL_TIM_OC_IsEnabledPreload

函数名	LL_TIM_OC_IsEnabledPreload
函数原形	__STATIC_INLINE uint32_t LL_TIM_OC_IsEnabledPreload(TIM_TypeDef *TIMx, uint32_t Channel)
功能描述	检查指定通道是否开启预装载
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输出参数	无
返回值	状态（1 或 0）
先决条件	无

Channel 可选参数：

表45-111 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1

LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

45.2.49 函数 LL_TIM_OC_EnableClear

描述了函数 LL_TIM_OC_EnableClear

表45-112 函数 LL_TIM_OC_EnableClear

函数名	LL_TIM_OC_EnableClear
函数原形	__STATIC_INLINE void LL_TIM_OC_EnableClear(TIM_TypeDef *TIMx, uint32_t Channel)
功能描述	使能外部事件清除通道输出功能
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表45-113 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

45.2.50 函数 LL_TIM_OC_DisableClear

描述了函数 LL_TIM_OC_DisableClear

表45-114 函数 LL_TIM_OC_DisableClear

函数名	LL_TIM_OC_DisableClear
函数原形	__STATIC_INLINE void LL_TIM_OC_DisableClear(TIM_TypeDef *TIMx, uint32_t Channel)
功能描述	禁用外部事件清除输出功能
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表45-115 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

45.2.51 函数 LL_TIM_OC_IsEnabledClear

描述了函数 LL_TIM_OC_IsEnabledClear

表45-116 函数 LL_TIM_OC_IsEnabledClear

函数名	LL_TIM_OC_IsEnabledClear
函数原形	__STATIC_INLINE uint32_t LL_TIM_OC_IsEnabledClear(TIM_TypeDef *TIMx, uint32_t Channel)
功能描述	检查是否使能外部事件清除输出功能
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

Channel 可选参数：

表45-117 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

45.2.52 函数 LL_TIM_OC_SetDeadTime

描述了函数 LL_TIM_OC_SetDeadTime

表45-118 函数 LL_TIM_OC_SetDeadTime

函数名	LL_TIM_OC_SetDeadTime
函数原形	__STATIC_INLINE void LL_TIM_OC_SetDeadTime(TIM_TypeDef *TIMx, uint32_t DeadTime)
功能描述	设置死区时间
输入参数 1	TIMx：定时器实例
输入参数 2	DeadTime：死区时间

输出参数	无
返回值	无
先决条件	无

45.2.53 函数 LL_TIM_OC_SetCompareCH1

描述了函数 LL_TIM_OC_SetCompareCH1

表45-119 函数 LL_TIM_OC_SetCompareCH1

函数名	LL_TIM_OC_SetCompareCH1
函数原形	__STATIC_INLINE void LL_TIM_OC_SetCompareCH1(TIM_TypeDef *TIMx, uint32_t CompareValue)
功能描述	设置通道 1 比较值
输入参数 1	TIMx：定时器实例
输入参数 2	CompareValue:比较值
输出参数	无
返回值	无
先决条件	无

45.2.54 函数 LL_TIM_OC_SetCompareCH2

描述了函数 LL_TIM_OC_SetCompareCH2

表45-120 函数 LL_TIM_OC_SetCompareCH2

函数名	LL_TIM_OC_SetCompareCH2
函数原形	__STATIC_INLINE void LL_TIM_OC_SetCompareCH2(TIM_TypeDef *TIMx, uint32_t CompareValue)
功能描述	设置通道 2 比较值
输入参数 1	TIMx：定时器实例
输入参数 2	CompareValue:比较值
输出参数	无
返回值	无
先决条件	无

45.2.55 函数 LL_TIM_OC_SetCompareCH3

描述了函数 LL_TIM_OC_SetCompareCH3

表45-121 函数 LL_TIM_OC_SetCompareCH3

函数名	LL_TIM_OC_SetCompareCH3
函数原形	__STATIC_INLINE void LL_TIM_OC_SetCompareCH3(TIM_TypeDef *TIMx, uint32_t CompareValue)
功能描述	设置通道 3 比较值
输入参数 1	TIMx：定时器实例
输入参数 2	CompareValue:比较值

输出参数	无
返回值	无
先决条件	无

45.2.56 函数 LL_TIM_OC_SetCompareCH4

描述了函数 LL_TIM_OC_SetCompareCH4

表45-122 函数 LL_TIM_OC_SetCompareCH4

函数名	LL_TIM_OC_SetCompareCH4
函数原形	__STATIC_INLINE void LL_TIM_OC_SetCompareCH4(TIM_TypeDef *TIMx, uint32_t CompareValue)
功能描述	设置通道 4 比较值
输入参数 1	TIMx：定时器实例
输入参数 2	CompareValue:比较值
输出参数	无
返回值	无
先决条件	无

45.2.57 函数 LL_TIM_OC_GetCompareCH1

描述了函数 LL_TIM_OC_GetCompareCH1

表45-123 函数 LL_TIM_OC_GetCompareCH1

函数名	LL_TIM_OC_GetCompareCH1
函数原形	__STATIC_INLINE uint32_t LL_TIM_OC_GetCompareCH1(TIM_TypeDef *TIMx)
功能描述	获取通道 1 比较值
输入参数	TIMx：定时器实例
输出参数	无
返回值	比较值
先决条件	无

45.2.58 函数 LL_TIM_OC_GetCompareCH2

描述了函数 LL_TIM_OC_GetCompareCH2

表45-124 函数 LL_TIM_OC_GetCompareCH2

函数名	LL_TIM_OC_GetCompareCH2
函数原形	__STATIC_INLINE uint32_t LL_TIM_OC_GetCompareCH2(TIM_TypeDef *TIMx)
功能描述	获取通道 2 比较值
输入参数	TIMx：定时器实例
输出参数	无
返回值	比较值

先决条件	无

45.2.59 函数 LL_TIM_OC_GetCompareCH3

描述了函数 LL_TIM_OC_GetCompareCH3

表45-125 函数 LL_TIM_OC_GetCompareCH3

函数名	LL_TIM_OC_GetCompareCH3
函数原形	__STATIC_INLINE uint32_t LL_TIM_OC_GetCompareCH3(TIM_TypeDef *TIMx)
功能描述	获取通道 3 比较值
输入参数	TIMx：定时器实例
输出参数	无
返回值	比较值
先决条件	无

45.2.60 函数 LL_TIM_OC_GetCompareCH4

描述了函数 LL_TIM_OC_GetCompareCH4

表45-126 函数 LL_TIM_OC_GetCompareCH4

函数名	LL_TIM_OC_GetCompareCH4
函数原形	__STATIC_INLINE uint32_t LL_TIM_OC_GetCompareCH4(TIM_TypeDef *TIMx)
功能描述	获取通道 4 比较值
输入参数	TIMx：定时器实例
输出参数	无
返回值	比较值
先决条件	无

45.2.61 函数 LL_TIM_IC_Config

描述了函数 LL_TIM_IC_Config

表45-127 函数 LL_TIM_IC_Config

函数名	LL_TIM_IC_Config
函数原形	__STATIC_INLINE void LL_TIM_IC_Config(TIM_TypeDef *TIMx, uint32_t Channel, uint32_t Configuration)
功能描述	配置通道为输入
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输入参数 3	Configuration：配置参数
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表45-128 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

Configuration 可选参数：

表45-129 Configuration 可选参数

参数	描述
LL_TIM_ACTIVEINPUT_DIRECTTI	输入模式，ICx 映射在 Tix 上
LL_TIM_ACTIVEINPUT_INDIRECTTI	输入模式，ICx 映射在 Tiy 上
LL_TIM_ACTIVEINPUT_TRC	输入模式，ICx 映射在 TRC 上
LL_TIM_ICPSC_DIV1	不分频
……	……
LL_TIM_ICPSC_DIV8	8 分频
LL_TIM_IC_FILTER_FDIV1	无滤波值
……	……
LL_TIM_IC_FILTER_FDIV32_N8	采样频率=死区发生器时钟频率 32 分频，采样 8 次
LL_TIM_IC_POLARITY_RISING	上升沿捕获
LL_TIM_IC_POLARITY_FALLING	下降沿捕获
LL_TIM_IC_POLARITY_BOTHEDGE	双边沿捕获

45.2.62 函数 LL_TIM_IC_SetActiveInput

描述了函数 LL_TIM_IC_SetActiveInput

表45-130 函数 LL_TIM_IC_SetActiveInput

函数名	LL_TIM_IC_SetActiveInput
函数原形	__STATIC_INLINE void LL_TIM_IC_SetActiveInput(TIM_TypeDef *TIMx, uint32_t Channel, uint32_t ICActiveInput)
功能描述	设置有效输入
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输入参数 3	ICActiveInput：输入源
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表45-131 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

ICActiveInput 可选参数：

表45-132 ICActiveInput 可选参数

参数	描述
LL_TIM_ACTIVEINPUT_DIRECTTI	输入模式，ICx 映射在 Tix 上
LL_TIM_ACTIVEINPUT_INDIRECTTI	输入模式，ICx 映射在 Tiy 上
LL_TIM_ACTIVEINPUT_TRC	输入模式，ICx 映射在 TRC 上

注意：当 *x=1* 时 *y=2*，当 *x=2* 时 *y=1*，当 *x=3* 时，*y=4*，当 *x=4* 时 *y=3*。

45.2.63 函数 LL_TIM_IC_GetActiveInput

描述了函数 LL_TIM_IC_GetActiveInput

表45-133 函数 LL_TIM_IC_GetActiveInput

函数名	LL_TIM_IC_GetActiveInput
函数原形	__STATIC_INLINE uint32_t LL_TIM_IC_GetActiveInput(TIM_TypeDef *TIMx, uint32_t Channel)
功能描述	获取有效输入
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输出参数	无
返回值	输入源
先决条件	无

Channel 可选参数：

表45-134 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

45.2.64 函数 LL_TIM_IC_SetPrescaler

描述了函数 LL_TIM_IC_SetPrescaler

表45-135 函数 LL_TIM_IC_SetPrescaler

函数名	LL_TIM_IC_SetPrescaler
函数原形	__STATIC_INLINE void LL_TIM_IC_SetPrescaler(TIM_TypeDef *TIMx, uint32_t Channel, uint32_t ICPrescaler)
功能描述	设置输入信号预分频值
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输入参数 3	ICPrescaler：预分频值
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表45-136 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

ICPrescaler 可选参数：

表45-137 ICPrescaler 可选参数

参数	描述
LL_TIM_ICPSC_DIV1	输入信号不分频
LL_TIM_ICPSC_DIV2	输入信号 2 分频
LL_TIM_ICPSC_DIV4	输入信号 4 分频
LL_TIM_ICPSC_DIV8	输入信号 8 分频

45.2.65 函数 LL_TIM_IC_GetPrescaler

描述了函数 LL_TIM_IC_GetPrescaler

表45-138 函数 LL_TIM_IC_GetPrescaler

函数名	LL_TIM_IC_GetPrescaler
函数原形	__STATIC_INLINE uint32_t LL_TIM_IC_GetPrescaler(TIM_TypeDef *TIMx, uint32_t Channel)
功能描述	获取输入信号预分频值
输入参数 1	TIMx：定时器实例

输入参数 2	Channel：通道
输出参数	无
返回值	预分频值
先决条件	无

Channel 可选参数：

表45-139 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

45.2.66 函数 LL_TIM_IC_SetFilter

描述了函数 LL_TIM_IC_SetFilter

表45-140 函数 LL_TIM_IC_SetFilter

函数名	LL_TIM_IC_SetFilter
函数原形	__STATIC_INLINE void LL_TIM_IC_SetFilter(TIM_TypeDef *TIMx, uint32_t Channel, uint32_t ICFilter)
功能描述	设置数字滤波器
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输入参数 3	ICFilter：滤波值
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表45-141 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

ICFilter 可选参数：

表45-142 ICFilter 可选参数

参数	描述

LL_TIM_IC_FILTER_FDIV1	无滤波
LL_TIM_IC_FILTER_FDIV1_N2	采样频率=定时器时钟频率，采样 2 次
LL_TIM_IC_FILTER_FDIV1_N4	采样频率=定时器时钟频率，采样 4 次
LL_TIM_IC_FILTER_FDIV1_N8	采样频率=定时器时钟频率，采样 8 次
LL_TIM_IC_FILTER_FDIV2_N6	采样频率=死区发生器时钟频率 2 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV2_N8	采样频率=死区发生器时钟频率 2 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV4_N6	采样频率=死区发生器时钟频率 4 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV4_N8	采样频率=死区发生器时钟频率 4 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV8_N6	采样频率=死区发生器时钟频率 8 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV8_N8	采样频率=死区发生器时钟频率 8 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV16_N5	采样频率=死区发生器时钟频率 16 分频，采样 5 次
LL_TIM_IC_FILTER_FDIV16_N6	采样频率=死区发生器时钟频率 16 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV16_N8	采样频率=死区发生器时钟频率 16 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV32_N5	采样频率=死区发生器时钟频率 32 分频，采样 5 次
LL_TIM_IC_FILTER_FDIV32_N6	采样频率=死区发生器时钟频率 32 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV32_N8	采样频率=死区发生器时钟频率 32 分频，采样 8 次

45.2.67 函数 LL_TIM_IC_GetFilter

描述了函数 LL_TIM_IC_GetFilter

表45-143 函数 LL_TIM_IC_GetFilter

函数名	LL_TIM_IC_GetFilter
函数原形	__STATIC_INLINE uint32_t LL_TIM_IC_GetFilter(TIM_TypeDef *TIMx, uint32_t Channel)
功能描述	获取滤波值
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输出参数	无
返回值	滤波值
先决条件	无

45.2.68 函数 LL_TIM_IC_SetPolarity

描述了函数 LL_TIM_IC_SetPolarity

表45-144 函数 LL_TIM_IC_SetPolarity

函数名	LL_TIM_IC_SetPolarity
函数原形	__STATIC_INLINE void LL_TIM_IC_SetPolarity(TIM_TypeDef *TIMx, uint32_t Channel, uint32_t ICPolarity)
功能描述	设置输入有效边沿
输入参数 1	TIMx：定时器实例

输入参数 2	Channel：通道
输入参数 3	ICPolarity：有效边沿
输出参数	无
返回值	无
先决条件	无

Channel 可选参数：

表45-145 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

ICPolarity 可选参数：

表45-146 ICPolarity 可选参数

参数	描述
LL_TIM_IC_POLARITY_RISING	上升沿捕获
LL_TIM_IC_POLARITY_FALLING	下降沿捕获
LL_TIM_IC_POLARITY_BOTHEDGE	双边沿捕获

45.2.69 函数 LL_TIM_IC_GetPolarity

描述了函数 LL_TIM_IC_GetPolarity

表45-147 函数 LL_TIM_IC_GetPolarity

函数名	LL_TIM_IC_GetPolarity
函数原形	__STATIC_INLINE uint32_t LL_TIM_IC_GetPolarity(TIM_TypeDef *TIMx, uint32_t Channel)
功能描述	获取输入有效边沿
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输出参数	无
返回值	有效边沿
先决条件	无

Channel 可选参数：

表45-148 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1

LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

45.2.70 函数 LL_TIM_IC_EnableXORCombination

描述了函数 LL_TIM_IC_EnableXORCombination

表45-149 函数

函数名	LL_TIM_IC_EnableXORCombination
函数原形	__STATIC_INLINE void LL_TIM_IC_EnableXORCombination(TIM_TypeDef *TIMx)
功能描述	使能 XOR 输入
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.71 函数 LL_TIM_IC_DisableXORCombination

描述了函数 LL_TIM_IC_DisableXORCombination

表45-150 函数 LL_TIM_IC_DisableXORCombination

函数名	LL_TIM_IC_DisableXORCombination
函数原形	__STATIC_INLINE void LL_TIM_IC_DisableXORCombination(TIM_TypeDef *TIMx)
功能描述	禁用 XOR 输入
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.72 函数 LL_TIM_IC_IsEnabledXORCombination

描述了函数 LL_TIM_IC_IsEnabledXORCombination

表45-151 函数 LL_TIM_IC_IsEnabledXORCombination

函数名	LL_TIM_IC_IsEnabledXORCombination
函数原形	__STATIC_INLINE uint32_t LL_TIM_IC_IsEnabledXORCombination(TIM_TypeDef *TIMx)
功能描述	检查是否开启 XOR 输入
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.73 函数 LL_TIM_IC_GetCaptureCH1

描述了函数 LL_TIM_IC_GetCaptureCH1

表45-152 函数 LL_TIM_IC_GetCaptureCH1

函数名	LL_TIM_IC_GetCaptureCH1
函数原形	__STATIC_INLINE uint32_t LL_TIM_IC_GetCaptureCH1(TIM_TypeDef *TIMx)
功能描述	获取通道 1 的捕获值
输入参数	TIMx：定时器实例
输出参数	无
返回值	通道 1 捕获值
先决条件	无

45.2.74 函数 LL_TIM_IC_GetCaptureCH2

描述了函数 LL_TIM_IC_GetCaptureCH2

表45-153 函数 LL_TIM_IC_GetCaptureCH2

函数名	LL_TIM_IC_GetCaptureCH2
函数原形	__STATIC_INLINE uint32_t LL_TIM_IC_GetCaptureCH2(TIM_TypeDef *TIMx)
功能描述	获取通道 2 的捕获值
输入参数	TIMx：定时器实例
输出参数	无
返回值	通道 2 捕获值
先决条件	无

45.2.75 函数 LL_TIM_IC_GetCaptureCH3

描述了函数 LL_TIM_IC_GetCaptureCH3

表45-154 函数 LL_TIM_IC_GetCaptureCH3

函数名	LL_TIM_IC_GetCaptureCH3
函数原形	__STATIC_INLINE uint32_t LL_TIM_IC_GetCaptureCH3(TIM_TypeDef *TIMx)
功能描述	获取通道 3 的捕获值
输入参数	TIMx：定时器实例
输出参数	无
返回值	通道 3 捕获值
先决条件	无

45.2.76 函数 LL_TIM_IC_GetCaptureCH4

描述了函数 LL_TIM_IC_GetCaptureCH4

表45-155 函数 LL_TIM_IC_GetCaptureCH4

函数名	LL_TIM_IC_GetCaptureCH4
函数原形	__STATIC_INLINE uint32_t LL_TIM_IC_GetCaptureCH4(TIM_TypeDef *TIMx)
功能描述	获取通道 4 的捕获值
输入参数	TIMx：定时器实例
输出参数	无
返回值	通道 4 捕获值
先决条件	无

45.2.77 函数 LL_TIM_EnableExternalClock

描述了函数 LL_TIM_EnableExternalClock

表45-156 函数 LL_TIM_EnableExternalClock

函数名	LL_TIM_EnableExternalClock
函数原形	__STATIC_INLINE void LL_TIM_EnableExternalClock(TIM_TypeDef *TIMx)
功能描述	使能外部时钟
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.78 函数 LL_TIM_DisableExternalClock

描述了函数 LL_TIM_DisableExternalClock

表45-157 函数 LL_TIM_DisableExternalClock

函数名	LL_TIM_DisableExternalClock
函数原形	__STATIC_INLINE void LL_TIM_DisableExternalClock(TIM_TypeDef *TIMx)
功能描述	禁用外部时钟
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.79 函数 LL_TIM_IsEnabledExternalClock

描述了函数 LL_TIM_IsEnabledExternalClock

表45-158 函数 LL_TIM_IsEnabledExternalClock

函数名	LL_TIM_IsEnabledExternalClock
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledExternalClock(TIM_TypeDef *TIMx)
功能描述	检查是否开启外部时钟

输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.80 函数 LL_TIM_SetClockSource

描述了函数 LL_TIM_SetClockSource

表45-159 函数 LL_TIM_SetClockSource

函数名	LL_TIM_SetClockSource
函数原形	__STATIC_INLINE void LL_TIM_SetClockSource(TIM_TypeDef *TIMx, uint32_t ClockSource)
功能描述	设置外部时钟源
输入参数 1	TIMx：定时器实例
输入参数 2	ClockSource：外部时钟源
输出参数	无
返回值	无
先决条件	无

ClockSource 可选参数：

表45-160 ClockSource 可选参数

参数	描述
LL_TIM_CLOCKSOURCE_INTERNAL	不使用外部时钟
LL_TIM_CLOCKSOURCE_EXT_MODE1	外部时钟模式 1
LL_TIM_CLOCKSOURCE_EXT_MODE2	外部时钟模式 2

45.2.81 函数 LL_TIM_SetEncoderMode

描述了函数 LL_TIM_SetEncoderMode

表45-161 函数 LL_TIM_SetEncoderMode

函数名	LL_TIM_SetEncoderMode
函数原形	__STATIC_INLINE void LL_TIM_SetEncoderMode(TIM_TypeDef *TIMx, uint32_t EncoderMode)
功能描述	设置编码器接口模式
输入参数 1	TIMx：定时器实例
输入参数 2	EncoderMode：编码器接口模式
输出参数	无
返回值	无
先决条件	无

EncoderMode 可选参数：

表45-162 EncoderMode 可选参数

参数	描述
LL_TIM_ENCODERMODE_X2_TI1	根据 TI1FP2 的电平，计数器在 TI2FP1 的边沿向上/下计数
LL_TIM_ENCODERMODE_X2_TI2	根据 TI1FP1 的电平，计数器在 TI2FP2 的边沿向上/下计数
LL_TIM_ENCODERMODE_X4_TI12	TI1FP1 和 TI1FP2 互相根据对方的电平同时向上/下计数

45.2.82 函数 LL_TIM_SetTriggerOutput

描述了函数 LL_TIM_SetTriggerOutput

表45-163 函数 LL_TIM_SetTriggerOutput

函数名	LL_TIM_SetTriggerOutput
函数原形	__STATIC_INLINE void LL_TIM_SetTriggerOutput(TIM_TypeDef *TIMx, uint32_t TimerSynchronization)
功能描述	设置主定时器的同步触发输出信号
输入参数 1	TIMx：定时器实例
输入参数 2	TimerSynchronization：主定时器的同步触发输出信号
输出参数	无
返回值	无
先决条件	无

TimerSynchronization 可选参数：

表45-164 TimerSynchronization 可选参数

参数	描述
LL_TIM_TRGO_RESET	复位信号作为触发输出
LL_TIM_TRGO_ENABLE	使能信号作为触发输出
LL_TIM_TRGO_UPDATE	更新事件作为触发输出
LL_TIM_TRGO_CC1IF	捕获/比较事件作为触发输出
LL_TIM_TRGO_OC1REF	OC1REF 信号作为触发输出
LL_TIM_TRGO_OC2REF	OC2REF 信号作为触发输出
LL_TIM_TRGO_OC3REF	OC3REF 信号作为触发输出
LL_TIM_TRGO_OC4REF	OC4REF 信号作为触发输出

45.2.83 函数 LL_TIM_SetSlaveMode

描述了函数 LL_TIM_SetSlaveMode

表45-165 函数 LL_TIM_SetSlaveMode

函数名	LL_TIM_SetSlaveMode

函数原形	__STATIC_INLINE void LL_TIM_SetSlaveMode(TIM_TypeDef *TIMx, uint32_t SlaveMode)
功能描述	设置从定时器的同步方式
输入参数 1	TIMx：定时器实例
输入参数 2	SlaveMode：从定时器的同步方式
输出参数	无
返回值	无
先决条件	无

SlaveMode 可选参数：

表45-166 SlaveMode 可选参数

参数	描述
LL_TIM_SLAVEMODE_DISABLED	关闭从模式
LL_TIM_SLAVEMODE_RESET	复位模式
LL_TIM_SLAVEMODE_GATED	门控模式
LL_TIM_SLAVEMODE_TRIGGER	触发模式

45.2.84 函数 LL_TIM_SetTriggerInput

描述了函数 LL_TIM_SetTriggerInput

表45-167 函数 LL_TIM_SetTriggerInput

函数名	LL_TIM_SetTriggerInput
函数原形	__STATIC_INLINE void LL_TIM_SetTriggerInput(TIM_TypeDef *TIMx, uint32_t TriggerInput)
功能描述	设置同步计数器的触发输入
输入参数 1	TIMx：定时器实例
输入参数 2	TriggerInput：同步计数器的触发输入
输出参数	无
返回值	无
先决条件	无

TriggerInput 可选参数：

表45-168 TriggerInput 可选参数

参数	描述
LL_TIM_TS_ITR0	内部定时器触发 ITR0
LL_TIM_TS_ITR1	内部定时器触发 ITR1
LL_TIM_TS_ITR2	内部定时器触发 ITR2
LL_TIM_TS_ITR3	内部定时器触发 ITR3
LL_TIM_TS_TI1F_ED	TI1 边沿检测器

LL_TIM_TS_TI1FP1	滤波后的定时器输入 1
LL_TIM_TS_TI2FP2	滤波后的定时器输入 2
LL_TIM_TS_ETRF	外部触发输入

45.2.85 函数 LL_TIM_EnableMasterSlaveMode

描述了函数 LL_TIM_EnableMasterSlaveMode

表45-169 函数 LL_TIM_EnableMasterSlaveMode

函数名	LL_TIM_EnableMasterSlaveMode
函数原形	__STATIC_INLINE void LL_TIM_EnableMasterSlaveMode(TIM_TypeDef *TIMx)
功能描述	使能主从模式
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.86 函数 LL_TIM_DisableMasterSlaveMode

描述了函数 LL_TIM_DisableMasterSlaveMode

表45-170 函数 LL_TIM_DisableMasterSlaveMode

函数名	LL_TIM_DisableMasterSlaveMode
函数原形	__STATIC_INLINE void LL_TIM_DisableMasterSlaveMode(TIM_TypeDef *TIMx)
功能描述	禁用主从模式
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.87 函数 LL_TIM_IsEnabledMasterSlaveMode

描述了函数 LL_TIM_IsEnabledMasterSlaveMode

表45-171 函数 LL_TIM_IsEnabledMasterSlaveMode

函数名	LL_TIM_IsEnabledMasterSlaveMode
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledMasterSlaveMode(TIM_TypeDef *TIMx)
功能描述	检查是否开启主从模式
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.88 函数 LL_TIM_ConfigETR

描述了函数 LL_TIM_ConfigETR

表45-172 函数 LL_TIM_ConfigETR

函数名	LL_TIM_ConfigETR
函数原形	__STATIC_INLINE void LL_TIM_ConfigETR(TIM_TypeDef *TIMx, uint32_t ETRPolarity, uint32_t ETRPrescaler, uint32_t ETRFilter)
功能描述	配置外部触发输入
输入参数 1	TIMx：定时器实例
输入参数 2	ETRPolarity：外部输入有效边沿
输入参数 3	ETRPrescaler：外部输入预分频值
输入参数 4	ETRFilter：外部输入滤波值
输出参数	无
返回值	无
先决条件	无

ETRPolarity 可选参数：

表45-173 ETRPolarity 可选参数

参数	描述
LL_TIM_ETR_POLARITY_NONINVERTED	上升沿有效，不反相
LL_TIM_ETR_POLARITY_INVERTED	下降沿有效，反相

ETRPrescaler 可选参数：

表45-174 ETRPrescaler 可选参数

参数	描述
LL_TIM_ETR_PRESCALER_DIV1	不分频
LL_TIM_ETR_PRESCALER_DIV2	2 分频
LL_TIM_ETR_PRESCALER_DIV4	4 分频
LL_TIM_ETR_PRESCALER_DIV8	8 分频

ETRFilter 可选参数：

表45-175 ETRFilter 可选参数

参数	描述
LL_TIM_IC_FILTER_FDIV1	无滤波
LL_TIM_IC_FILTER_FDIV1_N2	采样频率=定时器时钟频率，采样 2 次
LL_TIM_IC_FILTER_FDIV1_N4	采样频率=定时器时钟频率，采样 4 次
LL_TIM_IC_FILTER_FDIV1_N8	采样频率=定时器时钟频率，采样 8 次
LL_TIM_IC_FILTER_FDIV2_N6	采样频率=死区发生器时钟频率 2 分频，采样 6 次

LL_TIM_IC_FILTER_FDIV2_N8	采样频率=死区发生器时钟频率 2 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV4_N6	采样频率=死区发生器时钟频率 4 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV4_N8	采样频率=死区发生器时钟频率 4 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV8_N6	采样频率=死区发生器时钟频率 8 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV8_N8	采样频率=死区发生器时钟频率 8 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV16_N5	采样频率=死区发生器时钟频率 16 分频，采样 5 次
LL_TIM_IC_FILTER_FDIV16_N6	采样频率=死区发生器时钟频率 16 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV16_N8	采样频率=死区发生器时钟频率 16 分频，采样 8 次
LL_TIM_IC_FILTER_FDIV32_N5	采样频率=死区发生器时钟频率 32 分频，采样 5 次
LL_TIM_IC_FILTER_FDIV32_N6	采样频率=死区发生器时钟频率 32 分频，采样 6 次
LL_TIM_IC_FILTER_FDIV32_N8	采样频率=死区发生器时钟频率 32 分频，采样 8 次

45.2.89 函数 LL_TIM_EnableBRK

描述了函数 LL_TIM_EnableBRK

表45-176 函数 LL_TIM_EnableBRK

函数名	LL_TIM_EnableBRK
函数原形	__STATIC_INLINE void LL_TIM_EnableBRK(TIM_TypeDef *TIMx)
功能描述	使能刹车输入
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.90 函数 LL_TIM_DisableBRK

描述了函数 LL_TIM_DisableBRK

表45-177 函数 LL_TIM_DisableBRK

函数名	LL_TIM_DisableBRK
函数原形	__STATIC_INLINE void LL_TIM_DisableBRK(TIM_TypeDef *TIMx)
功能描述	禁用刹车输入
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.91 函数 LL_TIM_ConfigBRK

描述了函数 LL_TIM_ConfigBRK

表45-178 函数 LL_TIM_ConfigBRK

函数名	LL_TIM_ConfigBRK
函数原形	__STATIC_INLINE void LL_TIM_ConfigBRK(TIM_TypeDef *TIMx, uint32_t BreakPolarity)
功能描述	配置刹车输入
输入参数 1	TIMx：定时器实例
输入参数 2	BreakPolarity：刹车有效极性
输出参数	无
返回值	无
先决条件	无

BreakPolarity 可选参数：

表45-179 BreakPolarity 可选参数

参数	描述
LL_TIM_BREAK_POLARITY_LOW	低电平有效
LL_TIM_BREAK_POLARITY_HIGH	高电平有效

45.2.92 函数 LL_TIM_SetOffStates

描述了函数 LL_TIM_SetOffStates

表45-180 函数 LL_TIM_SetOffStates

函数名	LL_TIM_SetOffStates
函数原形	__STATIC_INLINE void LL_TIM_SetOffStates(TIM_TypeDef *TIMx, uint32_t OffStateIdle, uint32_t OffStateRun)
功能描述	设置运行和空闲模式下“关闭状态”
输入参数 1	TIMx：定时器实例
输入参数 2	OffStateIdle：空闲模式下“关闭状态”
输入参数 3	OffStateRun：运行模式下“关闭状态”
输出参数	无
返回值	无
先决条件	无

OffStateIdle 可选参数：

表45-181 OffStateIdle 可选参数

参数	描述
LL_TIM_OSSI_DISABLE	当定时器不工作时禁用 OC/OCN 输出
LL_TIM_OSSI_ENABLE	当定时器不工作时 OC/OCN 输出无效电平

OffStateRun 可选参数：

表45-182 OffStateRun 可选参数

参数	描述
LL_TIM_OSSR_DISABLE	当定时器不工作时禁用 OC/OCN 输出
LL_TIM_OSSR_ENABLE	当定时器不工作时 OC/OCN 输出空闲电平

45.2.93 函数 LL_TIM_EnableAutomaticOutput

描述了函数 LL_TIM_EnableAutomaticOutput

表45-183 函数 LL_TIM_EnableAutomaticOutput

函数名	LL_TIM_EnableAutomaticOutput
函数原形	__STATIC_INLINE void LL_TIM_EnableAutomaticOutput(TIM_TypeDef *TIMx)
功能描述	开启自动输出
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.94 函数 LL_TIM_DisableAutomaticOutput

描述了函数 LL_TIM_DisableAutomaticOutput

表45-184 函数 LL_TIM_DisableAutomaticOutput

函数名	LL_TIM_DisableAutomaticOutput
函数原形	__STATIC_INLINE void LL_TIM_DisableAutomaticOutput(TIM_TypeDef *TIMx)
功能描述	禁用自动输出
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.95 函数 LL_TIM_IsEnabledAutomaticOutput

描述了函数 LL_TIM_IsEnabledAutomaticOutput

表45-185 函数 LL_TIM_IsEnabledAutomaticOutput

函数名	LL_TIM_IsEnabledAutomaticOutput
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledAutomaticOutput(TIM_TypeDef *TIMx)
功能描述	检查自动输出是否开启
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）

先决条件	无

45.2.96 函数 LL_TIM_EnableAllOutputs

描述了函数 LL_TIM_EnableAllOutputs

表45-186 函数 LL_TIM_EnableAllOutputs

函数名	LL_TIM_EnableAllOutputs
函数原形	__STATIC_INLINE void LL_TIM_EnableAllOutputs(TIM_TypeDef *TIMx)
功能描述	使能主输出
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.97 函数 LL_TIM_DisableAllOutputs

描述了函数 LL_TIM_DisableAllOutputs

表45-187 函数 LL_TIM_DisableAllOutputs

函数名	LL_TIM_DisableAllOutputs
函数原形	__STATIC_INLINE void LL_TIM_DisableAllOutputs(TIM_TypeDef *TIMx)
功能描述	禁用主输出
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.98 函数 LL_TIM_IsEnabledAllOutputs

描述了函数 LL_TIM_IsEnabledAllOutputs

表45-188 函数 LL_TIM_IsEnabledAllOutputs

函数名	LL_TIM_IsEnabledAllOutputs
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledAllOutputs(TIM_TypeDef *TIMx)
功能描述	检查主输出是否开启
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.99 函数 LL_TIM_ConfigDMABurst

描述了函数 LL_TIM_ConfigDMABurst

表45-189 函数 LL_TIM_ConfigDMABurst

函数名	LL_TIM_ConfigDMABurst
函数原形	配置 DMA 突发模式
功能描述	__STATIC_INLINE void LL_TIM_ConfigDMABurst(TIM_TypeDef *TIMx, uint32_t DMABurstBaseAddress, uint32_t DMABurstLength)
输入参数 1	TIMx：定时器实例
输入参数 2	DMABurstBaseAddress：传输起始地址
输入参数 3	DMABurstLength：传输数据量
输出参数	无
返回值	无
先决条件	无

DMABurstBaseAddress 可选参数：

表45-190 DMABurstBaseAddress 可选参数

参数	描述
LL_TIM_DMABURST_BASEADDR_CR1	传输起始地址为 CR1 寄存器
LL_TIM_DMABURST_BASEADDR_CR2	传输起始地址为 CR2 寄存器
LL_TIM_DMABURST_BASEADDR_SMCR	传输起始地址为 SMCR 寄存器
LL_TIM_DMABURST_BASEADDR_DIER	传输起始地址为 DIER 寄存器
LL_TIM_DMABURST_BASEADDR_SR	传输起始地址为 SR 寄存器
LL_TIM_DMABURST_BASEADDR_EGR	传输起始地址为 EGR 寄存器
LL_TIM_DMABURST_BASEADDR_CCMR1	传输起始地址为 CCMR1 寄存器
LL_TIM_DMABURST_BASEADDR_CCMR2	传输起始地址为 CCMR2 寄存器
LL_TIM_DMABURST_BASEADDR_CCER	传输起始地址为 CCER 寄存器
LL_TIM_DMABURST_BASEADDR_CNT	传输起始地址为 CNT 寄存器
LL_TIM_DMABURST_BASEADDR_PSC	传输起始地址为 PSC 寄存器
LL_TIM_DMABURST_BASEADDR_ARR	传输起始地址为 ARR 寄存器
LL_TIM_DMABURST_BASEADDR_RCR	传输起始地址为 RCR 寄存器
LL_TIM_DMABURST_BASEADDR_CCR1	传输起始地址为 CCR1 寄存器
LL_TIM_DMABURST_BASEADDR_CCR2	传输起始地址为 CCR2 寄存器
LL_TIM_DMABURST_BASEADDR_CCR3	传输起始地址为 CCR3 寄存器
LL_TIM_DMABURST_BASEADDR_CCR4	传输起始地址为 CCR4 寄存器
LL_TIM_DMABURST_BASEADDR_BDTR	传输起始地址为 BDTR 寄存器

DMABurstLength 可选参数：

表45-191 DMABurstLength 可选参数

参数	描述

LL_TIM_DMABURST_LENGTH_1TRANSFER	传输 1 次
LL_TIM_DMABURST_LENGTH_2TRANSFERS	传输 2 次
LL_TIM_DMABURST_LENGTH_3TRANSFERS	传输 3 次
LL_TIM_DMABURST_LENGTH_4TRANSFERS	传输 4 次
LL_TIM_DMABURST_LENGTH_5TRANSFERS	传输 5 次
LL_TIM_DMABURST_LENGTH_6TRANSFERS	传输 6 次
LL_TIM_DMABURST_LENGTH_7TRANSFERS	传输 7 次
LL_TIM_DMABURST_LENGTH_8TRANSFERS	传输 8 次
LL_TIM_DMABURST_LENGTH_9TRANSFERS	传输 9 次
LL_TIM_DMABURST_LENGTH_10TRANSFERS	传输 10 次
LL_TIM_DMABURST_LENGTH_11TRANSFERS	传输 11 次
LL_TIM_DMABURST_LENGTH_12TRANSFERS	传输 12 次
LL_TIM_DMABURST_LENGTH_13TRANSFERS	传输 13 次
LL_TIM_DMABURST_LENGTH_14TRANSFERS	传输 14 次
LL_TIM_DMABURST_LENGTH_15TRANSFERS	传输 15 次
LL_TIM_DMABURST_LENGTH_16TRANSFERS	传输 16 次
LL_TIM_DMABURST_LENGTH_17TRANSFERS	传输 17 次
LL_TIM_DMABURST_LENGTH_18TRANSFERS	传输 18 次

45.2.100 函数 LL_TIM_SetOCRefClearInputSource

描述了函数 LL_TIM_SetOCRefClearInputSource

表45-192 函数 LL_TIM_SetOCRefClearInputSource

函数名	LL_TIM_SetOCRefClearInputSource
函数原形	__STATIC_INLINE void LL_TIM_SetOCRefClearInputSource(TIM_TypeDef *TIMx, uint32_t OCRefClearInputSource)
功能描述	设置 OCREF 清除输入源
输入参数 1	TIMx：定时器实例
输入参数 2	OCRefClearInputSource：OCREF 清除输入源
输出参数	无
返回值	无
先决条件	无

OCRefClearInputSource 可选参数:

表45-193 OCRefClearInputSource 可选参数

参数	描述
LL_TIM_OCREF_CLR_INT_OCREF_CLR	清除源为 OCREF_CLR
LL_TIM_OCREF_CLR_INT_ETR	清除源为 ETRF

45.2.101 函数 LL_TIM_ClearFlag_UPDATE

描述了函数 LL_TIM_ClearFlag_UPDATE

表45-194 函数 LL_TIM_ClearFlag_UPDATE

函数名	LL_TIM_ClearFlag_UPDATE
函数原形	__STATIC_INLINE void LL_TIM_ClearFlag_UPDATE(TIM_TypeDef *TIMx)
功能描述	清除更新中断标志
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.102 函数 LL_TIM_IsActiveFlag_UPDATE

描述了函数 LL_TIM_IsActiveFlag_UPDATE

表45-195 函数 LL_TIM_IsActiveFlag_UPDATE

函数名	LL_TIM_IsActiveFlag_UPDATE
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_UPDATE(TIM_TypeDef *TIMx)
功能描述	检查是否产生更新中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.103 函数 LL_TIM_ClearFlag_CC1

描述了函数 LL_TIM_ClearFlag_CC1

表45-196 函数 LL_TIM_ClearFlag_CC1

函数名	LL_TIM_ClearFlag_CC1
函数原形	__STATIC_INLINE void LL_TIM_ClearFlag_CC1(TIM_TypeDef *TIMx)
功能描述	清除捕获/比较 1 中断标志
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.104 函数 LL_TIM_IsActiveFlag_CC1

描述了函数 LL_TIM_IsActiveFlag_CC1

表45-197 函数 LL_TIM_IsActiveFlag_CC1

函数名	LL_TIM_IsActiveFlag_CC1
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC1(TIM_TypeDef *TIMx)
功能描述	检查是否产生捕获/比较 1 中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.105 函数 LL_TIM_ClearFlag_CC2

描述了函数 LL_TIM_ClearFlag_CC2

表45-198 函数 LL_TIM_ClearFlag_CC2

函数名	LL_TIM_ClearFlag_CC2
函数原形	__STATIC_INLINE void LL_TIM_ClearFlag_CC2(TIM_TypeDef *TIMx)
功能描述	清除捕获/比较 2 中断标志
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.106 函数 LL_TIM_IsActiveFlag_CC2

描述了函数 LL_TIM_IsActiveFlag_CC2

表45-199 函数 LL_TIM_IsActiveFlag_CC2

函数名	LL_TIM_IsActiveFlag_CC2
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC2(TIM_TypeDef *TIMx)
功能描述	检查是否产生捕获/比较 2 中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.107 函数 LL_TIM_ClearFlag_CC3

描述了函数 LL_TIM_ClearFlag_CC3

表45-200 函数 LL_TIM_ClearFlag_CC3

函数名	LL_TIM_ClearFlag_CC3
函数原形	__STATIC_INLINE void LL_TIM_ClearFlag_CC3(TIM_TypeDef *TIMx)
功能描述	清除捕获/比较 3 中断标志

输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.108 函数 LL_TIM_IsActiveFlag_CC3

描述了函数 LL_TIM_IsActiveFlag_CC3

表45-201 函数 LL_TIM_IsActiveFlag_CC3

函数名	LL_TIM_IsActiveFlag_CC3
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC3(TIM_TypeDef *TIMx)
功能描述	检查是否产生捕获/比较 3 中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.109 函数 LL_TIM_ClearFlag_CC4

描述了函数 LL_TIM_ClearFlag_CC4

表45-202 函数 LL_TIM_ClearFlag_CC4

函数名	LL_TIM_ClearFlag_CC4
函数原形	__STATIC_INLINE void LL_TIM_ClearFlag_CC4(TIM_TypeDef *TIMx)
功能描述	清除捕获/比较 4 中断标志
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.110 函数 LL_TIM_IsActiveFlag_CC4

描述了函数 LL_TIM_IsActiveFlag_CC4

表45-203 函数 LL_TIM_IsActiveFlag_CC4

函数名	LL_TIM_IsActiveFlag_CC4
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC4(TIM_TypeDef *TIMx)
功能描述	检查是否产生捕获/比较 4 中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）

先决条件	无

45.2.111 函数 LL_TIM_ClearFlag_COM

描述了函数 LL_TIM_ClearFlag_COM

表45-204 函数 LL_TIM_ClearFlag_COM

函数名	LL_TIM_ClearFlag_COM
函数原形	__STATIC_INLINE void LL_TIM_ClearFlag_COM(TIM_TypeDef *TIMx)
功能描述	清除换向中断标志
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.112 函数 LL_TIM_IsActiveFlag_COM

描述了函数 LL_TIM_IsActiveFlag_COM

表45-205 函数 LL_TIM_IsActiveFlag_COM

函数名	LL_TIM_IsActiveFlag_COM
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_COM(TIM_TypeDef *TIMx)
功能描述	检查是否产生换向中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.113 函数 LL_TIM_ClearFlag_TRIG

描述了函数 LL_TIM_ClearFlag_TRIG

表45-206 函数 LL_TIM_ClearFlag_TRIG

函数名	LL_TIM_ClearFlag_TRIG
函数原形	__STATIC_INLINE void LL_TIM_ClearFlag_TRIG(TIM_TypeDef *TIMx)
功能描述	清除触发器中断标志
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.114 函数 LL_TIM_IsActiveFlag_TRIG

描述了函数 LL_TIM_IsActiveFlag_TRIG

表45-207 函数 LL_TIM_IsActiveFlag_TRIG

函数名	LL_TIM_IsActiveFlag_TRIG
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_TRIG(TIM_TypeDef *TIMx)
功能描述	检查是否产生触发器中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.115 函数 LL_TIM_ClearFlag_BRK

描述了函数 LL_TIM_ClearFlag_BRK

表45-208 函数 LL_TIM_ClearFlag_BRK

函数名	LL_TIM_ClearFlag_BRK
函数原形	__STATIC_INLINE void LL_TIM_ClearFlag_BRK(TIM_TypeDef *TIMx)
功能描述	清除刹车中断标志
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.116 函数 LL_TIM_IsActiveFlag_BRK

描述了函数 LL_TIM_IsActiveFlag_BRK

表45-209 函数 LL_TIM_IsActiveFlag_BRK

函数名	LL_TIM_IsActiveFlag_BRK
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_BRK(TIM_TypeDef *TIMx)
功能描述	检查是否产生刹车中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.117 函数 LL_TIM_ClearFlag_CC1OVR

描述了函数 LL_TIM_ClearFlag_CC1OVR

表45-210 函数 LL_TIM_ClearFlag_CC1OVR

函数名	LL_TIM_ClearFlag_CC1OVR
函数原形	__STATIC_INLINE void LL_TIM_ClearFlag_CC1OVR(TIM_TypeDef *TIMx)

功能描述	清除捕获/比较 1 过捕获中断标志
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.118 函数 LL_TIM_IsActiveFlag_CC1OVR

描述了函数 LL_TIM_IsActiveFlag_CC1OVR

表45-211 函数 LL_TIM_IsActiveFlag_CC1OVR

函数名	LL_TIM_IsActiveFlag_CC1OVR
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC1OVR(TIM_TypeDef *TIMx)
功能描述	检查是否产生捕获/比较 1 过捕获中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.119 函数 LL_TIM_ClearFlag_CC2OVR

描述了函数 LL_TIM_ClearFlag_CC2OVR

表45-212 函数 LL_TIM_ClearFlag_CC2OVR

函数名	LL_TIM_ClearFlag_CC2OVR
函数原形	__STATIC_INLINE void LL_TIM_ClearFlag_CC2OVR(TIM_TypeDef *TIMx)
功能描述	清除捕获/比较 2 过捕获中断标志
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.120 函数 LL_TIM_IsActiveFlag_CC2OVR

描述了函数 LL_TIM_IsActiveFlag_CC2OVR

表45-213 函数 LL_TIM_IsActiveFlag_CC2OVR

函数名	LL_TIM_IsActiveFlag_CC2OVR
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC2OVR(TIM_TypeDef *TIMx)
功能描述	检查是否产生捕获/比较 2 过捕获中断
输入参数	TIMx：定时器实例
输出参数	无

返回值	状态位（1 或 0）
先决条件	无

45.2.121 函数 LL_TIM_ClearFlag_CC3OVR

描述了函数 LL_TIM_ClearFlag_CC3OVR

表45-214 函数 LL_TIM_ClearFlag_CC3OVR

函数名	LL_TIM_ClearFlag_CC3OVR
函数原形	__STATIC_INLINE void LL_TIM_ClearFlag_CC3OVR(TIM_TypeDef *TIMx)
功能描述	清除捕获/比较 3 过捕获中断标志
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.122 函数 LL_TIM_IsActiveFlag_CC3OVR

描述了函数 LL_TIM_IsActiveFlag_CC3OVR

表45-215 函数 LL_TIM_IsActiveFlag_CC3OVR

函数名	LL_TIM_IsActiveFlag_CC3OVR
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC3OVR(TIM_TypeDef *TIMx)
功能描述	检查是否产生捕获/比较 3 过捕获中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.123 函数 LL_TIM_ClearFlag_CC4OVR

描述了函数 LL_TIM_ClearFlag_CC4OVR

表45-216 函数 LL_TIM_ClearFlag_CC4OVR

函数名	LL_TIM_ClearFlag_CC4OVR
函数原形	__STATIC_INLINE void LL_TIM_ClearFlag_CC4OVR(TIM_TypeDef *TIMx)
功能描述	清除捕获/比较 4 过捕获中断标志
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.124 函数 LL_TIM_IsActiveFlag_CC4OVR

描述了函数 LL_TIM_IsActiveFlag_CC4OVR

表45-217 函数 LL_TIM_IsActiveFlag_CC4OVR

函数名	LL_TIM_IsActiveFlag_CC4OVR
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC4OVR(TIM_TypeDef *TIMx)
功能描述	检查是否产生捕获/比较 4 过捕获中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.125 函数 LL_TIM_EnableIT_UPDATE

描述了函数 LL_TIM_EnableIT_UPDATE

表45-218 函数 LL_TIM_EnableIT_UPDATE

函数名	LL_TIM_EnableIT_UPDATE
函数原形	__STATIC_INLINE void LL_TIM_EnableIT_UPDATE(TIM_TypeDef *TIMx)
功能描述	使能更新中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.126 函数 LL_TIM_DisableIT_UPDATE

描述了函数 LL_TIM_DisableIT_UPDATE

表45-219 函数 LL_TIM_DisableIT_UPDATE

函数名	LL_TIM_DisableIT_UPDATE
函数原形	__STATIC_INLINE void LL_TIM_DisableIT_UPDATE(TIM_TypeDef *TIMx)
功能描述	禁用更新中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.127 函数 LL_TIM_IsEnabledIT_UPDATE

描述了函数 LL_TIM_IsEnabledIT_UPDATE

表45-220 函数 LL_TIM_IsEnabledIT_UPDATE

函数名	LL_TIM_IsEnabledIT_UPDATE
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_UPDATE(TIM_TypeDef *TIMx)
功能描述	检查是否开启更新中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.128 函数 LL_TIM_EnableIT_CC1

描述了函数 LL_TIM_EnableIT_CC1

表45-221 函数 LL_TIM_EnableIT_CC1

函数名	LL_TIM_EnableIT_CC1
函数原形	__STATIC_INLINE void LL_TIM_EnableIT_CC1(TIM_TypeDef *TIMx)
功能描述	使能捕获/比较 1 中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.129 函数 LL_TIM_DisableIT_CC1

描述了函数 LL_TIM_DisableIT_CC1

表45-222 函数 LL_TIM_DisableIT_CC1

函数名	LL_TIM_DisableIT_CC1
函数原形	__STATIC_INLINE void LL_TIM_DisableIT_CC1(TIM_TypeDef *TIMx)
功能描述	禁用捕获/比较 1 中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.130 函数 LL_TIM_IsEnabledIT_CC1

描述了函数 LL_TIM_IsEnabledIT_CC1

表45-223 函数 LL_TIM_IsEnabledIT_CC1

函数名	LL_TIM_IsEnabledIT_CC1
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_CC1(TIM_TypeDef *TIMx)
功能描述	检查是否开启捕获/比较 1 中断

输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.131 函数 LL_TIM_EnableIT_CC2

描述了函数 LL_TIM_EnableIT_CC2

表45-224 函数 LL_TIM_EnableIT_CC2

函数名	LL_TIM_EnableIT_CC2
函数原形	__STATIC_INLINE void LL_TIM_EnableIT_CC2(TIM_TypeDef *TIMx)
功能描述	使能捕获/比较 2 中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.132 函数 LL_TIM_DisableIT_CC2

描述了函数 LL_TIM_DisableIT_CC2

表45-225 函数 LL_TIM_DisableIT_CC2

函数名	LL_TIM_DisableIT_CC2
函数原形	__STATIC_INLINE void LL_TIM_DisableIT_CC2(TIM_TypeDef *TIMx)
功能描述	禁用捕获/比较 2 中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.133 函数 LL_TIM_IsEnabledIT_CC2

描述了函数 LL_TIM_IsEnabledIT_CC2

表45-226 函数 LL_TIM_IsEnabledIT_CC2

函数名	LL_TIM_IsEnabledIT_CC2
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_CC2(TIM_TypeDef *TIMx)
功能描述	检查是否开启捕获/比较 2 中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）

先决条件	无

45.2.134 函数 LL_TIM_EnableIT_CC3

描述了函数 LL_TIM_EnableIT_CC3

表45-227 函数 LL_TIM_EnableIT_CC3

函数名	LL_TIM_EnableIT_CC3
函数原形	__STATIC_INLINE void LL_TIM_EnableIT_CC3(TIM_TypeDef *TIMx)
功能描述	使能捕获/比较 3 中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.135 函数 LL_TIM_DisableIT_CC3

描述了函数 LL_TIM_DisableIT_CC3

表45-228 函数 LL_TIM_DisableIT_CC3

函数名	LL_TIM_DisableIT_CC3
函数原形	__STATIC_INLINE void LL_TIM_DisableIT_CC3(TIM_TypeDef *TIMx)
功能描述	禁用捕获/比较 3 中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.136 函数 LL_TIM_IsEnabledIT_CC3

描述了函数 LL_TIM_IsEnabledIT_CC3

表45-229 函数 LL_TIM_IsEnabledIT_CC3

函数名	LL_TIM_IsEnabledIT_CC3
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_CC3(TIM_TypeDef *TIMx)
功能描述	检查是否开启捕获/比较 3 中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.137 函数 LL_TIM_EnableIT_CC4

描述了函数 LL_TIM_EnableIT_CC4

表45-230 函数 LL_TIM_EnableIT_CC4

函数名	LL_TIM_EnableIT_CC4
函数原形	__STATIC_INLINE void LL_TIM_EnableIT_CC4(TIM_TypeDef *TIMx)
功能描述	使能捕获/比较 4 中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.138 函数 LL_TIM_DisableIT_CC4

描述了函数 LL_TIM_DisableIT_CC4

表45-231 函数 LL_TIM_DisableIT_CC4

函数名	LL_TIM_DisableIT_CC4
函数原形	__STATIC_INLINE void LL_TIM_DisableIT_CC4(TIM_TypeDef *TIMx)
功能描述	禁用捕获/比较 4 中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.139 函数 LL_TIM_IsEnabledIT_CC4

描述了函数 LL_TIM_IsEnabledIT_CC4

表45-232 函数 LL_TIM_IsEnabledIT_CC4

函数名	LL_TIM_IsEnabledIT_CC4
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_CC4(TIM_TypeDef *TIMx)
功能描述	检查是否开启捕获/比较 4 中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.140 函数 LL_TIM_EnableIT_COM

描述了函数 LL_TIM_EnableIT_COM

表45-233 函数 LL_TIM_EnableIT_COM

函数名	LL_TIM_EnableIT_COM
函数原形	__STATIC_INLINE void LL_TIM_EnableIT_COM(TIM_TypeDef *TIMx)

功能描述	使能换向中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.141 函数 LL_TIM_DisableIT_COM

描述了函数 LL_TIM_DisableIT_COM

表45-234 函数 LL_TIM_DisableIT_COM

函数名	LL_TIM_DisableIT_COM
函数原形	__STATIC_INLINE void LL_TIM_DisableIT_COM(TIM_TypeDef *TIMx)
功能描述	禁用换向中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.142 函数 LL_TIM_IsEnabledIT_COM

描述了函数 LL_TIM_IsEnabledIT_COM

表45-235 函数 LL_TIM_IsEnabledIT_COM

函数名	LL_TIM_IsEnabledIT_COM
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_COM(TIM_TypeDef *TIMx)
功能描述	检查是否开启换向中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.143 函数 LL_TIM_EnableIT_TRIG

描述了函数 LL_TIM_EnableIT_TRIG

表45-236 函数 LL_TIM_EnableIT_TRIG

函数名	LL_TIM_EnableIT_TRIG
函数原形	__STATIC_INLINE void LL_TIM_EnableIT_TRIG(TIM_TypeDef *TIMx)
功能描述	使能触发器中断
输入参数	TIMx：定时器实例
输出参数	无

返回值	无
先决条件	无

45.2.144 函数 LL_TIM_DisableIT_TRIG

描述了函数 LL_TIM_DisableIT_TRIG

表45-237 函数 LL_TIM_DisableIT_TRIG

函数名	LL_TIM_DisableIT_TRIG
函数原形	__STATIC_INLINE void LL_TIM_DisableIT_TRIG(TIM_TypeDef *TIMx)
功能描述	禁用触发器中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.145 函数 LL_TIM_IsEnabledIT_TRIG

描述了函数 LL_TIM_IsEnabledIT_TRIG

表45-238 函数 LL_TIM_IsEnabledIT_TRIG

函数名	LL_TIM_IsEnabledIT_TRIG
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_TRIG(TIM_TypeDef *TIMx)
功能描述	检查是否开启触发器中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.146 函数 LL_TIM_EnableIT_BRK

描述了函数 LL_TIM_EnableIT_BRK

表45-239 函数 LL_TIM_EnableIT_BRK

函数名	LL_TIM_EnableIT_BRK
函数原形	__STATIC_INLINE void LL_TIM_EnableIT_BRK(TIM_TypeDef *TIMx)
功能描述	使能刹车中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.147 函数 LL_TIM_DisableIT_BRK

描述了函数 LL_TIM_DisableIT_BRK

表45-240 函数 LL_TIM_DisableIT_BRK

函数名	LL_TIM_DisableIT_BRK
函数原形	__STATIC_INLINE void LL_TIM_DisableIT_BRK(TIM_TypeDef *TIMx)
功能描述	禁用刹车中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.148 函数 LL_TIM_IsEnabledIT_BRK

描述了函数 LL_TIM_IsEnabledIT_BRK

表45-241 函数 LL_TIM_IsEnabledIT_BRK

函数名	LL_TIM_IsEnabledIT_BRK
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_BRK(TIM_TypeDef *TIMx)
功能描述	检查是否开启刹车中断
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.149 函数 LL_TIM_EnableDMAReq_UPDATE

描述了函数 LL_TIM_EnableDMAReq_UPDATE

表45-242 函数 LL_TIM_EnableDMAReq_UPDATE

函数名	LL_TIM_EnableDMAReq_UPDATE
函数原形	__STATIC_INLINE void LL_TIM_EnableDMAReq_UPDATE(TIM_TypeDef *TIMx)
功能描述	使能更新事件 DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.150 函数 LL_TIM_DisableDMAReq_UPDATE

描述了函数 LL_TIM_DisableDMAReq_UPDATE

表45-243 函数 LL_TIM_DisableDMAReq_UPDATE

函数名	LL_TIM_DisableDMAReq_UPDATE
函数原形	__STATIC_INLINE void LL_TIM_DisableDMAReq_UPDATE(TIM_TypeDef *TIMx)
功能描述	禁用更新事件 DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.151 函数 LL_TIM_IsEnabledDMAReq_UPDATE

描述了函数 LL_TIM_IsEnabledDMAReq_UPDATE

表45-244 函数 LL_TIM_IsEnabledDMAReq_UPDATE

函数名	LL_TIM_IsEnabledDMAReq_UPDATE
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledDMAReq_UPDATE(TIM_TypeDef *TIMx)
功能描述	检查是否开启更新事件 DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.152 函数 LL_TIM_EnableDMAReq_CC1

描述了函数 LL_TIM_EnableDMAReq_CC1

表45-245 函数 LL_TIM_EnableDMAReq_CC1

函数名	LL_TIM_EnableDMAReq_CC1
函数原形	__STATIC_INLINE void LL_TIM_EnableDMAReq_CC1(TIM_TypeDef *TIMx)
功能描述	使能捕获/比较 1 事件 DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.153 函数 LL_TIM_DisableDMAReq_CC1

描述了函数 LL_TIM_DisableDMAReq_CC1

表45-246 函数 LL_TIM_DisableDMAReq_CC1

函数名	LL_TIM_DisableDMAReq_CC1
函数原形	__STATIC_INLINE void LL_TIM_DisableDMAReq_CC1(TIM_TypeDef *TIMx)
功能描述	禁用捕获/比较 1 事件 DMA 请求

输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.154 函数 LL_TIM_IsEnabledDMAReq_CC1

描述了函数 LL_TIM_IsEnabledDMAReq_CC1

表45-247 函数 LL_TIM_IsEnabledDMAReq_CC1

函数名	LL_TIM_IsEnabledDMAReq_CC1
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledDMAReq_CC1(TIM_TypeDef *TIMx)
功能描述	检查是否开启捕获/比较 1 事件DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.155 函数 LL_TIM_EnableDMAReq_CC2

描述了函数 LL_TIM_EnableDMAReq_CC2

表45-248 函数 LL_TIM_EnableDMAReq_CC2

函数名	LL_TIM_EnableDMAReq_CC2
函数原形	__STATIC_INLINE void LL_TIM_EnableDMAReq_CC2(TIM_TypeDef *TIMx)
功能描述	使能捕获/比较 2 事件 DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.156 函数 LL_TIM_DisableDMAReq_CC2

描述了函数 LL_TIM_DisableDMAReq_CC2

表45-249 函数 LL_TIM_DisableDMAReq_CC2

函数名	LL_TIM_DisableDMAReq_CC2
函数原形	__STATIC_INLINE void LL_TIM_DisableDMAReq_CC2(TIM_TypeDef *TIMx)
功能描述	禁用捕获/比较 2 事件 DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	无

先决条件	无

45.2.157 函数 LL_TIM_IsEnabledDMAReq_CC2

描述了函数 LL_TIM_IsEnabledDMAReq_CC2

表45-250 函数 LL_TIM_IsEnabledDMAReq_CC2

函数名	LL_TIM_IsEnabledDMAReq_CC2
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledDMAReq_CC2(TIM_TypeDef *TIMx)
功能描述	检查是否开启捕获/比较 2 事件DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.158 函数 LL_TIM_EnableDMAReq_CC3

描述了函数 LL_TIM_EnableDMAReq_CC3

表45-251 函数 LL_TIM_EnableDMAReq_CC3

函数名	LL_TIM_EnableDMAReq_CC3
函数原形	__STATIC_INLINE void LL_TIM_EnableDMAReq_CC3(TIM_TypeDef *TIMx)
功能描述	使能捕获/比较 3 事件 DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.159 函数 LL_TIM_DisableDMAReq_CC3

描述了函数 LL_TIM_DisableDMAReq_CC3

表45-252 函数 LL_TIM_DisableDMAReq_CC3

函数名	LL_TIM_DisableDMAReq_CC3
函数原形	__STATIC_INLINE void LL_TIM_DisableDMAReq_CC3(TIM_TypeDef *TIMx)
功能描述	禁用捕获/比较 3 事件 DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.160 函数 LL_TIM_IsEnabledDMAReq_CC3

描述了函数 LL_TIM_IsEnabledDMAReq_CC3

表45-253 函数 LL_TIM_IsEnabledDMAReq_CC3

函数名	LL_TIM_IsEnabledDMAReq_CC3
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledDMAReq_CC3(TIM_TypeDef *TIMx)
功能描述	检查是否开启捕获/比较 3 事件DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.161 函数 LL_TIM_EnableDMAReq_CC4

描述了函数 LL_TIM_EnableDMAReq_CC4

表45-254 函数 LL_TIM_EnableDMAReq_CC4

函数名	LL_TIM_EnableDMAReq_CC4
函数原形	__STATIC_INLINE void LL_TIM_EnableDMAReq_CC4(TIM_TypeDef *TIMx)
功能描述	使能捕获/比较 4 事件 DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.162 函数 LL_TIM_DisableDMAReq_CC4

描述了函数 LL_TIM_DisableDMAReq_CC4

表45-255 函数 LL_TIM_DisableDMAReq_CC4

函数名	LL_TIM_DisableDMAReq_CC4
函数原形	__STATIC_INLINE void LL_TIM_DisableDMAReq_CC4(TIM_TypeDef *TIMx)
功能描述	禁用捕获/比较 4 事件 DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.163 函数 LL_TIM_IsEnabledDMAReq_CC4

描述了函数 LL_TIM_IsEnabledDMAReq_CC4

表45-256 函数 LL_TIM_IsEnabledDMAReq_CC4

函数名	LL_TIM_IsEnabledDMAReq_CC4
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledDMAReq_CC4(TIM_TypeDef *TIMx)
功能描述	检查是否开启捕获/比较 4 事件DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.164 函数 LL_TIM_EnableDMAReq_COM

描述了函数 LL_TIM_EnableDMAReq_COM

表45-257 函数 LL_TIM_EnableDMAReq_COM

函数名	LL_TIM_EnableDMAReq_COM
函数原形	__STATIC_INLINE void LL_TIM_EnableDMAReq_COM(TIM_TypeDef *TIMx)
功能描述	使能换向事件 DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.165 函数 LL_TIM_DisableDMAReq_COM

描述了函数 LL_TIM_DisableDMAReq_COM

表45-258 函数 LL_TIM_DisableDMAReq_COM

函数名	LL_TIM_DisableDMAReq_COM
函数原形	__STATIC_INLINE void LL_TIM_DisableDMAReq_COM(TIM_TypeDef *TIMx)
功能描述	禁用换向事件 DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.166 函数 LL_TIM_IsEnabledDMAReq_COM

描述了函数 LL_TIM_IsEnabledDMAReq_COM

表45-259 函数 LL_TIM_IsEnabledDMAReq_COM

函数名	LL_TIM_IsEnabledDMAReq_COM
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledDMAReq_COM(TIM_TypeDef *TIMx)
功能描述	检查是否开启换向事件 DMA 请求

输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

45.2.167 函数 LL_TIM_EnableDMAReq_TRIG

描述了函数 LL_TIM_EnableDMAReq_TRIG

表45-260 函数 LL_TIM_EnableDMAReq_TRIG

函数名	LL_TIM_EnableDMAReq_TRIG
函数原形	__STATIC_INLINE void LL_TIM_EnableDMAReq_TRIG(TIM_TypeDef *TIMx)
功能描述	使能触发器事件 DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.168 函数 LL_TIM_DisableDMAReq_TRIG

描述了函数 LL_TIM_DisableDMAReq_TRIG

表45-261 函数 LL_TIM_DisableDMAReq_TRIG

函数名	LL_TIM_DisableDMAReq_TRIG
函数原形	__STATIC_INLINE void LL_TIM_DisableDMAReq_TRIG(TIM_TypeDef *TIMx)
功能描述	禁用触发器事件 DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.169 函数 LL_TIM_IsEnabledDMAReq_TRIG

描述了函数 LL_TIM_IsEnabledDMAReq_TRIG

表45-262 函数 LL_TIM_IsEnabledDMAReq_TRIG

函数名	LL_TIM_IsEnabledDMAReq_TRIG
函数原形	__STATIC_INLINE uint32_t LL_TIM_IsEnabledDMAReq_TRIG(TIM_TypeDef *TIMx)
功能描述	检查是否开启触发器事件 DMA 请求
输入参数	TIMx：定时器实例
输出参数	无
返回值	状态位（1 或 0）

先决条件	无

45.2.170 函数 LL_TIM_GenerateEvent_UPDATE

描述了函数 LL_TIM_GenerateEvent_UPDATE

表45-263 函数 LL_TIM_GenerateEvent_UPDATE

函数名	LL_TIM_GenerateEvent_UPDATE
函数原形	__STATIC_INLINE void LL_TIM_GenerateEvent_UPDATE(TIM_TypeDef *TIMx)
功能描述	产生更新事件
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.171 函数 LL_TIM_GenerateEvent_CC1

描述了函数 LL_TIM_GenerateEvent_CC1

表45-264 函数 LL_TIM_GenerateEvent_CC1

函数名	LL_TIM_GenerateEvent_CC1
函数原形	__STATIC_INLINE void LL_TIM_GenerateEvent_CC1(TIM_TypeDef *TIMx)
功能描述	产生捕获/比较 1 事件
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.172 函数 LL_TIM_GenerateEvent_CC2

描述了函数 LL_TIM_GenerateEvent_CC2

表45-265 函数 LL_TIM_GenerateEvent_CC2

函数名	LL_TIM_GenerateEvent_CC2
函数原形	__STATIC_INLINE void LL_TIM_GenerateEvent_CC2(TIM_TypeDef *TIMx)
功能描述	产生捕获/比较 2 事件
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.173 函数 LL_TIM_GenerateEvent_CC3

描述了函数 LL_TIM_GenerateEvent_CC3

表45-266 函数 LL_TIM_GenerateEvent_CC3

函数名	LL_TIM_GenerateEvent_CC3
函数原形	__STATIC_INLINE void LL_TIM_GenerateEvent_CC3(TIM_TypeDef *TIMx)
功能描述	产生捕获/比较 3 事件
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.174 函数 LL_TIM_GenerateEvent_CC4

描述了函数 LL_TIM_GenerateEvent_CC4

表45-267 函数 LL_TIM_GenerateEvent_CC4

函数名	LL_TIM_GenerateEvent_CC4
函数原形	__STATIC_INLINE void LL_TIM_GenerateEvent_CC4(TIM_TypeDef *TIMx)
功能描述	产生捕获/比较 4 事件
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.175 函数 LL_TIM_GenerateEvent_COM

描述了函数 LL_TIM_GenerateEvent_COM

表45-268 函数 LL_TIM_GenerateEvent_COM

函数名	LL_TIM_GenerateEvent_COM
函数原形	__STATIC_INLINE void LL_TIM_GenerateEvent_COM(TIM_TypeDef *TIMx)
功能描述	产生换向事件
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.176 函数 LL_TIM_GenerateEvent_TRIG

描述了函数 LL_TIM_GenerateEvent_TRIG

表45-269 函数 LL_TIM_GenerateEvent_TRIG

函数名	LL_TIM_GenerateEvent_TRIG
函数原形	__STATIC_INLINE void LL_TIM_GenerateEvent_TRIG(TIM_TypeDef *TIMx)

功能描述	产生触发器事件
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.177 函数 LL_TIM_GenerateEvent_BRK

描述了函数 LL_TIM_GenerateEvent_BRK

表45-270 函数 LL_TIM_GenerateEvent_BRK

函数名	LL_TIM_GenerateEvent_BRK
函数原形	__STATIC_INLINE void LL_TIM_GenerateEvent_BRK(TIM_TypeDef *TIMx)
功能描述	产生刹车事件
输入参数	TIMx：定时器实例
输出参数	无
返回值	无
先决条件	无

45.2.178 函数 LL_TIM_DeInit

描述了函数 LL_TIM_DeInit

表45-271 函数 LL_TIM_DeInit

函数名	LL_TIM_DeInit
函数原形	ErrorStatus LL_TIM_DeInit(TIM_TypeDef *TIMx);
功能描述	将 TIMx 寄存器重设为缺省值
输入参数	TIMx：定时器实例
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

45.2.179 函数 LL_TIM_StructInit

描述了函数 LL_TIM_StructInit

表45-272 函数 LL_TIM_StructInit

函数名	LL_TIM_StructInit
函数原形	void LL_TIM_StructInit(LL_TIM_InitTypeDef *TIM_InitStruct);
功能描述	将时基结构体配置为默认值
输入参数	TIM_InitStruct：时基初始化结构体
输出参数	无
返回值	无

先决条件	无

45.2.180 函数 LL_TIM_Init

描述了函数 LL_TIM_Init

表45-273 函数 LL_TIM_Init

函数名	LL_TIM_Init
函数原形	ErrorStatus LL_TIM_Init(TIM_TypeDef *TIMx, LL_TIM_InitTypeDef *TIM_InitStruct);
功能描述	初始化时基配置
输入参数 1	TIMx：定时器实例
输入参数 2	TIM_InitStruct：时基初始化结构体
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

45.2.181 函数 LL_TIM_OC_StructInit

描述了函数 LL_TIM_OC_StructInit

表45-274 函数 LL_TIM_OC_StructInit

函数名	LL_TIM_OC_StructInit
函数原形	void LL_TIM_OC_StructInit(LL_TIM_OC_InitTypeDef *TIM_OC_InitStruct);
功能描述	将输出比较结构体配置为默认值
输入参数	TIM_OC_InitStruct：输出比较初始化结构体
输出参数	无
返回值	无
先决条件	无

45.2.182 函数 LL_TIM_OC_Init

描述了函数 LL_TIM_OC_Init

表45-275 函数 LL_TIM_OC_Init

函数名	LL_TIM_OC_Init
函数原形	ErrorStatus LL_TIM_OC_Init(TIM_TypeDef *TIMx, uint32_t Channel, LL_TIM_OC_InitTypeDef *TIM_OC_InitStruct);
功能描述	初始化输出比较配置
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输入参数 3	TIM_OC_InitStruct：输出比较结初始化构体
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）

先决条件	无

Channel 可选参数：

表45-276 Channel 可选参数

参数	描述
LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

45.2.183 函数 LL_TIM_IC_StructInit

描述了函数 LL_TIM_IC_StructInit

表45-277 函数 LL_TIM_IC_StructInit

函数名	LL_TIM_IC_StructInit
函数原形	void LL_TIM_IC_StructInit(LL_TIM_IC_InitTypeDef *TIM_ICInitStruct);
功能描述	将输入捕获结构体配置为默认值
输入参数	TIM_ICInitStruct：输入捕获初始化结构体
输出参数	无
返回值	无
先决条件	无

45.2.184 函数 LL_TIM_IC_Init

描述了函数 LL_TIM_IC_Init

表45-278 函数 LL_TIM_IC_Init

函数名	LL_TIM_IC_Init
函数原形	ErrorStatus LL_TIM_IC_Init(TIM_TypeDef *TIMx, uint32_t Channel, LL_TIM_IC_InitTypeDef *TIM_IC_InitStruct);
功能描述	初始化输入捕获配置
输入参数 1	TIMx：定时器实例
输入参数 2	Channel：通道
输入参数 3	TIM_IC_InitStruct：输入捕获初始化结构体
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

Channel 可选参数：

表45-279 Channel 可选参数

参数	描述

LL_TIM_CHANNEL_CH1	通道 1
LL_TIM_CHANNEL_CH2	通道 2
LL_TIM_CHANNEL_CH3	通道 3
LL_TIM_CHANNEL_CH4	通道 4

45.2.185 函数 LL_TIM_ENCODER_StructInit

描述了函数 LL_TIM_ENCODER_StructInit

表45-280 函数 LL_TIM_ENCODER_StructInit

函数名	LL_TIM_ENCODER_StructInit
函数原形	void LL_TIM_ENCODER_StructInit(LL_TIM_ENCODER_InitTypeDef *TIM_EncoderInitStruct);
功能描述	将编码器结构体配置为默认值
输入参数	TIM_EncoderInitStruct：编码器初始化结构体
输出参数	无
返回值	无
先决条件	无

45.2.186 函数 LL_TIM_ENCODER_Init

描述了函数 LL_TIM_ENCODER_Init

表45-281 函数 LL_TIM_ENCODER_Init

函数名	LL_TIM_ENCODER_Init
函数原形	ErrorStatus LL_TIM_ENCODER_Init(TIM_TypeDef *TIMx, LL_TIM_ENCODER_InitTypeDef *TIM_EncoderInitStruct);
功能描述	初始化编码器配置
输入参数 1	TIMx：定时器实例
输入参数 2	TIM_EncoderInitStruct：编码器初始化结构体
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

45.2.187 函数 LL_TIM_HALLSENSOR_StructInit

描述了函数 LL_TIM_HALLSENSOR_StructInit

表45-282 函数 LL_TIM_HALLSENSOR_StructInit

函数名	LL_TIM_HALLSENSOR_StructInit
函数原形	void LL_TIM_HALLSENSOR_StructInit(LL_TIM_HALLSENSOR_InitTypeDef *TIM_HallSensorInitStruct);
功能描述	将霍尔传感器结构体配置为默认值
输入参数	TIM_HallSensorInitStruct：霍尔传感器初始化结构体
输出参数	无

返回值	无
先决条件	无

45.2.188 函数 LL_TIM_HALLSENSOR_Init

描述了函数 LL_TIM_HALLSENSOR_Init

表45-283 函数 LL_TIM_HALLSENSOR_Init

函数名	LL_TIM_HALLSENSOR_Init
函数原形	ErrorStatus LL_TIM_HALLSENSOR_Init(TIM_TypeDef *TIMx, LL_TIM_HALLSENSOR_InitTypeDef *TIM_HallSensorInitStruct);
功能描述	初始化霍尔传感器配置
输入参数 1	TIMx：定时器实例
输入参数 2	TIM_HallSensorInitStruct：霍尔传感器初始化结构体
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

45.2.189 函数 LL_TIM_BDTR_StructInit

描述了函数 LL_TIM_BDTR_StructInit

表45-284 函数 LL_TIM_BDTR_StructInit

函数名	LL_TIM_BDTR_StructInit
函数原形	void LL_TIM_BDTR_StructInit(LL_TIM_BDTR_InitTypeDef *TIM_BDTRInitStruct);
功能描述	将 BDTR 结构体配置为默认值
输入参数	TIM_BDTRInitStruct：BDTR 初始化结构体
输出参数	无
返回值	无
先决条件	无

45.2.190 函数 LL_TIM_BDTR_Init

描述了函数 LL_TIM_BDTR_Init

表45-285 函数 LL_TIM_BDTR_Init

函数名	LL_TIM_BDTR_Init
函数原形	ErrorStatus LL_TIM_BDTR_Init(TIM_TypeDef *TIMx, LL_TIM_BDTR_InitTypeDef *TIM_BDTRInitStruct);
功能描述	初始化 BDTR 配置
输入参数 1	TIMx：
输入参数 2	TIM_BDTRInitStruct：
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）

先决条件	无

46 LL Utils 通用驱动程序（UTILS）

UTILS 包含对 SysTick 定时器、FLASH_LATENCY 和 PLL 的配置，以及获取 UID、FLASH 和 SRAM 大小。

46.1 UTILS 寄存器结构

46.1.1 LL_UTILS_ClkInitTypeDef

LL_UTILS_ClkInitTypeDef，定义于文件“air001xx_ll_utils.h”如下：

```
typedef struct
{
uint32_t AHBCLKDivider;
uint32_t APB1CLKDivider
} LL_UTILS_ClkInitTypeDef;
```

字段说明：

表46-1 LL_UTILS_ClkInitTypeDef 字段说明

字段	描述
AHBCLKDivider	AHB 时钟分频系数
APB1CLKDivider	APB1 时钟分频系数

参数说明：

AHBCLKDivider 可选参数：

表46-2 AHBCLKDivider 可选参数

参数	描述
LL_RCC_SYSCLK_DIV_1	1 分频
LL_RCC_SYSCLK_DIV_2	2 分频
LL_RCC_SYSCLK_DIV_4	4 分频
LL_RCC_SYSCLK_DIV_8	8 分频
LL_RCC_SYSCLK_DIV_16	16 分频
LL_RCC_SYSCLK_DIV_64	64 分频
LL_RCC_SYSCLK_DIV_128	128 分频
LL_RCC_SYSCLK_DIV_256	256 分频
LL_RCC_SYSCLK_DIV_512	512 分频

APB1CLKDivider 可选参数：

表46-3 APB1CLKDivider 可选参数

参数	描述
LL_RCC_APB1_DIV_1	1 分频
LL_RCC_APB1_DIV_2	2 分频
LL_RCC_APB1_DIV_4	4 分频
LL_RCC_APB1_DIV_8	8 分频
LL_RCC_APB1_DIV_16	16 分频

46.2 UTILS 固件库函数

表46-4 UTILS 固件库函数说明

函数名	描述
LL_GetUID_Word0	获取UID 第 1 个字(bit[31:0])
LL_GetUID_Word1	获取UID 第 2 个字(bit[63:32])
LL_GetUID_Word2	获取UID 第 3 个字(bit[95:64])
LL_GetFlashSize	获取 Main Flash 大小
LL_GetSramSize	获取SRAM 大小
LL_InitTick	初始化 SysTick 并启动SysTick
LL_Init1msTick	初始化 SysTick 定时时间为 1ms 并启动SysTick
LL_mDelay	SysTick 毫秒精确延时函数
LL_SetSystemCoreClock	设置全局变量SystecmCoreClock 的值
LL_PLL_ConfigSystemClock_HSI	配置系统时钟为 PLL，并设置 HSI 作为PLL 输入时钟源
LL_PLL_ConfigSystemClock_HSE	配置系统时钟为 PLL，并设置 HSE 作为PLL 输入时钟源
LL_SetFlashLatency	根据HCLK 频率设置 Flash Latency

46.2.1 函数 LL_GetUID_Word0

描述了函数 LL_GetUID_Word0

表46-5 函数 LL_GetUID_Word0

函数名	LL_GetUID_Word0
函数原形	__STATIC_INLINE uint32_t LL_GetUID_Word0(void)
功能描述	获取 UID 第 1 个字(bit[31:0])
输入参数	无
输出参数	无
返回值	UID(bit[31:0])
先决条件	无

46.2.2 函数 LL_GetUID_Word1

描述了函数 LL_GetUID_Word1

表46-6 函数 LL_GetUID_Word1

函数名	LL_GetUID_Word1
函数原形	__STATIC_INLINE uint32_t LL_GetUID_Word1(void)
功能描述	获取 UID 第 2 个字(bit[63:32])
输入参数	无
输出参数	无
返回值	UID(bit[63:32])
先决条件	无

46.2.3 函数 LL_GetUID_Word2

描述了函数 LL_GetUID_Word2

表46-7 函数 LL_GetUID_Word2

函数名	LL_GetUID_Word2
函数原形	__STATIC_INLINE uint32_t LL_GetUID_Word2(void)
功能描述	获取 UID 第 2 个字(bit[96:64])
输入参数	无
输出参数	无
返回值	UID(bit[96:63])
先决条件	无

46.2.4 函数 LL_GetFlashSize

描述了函数 LL_GetFlashSize

表46-8 函数 LL_GetFlashSize

函数名	LL_GetFlashSize
函数原形	__STATIC_INLINE uint32_t LL_GetFlashSize (void)
功能描述	获取 Main Flash 大小
输入参数	无
输出参数	无
返回值	Main Flash 大小
先决条件	无

46.2.5 函数 LL_GetSramSize

描述了函数 LL_GetSramSize

表46-9 函数 LL_GetSramSize

函数名	LL_GetSramSize
函数原形	__STATIC_INLINE uint32_t LL_GetSramSize (void)
功能描述	获取 SRAM 大小
输入参数	无
输出参数	无
返回值	SRAM 大小
先决条件	无

46.2.6 函数 LL_InitTick

描述了函数 LL_InitTick

表46-10　函数 LL_InitTick

函数名	LL_InitTick
函数原形	__STATIC_INLINE void LL_InitTick(uint32_t HCLKFrequency, uint32_t Ticks)
功能描述	初始化 SysTick 及其中断并启动 SysTick
输入参数 1	HCLKFrequency：系统时钟频率
输入参数 2	Ticks：时钟计时数
输出参数	无
返回值	无
先决条件	无

46.2.7 函数 LL_Init1msTick

描述了函数 LL_Init1msTick

表46-11　函数 LL_Init1msTick

函数名	LL_Init1msTick
函数原形	void LL_Init1msTick(uint32_t HCLKFrequency)
功能描述	初始化 SysTick 定时时间为 1ms 并启动 SysTick
输入参数	HCLKFrequency：系统时钟频率
输出参数	无
返回值	无
先决条件	无

46.2.8 函数 LL_mDelay

描述了函数 LL_mDelay

表46-12　函数 LL_mDelay

函数名	LL_mDelay
函数原形	void LL_mDelay(uint32_t Delay)

功能描述	SysTick 毫秒精确延时函数
输入参数	Delay：延时时间(ms)
输出参数	无
返回值	无
先决条件	无

46.2.9 函数 LL_SetSystemCoreClock

描述了函数 LL_SetSystemCoreClock

表46-13 函数 LL_SetSystemCoreClock

函数名	LL_SetSystemCoreClock
函数原形	void LL_SetSystemCoreClock(uint32_t HCLKFrequency)
功能描述	设置全局变量 SystemCoreClock 的值
输入参数	HCLKFrequency：系统时钟频率
输出参数	无
返回值	无
先决条件	无

46.2.10 函数 LL_PLL_ConfigSystemClock_HSI

描述了函数 LL_PLL_ConfigSystemClock_HSI

表46-14 函数 LL_PLL_ConfigSystemClock_HSI

函数名	LL_PLL_ConfigSystemClock_HSI
函数原形	ErrorStatus LL_PLL_ConfigSystemClock_HSI(LL_UTILS_ClkInitTypeDef *UTILS_ClkInitStruct)
功能描述	配置系统时钟为 PLL，并设置 HSI 作为 PLL 输入时钟源
输入参数	UTILS_ClkInitStruct：UTILS 寄存器结构体
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

46.2.11 函数 LL_PLL_ConfigSystemClock_HSE

描述了函数 LL_PLL_ConfigSystemClock_HSE

表46-15 函数 LL_PLL_ConfigSystemClock_HSE

函数名	LL_PLL_ConfigSystemClock_HSE
函数原形	ErrorStatus LL_PLL_ConfigSystemClock_HSE(uint32_t HSEFrequency, uint32_t HSEBypass, LL_UTILS_ClkInitTypeDef *UTILS_ClkInitStruct)
功能描述	配置系统时钟为 PLL，并设置 HSE 作为 PLL 输入时钟源
输入参数 1	HSEFrequency：HSE 时钟频率
输入参数 2	HSEBypass：HSE 是否使用外接晶振

输入参数 3	UTILS_ClkInitStruct：UTILS　寄存器结构体
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

HSEBypass 可选参数：

表46-16　HSEBypass 可选参数

参数	描述
LL_UTILS_HSEBYPASS_ON	HSE 使用外接晶振
LL_UTILS_HSEBYPASS_OFF	HSE 不使用外接晶振

46.2.12 函数 LL_SetFlashLatency

描述了函数 LL_SetFlashLatency

表46-17　函数 LL_SetFlashLatency

函数名	LL_SetFlashLatency
函数原形	ErrorStatus LL_SetFlashLatency(uint32_t HCLKFrequency)
功能描述	根据 HCLK 频率设置 Flash Latency
输入参数	HCLKFrequency：AHB 总线时钟频率
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

47 LL 同步异步收发器通用驱动程序（USART）

USART 能够灵活地与外部设备进行全双工数据交换，满足外部设备对工业标准 NRZ 异步串行数据格式的要求。USART 利用分数波特率发生器提供宽范围的波特率选择。

USART 不仅支持同步单向通信和半双工单线通信，还支持多处理器通信。

47.1 USART 固件驱动寄存器结构

47.1.1 LL_USART_InitTypeDef

LL_USART_InitTypeDef，定义于文件“air001xx_ll_usart.h”如下：

```
typedef struct
{
uint32_t BaudRate;
uint32_t DataWidth;
uint32_t StopBits;
uint32_t Parity;
uint32_t TransferDirection;
uint32_t HardwareFlowControl;
uint32_t OverSampling;
} LL_USART_InitTypeDef;
```

字段说明：

表47-1 LL_USART_InitTypeDef 字段说明

字段	描述
BaudRate	波特率
DataWidth	数据宽度
StopBits	停止位数
Parity	奇偶校验模式
TransferDirection	传输方向
HardwareFlowControl	硬件流控制
OverSampling	过采样

参数说明：

DataWidth 可选参数：

表47-2 DataWidth 可选参数

参数	描述

LL_USART_DATAWIDTH_8B	数据宽度为 8 位
LL_USART_DATAWIDTH_9B	数据宽度为 9 位

StopBits 可选参数：

表47-3 StopBits 可选参数

参数	描述
LL_USART_STOPBITS_1	停止位占 1 位
LL_USART_STOPBITS_2	停止位占 2 位

Parity 可选参数：

表47-4 Parity 可选参数

参数	描述
LL_USART_PARITY_NONE	奇偶校验未使能
LL_USART_PARITY_EVEN	使能奇偶校验，并选择偶校验
LL_USART_PARITY_ODD	使能奇偶校验，并选择奇校验

TransferDirection 可选参数：

表47-5 TransferDirection 可选参数

参数	描述
LL_USART_DIRECTION_NONE	发送和接收均未使能
LL_USART_DIRECTION_RX	使能接收
LL_USART_DIRECTION_TX	使能发送
LL_USART_DIRECTION_TX_RX	使能发送和接收

HardwareFlowControl 可选参数：

表47-6 HardwareFlowControl 可选参数

参数	描述
LL_USART_HWCONTROL_NONE	CTS 和 RTS 硬件流控制已禁用
LL_USART_HWCONTROL_RTS	RTS 使能
LL_USART_HWCONTROL_CTS	CTS 使能
LL_USART_HWCONTROL_RTS_CTS	RTS 和 CTS 使能

OverSampling 可选参数：

表47-7 OverSampling 可选参数

参数	描述
LL_USART_OVERSAMPLING_16	16 倍过采样
LL_USART_OVERSAMPLING_8	8 倍过采样

47.1.2 LL_USART_ClockInitTypeDef

LL_USART_InitTypeDef，定义于文件“air001xx_ll_usart.h”如下：

```
typedef struct
{
uint32_t ClockOutput;
uint32_t ClockPolarity;
uint32_t ClockPhase;
uint32_t LastBitClockPulse;
} LL_USART_ClockInitTypeDef;
```

字段说明：

表47-8 LL_USART_ClockInitTypeDef 字段说明

字段	描述
ClockOutput	指定USART 的时钟输出是否使能
ClockPolarity	时钟极性
ClockPhase	时钟相位
LastBitClockPulse	指定在同步模式下是否必须在 SCLK 引脚上输出与最后传输的数据位 (MSB) 对应的时钟脉冲。

参数说明：

ClockOutput 可选参数：

表47-9 ClockOutput 可选参数

参数	描述
LL_USART_CLOCK_DISABLE	禁用时钟输出
LL_USART_CLOCK_ENABLE	使能时钟输出

ClockPolarity 可选参数：

表47-10 ClockPolarity 可选参数

参数	描述
LL_USART_POLARITY_LOW	传输窗口外 SCLK 引脚上的稳定低电平
LL_USART_POLARITY_HIGH	传输窗口外 SCLK 引脚上的稳定高电平

ClockPhase 可选参数：

表47-11 ClockPhase 可选参数

参数	描述
LL_USART_PHASE_1EDGE	第一个时钟传输是首个数据捕获沿
LL_USART_PHASE_2EDGE	第二个时钟传输是首个数据捕获沿

LastBitClockPulse 可选参数:

表47-12 LastBitClockPulse 可选参数

参数	描述
LL_USART_LASTCLKPULSE_NO_OUTPUT	最后一个数据位的时钟脉冲不输出到 SCLK 引脚
LL_USART_LASTCLKPULSE_OUTPUT	最后一个数据位的时钟脉冲输出到 SCLK 引脚

47.2 USART 固件库函数

表47-13 USART 固件库函数说明

函数名	描述
LL_USART_Enable	使能USART
LL_USART_Disable	禁用USART
LL_USART_IsEnabled	检查USART 是否使能
LL_USART_EnableDirectionRx	使能接收
LL_USART_DisableDirectionRx	禁用接收
LL_USART_EnableDirectionTx	使能发送
LL_USART_DisableDirectionTx	禁用发送
LL_USART_SetTransferDirection	设置传输方向
LL_USART_GetTransferDirection	获取传输方向
LL_USART_SetParity	设置奇偶校验
LL_USART_GetParity	获取奇偶校验
LL_USART_SetWakeUpMethod	设置接收器从静默模式唤醒方法
LL_USART_GetWakeUpMethod	获取接收器从静默模式唤醒方法
LL_USART_SetDataWidth	设置数据宽度
LL_USART_GetDataWidth	获取数据宽度
LL_USART_SetOverSampling	设置过采样倍数
LL_USART_GetOverSampling	获取过采样倍数
LL_USART_SetLastClkPulseOutput	配置最后一个数据位的时钟脉冲是否输出到SCLK 引脚
LL_USART_GetLastClkPulseOutput	获取最后一个数据位输出配置的时钟脉冲
LL_USART_SetClockPhase	设置时钟相位
LL_USART_GetClockPhase	获取时钟相位
LL_USART_SetClockPolarity	设置时钟极性
LL_USART_GetClockPolarity	获取时钟极性

LL_USART_ConfigClock	配置时钟
LL_USART_EnableSCLKOutput	使能SCLK 引脚时钟输出
LL_USART_DisableSCLKOutput	禁用SCLK 引脚时钟输出
LL_USART_IsEnabledSCLKOutput	检查是否使能 SCLK 引脚时钟输出
LL_USART_SetStopBitsLength	设置停止位长度
LL_USART_GetStopBitsLength	获取停止位长度
LL_USART_ConfigCharacter	配置字符帧格式
LL_USART_SetNodeAddress	设置 USART 节点的地址
LL_USART_GetNodeAddress	获取USART 节点的地址
LL_USART_EnableRTSHWFlowCtrl	使能 RTS 硬件流控制
LL_USART_DisableRTSHWFlowCtrl	禁用 RTS 硬件流控制
LL_USART_EnableCTSHWFlowCtrl	使能CTS 硬件流控制
LL_USART_DisableCTSHWFlowCtrl	禁用CTS 硬件流控制
LL_USART_SetHWFlowCtrl	设置硬件流控制
LL_USART_GetHWFlowCtrl	获取硬件流控制
LL_USART_SetBaudRate	设置波特率
LL_USART_GetBaudRate	获取波特率
LL_USART_EnableHalfDuplex	使能半双工
LL_USART_DisableHalfDuplex	禁用半双工
LL_USART_IsEnabledHalfDuplex	检查是否使能半双工
LL_USART_ConfigAsyncMode	配置异步模式
LL_USART_ConfigSyncMode	配置同步模式
LL_USART_ConfigHalfDuplexMode	配置半双工模式
LL_USART_ConfigMultiProcessMode	配置多处理器模式
LL_USART_IsActiveFlag_PE	检查奇偶校验错误标志是否置位
LL_USART_IsActiveFlag_FE	检查帧错误标志是否置位
LL_USART_IsActiveFlag_NE	检查噪音错误标志是否置位
LL_USART_IsActiveFlag_ORE	检查溢出标志是否置位
LL_USART_IsActiveFlag_IDLE	检查空闲帧标志是否置位
LL_USART_IsActiveFlag_RXNE	检查接收寄存器非空标志是否置位
LL_USART_IsActiveFlag_TC	检查发送完成标志是否置位

LL_USART_IsActiveFlag_TXE	检查发送寄存器空标志是否置位
LL_USART_IsActiveFlag_nCTS	检查CTS 标志是否置位
LL_USART_IsActiveFlag_ABRF	检查ABRF 标志是否置位
LL_USART_IsActiveFlag_ABRE	检查ABRE 标志是否置位
LL_USART_IsActiveFlag_SBK	检查断开帧标志是否置位
LL_USART_IsActiveFlag_RWU	检查唤醒标志是否置位
LL_USART_ClearFlag_PE	清除奇偶校验错误标志
LL_USART_ClearFlag_FE	清除帧错误标志
LL_USART_ClearFlag_NE	清除噪音错误标志
LL_USART_ClearFlag_ORE	清除溢出错误标志
LL_USART_ClearFlag_IDLE	清除空闲帧标志
LL_USART_ClearFlag_TC	清除发送完成标志
LL_USART_ClearFlag_RXNE	清除接收数据寄存器非空标志
LL_USART_ClearFlag_nCTS	清除CTS 标志
LL_USART_EnableIT_IDLE	使能空闲帧中断
LL_USART_EnableIT_RXNE	使能接收数据寄存器非空中断
LL_USART_EnableIT_TC	使能发送完成中断
LL_USART_EnableIT_TXE	使能发送数据寄存器空中断
LL_USART_EnableIT_PE	使能奇偶校验错误中断
LL_USART_EnableIT_ERROR	使能错误中断
LL_USART_EnableIT_CTS	使能CTS 中断
LL_USART_DisableIT_IDLE	禁用空闲帧中断
LL_USART_DisableIT_RXNE	禁用接收数据寄存器非空中断
LL_USART_DisableIT_TC	禁用发送完成中断
LL_USART_DisableIT_TXE	禁用发送数据寄存器空中断
LL_USART_DisableIT_PE	禁用奇偶校验错误中断
LL_USART_DisableIT_ERROR	禁用错误中断
LL_USART_DisableIT_CTS	禁用CTS 中断
LL_USART_IsEnabledIT_IDLE	检查是否使能空闲帧中断
LL_USART_IsEnabledIT_RXNE	检查是否使能接收数据寄存器非空中断
LL_USART_IsEnabledIT_TC	检查是否使能发送完成中断

LL_USART_IsEnabledIT_TXE	检查是否使能发送数据寄存器空中断
LL_USART_IsEnabledIT_PE	检查是否使能奇偶校验错误中断
LL_USART_IsEnabledIT_ERROR	检查是否使能错误中断
LL_USART_IsEnabledIT_CTS	检查是否使能 CTS 中断
LL_USART_EnableDMAReq_RX	使能DMA 接收
LL_USART_DisableDMAReq_RX	禁用DMA 接收
LL_USART_IsEnabledDMAReq_RX	检查是否使能DMA 接收
LL_USART_EnableDMAReq_TX	使能DMA 发送
LL_USART_DisableDMAReq_TX	禁用DMA 发送
LL_USART_IsEnabledDMAReq_TX	检查是否使能DMA 发送
LL_USART_DMA_GetRegAddr	获取数据寄存器地址
LL_USART_ReceiveData8	读接收数据寄存器（8 位数据长度）
LL_USART_ReceiveData9	读接收数据寄存器（9 位数据长度）
LL_USART_TransmitData8	写发送数据寄存器（8 位数据长度）
LL_USART_TransmitData9	写发送数据寄存器（9 位数据长度）
LL_USART_RequestBreakSending	发送断开帧
LL_USART_RequestEnterMuteMode	进入静默模式
LL_USART_RequestExitMuteMode	退出静默模式
LL_USART_EnableAutoBaudate	使能自动波特率检测
LL_USART_DisableAutoBaudate	禁用自动波特率检测
LL_USART_IsEnabledAutoBaudate	检查是否使能自动波特率检测
LL_USART_SetAutoBaudateMode	设置自动波特率检测模式
LL_USART_GetAutoBaudateMode	获取自动波特率检测模式
LL_USART_SendAutoBaudateReq	发送自动波特率请求
LL_USART_DeInit	将 USART 配置重设为缺省值
LL_USART_Init	初始化 USART
LL_USART_StructInit	将结构体 LL_USART_InitTypeDef 字段设置为默认值
LL_USART_ClockInit	初始化 USART 时钟
LL_USART_ClockStructInit	将结构体 LL_USART_ClockInitTypeDef 字段设置为默认值

47.2.1 函数 LL_USART_Enable

描述了函数 LL_USART_Enable

表47-14 函数 LL_USART_Enable

函数名	LL_USART_Enable
函数原形	__STATIC_INLINE void LL_USART_Enable(USART_TypeDef *USARTx)
功能描述	使能 USART
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.2 函数 LL_USART_Disable

描述了函数 LL_USART_Disable

表47-15 函数 LL_USART_Disable

函数名	LL_USART_Disable
函数原形	__STATIC_INLINE void LL_USART_Disable(USART_TypeDef *USARTx)
功能描述	禁用 USART
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.3 函数 LL_USART_IsEnabled

描述了函数 LL_USART_IsEnabled

表47-16 函数 LL_USART_IsEnabled

函数名	LL_USART_IsEnabled
函数原形	__STATIC_INLINE uint32_t LL_USART_IsEnabled(USART_TypeDef *USARTx)
功能描述	检测是否使能 USART
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.4 函数 LL_USART_EnableDirectionRx

描述了函数 LL_USART_EnableDirectionRx

表47-17 函数 LL_USART_EnableDirectionRx

函数名	LL_USART_EnableDirectionRx
函数原形	__STATIC_INLINE void LL_USART_EnableDirectionRx(USART_TypeDef *USARTx)
功能描述	使能接收
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.5 函数 LL_USART_DisableDirectionRx

描述了函数 LL_USART_DisableDirectionRx

表47-18 函数 LL_USART_DisableDirectionRx

函数名	LL_USART_DisableDirectionRx
函数原形	__STATIC_INLINE void LL_USART_DisableDirectionRx(USART_TypeDef *USARTx)
功能描述	禁用接收
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.6 函数 LL_USART_EnableDirectionTx

描述了函数 LL_USART_EnableDirectionTx

表47-19 函数 LL_USART_EnableDirectionTx

函数名	LL_USART_EnableDirectionTx
函数原形	__STATIC_INLINE void LL_USART_EnableDirectionTx(USART_TypeDef *USARTx)
功能描述	使能发送
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.7 函数 LL_USART_DisableDirectionTx

描述了函数 LL_USART_DisableDirectionTx

表47-20 函数 LL_USART_DisableDirectionTx

函数名	LL_USART_DisableDirectionTx
函数原形	__STATIC_INLINE void LL_USART_DisableDirectionTx(USART_TypeDef *USARTx)

功能描述	禁用发送
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.8 函数 LL_USART_SetTransferDirection

描述了函数 LL_USART_SetTransferDirection

表47-21 函数 LL_USART_SetTransferDirection

函数名	LL_USART_SetTransferDirection
函数原形	__STATIC_INLINE void LL_USART_SetTransferDirection(USART_TypeDef *USARTx, uint32_t TransferDirection)
功能描述	设置传输方向
输入参数 1	USARTx：USART 实例
输入参数 2	TransferDirection：传输方向
输出参数	无
返回值	无
先决条件	无

TransferDirection 可选参数:

表47-22 TransferDirection 可选参数

参数	描述
LL_USART_DIRECTION_NONE	禁用发送和接收
LL_USART_DIRECTION_RX	使能接收
LL_USART_DIRECTION_TX	使能发送
LL_USART_DIRECTION_TX_RX	使能发送和接收

47.2.9 函数 LL_USART_GetTransferDirection

描述了函数 LL_USART_GetTransferDirection

表47-23 函数 LL_USART_GetTransferDirection

函数名	LL_USART_GetTransferDirection
函数原形	__STATIC_INLINE uint32_t LL_USART_GetTransferDirection(USART_TypeDef *USARTx)
功能描述	获取传输方向
输入参数	USARTx：USART 实例
输出参数	无
返回值	传输方向
先决条件	无

47.2.10 函数 LL_USART_SetParity

描述了函数 LL_USART_SetParity

表47-24 函数 LL_USART_SetParity

函数名	LL_USART_SetParity
函数原形	__STATIC_INLINE void LL_USART_SetParity(USART_TypeDef *USARTx, uint32_t Parity)
功能描述	设置奇偶校验
输入参数 1	USARTx：USART 实例
输入参数 2	Parity：奇偶校验性
输出参数	无
返回值	无
先决条件	无

Parity 可选参数:

表47-25 Parity 可选参数

参数	描述
LL_USART_PARITY_NONE	无校验
LL_USART_PARITY_EVEN	仅使能偶校验
LL_USART_PARITY_ODD	仅使能奇校验

47.2.11 函数 LL_USART_GetParity

描述了函数 LL_USART_GetParity

表47-26 函数 LL_USART_GetParity

函数名	LL_USART_GetParity
函数原形	__STATIC_INLINE uint32_t LL_USART_GetParity(USART_TypeDef *USARTx)
功能描述	获取奇偶校验配置
输入参数	USARTx：USART 实例
输出参数	无
返回值	奇偶校验配置
先决条件	无

47.2.12 函数 LL_USART_SetWakeUpMethod

描述了函数 LL_USART_SetWakeUpMethod

表47-27 函数 LL_USART_SetWakeUpMethod

函数名	LL_USART_SetWakeUpMethod
函数原形	__STATIC_INLINE void LL_USART_SetWakeUpMethod(USART_TypeDef *USARTx, uint32_t Method)
功能描述	设置接收器从静默模式唤醒方法。

输入参数 1	USARTx：USART 实例
输入参数 2	Method：唤醒方式
输出参数	无
返回值	无
先决条件	无

Method 可选参数：

表47-28 Method 可选参数

参数	描述
LL_USART_WAKEUP_IDLELINE	通过空闲帧唤醒
LL_USART_WAKEUP_ADDRESSMARK	通过地址匹配唤醒

47.2.13 函数 LL_USART_GetWakeUpMethod

描述了函数 LL_USART_GetWakeUpMethod

表47-29 函数 LL_USART_GetWakeUpMethod

函数名	LL_USART_GetWakeUpMethod
函数原形	__STATIC_INLINE uint32_t LL_USART_GetWakeUpMethod(USART_TypeDef *USARTx)
功能描述	获取接收器从静默模式唤醒方法
输入参数	USARTx：USART 实例
输出参数	无
返回值	唤醒方法
先决条件	无

47.2.14 函数 LL_USART_SetDataWidth

描述了函数 LL_USART_SetDataWidth

表47-30 函数 LL_USART_SetDataWidth

函数名	LL_USART_SetDataWidth
函数原形	__STATIC_INLINE void LL_USART_SetDataWidth(USART_TypeDef *USARTx, uint32_t DataWidth)
功能描述	设置数据长度
输入参数 1	USARTx：USART 实例
输入参数 2	DataWidth：数据长度
输出参数	无
返回值	无
先决条件	无

DataWidth 可选参数：

表47-31 DataWidth 可选参数

参数	描述
LL_USART_DATAWIDTH_8B	数据长度占 8 位
LL_USART_DATAWIDTH_9B	数据长度占 9 位

47.2.15 函数 LL_USART_GetDataWidth

描述了函数 LL_USART_GetDataWidth

表47-32 函数 LL_USART_GetDataWidth

函数名	LL_USART_GetDataWidth
函数原形	__STATIC_INLINE uint32_t LL_USART_GetDataWidth(USART_TypeDef *USARTx)
功能描述	获取数据长度
输入参数	USARTx：USART 实例
输出参数	无
返回值	数据长度
先决条件	无

47.2.16 函数 LL_USART_SetOverSampling

描述了函数 LL_USART_SetOverSampling

表47-33 函数 LL_USART_SetOverSampling

函数名	LL_USART_SetOverSampling
函数原形	__STATIC_INLINE void LL_USART_SetOverSampling(USART_TypeDef *USARTx, uint32_t OverSampling)
功能描述	设置过采样倍数
输入参数 1	USARTx：USART 实例
输入参数 2	OverSampling：过采样倍数
输出参数	无
返回值	无
先决条件	无

OverSampling 可选参数:

表47-34 OverSampling 可选参数

参数	描述
LL_USART_OVERSAMPLING_16	16 倍过采样
LL_USART_OVERSAMPLING_8	8 倍过采样

47.2.17 函数 LL_USART_GetOverSampling

描述了函数 LL_USART_GetOverSampling

表47-35 函数 LL_USART_GetOverSampling

函数名	LL_USART_GetOverSampling
函数原形	__STATIC_INLINE uint32_t LL_USART_GetOverSampling(USART_TypeDef *USARTx)
功能描述	获取过采样倍数
输入参数	USARTx：USART 实例
输出参数	无
返回值	过采样倍数
先决条件	无

47.2.18 函数 LL_USART_SetLastClkPulseOutput

描述了函数 LL_USART_SetLastClkPulseOutput

表47-36 函数 LL_USART_SetLastClkPulseOutput

函数名	LL_USART_SetLastClkPulseOutput
函数原形	__STATIC_INLINE void LL_USART_SetLastClkPulseOutput(USART_TypeDef *USARTx, uint32_t LastBitClockPulse)
功能描述	配置最后一个数据位的时钟脉冲是否输出到 SCLK 引脚
输入参数 1	USARTx：USART 实例
输入参数 2	LastBitClockPulse：最后一个数据位的时钟脉冲
输出参数	无
返回值	无
先决条件	无

LastBitClockPulse 可选参数:

表47-37 LastBitClockPulse 可选参数

参数	描述
LL_USART_LASTCLKPULSE_NO_OUTPUT	最后一个数据位的时钟脉冲不输出到 SCLK 引脚
LL_USART_LASTCLKPULSE_OUTPUT	最后一个数据位的时钟脉冲输出到 SCLK 引脚

47.2.19 函数 LL_USART_GetLastClkPulseOutput

描述了函数 LL_USART_GetLastClkPulseOutput

表47-38 函数 LL_USART_GetLastClkPulseOutput

函数名	LL_USART_GetLastClkPulseOutput
函数原形	__STATIC_INLINE uint32_t LL_USART_GetLastClkPulseOutput(USART_TypeDef *USARTx)
功能描述	获取最后一个数据位的时钟脉冲是否输出到 SCLK 引脚
输入参数	USARTx：USART 实例
输出参数	无
返回值	最后一个数据位的时钟脉冲
先决条件	无

47.2.20 函数 LL_USART_SetClockPhase

描述了函数 LL_USART_SetClockPhase

表47-39 函数 LL_USART_SetClockPhase

函数名	LL_USART_SetClockPhase
函数原形	_STATIC_INLINE void LL_USART_SetClockPhase(USART_TypeDef *USARTx, uint32_t ClockPhase)
功能描述	选择同步模式下 SCLK 引脚上时钟输出的相位
输入参数 1	USARTx：USART 实例
输入参数 2	ClockPhase：时钟相位
输出参数	无
返回值	无
先决条件	无

ClockPhase 可选参数:

表47-40 ClockPhase 可选参数

参数	描述
LL_USART_PHASE_1EDGE	第一个时钟传输是首个数据捕获沿
LL_USART_PHASE_2EDGE	第二个时钟传输是首个数据捕获沿

47.2.21 函数 LL_USART_GetClockPhase

描述了函数 LL_USART_GetClockPhase

表47-41 函数 LL_USART_GetClockPhase

函数名	LL_USART_GetClockPhase
函数原形	__STATIC_INLINE uint32_t LL_USART_GetClockPhase(USART_TypeDef *USARTx)
功能描述	获取同步模式下 SCLK 引脚上时钟输出的相位
输入参数	USARTx：USART 实例
输出参数	无
返回值	时钟相位
先决条件	无

47.2.22 函数 LL_USART_SetClockPolarity

描述了函数 LL_USART_SetClockPolarity

表47-42 函数 LL_USART_SetClockPolarity

函数名	LL_USART_SetClockPolarity
函数原形	__STATIC_INLINE void LL_USART_SetClockPolarity(USART_TypeDef *USARTx, uint32_t ClockPolarity)
功能描述	选择同步模式下 SCLK 引脚上时钟输出的极性

输入参数 1	USARTx：USART 实例
输入参数 2	ClockPolarity：时钟极性
输出参数	无
返回值	无
先决条件	无

ClockPolarity 可选参数:

表47-43 ClockPolarity 可选参数

参数	描述
LL_USART_POLARITY_LOW	传输窗外，CK pin 为稳定低值
LL_USART_POLARITY_HIGH	传输窗外，CK pin 为稳定高值

47.2.23 函数 LL_USART_GetClockPolarity

描述了函数 LL_USART_GetClockPolarity

表47-44 函数 LL_USART_GetClockPolarity

函数名	LL_USART_GetClockPolarity
函数原形	__STATIC_INLINE uint32_t LL_USART_GetClockPolarity(USART_TypeDef *USARTx)
功能描述	获取同步模式下 SCLK 引脚上时钟输出的极性
输入参数	USARTx：USART 实例
输出参数	无
返回值	时钟极性
先决条件	无

47.2.24 函数 LL_USART_ConfigClock

描述了函数 LL_USART_ConfigClock

表47-45 函数 LL_USART_ConfigClock

函数名	LL_USART_ConfigClock
函数原形	__STATIC_INLINE void LL_USART_ConfigClock(USART_TypeDef *USARTx, uint32_t Phase, uint32_t Polarity, uint32_t LBCPOutput)
功能描述	配置时钟
输入参数 1	USARTx：USART 实例
输入参数 2	Phase：时钟相位
输入参数 3	ClockPolarity：时钟极性
输入参数 4	LBCPOutput：最后一个数据位的时钟脉冲
输出参数	无
返回值	无
先决条件	无

Phase 可选参数:

表47-46 Phase 可选参数

参数	描述
LL_USART_PHASE_1EDGE	第一个时钟传输是首个数据捕获沿
LL_USART_PHASE_2EDGE	第二个时钟传输是首个数据捕获沿

ClockPolarity 可选参数:

表47-47 ClockPolarity 可选参数

参数	描述
LL_USART_POLARITY_LOW	传输窗外，CK pin 为稳定低值
LL_USART_POLARITY_HIGH	传输窗外，CK pin 为稳定高值

LBCPOutput 可选参数:

表47-48 LBCPOutput 可选参数

参数	描述
LL_USART_LASTCLKPULSE_NO_OUTPUT	不输出最后一个数据位的时钟脉冲
LL_USART_LASTCLKPULSE_OUTPUT	输出最后一个数据位的时钟脉冲

47.2.25 函数 LL_USART_EnableSCLKOutput

描述了函数 LL_USART_EnableSCLKOutput

表47-49 函数 LL_USART_EnableSCLKOutput

函数名	LL_USART_EnableSCLKOutput
函数原形	__STATIC_INLINE void LL_USART_EnableSCLKOutput(USART_TypeDef *USARTx)
功能描述	使能时钟输出
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.26 函数 LL_USART_DisableSCLKOutput

描述了函数 LL_USART_DisableSCLKOutput

表47-50 函数 LL_USART_DisableSCLKOutput

函数名	LL_USART_DisableSCLKOutput
函数原形	__STATIC_INLINE void LL_USART_DisableSCLKOutput(USART_TypeDef *USARTx)
功能描述	禁用时钟输出
输入参数 1	USARTx：USART 实例
输出参数	无

返回值	无
先决条件	无

47.2.27 函数 LL_USART_IsEnabledSCLKOutput

描述了函数 LL_USART_IsEnabledSCLKOutput

表47-51 函数 LL_USART_IsEnabledSCLKOutput

函数名	LL_USART_IsEnabledSCLKOutput
函数原形	__STATIC_INLINE uint32_t LL_USART_IsEnabledSCLKOutput(USART_TypeDef *USARTx)
功能描述	检测是否使能时钟输出
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.28 函数 LL_USART_SetStopBitsLength

描述了函数 LL_USART_SetStopBitsLength

表47-52 函数 LL_USART_SetStopBitsLength

函数名	LL_USART_SetStopBitsLength
函数原形	__STATIC_INLINE void LL_USART_SetStopBitsLength(USART_TypeDef *USARTx, uint32_t StopBits)
功能描述	设置停止位的长度
输入参数 1	USARTx：USART 实例
输入参数 2	StopBits：停止位长
输出参数	无
返回值	无
先决条件	无

StopBits 可选参数:

表47-53 StopBits 可选参数

参数	描述
LL_USART_STOPBITS_1	停止位占 1 位
LL_USART_STOPBITS_2	停止位占 2 位

47.2.29 函数 LL_USART_GetStopBitsLength

描述了函数 LL_USART_GetStopBitsLength

表47-54 函数 LL_USART_GetStopBitsLength

函数名	LL_USART_GetStopBitsLength

函数原形	__STATIC_INLINE uint32_t LL_USART_GetStopBitsLength(USART_TypeDef *USARTx)
功能描述	获取停止位长度
输入参数	USARTx：USART 实例
输出参数	无
返回值	停止位长度
先决条件	无

47.2.30 函数 LL_USART_ConfigCharacter

描述了函数 LL_USART_ConfigCharacter

表47-55 函数 LL_USART_ConfigCharacter

函数名	LL_USART_ConfigCharacter
函数原形	__STATIC_INLINE void LL_USART_ConfigCharacter(USART_TypeDef *USARTx, uint32_t DataWidth, uint32_t Parity, uint32_t StopBits)
功能描述	配置字符帧格式
输入参数 1	USARTx：USART 实例
输入参数 2	DataWidth：数据长度
输入参数 3	Parity：奇偶校验性
输入参数 4	StopBits：停止位长度
输出参数	无
返回值	无
先决条件	无

DataWidth 可选参数:

表47-56 DataWidth 可选参数

参数	描述
LL_USART_DATAWIDTH_8B	数据宽度为 8 位
LL_USART_DATAWIDTH_9B	数据宽度为 9 位

Parity 可选参数:

表47-57 Parity 可选参数

参数	描述
LL_USART_PARITY_NONE	奇偶校验未使能
LL_USART_PARITY_EVEN	使能奇偶校验，并选择偶校验
LL_USART_PARITY_ODD	使能奇偶校验，并选择奇校验

StopBits 可选参数:

表47-58 StopBits 可选参数

参数	描述

LL_USART_STOPBITS_1	停止位占 1 位
LL_USART_STOPBITS_2	停止位占 2 位

47.2.31 函数 LL_USART_SetNodeAddress

描述了函数 LL_USART_SetNodeAddress

表47-59 函数 LL_USART_SetNodeAddress

函数名	LL_USART_SetNodeAddress
函数原形	__STATIC_INLINE void LL_USART_SetNodeAddress(USART_TypeDef *USARTx, uint32_t NodeAddress)
功能描述	设置 USART 节点地址
输入参数 1	USARTx：USART 实例
输入参数 2	NodeAddress：USART 节点地址
输出参数	无
返回值	无
先决条件	无

47.2.32 函数 LL_USART_GetNodeAddress

描述了函数 LL_USART_GetNodeAddress

表47-60 函数 LL_USART_GetNodeAddress

函数名	LL_USART_GetNodeAddress
函数原形	__STATIC_INLINE uint32_t LL_USART_GetNodeAddress(USART_TypeDef *USARTx)
功能描述	获取 USART 节点地址
输入参数	USARTx：USART 实例
输出参数	无
返回值	USART 节点地址
先决条件	无

47.2.33 函数 LL_USART_EnableRTSHWFlowCtrl

描述了函数 LL_USART_EnableRTSHWFlowCtrl

表47-61 函数 LL_USART_EnableRTSHWFlowCtrl

函数名	LL_USART_EnableRTSHWFlowCtrl
函数原形	__STATIC_INLINE void LL_USART_EnableRTSHWFlowCtrl(USART_TypeDef *USARTx)
功能描述	使能 RTS 硬件流控制
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.34 函数 LL_USART_DisableRTSHWFlowCtrl

描述了函数 LL_USART_DisableRTSHWFlowCtrl

表47-62 函数 LL_USART_DisableRTSHWFlowCtrl

函数名	LL_USART_DisableRTSHWFlowCtrl
函数原形	__STATIC_INLINE void LL_USART_DisableRTSHWFlowCtrl(USART_TypeDef *USARTx)
功能描述	禁用 RTS 硬件流控制
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.35 函数 LL_USART_EnableCTSHWFlowCtrl

描述了函数 LL_USART_EnableCTSHWFlowCtrl

表47-63 函数 LL_USART_EnableCTSHWFlowCtrl

函数名	LL_USART_EnableCTSHWFlowCtrl
函数原形	__STATIC_INLINE void LL_USART_EnableCTSHWFlowCtrl(USART_TypeDef *USARTx)
功能描述	使能 CTS 硬件流控制
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.36 函数 LL_USART_DisableCTSHWFlowCtrl

描述了函数 LL_USART_DisableCTSHWFlowCtrl

表47-64 函数 LL_USART_DisableCTSHWFlowCtrl

函数名	LL_USART_DisableRTSHWFlowCtrl
函数原形	__STATIC_INLINE void LL_USART_DisableCTSHWFlowCtrl(USART_TypeDef *USARTx)
功能描述	禁用 CTS 硬件流控制
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.37 函数 LL_USART_SetHWFlowCtrl

描述了函数 LL_USART_SetHWFlowCtrl

表47-65 函数 LL_USART_SetHWFlowCtrl

函数名	LL_USART_SetHWFlowCtrl
函数原形	__STATIC_INLINE void LL_USART_SetHWFlowCtrl(USART_TypeDef *USARTx, uint32_t HardwareFlowControl)
功能描述	设置硬件流控制
输入参数 1	USARTx：USART 实例
输入参数 2	HardwareFlowControl：硬件流控制
输出参数	无
返回值	无
先决条件	无

HardwareFlowControl 可选参数:

表47-66 HardwareFlowControl 可选参数

参数	描述
LL_USART_HWCONTROL_NONE	无硬件流控制
LL_USART_HWCONTROL_RTS	使能 RTS 硬件流控制
LL_USART_HWCONTROL_CTS	使能 CTS 硬件流控制
LL_USART_HWCONTROL_RTS_CTS	使能 RTS、CTS 硬件流控制

47.2.38 函数 LL_USART_GetHWFlowCtrl

描述了函数 LL_USART_GetHWFlowCtrl

表47-67 函数 LL_USART_GetHWFlowCtrl

函数名	LL_USART_GetHWFlowCtrl
函数原形	__STATIC_INLINE uint32_t LL_USART_GetHWFlowCtrl(USART_TypeDef *USARTx)
功能描述	获取硬件流控制
输入参数	USARTx：USART 实例
输出参数	无
返回值	硬件流控制
先决条件	无

47.2.39 函数 LL_USART_SetBaudRate

描述了函数 LL_USART_SetBaudRate

表47-68 函数 LL_USART_SetBaudRate

函数名	LL_USART_SetBaudRate
函数原形	__STATIC_INLINE void LL_USART_SetBaudRate(USART_TypeDef *USARTx, uint32_t PeriphClk, uint32_t OverSampling, uint32_t BaudRate)
功能描述	设置波特率
输入参数 1	USARTx：USART 实例

输入参数 2	PeriphClk：外设时钟频率
输入参数 3	OverSampling：过采样倍数
输入参数 4	BaudRate：波特率
输出参数	无
返回值	无
先决条件	无

OverSampling 可选参数:

表47-69 OverSampling 可选参数

参数	描述
LL_USART_OVERSAMPLING_16	16 倍过采样
LL_USART_OVERSAMPLING_8	8 倍过采样

47.2.40 函数 LL_USART_GetBaudRate

描述了函数 LL_USART_GetBaudRate

表47-70 函数 LL_USART_GetBaudRate

函数名	LL_USART_GetBaudRate
函数原形	__STATIC_INLINE uint32_t LL_USART_GetBaudRate(USART_TypeDef *USARTx, uint32_t PeriphClk, uint32_t OverSampling)
功能描述	获取波特率
输入参数 1	USARTx：USART 实例
输入参数 2	PeriphClk：外设时钟频率
输入参数 3	OverSampling：过采样倍数
输出参数	无
返回值	波特率
先决条件	无

OverSampling 可选参数:

表47-71 OverSampling 可选参数

参数	描述
LL_USART_OVERSAMPLING_16	16 倍过采样
LL_USART_OVERSAMPLING_8	8 倍过采样

47.2.41 函数 LL_USART_EnableHalfDuplex

描述了函数 LL_USART_EnableHalfDuplex

表47-72 函数 LL_USART_EnableHalfDuplex

函数名	LL_USART_EnableHalfDuplex
函数原形	__STATIC_INLINE void LL_USART_EnableHalfDuplex(USART_TypeDef *USARTx)

功能描述	使能半双工
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.42 函数 LL_USART_DisableHalfDuplex

描述了函数 LL_USART_DisableHalfDuplex

表47-73 函数 LL_USART_DisableHalfDuplex

函数名	LL_USART_DisableHalfDuplex
函数原形	__STATIC_INLINE void LL_USART_DisableHalfDuplex(USART_TypeDef *USARTx)
功能描述	不使半双工
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.43 函数 LL_USART_IsEnabledHalfDuplex

描述了函数 LL_USART_IsEnabledHalfDuplex

表47-74 函数 LL_USART_IsEnabledHalfDuplex

函数名	LL_USART_IsEnabledHalfDuplex
函数原形	__STATIC_INLINE uint32_t LL_USART_IsEnabledHalfDuplex(USART_TypeDef *USARTx)
功能描述	检查否使能半双工
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.44 函数 LL_USART_ConfigAsyncMode

描述了函数 LL_USART_ConfigAsyncMode

表47-75 函数 LL_USART_ConfigAsyncMode

函数名	LL_USART_ConfigAsyncMode
函数原形	__STATIC_INLINE void LL_USART_ConfigAsyncMode(USART_TypeDef *USARTx)
功能描述	配置异步模式
输入参数	USARTx：USART 实例
输出参数	无

返回值	无
先决条件	无

47.2.45 函数 LL_USART_ConfigSyncMode

描述了函数 LL_USART_ConfigSyncMode

表47-76 函数 LL_USART_ConfigSyncMode

函数名	LL_USART_ConfigSyncMode
函数原形	__STATIC_INLINE void LL_USART_ConfigSyncMode (USART_TypeDef *USARTx)
功能描述	配置同步式
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.46 函数 LL_USART_ConfigHalfDuplexMode

描述了函数 LL_USART_ConfigHalfDuplexMode

表47-77 函数 LL_USART_ConfigHalfDuplexMode

函数名	LL_USART_ConfigHalfDuplexMode
函数原形	__STATIC_INLINE void LL_USART_ConfigHalfDuplexMode (USART_TypeDef *USARTx)
功能描述	配置半双工模式
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.47 函数 LL_USART_ConfigMultiProcessMode

描述了函数 LL_USART_ConfigMultiProcessMode

表47-78 函数 LL_USART_ConfigMultiProcessMode

函数名	LL_USART_ConfigMultiProcessMode
函数原形	__STATIC_INLINE void LL_USART_ConfigMultiProcessMode (USART_TypeDef *USARTx)
功能描述	配置多机通信
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.48 函数 LL_USART_IsActiveFlag_PE

描述了函数 LL_USART_IsActiveFlag_PE

表47-79 函数 LL_USART_IsActiveFlag_PE

函数名	LL_USART_IsActiveFlag_PE
函数原形	__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_PE(USART_TypeDef *USARTx)
功能描述	检查奇偶校验错误标志是否置位
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.49 函数 LL_USART_IsActiveFlag_FE

描述了函数 LL_USART_IsActiveFlag_FE

表47-80 函数 LL_USART_IsActiveFlag_FE

函数名	LL_USART_IsActiveFlag_FE
函数原形	__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_FE(USART_TypeDef *USARTx)
功能描述	检查帧错标志是否置位
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.50 函数 LL_USART_IsActiveFlag_NE

描述了函数 LL_USART_IsActiveFlag_NE

表47-81 函数 LL_USART_IsActiveFlag_NE

函数名	LL_USART_IsActiveFlag_NE
函数原形	__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_NE (USART_TypeDef *USARTx)
功能描述	检查噪音错误标志是否置位
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.51 函数 LL_USART_IsActiveFlag_ORE

描述了函数 LL_USART_IsActiveFlag_ORE

表47-82 函数 LL_USART_IsActiveFlag_ORE

函数名	LL_USART_IsActiveFlag_ORE
函数原形	__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_ORE (USART_TypeDef *USARTx)
功能描述	检查溢出错误标志是否置位
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.52 函数 LL_USART_IsActiveFlag_IDLE

描述了函数 LL_USART_IsActiveFlag_IDLE

表47-83 函数 LL_USART_IsActiveFlag_IDLE

函数名	LL_USART_IsActiveFlag_IDLE
函数原形	__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_IDLE (USART_TypeDef *USARTx)
功能描述	检查空闲帧标志是否置位
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.53 函数 LL_USART_IsActiveFlag_RXNE

描述了函数 LL_USART_IsActiveFlag_RXNE

表47-84 函数 LL_USART_IsActiveFlag_RXNE

函数名	LL_USART_IsActiveFlag_RXNE
函数原形	__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_RXNE (USART_TypeDef *USARTx)
功能描述	检查接收数据非空标志是否置位
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.54 函数 LL_USART_IsActiveFlag_TC

描述了函数 LL_USART_IsActiveFlag_TC

表47-85 函数 LL_USART_IsActiveFlag_TC

函数名	LL_USART_IsActiveFlag_TC
函数原形	__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_TC (USART_TypeDef

	*USARTx)
功能描述	检查发送完成标志是否置位
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.55 函数 LL_USART_IsActiveFlag_TXE

描述了函数 LL_USART_IsActiveFlag_TXE

表47-86 函数 LL_USART_IsActiveFlag_TXE

函数名	LL_USART_IsActiveFlag_TXE
函数原形	__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_TXE (USART_TypeDef *USARTx)
功能描述	检查发送数据寄存器空标志是否置位
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.56 函数 LL_USART_IsActiveFlag_nCTS

描述了函数 LL_USART_IsActiveFlag_nCTS

表47-87 函数 LL_USART_IsActiveFlag_nCTS

函数名	LL_USART_IsActiveFlag_nCTS
函数原形	__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_nCTS (USART_TypeDef *USARTx)
功能描述	检查 CTS 标志是否置位
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.57 函数 LL_USART_IsActiveFlag_ABRF

描述了函数 LL_USART_IsActiveFlag_ABRF

表47-88 函数 LL_USART_IsActiveFlag_ABRF

函数名	LL_USART_IsActiveFlag_ABRF
函数原形	__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_ABRF (USART_TypeDef *USARTx)
功能描述	检查 ABRF 标志是否置位
输入参数	USARTx：USART 实例

输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.58 函数 LL_USART_IsActiveFlag_ABRE

描述了函数 LL_USART_IsActiveFlag_ABRE

表47-89 函数 LL_USART_IsActiveFlag_ABRE

函数名	LL_USART_IsActiveFlag_ABRE
函数原形	__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_ABRE (USART_TypeDef *USARTx)
功能描述	检查 ABRE 标志是否置位
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.59 函数 LL_USART_IsActiveFlag_SBK

描述了函数 LL_USART_IsActiveFlag_SBK

表47-90 函数 LL_USART_IsActiveFlag_SBK

函数名	LL_USART_IsActiveFlag_SBK
函数原形	__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_SBK (USART_TypeDef *USARTx)
功能描述	检查断开帧标志是否置位
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.60 函数 LL_USART_IsActiveFlag_RWU

描述了函数 LL_USART_IsActiveFlag_RWU

表47-91 函数 LL_USART_IsActiveFlag_RWU

函数名	LL_USART_IsActiveFlag_RWU
函数原形	__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_RWU (USART_TypeDef *USARTx)
功能描述	检查唤醒标志是否置位
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.61 函数 LL_USART_ClearFlag_PE

描述了函数 LL_USART_ClearFlag_PE

表47-92 函数 LL_USART_ClearFlag_PE

函数名	LL_USART_ClearFlag_PE
函数原形	__STATIC_INLINE void LL_USART_ClearFlag_PE (USART_TypeDef *USARTx)
功能描述	清除奇偶校验错误标志
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.62 函数 LL_USART_ClearFlag_FE

描述了函数 LL_USART_ClearFlag_FE

表47-93 函数 LL_USART_ClearFlag_FE

函数名	LL_USART_ClearFlag_FE
函数原形	__STATIC_INLINE void LL_USART_ClearFlag_FE (USART_TypeDef *USARTx)
功能描述	清除帧错误标志
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.63 函数 LL_USART_ClearFlag_NE

描述了函数 LL_USART_ClearFlag_NE

表47-94 函数 LL_USART_ClearFlag_NE

函数名	LL_USART_ClearFlag_NE
函数原形	__STATIC_INLINE void LL_USART_ClearFlag_NE (USART_TypeDef *USARTx)
功能描述	清除噪音错误标志
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.64 函数 LL_USART_ClearFlag_ORE

描述了函数 LL_USART_ClearFlag_ORE

表47-95 函数 LL_USART_ClearFlag_ORE

函数名	LL_USART_ClearFlag_ORE
函数原形	__STATIC_INLINE void LL_USART_ClearFlag_ORE (USART_TypeDef *USARTx)
功能描述	清除溢出错误标志
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.65 函数 LL_USART_ClearFlag_IDLE

描述了函数 LL_USART_ClearFlag_IDLE

表47-96 函数 LL_USART_ClearFlag_IDLE

函数名	LL_USART_ClearFlag_IDLE
函数原形	__STATIC_INLINE void LL_USART_ClearFlag_IDLE (USART_TypeDef *USARTx)
功能描述	清除空闲帧标志
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.66 函数 LL_USART_ClearFlag_TC

描述了函数 LL_USART_ClearFlag_TC

表47-97 函数 LL_USART_ClearFlag_TC

函数名	LL_USART_ClearFlag_TC
函数原形	__STATIC_INLINE void LL_USART_ClearFlag_TC (USART_TypeDef *USARTx)
功能描述	清除发送完成标志
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.67 函数 LL_USART_ClearFlag_RXNE

描述了函数 LL_USART_ClearFlag_RXNE

表47-98 函数 LL_USART_ClearFlag_RXNE

函数名	LL_USART_ClearFlag_RXNE
函数原形	__STATIC_INLINE void LL_USART_ClearFlag_RXNE (USART_TypeDef *USARTx)

功能描述	清除接收数据寄存器非空标志
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.68 函数 LL_USART_ClearFlag_nCTS

描述了函数 LL_USART_ClearFlag_nCTS

表47-99　函数 LL_USART_ClearFlag_nCTS

函数名	LL_USART_ClearFlag_nCTS
函数原形	__STATIC_INLINE void　LL_USART_ClearFlag_nCTS (USART_TypeDef *USARTx)
功能描述	清除 CTS 标志
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.69 函数 LL_USART_EnableIT_IDLE

描述了函数 LL_USART_EnableIT_IDLE

表47-100 函数 LL_USART_EnableIT_IDLE

函数名	LL_USART_EnableIT_IDLE
函数原形	__STATIC_INLINE void LL_USART_EnableIT_IDLE (USART_TypeDef *USARTx)
功能描述	使能空闲帧中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.70 函数 LL_USART_EnableIT_RXNE

描述了函数 LL_USART_EnableIT_RXNE

表47-101 函数 LL_USART_EnableIT_RXNE

函数名	LL_USART_EnableIT_RXNE
函数原形	__STATIC_INLINE void LL_USART_EnableIT_RXNE (USART_TypeDef *USARTx)
功能描述	使能接收数据寄存器非空中断
输入参数	USARTx：USART 实例
输出参数	无

返回值	无
先决条件	无

47.2.71 函数 LL_USART_EnableIT_TC

描述了函数 LL_USART_EnableIT_TC

表47-102 函数 LL_USART_EnableIT_TC

函数名	LL_USART_EnableIT_TC
函数原形	__STATIC_INLINE void LL_USART_EnableIT_TC (USART_TypeDef *USARTx)
功能描述	使能发送完成中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.72 函数 LL_USART_EnableIT_TXE

描述了函数 LL_USART_EnableIT_TXE

表47-103 函数 LL_USART_EnableIT_TXE

函数名	LL_USART_EnableIT_TXE
函数原形	__STATIC_INLINE void LL_USART_EnableIT_TXE (USART_TypeDef *USARTx)
功能描述	使能发送数据寄存器空中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.73 函数 LL_USART_EnableIT_PE

描述了函数 LL_USART_EnableIT_PE

表47-104 函数 LL_USART_EnableIT_PE

函数名	LL_USART_EnableIT_PE
函数原形	__STATIC_INLINE void LL_USART_EnableIT_PE (USART_TypeDef *USARTx)
功能描述	使能奇偶校验错误中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.74 函数 LL_USART_EnableIT_ERROR

描述了函数 LL_USART_EnableIT_ERROR

表47-105 函数 LL_USART_EnableIT_ERROR

函数名	LL_USART_EnableIT_ERROR
函数原形	__STATIC_INLINE void LL_USART_EnableIT_ERROR (USART_TypeDef *USARTx)
功能描述	使能错误中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.75 函数 LL_USART_EnableIT_CTS

描述了函数 LL_USART_EnableIT_CTS

表47-106 函数 LL_USART_EnableIT_CTS

函数名	LL_USART_EnableIT_CTS
函数原形	__STATIC_INLINE void LL_USART_EnableIT_CTS (USART_TypeDef *USARTx)
功能描述	使能 CTS 中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.76 函数 LL_USART_DisableIT_IDLE

描述了函数 LL_USART_DisableIT_IDLE

表47-107 函数 LL_USART_DisableIT_IDLE

函数名	LL_USART_DisableIT_IDLE
函数原形	__STATIC_INLINE void LL_USART_DisableIT_IDLE (USART_TypeDef *USARTx)
功能描述	禁用空闲帧中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.77 函数 LL_USART_DisableIT_RXNE

描述了函数 LL_USART_DisableIT_RXNE

表47-108 函数 LL_USART_DisableIT_RXNE

函数名	LL_USART_DisableIT_RXNE
函数原形	__STATIC_INLINE void LL_USART_DisableIT_RXNE (USART_TypeDef *USARTx)
功能描述	禁用接收数据寄存器非空中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.78 函数 LL_USART_DisableIT_TC

描述了函数 LL_USART_DisableIT_TC

表47-109 函数 LL_USART_DisableIT_TC

函数名	LL_USART_DisableIT_TC
函数原形	__STATIC_INLINE void LL_USART_DisableIT_TC (USART_TypeDef *USARTx)
功能描述	禁用发送完成中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.79 函数 LL_USART_DisableIT_TXE

描述了函数 LL_USART_DisableIT_TXE

表47-110 函数 LL_USART_DisableIT_TXE

函数名	LL_USART_DisableIT_TXE
函数原形	__STATIC_INLINE void LL_USART_DisableIT_TXE (USART_TypeDef *USARTx)
功能描述	禁用发送数据寄存器空中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.80 函数 LL_USART_DisableIT_PE

描述了函数 LL_USART_DisableIT_PE

表47-111 函数 LL_USART_DisableIT_PE

函数名	LL_USART_DisableIT_PE
函数原形	__STATIC_INLINE void LL_USART_DisableIT_PE (USART_TypeDef *USARTx)

功能描述	禁用奇偶校验错误中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.81 函数 LL_USART_DisableIT_ERROR

描述了函数 LL_USART_DisableIT_ERROR

表47-112 函数 LL_USART_DisableIT_ERROR

函数名	LL_USART_DisableIT_ERROR
函数原形	__STATIC_INLINE void LL_USART_DisableIT_ERROR (USART_TypeDef *USARTx)
功能描述	禁用错误中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.82 函数 LL_USART_DisableIT_CTS

描述了函数 LL_USART_DisableIT_CTS

表47-113 函数 LL_USART_DisableIT_CTS

函数名	LL_USART_DisableIT_CTS
函数原形	__STATIC_INLINE void LL_USART_DisableIT_CTS (USART_TypeDef *USARTx)
功能描述	禁用 CTS 中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.83 函数 LL_USART_IsEnabledIT_IDLE

描述了函数 LL_USART_IsEnabledIT_IDLE

表47-114 函数 LL_USART_IsEnabledIT_IDLE

函数名	LL_USART_IsEnabledIT_IDLE
函数原形	__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_IDLE (USART_TypeDef *USARTx)
功能描述	检查是否使能空闲帧中断
输入参数	USARTx：USART 实例
输出参数	无

返回值	状态位（1 或 0）
先决条件	无

47.2.84 函数 LL_USART_IsEnabledIT_RXNE

描述了函数 LL_USART_IsEnabledIT_RXNE

表47-115 函数 LL_USART_IsEnabledIT_RXNE

函数名	LL_USART_IsEnabledIT_RXNE
函数原形	__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_RXNE (USART_TypeDef *USARTx)
功能描述	检查是否使能接收数据寄存器非空中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.85 函数 LL_USART_IsEnabledIT_TC

描述了函数 LL_USART_IsEnabledIT_TC

表47-116 函数 LL_USART_IsEnabledIT_TC

函数名	LL_USART_IsEnabledIT_TC
函数原形	__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_TC (USART_TypeDef *USARTx)
功能描述	检查是否使能发送完成中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.86 函数 LL_USART_IsEnabledIT_TXE

描述了函数 LL_USART_IsEnabledIT_TXE

表47-117 函数 LL_USART_IsEnabledIT_TXE

函数名	LL_USART_IsEnabledIT_TXE
函数原形	__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_TXE (USART_TypeDef *USARTx)
功能描述	检查是否使能发送数据寄存器空中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.87 函数 LL_USART_IsEnabledIT_PE

描述了函数 LL_USART_IsEnabledIT_PE

表47-118 函数 LL_USART_IsEnabledIT_PE

函数名	LL_USART_IsEnabledIT_PE
函数原形	__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_PE (USART_TypeDef *USARTx)
功能描述	检查是否使能奇偶校验错误中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.88 函数 LL_USART_IsEnabledIT_ERROR

描述了函数 LL_USART_IsEnabledIT_ERROR

表47-119 函数 LL_USART_IsEnabledIT_ERROR

函数名	LL_USART_IsEnabledIT_ERROR
函数原形	__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_ERROR (USART_TypeDef *USARTx)
功能描述	检查是否使能错误中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.89 函数 LL_USART_IsEnabledIT_CTS

描述了函数 LL_USART_IsEnabledIT_CTS

表47-120 函数 LL_USART_IsEnabledIT_CTS

函数名	LL_USART_IsEnabledIT_CTS
函数原形	__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_CTS (USART_TypeDef *USARTx)
功能描述	检查是否使能 CTS 中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.90 函数 LL_USART_EnableDMAReq_RX

描述了函数 LL_USART_EnableDMAReq_RX

表47-121 函数 LL_USART_EnableDMAReq_RX

函数名	LL_USART_EnableDMAReq_RX
函数原形	__STATIC_INLINE void LL_USART_EnableDMAReq_RX (USART_TypeDef *USARTx)
功能描述	使能 DMA 接收中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.91 函数 LL_USART_DisableDMAReq_RX

描述了函数 LL_USART_DisableDMAReq_RX

表47-122 函数 LL_USART_DisableDMAReq_RX

函数名	LL_USART_DisableDMAReq_RX
函数原形	__STATIC_INLINE void LL_USART_DisableDMAReq_RX (USART_TypeDef *USARTx)
功能描述	禁用 DMA 接收中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.92 函数 LL_USART_IsEnabledDMAReq_RX

描述了函数 LL_USART_IsEnabledDMAReq_RX

表47-123 函数 LL_USART_IsEnabledDMAReq_RX

函数名	LL_USART_IsEnabledDMAReq_RX
函数原形	__STATIC_INLINE uint32_t LL_USART_IsEnabledDMAReq_RX (USART_TypeDef *USARTx)
功能描述	检查是否使能 DMA 接收中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.93 函数 LL_USART_EnableDMAReq_TX

描述了函数 LL_USART_EnableDMAReq_TX

表47-124 函数 LL_USART_EnableDMAReq_TX

函数名	LL_USART_EnableDMAReq_TX
函数原形	__STATIC_INLINE void LL_USART_EnableDMAReq_TX (USART_TypeDef *USARTx)
功能描述	使能 DMA 发送中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.94 函数 LL_USART_DisableDMAReq_TX

描述了函数 LL_USART_DisableDMAReq_TX

表47-125 函数 LL_USART_DisableDMAReq_TX

函数名	LL_USART_DisableDMAReq_TX
函数原形	__STATIC_INLINE void LL_USART_DisableDMAReq_RX (USART_TypeDef *USARTx)
功能描述	禁用 DMA 发送中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.95 函数 LL_USART_IsEnabledDMAReq_TX

描述了函数 LL_USART_IsEnabledDMAReq_TX

表47-126 函数 LL_USART_IsEnabledDMAReq_TX

函数名	LL_USART_IsEnabledDMAReq_TX
函数原形	__STATIC_INLINE uint32_t LL_USART_IsEnabledDMAReq_TX (USART_TypeDef *USARTx)
功能描述	检查是否使能 DMA 发送中断
输入参数	USARTx：USART 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

47.2.96 函数 LL_USART_DMA_GetRegAddr

描述了函数 LL_USART_DMA_GetRegAddr

表47-127 函数 LL_USART_DMA_GetRegAddr

函数名	LL_USART_DMA_GetRegAddr
函数原形	__STATIC_INLINE uint32_t LL_USART_DMA_GetRegAddr (USART_TypeDef

	*USARTx)
功能描述	获取数据寄存器地址
输入参数	USARTx：USART 实例
输出参数	无
返回值	数据寄存器地址
先决条件	无

47.2.97 函数 LL_USART_ReceiveData8

描述了函数 LL_USART_ReceiveData8

表47-128 函数 LL_USART_ReceiveData8

函数名	LL_USART_ReceiveData8
函数原形	__STATIC_INLINE uint32_t LL_USART_ReceiveData8 (USART_TypeDef *USARTx)
功能描述	读接收数据寄存器（8 位数据长度）
输入参数	USARTx：USART 实例
输出参数	无
返回值	8 位数据
先决条件	无

47.2.98 函数 LL_USART_ReceiveData9

描述了函数 LL_USART_ReceiveData9

表47-129 函数 LL_USART_ReceiveData9

函数名	LL_USART_ReceiveData9
函数原形	__STATIC_INLINE uint32_t LL_USART_ReceiveData9 (USART_TypeDef *USARTx)
功能描述	读接收数据寄存器（9 位数据长度）
输入参数	USARTx：USART 实例
输出参数	无
返回值	9 位数据
先决条件	无

47.2.99 函数 LL_USART_TransmitData8

描述了函数 LL_USART_TransmitData8

表47-130 函数 LL_USART_TransmitData8

函数名	LL_USART_TransmitData8
函数原形	__STATIC_INLINE uint32_t LL_USART_TransmitData8 (USART_TypeDef *USARTx)
功能描述	写发送数据寄存器（8 位数据长度）
输入参数	USARTx：USART 实例

输出参数	无
返回值	无
先决条件	无

47.2.100 函数 LL_USART_TransmitData9

描述了函数 LL_USART_TransmitData9

表47-131 函数 LL_USART_TransmitData9

函数名	LL_USART_TransmitData9
函数原形	__STATIC_INLINE uint32_t LL_USART_TransmitData9 (USART_TypeDef *USARTx)
功能描述	写发送数据寄存器（9 位数据长度）
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.101 函数 LL_USART_RequestBreakSending

描述了函数 LL_USART_RequestBreakSending

表47-132 函数 LL_USART_RequestBreakSending

函数名	LL_USART_RequestBreakSending
函数原形	__STATIC_INLINE uint32_t LL_USART_RequestBreakSending (USART_TypeDef *USARTx)
功能描述	发送断开帧
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.102 函数 LL_USART_RequestEnterMuteMode

描述了函数 LL_USART_RequestEnterMuteMode

表47-133 函数 LL_USART_RequestEnterMuteMode

函数名	LL_USART_RequestEnterMuteMode
函数原形	__STATIC_INLINE uint32_t LL_USART_RequestEnterMuteMode (USART_TypeDef *USARTx)
功能描述	进入静默模式
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.103 函数 LL_USART_RequestExitMuteMode

描述了函数 LL_USART_RequestExitMuteMode

表47-134 函数 LL_USART_RequestExitMuteMode

函数名	LL_USART_RequestExitMuteMode
函数原形	__STATIC_INLINE uint32_t LL_USART_RequestExitMuteMode (USART_TypeDef *USARTx)
功能描述	退出静默模式
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.104 函数 LL_USART_EnableAutoBaudate

描述了函数 LL_USART_EnableAutoBaudate

表47-135 函数 LL_USART_EnableAutoBaudate

函数名	LL_USART_EnableAutoBaudate
函数原形	__STATIC_INLINE void LL_USART_EnableAutoBaudate (USART_TypeDef *USARTx)
功能描述	使能自动波特率检测
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.105 函数 LL_USART_DisableAutoBaudate

描述了函数 LL_USART_DisableAutoBaudate

表47-136 函数 LL_USART_DisableAutoBaudate

函数名	LL_USART_DisableAutoBaudate
函数原形	__STATIC_INLINE void LL_USART_DisableAutoBaudate (USART_TypeDef *USARTx)
功能描述	禁用自动波特率检测
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.106 函数 LL_USART_IsEnabledAutoBaudate

描述了函数 LL_USART_IsEnabledAutoBaudate

表47-137 函数 LL_USART_IsEnabledAutoBaudate

函数名	LL_USART_IsEnabledAutoBaudate
函数原形	__STATIC_INLINE uint32_t LL_USART_IsEnabledAutoBaudate (USART_TypeDef *USARTx)
功能描述	检查是否使能自动波特率检测
输入参数	USARTx：USART 实例
输出参数	无
返回值	1 或 0
先决条件	无

47.2.107 函数 LL_USART_SetAutoBaudateMode

描述了函数 LL_USART_SetAutoBaudateMode

表47-138 函数 LL_USART_SetAutoBaudateMode

函数名	LL_USART_SetAutoBaudateMode
函数原形	__STATIC_INLINE void LL_USART_SetAutoBaudateMode (USART_TypeDef *USARTx, uint32_t mode)
功能描述	设置自动波特率检测模式
输入参数 1	USARTx：USART 实例
输入参数 2	mode：自动波特率检测模式
输出参数	无
返回值	无
先决条件	无

mode 可选参数:

表47-139 mode 可选参数

参数	描述
LL_USART_AUTOBAUDRATE_ONSTARTBIT	从 start 位开始测量波特率
LL_USART_AUTOBAUDRATE_ONFALLINGEDGE	下降沿到下降沿测量

47.2.108 函数 LL_USART_GetAutoBaudateMode

描述了函数 LL_USART_GetAutoBaudateMode

表47-140 函数 LL_USART_GetAutoBaudateMode

函数名	LL_USART_GetAutoBaudateMode
函数原形	__STATIC_INLINE uint32_t LL_USART_GetAutoBaudateMode (USART_TypeDef *USARTx)
功能描述	获取自动波特率检测模式
输入参数	USARTx：USART 实例
输出参数	无
返回值	自动波特率检测模式

先决条件	无

47.2.109 函数 LL_USART_SendAutoBaudateReq

描述了函数 LL_USART_SendAutoBaudateReq

表47-141 函数 LL_USART_SendAutoBaudateReq

函数名	LL_USART_SendAutoBaudateReq
函数原形	__STATIC_INLINE uint32_t LL_USART_SendAutoBaudateReq (USART_TypeDef *USARTx)
功能描述	发送自动波特率请求
输入参数	USARTx：USART 实例
输出参数	无
返回值	无
先决条件	无

47.2.110 函数 LL_USART_DeInit

描述了函数 LL_USART_DeInit

表47-142 函数 LL_USART_DeInit

函数名	LL_USART_DeInit
函数原形	__STATIC_INLINE uint32_t LL_USART_DeInit (USART_TypeDef *USARTx)
功能描述	将 USART 配置重设为缺省值
输入参数	USARTx：USART 实例
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

47.2.111 函数 LL_USART_Init

描述了函数 LL_USART_Init

表47-143 函数 LL_USART_Init

函数名	LL_USART_Init
函数原形	__STATIC_INLINE uint32_t LL_USART_Init (USART_TypeDef *USARTx, LL_USART_InitTypeDef *USART_InitStruct)
功能描述	初始化 USART
输入参数 1	USARTx：USART 实例
输入参数 2	USART_InitStruct：指向 LL_USART_InitTypeDef 结构的指针，该结构包含指定 USART 外设的配置信息
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

47.2.112 函数 LL_USART_StructInit

描述了函数 LL_USART_StructInit

表47-144 函数 LL_USART_StructInit

函数名	LL_USART_StructInit
函数原形	void LL_USART_StructInit(LL_USART_InitTypeDef *USART_InitStruct)
功能描述	将结构体 LL_USART_InitTypeDef 字段设置为默认值
输入参数	无
输出参数	USART_InitStruct：指向结构体 LL_USART_InitTypeDef 指针
返回值	无
先决条件	无

47.2.113 函数 LL_USART_ClockInit

描述了函数 LL_USART_ClockInit

表47-145 函数 LL_USART_ClockInit

函数名	LL_USART_ClockInit
函数原形	ErrorStatus LL_USART_ClockInit(USART_TypeDef *USARTx, LL_USART_ClockInitTypeDef *USART_ClockInitStruct)
功能描述	初始化 USART 时钟
输入参数 1	USARTx：USART 实例
输入参数 2	USART_ClockInitStruct：指向 LL_USART_ClockInitTypeDef 结构的指针，该结构包含指定 USART 时钟的配置信息
输出参数	无
返回值	函数执行结果（成功 SUCCESS 或失败 ERROR）
先决条件	无

47.2.114 函数 LL_USART_ClockStructInit

描述了函数 LL_USART_ClockStructInit

表47-146 函数 LL_USART_ClockStructInit

函数名	LL_USART_ClockStructInit
函数原形	void LL_USART_ClockStructInit(LL_USART_ClockInitTypeDef *USART_ClockInitStruct)
功能描述	将结构体 LL_USART_ClockInitTypeDef 字段设置为默认值
输入参数	无
输出参数	USART_ClockInitStruct：指向 LL_USART_ClockInitTypeDef 结构的指针
返回值	无
先决条件	无

48 LL 窗口看门狗通用驱动程序（WWDG）

系统窗口看门狗(WWDG)通常被用来监测由外部干扰或不可预见的逻辑条件造成的应用程序背离正常的运行序列而产生的软件故障。除非程序在 T6 位变成 0 前刷新递减计数器的值，否则看门狗电路在达到预置的时间周期时，会产生一个 MCU 复位。如果在递减计数器达到窗口寄存器值之前刷新控制寄存器中的 7 位递减计数器值，也会产生 MCU 复位。这意味着必须在限定的时间窗口内刷新计数器。

WWDG 时钟由APB 时钟经预分频后提供，通过可配置的时间窗口来检测应用程序非正常的过迟或过早的操作。

WWDG 最适合那些要求看门狗在精确计时窗口起作用的应用程序。

48.1 WWDG 固件库函数

表48-1 CRC 固件库函数说明

函数名	描述
LL_WWDG_Enable	启用WWDG
LL_WWDG_IsEnabled	检查是否启用了窗口看门狗
LL_WWDG_SetCounter	将看门狗计数器值设置为提供的值
LL_WWDG_GetCounter	返回当前 WWDG 计数器值
LL_WWDG_SetPrescaler	设置预分频器
LL_WWDG_GetPrescaler	返回当前 WWDG 预分频器值
LL_WWDG_SetWindow	设置要与递减计数器比较的 WWDG 窗口值
LL_WWDG_GetWindow	返回当前 WWDG 窗口值
LL_WWDG_IsActiveFlag_EWKUP	是否激活置了 WWDG 提前唤醒中断标志
LL_WWDG_ClearFlag_EWKUP	清除 WWDG 提前唤醒中断标志 (EWIF)
LL_WWDG_EnableIT_EWKUP	启用提前唤醒中断
LL_WWDG_IsEnabledIT_EWKUP	检查是否启用了提前唤醒中断

48.1.1 函数 LL_WWDG_Enable

描述了函数 LL_WWDG_Enable

表48-2 函数 LL_WWDG_Enable

函数名	LL_WWDG_Enable
函数原形	__STATIC_INLINE void LL_WWDG_Enable(WWDG_TypeDef *WWDGx)
功能描述	启用WWDG
输入参数	WWDGx：WWDG 实例
输出参数	无

返回值	无
先决条件	无

48.1.2 函数 LL_WWDG_IsEnabled

描述了函数 LL_WWDG_IsEnabled

表48-3 函数 LL_WWDG_IsEnabled

函数名	LL_WWDG_IsEnabled
函数原形	__STATIC_INLINE uint32_t LL_WWDG_IsEnabled(WWDG_TypeDef *WWDGx)
功能描述	检查是否启用了窗口看门狗
输入参数	WWDGx：WWDG 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

48.1.3 函数 LL_WWDG_SetCounter

描述了函数 LL_WWDG_SetCounter

表48-4 函数 LL_WWDG_SetCounter

函数名	LL_WWDG_SetCounter
函数原形	__STATIC_INLINE void LL_WWDG_SetCounter(WWDG_TypeDef *WWDGx, uint32_t Counter)
功能描述	将看门狗计数器值设置为提供的值
输入参数 1	WWDGx：WWDG 实例
输入参数 2	Counter：7 位计数器值
输出参数	无
返回值	无
先决条件	无

48.1.4 函数 LL_WWDG_GetCounter

描述了函数 LL_WWDG_GetCounter

表48-5 函数 LL_WWDG_GetCounter

函数名	LL_WWDG_GetCounter
函数原形	__STATIC_INLINE uint32_t LL_WWDG_GetCounter(WWDG_TypeDef *WWDGx)
功能描述	返回当前WWDG 计数器值
输入参数	WWDGx：WWDG 实例
输出参数	无
返回值	WWDG 计数器值
先决条件	无

48.1.5 函数 LL_WWDG_SetPrescaler

描述了函数 LL_WWDG_SetPrescaler

表48-6 函数 LL_WWDG_SetPrescaler

函数名	LL_WWDG_SetPrescaler
函数原形	__STATIC_INLINE void LL_WWDG_SetPrescaler(WWDG_TypeDef *WWDGx, uint32_t Prescaler)
功能描述	设置预分频器
输入参数 1	WWDGx：WWDG 实例
输入参数 2	Prescaler：预分频值
输出参数	无
返回值	无
先决条件	无

48.1.6 函数 LL_WWDG_GetPrescaler

描述了函数 LL_WWDG_GetPrescaler

表48-7 函数 LL_WWDG_GetPrescaler

函数名	LL_WWDG_GetPrescaler
函数原形	__STATIC_INLINE uint32_t LL_WWDG_GetPrescaler(WWDG_TypeDef *WWDGx)
功能描述	返回当前WWDG 预分频器值
输入参数	WWDGx：WWDG 实例
输出参数	无
返回值	WWDG 预分频器值
先决条件	无

48.1.7 函数 LL_WWDG_SetWindow

描述了函数 LL_WWDG_SetWindow

表48-8 函数 LL_WWDG_SetWindow

函数名	LL_WWDG_SetWindow
函数原形	__STATIC_INLINE void LL_WWDG_SetWindow(WWDG_TypeDef *WWDGx, uint32_t Window)
功能描述	设置要与递减计数器比较的WWDG 窗口值
输入参数 1	WWDGx：WWDG 实例
输入参数 2	Window ：WWDG 窗口值
输出参数	无
返回值	无
先决条件	无

48.1.8 函数 LL_WWDG_GetWindow

描述了函数 LL_WWDG_GetWindow

表48-9 函数 LL_WWDG_GetWindow

函数名	LL_WWDG_GetWindow
函数原形	__STATIC_INLINE uint32_t LL_WWDG_GetWindow(WWDG_TypeDef *WWDGx)
功能描述	返回当前WWDG 窗口值
输入参数	WWDGx：WWDG 实例
输出参数	无
返回值	WWDG 窗口值
先决条件	无

48.1.9 函数 LL_WWDG_IsActiveFlag_EWKUP

描述了函数 LL_WWDG_IsActiveFlag_EWKUP

表48-10 函数 LL_WWDG_IsActiveFlag_EWKUP

函数名	LL_WWDG_IsActiveFlag_EWKUP
函数原形	__STATIC_INLINE uint32_t LL_WWDG_IsActiveFlag_EWKUP(WWDG_TypeDef *WWDGx)
功能描述	是否激活置了 WWDG 提前唤醒标志
输入参数	WWDGx：WWDG 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

48.1.10 函数 LL_WWDG_ClearFlag_EWKUP

描述了函数 LL_WWDG_ClearFlag_EWKUP

表48-11 函数 LL_WWDG_ClearFlag_EWKUP

函数名	LL_WWDG_ClearFlag_EWKUP
函数原形	__STATIC_INLINE void LL_WWDG_ClearFlag_EWKUP(WWDG_TypeDef *WWDGx)
功能描述	清除WWDG 提前唤醒标志(EWIF)
输入参数	WWDGx：WWDG 实例
输出参数	无
返回值	无
先决条件	无

48.1.11 函数 LL_WWDG_EnableIT_EWKUP

描述了函数 LL_WWDG_EnableIT_EWKUP

表48-12 函数 LL_WWDG_EnableIT_EWKUP

函数名	LL_WWDG_EnableIT_EWKUP
函数原形	__STATIC_INLINE void LL_WWDG_EnableIT_EWKUP(WWDG_TypeDef *WWDGx)
功能描述	启用提前唤醒中断
输入参数	WWDGx：WWDG 实例
输出参数	无
返回值	无
先决条件	无

48.1.12 函数 LL_WWDG_IsEnabledIT_EWKUP

描述了函数 LL_WWDG_IsEnabledIT_EWKUP

表48-13 函数 LL_WWDG_IsEnabledIT_EWKUP

函数名	LL_WWDG_IsEnabledIT_EWKUP
函数原形	__STATIC_INLINE uint32_t LL_WWDG_IsEnabledIT_EWKUP(WWDG_TypeDef *WWDGx)
功能描述	检查是否启用了提前唤醒中断
输入参数	WWDGx：WWDG 实例
输出参数	无
返回值	状态位（1 或 0）
先决条件	无

49 历史版本

版本	日期	更新记录
V1.0.0	2023.7.17	初版