PUYA

用户手册

32 位基于 ARM 微控制器 PY32F030

HAL 库函数说明

介绍

HAL 驱动库已经实现了适用于 PY32F0xx 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 PY32F0xx 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。

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

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
py32f0xx_hal_ppp.c	主要外设的驱动程序。 该类文件包含所有 PY32F0xx 芯片的通用 APIs。 例如：py32f0xx_hal_adc.c，py32f0xx_hal_uart.c，……
py32f0xx_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：py32f0xx_hal_adc.h，py32f0xx_hal_uart.h，……
py32f0xx_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：py32f0xx_hal_adc_ex.c，py32f0xx_hal_flash_ex.c

py32f0xx_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：py32f0xx_hal_adc_ex.h，py32f0xx_hal_flash_ex.h
py32f0xx_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。
py32f0xx_hal_h	py32f0xx_hal.c 的头文件。
py32f0xx_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_py32f0xx.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_py32f0xx.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。
py32f0xx_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。
py32f0xx_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。
py32f0xx_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）
py32f0xx_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 PY32F0xx 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 py32f0xx_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，py32f0xx_hal_def.h 文件还会调用 CMSIS 库中的 py32f0xx.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“py32f0xx_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，py32f0xx_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 HAL 系统驱动程序

3.1 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。

3.1.1 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。

3.1.2 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3-1 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

3.1.3 HAL 控制功能

表3-2 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

3.2 功能详细说明

3.2.1 函数 HAL_Init

描述了函数 HAL_Init

表3-3 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

3.2.2 函数 HAL_DeInit

描述了函数 HAL_DeInit

表3-4 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

3.2.3 函数 HAL_MspInit

描述了函数 HAL_MspInit

表3-5 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

3.2.4 函数 HAL_MspDeInit

描述了函数 HAL_MspDeInit

表3-6 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

3.2.5 函数 HAL_InitTick

描述了函数 HAL_InitTick

表3-7 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

3.2.6 函数 HAL_IncTick

描述了函数 HAL_IncTick

表3-8 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

3.2.7 函数 HAL_Delay

描述了函数 HAL_Delay

表3-9 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

3.2.8 函数 HAL_GetTick

描述了函数 HAL_GetTick

表3-10 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

3.2.9 函数 HAL_SuspendTick

描述了函数 HAL_SuspendTick

表3-11 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

3.2.10 函数 HAL_ResumeTick

描述了函数 HAL_ResumeTick

表3-12 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

3.2.11 函数 HAL_GetHalVersion

描述了函数 HAL_GetHalVersion

表3-13 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

3.2.12 函数 HAL_GetREVID

描述了函数 HAL_GetREVID

表3-14 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

3.2.13 函数 HAL_GetDEVID

描述了函数 HAL_GetDEVID

表3-15 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

3.2.14 函数 HAL_GetUIDw0

描述了函数 HAL_GetUIDw0

表3-16 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

3.2.15 函数 HAL_GetUIDw1

描述了函数 HAL_GetUIDw1

表3-17 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

3.2.16 函数 HAL_GetUIDw2

描述了函数 HAL_GetUIDw2

表3-18 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

4 HAL 模拟/数字转换器通用驱动程序（ADC）

模拟/数字转换器（ADC）是一种提供可选择多通道输入，逐次逼近型的模数转换器。分辨率可选 6、8、10、12 位。

4.1 ADC 固件驱动寄存器结构

4.1.1 ADC_InitTypeDef

ADC_InitTypeDef，定义于文件“py32f0xx_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4-1 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4-2 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4-3 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4-4 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4-5 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4-6 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4-7 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4-8 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4-9 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4-10 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4-11 ExternalTrigConvEdge 可选参数

参数	描述
ADC_EXTERNALTRIGCONVEDGE_NONE	禁止硬件触发检测（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4-12 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4-13 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4-14 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

4.1.2 ADC_ChannelConfTypeDef

ADC_ChannelConfTypeDef，定义于文件“py32f0xx_hal_adc.h”如下：

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

字段说明：

表4-15 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4-16 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4-17 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4-18 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

4.1.3 ADC_AnalogWDGConfTypeDef

ADC_AnalogWDGConfTypeDef，定义于文件“py32f0xx_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4-19 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4-20 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4-21 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4-22 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4-23 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4-24 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）

4.1.4 ADC_HandleTypeDef

ADC_HandleTypeDef，定义于文件“py32f0xx_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4-25 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

4.2 ADC 固件库函数

表4-26 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

4.2.1 函数 HAL_ADC_Init

描述了函数 HAL_ADC_Init

表4-27 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

4.2.2 函数 HAL_ADC_DeInit

描述了函数 HAL_ADC_DeInit

表4-28 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

4.2.3 函数 HAL_ADC_MspInit

描述了函数 HAL_ADC_MspInit

表4-29 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

4.2.4 函数 HAL_ADC_MspDeInit

描述了函数 HAL_ADC_MspDeInit

表4-30 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

4.2.5 函数 HAL_ADC_Start

描述了函数 HAL_ADC_Start

表4-31 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

4.2.6 函数 HAL_ADC_Stop

描述了函数 HAL_ADC_Stop

表4-32 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

4.2.7 函数 HAL_ADC_PollForConversion

描述了函数 HAL_ADC_PollForConversion

表4-33 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

4.2.8 函数 HAL_ADC_PollForEvent

描述了函数 HAL_ADC_PollForEvent

表4-34 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

4.2.9 函数 HAL_ADC_Start_IT

描述了函数 HAL_ADC_Start_IT

表4-35 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

4.2.10 函数 HAL_ADC_Stop_IT

描述了函数 HAL_ADC_Stop_IT

表4-36 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

4.2.11 函数 HAL_ADC_Start_DMA

描述了函数 HAL_ADC_Start_DMA

表4-37 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

4.2.12 函数 HAL_ADC_Stop_DMA

描述了函数 HAL_ADC_Stop_DMA

表4-38 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

4.2.13 函数 HAL_ADC_GetValue

描述了函数 HAL_ADC_GetValue

表4-39 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

4.2.14 函数 HAL_ADC_IRQHandler

描述了函数 HAL_ADC_IRQHandler

表4-40 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

4.2.15 函数 HAL_ADC_ConvCpltCallback

描述了函数 HAL_ADC_ConvCpltCallback

表4-41 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

4.2.16 函数 HAL_ADC_ConvHalfCpltCallback

描述了函数 HAL_ADC_ConvHalfCpltCallback

表4-42 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

4.2.17 函数 HAL_ADC_LevelOutOfWindowCallback

描述了函数 HAL_ADC_LevelOutOfWindowCallback

表4-43 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

4.2.18 函数 HAL_ADC_ErrorCallback

描述了函数 HAL_ADC_ErrorCallback

表4-44 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

4.2.19 函数 HAL_ADC_Calibration_Start

描述了函数 HAL_ADC_Calibration_Start

表4-45 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

4.2.20 函数 HAL_ADC_ConfigChannel

描述了函数 HAL_ADC_ConfigChannel

表4-46 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

4.2.21 函数 HAL_ADC_AnalogWDGConfig

描述了函数 HAL_ADC_AnalogWDGConfig

表4-47 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

4.2.22 函数 HAL_ADC_GetState

描述了函数 HAL_ADC_GetState

表4-48 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

4.2.23 函数 HAL_ADC_GetError

描述了函数 HAL_ADC_GetError

表4-49 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

5 HAL 比较器通用驱动程序（COMP）

比较器（COMP）组件提供了一种用硬件来比较两个模拟输入电压的解决方案。输出可在软件中采样，或者以数字方式路由至另一个组件。

5.1 比较器固件驱动寄存器结构

5.1.1 COMP_InitTypeDef

COMP_InitTypeDef，定义于文件“py32f0xx_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5-1 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5-2 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5-3 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5-4 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5-5 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5-6 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5-7 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5-8 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5-9 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

5.1.2 COMP_HandleTypeDef

COMP_HandleTypeDef，定义于文件“py32f0xx_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5-10 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

5.2 COMP 固件库函数

表5-11 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

5.2.1 函数 HAL_COMP_Init

描述了函数 HAL_COMP_Init

表5-12 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

5.2.2 函数 HAL_COMP_DeInit

描述了函数 HAL_COMP_DeInit

表5-13 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

5.2.3 函数 HAL_COMP_MspInit

描述了函数 HAL_COMP_MspInit

表5-14 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

5.2.4 函数 HAL_COMP_MspDeInit

描述了函数 HAL_COMP_MspDeInit

表5-15 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

5.2.5 函数 HAL_COMP_Start

描述了函数 HAL_COMP_Start

表5-16 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

5.2.6 函数 HAL_COMP_Stop

描述了函数 HAL_COMP_Stop

表5-17 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

5.2.7 函数 HAL_COMP_IRQHandler

描述了函数 HAL_COMP_IRQHandler

表5-18 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

5.2.8 函数 HAL_COMP_Lock

描述了函数 HAL_COMP_Lock

表5-19 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

5.2.9 函数 HAL_COMP_GetOutputLevel

描述了函数 HAL_COMP_GetOutputLevel

表5-20 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

5.2.10 函数 HAL_COMP_TriggerCallback

描述了函数 HAL_COMP_TriggerCallback

表5-21 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

5.2.11 函数 HAL_COMP_GetState

描述了函数 HAL_COMP_GetState

表5-22 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

5.2.12 函数 HAL_COMP_GetError

描述了函数 HAL_COMP_GetError

表5-23 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

6 HAL Cortex 通用驱动程序（CORTEX）

Cortex 包含对 NVIC、SYSTICK 的配置。

6.1 Cortex 固件库函数

表6-1 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

6.1.1 函数 HAL_NVIC_SetPriority

描述了函数 HAL_NVIC_SetPriority

表6-2 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6-3 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

6.1.2 函数 HAL_NVIC_EnableIRQ

描述了函数 HAL_NVIC_EnableIRQ

表6-4 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

6.1.3 函数 HAL_NVIC_DisableIRQ

描述了函数 HAL_NVIC_DisableIRQ

表6-5 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

6.1.4 函数 HAL_NVIC_SystemReset

描述了函数 HAL_NVIC_SystemReset

表6-6 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

6.1.5 函数 HAL_SYSTICK_Config

描述了函数 HAL_SYSTICK_Config

表6-7 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

6.1.6 函数 HAL_NVIC_GetPriority

描述了函数 HAL_NVIC_GetPriority

表6-8 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

6.1.7 函数 HAL_NVIC_GetPendingIRQ

描述了函数 HAL_NVIC_GetPendingIRQ

表6-9 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

6.1.8 函数 HAL_NVIC_SetPendingIRQ

描述了函数 HAL_NVIC_SetPendingIRQ

表6-10 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

6.1.9 函数 HAL_NVIC_ClearPendingIRQ

描述了函数 HAL_NVIC_ClearPendingIRQ

表6-11 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

6.1.10 函数 HAL_SYSTICK_CLKSourceConfig

描述了函数 HAL_SYSTICK_CLKSourceConfig

表6-12 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6-13 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

6.1.11 函数 HAL_SYSTICK_IRQHandler

描述了函数 HAL_SYSTICK_IRQHandler

表6-14 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

6.1.12 函数 HAL_SYSTICK_Callback

描述了函数 HAL_SYSTICK_Callback

表6-15 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

7 HAL 循环冗余校验（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。

7.1 CRC 固件驱动寄存器结构

7.1.1 CRC_HandleTypeDef

CRC_HandleTypeDef，定义于文件“py32f0xx_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7-1 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

7.2 CRC 固件库函数

表7-2 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

7.2.1 函数 HAL_CRC_Init

描述了函数 HAL_CRC_Init

表7-3 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

7.2.2 函数 HAL_CRC_DeInit

描述了函数 HAL_CRC_DeInit

表7-4 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

7.2.3 函数 HAL_CRC_MspInit

描述了函数 HAL_CRC_MspInit

表7-5 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

7.2.4 函数 HAL_CRC_MspDeInit

描述了函数 HAL_CRC_MspDeInit

表7-6 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

7.2.5 函数 HAL_CRC_Accumulate

描述了函数 HAL_CRC_Accumulate

表7-7 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

7.2.6 函数 HAL_CRC_Calculate

描述了函数 HAL_CRC_Calculate

表7-8 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

7.2.7 函数 HAL_CRC_GetState

描述了函数 HAL_CRC_GetState

表7-9 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

8 DMA 控制器（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。

8.1 DMA 固件驱动寄存器结构

8.1.1 DMA_InitTypeDef

DMA_InitTypeDef，定义于文件"py32f0xx_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8-1 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8-2 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8-3 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8-4 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8-5 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8-6 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8-7 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8-8 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

8.1.2 DMA_HandleTypeDef

DMA_HandleTypeDef，定义于文件"py32f0xx_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8-9 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

8.2 DMA 固件库函数

表8-10 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

8.2.1 函数 HAL_DMA_Init

描述了函数 HAL_DMA_Init

表8-11 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

8.2.2 函数 HAL_DMA_DeInit

描述了函数 HAL_DMA_DeInit

表8-12 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

8.2.3 函数 HAL_DMA_Start

描述了函数 HAL_DMA_Start

表8-13 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

8.2.4 函数 HAL_DMA_Start_IT

描述了函数 HAL_DMA_Start_IT

表8-14 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

8.2.5 函数 HAL_DMA_Abort

描述了函数 HAL_DMA_Abort

表8-15 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

8.2.6 函数 HAL_DMA_Abort_IT

描述了函数 HAL_DMA_Abort_IT

表8-16 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

8.2.7 函数 HAL_DMA_PollForTransfer

描述了函数 HAL_DMA_PollForTransfer

表8-17 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

8.2.8 函数 HAL_DMA_IRQHandler

描述了函数 HAL_DMA_IRQHandler

表8-18 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

8.2.9 函数 HAL_DMA_RegisterCallback

描述了函数 HAL_DMA_RegisterCallback

表8-19 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

8.2.10 函数 HAL_DMA_ChannelMap

描述了函数 HAL_DMA_ChannelMap

表8-20 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8-21 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

8.2.11 函数 HAL_DMA_UnRegisterCallback

描述了函数 HAL_DMA_UnRegisterCallback

表8-22 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

8.2.12 函数 HAL_DMA_GetState

描述了函数 HAL_DMA_GetState

表8-23 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

8.2.13 函数 HAL_DMA_GetError

描述了函数 HAL_DMA_GetError

表8-24 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

9 HAL 外部中断/事件控制器（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。

9.1 EXTI 固件驱动寄存器结构

9.1.1 EXTI_HandleTypeDef

EXTI_HandleTypeDef，定义于文件”py32f0xx_hal_exti.h”如下：

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

字段说明：

表9-1 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

9.1.2 EXTI_ConfigTypeDef

EXTI_ConfigTypeDef，定义于文件”py32f0xx_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9-2 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9-3 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9-4 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9-5 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9-6 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

9.2 EXTI 固件库函数

表9-7 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

9.2.1 函数 HAL_EXTI_SetConfigLine

描述了函数 HAL_EXTI_SetConfigLine

表9-8 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

9.2.2 函数 HAL_EXTI_GetConfigLine

描述了函数 HAL_EXTI_GetConfigLine

表9-9 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

9.2.3 函数 HAL_EXTI_ClearConfigLine

描述了函数 HAL_EXTI_ClearConfigLine

表9-10 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

9.2.4 函数 HAL_EXTI_RegisterCallback

描述了函数 HAL_EXTI_RegisterCallback

表9-11 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

9.2.5 函数 HAL_EXTI_GetHandle

描述了函数 HAL_EXTI_GetHandle

表9-12 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

9.2.6 函数 HAL_EXTI_IRQHandler

描述了函数 HAL_EXTI_IRQHandler

表9-13 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

9.2.7 函数 HAL_EXTI_GetPending

描述了函数 HAL_EXTI_GetPending

表9-14 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9-15 Edge 可选参数

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

9.2.8 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9-16 Edge 可选参数

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

9.2.9 函数 HAL_EXTI_GenerateSWI

描述了函数 HAL_EXTI_GenerateSWI

表9-17 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

10 HAL 闪存存储器通用驱动程序（FLASH）

Flash 存储器由 32bit 宽的存储单元组成，可以用作程序和数据的存储，Page 大小为 128Bytes，Sector 大小为 4KBytes。

10.1 FLASH 固件驱动寄存器结构

10.1.1 FLASH_EraseInitTypeDef

FLASH_EraseInitTypeDef，定义于文件"py32f0xx_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10-1 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10-2 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

10.1.2 FLASH_OBProgramInitTypeDef

FLASH_OBProgramInitTypeDef，定义于文件"py32f0xx_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10-3 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10-4 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10-5 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10-6 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10-7 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10-8 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

10.1.3 FLASH_ProcessTypeDef

FLASH_ProcessTypeDef，定义于文件”py32f0xx_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10-9 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

10.2 FLASH 固件库函数

表10-10 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

10.2.1 函数 HAL_FLASH_Init

描述了函数 HAL_FLASH_Init

表10-11 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10-12 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

10.2.2 函数 HAL_FLASH_PageProgram

描述了函数 HAL_FLASH_PageProgram

表10-13 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

10.2.3 函数 HAL_FLASH_PageProgram_IT

描述了函数 HAL_FLASH_PageProgram_IT

表10-14 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

10.2.4 函数 HAL_FLASH_IRQHandler

描述了函数 HAL_FLASH_IRQHandler

表10-15 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

10.2.5 函数 HAL_FLASH_EndOfOperationCallback

描述了函数 HAL_FLASH_EndOfOperationCallback

表10-16 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

10.2.6 函数 HAL_FLASH_OperationErrorCallback

描述了函数 HAL_FLASH_OperationErrorCallback

表10-17 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

10.2.7 函数 HAL_FLASH_Erase

描述了函数 HAL_FLASH_Erase

表10-18 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

10.2.8 函数 HAL_FLASH_Erase_IT

描述了函数 HAL_FLASH_Erase_IT

表10-19 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

10.2.9 函数 HAL_FLASH_Unlock

描述了函数 HAL_FLASH_Unlock

表10-20 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

10.2.10 函数 HAL_FLASH_Lock

描述了函数 HAL_FLASH_Lock

表10-21 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

10.2.11 函数 HAL_FLASH_OB_Unlock

描述了函数 HAL_FLASH_OB_Unlock

表10-22 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

10.2.12 函数 HAL_FLASH_OB_Lock

描述了函数 HAL_FLASH_OB_Lock

表10-23 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

10.2.13 函数 HAL_FLASH_OB_Launch

描述了函数 HAL_FLASH_OB_Launch

表10-24 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

10.2.14 函数 HAL_FLASH_OBProgram

描述了函数 HAL_FLASH_OBProgram

表10-25 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

10.2.15 函数 HAL_FLASH_OBGetConfig

描述了函数 HAL_FLASH_OBGetConfig

表10-26 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

10.2.16 函数 HAL_OB_RDP_LevelConfig

描述了函数 HAL_OB_RDP_LevelConfig

表10-27　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

10.2.17 函数 HAL_FLASH_GetError

描述了函数 HAL_FLASH_GetError

表10-28　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

11 HAL 通用输入/输出通用驱动（GPIO）

每个 GPIO 的每个位可以通过软件编程，进行多种模式的配置。

11.1 GPIO 寄存器结构

11.1.1 GPIO_InitTypeDef

GPIO_InitTypeDef，定义于文件”py32f0xx_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11-1 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11-2 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11-3 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11-4 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11-5 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11-6 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

11.2 GPIO 固件库函数

表11-7 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

11.2.1 函数 HAL_GPIO_Init

描述了函数 HAL_GPIO_Init

表11-8 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11-9 GPIOx 可选参数

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

11.2.2 函数 HAL_GPIO_DeInit

描述了函数 HAL_GPIO_DeInit

表11-10 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11-11 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11-12　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

11.2.3 函数 HAL_GPIO_ReadPin

描述了函数 HAL_GPIO_ReadPin

表11-13　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11-14 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11-15 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

11.2.4 函数 HAL_GPIO_WritePin

描述了函数 HAL_GPIO_WritePin

表11-16 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11-17 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11-18 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

11.2.5 函数 HAL_GPIO_TogglePin

描述了函数 HAL_GPIO_TogglePin

表11-19 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11-20 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11-21 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

11.2.6 函数 HAL_GPIO_LockPin

描述了函数 HAL_GPIO_LockPin

表11-22 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11-23 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11-24 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

11.2.7 函数 HAL_GPIO_EXTI_IRQHandler

描述了函数 HAL_GPIO_EXTI_IRQHandler

表11-25 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

11.2.8 函数 HAL_GPIO_EXTI_Callback

描述了函数 HAL_GPIO_EXTI_Callback

表11-26 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

12 HAL 内部集成电路总线通用驱动（I2C）

I2C(inter-integrated circuit)总线接口连接微控制器和串行 I2C 总线。它提供多主机功能，控制所有 I2C 总线特定的顺序、协议、仲裁和时序。支持标准（Sm）、快速（Fm）。

12.1 I2C 固件驱动寄存器结构

12.1.1 I2C_InitTypeDef

I2C_InitTypeDef，定于文件”py32f0xx_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12-1 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12-2 ClockSpeed 可选参数

参数	描述
0~400	时钟频率最高 400kHz

DutyCycle 可选参数：

表12-3 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12-4 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12-5 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

12.1.2 I2C_HandleTypeDef

I2C_HandleTypeDef，定于文件”py32f0xx_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12-6 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

12.2 I2C 固件库函数

表12-7 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

12.2.1 函数 HAL_I2C_Init

描述了函数 HAL_I2C_Init

表12-8 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

12.2.2 函数 HAL_I2C_DeInit

描述了函数 HAL_I2C_DeInit

表12-9 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

12.2.3 函数 HAL_I2C_MspInit

描述了函数 HAL_I2C_MspInit

表12-10 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

12.2.4 函数 HAL_I2C_MspDeInit

描述了函数 HAL_I2C_MspDeInit

表12-11 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

12.2.5 函数 HAL_I2C_Master_Transmit

描述了函数 HAL_I2C_Master_Transmit

表12-12 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

12.2.6 函数 HAL_I2C_Master_Receive

描述了函数 HAL_I2C_Master_Receive

表12-13 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

12.2.7 函数 HAL_I2C_Slave_Transmit

描述了函数 HAL_I2C_Slave_Transmit

表12-14 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

12.2.8 函数 HAL_I2C_Slave_Receive

描述了函数 HAL_I2C_Slave_Receive

表12-15 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

12.2.9 函数 HAL_I2C_Mem_Write

描述了函数 HAL_I2C_Mem_Write

表12-16 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

12.2.10 函数 HAL_I2C_Mem_Read

描述了函数 HAL_I2C_Mem_Read

表12-17 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

12.2.11 函数 HAL_I2C_IsDeviceReady

描述了函数 HAL_I2C_IsDeviceReady

表12-18 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

12.2.12 函数 HAL_I2C_Master_Transmit_IT

描述了函数 HAL_I2C_Master_Transmit_IT

表12-19 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

12.2.13 函数 HAL_I2C_Master_Receive_IT

描述了函数 HAL_I2C_Master_Receive_IT

表12-20 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

12.2.14 函数 HAL_I2C_Slave_Transmit_IT

描述了函数 HAL_I2C_Slave_Transmit_IT

表12-21 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

12.2.15 函数 HAL_I2C_Slave_Receive_IT

描述了函数 HAL_I2C_Slave_Receive_IT

表12-22　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

12.2.16 函数 HAL_I2C_Mem_Write_IT

描述了函数 HAL_I2C_Mem_Write_IT

表12-23　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

12.2.17 函数 HAL_I2C_Mem_Read_IT

描述了函数 HAL_I2C_Mem_Read_IT

表12-24 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

12.2.18 函数 HAL_I2C_EnableListen_IT

描述了函数 HAL_I2C_EnableListen_IT

表12-25 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

12.2.19 函数 HAL_I2C_DisableListen_IT

描述了函数 HAL_I2C_DisableListen_IT

表12-26 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

12.2.20 函数 HAL_I2C_Master_Abort_IT

描述了函数 HAL_I2C_Master_Abort_IT

表12-27 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

12.2.21 函数 HAL_I2C_Master_Transmit_DMA

描述了函数 HAL_I2C_Master_Transmit_DMA

表12-28 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

12.2.22 函数 HAL_I2C_Master_Receive_DMA

描述了函数 HAL_I2C_Master_Receive_DMA

表12-29 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

12.2.23 函数 HAL_I2C_Slave_Transmit_DMA

描述了函数 HAL_I2C_Slave_Transmit_DMA

表12-30 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

12.2.24 函数 HAL_I2C_Slave_Receive_DMA

描述了函数 HAL_I2C_Slave_Receive_DMA

表12-31 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

12.2.25 函数 HAL_I2C_Mem_Write_DMA

描述了函数 HAL_I2C_Mem_Write_DMA

表12-32 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

12.2.26 函数 HAL_I2C_Mem_Read_DMA

描述了函数 HAL_I2C_Mem_Read_DMA

表12-33 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

12.2.27 函数 HAL_I2C_EV_IRQHandler

描述了函数 HAL_I2C_EV_IRQHandler

表12-34 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

12.2.28 函数 HAL_I2C_ER_IRQHandler

描述了函数 HAL_I2C_ER_IRQHandler

表12-35 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

12.2.29 函数 HAL_I2C_MasterTxCpltCallback

描述了函数 HAL_I2C_MasterTxCpltCallback

表12-36 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

12.2.30 函数 HAL_I2C_MasterRxCpltCallback

描述了函数 HAL_I2C_MasterRxCpltCallback

表12-37 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

12.2.31 函数 HAL_I2C_SlaveTxCpltCallback

描述了函数 HAL_I2C_SlaveTxCpltCallback

表12-38 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

12.2.32 函数 HAL_I2C_SlaveRxCpltCallback

描述了函数 HAL_I2C_SlaveRxCpltCallback

表12-39 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

12.2.33 函数 HAL_I2C_AddrCallback

描述了函数 HAL_I2C_AddrCallback

表12-40 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

12.2.34 函数 HAL_I2C_ListenCpltCallback

描述了函数 HAL_I2C_ListenCpltCallback

表12-41 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

12.2.35 函数 HAL_I2C_MemTxCpltCallback

描述了函数 HAL_I2C_MemTxCpltCallback

表12-42 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

12.2.36 函数 HAL_I2C_MemRxCpltCallback

描述了函数 HAL_I2C_MemRxCpltCallback

表12-43 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

12.2.37 函数 HAL_I2C_ErrorCallback

描述了函数 HAL_I2C_ErrorCallback

表12-44 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

12.2.38 函数 HAL_I2C_AbortCpltCallback

描述了函数 HAL_I2C_AbortCpltCallback

表12-45 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

12.2.39 函数 HAL_I2C_GetState

描述了函数 HAL_I2C_GetState

表12-46 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

12.2.40 函数 HAL_I2C_GetMode

描述了函数 HAL_I2C_GetMode

表12-47 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

12.2.41 函数 HAL_I2C_GetError

描述了函数 HAL_I2C_GetError

表12-48　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

13 HAL 独立看门狗通用驱动程序（IWDG）

13.1 IWDG 固件驱动寄存器结构

13.1.1 IWDG_InitTypeDef

IWDG_InitTypeDef，定义于文件”py32f0xx_hal_iwdg.h”如下：

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

字段说明：

表13-1 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13-2 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

13.1.2 IWDG_HandleTypeDef

IWDG_HandleTypeDef，定义于文件”py32f0xx_hal_iwdg.h”如下：

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

字段说明：

表13-3 IWDG_HandleTypeDef 字段说明

字段	描述
Instance	外设基地址
Init	初始化结构体

13.2 IWDG 固件库函数

表13-4 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

13.2.1 函数 HAL_IWDG_Init

描述了函数 HAL_IWDG_Init

表13-5 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

13.2.2 函数 HAL_IWDG_Refresh

描述了函数 HAL_IWDG_Refresh

表13-6 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

14 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。

14.1 LED 固件驱动寄存器结构

14.1.1 LED_InitTypeDef

LED_InitTypeDef，定义于文档”py32f0xx_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14-1 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14-2 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14-3 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）

14.1.2 LED_HandleTypeDef

LED_HandleTypeDef，定义于文档”py32f0xx_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14-4 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

14.2 LED 固件库函数

表14-5 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

14.2.1 函数 HAL_LED_Init

描述了函数 HAL_LED_Init

表14-6 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

14.2.2 函数 HAL_LED_MspInit

描述了函数 HAL_LED_MspInit

表14-7 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

14.2.3 函数 HAL_LED_SetComDisplay

描述了函数 HAL_LED_SetComDisplay

表14-8 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14-9 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14-10 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 .

14.2.4 函数 HAL_LED_LightCompleteCallback

描述了函数 HAL_LED_LightCompleteCallback

表14-11 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

14.2.5 函数 HAL_LED_IRQHandler

描述了函数 HAL_LED_IRQHandler

表14-12 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

15 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。

15.1 LPTIM 固件驱动寄存器结构

15.1.1 LPTIM_InitTypeDef

LPTIM_InitTypeDef，定义于文件"py32f0xx_hal_lptim.h"如下：

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

字段说明：

表15-1 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15-2 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15-3 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

15.1.2 LPTIM_HandleTypeDef

LPTIM_HandleTypeDef，定义于文件"py32f0xx_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15-4 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

15.2 LPTIM 固件库函数

表15-5 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

15.2.1 函数 HAL_LPTIM_Init

描述了函数 HAL_LPTIM_Init

表15-6 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

15.2.2 函数 HAL_LPTIM_DeInit

描述了函数 HAL_LPTIM_DeInit

表15-7 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

15.2.3 函数 HAL_LPTIM_MspInit

描述了函数 HAL_LPTIM_MspInit

表15-8 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

15.2.4 函数 HAL_LPTIM_MspDeInit

描述了函数 HAL_LPTIM_MspDeInit

表15-9 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

15.2.5 函数 HAL_LPTIM_SetOnce_Start

描述了函数 HAL_LPTIM_SetOnce_Start

表15-10 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

15.2.6 函数 HAL_LPTIM_SetOnce_Stop

描述了函数 HAL_LPTIM_SetOnce_Stop

表15-11 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

15.2.7 函数 HAL_LPTIM_SetOnce_Start_IT

描述了函数 HAL_LPTIM_SetOnce_Start_IT

表15-12 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

15.2.8 函数 HAL_LPTIM_SetOnce_Stop_IT

描述了函数 HAL_LPTIM_SetOnce_Stop_IT

表15-13 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

15.2.9 函数 HAL_LPTIM_ReadCounter

描述了函数 HAL_LPTIM_ReadCounter

表15-14 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

15.2.10 函数 HAL_LPTIM_ReadAutoReload

描述了函数 HAL_LPTIM_ReadAutoReload

表15-15 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

15.2.11 函数 HAL_LPTIM_IRQHandler

描述了函数 HAL_LPTIM_IRQHandler

表15-16 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

15.2.12 函数 HAL_LPTIM_AutoReloadMatchCallback

描述了函数 HAL_LPTIM_AutoReloadMatchCallback

表15-17 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

15.2.13 函数 HAL_LPTIM_GetState

描述了函数 HAL_LPTIM_GetState

表15-18 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

16 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。

16.1 PWR 固件驱动寄存器结构

16.1.1 PWR_PVDTypeDef

PWR_PVDTypeDef，定义于文件"py32f0xx_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16-1 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16-2 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16-3 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16-4 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16-5 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

16.2 PWR 固件库函数

表16-6 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

16.2.1 函数 HAL_PWR_DeInit

描述了函数 HAL_PWR_DeInit

表16-7 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

16.2.2 函数 HAL_PWR_EnableBkUpAccess

描述了函数 HAL_PWR_EnableBkUpAccess

表16-8 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

16.2.3 函数 HAL_PWR_DisableBkUpAccess

描述了函数 HAL_PWR_DisableBkUpAccess

表16-9 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

16.2.4 函数 HAL_PWR_ConfigPVD

描述了函数 HAL_PWR_ConfigPVD

表16-10 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

16.2.5 函数 HAL_PWR_EnablePVD

描述了函数 HAL_PWR_EnablePVD

表16-11 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

16.2.6 函数 HAL_PWR_DisablePVD

描述了函数 HAL_PWR_DisablePVD

表16-12 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

16.2.7 函数 HAL_PWR_EnterSLEEPMode

描述了函数 HAL_PWR_EnterSLEEPMode

表16-13 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16-14 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

16.2.8 函数 HAL_PWR_EnterSTOPMode

描述了函数 HAL_PWR_EnterSTOPMode

表16-15 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16-16 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16-17 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

16.2.9 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

16.2.10 函数 HAL_PWR_DisableSleepOnExit

描述了函数 HAL_PWR_DisableSleepOnExit

表16-18 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

16.2.11 函数 HAL_PWR_EnableSEVOnPend

描述了函数 HAL_PWR_EnableSEVOnPend

表16-19 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

16.2.12 函数 HAL_PWR_DisableSEVOnPend

描述了函数 HAL_PWR_DisableSEVOnPend

表16-20 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

16.2.13 函数 HAL_PWR_PVD_IRQHandler

描述了函数 HAL_PWR_PVD_IRQHandler

表16-21 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

16.2.14 函数 HAL_PWR_PVD_Callback

描述了函数 HAL_PWR_PVD_Callback

表16-22 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

17 HAL 复位和时钟通驱动程序（RCC）

RCC 有多种用途，包括时钟设置，外设复位和时钟管理。

17.1 RCC 固件驱动寄存器结构

17.1.1 RCC_PLLInitTypeDef

RCC_PLLInitTypeDef，定义于文件"py32f0xx_hal_rcc.h"如下：

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

字段说明：

表17-1 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17-2 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17-3 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

17.1.2 RCC_OscInitTypeDef

RCC_OscInitTypeDef，定义于文件"py32f0xx_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17-4 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17-5 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17-6 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17-7 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17-8 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17-9 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17-10 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17-11 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17-12　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17-13　LSIState 可选参数

参数	描述
RCC_LSI_OFF	关闭 LSI
RCC_LSI_ON	打开 LSI

17.1.3 RCC_ClkInitTypeDef

RCC_ClkInitTypeDef，定义于文件”py32f0xx_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17-14　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17-15 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17-16 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17-17 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17-18 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

17.2 RCC 固件库函数

表17-19 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

17.2.1 函数 HAL_RCC_DeInit

描述了函数 HAL_RCC_DeInit

表17-20 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

17.2.2 函数 HAL_RCC_OscConfig

描述了函数 HAL_RCC_OscConfig

表17-21 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

17.2.3 函数 HAL_RCC_ClockConfig

描述了函数 HAL_RCC_ClockConfig

表17-22 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17-23 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

17.2.4 函数 HAL_RCC_MCOConfig

描述了函数 HAL_RCC_MCOConfig

表17-24 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17-25 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17-26 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17-27 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

17.2.5 函数 HAL_RCC_EnableCSS

描述了函数 HAL_RCC_EnableCSS

表17-28 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

17.2.6 函数 HAL_RCC_EnableLSECSS

描述了函数 HAL_RCC_EnableLSECSS

表17-29 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

17.2.7 函数 HAL_RCC_DisableLSECSS

描述了函数 HAL_RCC_DisableLSECSS

表17-30 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

17.2.8 函数 HAL_RCC_GetSysClockFreq

描述了函数 HAL_RCC_GetSysClockFreq

表17-31 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

17.2.9 函数 HAL_RCC_GetHCLKFreq

描述了函数 HAL_RCC_GetHCLKFreq

表17-32 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

17.2.10 函数 HAL_RCC_GetPCLK1Freq

描述了函数 HAL_RCC_GetPCLK1Freq

表17-33 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

17.2.11 函数 HAL_RCC_GetOscConfig

描述了函数 HAL_RCC_GetOscConfig

表17-34 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

17.2.12 函数 HAL_RCC_GetClockConfig

描述了函数 HAL_RCC_GetClockConfig

表17-35 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

17.2.13 函数 HAL_RCC_NMI_IRQHandler

描述了函数 HAL_RCC_NMI_IRQHandler

表17-36 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

17.2.14 函数 HAL_RCC_CSSCallback

描述了函数 HAL_RCC_CSSCallback

表17-37 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

17.2.15 函数 HAL_RCC_LSECSSCallback

描述了函数 HAL_RCC_LSECSSCallback

表17-38　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

18 HAL 复位和时钟扩展驱动程序（RCC_Ex）

18.1 RCC_Ex 固件驱动寄存器结构

18.1.1 RCC_PeriphCLKInitTypeDef

RCC_PeriphCLKInitTypeDef，定义于文件"py32f0xx_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18-1 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18-2 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18-3 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18-4 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18-5 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18-6 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18-7 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

18.2 RCC_Ex 固件库函数

表18-8 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

18.2.1 函数 HAL_RCCEx_PeriphCLKConfig

描述了函数 HAL_RCCEx_PeriphCLKConfig

表18-9 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

18.2.2 函数 HAL_RCCEx_GetPeriphCLKConfig

描述了函数 HAL_RCCEx_GetPeriphCLKConfig

表18-10 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

18.2.3 函数 HAL_RCCEx_GetPeriphCLKFreq

描述了函数 HAL_RCCEx_GetPeriphCLKFreq

表18-11 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18-12 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

18.2.4 函数 HAL_RCCEx_EnableLSCO

描述了函数 HAL_RCCEx_EnableLSCO

表18-13 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18-14 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

18.2.5 函数 HAL_RCCEx_DisableLSCO

描述了函数 HAL_RCCEx_DisableLSCO

表18-15 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

19 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。

19.1 RTC 固件驱动寄存器结构

19.1.1 RTC_TimeTypeDef

RTC_TimeTypeDef，定义于文件"py32f0xx_hal_rtc.h"如下：

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

字段说明：

表19-1 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

19.1.2 RTC_AlarmTypeDef

RTC_AlarmTypeDef，定义于文件"py32f0xx_hal_rtc.h"如下：

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

字段说明：

表19-2 RTC_AlarmTypeDef 字段说明

字段	描述
AlarmTime	闹钟时间结构体

19.1.3 RTC_InitTypeDef

RTC_InitTypeDef，定义于文件"py32f0xx_hal_rtc.h"如下：

```
typedef struct
{
```

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

字段说明：

表19-3 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19-4 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

19.1.4 RTC_DateTypeDef

RTC_DateTypeDef，定义于文件”py32f0xx_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19-5 RTC_DateTypeDef 字段说明

字段	描述
WeekDay	周
Month	月
Date	日
Year	年

19.1.5 RTC_HandleTypeDef

RTC_HandleTypeDef，定义于文件"py32f0xx_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19-6 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

19.2 RTC 固件库函数

表19-7 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

19.2.1 函数 HAL_RTC_Init

描述了函数 HAL_RTC_Init

表19-8 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

19.2.2 函数 HAL_RTC_DeInit

描述了函数 HAL_RTC_DeInit

表19-9 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

19.2.3 函数 HAL_RTC_MspInit

描述了函数 HAL_RTC_MspInit

表19-10 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

19.2.4 函数 HAL_RTC_MspDeInit

描述了函数 HAL_RTC_MspDeInit

表19-11 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

19.2.5 函数 HAL_RTC_SetTime

描述了函数 HAL_RTC_SetTime

表19-12 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19-13 Format 可选参数

参数	描述
RTC_FORMAT_BIN	二进制格式
RTC_FORMAT_BCD	BCD 码格式

19.2.6 函数 HAL_RTC_GetTime

描述了函数 HAL_RTC_GetTime

表19-14 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19-15 Format 可选参数

参数	描述
RTC_FORMAT_BIN	二进制格式
RTC_FORMAT_BCD	BCD 码格式

19.2.7 函数 HAL_RTC_SetDate

描述了函数 HAL_RTC_SetDate

表19-16 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19-17 Format 可选参数

参数	描述
RTC_FORMAT_BIN	二进制格式
RTC_FORMAT_BCD	BCD 码格式

19.2.8 函数 HAL_RTC_GetDate

描述了函数 HAL_RTC_GetDate

表19-18 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19-19 Format 可选参数

参数	描述
RTC_FORMAT_BIN	二进制格式
RTC_FORMAT_BCD	BCD 码格式

19.2.9 函数 HAL_RTC_SetAlarm

描述了函数 HAL_RTC_SetAlarm

表19-20 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19-21 Format 可选参数

参数	描述
RTC_FORMAT_BIN	二进制格式
RTC_FORMAT_BCD	BCD 码格式

19.2.10 函数 HAL_RTC_SetAlarm_IT

描述了函数 HAL_RTC_SetAlarm_IT

表19-22 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19-23 Format 可选参数

参数	描述
RTC_FORMAT_BIN	二进制格式
RTC_FORMAT_BCD	BCD 码格式

19.2.11 函数 HAL_RTC_DeactivateAlarm

描述了函数 HAL_RTC_DeactivateAlarm

表19-24 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

19.2.12 函数 HAL_RTC_AlarmIRQHandler

描述了函数 HAL_RTC_AlarmIRQHandler

表19-25 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

19.2.13 函数 HAL_RTC_PollForAlarmAEvent

描述了函数 HAL_RTC_PollForAlarmAEvent

表19-26 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

19.2.14 函数 HAL_RTC_AlarmAEventCallback

描述了函数 HAL_RTC_AlarmAEventCallback

表19-27 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

19.2.15 函数 HAL_RTC_GetState

描述了函数 HAL_RTC_GetState

表19-28 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

19.2.16 函数 HAL_RTC_WaitForSynchro

描述了函数 HAL_RTC_WaitForSynchro

表19-29 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

20 HAL 实时时钟扩展驱动程序（RTC_Ex）

20.1 RTC_Ex 固件库函数

表20-1 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

20.1.1 函数 HAL_RTCEx_SetSecond_IT

描述了函数 HAL_RTCEx_SetSecond_IT

表20-2 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

20.1.2 函数 HAL_RTCEx_DeactivateSecond

描述了函数 HAL_RTCEx_DeactivateSecond

表20-3 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

20.1.3 函数 HAL_RTCEx_RTCIRQHandler

描述了函数 HAL_RTCEx_RTCIRQHandler

表20-4 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

20.1.4 函数 HAL_RTCEx_RTCEventCallback

描述了函数 HAL_RTCEx_RTCEventCallback

表20-5 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

20.1.5 函数 HAL_RTCEx_RTCEventErrorCallback

描述了函数 HAL_RTCEx_RTCEventErrorCallback

表20-6 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

20.1.6 函数 HAL_RTCEx_SetSmoothCalib

描述了函数 HAL_RTCEx_SetSmoothCalib

表20-7 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

21 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。

21.1 SPI 固件驱动寄存器结构

21.1.1 SPI_InitTypeDef

SPI_InitTypeDef，定义于文件"py32f0xx_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21-1 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21-2 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21-3 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21-4 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21-5 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21-6 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21-7 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21-8 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21-9 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21-10 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

21.1.2 SPI_HandleTypeDef

SPI_HandleTypeDef，定义于文件"py32f0xx_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21-11　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

21.2　SPI 固件库函数

表21-12　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

21.2.1 函数 HAL_SPI_Init

描述了函数 HAL_SPI_Init

表21-13 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

21.2.2 函数 HAL_SPI_DeInit

描述了函数 HAL_SPI_DeInit

表21-14 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

21.2.3 函数 HAL_SPI_MspInit

描述了函数 HAL_SPI_MspInit

表21-15 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

21.2.4 函数 HAL_SPI_MspDeInit

描述了函数 HAL_SPI_MspDeInit

表21-16 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

21.2.5 函数 HAL_SPI_Transmit

描述了函数 HAL_SPI_Transmit

表21-17 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

21.2.6 函数 HAL_SPI_Receive

描述了函数 HAL_SPI_Receive

表21-18 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

21.2.7 函数 HAL_SPI_TransmitReceive

描述了函数 HAL_SPI_TransmitReceive

表21-19 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

21.2.8 函数 HAL_SPI_Transmit_IT

描述了函数 HAL_SPI_Transmit_IT

表21-20 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

21.2.9 函数 HAL_SPI_Receive_IT

描述了函数 HAL_SPI_Receive_IT

表21-21 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

21.2.10 函数 HAL_SPI_TransmitReceive_IT

描述了函数 HAL_SPI_TransmitReceive_IT

表21-22 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

21.2.11 函数 HAL_SPI_Transmit_DMA

描述了函数 HAL_SPI_Transmit_DMA

表21-23 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

21.2.12 函数 HAL_SPI_Receive_DMA

描述了函数 HAL_SPI_Receive_DMA

表21-24 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

21.2.13 函数 HAL_SPI_TransmitReceive_DMA

描述了函数 HAL_SPI_TransmitReceive_DMA

表21-25 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

21.2.14 函数 HAL_SPI_DMAPause

描述了函数 HAL_SPI_DMAPause

表21-26 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

21.2.15 函数 HAL_SPI_DMAResume

描述了函数 HAL_SPI_DMAResume

表21-27 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

21.2.16 函数 HAL_SPI_DMAStop

描述了函数 HAL_SPI_DMAStop

表21-28 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

21.2.17 函数 HAL_SPI_Abort

描述了函数 HAL_SPI_Abort

表21-29 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

21.2.18 函数 HAL_SPI_Abort_IT

描述了函数 HAL_SPI_Abort_IT

表21-30 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

21.2.19 函数 HAL_SPI_IRQHandler

描述了函数 HAL_SPI_IRQHandler

表21-31 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

21.2.20 函数 HAL_SPI_TxCpltCallback

描述了函数 HAL_SPI_TxCpltCallback

表21-32 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

21.2.21 函数 HAL_SPI_RxCpltCallback

描述了函数 HAL_SPI_RxCpltCallback

表21-33 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

21.2.22 函数 HAL_SPI_TxRxCpltCallback

描述了函数 HAL_SPI_TxRxCpltCallback

表21-34 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

21.2.23 函数 HAL_SPI_TxHalfCpltCallback

描述了函数 HAL_SPI_TxHalfCpltCallback

表21-35 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

21.2.24 函数 HAL_SPI_RxHalfCpltCallback

描述了函数 HAL_SPI_RxHalfCpltCallback

表21-36 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

21.2.25 函数 HAL_SPI_TxRxHalfCpltCallback

描述了函数 HAL_SPI_TxRxHalfCpltCallback

表21-37 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

21.2.26 函数 HAL_SPI_ErrorCallback

描述了函数 HAL_SPI_ErrorCallback

表21-38 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

21.2.27 函数 HAL_SPI_AbortCpltCallback

描述了函数 HAL_SPI_AbortCpltCallback

表21-39 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

21.2.28 函数 HAL_SPI_GetState

描述了函数 HAL_SPI_GetState

表21-40　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

21.2.29 函数 HAL_SPI_GetError

描述了函数 HAL_SPI_GetError

表21-41　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

22 HAL 定时器通用驱动程序（TIM）

TIM1 由 16 位被可编程分频器驱动的自动装载计数器组成。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）。

22.1 TIM 固件驱动寄存器结构

22.1.1 TIM_Base_InitTypeDef

TIM_Base_InitTypeDef，定义于文件"py32f0xx_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22-1 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22-2 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22-3 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22-4 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

22.1.2 TIM_OC_InitTypeDef

TIM_OC_InitTypeDef，定义于文件”py32f0xx_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22-5 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22-6 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22-7 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22-8 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22-9 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22-10 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22-11 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

22.1.3 TIM_OnePulse_InitTypeDef

TIM_OnePulse_InitTypeDef，定义于文件"py32f0xx_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22-12 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22-13 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22-14 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22-15 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22-16 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22-17 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22-18 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22-19 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。

22.1.4 TIM_IC_InitTypeDef

TIM_IC_InitTypeDef，定义于文件"py32f0xx_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22-20 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22-21 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22-22 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22-23 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

22.1.5 TIM_Encoder_InitTypeDef

TIM_Encoder_InitTypeDef，定义于文件"py32f0xx_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22-24 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22-25 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22-26 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22-27 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22-28 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22-29 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22-30 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22-31 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

22.1.6 TIM_ClockConfigTypeDef

TIM_ClockConfigTypeDef，定义于文件”py32f0xx_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22-32 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22-33 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22-34 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22-35 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

22.1.7 TIM_ClearInputConfigTypeDef

TIM_ClearInputConfigTypeDef，定义于文件”py32f0xx_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22-36 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22-37 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22-38 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22-39 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

22.1.8 TIM_MasterConfigTypeDef

TIM_MasterConfigTypeDef，定义于文件”py32f0xx_hal_tim.h”如下：

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

```
} TIM_MasterConfigTypeDef;
```

字段说明：

表22-40　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22-41　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22-42　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

22.1.9 TIM_SlaveConfigTypeDef

TIM_SlaveConfigTypeDef，定义于文件”py32f0xx_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22-43 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22-44 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22-45 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22-46 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22-47 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

22.1.10 TIM_BreakDeadTimeConfigTypeDef

TIM_BreakDeadTimeConfigTypeDef，定义于文件”py32f0xx_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22-48 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22-49 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22-50 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22-51 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22-52 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22-53 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22-54 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

22.1.11 TIM_HandleTypeDef

TIM_HandleTypeDef，定义于文件"py32f0xx_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22-55 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

22.2 TIM 固件库函数

表22-56 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

22.2.1 函数 HAL_TIM_Base_Init

描述了函数 HAL_TIM_Base_Init

表22-57 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

22.2.2 函数 HAL_TIM_Base_DeInit

描述了函数 HAL_TIM_Base_DeInit

表22-58 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

22.2.3 函数 HAL_TIM_Base_MspInit

描述了函数 HAL_TIM_Base_MspInit

表22-59 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

22.2.4 函数 HAL_TIM_Base_MspDeInit

描述了函数 HAL_TIM_Base_MspDeInit

表22-60 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

22.2.5 函数 HAL_TIM_Base_Start

描述了函数 HAL_TIM_Base_Start

表22-61 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

22.2.6 函数 HAL_TIM_Base_Stop

描述了函数 HAL_TIM_Base_Stop

表22-62 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

22.2.7 函数 HAL_TIM_Base_Start_IT

描述了函数 HAL_TIM_Base_Start_IT

表22-63 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

22.2.8 函数 HAL_TIM_Base_Stop_IT

描述了函数 HAL_TIM_Base_Stop_IT

表22-64 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

22.2.9 函数 HAL_TIM_Base_Start_DMA

描述了函数 HAL_TIM_Base_Start_DMA

表22-65 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

22.2.10 函数 HAL_TIM_Base_Stop_DMA

描述了函数 HAL_TIM_Base_Stop_DMA

表22-66 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

22.2.11 函数 HAL_TIM_OC_Init

描述了函数 HAL_TIM_OC_Init

表22-67 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

22.2.12 函数 HAL_TIM_OC_DeInit

描述了函数 HAL_TIM_OC_DeInit

表22-68 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

22.2.13 函数 HAL_TIM_OC_MspInit

描述了函数 HAL_TIM_OC_MspInit

表22-69 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

22.2.14 函数 HAL_TIM_OC_MspDeInit

描述了函数 HAL_TIM_OC_MspDeInit

表22-70 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

22.2.15 函数 HAL_TIM_OC_Start

描述了函数 HAL_TIM_OC_Start

表22-71 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22-72 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

22.2.16 函数 HAL_TIM_OC_Stop

描述了函数 HAL_TIM_OC_Stop

表22-73 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22-74 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

22.2.17 函数 HAL_TIM_OC_Start_IT

描述了函数 HAL_TIM_OC_Start_IT

表22-75 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22-76 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

22.2.18 函数 HAL_TIM_OC_Stop_IT

描述了函数 HAL_TIM_OC_Stop_IT

表22-77 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22-78 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

22.2.19 函数 HAL_TIM_OC_Start_DMA

描述了函数 HAL_TIM_OC_Start_DMA

表22-79 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22-80 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

22.2.20 函数 HAL_TIM_OC_Stop_DMA

描述了函数 HAL_TIM_OC_Stop_DMA

表22-81 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22-82 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

22.2.21 函数 HAL_TIM_PWM_Init

描述了函数 HAL_TIM_PWM_Init

表22-83 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

22.2.22 函数 HAL_TIM_PWM_DeInit

描述了函数 HAL_TIM_PWM_DeInit

表22-84 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

22.2.23 函数 HAL_TIM_PWM_MspInit

描述了函数 HAL_TIM_PWM_MspInit

表22-85 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

22.2.24 函数 HAL_TIM_PWM_MspDeInit

描述了函数 HAL_TIM_PWM_MspDeInit

表22-86 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

22.2.25 函数 HAL_TIM_PWM_Start

描述了函数 HAL_TIM_PWM_Start

表22-87 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22-88 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

22.2.26 函数 HAL_TIM_PWM_Stop

描述了函数 HAL_TIM_PWM_Stop

表22-89 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22-90 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

22.2.27 函数 HAL_TIM_PWM_Start_IT

描述了函数 HAL_TIM_PWM_Start_IT

表22-91 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22-92 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

22.2.28 函数 HAL_TIM_PWM_Stop_IT

描述了函数 HAL_TIM_PWM_Stop_IT

表22-93 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22-94 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

22.2.29 函数 HAL_TIM_PWM_Start_DMA

描述了函数 HAL_TIM_PWM_Start_DMA

表22-95 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22-96 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

22.2.30 函数 HAL_TIM_PWM_Stop_DMA

描述了函数 HAL_TIM_PWM_Stop_DMA

表22-97 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22-98 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

22.2.31 函数 HAL_TIM_IC_Init

描述了函数 HAL_TIM_IC_Init

表22-99 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

22.2.32 函数 HAL_TIM_IC_DeInit

描述了函数 HAL_TIM_IC_DeInit

表22-100 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

22.2.33 函数 HAL_TIM_IC_MspInit

描述了函数 HAL_TIM_IC_MspInit

表22-101 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

22.2.34 函数 HAL_TIM_IC_MspDeInit

描述了函数 HAL_TIM_IC_MspDeInit

表22-102 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

22.2.35 函数 HAL_TIM_IC_Start

描述了函数 HAL_TIM_IC_Start

表22-103 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22-104 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

22.2.36 函数 HAL_TIM_IC_Stop

描述了函数 HAL_TIM_IC_Stop

表22-105 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

22.2.37 函数 HAL_TIM_IC_Start_IT

描述了函数 HAL_TIM_IC_Start_IT

表22-106 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

22.2.38 函数 HAL_TIM_IC_Stop_IT

描述了函数 HAL_TIM_IC_Stop_IT

表22-107 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

22.2.39 函数 HAL_TIM_IC_Start_DMA

描述了函数 HAL_TIM_IC_Start_DMA

表22-108 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22-109 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

22.2.40 函数 HAL_TIM_IC_Stop_DMA

描述了函数 HAL_TIM_IC_Stop_DMA

表22-110 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22-111 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

22.2.41 函数 HAL_TIM_OnePulse_Init

描述了函数 HAL_TIM_OnePulse_Init

表22-112 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22-113 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

22.2.42 函数 HAL_TIM_OnePulse_DeInit

描述了函数 HAL_TIM_OnePulse_DeInit

表22-114 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

22.2.43 函数 HAL_TIM_OnePulse_MspInit

描述了函数 HAL_TIM_OnePulse_MspInit

表22-115 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

22.2.44 函数 HAL_TIM_OnePulse_MspDeInit

描述了函数 HAL_TIM_OnePulse_MspDeInit

表22-116 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

22.2.45 函数 HAL_TIM_OnePulse_Start

描述了函数 HAL_TIM_OnePulse_Start

表22-117 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22-118 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

22.2.46 函数 HAL_TIM_OnePulse_Stop

描述了函数 HAL_TIM_OnePulse_Stop

表22-119 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22-120 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

22.2.47 函数 HAL_TIM_OnePulse_Start_IT

描述了函数 HAL_TIM_OnePulse_Start_IT

表22-121 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22-122 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

22.2.48 函数 HAL_TIM_OnePulse_Stop_IT

描述了函数 HAL_TIM_OnePulse_Stop_IT

表22-123 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22-124 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

22.2.49 函数 HAL_TIM_Encoder_Init

描述了函数 HAL_TIM_Encoder_Init

表22-125 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

22.2.50 函数 HAL_TIM_Encoder_DeInit

描述了函数 HAL_TIM_Encoder_DeInit

表22-126 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

22.2.51 函数 HAL_TIM_Encoder_MspInit

描述了函数 HAL_TIM_Encoder_MspInit

表22-127 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

22.2.52 函数 HAL_TIM_Encoder_MspDeInit

描述了函数 HAL_TIM_Encoder_MspDeInit

表22-128 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

22.2.53 函数 HAL_TIM_Encoder_Start

描述了函数 HAL_TIM_Encoder_Start

表22-129 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22-130 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

22.2.54 函数 HAL_TIM_Encoder_Stop

描述了函数 HAL_TIM_Encoder_Stop

表22-131 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22-132 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

22.2.55 函数 HAL_TIM_Encoder_Start_IT

描述了函数 HAL_TIM_Encoder_Start_IT

表22-133 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22-134 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

22.2.56 函数 HAL_TIM_Encoder_Stop_IT

描述了函数 HAL_TIM_Encoder_Stop_IT

表22-135 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22-136 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

22.2.57 函数 HAL_TIM_Encoder_Start_DMA

描述了函数 HAL_TIM_Encoder_Start_DMA

表22-137 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22-138 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

22.2.58 函数 HAL_TIM_Encoder_Stop_DMA

描述了函数 HAL_TIM_Encoder_Stop_DMA

表22-139 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

22.2.59 函数 HAL_TIM_IRQHandler

描述了函数 HAL_TIM_IRQHandler

表22-140 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

22.2.60 函数 HAL_TIM_OC_ConfigChannel

描述了函数 HAL_TIM_OC_ConfigChannel

表22-141 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22-142 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

22.2.61 函数 HAL_TIM_PWM_ConfigChannel

描述了函数 HAL_TIM_PWM_ConfigChannel

表22-143 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22-144 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

22.2.62 函数 HAL_TIM_IC_ConfigChannel

描述了函数 HAL_TIM_IC_ConfigChannel

表22-145 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22-146 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

22.2.63 函数 HAL_TIM_OnePulse_ConfigChannel

描述了函数 HAL_TIM_OnePulse_ConfigChannel

表22-147 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22-148 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22-149 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

22.2.64 函数 HAL_TIM_ConfigOCrefClear

描述了函数 HAL_TIM_ConfigOCrefClear

表22-150 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22-151 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

22.2.65 函数 HAL_TIM_ConfigClockSource

描述了函数 HAL_TIM_ConfigClockSource

表22-152 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

22.2.66 函数 HAL_TIM_ConfigTI1Input

描述了函数 HAL_TIM_ConfigTI1Input

表22-153 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22-154 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

22.2.67 函数 HAL_TIM_SlaveConfigSynchro

描述了函数 HAL_TIM_SlaveConfigSynchro

表22-155 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

22.2.68 函数 HAL_TIM_SlaveConfigSynchro_IT

描述了函数 HAL_TIM_SlaveConfigSynchro_IT

表22-156 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

22.2.69 函数 HAL_TIM_DMABurst_WriteStart

描述了函数 HAL_TIM_DMABurst_WriteStart

表22-157 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22-158 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22-159 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

22.2.70 函数 HAL_TIM_DMABurst_MultiWriteStart

描述了函数 HAL_TIM_DMABurst_MultiWriteStart

表22-160 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22-161 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22-162 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

22.2.71 函数 HAL_TIM_DMABurst_WriteStop

描述了函数 HAL_TIM_DMABurst_WriteStop

表22-163 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22-164 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

22.2.72 函数 HAL_TIM_DMABurst_ReadStart

描述了函数 HAL_TIM_DMABurst_ReadStart

表22-165 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22-166 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22-167 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

22.2.73 函数 HAL_TIM_DMABurst_MultiReadStart

描述了函数 HAL_TIM_DMABurst_MultiReadStart

表22-168 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22-169 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22-170 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

22.2.74 函数 HAL_TIM_DMABurst_ReadStop

描述了函数 HAL_TIM_DMABurst_ReadStop

表22-171 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22-172 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

22.2.75 函数 HAL_TIM_GenerateEvent

描述了函数 HAL_TIM_GenerateEvent

表22-173 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22-174 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

22.2.76 函数 HAL_TIM_ReadCapturedValue

描述了函数 HAL_TIM_ReadCapturedValue

表22-175 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22-176 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

22.2.77 函数 HAL_TIM_PeriodElapsedCallback

描述了函数 HAL_TIM_PeriodElapsedCallback

表22-177 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

22.2.78 函数 HAL_TIM_PeriodElapsedHalfCpltCallback

描述了函数 HAL_TIM_PeriodElapsedHalfCpltCallback

表22-178 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

22.2.79 函数 HAL_TIM_OC_DelayElapsedCallback

描述了函数 HAL_TIM_OC_DelayElapsedCallback

表22-179 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

22.2.80 函数 HAL_TIM_IC_CaptureCallback

描述了函数 HAL_TIM_IC_CaptureCallback

表22-180 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

22.2.81 函数 HAL_TIM_IC_CaptureHalfCpltCallback

描述了函数 HAL_TIM_IC_CaptureHalfCpltCallback

表22-181 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

22.2.82 函数 HAL_TIM_PWM_PulseFinishedCallback

描述了函数 HAL_TIM_PWM_PulseFinishedCallback

表22-182 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

22.2.83 函数 HAL_TIM_PWM_PulseFinishedHalfCpltCallback

描述了函数 HAL_TIM_PWM_PulseFinishedHalfCpltCallback

表22-183 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

22.2.84 函数 HAL_TIM_TriggerCallback

描述了函数 HAL_TIM_TriggerCallback

表22-184 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

22.2.85 函数 HAL_TIM_TriggerHalfCpltCallback

描述了函数 HAL_TIM_TriggerHalfCpltCallback

表22-185 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

22.2.86 函数 HAL_TIM_ErrorCallback

描述了函数 HAL_TIM_ErrorCallback

表22-186 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

22.2.87 函数 HAL_TIM_Base_GetState

描述了函数 HAL_TIM_Base_GetState

表22-187 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

22.2.88 函数 HAL_TIM_OC_GetState

描述了函数 HAL_TIM_OC_GetState

表22-188 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

22.2.89 函数 HAL_TIM_PWM_GetState

描述了函数 HAL_TIM_PWM_GetState

表22-189 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

22.2.90 函数 HAL_TIM_IC_GetState

描述了函数 HAL_TIM_IC_GetState

表22-190 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

22.2.91 函数 HAL_TIM_OnePulse_GetState

描述了函数 HAL_TIM_OnePulse_GetState

表22-191 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

22.2.92 函数 HAL_TIM_Encoder_GetState

描述了函数 HAL_TIM_Encoder_GetState

表22-192 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

23 HAL 定时器扩展驱动程序（TIM_Ex）

23.1 TIM_Ex 固件驱动寄存器结构

23.1.1 TIM_HallSensor_InitTypeDef

TIM_HallSensor_InitTypeDef，定义于文件”py32f0xx_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23-1 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23-2 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23-3 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

23.2 TIM_Ex 固件库函数

表23-4 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

23.2.1 函数 HAL_TIMEx_HallSensor_Init

描述了函数 HAL_TIMEx_HallSensor_Init

表23-5 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

23.2.2 函数 HAL_TIMEx_HallSensor_DeInit

描述了函数 HAL_TIMEx_HallSensor_DeInit

表23-6 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

23.2.3 函数 HAL_TIMEx_HallSensor_MspInit

描述了函数 HAL_TIMEx_HallSensor_MspInit

表23-7 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

23.2.4 函数 HAL_TIMEx_HallSensor_MspDeInit

描述了函数 HAL_TIMEx_HallSensor_MspDeInit

表23-8 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

23.2.5 函数 HAL_TIMEx_HallSensor_Start

描述了函数 HAL_TIMEx_HallSensor_Start

表23-9 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

23.2.6 函数 HAL_TIMEx_HallSensor_Stop

描述了函数 HAL_TIMEx_HallSensor_Stop

表23-10 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

23.2.7 函数 HAL_TIMEx_HallSensor_Start_IT

描述了函数 HAL_TIMEx_HallSensor_Start_IT

表23-11 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

23.2.8 函数 HAL_TIMEx_HallSensor_Stop_IT

描述了函数 HAL_TIMEx_HallSensor_Stop_IT

表23-12 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

23.2.9 函数 HAL_TIMEx_HallSensor_Start_DMA

描述了函数 HAL_TIMEx_HallSensor_Start_DMA

表23-13 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

23.2.10 函数 HAL_TIMEx_HallSensor_Stop_DMA

描述了函数 HAL_TIMEx_HallSensor_Stop_DMA

表23-14 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

23.2.11 函数 HAL_TIMEx_OCN_Start

描述了函数 HAL_TIMEx_OCN_Start

表23-15 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23-16 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

23.2.12 函数 HAL_TIMEx_OCN_Stop

描述了函数 HAL_TIMEx_OCN_Stop

表23-17 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23-18 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

23.2.13 函数 HAL_TIMEx_OCN_Start_IT

描述了函数 HAL_TIMEx_OCN_Start_IT

表23-19 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23-20 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

23.2.14 函数 HAL_TIMEx_OCN_Stop_IT

描述了函数 HAL_TIMEx_OCN_Stop_IT

表23-21 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23-22 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

23.2.15 函数 HAL_TIMEx_OCN_Start_DMA

描述了函数 HAL_TIMEx_OCN_Start_DMA

表23-23 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23-24 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

23.2.16 函数 HAL_TIMEx_OCN_Stop_DMA

描述了函数 HAL_TIMEx_OCN_Stop_DMA

表23-25 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23-26 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

23.2.17 函数 HAL_TIMEx_PWMN_Start

描述了函数 HAL_TIMEx_PWMN_Start

表23-27 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23-28　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

23.2.18 函数 HAL_TIMEx_PWMN_Stop

描述了函数 HAL_TIMEx_PWMN_Stop

表23-29　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23-30　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

23.2.19 函数 HAL_TIMEx_PWMN_Start_IT

描述了函数 HAL_TIMEx_PWMN_Start_IT

表23-31　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23-32 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

23.2.20 函数 HAL_TIMEx_PWMN_Stop_IT

描述了函数 HAL_TIMEx_PWMN_Stop_IT

表23-33 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23-34 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

23.2.21 函数 HAL_TIMEx_PWMN_Start_DMA

描述了函数 HAL_TIMEx_PWMN_Start_DMA

表23-35 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23-36 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

23.2.22 函数 HAL_TIMEx_PWMN_Stop_DMA

描述了函数 HAL_TIMEx_PWMN_Stop_DMA

表23-37 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23-38 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

23.2.23 函数 HAL_TIMEx_OnePulseN_Start

描述了函数 HAL_TIMEx_OnePulseN_Start

表23-39 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23-40 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

23.2.24 函数 HAL_TIMEx_OnePulseN_Stop

描述了函数 HAL_TIMEx_OnePulseN_Stop

表23-41 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23-42 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

23.2.25 函数 HAL_TIMEx_OnePulseN_Start_IT

描述了函数 HAL_TIMEx_OnePulseN_Start_IT

表23-43 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23-44 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

23.2.26 函数 HAL_TIMEx_OnePulseN_Stop_IT

描述了函数 HAL_TIMEx_OnePulseN_Stop_IT

表23-45 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23-46 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

23.2.27 函数 HAL_TIMEx_ConfigCommutEvent

描述了函数 HAL_TIMEx_ConfigCommutEvent

表23-47　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23-48　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23-49　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

23.2.28 函数 HAL_TIMEx_ConfigCommutEvent_IT

描述了函数 HAL_TIMEx_ConfigCommutEvent_IT

表23-50　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23-51 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23-52 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

23.2.29 函数 HAL_TIMEx_ConfigCommutEvent_DMA

描述了函数 HAL_TIMEx_ConfigCommutEvent_DMA

表23-53 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23-54 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23-55 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

23.2.30 函数 HAL_TIMEx_MasterConfigSynchronization

描述了函数 HAL_TIMEx_MasterConfigSynchronization

表23-56 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

23.2.31 函数 HAL_TIMEx_ConfigBreakDeadTime

描述了函数 HAL_TIMEx_ConfigBreakDeadTime

表23-57 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

23.2.32 函数 HAL_TIMEx_RemapConfig

描述了函数 HAL_TIMEx_RemapConfig

表23-58 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23-59 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

23.2.33 函数 HAL_TIMEx_CommutCallback

描述了函数 HAL_TIMEx_CommutCallback

表23-60 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

23.2.34 函数 HAL_TIMEx_CommutHalfCpltCallback

描述了函数 HAL_TIMEx_CommutHalfCpltCallback

表23-61 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

23.2.35 函数 HAL_TIMEx_BreakCallback

描述了函数 HAL_TIMEx_BreakCallback

表23-62 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

23.2.36 函数 HAL_TIM_StateTypeDef

描述了函数 HAL_TIM_StateTypeDef

表23-63 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

24 HAL 异步收发器通用驱动程序（UART）

24.1 UART 固件驱动寄存器结构

24.1.1 UART_InitTypeDef

UART_InitTypeDef，定义于文件”py32f0xx_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24-1 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24-2 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24-3 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24-4 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24-5 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24-6 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24-7 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

24.1.2 UART_AdvFeatureInitTypeDef

UART_AdvFeatureInitTypeDef，定义于文件”py32f0xx_hal_uart.h”如下：

```
typedef struct
{
uint32_t AdvFeatureInit;
uint32_t AutoBaudRateEnable;
```

```
uint32_t AutoBaudRateMode;
} UART_AdvFeatureInitTypeDef;
```

字段说明：

表24-8 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24-9 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24-10 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24-11 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

24.1.3 UART_HandleTypeDef

UART_HandleTypeDef，定义于文件”py32f0xx_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24-12 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

24.2 UART 固件库函数

表24-13 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

24.2.1 函数 HAL_UART_Init

描述了函数 HAL_UART_Init

表24-14 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

24.2.2 函数 HAL_HalfDuplex_Init

描述了函数 HAL_HalfDuplex_Init

表24-15 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

24.2.3 函数 HAL_MultiProcessor_Init

描述了函数 HAL_MultiProcessor_Init

表24-16 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

24.2.4 函数 HAL_UART_DeInit

描述了函数 HAL_UART_DeInit

表24-17 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

24.2.5 函数 HAL_UART_MspInit

描述了函数 HAL_UART_MspInit

表24-18 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

24.2.6 函数 HAL_UART_MspDeInit

描述了函数 HAL_UART_MspDeInit

表24-19 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

24.2.7 函数 HAL_UART_Transmit

描述了函数 HAL_UART_Transmit

表24-20 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

24.2.8 函数 HAL_UART_Receive

描述了函数 HAL_UART_Receive

表24-21 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

24.2.9 函数 HAL_UART_Transmit_IT

描述了函数 HAL_UART_Transmit_IT

表24-22 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

24.2.10 函数 HAL_UART_Receive_IT

描述了函数 HAL_UART_Receive_IT

表24-23 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

24.2.11 函数 HAL_UART_Transmit_DMA

描述了函数 HAL_UART_Transmit_DMA

表24-24 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

24.2.12 函数 HAL_UART_Receive_DMA

描述了函数 HAL_UART_Receive_DMA

表24-25 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

24.2.13 函数 HAL_UART_DMAPause

描述了函数 HAL_UART_DMAPause

表24-26 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

24.2.14 函数 HAL_UART_DMAResume

描述了函数 HAL_UART_DMAResume

表24-27 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

24.2.15 函数 HAL_UART_DMAStop

描述了函数 HAL_UART_DMAStop

表24-28 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

24.2.16 函数 HAL_UART_Abort

描述了函数 HAL_UART_Abort

表24-29 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

24.2.17 函数 HAL_UART_AbortTransmit

描述了函数 HAL_UART_AbortTransmit

表24-30 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

24.2.18 函数 HAL_UART_AbortReceive

描述了函数 HAL_UART_AbortReceive

表24-31 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

24.2.19 函数 HAL_UART_Abort_IT

描述了函数 HAL_UART_Abort_IT

表24-32 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

24.2.20 函数 HAL_UART_AbortTransmit_IT

描述了函数 HAL_UART_AbortTransmit_IT

表24-33 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

24.2.21 函数 HAL_UART_AbortReceive_IT

描述了函数 HAL_UART_AbortReceive_IT

表24-34 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

24.2.22 函数 HAL_UART_IRQHandler

描述了函数 HAL_UART_IRQHandler

表24-35 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

24.2.23 函数 HAL_UART_TxCpltCallback

描述了函数 HAL_UART_TxCpltCallback

表24-36 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

24.2.24 函数 HAL_UART_TxHalfCpltCallback

描述了函数 HAL_UART_TxHalfCpltCallback

表24-37 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

24.2.25 函数 HAL_UART_RxCpltCallback

描述了函数 HAL_UART_RxCpltCallback

表24-38 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

24.2.26 函数 HAL_UART_RxHalfCpltCallback

描述了函数 HAL_UART_RxHalfCpltCallback

表24-39 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

24.2.27 函数 HAL_UART_ErrorCallback

描述了函数 HAL_UART_ErrorCallback

表24-40 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

24.2.28 函数 HAL_UART_AbortCpltCallback

描述了函数 HAL_UART_AbortCpltCallback

表24-41 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

24.2.29 函数 HAL_UART_AbortTransmitCpltCallback

描述了函数 HAL_UART_AbortTransmitCpltCallback

表24-42 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

24.2.30 函数 HAL_UART_AbortReceiveCpltCallback

描述了函数 HAL_UART_AbortReceiveCpltCallback

表24-43 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

24.2.31 函数 HAL_UART_SendBreak

描述了函数 HAL_UART_SendBreak

表24-44 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

24.2.32 函数 HAL_MultiProcessor_EnterMuteMode

描述了函数 HAL_MultiProcessor_EnterMuteMode

表24-45 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

24.2.33 函数 HAL_MultiProcessor_ExitMuteMode

描述了函数 HAL_MultiProcessor_ExitMuteMode

表24-46 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

24.2.34 函数 HAL_HalfDuplex_EnableTransmitter

描述了函数 HAL_HalfDuplex_EnableTransmitter

表24-47 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

24.2.35 函数 HAL_HalfDuplex_EnableReceiver

描述了函数 HAL_HalfDuplex_EnableReceiver

表24-48 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

24.2.36 函数 HAL_UART_GetState

描述了函数 HAL_UART_GetState

表24-49 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

24.2.37 函数 HAL_UART_GetError

描述了函数 HAL_UART_GetError

表24-50 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

25 HAL 同步异步收发器通用驱动程序（USART）

通用同步异步收发器(USART)提供了一种灵活的方法与使用工业标准 NRZ 异步串行数据格式的外部设备之间进行全双工数据交换。USART 利用分数波特率发生器提供宽范围的波特率选择。

25.1 USART 固件驱动寄存器结构

25.1.1 USART_InitTypeDef

USART_InitTypeDef，定义于文件”py32f0xx_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25-1 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25-2 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25-3 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25-4 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25-5 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25-6 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25-7 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25-8 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

25.1.2 USART_HandleTypeDef

USART_HandleTypeDef，定义于文件"py32f0xx_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25-9 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

25.2 USART 固件库函数

表25-10 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

25.2.1 函数 HAL_USART_Init

描述了函数 HAL_USART_Init

表25-11 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

25.2.2 函数 HAL_USART_DeInit

描述了函数 HAL_USART_DeInit

表25-12 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

25.2.3 函数 HAL_USART_MspInit

描述了函数 HAL_USART_MspInit

表25-13 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

25.2.4 函数 HAL_USART_MspDeInit

描述了函数 HAL_USART_MspDeInit

表25-14 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

25.2.5 函数 HAL_USART_Transmit

描述了函数 HAL_USART_Transmit

表25-15 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

25.2.6 函数 HAL_USART_Receive

描述了函数 HAL_USART_Receive

表25-16 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

25.2.7 函数 HAL_USART_TransmitReceive

描述了函数 HAL_USART_TransmitReceive

表25-17 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

25.2.8 函数 HAL_USART_Transmit_IT

描述了函数 HAL_USART_Transmit_IT

表25-18 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

25.2.9 函数 HAL_USART_Receive_IT

描述了函数 HAL_USART_Receive_IT

表25-19 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

25.2.10 函数 HAL_USART_TransmitReceive_IT

描述了函数 HAL_USART_TransmitReceive_IT

表25-20 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

25.2.11 函数 HAL_USART_Transmit_DMA

描述了函数 HAL_USART_Transmit_DMA

表25-21 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

25.2.12 函数 HAL_USART_Receive_DMA

描述了函数 HAL_USART_Receive_DMA

表25-22 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

25.2.13 函数 HAL_USART_TransmitReceive_DMA

描述了函数 HAL_USART_TransmitReceive_DMA

表25-23 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

25.2.14 函数 HAL_USART_DMAPause

描述了函数 HAL_USART_DMAPause

表25-24 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

25.2.15 函数 HAL_USART_DMAResume

描述了函数 HAL_USART_DMAResume

表25-25 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

25.2.16 函数 HAL_USART_DMAStop

描述了函数 HAL_USART_DMAStop

表25-26 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

25.2.17 函数 HAL_USART_Abort

描述了函数 HAL_USART_Abort

表25-27 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

25.2.18 函数 HAL_USART_Abort_IT

描述了函数 HAL_USART_Abort_IT

表25-28 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

25.2.19 函数 HAL_USART_IRQHandler

描述了函数 HAL_USART_IRQHandler

表25-29 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

25.2.20 函数 HAL_USART_TxCpltCallback

描述了函数 HAL_USART_TxCpltCallback

表25-30 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

25.2.21 函数 HAL_USART_TxHalfCpltCallback

描述了函数 HAL_USART_TxHalfCpltCallback

表25-31 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

25.2.22 函数 HAL_USART_RxCpltCallback

描述了函数 HAL_USART_RxCpltCallback

表25-32　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

25.2.23 函数 HAL_USART_RxHalfCpltCallback

描述了函数 HAL_USART_RxHalfCpltCallback

表25-33　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

25.2.24 函数 HAL_USART_TxRxCpltCallback

描述了函数 HAL_USART_TxRxCpltCallback

表25-34　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

25.2.25 函数 HAL_USART_ErrorCallback

描述了函数 HAL_USART_ErrorCallback

表25-35 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

25.2.26 函数 HAL_USART_AbortCpltCallback

描述了函数 HAL_USART_AbortCpltCallback

表25-36 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

25.2.27 函数 HAL_USART_GetState

描述了函数 HAL_USART_GetState

表25-37 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

25.2.28 函数 HAL_USART_GetError

描述了函数 HAL_USART_GetError

表25-38　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

26 HAL 窗口看门狗通用驱动程序（WWDG）

System window watchdog(WWDG)被用作发现一个软件故障的出现。通常这类软件故障产生于外部干扰，或者未预测到的逻辑条件，该类故障引起应用程序终止正常的操作顺序。

26.1 WWDG 固件驱动寄存器结构

26.1.1 WWDG_InitTypeDef

WWDG_InitTypeDef，定义于文件"py32f0xx_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26-1 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26-2 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26-3 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

26.1.2 WWDG_HandleTypeDef

WWDG_HandleTypeDef，定义于文件"py32f0xx_hal_wwdg.h"如下：

```
typedef struct
{
WWDG_TypeDef *Instance;
WWDG_InitTypeDef Init;
} WWDG_HandleTypeDef;
```

字段说明：

表26-4 WWDG_HandleTypeDef 字段说明

字段	描述
Instance	WWDG 基地址
Init	初始化参数结构体

26.2 WWDG 固件库函数

表26-5 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

26.2.1 函数 HAL_WWDG_Init

描述了函数 HAL_WWDG_Init

表26-6 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

26.2.2 函数 HAL_WWDG_MspInit

描述了函数 HAL_WWDG_MspInit

表26-7 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

26.2.3 函数 HAL_WWDG_Refresh

描述了函数 HAL_WWDG_Refresh

表26-8 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

26.2.4 函数 HAL_WWDG_IRQHandler

描述了函数 HAL_WWDG_IRQHandler

表26-9 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

26.2.5 函数 HAL_WWDG_EarlyWakeupCallback

描述了函数 HAL_WWDG_EarlyWakeupCallback

表26-10 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

27 历史版本

版本	日期	更新记录
V1.0	2022.08.15	初版

Puya Semiconductor Ltd.