



USART+中断机制讲解

目录

使用 STM32CubeMX 配置 USART	2
初始化代码.....	3
常用 USART 相关操作函数 stm32f4xx_hal_uart.c	3
1.1 开始串口中断接收 HAL_UART_Receive_IT	3
1.2 开启串口 DMA 接收 HAL_UART_Receive_DMA.....	4
1.3 串口接收中断回调函数 HAL_UART_RxCpltCallback.....	5
中断机制讲解.....	6
中断流程.....	6
判断是何种错误发生.....	6



使用 STM32CubeMX 配置 USART

STM32CubeMX Project_TestIoc*: STM32F405RGTx

File Window Help

Home STM32F405RGTx Project_TestIoc - Pinout & Configuration GENERATE CODE

Pinout & Configuration Clock Configuration Project Manager Tools

USART1 Mode and Configuration

Mode: Asynchronous 模式-异步通信
Hardware Flow Control (RS232): Disable 硬件流控-无

USART配置界面

Configuration

Parameter Settings

Configure the below parameters:

Basic Parameters

Baud Rate: 115200 Bits/s 波特率-115200
Word Length: 8 Bits (including Parity) 字长-8
Parity: None 校验位-无
Stop Bits: 1 停止位-1
Data Direction: Receive and Transmit 数据方向-接收、发送
Over Sampling: 16 Samples 超时-16

Advanced Parameters

Pinout view

System view

STM32F405RGTx LQFP64

PA13: USART1_RX
PA10: USART1_TX

STM32CubeMX Project_TestIoc*: STM32F405RGTx

File Window Help

Home STM32F405RGTx Project_TestIoc - Pinout & Configuration GENERATE CODE

Pinout & Configuration Clock Configuration Project Manager Tools

NVIC Mode and Configuration

Configuration

Priority Group: 4 bits for pre-emption...
Sort by Preemption Priority and Sub Priority
Sort by interrupts names

Search: Search (Ctrl+F)

Show only enabled interrupts
Force DMA channels interrupts

NVIC Interrupt Table

Interrupt	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	0	0	0
Hard fault interrupt	0	0	0
Memory management fault	0	0	0
Pre-fetch fault, memory access fault	0	0	0
Undefined instruction or illegal state	0	0	0
System service call via SWI instruction	0	0	0
Debug monitor	0	0	0
Pendable request for system service	0	0	0
System tick timer	0	0	0
PVD interrupt through EXTI line 16	0	0	0
Flash global interrupt	0	0	0
RCC global interrupt	0	0	0
EXTI line4 interrupt	0	0	0
ADC1, ADC2 and ADC3 global interrupts	0	0	0
TIM2 global interrupt	0	0	0
USART1 global interrupt	1	4	0
Timer/TIM1 trigger and commutation interrupts and TIM14 global interrupt	0	0	0
DMA2 stream0 global interrupt	0	0	0
FPU global interrupt	0	0	0

中断配置界面

使能USART1全局中断

优先级为4

Pinout view

System view

STM32F405RGTx LQFP64

PA13: USART1_RX
PA10: USART1_TX



初始化代码

在 usart.c 源文件中，有如下配置代码

```
1. void MX_USART1_UART_Init(void)
2. {
3.
4.     /* USER CODE BEGIN USART1_Init 0 */
5.
6.     /* USER CODE END USART1_Init 0 */
7.
8.     /* USER CODE BEGIN USART1_Init 1 */
9.
10.    /* USER CODE END USART1_Init 1 */
11.    huart1.Instance = USART1;
12.    huart1.Init.BaudRate = 115200;
13.    huart1.Init.WordLength = UART_WORDLENGTH_8B;
14.    huart1.Init.StopBits = UART_STOPBITS_1;
15.    huart1.Init.Parity = UART_PARITY_NONE;
16.    huart1.Init.Mode = UART_MODE_TX_RX;
17.    huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
18.    huart1.Init.OverSampling = UART_OVERSAMPLING_16;
19.    if (HAL_UART_Init(&huart1) != HAL_OK)
20.    {
21.        Error_Handler();
22.    }
23.    /* USER CODE BEGIN USART1_Init 2 */
24.
25.    /* USER CODE END USART1_Init 2 */
26.
27. }
```

常用 USART 相关操作函数 stm32f4xx_hal_uart.c

1.1 开始串口中断接收 HAL_UART_Receive_IT

```
1. HAL_StatusTypeDef HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)
2. {
3.     /* Check that a Rx process is not already ongoing */
4.     if (huart->RxState == HAL_UART_STATE_READY)
5.     {
```



```
6.  if ((pData == NULL) || (Size == 0U))
7.  {
8.      return HAL_ERROR;
9.  }
10.
11.  /* Process Locked */
12.  __HAL_LOCK(huart);
13.
14.  /* Set Reception type to Standard reception */
15.  huart->ReceptionType = HAL_UART_RECEPTION_STANDARD;
16.
17.  return(UART_Start_Receive_IT(huart, pData, Size));
18. }
19. else
20. {
21.     return HAL_BUSY;
22. }
23. }
```

入口参数: USART 句柄、数据存放地址指针、接收数据的大小(Byte)

返回值: HAL_ERROR(失败) or Data or HAL_BUSY(繁忙)

使用示例:

```
1. uint8_t Buffer;
2. HAL_UART_Receive_IT(&huart1,&Buffer,1);
```

1.2 开启串口 DMA 接收 HAL_UART_Receive_DMA

```
1. HAL_StatusTypeDef HAL_UART_Receive_DMA(UART_HandleTypeDef *huart, uint8_t *p
   Data, uint16_t Size)
2. {
3.  /* Check that a Rx process is not already ongoing */
4.  if (huart->RxState == HAL_UART_STATE_READY)
5.  {
6.      if ((pData == NULL) || (Size == 0U))
7.      {
8.          return HAL_ERROR;
9.      }
10.
11.     /* Process Locked */
12.     __HAL_LOCK(huart);
13.
14.     /* Set Reception type to Standard reception */
15.     huart->ReceptionType = HAL_UART_RECEPTION_STANDARD;
```



```
16.  
17.  return(UART_Start_Receive_DMA(huart, pData, Size));  
18. }  
19. else  
20. {  
21.  return HAL_BUSY;  
22. }  
23. }
```

入口参数：USART 句柄、数据存放地址指针、接收数据的大小(Byte)

返回值：HAL_ERROR(失败) or Data or HAL_BUSY(繁忙)

使用示例：

```
1. uint8_t Buffer[10];  
2. HAL_UART_Receive_DMA(&huart1, Buffer, 10);
```

1.3 串口接收中断回调函数 HAL_UART_RxCpltCallback

```
1. __weak void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)  
2. {  
3.  /* Prevent unused argument(s) compilation warning */  
4.  UNUSED(huart);  
5.  /* NOTE: This function should not be modified, when the callback is needed,  
6.           the HAL_UART_RxCpltCallback could be implemented in the user file  
7.  */  
8. }
```

在 HAL 库中，我们对__weak 函数重写，将函数重写至 usart.c 原文件中

使用示例：

```
1. void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)  
2. {  
3.  /* Prevent unused argument(s) compilation warning */  
4.  UNUSED(huart);  
5.  /* NOTE: This function should not be modified, when the callback is needed  
6.           the HAL_UART_RxCpltCallback could be implemented in the user file  
7.  */  
8.  if(huart->Instance == USART1)  
9.  {  
10.   //用户代码  
11.   HAL_UART_Receive_IT(&huart1, &Buffer, 1);  
12. }
```



```
13. }
```

- >判断是否是串口 1 发生了接收中断
- >用户代码处理
- >重新开始接收 1 个字节的串口数据

中断机制讲解

HAL 库的中断机制大同小异，这里以 USART 串口中断为例，讲解 HAL 库中的中断

中断流程

在 stm32f4xx_it.c 源文件中，存放着单片机所有的硬件中断服务函数

以下为串口中断的流程

>硬件调用函数 USART1_IRQHandler

```
1. void USART1_IRQHandler(void)
2. {
3. /* USER CODE BEGIN USART1_IRQn 0 */
4.
5. /* USER CODE END USART1_IRQn 0 */
6. HAL_UART_IRQHandler(&huart1);
7. /* USER CODE BEGIN USART1_IRQn 1 */
8.
9. /* USER CODE END USART1_IRQn 1 */
10. }
```

>在中断服务函数中，调用了中断处理函数 HAL_UART_IRQHandler

```
1. HAL_UART_IRQHandler(&huart1);
```

>在中断处理函数中，进行了一系列串口错误的判断(parity,noise,frame,over-run...)

>如果某些错误发生，则停止串口接收

>如果没有错误发生，调用中断接收回调函数 HAL_UART_RxCpltCallback，也就是之前用户重写的函数，在中断接收回调函数中，用户可进行数据的处理，以及开启下一次接收等操作。

判断是何种错误发生

在函数 HAL_UART_IRQHandler 中，有进行类似以下的判断

```
1. if (((isrflags & USART_SR_PE) != RESET) && ((cr1its & USART_CR1_PEIE) != RES
    ET))
2. {
3. huart->ErrorCode |= HAL_UART_ERROR_PE;
```



4. }

- >每当有错误发生，错误代码 != 错误位
- >用户可根据 `huart->ErrorCode` 判断是何种错误

6-APR-2021

厦大嘉庚 TCR 嵌入式