UM32x13x API 参考手册

版本：V1.0

广芯微电子（广州）股份有限公司

http://www.unicmicro.com/

条款协议

本文档的所有部分，其著作产权归广芯微电子（广州）股份有限公司（以下简称广芯微电子）所有，未经广芯微电子授权许可，任何个人及组织不得复制、转载、仿制本文档的全部或部分组件。本文档没有任何形式的担保、立场表达或其他暗示，若有任何因本文档或其中提及的产品所有资讯所引起的直接或间接损失，广芯微电子及所属员工恕不为其担保任何责任。除此以外，本文档所提到的产品规格及资讯仅供参考，内容亦会随时更新，恕不另行通知。

1. 本文档中所记载的关于电路、软件和其他相关信息仅用于说明半导体产品的操作和应用实例。用户如在设备设计中应用本文档中的电路、软件和相关信息，请自行负责。对于用户或第三方因使用上述电路、软件或信息而遭受的任何损失，广芯微电子不承担任何责任。
2. 在准备本文档所记载的信息的过程中，广芯微电子已尽量做到合理注意，但是，广芯微电子并不保证这些信息都是准确无误的。用户因本文档中所记载的信息的错误或遗漏而遭受的任何损失，广芯微电子不承担任何责任。
3. 对于因使用本文档中的广芯微电子产品或技术信息而造成的侵权行为或因此而侵犯第三方的专利、版权或其他知识产权的行为，广芯微电子不承担任何责任。本文档所记载的内容不应视为对广芯微电子或其他人所有的专利、版权或其他知识产权作出任何明示、默示或其它方式的许可及授权。
4. 使用本文档中记载的广芯微电子产品时，应在广芯微电子指定的范围内，特别是在最大额定值、电源工作电压范围、热辐射特性、安装条件以及其他产品特性的范围内使用。对于在上述指定范围之外使用广芯微电子产品而产生的故障或损失，广芯微电子不承担任何责任。
5. 虽然广芯微电子一直致力于提高广芯微电子产品的质量和可靠性，但是，半导体产品有其自身的具体特性，如一定的故障发生率以及在某些使用条件下会发生故障等。此外，广芯微电子产品均未进行防辐射设计。所以请采取安全保护措施，以避免当广芯微电子产品在发生故障而造成火灾时导致人身事故、伤害或损害的事故。例如进行软硬件安全设计（包括但不限于冗余设计、防火控制以及故障预防等）、适当的老化处理或其他适当的措施等。

目录

1 UM32x13x_HAL_Driver

1.1 模块

ADC： ADC HAL module driver

BTIMER： HAL BTIMER module driver

COMP： HAL COMP module driver

DMA： HAL DMA module driver

GTIMER： HAL GTIMER module driver

LIN： LIN HAL module driver

LPTIMER： LPTIMER HAL module driver

LPUART： HAL LPUART module driver

LVD： HAL LVD module driver

OPA： HAL OPA module driver

RTC： HAL RTC module driver

SPI： GPIO HAL module driver

UART： HAL UART module driver

UART1： UART1 HAL module driver

AES： AES HAL module driver

ATIMER： ATIMER HAL module driver

CAN： CAN HAL module driver

CRC： CRC HAL module driver

DIV： DIV HAL module driver

EEPROM： EEPROM HAL module driver

EFC： EFLASH HAL module driver

GPIO： GPIO HAL module driver

I2C： I2C HAL module driver

PWR： PWR HAL module driver

RCC： RCC HAL module driver

SYSTICK： SYSTICK HAL module driver

WDT： WDT HAL module driver

WWDT： WWDT HAL module driver

2 ADC

一个12位的ADC逐次接近型模数转换器，它具有多达16个输入通道，可测量来自14个外部源的信号、1个内部LDO输出和1个内部1/4 VDDH输出。这些通道的A/D转换可在单次或连续扫描模式下进行。 ADC控制器实现CPU和SAR ADC之间的通信。ADC转换的结果存储在数据寄存器的低12位。

2.1 ADC Exported Types

2.1.1 结构体

- struct **ADC_InitTypeDef**：Structure definition of ADC instance and ADC group regular.
- struct __ADC_HandleTypeDef

2.1.2 类型定义

- typedef struct **__ADC_HandleTypeDef ADC_HandleTypeDef**

2.2 ADC Exported Functions函数说明

2.2.1 Initialization and Configuration functions

2.2.1.1 HAL_StatusTypeDef HAL_ADC_Init (ADC_HandleTypeDef * *hadc*)

ADC initialization

参数

hadc	A pointer to a ADC_HandleTypeDef structure that contains configuration information for the specified ADC module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

2.2.2 ADC Input and Output operation functions

ADC IO operation functions

2.2.2.1 uint32_t HAL_ADC_GetValue (ADC_HandleTypeDef * *hadc*, uint32_t *Channel*)

Get ADC regular group conversion result.

参数

hadc	A pointer to a ADC_HandleTypeDef structure that contains configuration information for the specified ADC module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

2.2.2.2 HAL_StatusTypeDef HAL_ADC_PollForConversion (ADC_HandleTypeDef * *hadc*, uint32_t *Timeout*)

Wait for regular group conversion to be completed.

参数

hadc	A pointer to a ADC_HandleTypeDef structure that contains configuration information for the specified ADC module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

2.2.2.3 HAL_StatusTypeDef HAL_ADC_Start (ADC_HandleTypeDef * *hadc*)

ADC start conversion .

参数

hadc	A pointer to a ADC_HandleTypeDef structure that contains configuration information for the specified ADC module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

2.2.2.4 HAL_StatusTypeDef HAL_ADC_Start_IT (ADC_HandleTypeDef * *hadc*)

Start conversion of regular group with interruption.

参数

hadc	A pointer to a ADC_HandleTypeDef structure that contains configuration information for the specified ADC module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

2.2.2.5 HAL_StatusTypeDef HAL_ADC_Stop (ADC_HandleTypeDef * *hadc*)

ADC Stop conversion .

参数

hadc	A pointer to a ADC_HandleTypeDef structure that contains configuration information for the specified ADC module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

2.2.2.6 HAL_StatusTypeDef HAL_ADC_Stop_IT (ADC_HandleTypeDef * *hadc*)

Stop adc conversion with interruption.

参数

hadc	A pointer to a ADC_HandleTypeDef structure that contains configuration information for the specified ADC module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

2.2.2.7 __weak void HAL_ADC_ConvCpltCallback (ADC_HandleTypeDef * *hadc*)

Conversion complete callback in non-blocking mode.

参数

hadc	ADC handle

返回值

None	

2.2.2.8 void HAL_ADC_IRQHandler (ADC_HandleTypeDef * *hadc*)

Handle ADC interrupt request.

参数

hadc	ADC handle

返回值

None	

2.2.2.9 __weak void HAL_ADC_RxfifoCallback (ADC_HandleTypeDef * *hadc*)

Rxfifo callback in non-blocking mode.

参数

hadc	ADC handle

返回值

None	

2.2.2.10 __weak void HAL_ADC_RxfifofullCallback (ADC_HandleTypeDef * *hadc*)

Rxfifo full callback in non-blocking mode.

参数

hadc	ADC handle

返回值

None	

3 BTIMER

基本定时/计数器BTIMER，包含多种用途，16bit向上定时/计数器，产生输出PWM波形，脉冲输出，且计数值可以随时由软件设置和读取。

3.1 BTIMER Exported Types

3.1.1 结构体

- struct **BTIMER_HandleTypeDef**：BTIMER Handle structure definition
- struct **BTIMER_OC_InitTypeDef**：BTIMER OC Init structure definition
- struct **BTIMER_Base_InitTypeDef**：BTIMER BASE Init structure definition

3.2 BTIMER Exported Functions函数说明

3.2.1 HAL_StatusTypeDef BTIMER_Base_SetConfig (BTIMER_HandleTypeDef * *hbtimer*)

Configure the BTIMER Base Config ， such as ARR/PSC

参数

hbtimer	A pointer to a BTIMER_HandleTypeDef structure that contains configuration information for the specified BTIMER module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

3.2.2 HAL_StatusTypeDef BTIMER_IrqConfig (BTIMER_HandleTypeDef * *hbtimer*)

btimer interrupt config

参数

hbtimer	A pointer to a BTIMER_HandleTypeDef structure that contains configuration information for the specified BTIMER module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

3.2.3 HAL_StatusTypeDef BTIMER_OC_SetConfig (BTIMER_HandleTypeDef * *hbtimer*)

Config the BTIMER output

参数

hbtimer	A pointer to a BTIMER_HandleTypeDef structure that contains configuration information for the specified BTIMER module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

3.2.4 HAL_StatusTypeDef HAL_BTIMER_Base_Init (BTIMER_HandleTypeDef * *hbtimer*)

init the BTIMER Base

参数

hbtimer	A pointer to a BTIMER_HandleTypeDef structure that contains configuration information for the specified BTIMER module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

3.2.5 HAL_StatusTypeDef HAL_BTIMER_Base_Start (BTIMER_HandleTypeDef * *hbtimer*)

start btimer to work

参数

hbtimer	A pointer to a BTIMER_HandleTypeDef structure that contains configuration information for the specified BTIMER module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

3.2.6 HAL_StatusTypeDef HAL_BTIMER_Base_Stop (BTIMER_HandleTypeDef * *hbtimer*)

stop btimer

参数

hbtimer	A pointer to a BTIMER_HandleTypeDef structure that contains configuration information for the specified BTIMER module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

3.2.7 void HAL_BTIMER_IRQHandler (BTIMER_HandleTypeDef * *hbtimer*)

Reference interrupt function

参数

hbtimer	A pointer to a BTIMER_HandleTypeDef structure that contains configuration information for the specified BTIMER module

返回值

None	

3.2.8 HAL_StatusTypeDef HAL_BTIMER_OC_Config (BTIMER_HandleTypeDef * *hbtimer*)

call the BTIMER output Config

参数

hbtimer	A pointer to a BTIMER_HandleTypeDef structure that contains configuration information for the specified BTIMER module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

4 COMP

COMP是一款具有轨到轨输入的迟滞比较器，输入端可根据需要配置。COMP可用作电压比较，有两个输入端IN+和IN-，可选择其中一个输入端作为参考点来比较，当另一输入端电压小于参考电压时比较器输出低电平，反之输出高电平。

4.1 COMP Exported Types

4.1.1 结构体

- struct **COMP_InitTypeDef**：COMP Init structure definition
- **struct COMP_HandleTypeDef**

4.2 COMP Exported Functions函数说明

4.2.1 HAL_StatusTypeDef COMP_IrqConfig (COMP_HandleTypeDef * hcomp)

COMP interrupt configuration

参数

hcomp	Pointer to the COMP_HandleTypeDef structure that contains configuration information for the specified COMP module

返回

None

4.2.2 HAL_StatusTypeDef HAL_COMP_Init (COMP_HandleTypeDef * hcomp)

Initialize the COMP comparison function

参数

hcomp	Pointer to the COMP_HandleTypeDef structure that contains configuration information for the specified COMP module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

4.2.3 void HAL_COMP_IRQHandler (COMP_HandleTypeDef * *hcomp*)

Reference interrupt function

参数

hcomp	Pointer to the COMP_HandleTypeDef structure that contains configuration information for the specified COMP module

返回值

None	

4.2.4 HAL_StatusTypeDef HAL_COMP_Start (COMP_HandleTypeDef * *hcomp*)

Enable the comparator function

参数

hcomp	Pointer to the COMP_HandleTypeDef structure that contains configuration information for the specified COMP module

返回值

HAL	status information

4.2.5 HAL_StatusTypeDef HAL_COMP_Stop (COMP_HandleTypeDef * hcomp)

Disable the comparator function

参数

hcomp	Pointer to the COMP_HandleTypeDef structure that contains configuration information for the specified COMP module

返回值

HAL	status information

5 DMA

直接存储器访问(DMA)，支持2通道数据传输。

DMA可以协助CPU进行数据搬运的工作，减轻CPU的工作负担并提升系统效率。

5.1 DMA Exported Types

5.1.1 结构体

- struct **DMA_InitTypeDef**：DMA Init structure definition

5.2 DMA Exported Functions函数说明

5.2.1 HAL_StatusTypeDef DMA_Set_ITConfig (DMA_HandleTypeDef * *hdma*)

config dma interrupt

参数

hdma	A pointer to a DMA_HandleTypeDef structure that contains configuration information for the specified

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

5.2.2 HAL_StatusTypeDef DMA_SetConfig (DMA_HandleTypeDef * *hdma*)

dma Setting Config

参数

hdma	A pointer to a DMA_HandleTypeDef structure that contains configuration information for the specified

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

5.2.3 HAL_StatusTypeDef HAL_DMA_DeInit (DMA_HandleTypeDef * *hdma*)

disable dma

参数

hdma	A pointer to a DMA_HandleTypeDef structure that contains configuration information for the specified

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

5.2.4 uint8_t HAL_DMA_GetChxStatus (DMA_HandleTypeDef * *hdma*)

get dma channel

参数

hdma	A pointer to a DMA_HandleTypeDef structure that contains configuration information for the specified

返回值

dma	channel index

5.2.5 uint8_t HAL_DMA_GetITState (DMA_HandleTypeDef * *hdma*)

get interrupt state

参数

hdma	A pointer to a DMA_HandleTypeDef structure that contains configuration information for the specified

返回值

interrupt	status information

5.2.6 uint8_t HAL_DMA_GetPerReq (DMA_HandleTypeDef * *hdma*)

get dma request

参数

hdma	A pointer to a DMA_HandleTypeDef structure that contains configuration information for the specified

返回值

dma	request information

5.2.7 uint8_t HAL_DMA_GetTransferred_Length (DMA_HandleTypeDef * *hdma*)

get dma transmitted length

参数

hdma	A pointer to a DMA_HandleTypeDef structure that contains configuration information for the specified

返回值

dma	transmitted length

5.2.8 HAL_StatusTypeDef HAL_DMA_Init (DMA_HandleTypeDef * *hdma*)

dma initialization

参数

hdma	A pointer to a DMA_HandleTypeDef structure that contains configuration information for the specified

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

5.2.9 void HAL_DMA_IRQHandler (DMA_HandleTypeDef * *hdma*)

Reference interrupt function

参数

hdma	A pointer to a DMA_HandleTypeDef structure that contains configuration information for the specified DMA module

返回值

None	

5.2.10 HAL_StatusTypeDef HAL_DMA_Start (DMA_HandleTypeDef * *hdma*, uint32_t *SrcAddress*, uint32_t *DstAddress*, uint32_t *DataLength*)

start dma transmit

参数

hdma	A pointer to a DMA_HandleTypeDef structure that contains configuration information for the specified
SrcAddress	source address
DstAddress	dst address
DataLength	data len

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

6 GTIMER

通用定时/计数器GTIMER，每个定时器都有自己独立的中断。这些Timer可以有多种用途，包括测量输入信号的脉冲宽度（输入捕获），产生输出波形（PWM、带死区时间的互补PWM），计数器可以向上，向下，向上/下三种计数方向，且计数值可以随时由软件读取。每个Timer有2路PWM输出(可选是否互补)，有1路输入捕获。

6.1 GTIMER Exported Types

6.1.1 结构体

- struct **GTIMER_Base_InitTypeDef**：GTIMER Defines the basic timer initialization structure
- struct **GTIMER_OC_InitTypeDef**：GTIMER Indicates the output configuration of the channel
- struct **GTIMER_IC_InitTypeDef**：GTIMER Configures the input channel
- struct **GTIMER_BreakConfigTypeDef**：GTIMER Brake structure definition
- struct **GTIMER_HandleTypeDef**：GTIMER Handle structure definition

6.2 GTIMER Exported Functions函数说明

6.2.1 HAL_StatusTypeDef GTIMER_Break_Start (GTIMER_HandleTypeDef * *hgtimer*)

The GTIMER brake function was enabled

参数

hgtimer	Pointer to the GTIMER_HandleTypeDef structure that contains configuration information for the specified GTIMER module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

6.2.2 HAL_StatusTypeDef GTIMER_Break_Stop (GTIMER_HandleTypeDef * *hgtimer*)

Disable the GTIME braking function

参数

hgtimer	Pointer to the GTIMER_HandleTypeDef structure that contains configuration information for the specified GTIMER module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

6.2.3 HAL_StatusTypeDef GTIMER_ICChannel_Input_Start (GTIMER_HandleTypeDef * *hgtimer*)

GTIMER Enables the channel capture input

参数

hgtimer	Pointer to the GTIMER_HandleTypeDef structure that contains configuration information for the specified GTIMER module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

6.2.4 HAL_StatusTypeDef GTIMER_ICChannel_Input_Stop (GTIMER_HandleTypeDef * *hgtimer*)

GTIMER disables the channel from capturing input

参数

hgtimer	Pointer to the GTIMER_HandleTypeDef structure that contains configuration information for the specified GTIMER module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

6.2.5 HAL_StatusTypeDef GTIMER_IrqConfig (GTIMER_HandleTypeDef * *hgtimer*)

GTIMER Indicates the interrupt configuration

参数

hgtimer	Pointer to the GTIMER_HandleTypeDef structure that contains configuration information for the specified GTIMER module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

6.2.6 HAL_StatusTypeDef GTIMER_OCChannel_Output_Start (GTIMER_HandleTypeDef * *hgtimer*)

The GTIMER channel output comparison was enabled

参数

hgtimer	Pointer to the GTIMER_HandleTypeDef structure that contains configuration information for the specified GTIMER module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

6.2.7 HAL_StatusTypeDef GTIMER_OCChannel_Output_Stop (GTIMER_HandleTypeDef * *hgtimer*)

The GTIMER channel output comparison is disabled

参数

hgtimer	Pointer to the GTIMER_HandleTypeDef structure that contains configuration information for the specified GTIMER module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

6.2.8 HAL_StatusTypeDef HAL_GTIMER_Base_Init (GTIMER_HandleTypeDef * *hgtimer*)

GTIMER Indicates the initialization of the basic configuration timer

参数

hgtimer	Pointer to the GTIMER_HandleTypeDef structure that contains configuration information for the specified GTIMER module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

6.2.9 HAL_StatusTypeDef HAL_GTIMER_BREAK_Config (GTIMER_HandleTypeDef * *hgtimer*)

GTIMER brake configuration

参数

hgtimer	Pointer to the GTIMER_HandleTypeDef structure that contains configuration information for the specified GTIMER module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

6.2.10 void HAL_GTIMER_Counter_Start (GTIMER_HandleTypeDef * *hgtimer*)

Example Start the GTIMER counter

参数

hgtimer	Pointer to the GTIMER_HandleTypeDef structure that contains configuration information for the specified GTIMER module

返回

None

6.2.11 void HAL_GTIMER_Counter_Stop (GTIMER_HandleTypeDef * *hgtimer*)

Disable the GTIMER counter

参数

hgtimer	Pointer to the GTIMER_HandleTypeDef structure that contains configuration information for the specified GTIMER module

返回

None

6.2.12 HAL_StatusTypeDef HAL_GTIMER_IC_Config (GTIMER_HandleTypeDef * *hgtimer*)

The GTIMER channel output comparison is disabled

参数

hgtimer	Pointer to the GTIMER_HandleTypeDef structure that contains configuration information for the specified GTIMER module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

6.2.13 void HAL_GTIMER_IRQHandler (GTIMER_HandleTypeDef * *hgtimer*)

Reference interrupt function

参数

hgtimer	Pointer to the GTIMER_HandleTypeDef structure that contains configuration information for the specified GTIMER module

返回

None

6.2.14 HAL_StatusTypeDef HAL_GTIMER_OC_Config (GTIMER_HandleTypeDef * *hgtimer*)

GTIMER Indicates the output configuration of the channel

参数

hgtimer	Pointer to the GTIMER_HandleTypeDef structure that contains configuration information for the specified GTIMER module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong

HAL_ERROR	something wrong

7 LIN

LIN(Local Interconnect Network) 控制器可用于分布式汽车应用中机电一体化节点的控制，本模块支持LIN总线上的主节点和从节点的连接。

7.1 LIN Exported Types

7.1.1 结构体

- struct **LIN_InitTypeDef**：LIN Init Structure definition
- struct **__LIN_HandleTypeDef**：LIN handle Structure definition

7.1.2 类型定义

- typedef struct __LIN_HandleTypeDef LIN_HandleTypeDef
  LIN handle Structure definition

7.2 LIN Exported Functions函数说明

7.2.1 Initialization/de-initialization functions

Initialization and Configuration functions

7.2.1.1 HAL_StatusTypeDef HAL_LIN_Init (LIN_HandleTypeDef * *hlin*)

init

参数

hlin	LIN handle.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

7.2.1.2 HAL_StatusTypeDef LIN_BaudConfig (LIN_HandleTypeDef * *hlin*)

baudrate config

参数

hlin	LIN handle.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

7.2.1.3 HAL_StatusTypeDef LIN_IrqConfig (LIN_HandleTypeDef * *hlin*)

irq config

参数

hlin	LIN handle.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

7.2.1.4 HAL_StatusTypeDef LIN_SetConfig (LIN_HandleTypeDef * *hlin*)

set config

参数

hlin	LIN handle.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

7.2.2 IO operation functions

LIN Read, Write, Toggle, Lock and EXTI management functions.

7.2.2.1 void HAL_LIN_IRQHandler (LIN_HandleTypeDef * *hlin*)

lin irq handler function

参数

hlin	LIN handle.

返回

void

7.2.2.2 HAL_StatusTypeDef HAL_LIN_Master_Receive (LIN_HandleTypeDef * *hlin*, uint8_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

receive an amount of data in master mode.

参数

hlin	LIN handle.
pData	Pointer to data buffer .
Size	Amount of data elements to be received.
Timeout	Timeout duration.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

7.2.2.3 HAL_StatusTypeDef HAL_LIN_Master_Receive_IT (LIN_HandleTypeDef * *hlin*, uint8_t * *pData*, uint16_t *Size*, HAL_StatusTypeDef(*)() *recv_callback*)

Receive an amount of data in master interrupt mode.

参数

hlin	LIN handle.
pData	Pointer to data buffer
Size	Amount of data elements to be received.
recv_callback	receive irq callback

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

7.2.2.4 HAL_StatusTypeDef HAL_LIN_Master_Transmit (LIN_HandleTypeDef * *hlin*, uint8_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Send an amount of data in master mode.

参数

hlin	LIN handle.
pData	Pointer to data buffer .
Size	Amount of data elements to be sent.
Timeout	Timeout duration.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	transmit timeout

7.2.2.5 HAL_StatusTypeDef HAL_LIN_Master_Transmit_IT (LIN_HandleTypeDef * *hlin*, uint8_t * *pData*, uint16_t *Size*, HAL_StatusTypeDef(*)() *send_callback*)

Send an amount of data in master interrupt mode.

参数

hlin	LIN handle.
pData	Pointer to data buffer .
Size	Amount of data elements to be sent.
send_callback	send irq callback

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	send timeout

7.2.2.6 HAL_StatusTypeDef HAL_LIN_SetDataLen (LIN_HandleTypeDef * *hlin*, uint16_t *len*)

set length of receive data or transmit data

参数

hlin	LIN handle.
len	length of data

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong

7.2.2.7 HAL_StatusTypeDef HAL_LIN_SetTransMode (LIN_HandleTypeDef * *hlin*, uint32_t *Mode*)

set lin node mode tx or rx

参数

hlin	LIN handle.
Mode	tx or rx

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong

7.2.2.8 HAL_StatusTypeDef HAL_LIN_Slave_Receive (LIN_HandleTypeDef * *hlin*, uint8_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Receive an amount of data in slave mode.

参数

hlin	LIN handle.
pData	Pointer to data buffer.
Size	Amount of data elements to be received.
Timeout	Timeout duration.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	receive timeout

7.2.2.9 HAL_StatusTypeDef HAL_LIN_Slave_Receive_IT (LIN_HandleTypeDef * *hlin*, uint8_t * *pData*, uint16_t *Size*, HAL_StatusTypeDef(*)() *recv_callback*)

Receive an amount of data in slave interrupt mode.

参数

hlin	LIN handle.
pData	Pointer to data buffer
Size	Amount of data elements to be received.
recv_callback	receive irq callback

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

7.2.2.10 HAL_StatusTypeDef HAL_LIN_Slave_Transmit (LIN_HandleTypeDef * *hlin*, uint8_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Send an amount of data in slave mode.

参数

hlin	LIN handle.
pData	Pointer to data buffer.
Size	Amount of data elements to be sent.
Timeout	Timeout duration.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	transmit timeout

7.2.2.11 HAL_StatusTypeDef HAL_LIN_Slave_Transmit_IT (LIN_HandleTypeDef * *hlin*, uint8_t * *pData*, uint16_t *Size*, HAL_StatusTypeDef(*)() *send_callback*)

Send an amount of data in slave interrupt mode.

参数

hlin	LIN handle.
pData	Pointer to data buffer .
Size	Amount of data elements to be sent.
send_callback	send irq callback

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	send timeout

8 LPTIMER

LPTIMER是32位的低功耗定时/计数器模块。 由于其时钟源具有多样性，因此能够在所有电源模式下保持运行状态，并且只消耗很低的功耗。LPTIMER0和LPTIMER1可以在没有内部时钟的条件下工作，实现休眠模式下的外部脉冲计数功能，还可以与外部输入的触发信号结合，可以实现低功耗超时唤醒功能。

8.1 LPTIMER Exported Types

8.1.1 结构体

- struct **Lptimer_Base_InitTypeDef**：LPTIMER Init Structure definition
- struct **Lptimer_OC_InitTypeDef**：TIM Output Compare Configuration Structure definition
- struct **Lptimer_IC_InitTypeDef**：TIM Input Capture Configuration Structure definition
- struct **__LPTIMER_HandleTypeDef**：LPTIMER handle Structure definition

8.1.2 类型定义

- typedef struct **__LPTIMER_HandleTypeDef LPTIMER_HandleTypeDef**
  LPTIMER handle Structure definition

8.2 LPTIMER Exported Functions函数说明

8.2.1 Initialization/de-initialization functions

Initialization and Configuration functions

8.2.1.1 HAL_StatusTypeDef HAL_LPTIMER_Base_Init (LPTIMER_HandleTypeDef * *hlptimer*)

Initializes the TIM Time base Unit according to the specified parameters in the TIM_HandleTypeDef and initialize the associated handle.

参数

hlptimer	LPTIMER handle.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

8.2.1.2 HAL_StatusTypeDef HAL_LPTIMER_IC_Init (LPTIMER_HandleTypeDef * *hlptimer*)

input capture init

参数

hlptimer	LPTIMER handle.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

8.2.1.3 HAL_StatusTypeDef HAL_LPTIMER_OC_Init (LPTIMER_HandleTypeDef * *hlptimer*)

Output Compare init

参数

hlptimer	LPTIMER handle.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

8.2.1.4 HAL_StatusTypeDef LPTIMER_IrqConfig (LPTIMER_HandleTypeDef * *hlptimer*)

irq config

参数

hlptimer	LPTIMER handle.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

8.2.2 IO operation functions

LPTIMER Read, Write, Toggle, Lock and EXTI management functions.

8.2.2.1 void HAL_LPTIMER_IRQHandler (LPTIMER_HandleTypeDef * *hlptimer*)

irq handler function

参数

hlptimer	LPTIMER handle.

返回

void

9 LPUART

芯片低功耗串口模块LPUART，其工作仅需32kHz时钟，可以支持到最高9600波特率的数据接收。LPUART功耗极低，可以在Sleep/DeepSleep模式下工作。

9.1 LPUART Exported Types

9.1.1 结构体

- struct **LPUART_InitTypeDef**：LPUART Init structure definition
- struct **__LPUART_HandleTypeDef**：LPUART Handle structure definition

9.1.2 类型定义

- typedef struct **__LPUART_HandleTypeDef LPUART_HandleTypeDef**
  LPUART Handle structure definition

9.2 LPUART Exported Functions函数说明

9.2.1 HAL_StatusTypeDef HAL_LPUART_Init (LPUART_HandleTypeDef * *hlpuart*)

Initializes the LPUART with the specified argument in LPUART_InitTypeDef and creates the associated handle

参数

hlpuart	LPUART_HandleTypeDef pointer ● None

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

9.2.2 void HAL_LPUART_IRQHandler (LPUART_HandleTypeDef * *hlpuart*)

Reference interrupt function

参数

hlpuart	A pointer to a LPUART_HandleTypeDef structure that contains configuration information for the specified LPUART module

返回值

None	

9.2.3 HAL_StatusTypeDef HAL_LPUART_Receive (LPUART_HandleTypeDef * *hlpuart*, uint8_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Blocking receives multiple data

参数

hlpuart	A pointer to a LPUART_HandleTypeDef structure that contains configuration information for the specified LPUART module
pData	A pointer to the address to receive data
Size	The amount of data to be received
Timeout	Wait to receive timeout

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

9.2.4 HAL_StatusTypeDef HAL_LPUART_Receive_IT (LPUART_HandleTypeDef * *hlpuart*, uint8_t * *pData*, uint16_t *Size*, HAL_StatusTypeDef(*)() *recv_callback*)

Receive multiple data non-blocking

参数

hlpuart	A pointer to a LPUART_HandleTypeDef structure that contains onfiguration information for the specified LPUART module
pData	A pointer to the address to receive data
Size	The amount of data to be received
recv_callback	Receive the callback function

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

9.2.5 HAL_StatusTypeDef HAL_LPUART_Transmit (LPUART_HandleTypeDef * *hlpuart*, uint8_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Blocking transmission

参数

hlpuart	A pointer to a LPUART_HandleTypeDef structure that contains configuration information for the specified LPUART module
pData	Pointer to the address to send data
Size	The amount of data to be sent
Timeout	Timeout period for waiting for sending to complete

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

9.2.6 HAL_StatusTypeDef HAL_LPUART_Transmit_IT (LPUART_HandleTypeDef * *hlpuart*, uint8_t * *pData*, uint16_t *Size*, HAL_StatusTypeDef(*)() *send_callback*)

Sending multiple data in non-blocking mode

参数

hlpuart	A pointer to a LPUART_HandleTypeDef structure that contains configuration information for the specified LPUART module
pData	Pointer to the address to send data
Size	The amount of data to be sent
Timeout	Timeout period for waiting for sending to complete

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

9.2.7 HAL_StatusTypeDef LPUART_BaudConfig (LPUART_HandleTypeDef * *hlpuart*)

Configure the LPUART baud rate

参数

hlpuart	LPUART_HandleTypeDef pointer ● None

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

9.2.8 HAL_StatusTypeDef LPUART_IrqConfig (LPUART_HandleTypeDef * *hlpuart*)

Configure the LPUART interrupt

参数

hlpuart	LPUART_HandleTypeDef pointer ● None

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

9.2.9 HAL_StatusTypeDef LPUART_SetConfig (LPUART_HandleTypeDef * *hlpuart*)

Initializes the LPUART with the specified argument in LPUART_InitTypeDef

参数

hlpuart	LPUART_HandleTypeDef pointer ● None

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

10 LVD

10.1 LVD Exported Types

10.1.1 结构体

- struct **LVD_InitTypeDef**：LVD Init structure definition
- struct **__LVD_HandleTypeDef**

10.1.2 类型定义

- typedef struct **__LVD_HandleTypeDef LVD_HandleTypeDef**

10.2 LVD Exported Functions函数说明

10.2.1 __weak void HAL_LVD_Callback (LVD_HandleTypeDef * *hlvd*)

Conversion complete callback in non-blocking mode.

参数

hlvd	Pointer to the LVD_HandleTypeDef structure that contains configuration information for the specified LVD module

返回值

None	

10.2.2 HAL_StatusTypeDef HAL_LVD_Init (LVD_HandleTypeDef * *hlvd*)

Initialize the LVD comparison function

参数

hlvd	Pointer to the LVD_HandleTypeDef structure that contains configuration information for the specified LVD module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

10.2.3 void HAL_LVD_IRQHandler (LVD_HandleTypeDef * *hlvd*)

Reference interrupt function

参数

hlvd	Pointer to the LVD_HandleTypeDef structure that contains configuration information for the specified LVD module

返回值

None	

11 OPA

OPA是一款具有轨到轨输入和AB类输出级的运算放大器。输入输出端可以根据需要配置成不同连接。偏移电压可以被修调。

11.1 OPA Exported Types

11.1.1 结构体

- struct OPA_InitTypeDefstruct：OPA_HandleTypeDef

11.2 OPA Exported Functions

11.2.1 HAL_StatusTypeDef HAL_OPA_Init (OPA_HandleTypeDef * *hopa*)

Initializing the OPA function

参数

hopa	Pointer to the OPA_HandleTypeDef structure that contains configuration information for the specified OPA module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

11.2.2 void HAL_OPA_IRQHandler (OPA_HandleTypeDef * *hopa*)

Reference interrupt function

参数

hopa	Pointer to the OPA_HandleTypeDef structure that contains configuration information for the specified OPA module

返回

None

11.2.3 HAL_StatusTypeDef HAL_OPA_Start (OPA_HandleTypeDef * *hopa*)

OPA ENABLE

11.2.4 HAL_StatusTypeDef OPA_IrqConfig (OPA_HandleTypeDef * *hopa*)

Configure the OPA interrupt function

参数

hopa	Pointer to the OPA_HandleTypeDef structure that contains configuration information for the specified OPA module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

12 RTC

实时时钟（RTC）是一个独立的定时器/计数器，可提供基本的闹钟中断或者长时间的计数服务。闹钟中断通过可配置的实时时钟计数周期实现。

12.1 RTC Exported Types

12.1.1 结构体

- struct **s_calendar_obj**：RTC calendar structure definition
- struct **RTC_InitTypeDef**：RTC Init structure definition

12.2 RTC Exported Functions函数说明

12.2.1 HAL_StatusTypeDef HAL_rtc_alarm_SetConfig (RTC_HandleTypeDef * *hrtc*)

rtc calendar alarm reg config

参数

hrtc	A pointer to a RTC_HandleTypeDef structure that contains configuration information for the specified

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

12.2.2 HAL_StatusTypeDef HAL_rtc_calendar_alarm_data_SetConfig (RTC_HandleTypeDef * *hrtc*)

rtc calendar alarm data config

参数

hrtc	A pointer to a RTC_HandleTypeDef structure that contains configuration information for the specified

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

12.2.3 HAL_StatusTypeDef HAL_rtc_calendar_data_SetConfig (RTC_HandleTypeDef * *hrtc*)

rtc calendar data config

参数

hrtc	A pointer to a RTC_HandleTypeDef structure that contains configuration information for the specified

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

12.2.4 HAL_StatusTypeDef HAL_RTC_calendar_SetConfig (RTC_HandleTypeDef * *hrtc*)

rtc calendar reg config

参数

hrtc	A pointer to a RTC_HandleTypeDef structure that contains configuration information for the specified

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

12.2.5 HAL_StatusTypeDef HAL_rtc_fout (RTC_HandleTypeDef * *hrtc*)

rtc output config

参数

hrtc	A pointer to a RTC_HandleTypeDef structure that contains configuration information for the specified

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

12.2.6 void HAL_RTC_IRQHandler (RTC_HandleTypeDef * *hrtc*)

Reference interrupt function

参数

hrtc	A pointer to a RTC_HandleTypeDef structure that contains configuration information for the specified RTC module

返回值

None	

12.2.7 HAL_StatusTypeDef HAL_RTC_IT (RTC_HandleTypeDef * *hrtc*, uint8_t *irq_enable*, uint32_t *irq_type*)

rtc interrupt config

参数

hrtc	A pointer to a RTC_HandleTypeDef structure that contains configuration information for the specified

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

12.2.8 HAL_StatusTypeDef HAL_rtc_ltbc_SetConfig (RTC_HandleTypeDef * *hrtc*)

rtc ltbc config

参数

hrtc	A pointer to a RTC_HandleTypeDef structure that contains configuration information for the specified

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

12.2.9 HAL_StatusTypeDef HAL_rtc_PR1SEN_init (RTC_HandleTypeDef * *hrtc*)

rtc virtual alignment enable

参数

hrtc	A pointer to a RTC_HandleTypeDef structure that contains configuration information for the specified

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

12.2.10 HAL_StatusTypeDef HAL_rtc_stamp0_init (RTC_HandleTypeDef * *hrtc*)

rtc stamp0 enable

参数

hrtc	A pointer to a RTC_HandleTypeDef structure that contains configuration information for the specified

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

12.2.11 HAL_StatusTypeDef HAL_rtc_stamp1_init (RTC_HandleTypeDef * *hrtc*)

rtc stamp1 enable

参数

hrtc	A pointer to a RTC_HandleTypeDef structure that contains configuration information for the specified

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

13 SPI

串行外设接口（Serial Peripheral Interface，SPI）是外部设备通过单线交换数据的串行同步通讯手段。芯片提供了2个SPI接口模块，可配置为主设备或从设备，实现与外部的SPI通信。

13.1 SPI Exported Types

13.1.1 结构体

- struct **SPI_InitTypeDef：**SPI Configuration Structure definition
- struct **__SPI_HandleTypeDef：**SPI handle Structure definition

13.1.2 类型定义

- typedef struct __SPI_HandleTypeDef SPI_HandleTypeDef
  SPI handle Structure definition

13.1.3 枚举

- enum HAL_SPI_StateTypeDef { HAL_SPI_STATE_RESET = 0x00U, HAL_SPI_STATE_READY = 0x01U, HAL_SPI_STATE_BUSY = 0x02U, HAL_SPI_STATE_BUSY_TX = 0x03U, HAL_SPI_STATE_BUSY_RX = 0x04U, HAL_SPI_STATE_BUSY_TX_RX = 0x05U, HAL_SPI_STATE_ERROR = 0x06U, HAL_SPI_STATE_ABORT = 0x07U }
  HAL SPI State structure definition

13.2 SPI Exported Functions函数说明

13.2.1 Initialization/de-initialization functions

Initialization and Configuration functions

13.2.1.1 HAL_StatusTypeDef HAL_SPI_Init (SPI_HandleTypeDef * *hspi*)

参数

hspi	A pointer to a SPI_HandleTypeDef structure that contains configuration information for the specified SPI module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

13.2.1.2 HAL_StatusTypeDef HAL_SPI_SetConfig (SPI_HandleTypeDef * *hspi*)

set config,write register

参数

hspi	A pointer to a SPI_HandleTypeDef structure that contains configuration information for the specified SPI module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

13.2.1.3 HAL_StatusTypeDef SPI_IrqConfig (SPI_HandleTypeDef * *hspi*)

irq config

参数

hspi	A pointer to a SPI_HandleTypeDef structure that contains configuration information for the specified SPI module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

13.2.2 IO operation functions

SPI Read, Write, Toggle, Lock and EXTI management functions.

13.2.2.1 void HAL_SPI_IRQHandler (SPI_HandleTypeDef * *hspi*)

SPI irq function

参数

hspi	A pointer to a SPI_HandleTypeDef structure that contains configuration information for the specified SPI module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

13.2.2.2 HAL_StatusTypeDef HAL_SPI_Master_Receive_DC (SPI_HandleTypeDef * *hspi*, uint32_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Receive an amount of data in blocking mode.

参数

hspi	A pointer to a SPI_HandleTypeDef structure that contains configuration information for the specified SPI module
pData	A pointer to the address to receive data
Size	The amount of data to be received
Timeout	Wait to receive timeout

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	receive timeout

13.2.2.3 HAL_StatusTypeDef HAL_SPI_Master_Transmit_DC (SPI_HandleTypeDef * *hspi*, uint32_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Send an amount of data in blocking mode.

参数

hspi	A pointer to a SPI_HandleTypeDef structure that contains configuration information for the specified SPI module
pData	Pointer to the address to send data
Size	The amount of data to be sent
Timeout	Timeout period for waiting for sending to complete

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	transmit timeout

13.2.2.4 HAL_StatusTypeDef HAL_SPI_Receive (SPI_HandleTypeDef * *hspi*, uint8_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Receive an amount of data in blocking mode.

参数

hspi	A pointer to a SPI_HandleTypeDef structure that contains configuration information for the specified SPI module
pData	A pointer to the address to receive data
Size	The amount of data to be received
Timeout	Wait to receive timeout

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	receive timeout

13.2.2.5 HAL_StatusTypeDef HAL_SPI_Receive_32Bit (SPI_HandleTypeDef * *hspi*, uint32_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Receive an amount of data in blocking mode.

参数

hspi	A pointer to a SPI_HandleTypeDef structure that contains configuration information for the specified SPI module
pData	A pointer to the address to receive data
Size	The amount of data to be received
Timeout	Wait to receive timeout

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	receive timeout

13.2.2.6 HAL_StatusTypeDef HAL_SPI_Receive_IT (SPI_HandleTypeDef * *hspi*, uint8_t * *pData*, uint16_t *Size*, HAL_StatusTypeDef(*)() *recv_callback*)

Receive an amount of data in non-blocking mode with Interrupt.

参数

hspi	pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
pData	pointer to data buffer
Size	amount of data to be sent
recv_callback	receive irq callback

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

13.2.2.7 HAL_StatusTypeDef HAL_SPI_Transmit (SPI_HandleTypeDef * *hspi*, uint8_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Send an amount of data in blocking mode.

参数

hspi	A pointer to a SPI_HandleTypeDef structure that contains configuration information for the specified SPI module
pData	Pointer to the address to send data
Size	The amount of data to be sent
Timeout	Timeout period for waiting for sending to complete

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	transmit timeout

13.2.2.8 HAL_StatusTypeDef HAL_SPI_Transmit_32Bit (SPI_HandleTypeDef * *hspi*, uint32_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Send an amount of data in blocking mode.

参数

hspi	A pointer to a SPI_HandleTypeDef structure that contains configuration information for the specified SPI module
pData	Pointer to the address to send data
Size	The amount of data to be sent
Timeout	Timeout period for waiting for sending to complete

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	transmit timeout

13.2.2.9 HAL_StatusTypeDef HAL_SPI_Transmit_IT (SPI_HandleTypeDef * *hspi*, uint8_t * *pData*, uint16_t *Size*, HAL_StatusTypeDef(*)() *recv_callback*)

Transmit an amount of data in non-blocking mode with Interrupt.

参数

hspi	hspi pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
pData	pointer to data buffer
Size	amount of data to be sent
recv_callback	transmit irq callback

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

13.2.2.10 HAL_StatusTypeDef HAL_SPI_TransmitReceive (SPI_HandleTypeDef * *hspi*, uint8_t * *pTxData*, uint8_t * *pRxData*, uint16_t *Size*, uint32_t *Timeout*)

Transmit and Receive an amount of data in blocking mode.

参数

hspi	
pTxData	
pRxData	
Size	
Timeout	

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	transmit or receive timeout

13.2.2.11 HAL_StatusTypeDef HAL_SPI_TransmitReceive_IT (SPI_HandleTypeDef * *hspi*, uint8_t * *pTxData*, uint8_t * *pRxData*, uint16_t *Size*)

Transmit and Receive an amount of data in non-blocking mode with Interrupt.

参数

hspi	hspi pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
pTxData	pointer to transmission data buffer
pRxData	pointer to reception data buffer
Size	amount of data to be sent and received

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong

HAL_ERROR	something wrong

14 UART

通用异步串口收发器（UART）是使用非常广泛的串行通信接口，支持全双工通信。通用异步串口收发器是把存储器或处理器中并行传输的数据串行的发送到外设的UART接收端，或接收UART外设的串行数据并转换为并行数据提供给处理器。UART支持与外部接口设备的串行通信。

14.1 UART Exported Types

14.1.1 结构体

- struct **UART_InitTypeDef**：UART Init structure definition
- struct **__UART_HandleTypeDef**：UART Handle structure definition

14.1.2 类型定义

- typedef struct __UART_HandleTypeDef UART_HandleTypeDef
  UART Handle structure definition

14.2 UART Exported Functions函数说明

14.2.1 HAL_StatusTypeDef HAL_UART_Init (UART_HandleTypeDef * *huart*)

Initializes the UART with the specified argument in UART_InitTypeDef and creates the associated handle

参数

huart	A pointer to a UART_HandleTypeDef structure that contains configuration information for the specified UART module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

14.2.2 void HAL_UART_IRQHandler (UART_HandleTypeDef * *huart*)

Reference interrupt function

参数

huart	A pointer to a UART_HandleTypeDef structure that contains configuration information for the specified UART module

返回

None

14.2.3 HAL_StatusTypeDef HAL_UART_Receive (UART_HandleTypeDef * *huart*, uint8_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Blocking receives multiple data

参数

huart	A pointer to a UART_HandleTypeDef structure that contains configuration information for the specified UART module
pData	A pointer to the address to receive data
Size	The amount of data to be received
Timeout	Wait to receive timeout

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

14.2.4 HAL_StatusTypeDef HAL_UART_Receive_IT (UART_HandleTypeDef * *huart*, uint8_t * *pData*, uint16_t *Size*, HAL_StatusTypeDef(*)() *recv_callback*)

Receive multiple data non-blocking

参数

huart	A pointer to a UART_HandleTypeDef structure that contains onfiguration information for the specified UART module
pData	A pointer to the address to receive data
Size	The amount of data to be received
recv_callback	Receive the callback function

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

14.2.5 HAL_StatusTypeDef HAL_UART_Transmit (UART_HandleTypeDef * *huart*, uint8_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Blocking transmission

参数

huart	A pointer to a UART_HandleTypeDef structure that contains configuration information for the specified UART module
pData	Pointer to the address to send data
Size	The amount of data to be sent
Timeout	Timeout period for waiting for sending to complete

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

14.2.6 HAL_StatusTypeDef HAL_UART_Transmit_IT (UART_HandleTypeDef * *huart*, uint8_t * *pData*, uint16_t *Size*, HAL_StatusTypeDef(*)() *send_callback*)

Sending multiple data in non-blocking mode

参数

huart	A pointer to a UART_HandleTypeDef structure that contains configuration information for the specified UART module
pData	Pointer to the address to send data
Size	The amount of data to be sent
Timeout	Timeout period for waiting for sending to complete

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

14.2.7 HAL_StatusTypeDef UART_BaudConfig (UART_HandleTypeDef * *huart*)

Configure the UART baud rate

参数

huart	A pointer to a UART_HandleTypeDef structure that contains configuration information for the specified UART module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

14.2.8 HAL_StatusTypeDef UART_IrqConfig (UART_HandleTypeDef * *huart*)

Configure UART interrupt

参数

huart	A pointer to a UART_HandleTypeDef structure that contains configuration information for the specified UART module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

14.2.9 HAL_StatusTypeDef UART_SetConfig (UART_HandleTypeDef * *huart*)

Initializes the UART with the specified argument in UART_InitTypeDef

参数

huart	A pointer to a UART_HandleTypeDef structure that contains configuration information for the specified UART module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

15 UART1

通用异步串口收发器（UART）是使用非常广泛的串行通信接口，支持全双工通信。通用异步串口收发器是把存储器或处理器中并行传输的数据串行的发送到外设的UART接收端，或接收UART外设的串行数据并转换为并行数据提供给处理器。UART支持与外部接口设备的串行通信。

15.1 UART1 Exported Types

15.1.1 结构体

- struct **UART1_InitTypeDef**：UART Init Structure definition
- struct **__UART1_HandleTypeDef**：UART handle Structure definition

15.1.2 类型定义

- typedef struct __UART1_HandleTypeDef UART1_HandleTypeDef
  UART handle Structure definition

15.2 UART1 Exported Functions函数说明

15.2.1 Initialization/de-initialization functions

Initialization and Configuration functions

15.2.1.1 HAL_StatusTypeDef HAL_UART1_Init (UART1_HandleTypeDef * *huart*)

uart1 init

参数

huart	UART handle.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

15.2.1.2 HAL_StatusTypeDef UART1_BaudConfig (UART1_HandleTypeDef * *huart*)

Configure the UART1 baud rate

参数

huart	UART handle.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

15.2.1.3 HAL_StatusTypeDef UART1_IrqConfig (UART1_HandleTypeDef * *huart*)

Configure UART1 interrupt

参数

huart	UART handle.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

15.2.1.4 HAL_StatusTypeDef UART1_SetConfig (UART1_HandleTypeDef * *huart*)

uart1 config

参数

huart	UART handle.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

15.2.2 IO operation functions

UART1 Read, Write, Toggle, Lock and EXTI management functions.

15.2.2.1 HAL_StatusTypeDef HAL_UART1_9Bit_Receive (UART1_HandleTypeDef * *huart*, uint32_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Receive an amount of data in 9bit mode.

参数

huart	UART handle.
pData	Pointer to data buffer .
Size	Amount of data elements to be received.
Timeout	Timeout duration.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	receive timeout

15.2.2.2 HAL_StatusTypeDef HAL_UART1_9Bit_Transmit (UART1_HandleTypeDef * *huart*, uint32_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Send an amount of data in 9bit mode.

参数

huart	UART handle.
pData	Pointer to data buffer .
Size	Amount of data elements to be sent.
Timeout	Timeout duration.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	send timeout

15.2.2.3 void HAL_UART1_IRQHandler (UART1_HandleTypeDef * *huart*)

uatri irq handlers

参数

huart	UART handle.

返回

void

15.2.2.4 HAL_StatusTypeDef HAL_UART1_Receive (UART1_HandleTypeDef * *huart*, uint8_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Receive an amount of data in blocking mode.

参数

huart	UART handle.
pData	Pointer to data buffer.
Size	Amount of data elements to be received.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	receive timeout

15.2.2.5 HAL_StatusTypeDef HAL_UART1_Receive_IT (UART1_HandleTypeDef * *huart*, uint8_t * *pData*, uint16_t *Size*, HAL_StatusTypeDef(*)() *recv_callback*)

Receive an amount of data in interrupt mode.

参数

huart	UART handle.
pData	Pointer to data buffer
Size	Amount of data elements to be received.
recv_callback	receive irq callback

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

15.2.2.6 HAL_StatusTypeDef HAL_UART1_Receive_RTS (UART1_HandleTypeDef * *huart*, uint8_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Receive an amount of data in RTS mode

参数

huart	UART handle.
pData	Pointer to data buffer.
Size	Amount of data elements to be received.
Timeout	Timeout duration.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	receive timeout

15.2.2.7 HAL_StatusTypeDef HAL_UART1_Transmit (UART1_HandleTypeDef * *huart*, uint8_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Send an amount of data in blocking mode.

参数

huart	UART handle.
pData	Pointer to data buffer .
Size	Amount of data elements to be sent.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

15.2.2.8 HAL_StatusTypeDef HAL_UART1_Transmit_CTS (UART1_HandleTypeDef * *huart*, uint8_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Send an amount of data in cts mode.

参数

huart	UART handle.
pData	Pointer to data buffer .
Size	Amount of data elements to be sent.
Timeout	Timeout duration.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	transmit timeout

15.2.2.9 HAL_StatusTypeDef HAL_UART1_Transmit_IT (UART1_HandleTypeDef * *huart*, uint8_t * *pData*, uint16_t *Size*, HAL_StatusTypeDef(*)() *send_callback*)

Send an amount of data in interrupt mode.

参数

huart	UART handle.

pData	Pointer to data buffer.
Size	Amount of data elements to be sent.
send_callback	send irq callback

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

15.2.2.10 HAL_StatusTypeDef HAL_UART1_Transmit_TargetAddr (UART1_HandleTypeDef * *huart*, uint32_t *Addr*)

set transmit target address

参数

huart	UART handle.
Addr	address

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong

16 AES

AES算法是一个分组算法。加密算法与密钥扩展算法都采用非线性迭代结构。解密算法与加密算法的结构相同，只是轮密钥的使用顺序相反，解密轮密钥是加密轮密钥的逆序。

16.1 AES Exported Types

16.1.1 结构体

- struct **AES_InitTypeDef**：AES Init structure definition
- struct **AES_HandleTypeDef**：DIV Handle Structure definition

16.2 AES Exported Functions函数说明

16.2.1 HAL_StatusTypeDef HAL_AES_CRYPT_Start (AES_HandleTypeDef * *haes*, uint8_t *CRYPT_Type*)

AES Start encryption/decryption operations

参数

haes	Pointer to the AES_HandleTypeDef structure that contains configuration information for the specified AES module
CRYPT_Type	Operation type ● AES_CRYPT_DECRYP: Decryption operation ● AES_CRYPT_ENCRYP: Encryption operation

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

在文件 **um32x13x_hal_aes.c** 第 **377** 行定义.

16.2.2 HAL_StatusTypeDef HAL_AES_Init (AES_HandleTypeDef * *haes*)

AES Initial configuration

参数

haes	Pointer to the AES_HandleTypeDef structure that contains configuration information for the specified AES module

返回

None

16.2.3 HAL_StatusTypeDef HAL_AES_IrqConfig (AES_HandleTypeDef * *haes*)

AES Indicates the interrupt configuration

参数

haes	Pointer to the AES_HandleTypeDef structure that contains configuration information for the specified AES module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

16.2.4 void HAL_AES_IRQHandler (AES_HandleTypeDef * *haes*)

Reference interrupt function

参数

haes	Pointer to the AES_HandleTypeDef structure that contains configuration information for the specified AES module

返回

None

16.2.5 HAL_StatusTypeDef HAL_AES_Read_Data (AES_HandleTypeDef * *haes*, uint8_t * *data*)

AES Read ciphertext/plaintext data

参数

haes	Pointer to the AES_HandleTypeDef structure that contains configuration information for the specified AES module
data	Ciphertext/Plaintext data

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

16.2.6 HAL_StatusTypeDef HAL_AES_Write_Data (AES_HandleTypeDef * *haes*, uint8_t * *data*)

AES Write plaintext data

参数

haes	Pointer to the AES_HandleTypeDef structure that contains configuration information for the specified AES module
data	Plaintext data

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

16.2.7 HAL_StatusTypeDef HAL_AES_Write_Ivin (AES_HandleTypeDef * *haes*, uint8_t * *pAesivin*)

AES Write the initial vector data

参数

haes	Pointer to the AES_HandleTypeDef structure that contains configuration information for the specified AES module
pAesivin	Initial vector data

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

16.2.8 HAL_StatusTypeDef HAL_AES_Write_Key (AES_HandleTypeDef * *haes*, uint8_t * *pAeskey*, uint32_t *KeyLength*)

AES Write key

参数

haes	Pointer to the AES_HandleTypeDef structure that contains configuration information for the specified AES module
pAeskey	Key value
KeyLength	Key length ● AES_KEY_LENGTH_128: The AES key length mode is 128 ● AES_KEY_LENGTH_192: The AES key length mode is 192 ● AES_KEY_LENGTH_256: The AES key length mode is 256

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

17 ATIMER

高级定时器ATIMER包含一个16bit自动重载计数器及一个可编程预分频器，可以支持多种应用，包括输入捕捉、输出比较、PWM、带死区插入的互补PWM等。

17.1 ATIMER Exported Typedefs

17.1.1 结构体

- struct **ATIMER_Base_InitTypeDef**：ATIMER base Configuration Structure definition
- struct **ATIMER_OC_InitTypeDef**：ATIMER Output Compare Configuration Structure definition
- struct **ATIMER_IC_InitTypeDef**：ATIMER Input Capture Configuration Structure definition
- struct **ATIMER_EncoderConfigTypeDef**：ATIMER Encoder Configuration Structure definition
- struct **ATIMER_DMAConfigTypeDef**：ATIMER DMA Configuration Structure definition
- struct **ATIMER_MasterConfigTypeDef**：Atimer Master Configuration Structure definition
- struct **ATIMER_SlaveConfigTypeDef**：Atimer Slave Configuration Structure definition
- struct **ATIMER_BreakDeadTimeConfigTypeDef**：Break and Dead Time Configuration Structure definition
- struct **ATIMER_ClockConfigTypeDef**：Clock Configuration Structure definition
- struct **__ATIMER_HandleTypeDef**：Atimer Header Structure definition

17.1.2 类型定义

- typedef struct __ATIMER_HandleTypeDef ATIMER_HandleTypeDef
  Atimer Header Structure definition

17.1.3 枚举

- enum HAL_ATIMER_StateTypeDef { HAL_ATIMER_STATE_RESET = 0x00U, HAL_ATIMER_STATE_READY = 0x01U, HAL_ATIMER_STATE_BUSY = 0x02U, HAL_ATIMER_STATE_TIMEOUT = 0x03U, HAL_ATIMER_STATE_ERROR = 0x04U }
  ATIMER States definition

17.2 ATIMER Exported Functions函数说明

17.2.1 HAL_StatusTypeDef ATIMER_IrqConfig (ATIMER_HandleTypeDef * *hatimer*)

ATIMER Indicates the interrupt configuration

参数

hatimer	Pointer to the GTIMER_HandleTypeDef structure

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

17.2.2 HAL_StatusTypeDef HAL_ATIMER_Base_Init (ATIMER_HandleTypeDef * *hatimer*)

ATIMER Base initialization

参数

hatimer	ATIMER_HandleTypeDef pointer

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

17.2.3 __weak void HAL_ATIMER_Base_MspInit (ATIMER_HandleTypeDef * *hatimer*)

Some preparation for initialization

参数

hatimer	ATIMER_HandleTypeDef pointer

返回

None

注解

This is a weak function that can be redefined by the user

17.2.4 HAL_StatusTypeDef HAL_ATIMER_Base_Start (ATIMER_HandleTypeDef * *hatimer*)

Starts the ATIMER Base generation.

参数

htim	TIM Base handle

返回

HAL status

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

17.2.5 HAL_StatusTypeDef HAL_ATIMER_ConfigBreakDeadTime (ATIMER_HandleTypeDef * *hatimer*, ATIMER_BreakDeadTimeConfigTypeDef * *sConfig*)

Configures the Break feature, dead time, Lock level, OSSI/OSSR State and the AOE(automatic output enable).

参数

hatimer	ATIMER handle
sConfig	pointer to a ATIMER_ConfigBreakDeadConfigTypeDef structure that contains the BDTR Register configuration information for the ATIMER peripheral.

注解

Interrupts can be generated when an active level is detected on the break input, the break 2 input or the system break input. Break interrupt can be enabled by calling the __HAL_ATIMER_ENABLE_IT macro.

返回

HAL status

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

17.2.6 HAL_StatusTypeDef HAL_ATIMER_ConfigClockSource (ATIMER_HandleTypeDef * *hatimer*, ATIMER_ClockConfigTypeDef * *sConfig*)

Configures the clock source to be used

参数

hatimer	ATIMER handle
sConfig	pointer to a ATIMER_ClockConfigTypeDef structure that contains the clock source information for the ATIMER peripheral.

返回

HAL status

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

17.2.7 HAL_StatusTypeDef HAL_ATIMER_ConfigDMA (ATIMER_HandleTypeDef * *hatimer*, ATIMER_DMAConfigTypeDef * *sConfig*)

Configures the DMA Transfrom.

参数

hatimer	ATIMER handle
sConfig	pointer to a ATIMER_DMAConfigTypeDef structure

返回

HAL status

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

17.2.8 HAL_StatusTypeDef HAL_ATIMER_Encoder_ConfigChannel (ATIMER_HandleTypeDef * *hatimer*, ATIMER_EncoderConfigTypeDef * *sConfig*, uint32_t *Channel*)

Configures the ATIMER Encoder Input Capture Channels according to the specified parameters in the ATIMER_EncoderConfigTypeDef.

参数

hatimer	ATIMER handle
sConfig	ATIMER Encoder Input Capture configuration structure
Channel	ATIMER Encoder Channel to configure This parameter can be one of the following values: ● ATIMER_ENCODER_CHANNEL_1: ATIMER Encoder Channel 1 selected ● ATIMER_ENCODER_CHANNEL_2: ATIMER Encoder Channel 2 selected

注解

Encoder Channel 1 equal to the ATIMER Channel 1&2 ; Encoder Channel 2 equal to the ATIMER Channel 3&4 ;

返回

HAL status

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

17.2.9 HAL_StatusTypeDef HAL_ATIMER_IC_ConfigChannel (ATIMER_HandleTypeDef * *hatimer*, ATIMER_IC_InitTypeDef * *sConfig*, uint32_t *Channel*)

Initializes the ATIMER Input Capture Channels according to the specified parameters in the ATIMER_IC_InitTypeDef.

参数

hatimer	ATIMER handle
sConfig	ATIMER Input Capture configuration structure
Channel	ATIMER Channels to configure This parameter can be one of the following values: ● ATIMER_CHANNEL_1: ATIMER Channel 1 selected ● ATIMER_CHANNEL_2: ATIMER Channel 2 selected ● ATIMER_CHANNEL_3: ATIMER Channel 3 selected ● ATIMER_CHANNEL_4: ATIMER Channel 4 selected

返回

HAL status

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

17.2.10 void HAL_ATIMER_IRQHandler (ATIMER_HandleTypeDef * *hatimer*)

Atimer interrupt example handle

参数

hatimer	ATIMER_HandleTypeDef pointer

返回

None

17.2.11 HAL_StatusTypeDef HAL_ATIMER_MasterConfigSynchro (ATIMER_HandleTypeDef * *hatimer*, ATIMER_MasterConfigTypeDef * *sConfig*)

Configures the ATIMER in master mode.

参数

hatimer	ATIMER handle
sConfig	pointer to a ATIMER_MasterConfigTypeDef structure that contains the selected trigger output (TRGO) and the Master/Slave mode.

返回

HAL status

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

17.2.12 HAL_StatusTypeDef HAL_ATIMER_OC_ConfigChannel (ATIMER_HandleTypeDef * *hatimer*, ATIMER_OC_InitTypeDef * *sConfig*, uint32_t *Channel*)

Initializes the ATIMER Output Compare Channels according to the specified parameters in the ATIMER_OC_InitTypeDef.

参数

atimer	ATIMER Output Compare handle
sConfig	ATIMER Output Compare configuration structure
Channel	ATIMER Channels to configure This parameter can be one of the following values: ● ATIMER_CHANNEL_1: ATIMER Channel 1 selected ● ATIMER_CHANNEL_2: ATIMER Channel 2 selected ● ATIMER_CHANNEL_3: ATIMER Channel 3 selected ● ATIMER_CHANNEL_4: ATIMER Channel 4 selected

返回

HAL status

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

17.2.13 HAL_StatusTypeDef HAL_ATIMER_SlaveConfigSynchro (ATIMER_HandleTypeDef * *hatimer*, ATIMER_SlaveConfigTypeDef * *sConfig*)

Configures the ATIMER in Slave mode

参数

hatimer	ATIMER handle
sConfig	pointer to a ATIMER_SlaveConfigTypeDef structure that contains the selected trigger (internal trigger input, filtered timer input or external trigger input) and the Slave mode

返回

HAL status

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

18 CAN

CAN（Controller Area Network） 控制器可以用于汽车电子和工业控制领域，支持 CAN2.0A/B 协议。

18.1 CAN Exported Types

18.1.1 结构体

- struct **CAN_InitTypeDef**：CAN init structure definition
- struct **CAN_FilterTypeDef**：CAN filter configuration structure definition
- struct **CAN_TxHeaderTypeDef**：CAN Tx message header structure definition
- struct **CAN_RxHeaderTypeDef**：CAN Rx message header structure definition
- struct **__CAN_HandleTypeDef**：CAN handle Structure definition

18.1.2 类型定义

- typedef struct __CAN_HandleTypeDef CAN_HandleTypeDef
  CAN handle Structure definition

18.1.3 枚举

- enum HAL_CAN_StateTypeDef { HAL_CAN_STATE_RESET = 0x00U, HAL_CAN_STATE_READY = 0x01U, HAL_CAN_STATE_LISTENING = 0x02U, HAL_CAN_STATE_SLEEP_PENDING = 0x03U, HAL_CAN_STATE_SLEEP_ACTIVE = 0x04U, HAL_CAN_STATE_ERROR = 0x05U }
  HAL State structures definition
- enum HAL_CAN_FilterModeTypeDef { HAL_CAN_FilterMode_MASK = 0x00U, HAL_CAN_FilterMode_SINGLE = 0x01U, HAL_CAN_FilterMode_DOUBLE = 0x02U }

18.2 Can Exported Functions函数说明

18.2.1 Initialization/de-initialization functions

Initialization and Configuration functions

18.2.1.1 HAL_StatusTypeDef HAL_CAN_Init (CAN_HandleTypeDef * *hcan*)

Initialize configuration of Can

参数

hcan	Pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

18.2.2 IO operation functions

CAN Send, receivede and EXTI management functions.

18.2.2.1 HAL_StatusTypeDef HAL_CAN_AddTxMessage (CAN_HandleTypeDef * *hcan*, CAN_TxHeaderTypeDef * *pHeader*, uint8_t *pData*[])

loading the tx message into the can txfifo.

Add a message to the first free Tx mailbox and activate the corresponding transmission request.

loading the tx message into the can txfifo and Write the CAN frame to be sent through the CAN network.

参数

hcan	pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.
pHeader	pointer to a CAN_TxHeaderTypeDef structure.
pData	array containing the payload of the Tx frame.
Timeout	Timeout duration.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	something wrong

18.2.2.2 HAL_StatusTypeDef HAL_CAN_ConfigFilter (CAN_HandleTypeDef * *hcan*, CAN_FilterTypeDef * *sFilterConfig*)

Configures the CAN reception filter according to the specified parameters in the CAN_FilterInitStruct.

参数

hcan	pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.
sFilterConfig	pointer to a CAN_FilterTypeDef structure that contains the filter configuration information.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

18.2.2.3 HAL_StatusTypeDef HAL_CAN_GetRxMessage (CAN_HandleTypeDef * *hcan*, CAN_RxHeaderTypeDef * *pHeader*, uint8_t * *pData*)

Get an CAN frame from the Rx FIFO zone into the message RAM.

参数

hcan	pointer to an CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.
pHeader	pointer to a CAN_RxHeaderTypeDef structure where the header of the Rx frame will be stored.
pData	array where the payload of the Rx frame will be stored.
Timeout	Timeout duration.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	something wrong

18.2.2.4 void HAL_CAN_IRQHandler (CAN_HandleTypeDef * *hcan*)

This function handles CAN event interrupt request.

参数

hcan	Pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

返回值

None	

18.2.2.5 HAL_StatusTypeDef HAL_CAN_Receive (CAN_HandleTypeDef * *hcan*, CAN_RxHeaderTypeDef * *pHeader*, uint32_t *Timeout*)

Get an CAN frame from the Rx FIFO zone into the message RAM.

参数

hcan	pointer to an CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.
pHeader	pointer to a CAN_RxHeaderTypeDef structure where the header of the Rx frame will be stored.
pData	array where the payload of the Rx frame will be stored.
Timeout	Timeout duration.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	something wrong

18.2.2.6 HAL_StatusTypeDef HAL_CAN_Receive_IT (CAN_HandleTypeDef * *hcan*, CAN_RxHeaderTypeDef * *pHeader*, HAL_StatusTypeDef(*)() *recv_callback*)

Get an CAN frame from the Rx FIFO zone into the message RAM by Interrupt . parameters in the CAN_FilterInitStruct.

参数

hcan	pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.
sFilterConfig	pointer to a CAN_FilterTypeDef structure that contains the filter configuration information.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

18.2.2.7 HAL_StatusTypeDef HAL_CAN_Transmit (CAN_HandleTypeDef * *hcan*, CAN_TxHeaderTypeDef * *pHeader*, uint32_t *Timeout*)

Write the CAN frame to be sent through the CAN network.

参数

hcan	pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.
pHeader	pointer to a CAN_TxHeaderTypeDef structure.
pData	array containing the payload of the Tx frame.

Timeout	Timeout duration.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong
HAL_TIMEOUT	something wrong

19 CRC

19.1 CRC Exported Types

19.1.1 结构体

- struct **CRC_InitTypeDef**：CRC Init structure definition
- struct **CRC_HandleTypeDef**：CRC Handle Structure definition

19.2 CRC Exported Functions函数说明

19.2.1 uint32_t HAL_CRC_Calculate (CRC_HandleTypeDef * *hcrc*, uint8_t *pBuffer*[], uint16_t *BufferLength*)

Calculate the CRC value obtained in pBuffer

参数

hcrc	CRC handle
pBuffer	Pointer to the input data buffer.
BufferLength	Input data buffer length

返回

temp

返回值

CRC	value

19.2.2 HAL_StatusTypeDef HAL_CRC_Init (CRC_HandleTypeDef * *hcrc*)

Initializes the CRC according to the argument in CRC_InitTypeDef

参数

hcrc	CRC handle

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

20 DIV

DIV的实现目标是能够支持不超过32bit的除法。除数不大于32bit，被除数可以为任意bit。主要应用于除数小于32bit的应用。

20.1 DIV Exported Types

20.1.1 结构体

- struct **DIV_InitTypeDef**：DIV Init structure definition
- struct **DIV_HandleTypeDef**：DIV Handle Structure definition

20.2 DIV Exported Functions函数说明

20.2.1 HAL_StatusTypeDef DIV_IrqConfig (DIV_HandleTypeDef * *hdiv*)

DIV Indicates the interrupt configuration

参数

hdiv	Pointer to the DIV_HandleTypeDef structure that contains configuration information for the specified DIV module

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

20.2.2 HAL_StatusTypeDef HAL_DIV_Calculate (DIV_HandleTypeDef * *hdiv*, uint32_t *divisor*, uint32_t *dividend*, uint32_t * *quotients*, uint32_t * *remainder*)

DIV calculates the quotient and remainder

参数

**hdiv*	Pointer to the DIV_HandleTypeDef structure that contains configuration information for the specified DIV module
divisor	The entered division value

dividend	The entered dividend value
quotients	The calculated quotient
remainder	The calculated remainder

注解

This function cannot be used when the compute completion interrupt is enabled

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

20.2.3 HAL_StatusTypeDef HAL_DIV_Init (DIV_HandleTypeDef * *hdiv*)

Initializes the DIV according to the argument in DIV_HandleTypeDef

参数

hdiv	div handle

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

20.2.4 void HAL_DIV_IRQHandler (DIV_HandleTypeDef * *hdiv*)

Reference interrupt function

参数

hdiv	Pointer to the DIV_HandleTypeDef structure that contains configuration information for the specified DIV module

返回

None

21 EEPROM

21.1 EEPROM_Registration_Init函数说明

21.1.1 EEPROM_Status_t EE_FlashErasePage (FLASH_Addr_t *PageBase*, uint8_t *PageCount*)

EEPROM Page erase

参数

PageBase	page base addresss
PageCount	page count number

返回

EEPROM_Status_t

返回值

EEPROM_SUCCESS	Everything is OK
EEPROM_FLASH_FAULT	Flash Hardware Fault, Erase Error, Program Error
EEPROM_UNFORMATED	All Pages are ERASED, Eeprom need be formatted with init data
EEPROM_PAGE_FULL	Can not write Current Page because it is full

21.1.2 EEPROM_Status_t EE_FlashInit (FLASH_Addr_t *PageBase*, uint8_t *PageCount*, uint8_t *wait_time*)

Configuration initialization of eeprom

参数

PageBase	page base addresss
PageCount	page count number
wait_time	Timeout times

返回

EEPROM_Status_t

返回值

EEPROM_SUCCESS	Everything is OK
EEPROM_FLASH_FAULT	Flash Hardware Fault, Erase Error, Program Error
EEPROM_UNFORMATED	All Pages are ERASED, Eeprom need be formatted with init data

EEPROM_PAGE_FULL	Can not write Current Page because it is full

21.1.3 EEPROM_Status_t EE_FlashProgram (FLASH_Addr_t *Addr*, uint32_t * *Val*)

EEPROM program

参数

PageBase	page base addresss
PageCount	page count number

返回

EEPROM_Status_t

返回值

EEPROM_SUCCESS	Everything is OK
EEPROM_FLASH_FAULT	Flash Hardware Fault, Erase Error, Program Error
EEPROM_UNFORMATED	All Pages are ERASED, Eeprom need be formatted with init data
EEPROM_PAGE_FULL	Can not write Current Page because it is full

21.1.4 EEPROM_Status_t EE_FlashRead (FLASH_Addr_t *Addr*, uint32_t * *Val*)

EEPROM read data

参数

PageBase	page base addresss
PageCount	page count number

返回

EEPROM_Status_t

返回值

EEPROM_SUCCESS	Everything is OK
EEPROM_FLASH_FAULT	Flash Hardware Fault, Erase Error, Program Error
EEPROM_UNFORMATED	All Pages are ERASED, Eeprom need be formatted with init data
EEPROM_PAGE_FULL	Can not write Current Page because it is full

22 EFC

22.1 EFC Exported Types

22.1.1 结构体

- union **FLASH_ValuePacket**：Data union, which can be used as one byte (4Bytes), two half-bytes (2Bytes) and four words (1Bytes)
- struct **FLASH_ModifyDataTypeDef**：*FLASH Data struct*
- struct **FLASH_PageModifyTypeDef**：*FLASH page struct*

22.2 EFC Exported Functions函数说明

22.2.1 Initialization and de-initialization functions

Initialization and Configuration functions

22.2.1.1 __INLINE HAL_StatusTypeDef __HAL_FLASH_WAIT_STATUS (uint32_t *maxtime*)

Wait for the flash operation to finish

参数

maxtime	Timeout times

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

22.2.1.2 uint16_t HAL_FLASH_Check_CRC_SN (void)

Check CRC SN

注解

read crc for unique SN（16bytes）

参数

none	

返回

0 CRC OK 0xFFFF CRC not write other CRC fail

22.2.1.3 HAL_StatusTypeDef HAL_FLASH_Chip_Erase ()

Erase one chip of flash

参数

None	

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

22.2.1.4 uint16_t HAL_FLASH_CRC (uint32_t *addr*, uint32_t *len*, uint16_t *crc_init*)

FLASH CRC

参数

uint32_t	addr start address
uint32_t	len data length
uint16_t	crc_init initial value for CRC16-CCITT

返回

CRC value

22.2.1.5 __WEAK void HAL_FLASH_FLAG_BOOT_ErrorCallback (void)

Flash flag boot error callbacks.

参数

None	

返回值

None	

22.2.1.6 __WEAK void HAL_FLASH_FLAG_EEPROM_ErrorCallback (void)

Flash flag eeprom error callbacks.

参数

None	

返回值

None	

22.2.1.7 __WEAK void HAL_FLASH_FLAG_ERASE_WRITE_ErrorCallback (void)

Flash flag erase and write error callbacks.

参数

None	

返回值

None	

22.2.1.8 __WEAK void HAL_FLASH_FLAG_NVR1_ErrorCallback (void)

Flash flag NVR1 error callbacks.

参数

None	

返回值

None	

22.2.1.9 __WEAK void HAL_FLASH_FLAG_NVR2_ErrorCallback (void)

Flash flag NVR2 error callbacks.

参数

None	

返回值

None	

22.2.1.10 __WEAK void HAL_FLASH_FLAG_PROGRAM_ErrorCallback (void)

Flash flag program error callbacks.

参数

None	

返回值

None	

22.2.1.11 void HAL_FLASH_Init (uint8_t *wait_time*)

Flash initialization

参数

maxtime	Timeout times

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong

HAL_ERROR	something wrong

22.2.1.12 void HAL_FLASH_IRQHandler (void)

Flash interrupt service function

参数

None	

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

22.2.1.13 HAL_StatusTypeDef HAL_FLASH_Page_Erase (uint32_t *page_addr*)

Erase one page of flash

参数

page_addr	The first address of the page to be erased

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

22.2.1.14 HAL_StatusTypeDef HAL_FLASH_Page_ReWrite (FLASH_PageModifyTypeDef * *data*)

ReWrite one page of flash

参数

data	A pointer to a FLASH_PageModifyTypeDef structure that contains configuration information for the specified FLASH_Page

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

22.2.1.15 HAL_StatusTypeDef HAL_FLASH_Program (uint32_t *TypeProgram*, uint32_t *Address*, uint32_t *Data*)

Program halfword, word or byte at a specified address

参数

TypeProgram	Indicate the way to program at a specified address. This parameter can be a value of ref FLASH_Type_Program
Address	Specifie the address to be programmed.
Data	Specifie the data to be programmed

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

22.2.1.16 HAL_StatusTypeDef HAL_FLASH_Program_Byte (uint32_t *addr*, uint8_t *value*)

Write a Byte(8-bit) data at the specified address of flash

参数

addr	Address to write data to flash
value	value of write to flash

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

22.2.1.17 HAL_StatusTypeDef HAL_FLASH_Program_HalfWord (uint32_t *addr*, uint16_t *value*)

Write a 2Bytes(16-bit) data at the specified address of flash

参数

addr	Address to write data to flash
value	value of write to flash

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

22.2.1.18 HAL_StatusTypeDef HAL_FLASH_Program_Word (uint32_t *addr*, uint32_t *value*)

Write a 4Bytes(32-bit) data at the specified address of flash

参数

addr	Address to write data to flash

value	value of write to flash

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

22.2.1.19 uint16_t HAL_FLASH_Read_Sequence (uint8_t * *buff*)

FLASH Read Sequence

注解

read unique SN（16 bytes）

参数

**buff*	SN buff pointer

返回

SN CRC result

22.2.1.20 uint16_t HAL_FLASH_Read_UID (uint8_t * *buff*)

Read UID

注解

read unique UID（8 bytes）

参数

**buff*	SN buff pointer

返回

UID CRC result

23 GPIO

GPIO包含通用数据输入输出接口，这些管脚可以与其他功能管脚共享，这取决于芯片的配置。通过这些数据接口，可以配置任意数目的管脚作为中断信号。该芯片有4组GPIO，分别是GPIOA、GPIOB、GPIOC、GPIOD分别简称为PA、PB、PC、PD。GPIO的相关寄存器的功能需要设置对应的比特位，例如设置PA1方向为输出，GPIO_DIR的bit[1]控制位需要设置为1，其他位的设置遵循此原则，也即是PAx对应寄存器GPIO_DIR的bit[x]控制位。

23.1 GPIO Exported Types

23.1.1 结构体

- struct **GPIO_InitTypeDef**：GPIO Init structure definition

23.1.2 枚举

- enum GPIO_PinState { GPIO_PIN_RESET = 0U, GPIO_PIN_SET }
  GPIO Bit SET and Bit RESET enumeration

23.2 GPIO Exported Functions函数说明

23.2.1 void GPIO_Interrupt_Switch (GPIO_TypeDef * *GPIOx*, uint8_t *GPIOx_ITEN*)

GPIO interrupt switch

参数

GPIOx	Where x can be (A/B/C/D depending on device used) to select the GPIO peripheral
GPIOx_ITEN	GPIO interrupt switch selection

返回

None

23.2.2 HAL_StatusTypeDef HAL_GPIO_Init (GPIO_TypeDef * *GPIOx*, GPIO_InitTypeDef * *GPIO_CONFIG*)

Initialize the GPIO according to the parameters specified in HAL_GPIO_Init

参数

GPIOx	Where x can be (A/B/C/D depending on device used) to select the GPIO peripheral
GPIO_CONFIG	Pointer to a GPIO_InitTypeDef structure containing the specified GPIO configuration information

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

23.2.3 IO operation functions

GPIO Read and Write

23.2.3.1 uint8_t HAL_GPIO_GetIoMisStatus (GPIO_TypeDef * *GPIOx*)

Obtain the interrupted status after masking

参数

GPIOx	Where x can be (A/B/C/D depending on device used) to select the GPIO peripheral

返回

Masked interrupt status register value

23.2.3.2 uint8_t HAL_GPIO_GetIoRisStatus (GPIO_TypeDef * *GPIOx*)

Gets the original interrupt status

参数

GPIOx	Where x can be (A/B/C/D depending on device used) to select the GPIO peripheral

返回

Original interrupt status register value

23.2.3.3 GPIO_PinState HAL_GPIO_ReadPin (GPIO_TypeDef * *GPIOx*, uint16_t *GPIO_Pin*)

Reads the specified input port

参数

GPIOx	where x can be (A/B/C/D depending on device used) to select the GPIO peripheral
GPIO_Pin	Specifies the port bit to read. This parameter can be GPIO_PIN_x, where x can be (0.. 7)

返回

Enter the port pin value

返回值

1	GPIO is high level
0	GPIO is low level

23.2.3.4 void HAL_GPIO_WritePin (GPIO_TypeDef * *GPIOx*, uint16_t *GPIO_Pin*, GPIO_PinState *PinState*)

Sets or clears the selected data port bit

参数

GPIOx	Where x can be (A/B/C/D depending on device used) to select the GPIO peripheral
GPIO_Pin	Specifies the port bit to write to. This argument can be GPIO_PIN_x, where x can be (0.. 7)
PinState	Specifies the value to write to the selected location. This parameter can be one of the GPIO_PinState enumeration values: ● GPIO_PIN_RESET: Clear port pins ● GPIO_PIN_SET: Set port pins

返回

None

23.2.3.5 void HAL_GPIOA_IRQHandler (GPIO_InitTypeDef * *GPIO*)

GPIOA reference interrupt function

参数

GPIO	A pointer to a GPIO_InitTypeDef structure that contains configuration information for the specified GPIO module

返回

None

23.2.3.6 void HAL_GPIOB_IRQHandler (GPIO_InitTypeDef * *GPIO*)

GPIOB reference interrupt function

参数

GPIO	A pointer to a GPIO_InitTypeDef structure that contains configuration information for the specified GPIO module

返回

None

23.2.3.7 void HAL_GPIOC_IRQHandler (GPIO_InitTypeDef * *GPIO*)

GPIOC reference interrupt function

参数

GPIO	A pointer to a GPIO_InitTypeDef structure that contains configuration information for the specified GPIO module

返回

None

23.2.3.8 void HAL_GPIOD_IRQHandler (GPIO_InitTypeDef * *GPIO*)

GPIOD reference interrupt function

参数

GPIO	A pointer to a GPIO_InitTypeDef structure that contains configuration information for the specified GPIO module

返回

None

24 I2C

I2C总线接口连接微控制器和串行I2C总线。I2C模块接收和发送数据，并将数据从串行转换成并行，或并行转换成串行。I2C模块通过数据引脚SDA和时钟引脚SCL连接到I2C总线，控制所有I2C总线规定的时序。本模块支持主模式和从模式。

24.1 I2C Exported Types

24.1.1 结构体

- struct I2C_InitTypeDef

24.1.2 枚举

- enum HAL_I2C_ModeTypeDef { HAL_I2C_MODE_NONE = 0x00U, HAL_I2C_MODE_MASTER = 0x10U, HAL_I2C_MODE_SLAVE = 0x20U }
- enum I2C_SLAVE_ADDR { I2C_SLAVE_ADDR0 = 0x01U, I2C_SLAVE_ADDR1 = 0x02U, I2C_SLAVE_ADDR2 = 0x04U, I2C_SLAVE_ADDR3 = 0x08U }

24.2 I2C Exported Functions函数说明

24.2.1 Initialization and de-initialization functions

Initialization and Configuration functions

24.2.1.1 HAL_StatusTypeDef HAL_I2C_Init (I2C_HandleTypeDef * *hi2c*)

Configure the I2C Initialize

参数

hi2c	Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

24.2.1.2 HAL_StatusTypeDef HAL_I2C_Master_SetConfig (I2C_HandleTypeDef * *hi2c*)

Configure the I2C Master Setconfig

参数

hi2c	Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

24.2.1.3 HAL_StatusTypeDef HAL_I2C_Slave_SetConfig (I2C_HandleTypeDef * *hi2c*)

Configure the I2C Slave Setconfig

参数

hi2c	Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

24.2.2 operation functions

Initialization and Configuration functions

24.2.2.1 HAL_StatusTypeDef HAL_I2C_Master_Receive (I2C_HandleTypeDef * *hi2c*, uint16_t *DevAddress*, uint8_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Receive in slave mode an amount of data in blocking mode

参数

hi2c	Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
pData	Pointer to data buffer
Size	Amount of data to be sent
Timeout	Timeout duration

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

24.2.2.2 HAL_StatusTypeDef HAL_I2C_Master_Receive_IT (I2C_HandleTypeDef * *hi2c*, uint16_t *DevAddress*, uint8_t * *pData*, uint16_t *Size*, HAL_StatusTypeDef(*)() *recv_callback*)

Receive in master mode an amount of data in non-blocking mode with Interrupt

参数

hi2c	Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
DevAddress	Target device address: The device 7 bits address value in datasheet must be shifted to the left before calling the interface
pData	Pointer to data buffer
Size	Amount of data to be sent

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

24.2.2.3 HAL_StatusTypeDef HAL_I2C_Master_Transmit (I2C_HandleTypeDef * *hi2c*, uint16_t *DevAddress*, uint8_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Transmits in master mode an amount of data in blocking mode.

参数

hi2c	Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
DevAddress	Target device address: The device 7 bits address value in datasheet must be shifted to the left before calling the interface
pData	Pointer to data buffer
Size	Amount of data to be sent
Timeout	Timeout duration

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

24.2.2.4 HAL_StatusTypeDef HAL_I2C_Slave_Receive (I2C_HandleTypeDef * *hi2c*, uint8_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Receive in slave mode an amount of data in blocking mode

参数

hi2c	Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
pData	Pointer to data buffer
Size	Amount of data to be sent
Timeout	Timeout duration

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

24.2.2.5 HAL_StatusTypeDef HAL_I2C_Slave_Receive_IT (I2C_HandleTypeDef * *hi2c*, uint8_t * *pData*, uint16_t *Size*, HAL_StatusTypeDef(*)() *recv_callback*)

Receive in slave mode an amount of data in non-blocking mode with Interrupt

参数

hi2c	Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
pData	Pointer to data buffer
Size	Amount of data to be sent

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

24.2.2.6 HAL_StatusTypeDef HAL_I2C_Slave_Transmit (I2C_HandleTypeDef * *hi2c*, uint8_t * *pData*, uint16_t *Size*, uint32_t *Timeout*)

Transmits in slave mode an amount of data in blocking mode.

参数

hi2c	Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
pData	Pointer to data buffer
Size	Amount of data to be sent
Timeout	Timeout duration

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

25 PWR

25.1 PWR Exported Functions函数说明

25.1.1 void HAL_PWR_EnterDEEPSLEEPMode (void)

The system enters the low-power deepsleep mode

参数

SLEEPEntry	Specifies if SLEEP mode in entered with WFI or WFE instruction. This parameter can be one of the following values: ● PWR_SLEEPENTRY_WFI: enter SLEEP mode with WFI instruction ● PWR_SLEEPENTRY_WFE: enter SLEEP mode with WFE instruction

返回

None

25.1.2 void HAL_PWR_EnterLprun (void)

The system enters the lprun mode

参数

None	

返回

None

25.1.3 void HAL_PWR_EnterSLEEPMode (uint8_t *SLEEPEntry*)

The system enters the low-power sleep mode

参数

SLEEPEntry	Specifies if SLEEP mode in entered with WFI or WFE instruction. This parameter can be one of the following values: ● PWR_SLEEPENTRY_WFI: enter SLEEP mode with WFI instruction ● PWR_SLEEPENTRY_WFE: enter SLEEP mode with WFE instruction

返回

None

25.1.4 void HAL_PWR_EnterSTOPMode (uint8_t *STOPEntry*)

Enters Stop mode.

参数

STOPEntry	Specifies if Stop mode in entered with WFI or WFE instruction. This parameter can be one of the following values: ● PMU_STOPENTRY_WFI: Enter Stop mode with WFI instruction ● PMU_STOPENTRY_WFE: Enter Stop mode with WFE instruction

返回值

None	

26 RCC

26.1 RCC Exported Types

26.1.1 结构体

- struct **RCC_OscInitTypeDef：**RCC Internal/External Oscillator(RCH\RCL) configuration structure definition
- struct **RCC_ClkInitTypeDef：**RCC System, AHB and APB busses clock configuration structure definition
- struct **RCC_32KInitTypeDef：**The low-frequency 32K clock source is an independent configuration structure. It applies to the scenario where the low-frequency 32K clock needs to be configured as an internal clock or an external crystal oscillator
- struct **RCC_Hf_InitTypeDef：**The high frequency clock source is an independent configuration structure. It applies to the scenario where the high speed clock needs to be configured as an internal clock or an external crystal oscillator

26.2 RCC Exported Functions函数说明

26.2.1 HAL_StatusTypeDef HAL_PC2_OutClk_Close (uint8_t *last_cfgval*)

The PC2 clock frequency is disabled

参数

last_cfgval	Pin multiplexing information of PC2 needs to be configured

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

26.2.2 uint8_t HAL_PC2_OutClk_Open (uint32_t *clock_tpye*)

PC2 Outputs the clock frequency

参数

clock_tpye	Type of clock source to be exported (macro)

返回

PC2 Indicates the original pin configuration

26.2.3 HAL_StatusTypeDef HAL_RCC_Clock_Disable (uint32_t *clk_name*)

The system RCC clock is disabled

参数

clk_name	Turn off the clock source (external high speed, low speed crystal oscillator, internal high speed RCH, internal low speed RCL)

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

26.2.4 HAL_StatusTypeDef HAL_RCC_Clock_Enable (uint32_t *clk_name*)

The system RCC clock was enabled

参数

clk_name	Enable the clock source (external high speed, low speed crystal oscillator, internal high speed RCH, internal low speed RCL).

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

26.2.5 HAL_StatusTypeDef HAL_RCC_ClockConfig (RCC_ClkInitTypeDef * *RCC_ClkInitStruct*)

Configure the system RCC clock

参数

RCC_ClkInitTypeDef	A pointer to the RCC_ClkInitTypeDef structure that contains configuration information for the specified system clock

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

26.2.6 uint32_t HAL_RCC_GetHCLKFreq (void)

Get AHB clock frequency (HCLK)

参数

None	

注解

If the system clock is the external high speed clock XTH or external low speed clock XTL, you need to configure the clock frequency for the XTH_VALUE_Hz and XTL_VALUE_Hz macros defined in um32x13x_hal_conf.h

返回

AHB Clock frequency (HCLK)

26.2.7 uint32_t HAL_RCC_GetPCLKFreq (void)

Get the APB Clock Frequency (PCLK)

参数

None	

注解

If the system clock is the external high speed clock XTH or external low speed clock XTL, you need to configure the clock frequency for the XTH_VALUE_Hz and XTL_VALUE_Hz macros defined in um32x13x_hal_conf.h

返回

APB clock frequency (PCLK)

26.2.8 uint32_t HAL_RCC_GetRCHFreq (void)

Get the RCH Clock Frequency (RCH)

参数

None	

返回

RCH clock frequency

26.2.9 uint32_t HAL_RCC_GetRCLFreq (void)

Get the RCL Clock Frequency (RCL)

参数

None	

返回

RCL clock frequency

26.2.10 uint32_t HAL_RCC_GetSysClockFreq (void)

Get the system clock frequency value (sysclock)

参数

None	

注解

If the system clock is the external high speed clock XTH or external low speed clock XTL, you need to configure the clock frequency for the XTH_VALUE_Hz and XTL_VALUE_Hz macros defined in um32x13x_hal_conf.h

返回

System clock frequency

26.2.11 HAL_StatusTypeDef HAL_RCC_OscConfig (RCC_OscInitTypeDef * *RCC_OscInitStruct*)

Configure the system RCC clock source

参数

RCC_OscInitStruct	A pointer to the RCC_OscInitTypeDef structure that contains configuration information for the specified system clock source

返回

HAL_StatusTypeDef

返回值

HAL_OK	nothing wrong
HAL_ERROR	something wrong

26.2.12 void Wait_Clock_Stabilize (uint32_t *SCU_OSC_x_STABLE*)

Wait for the input clock to stabilize

参数

SCU_OSC_x_STABLE	Wait for the clock stable type

返回

None

27 SYSTICK

27.1 SYSTICK Exported Functions函数说明

27.1.1 void HAL_ClearSysTickCount (void)

Clear the systick meter value

参数

None	

返回

None

27.1.2 uint32_t HAL_GetSysTickCount (void)

Gets the systick meter value

参数

None	

返回

systick counts the value

27.1.3 uint32_t HAL_GetSysTickLoad (void)

Gets the systick overload value

参数

None	

返回

systick overload value

27.1.4 HAL_StatusTypeDef HAL_InitTick (uint32_t *INT_EN*)

Initialize systick

参数

None	

返回

HAL status information

27.1.5 void HAL_ResumeTick (void)

Enable SysTick Interrupt

参数

None	

返回

None

27.1.6 void HAL_SetSysTickLoad (uint32_t *load*)

Set the systick overload value

参数

load	Overload value to set

返回

None

27.1.7 void HAL_SuspendTick (void)

Disable SysTick Interrupt

参数

None	

返回

None

27.1.8 uint32_t HAL_SYSTICK_Config (uint32_t *Load*)

Clear the systick meter value

参数

Load	Overload value to be set

返回

Initialization information

返回值

0	Function succeeded.
1	Function failed.

27.1.9 uint32_t HAL_SysTickGetFlag (void)

Gets the systick flag bit

参数

None	

返回

The value of the systick flag bit

27.1.10 HAL_StatusTypeDef SYSTICK_IrqConfig (uint32_t *INT_EN*)

systick interrupt configuration

参数

None	

返回

None

28 WDT

看门狗定时器在到达超时的值的时候可以产生不可屏蔽中断或者是复位。当系统由于软件错误或是由于外部设备故障而无法按照预期的方式响应的时候，使用看门狗定时器可以重新获得控制权。

28.1 WDT Exported Typedefs

28.1.1 结构体

- struct **WDT_InitTypeDef**：WDT Init structure definition
- struct **__WDT_HandleTypeDef**：WDT handle Structure definition

28.1.2 类型定义

- typedef struct **__WDT_HandleTypeDef WDT_HandleTypeDef**
  WDT handle Structure definition

28.2 WDT Exported Functions函数说明

28.2.1 HAL_StatusTypeDef HAL_WDT_Init (WDT_HandleTypeDef * *hwdt*)

Used to initialize WDT IP

参数

hwdt	WDT_HandleTypeDef pointer

注解

Time = ((Prescaler + 1)*Reload)/32k (s)

返回

HAL status

28.2.2 void HAL_WDT_IRQHandler (WDT_HandleTypeDef * *hwdt*)

Reference WDT interrupt function

参数

hwdt	WDT_HandleTypeDef pointer

返回

None

28.2.3 __INLINE void HAL_WDT_LOCK (WDT_HandleTypeDef * *hwdt*)

Lock the WDT register

参数

hwdt	WDT_HandleTypeDef pointer

返回

None

28.2.4 __weak void HAL_WDT_MspInit (WDT_HandleTypeDef * *hwdt*)

User initialization WDT

参数

hwdt	WDT_HandleTypeDef pointer

注解

If user need to operate during WDT initialization, user can reconstruct this function

返回

None

28.2.5 HAL_StatusTypeDef HAL_WDT_Refresh (WDT_HandleTypeDef * *hwdt*)

This function is used to feed dogs

参数

hwdt	WDT_HandleTypeDef pointer

返回

HAL status

28.2.6 __INLINE void HAL_WDT_UNLOCK (WDT_HandleTypeDef * *hwdt*)

Unlock the WDT register

参数

hwdt	WDT_HandleTypeDef pointer

返回

None

29 WWDT

窗口看门狗是一个与CPU同步运行的看门狗，目的是实时监控CPU运行状态，在CPU运行异常的情况下复位CPU，避免不可预计的后果。

29.1 Wwdt Exported Typedefs

29.1.1 结构体

- struct **WWDT_InitTypeDef**：WWDT Init structure definition
- struct **__WWDT_HandleTypeDef**：WWDT handle Structure definition

29.1.2 类型定义

- typedef struct **__WWDT_HandleTypeDef WWDT_HandleTypeDef**
  WWDT handle Structure definition

29.2 WWDT Exported Functions函数说明

29.2.1 HAL_StatusTypeDef HAL_WWDT_Init (WWDT_HandleTypeDef * *hwwdt*)

Used to initialize WWDT

参数

hwwdt	WWDT_HandleTypeDef pointer

注解

Time = (Prescale *4096) / SystemClock (S)

返回

HAL status

29.2.2 void HAL_WWDT_IRQHandler (WWDT_HandleTypeDef * *hwwdt*)

Default WWDT interrupt function

参数

hwwdt	WWDT_HandleTypeDef pointer

返回

None

29.2.3 __WEAK void HAL_WWDT_MspInit (WWDT_HandleTypeDef * *hwwdt*)

User initialization WWDT

参数

hwwdt	WWDT_HandleTypeDef pointer

注解

If user need to operate during WWDT initialization, he can reconstruct this function to avoid calling HAL_WWDT_Init function repeatedly

返回

None

29.2.4 __INLINE HAL_StatusTypeDef HAL_WWDT_Refresh (WWDT_HandleTypeDef * *hwwdt*)

WWDT feed dog

参数

hwwdt	WWDT_HandleTypeDef pointer

返回

HAL status

29.2.5 __INLINE void HAL_WWDT_Start (WWDT_HandleTypeDef * *hwwdt*)

WWDT run

参数

hwwdt	WWDT_HandleTypeDef pointer

返回

None

30 版本维护

日期	版本	描述
2024/03/04	V1.0	初始版本