## Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Author** | **Changes** |
| 2022/2/18 | 1.0 | Pantianwen | Initial version. |

IP Description:

## Feature set: （功能列表）：

1. GPIO输入

2 GPIO输出

3 外部中断

4 防抖

5 功能复用.

## Feature comparison with STM32: （跟STM32的功能比较）

**Chip-Platform:**

|  |  |  |
| --- | --- | --- |
| 符号描述：   1. ✅支持 2. ❌不支持 3. ❌硬件有，但SDK计划不支持 | AT1K | STM32 |
| 输入浮空 | ✅ | ✅ |
| 输入上拉 | ✅ | ✅ |
| 输入下拉 | ✅ | ✅ |
| 模拟输入 | ✅ | ✅ |
| 开漏输出 | ✅ | ✅ |
| 推挽式输出 | ✅ | ✅ |
| 推挽式复用功能 | ✅ | ✅ |
| 开漏复用功能 | ✅ | ✅ |
| 外部中断 | ✅ | ✅ |
| 滤波防抖 | ✅ | ❌ |

API Design:

|  |  |  |
| --- | --- | --- |
|  | AT103 | STM32 |
| init/de-init | GPIO\_AFIODeInit(void)  GPIO\_DeInit(GPIO\_TypeDef\* GPIOx)  void GPIO\_Init(GPIO\_TypeDef \*GPIOx, GPIO\_InitTypeDef \*GPIO\_InitStruct)  void EXTI\_Init(GPIO\_TypeDef \*GPIOx, EXTI\_InitTypeDef \*EXTI\_InitStruct)  void EXTI\_StructInit(EXTI\_InitTypeDef \*EXTI\_InitStruct) | GPIO\_AFIODeInit(void)  GPIO\_DeInit(GPIO\_TypeDef\* GPIOx)  GPIO\_Init(GPIO\_TypeDef\* GPIOx, GPIO\_InitTypeDef\* GPIO\_InitStruct)  EXTI\_Init(EXTI\_InitTypeDef\* EXTI\_InitStruct)  void EXTI\_StructInit(EXTI\_InitTypeDef \*EXTI\_InitStruct) |
| Config | GPIO\_EXTILineConfig(uint8\_t GPIO\_PortSource, uint8\_t GPIO\_PinSource) | GPIO\_ETH\_MediaInterfaceConfig(uint32\_t GPIO\_ETH\_MediaInterface)  GPIO\_EXTILineConfig(uint8\_t GPIO\_PortSource, uint8\_t GPIO\_PinSource)  GPIO\_PinLockConfig(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)  GPIO\_EventOutputConfig(uint8\_t GPIO\_PortSource, uint8\_t GPIO\_PinSource) |
| output | GPIO\_ResetBits(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)  GPIO\_SetBits(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)  GPIO\_StructInit(GPIO\_InitTypeDef\* GPIO\_InitStruct)  GPIO\_Write(GPIO\_TypeDef\* GPIOx, uint16\_t PortVal)  GPIO\_WriteBit(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin, BitAction BitVal) | GPIO\_EventOutputCmd(FunctionalState NewState)  GPIO\_ResetBits(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)  GPIO\_SetBits(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)  GPIO\_StructInit(GPIO\_InitTypeDef\* GPIO\_InitStruct)  GPIO\_Write(GPIO\_TypeDef\* GPIOx, uint16\_t PortVal)  GPIO\_WriteBit(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin, BitAction BitVal) |
| read input and output | GPIO\_ReadOutputDataBit(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)  GPIO\_ReadInputData(GPIO\_TypeDef\* GPIOx)  GPIO\_ReadInputDataBit(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)  GPIO\_ReadOutputData(GPIO\_TypeDef\* GPIOx) | GPIO\_ReadOutputDataBit(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)  GPIO\_ReadInputData(GPIO\_TypeDef\* GPIOx)  GPIO\_ReadInputDataBit(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)  GPIO\_ReadOutputData(GPIO\_TypeDef\* GPIOx) |
| remap | GPIO\_PinRemapConfig(uint32\_t GPIO\_Remap, FunctionalState NewState) | GPIO\_PinRemapConfig(uint32\_t GPIO\_Remap, FunctionalState NewState) |
| interrupt status | EXTI\_GetFlagStatus(uint32\_t EXTI\_Line)  EXTI\_ClearFlag(uint32\_t EXTI\_Line)  EXTI\_GetITStatus(uint32\_t EXTI\_Line)  void EXTI\_ClearITPendingBit(GPIO\_TypeDef \*GPIOx, uint32\_t EXTI\_Line); | EXTI\_GetFlagStatus(uint32\_t EXTI\_Line)  EXTI\_ClearFlag(uint32\_t EXTI\_Line)  EXTI\_GetITStatus(uint32\_t EXTI\_Line)  void EXTI\_ClearITPendingBit(uint32\_t EXTI\_Line) |

## Data structure design:

/\*\*

\* @brief driving strength selection

\*/

typedef enum {

GPIO\_Strength\_Low = 0x1,

GPIO\_Strength\_Middle = 0x2,

GPIO\_Strength\_High = 0x3,

} GPIO\_Strength\_TypeDef;

#define IS\_GPIO\_STRENGTH(STRENGTH) (((STRENGTH) == GPIO\_Strength\_Low) || ((STRENGTH) == GPIO\_Strength\_Middle) || \

(STRENGTH) == GPIO\_Strength\_High)

/\*\*

\* @brief configuration Mode enumeration

\*

\*/

typedef enum {

GPIO\_Mode\_AIN = 0x0,

SCHMITT\_IN = 0x1,

BUFFER\_IN = 0x2,

GPIO\_Mode\_IPD = 0x4,

GPIO\_Mode\_IPU = 0x8,

GPIO\_Mode\_AF = 0x10,

GPIO\_Mode\_OUT\_OD = 0x20,

GPIO\_Mode\_OUT\_PP = 0x40

} GPIOMode\_TypeDef;

#define IS\_GPIO\_MODE(MODE) (((MODE) == SCHMITT\_IN) || ((MODE) == BUFFER\_IN) || \

((MODE) == GPIO\_Mode\_IPD) || ((MODE) == GPIO\_Mode\_IPU))

/\*\*

\* @brief GPIO Init structure definition

\*

\*/

typedef struct

{

uint16\_t GPIO\_Pin; /\*\*<Specifies the GPIO pins to be configured.

This parameter can be any value of @ref GPIO\_pins\_define \*/

GPIO\_Strength\_TypeDef GPIO\_Strength; /\*\*< Specifies the Streegth for the selected pins.\*/

GPIOMode\_TypeDef GPIO\_Mode; /\*\*< Specifies the operating mode for the selected pins.

This parameter can be a value of @ref GPIOMode\_TypeDef \*/

} GPIO\_InitTypeDef;

/\*\*

\* @brief Bit\_SET and Bit\_RESET enumeration

\*/

typedef enum {

Bit\_RESET = 0,

Bit\_SET

} BitAction;

/\*\*

\* @brief EXTI ENABLE enumeration

\*/

typedef enum {

EXTI\_INT\_DISABLE = 0,

EXTI\_INT\_ENABLE

} EXTI\_ENABLE\_TypeDef;

/\*\*

\* @brief EXTI mode enumeration

\*

\*/

typedef enum {

EXTI\_Edge\_Detected = 0,

EXTI\_Level\_Detectd

} EXTI\_Mode\_TypeDef;

/\*\*

\* @brief EXTI Trigger enumeration

\*/

typedef enum {

EXTI\_Trigger\_Falling = 0x1,

EXTI\_Trigger\_Rising = 0x2,

EXTI\_Trigger\_Rising\_Falling

} EXTITrigger\_TypeDef;

/\*\*

\* @brief EXTI Event Mask enumeration

\*/

typedef enum {

EXTI\_EVENT\_MASKED = 0x0,

EXTI\_EVENT\_UNMASKED

} EXT\_EVENT\_MASK\_TypeDef;

/\*\*

\* @brief EXTI Interrupt Mask enumeration

\*/

typedef enum {

EXTI\_INT\_MASKED = 0x0,

EXTI\_INT\_UNMASKED

} EXT\_INT\_MASK\_TypeDef;

/\*\*

\* @brief EXTI Init Structrure definition

\*/

typedef struct

{

uint32\_t EXTI\_Line; /\*\*< Specifies the EXTI lines to be enabled or disabled\*/

uint32\_t Index; /\*\*< EXTI line index.(0 ~ 15) \*/

EXTITrigger\_TypeDef EXTI\_Trigger; /\*\*< Specifies the trigger signal active edge for the EXTI lines\*/

FunctionalState EXTI\_LineCmd; /\*\*< Specifies the new state of the selected EXTI lines\*/

} EXTI\_InitTypeDef;

/\*\*

\* @brief GPIO\_pins\_define

\*/

#define GPIO\_Pin\_0 ((uint16\_t)0x0001) /\*\*< Pin 0 selected \*/

#define GPIO\_Pin\_1 ((uint16\_t)0x0002) /\*\*< Pin 1 selected \*/

#define GPIO\_Pin\_2 ((uint16\_t)0x0004) /\*\*< Pin 2 selected \*/

#define GPIO\_Pin\_3 ((uint16\_t)0x0008) /\*\*< Pin 3 selected \*/

#define GPIO\_Pin\_4 ((uint16\_t)0x0010) /\*\*< Pin 4 selected \*/

#define GPIO\_Pin\_5 ((uint16\_t)0x0020) /\*\*< Pin 5 selected \*/

#define GPIO\_Pin\_6 ((uint16\_t)0x0040) /\*\*< Pin 6 selected \*/

#define GPIO\_Pin\_7 ((uint16\_t)0x0080) /\*\*< Pin 7 selected \*/

#define GPIO\_Pin\_8 ((uint16\_t)0x0100) /\*\*< Pin 8 selected \*/

#define GPIO\_Pin\_9 ((uint16\_t)0x0200) /\*\*< Pin 9 selected \*/

#define GPIO\_Pin\_10 ((uint16\_t)0x0400) /\*\*< Pin 10 selected \*/

#define GPIO\_Pin\_11 ((uint16\_t)0x0800) /\*\*< Pin 11 selected \*/

#define GPIO\_Pin\_12 ((uint16\_t)0x1000) /\*\*< Pin 12 selected \*/

#define GPIO\_Pin\_13 ((uint16\_t)0x2000) /\*\*< Pin 13 selected \*/

#define GPIO\_Pin\_14 ((uint16\_t)0x4000) /\*\*< Pin 14 selected \*/

#define GPIO\_Pin\_15 ((uint16\_t)0x8000) /\*\*< Pin 15 selected \*/

#define GPIO\_Pin\_All ((uint16\_t)0xFFFF) /\*\*< All pins selected \*/

/\*\*

\* @brief GPIO\_remap\_define

\*

\*/

#define GPIO\_Remap\_SPI1 ((uint32\_t)0x00000001U) /\*\*< SPI1remapping \*/

#define GPIO\_Remap\_I2C1 ((uint32\_t)0x00000002U) /\*\*< I2C1 remapping \*/

#define GPIO\_Remap\_USART1 ((uint32\_t)0x00000004U) /\*\*< USART1 remapping \*/

#define GPIO\_Remap\_USART2 ((uint32\_t)0x00000008U) /\*\*< USART2 remapping \*/

#define USART3\_PARTIAL\_REMAP ((uint32\_t)0x00000010U) /\*\*< USART3 partial remapping \*/

#define USART3\_FULL\_REMAP ((uint32\_t)0x00000030U) /\*\*< USART3 full remapping \*/

#define GPIO\_TIMER1\_PARTIAL\_REMAP ((uint32\_t)0x00000040U) /\*\*< TIMER1 partial remapping \*/

#define GPIO\_TIMER1\_FULL\_REMAP ((uint32\_t)0x000000C0U) /\*\*< TIMER1 full remapping \*/

#define GPIO\_PartialRemap1\_TIM2 ((uint32\_t)0x00000100U) /\*\*< TIM2 Partial1 Alternate Function mapping \*/

#define GPIO\_PartialRemap2\_TIM2 ((uint32\_t)0x00000200U) /\*\*< TIM2 Partial2 Alternate Function mapping \*/

#define GPIO\_PartialRemap\_TIM3 ((uint32\_t)0x00000400U) /\*\*< TIM3 Partial Alternate Function mapping \*/

#define GPIO\_FullRemap\_TIM3 ((uint32\_t)0x00000C00U) /\*\*< TIM3 Full Alternate Function mapping \*/

#define GPIO\_Remap\_TIM4 ((uint32\_t)0x00001000U) /\*\*< TIM4 Alternate Function mapping \*/

#define GPIO\_Remap1\_CAN ((uint32\_t)0x00002000U) /\*\*< CAN1 Alternate Function mapping \*/

#define GPIO\_Remap2\_CAN ((uint32\_t)0x00006000U) /\*\*< CAN1 Alternate Function mapping \*/

#define GPIO\_Remap\_PD01 ((uint32\_t)0x00008000U) /\*\*< PD01 Alternate Function mapping \*/

#define GPIO\_Remap\_SPI2 ((uint32\_t)0x00010000U) /\*\*< SPI2 remapping \*/

#define IS\_GPIO\_ALL\_PERIPH(PERIPH) (((PERIPH) == GPIOA) || \

((PERIPH) == GPIOB) || \

((PERIPH) == GPIOC) || \

((PERIPH) == GPIOD) || \

((PERIPH) == GPIOE))

#define IS\_GPIO\_PIN(PIN) ((((PIN) & (uint16\_t)0x00) == 0x00) && ((PIN) != (uint16\_t)0x00))

/\*\*

\* @brief EXTI\_Lines

\*

\*/

#define EXTI\_Line0 ((uint32\_t)0x0001)

#define EXTI\_Line1 ((uint32\_t)0x0002)

#define EXTI\_Line2 ((uint32\_t)0x0004)

#define EXTI\_Line3 ((uint32\_t)0x0008)

#define EXTI\_Line4 ((uint32\_t)0x0010)

#define EXTI\_Line5 ((uint32\_t)0x0020)

#define EXTI\_Line6 ((uint32\_t)0x0040)

#define EXTI\_Line7 ((uint32\_t)0x0080)

#define EXTI\_Line8 ((uint32\_t)0x0100)

#define EXTI\_Line9 ((uint32\_t)0x0200)

#define EXTI\_Line10 ((uint32\_t)0x0400)

#define EXTI\_Line11 ((uint32\_t)0x0800)

#define EXTI\_Line12 ((uint32\_t)0x1000)

#define EXTI\_Line13 ((uint32\_t)0x2000)

#define EXTI\_Line14 ((uint32\_t)0x4000)

#define EXTI\_Line15 ((uint32\_t)0x8000)

#define IS\_EXTI\_LINE(LINE) (((LINE) & (uint32\_t)0xFFFF0000 == 0x0) && (LINE) != (uint16\_t)0x0)

#define IS\_GET\_EXTI\_Line(LINE) ((LINE = EXTI\_Line0) || (LINE = EXTI\_Line1) || \

(LINE = EXTI\_Line2) || (LINE = EXTI\_Line3) || \

(LINE = EXTI\_Line4) || (LINE = EXTI\_Line5) || \

(LINE = EXTI\_Line6) || (LINE = EXTI\_Line7) || \

(LINE = EXTI\_Line8) || (LINE = EXTI\_Line9) || \

(LINE = EXTI\_Line10) || (LINE = EXTI\_Line11) || \

(LINE = EXTI\_Line12) || (LINE = EXTI\_Line13) || \

(LINE = EXTI\_Line14) || (LINE = EXTI\_Line15))

#define EXTI\_LINENONE ((uint32\_t)0x0000)