嵌入式系统原理及实验

顾　震

信息科学与工程学院自动化系

华东理工大学

Email: guzhen@ecust.edu.cn

课程大纲

1. 嵌入式系统导论
2. Cortex-M3微处理器
3. STM32最小系统及开发环境
4. 嵌入式C语言
5. 通用输入输出GPIO模块
6. 中断
7. 定时器原理与应用
8. USART通信原理及实现
9. DMA控制器
10. SPI与I2C通信原理及实现
11. 模数转换原理及实现
12. 人工智能辅助的嵌入式项目开发
13. 嵌入式应用前沿

8. USART通信原理及实现

本章知识与能力要求

- 理解和掌握通信的基本概念；
- 了解常见的串行通信接口；
- 掌握USART外设及应用；
- 掌握基于HAL库开发USART应用的方法。

8. USART通信原理及实现

8.1通信协议概述

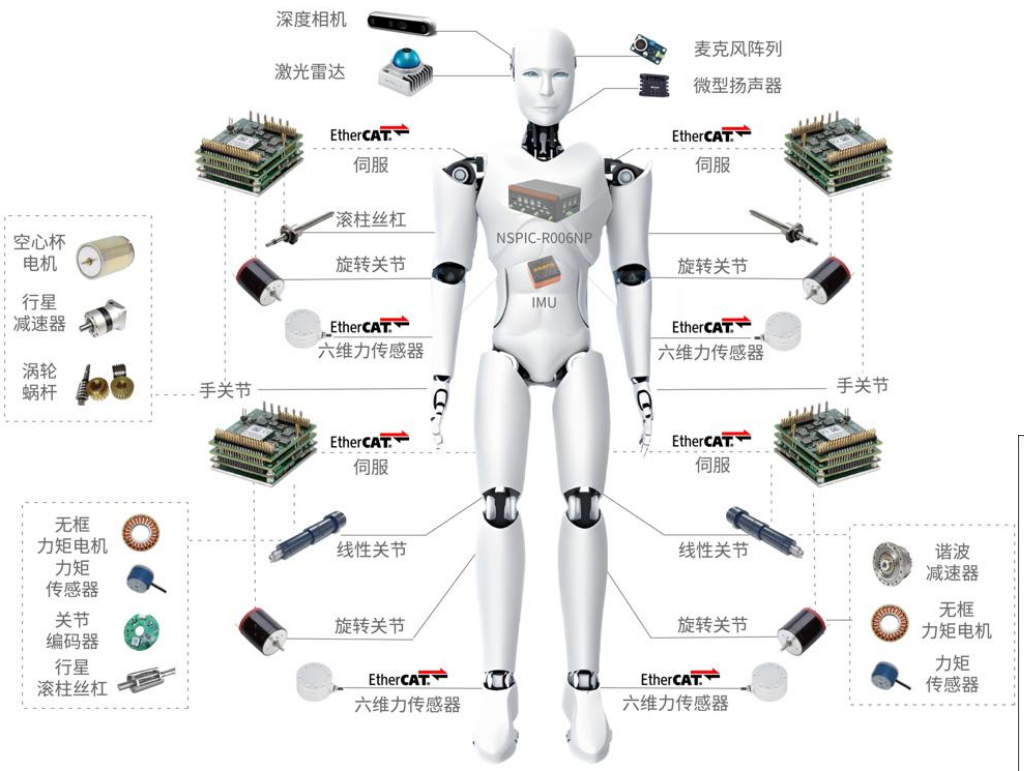
8.2异步串行通信

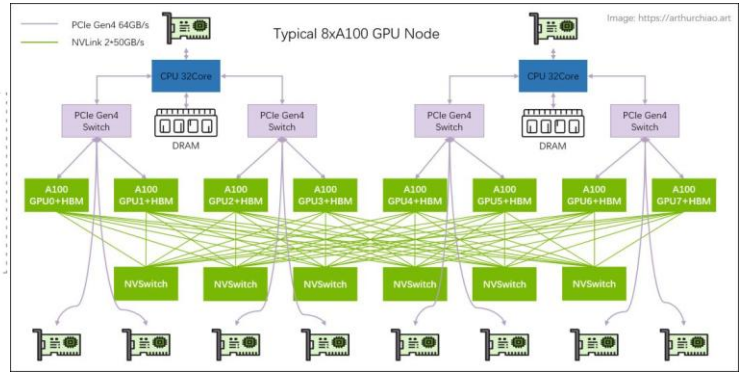
8.3 STM32的USART模块

8.4 USART模块的HAL库接口函数及应用

8.1 通信协议概述

- **通信技术在具身智能、AI等领域具有重要价值**





8.1 通信协议概述

- 现代通信方式种类丰富，分类可以从多个维度划分，不同维度对应不同的技术特性和应用场景。

VGA

串口

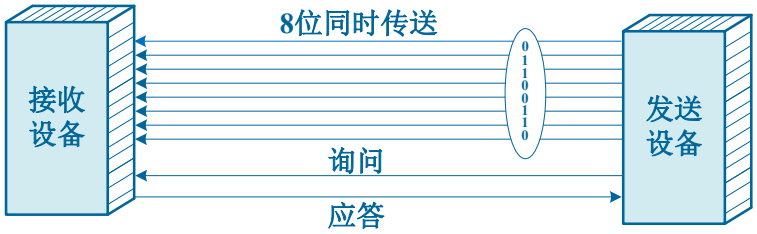
USB

数据传输格式　同步方式　数据传输方向

8.1 通信协议概述

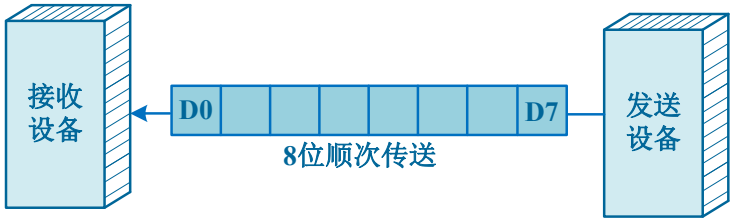
按数据传输格式划分

- **并行通信：**多位数据通过**多条独立数据线**同时传输，每条线对应一位数据。



例： PCI、硬盘接口、计算机内部的总线结构

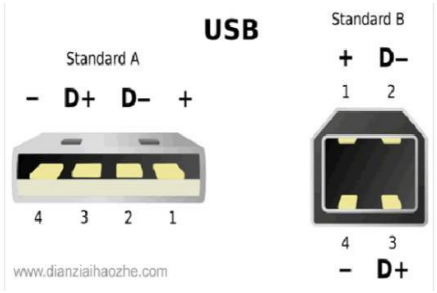
- **串行通信：**数据通过**单条或两条数据线**逐位传输。



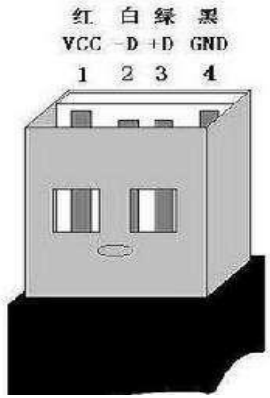
例：串口（UART）、USB、SPI、I2C、以太网、蓝牙。

8.1 通信协议概述

- **串行通信是目前最为流行的一种通信方式，如U盘以及采用USB接口的设备、工业上常用的RS-485和RS-232等，都是串行通信设备。**



USB的串行通信接口



U盘的串行通信接口

计算机网络中所使用的传输方式均是串行传输，单片机与外设之间也大多采用各类串行接口，比如USART、USB、IIC、SPI等。

8.1 通信协议概述

按同步方式划分

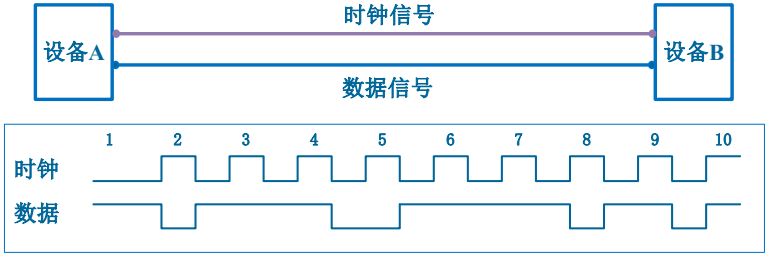
- **异步通信：**发送方和接收方无统一时钟线。

在发送的有效数据中增加一些用于同步的控制位，比如开始位和停止位等，数据以字符为单位组成数据帧进行传送，收发双方需约定数据的传输速率。

例： UART 串口通信

- **同步通信：**发送方通过**时钟线（CLK）**向接收方提供同步信号，接收方严格按时钟节拍采样数据，无需额外起始 / 停止位。

连续串行传送数据的通信方式，要求收发双方的时钟必须保持严格的同步。



例： SPI、I2C、以太网

8.1 通信协议概述

按数据传输方向划分

- **单工通信：**数据只能从一方单向传输到另一方，接收端无法回传数据。

  例：广播、电视信号传输、打印机、BB机

- **半双工通信：**数据可以双向传输，但**不能同时进行。**

  例：对讲机、IIC通信

- **全双工通信：**数据可以**同时双向传输**，发送和接收独立进行。

  例：电话、USB 通信、以太网

8.1 通信协议概述

通信传输速率

- **波特率：**每秒传输的二进制位数，单位为比特每秒（bit/s，bps），是衡量串行数据传输速度快慢的指标。

> 常用的串口传输速率（波特率）有1200、2400、4800、9600、19200和115200等。

- **字符速率：**每秒所传输的字符数：波特率=字符速率×每个字符包含的位数

例：

字符速率为120字符/秒，若每个字符包含10位（1个起始位，7个数据位，1个校验位，1个结束位），则波特率为

10位/字符×120字符/秒＝1200bps。

8.2 异步串行通信

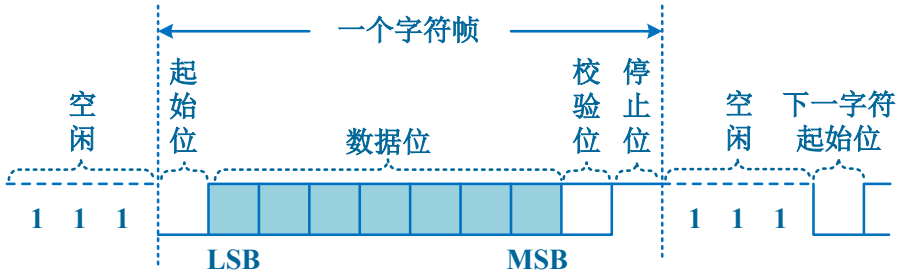
8.2.1 异步串行通信协议

8.2.2 异步串行通信接口

8.2.1 异步串行通信协议

- **通信传输速率和数据帧格式一起称为通信协议。**

异步串行通信标准的数据帧由起始位、数据位、校验位、停止位四部分组成。数据传输速率为50、75、100、150、300、600、1200、2400、4800、9600、19200和38400波特。



起始位：占一位，位于数据帧的开头，以逻辑“0”表示传输数据的开始。

数据位：要发送的数据，数据长度可以是5～8位。

校验位：占一位，用于检测数据是否有效。

停止位：一帧传送结束的标志，根据实际情况定，可以是1、1.5或2位。

空闲位：数据传输完毕，用“1”表示当前线路上没有数据传输。

8.2.1 异步串行通信协议

校验模式

1. 无校验（None）
不额外加校验位，数据位直接接停止位，是最简单、最兼容的方案。
帧结构：起始位→数据位→停止位

2. 偶校验（Even）
让数据位和校验位中 逻辑1的出现次数为**偶数**，通过计数实现简单检错。
帧结构：起始位→数据位→校验位→停止位

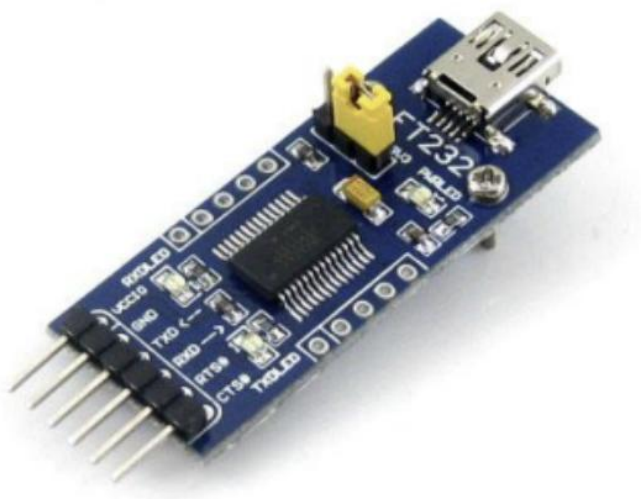
具体实现：
先数数据位里逻辑1的个数 ， 若为偶数，校验位设为 0； 若为奇数，校验位设为 1。

3. 奇校验（Odd）
让数据位和校验位中 逻辑1的出现次数为**奇数**，通过计数实现简单检错。
帧结构：起始位→数据位→校验位→停止位

具体实现：
先数数据位里逻辑1的个数 ， 若为奇数，校验位设为 0； 若为偶数，校验位设为 1。

8.2.1 异步串行通信协议

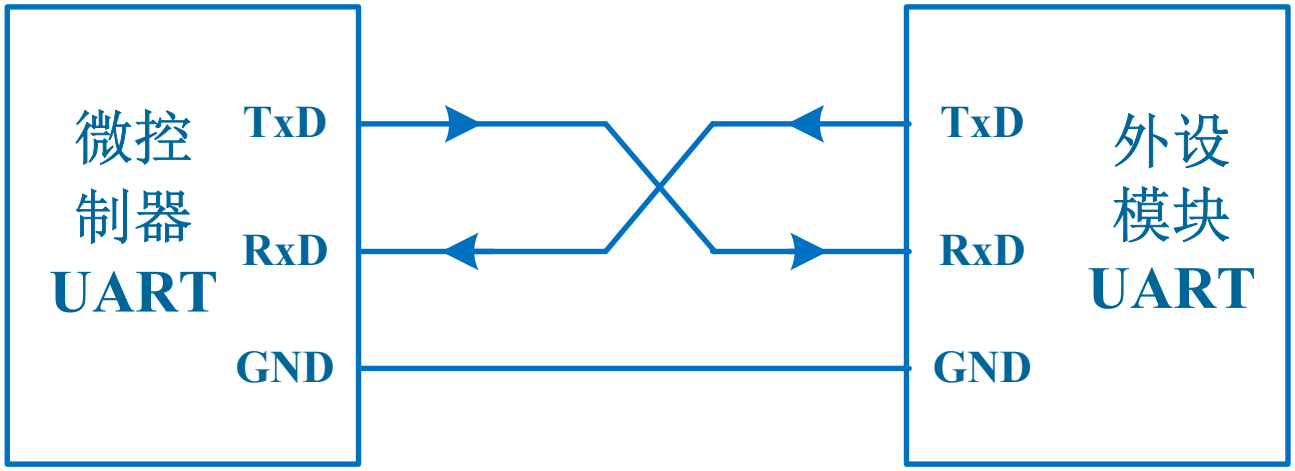
- UART（Universal Asynchronous Receiver Transmitter，通用异步收发传输器）是一个**全双工**通用**异步串行**收/发模块，主要用于打印程序调试信息、上位机和下位机的通信以及ISP程序下载等场合。



8.2.1 异步串行通信协议

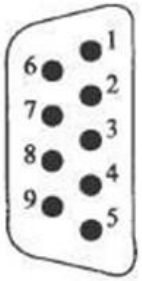
- UART至少需要**两根数据线**用于通信双方进行数据**双向同时传输**，最简单的UART接口由**TxD、RxD、GND**共3根线组成。
- TxD用于发送数据，RxD用于接收数据，GND为信号地线，通过**交叉连接**实现两个芯片间的串口通信。

-RxD：数据输入引脚，数据接受。
-TxD：数据输出引脚，数据发送。



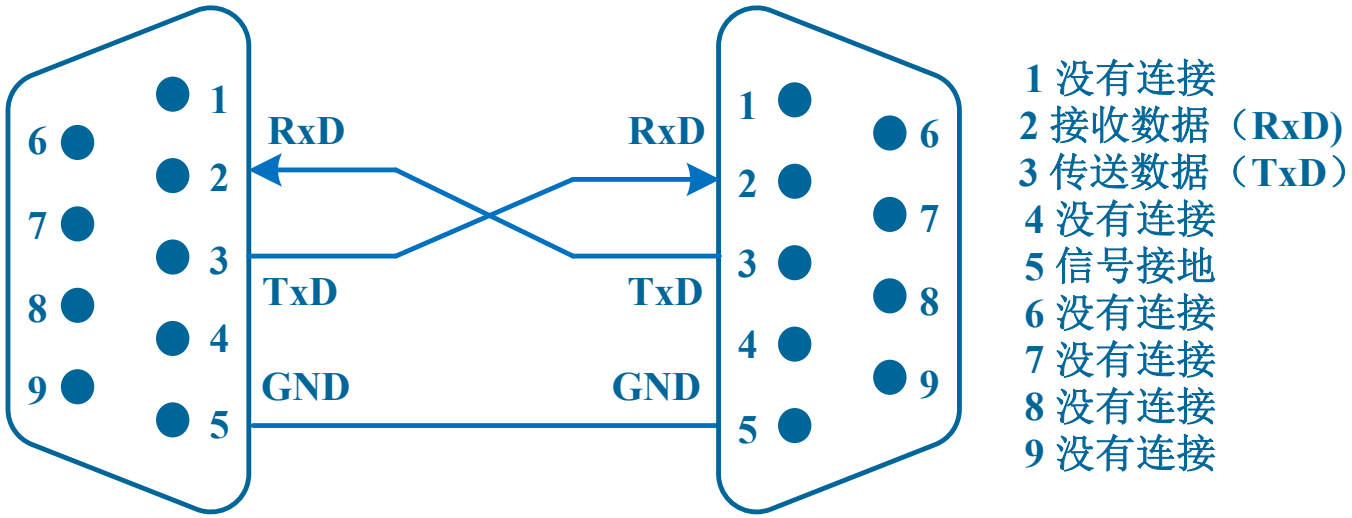
8.2.1 异步串行通信协议

- RS-232是美国电子工业协会（EIA，Electronic Industry Association）制定的串行通信**物理接口标准**，规定了相应的电气特性和物理特性，通常采用DB-9或DB-25的形式，以DB-9最为常见。

外形	针脚号	符号	功能
	1	DCD	数据载波检测，输入
	2	RxD	接收数据，输入
	3	TxD	发送数据，输出
	4	DTR	数据终端准备就绪，输出
	5	GND	信号地
	6	DSR	数据设备准备就绪，输入
	7	RTS	请求发送，输出
	8	CTS	清除发送，输入
	9	RI	振铃指示，输入

8.2.1 异步串行通信协议

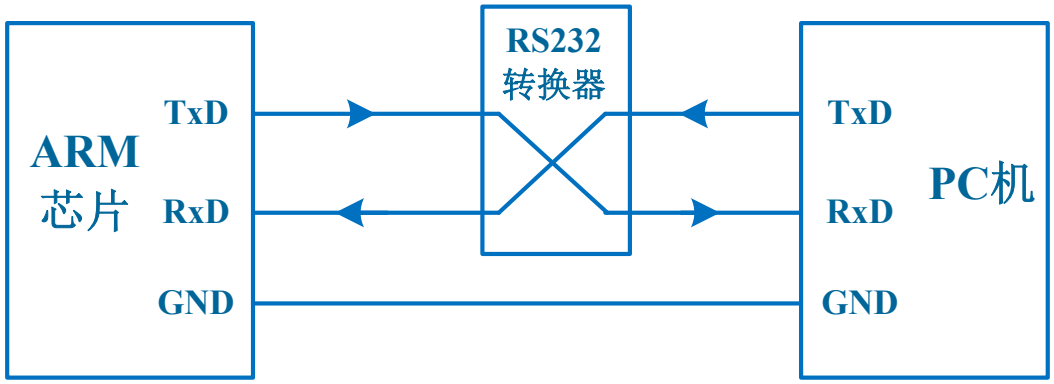
- 对可靠性要求不高的场合，DB-9通常采用三线制串口，仅需**发送（Tx）、接收（Rx）和地（GND）**三条线，即可实现**全双工通信**，进行最基本的数据收发传送，最高传输速率可达20kbps

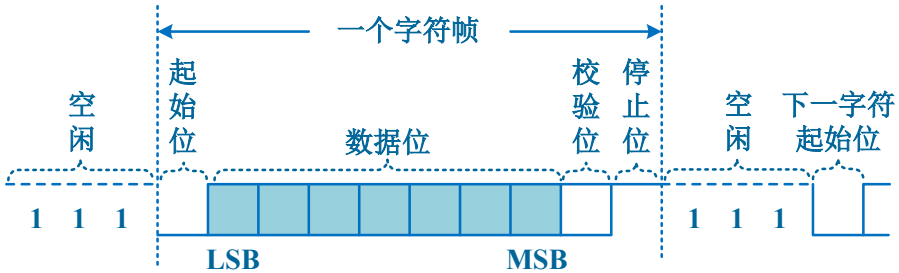


8.2.1 异步串行通信协议

- RS-232为异步串行通信接口，其电气标准采用**负逻辑**，不能与TTL电平设备直接相连，而微控制器中的UART采用的是TTL电平标准，因此，RS-232与微控制器相连时必须进行**电平转换**。

-RxD：数据输入引脚，数据接受。
-TxD：数据输出引脚，数据发送。





如何通过程序来实现UART通信?

8.3 STM32的USART模块

8.3.1 USART内部结构

8.3.2 USART接口

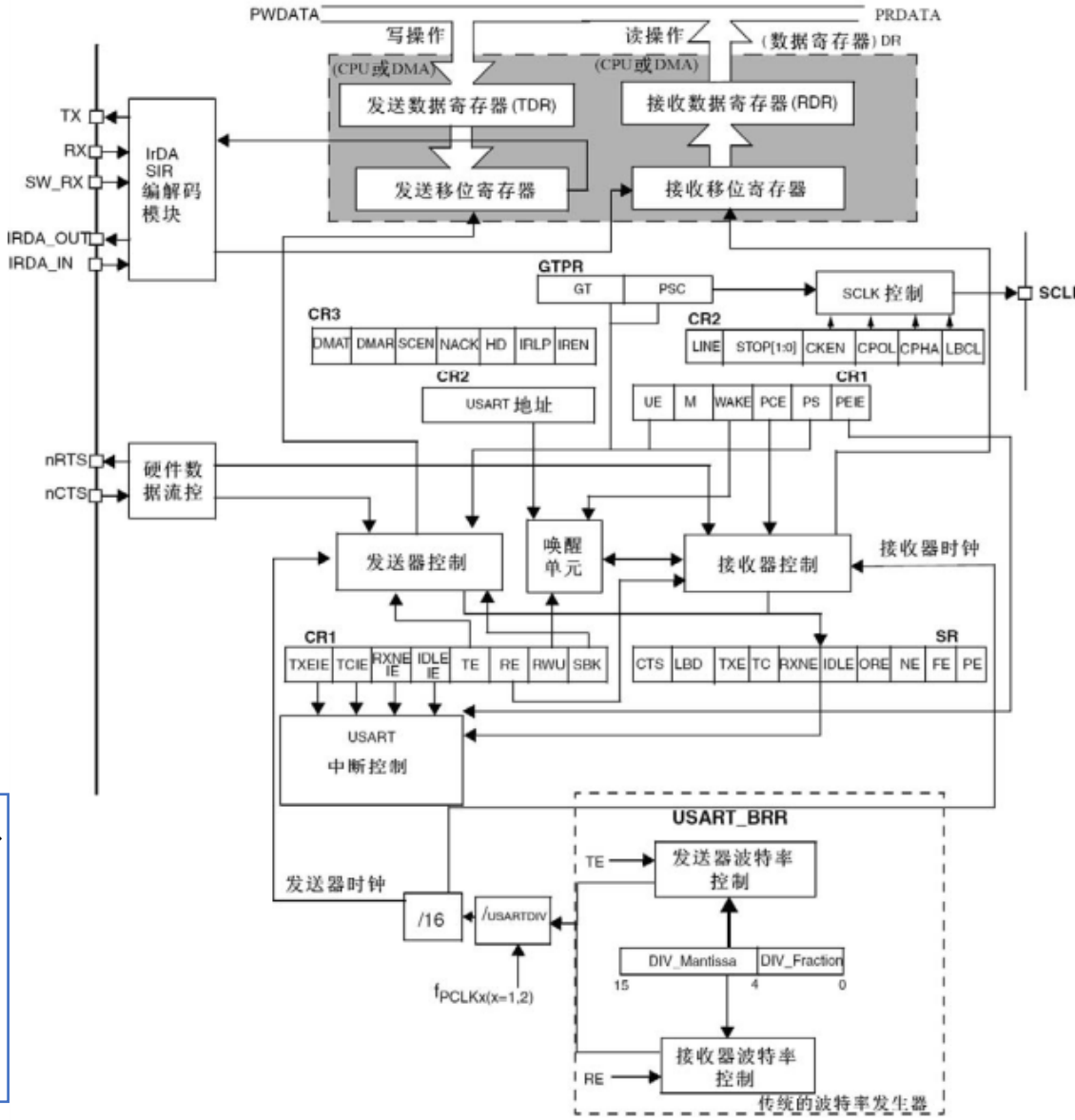
8.3.3 USART编程模式

8.3.1 USART内部结构

USART内部功能实现由三部分构成:

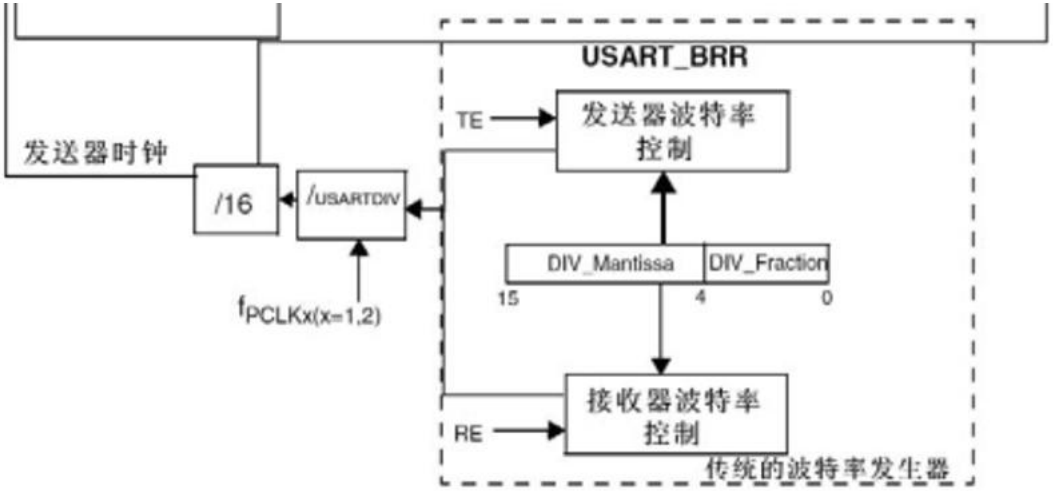
由三个配置寄存器控制

- **CR1主要用于配置USART的数据位、校验位及中断使能;**
- **CR2用于配置USART停止位以及SCLK时钟控制;**
- **CR3主要涉及CTS硬件流控制和单字节/DMA多缓冲传输控制的配置。**



8.3.1 USART内部结构

1. 波特率发生器（Baud Rate Generator）为发送器和接收器提供同步时钟信号，确保收发双方的速率一致（波特率匹配），是串行通信时序的基准。



由系统时钟经过分频电路（分频系数可配置）生成：

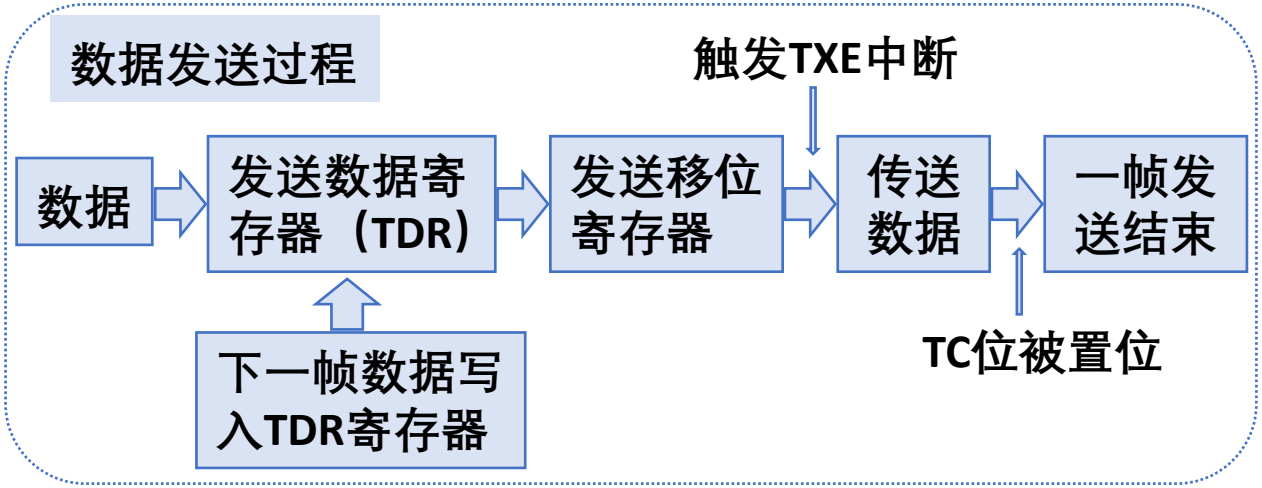
波特率时钟频率 = 波特率 × 采样倍数

例：采样倍数为 16，即每 bit 采样 16 次以提高抗干扰性

8.3.1 USART内部结构

2. 发送器（Transmitter）:将并行数据（来自 CPU 或其他外设）转换为串行数据，并按照通信协议（起始位、数据位、校验位、停止位）发送出去。

工作流程:
1.接收来自 CPU 的并行数据（通常通过数据寄存器写入）；
2.自动添加通信格式所需的控制位（如起始位 “0”、停止位 “1” ）；
3.通过移位寄存器将并行数据逐位转换为串行信号；
4.在波特率时钟的驱动下，将串行数据按设定速率发送到 TX 引脚。



TXE中断：当发送器将发送数据寄存器（TDR）中的数据加载到发送移位寄存器后，TDR 变为空。
TC中断：发送移位寄存器中的最后一位（包括停止位）发送完毕，且 TDR 也为空。

8.3.1 USART内部结构

3. 接收器（Receiver）： 从 RX 引脚接收串行数据，将其转换为并行数据，并验证数据的合法性（如校验位、帧格式），最终提供给 CPU 或其他外设。

工作流程：

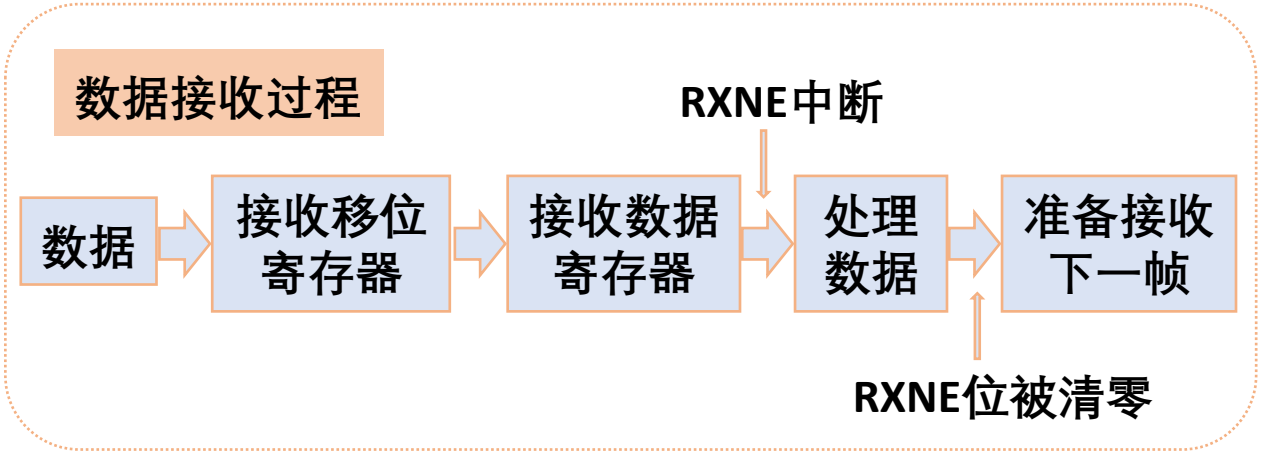
1.检测 RX 引脚上的起始位，触发接收开始；

2.在波特率时钟的同步下，对串行数据进行采样；

3.通过移位寄存器将串行数据逐位拼接为并行数据；

4.验证校验位和帧格式（停止位是否为 “1”），若出错则标记错误状态；

5.将正确的并行数据存入接收数据寄存器（RDR），供 CPU 读取。



RXNE中断：接收器将串行数据转换为并行数据并存入接收数据寄存器（RDR）后，且验证通过。

8.3.2 USART接口

STM32F103大容量产品有5个串行通信接口，USART4和USART5为2个通用异步收发器，不支持同步模式，只有异步通信功能。

USART模式	USART1	USART2	USART3	USART4	USART5
异步模式	✓（支持）	✓	✓	✓	✓
硬件流控制	✓	✓	✓	×（不支持）	×
多缓存通信（DMA）	✓	✓	✓	✓	✓
多处理器通信	✓	✓	✓	✓	✓
同步	✓	✓	✓	×	×
智能卡	✓	✓	✓	×	×
半双工（单线模式）	✓	✓	✓	✓	✓
IrDA	✓	✓	✓	✓	✓
LIN	✓	✓	✓	✓	✓

8.3.2 USART接口

USART异步模式至少需要两个引脚：Rx（接收数据引脚）和Tx（发送数据引脚），使用了复用I/O功能。

总线	USARTx	复用功能	USART1_REMAP=0	USART1_REMAP=1	GPIO配置
APB2总线	USART1	USART1_TX	PA9	PB6	复用推挽输出
		USART1_RX	PA10	PB7	浮空输入
APB1总线	USART2	USART2_TX	PA2	PD5	复用推挽输出
		USART2_RX	PA3	PD6	浮空输入
	USART3	USART3_TX	PB10	PC10	复用推挽输出
		USART3_RX	PB11	PC11	浮空输入

8.3.3 USART编程模式

- **轮询模式**：CPU不断地查询I/O设备是否准备就绪，如果准备就绪就发送，否则提示超时错误；会占用CPU的大量时间，效率低。

- **中断方式**：通过中断请求线，在I/O设备准备就绪时**向CPU发出中断请求**，CPU中止正在进行的工作转向处理I/O设备的中断事件中断方式相比轮询方式效率较高。

- **直接内存访问方式**：直接存储器传送，**不经过CPU**直接在内存和外设之间进行批量数据交换，适用于高速大批量成组数据的传输；满足高速I/O设备的传输要求，有利于提高CPU的利用率。

8.3.3 USART编程模式

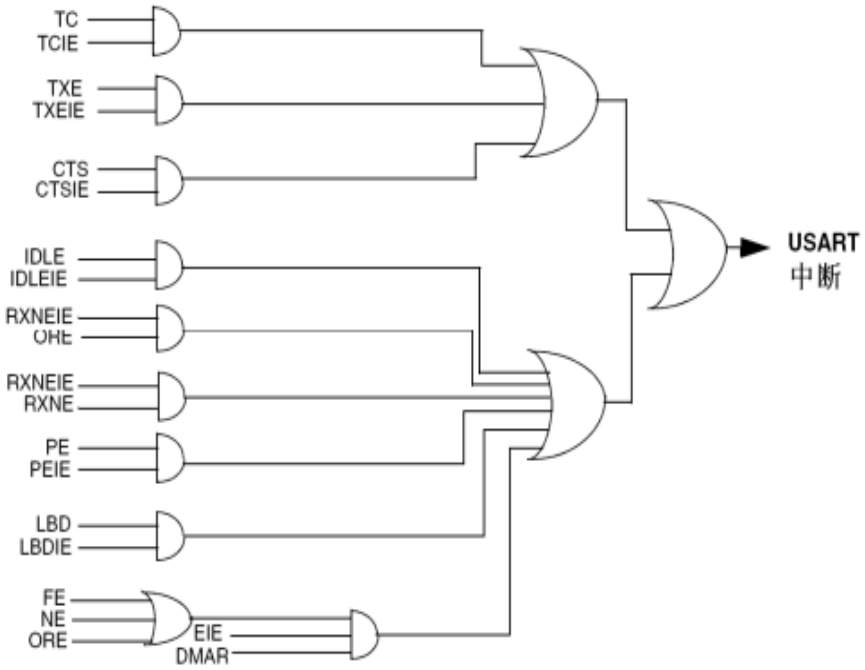
轮询方式

通过定期或循环查询USART状态寄存器SR各位的状态，编写应用程序对接收或发送的数据进行处理。

中断方式

USART主要有以下中断事件：

- ✓ **发送期间：**发送完成、清除发送和发送数据寄存器空；
- ✓ **接收期间：**空闲总线检测、溢出错误、接收数据寄存器非空、校验错误、LIN断开符号检测、噪音标志（仅在多缓冲器通信）和帧错误（仅在多缓冲器通信）。



USART中断映射

8.3.3 USART编程模式

中断方式

USART中断事件及中断使能标志位

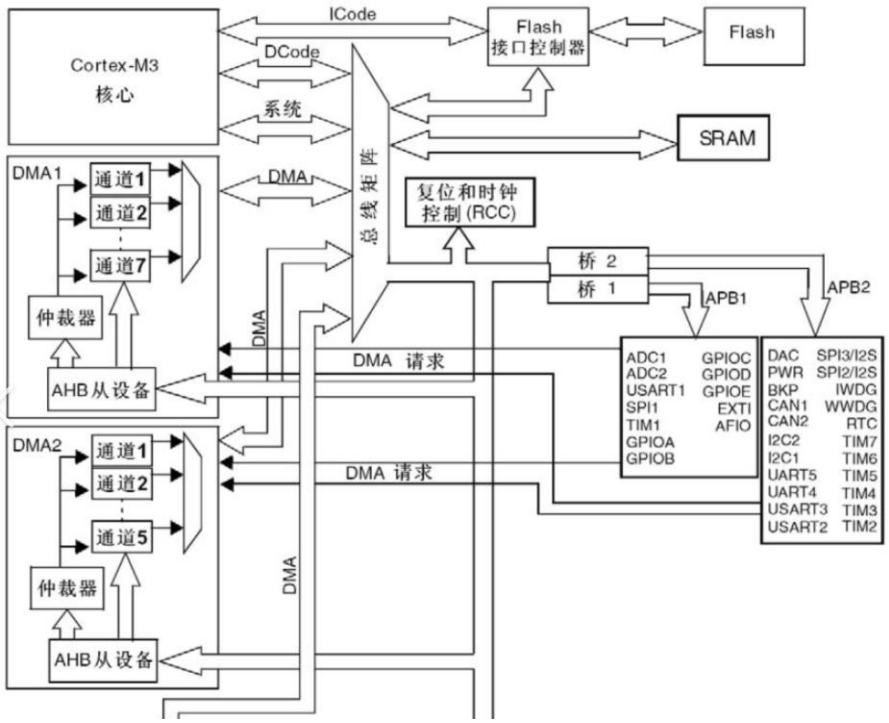
中断标志	中断使能位	STM32中的定义	中断事件
TXE	TXEIE	UART_IT_TXE	发送数据寄存器空
CTS	CTSIE	UART_IT_CTS	CTS标志(Clear to Send Flag)
TC	TCIE	UART_IT_TC	发送完成
RXNE	RXNEIE	UART_IT_RXNE	接收数据就绪（可读）
ORE	TXNEIE		检测到数据溢出
IDLE	IDLEIE	UART_IT_IDLE	检测到空闲线路标志
PE	PEIE	UART_IT_PE	奇偶检验错误标志
LBD	LBDIE	UART_IT_LBD	断开标志
NE或ORT或PE	EIE	UART_IT_ERR	噪声标志，溢出错误和帧错误

8.3.3 USART编程模式

DMA方式

USART模块可通过DMA方式进行大批量数据的连续传输。

STM32F103微控制器的USART的发送和接收分别映射到不同的DMA通道上，如USART1_Rx被映射到DMA1的通道5，USART1_Tx被映射到DMA1的通道4。



8.4 USART模块的HAL库接口函数及应用

8.4.1 UART的HAL库接口函数

8.4.2 USART的HAL库应用实例

8.4.3 USART模块应用实例拓展

8.4.1 UART的HAL库接口函数

```
/* Initialization and de-initialization functions  ****************************/
HAL_StatusTypeDef HAL_UART_Init(UART_HandleTypeDef *huart);
HAL_StatusTypeDef HAL_HalfDuplex_Init(UART_HandleTypeDef *huart);
HAL_StatusTypeDef HAL_LIN_Init(UART_HandleTypeDef *huart, uint32_t BreakDetectLength);
HAL_StatusTypeDef HAL_MultiProcessor_Init(UART_HandleTypeDef *huart, uint8_t Address, uint32_t WakeUpMethod);
HAL_StatusTypeDef HAL_UART_DeInit (UART_HandleTypeDef *huart);
void HAL_UART_MspInit(UART_HandleTypeDef *huart);
void HAL_UART_MspDeInit(UART_HandleTypeDef *huart);

/**
  * @}
  */

/** @addtogroup UART_Exported_Functions_Group2 IO operation functions
  * @{
  */

/* IO operation functions *****************************************************/
HAL_StatusTypeDef HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout);
HAL_StatusTypeDef HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout);
HAL_StatusTypeDef HAL_UART_Transmit_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);
HAL_StatusTypeDef HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);
HAL_StatusTypeDef HAL_UART_Transmit_DMA(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);
HAL_StatusTypeDef HAL_UART_Receive_DMA(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);
HAL_StatusTypeDef HAL_UART_DMAPause(UART_HandleTypeDef *huart);
HAL_StatusTypeDef HAL_UART_DMAResume(UART_HandleTypeDef *huart);
HAL_StatusTypeDef HAL_UART_DMAStop(UART_HandleTypeDef *huart);
void HAL_UART_IRQHandler(UART_HandleTypeDef *huart);
void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart);
void HAL_UART_TxHalfCpltCallback(UART_HandleTypeDef *huart);
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart);
void HAL_UART_RxHalfCpltCallback(UART_HandleTypeDef *huart);
void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart);
```

UART模块的HAL库相关接口函数定义在源文件stm32f1xx_hal_uart.c中，在stm32f1xx_hal_uart.h头文件中可以查看相关库函数声明以及相关的结构体定义。

8.4.1 UART的HAL库接口函数

USART模块的HAL库常用接口函数

类型		函数及功能描述
引脚功能操作函数	轮询方式	HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout); 功能描述：串口轮询模式发送函数，使用超时管理机制
		HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout); 功能描述：串口轮询模式接收函数，使用超时管理机制
	中断方式	HAL_UART_Transmit_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size); 功能描述：串口中断模式发送函数
		HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size); 功能描述：串口中断模式接收函数
	DMA方式	HAL_UART_Transmit_DMA(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size); 功能描述：串口DMA模式发送函数
		HAL_UART_Receive_DMA(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size); 功能描述：串口DMA模式接收函数
		HAL_UART_DMAPause(UART_HandleTypeDef *huart); 功能描述：串口DMA模式暂停函数
		HAL_UART_DMAStop(UART_HandleTypeDef *huart); 功能描述：串口DMA模式停止函数

8.4.1 UART的HAL库接口函数

USART模块的HAL库常用接口函数

类型	函数及功能描述
初始化及复位函数	HAL_UART_Init(UART_HandleTypeDef *huart) 功能描述：UART初始化函数
	HAL_UART_DeInit(UART_HandleTypeDef *huart) 功能描述：UART复位函数
中断处理函数（回调函数）	void HAL_UART_IRQHandler(UART_HandleTypeDef *huart); 功能描述：UART中断处理函数
	void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart); 功能描述：UART数据发送完成时的回调函数，用户在该函数内编写中断处理程序
	void HAL_UART_TxHalfCpltCallback(UART_HandleTypeDef *huart); 功能描述：UART数据发送一半时的回调函数
	void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart); 功能描述：UART数据接收完成时的回调函数，用户在该函数内编写中断处理程序
	void HAL_UART_RxHalfCpltCallback(UART_HandleTypeDef *huart); 功能描述：UART数据接收一半时的回调函数
	void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart); 功能描述：UART传输出现错误时调用的回调函数
外设状态函数	HAL_UART_GetState(UART_HandleTypeDef *huart); 功能描述：获取串口状态函数
	HAL_UART_GetError(UART_HandleTypeDef *huart); 功能描述：获取串口错误函数

8.4.1 UART的HAL库接口函数

UART发送函数HAL_UART_Transmit()

函数原型

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

函数解析

该函数为串口在轮询方式下发送指定长度的数据，采用超时管理机制，若超出设定值时数据还没有发送完成，则不再发送，返回超时标志（HAL_TIMEOUT）。

使用范例

采用串口2发送固定数据，无限等待，间隔1s循环发送。

```
const char  *Send_Data = "Welcome to School ! \r\n";
HAL_UART_Transmit(&huart2, (uint8_t *) Send_Data,
    strlen(Send_Data), 0xFFFF);
HAL_Delay(1000);
```

8.4.1 UART的HAL库接口函数

UART发送函数HAL_UART_Transmit()

HAL_UART_Transmit()共有四个参数

第一个参数huart为结构体类型UART_HandleTypeDef定义的指针，取值为huart1 ~ huart5。

第二个参数uint8_t *pData为要发送的数据，pData为指向数据缓冲区的指针，类型为uint8_t，若发送的数据不是此类型，需要进行类型转换。

第三个参数uint16_t Size为要发送数据的大小。

第四个参数uint32_t Timeout为超时等待时间。

8.4.1 UART的HAL库接口函数

UART接收函数HAL_UART_Receive()

函数原型

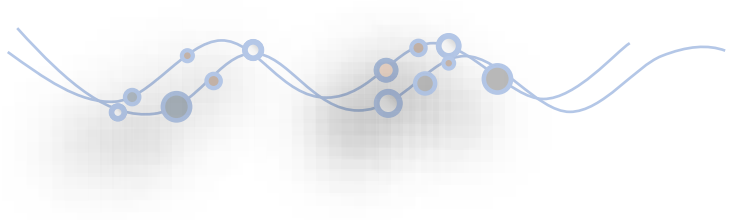
```
HAL_StatusTypeDef  HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData,
uint16_t Size, uint32_t Timeout);
```

函数解析

HAL_UART_Receive()为UART在轮询模式下的接收函数，采用超时管理机制，使用方式同HAL_UART_Transmit()

使用范例

```
uint8_t  ch;
HAL_UART_Receive(&huart3,(uint8_t *)&ch,
1, 1000);
```



8.4.1 UART的HAL库接口函数

UART中断接收函数HAL_UART_Receive_IT( )

```
HAL_StatusTypeDef
HAL_UART_Receive_IT(UART_HandleTypeDef *huart,
uint8_t *pData, uint16_t Size)
{
 if (huart->RxState == HAL_UART_STATE_READY)
    //如果串口处于空闲状态，则执行以下语句
 {
    if ((pData == NULL) || (Size == 0U))
  {     return HAL_ERROR;
  }
  __HAL_LOCK(huart); //配置huart参数前先上锁
     /*以下为结构体变量huart的参数配置
*/
   huart->pRxBuffPtr = pData;  //设置缓存指针
   huart->RxXferSize = Size;  //设置接收数据数量
   huart->RxXferCount = Size; //设置接收计数器
   huart->ErrorCode = HAL_UART_ERROR_NONE;
     //ErrorCode设置为无错误
   huart->RxState = HAL_UART_STATE_BUSY_RX;
    //接收状态设置为接收忙碌
```

```
     /* huart解锁 */
  __HAL_UNLOCK(huart);
     /*使能UART的PE（奇偶检验错误）中断 */
  __HAL_UART_ENABLE_IT(huart, UART_IT_PE);
    /*使能UART错误中断*/
  __HAL_UART_ENABLE_IT(huart, UART_IT_ERR);
     /*使能UART数据寄存器非空中断*/
  __HAL_UART_ENABLE_IT(huart, UART_IT_RXNE);
  return HAL_OK; //配置完成，则返回HAL_OK
 }
 else
 {   return HAL_BUSY;
    //若huart不是READY状态，则返回HAL_BUSY忙状态
 }
}
```

例： uint8_t RxBuffer[10];

HAL_UART_Receive_IT(&huart1,(uint8_t *)&RxBuffer,1);

8.4.2 USART的HAL库应用实例

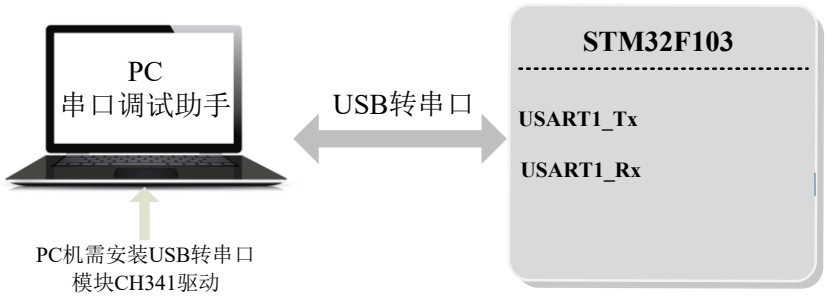
功能

实现简单的STM32串口发送和接收数据，调试此应用实例，需要在PC机端安装“串口调试助手”。

实例效果：

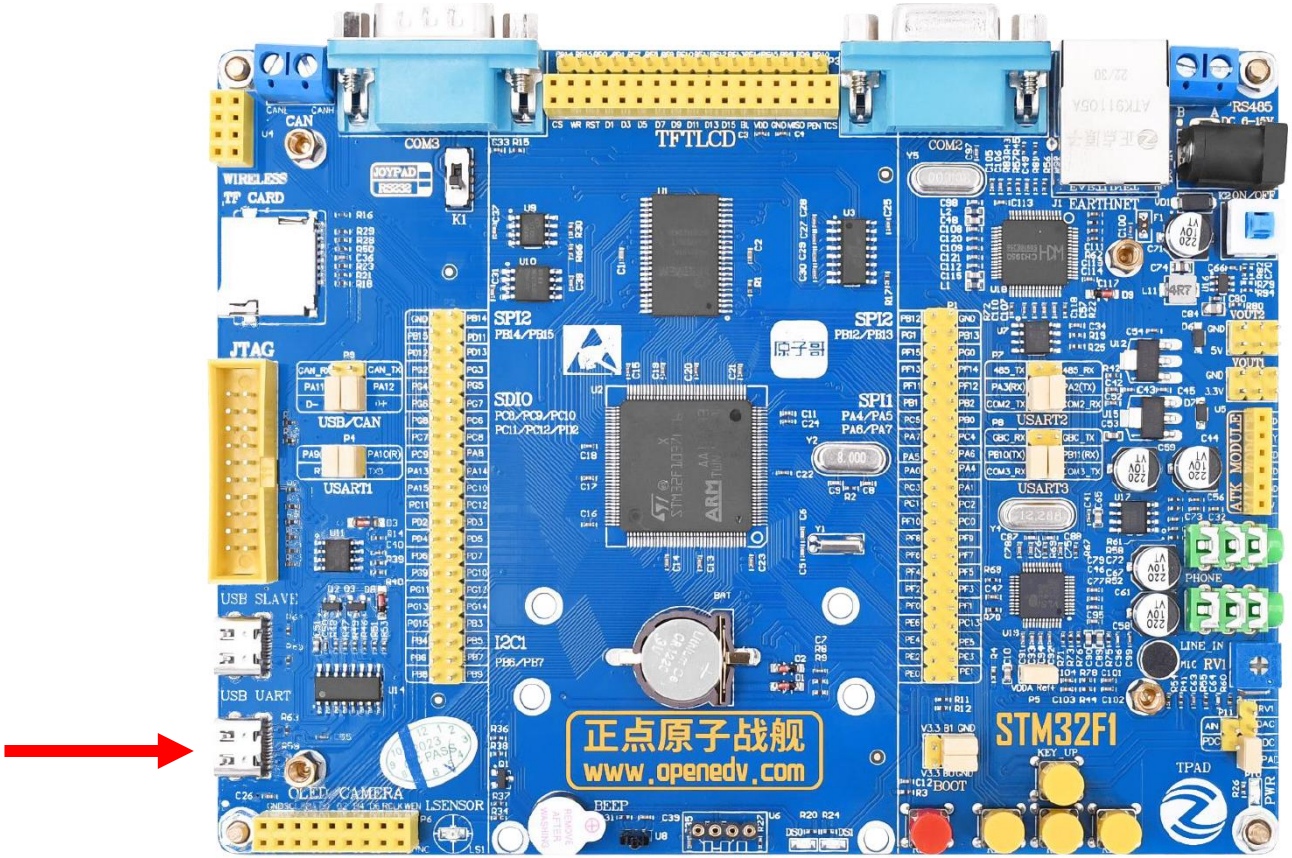
STM32目标板每隔1s循环发送字符“欢迎来到串口通信模块”到PC机上的串口助手。

硬件设计



通过STM32F103目标板上的串口1实现与上位机的通信，可通过串口调试助手查看测试结果。

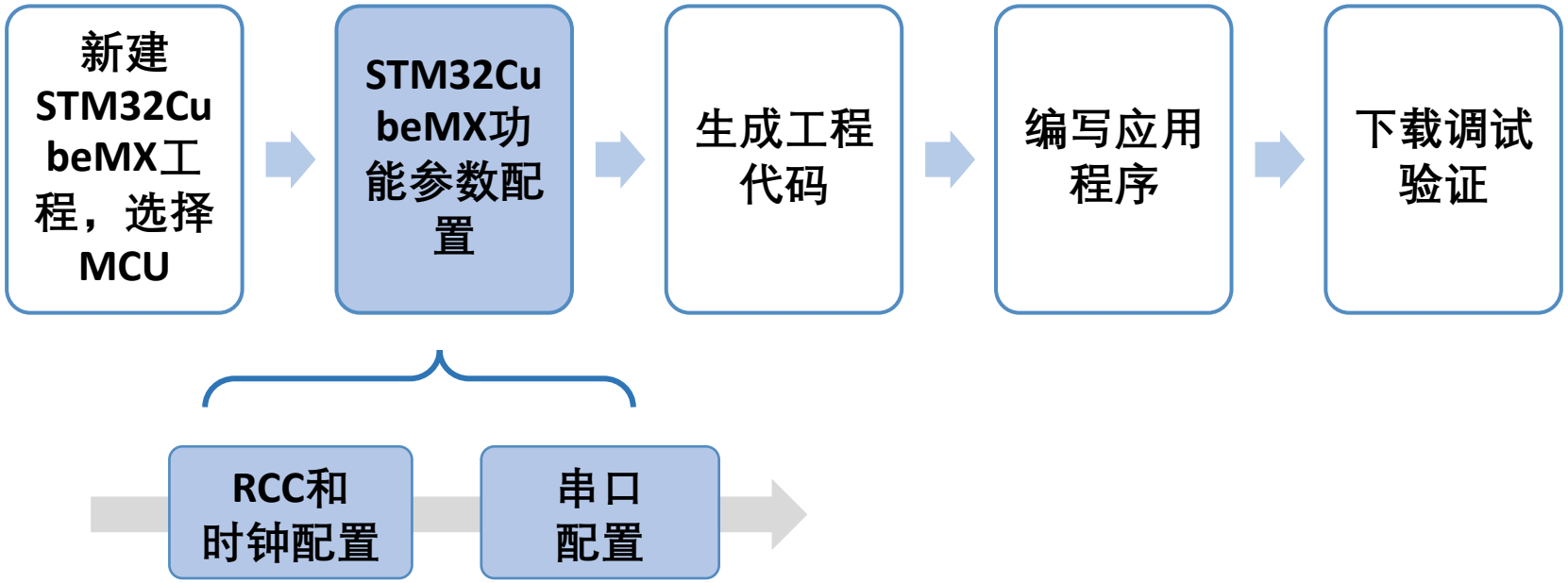
8.4.2 USART的HAL库应用实例



开发板UART接口

8.4.2 USART的HAL库应用实例

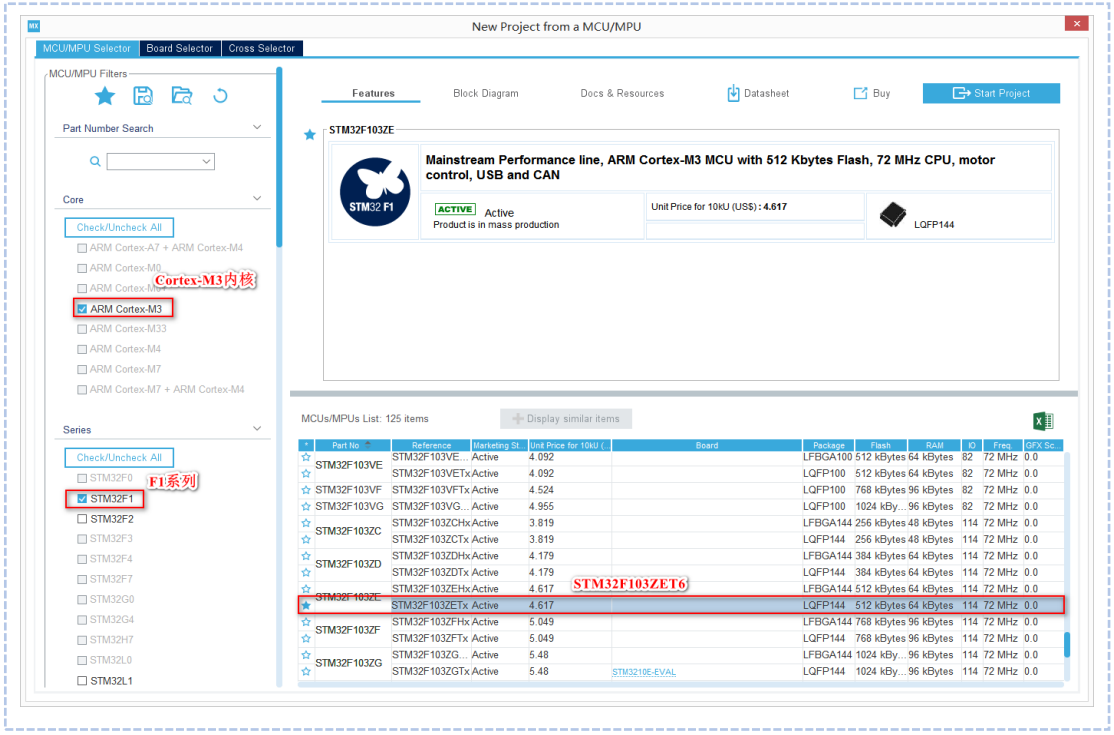
软件设计流程



8.4.2 USART的HAL库应用实例

软件设计——新建STM32CubeMX工程，选择MCU

新建一个STM32CubeMx工程，选择选择MCU，这里选择STM32F103ZETx芯片，读者可根据自己的目标板选择相应的芯片

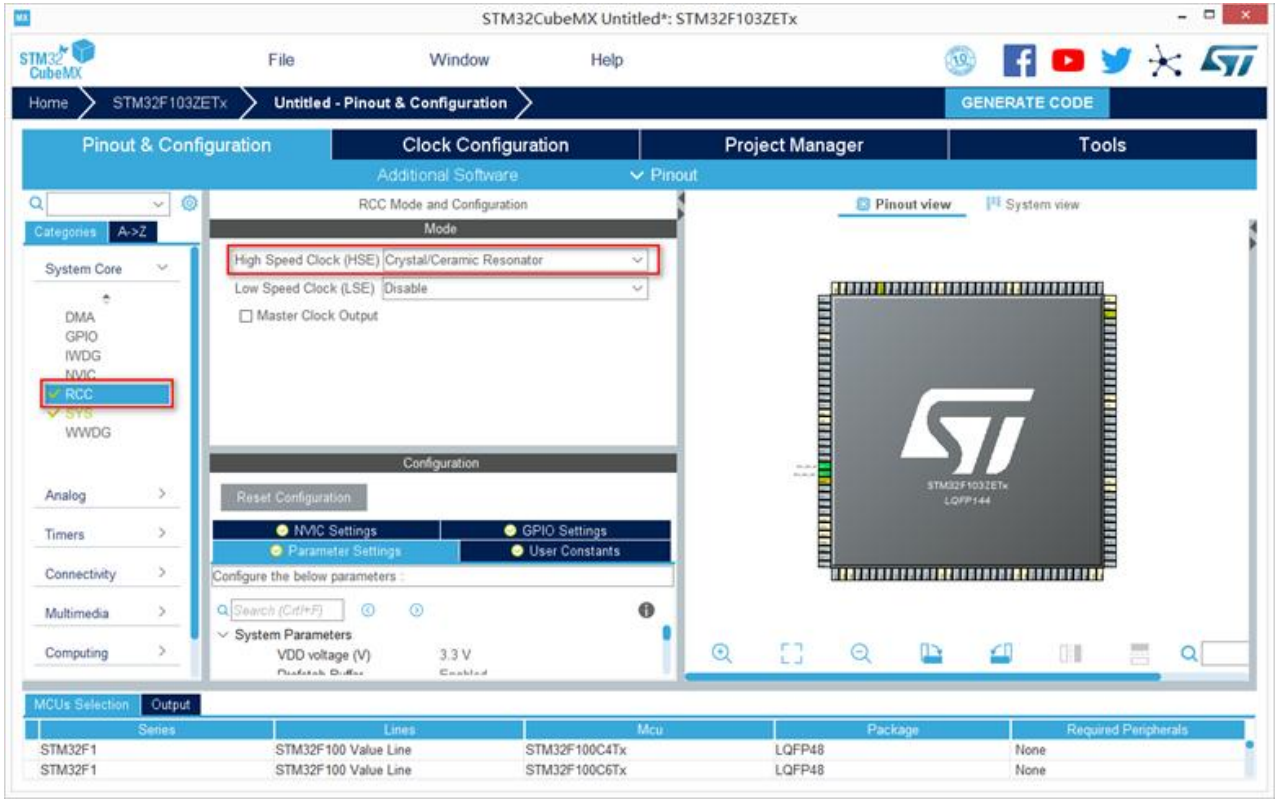


8.4.2 USART的HAL库应用实例

软件设计——STM32CubeMX功能参数配置

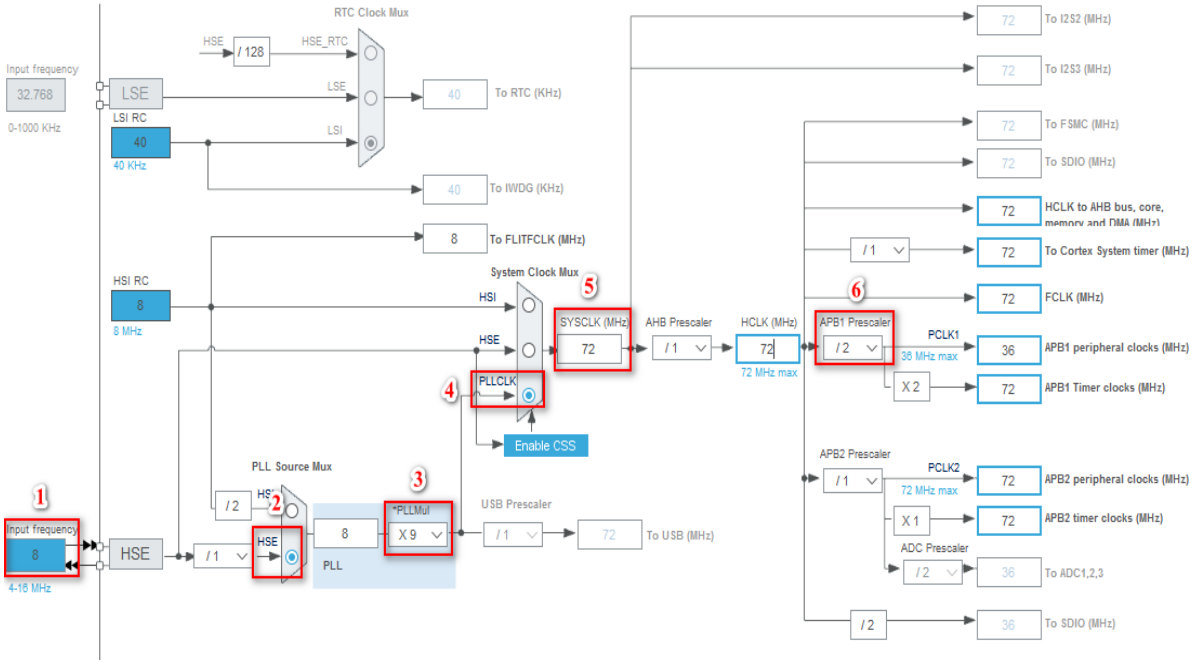
RCC配置

HSE选择“Crystal/Ceramic Resonator”（晶振/陶瓷谐振器



8.4.2 USART的HAL库应用实例

软件设计——STM32CubeMX功能参数配置



时钟配置

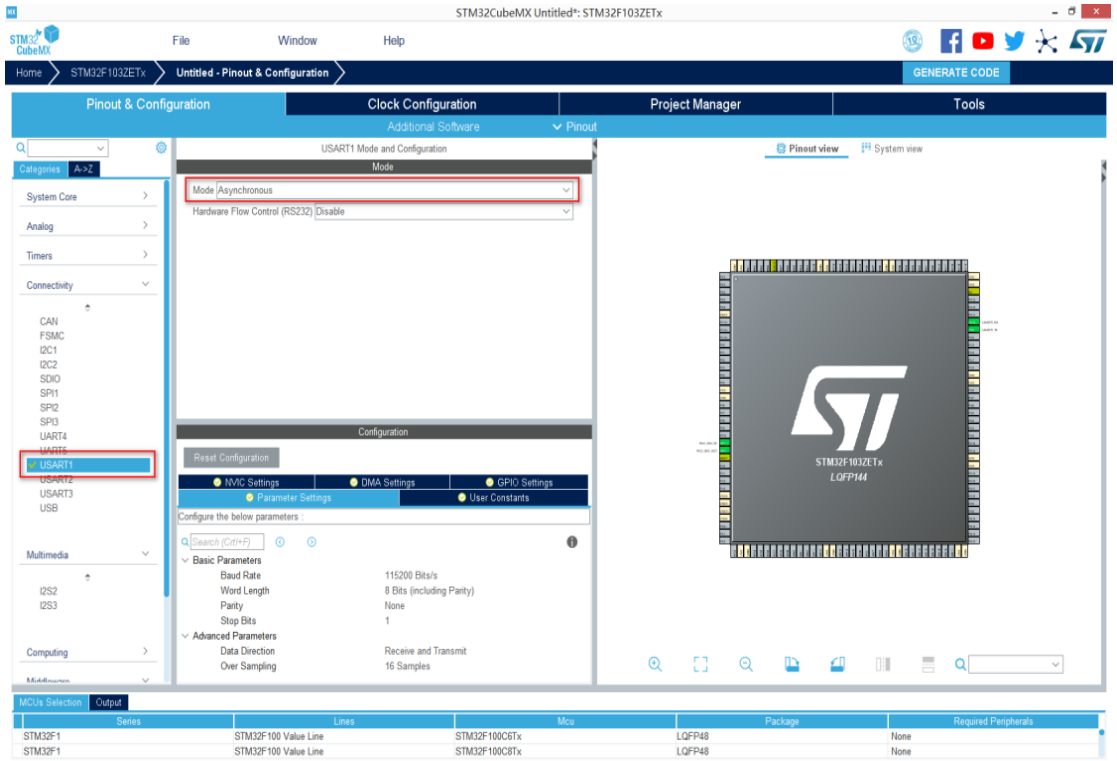
通过图形化方式直观地对系统时钟进行配置，系统时钟采用外部高速时钟，配置STM32F103系列芯片最大时钟为72MHz，配置APB2为72MHz，配置APB1为36MHz。

8.4.2 USART的HAL库应用实例

软件设计——STM32CubeMX功能参数配置

串口配置

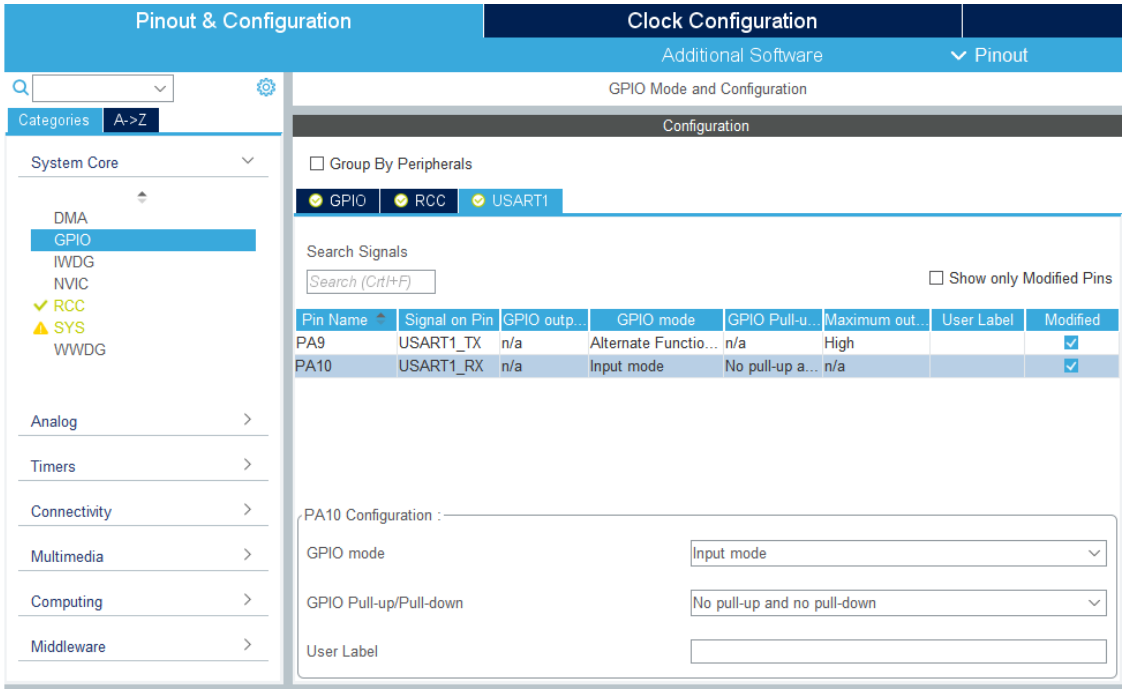
- 在“Connectivity”中选择“USART1”；
- 设置“Mode”为“Asynchronous”（异步传输模式）
- 设置USART1的相关参数，配置为默认值：115200、8、None和1。



8.4.2 USART的HAL库应用实例

软件设计——STM32CubeMX功能参数配置

串口引脚参数设置



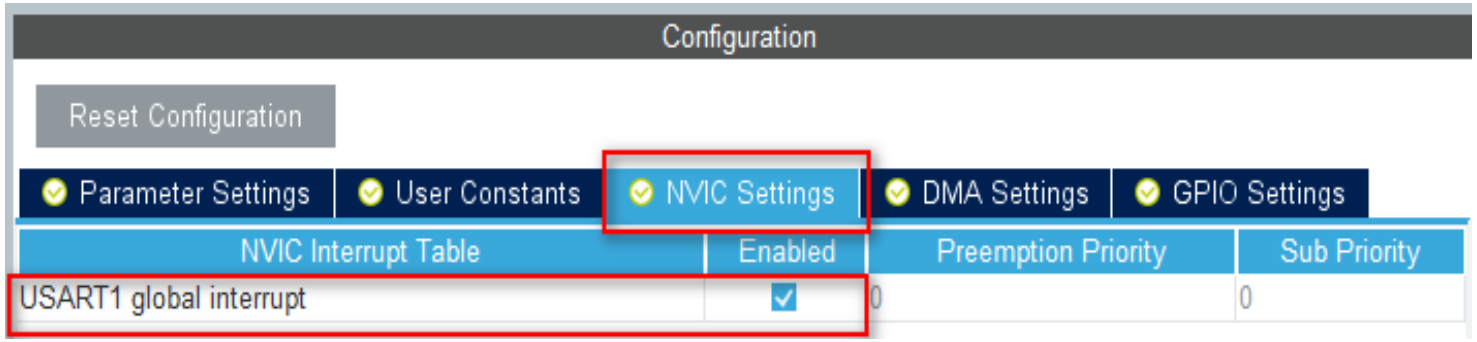
- USART1的发送引脚USART1_Tx默认连接在PA9；
- 接收引脚USART1_Rx默认连接在PA10；
- 在配置USART1时已经将PA9引脚设置为复用推挽输出，PA10设置为输入模式。

8.4.2 USART的HAL库应用实例

软件设计——STM32CubeMX功能参数配置

开启串口USART1的中断

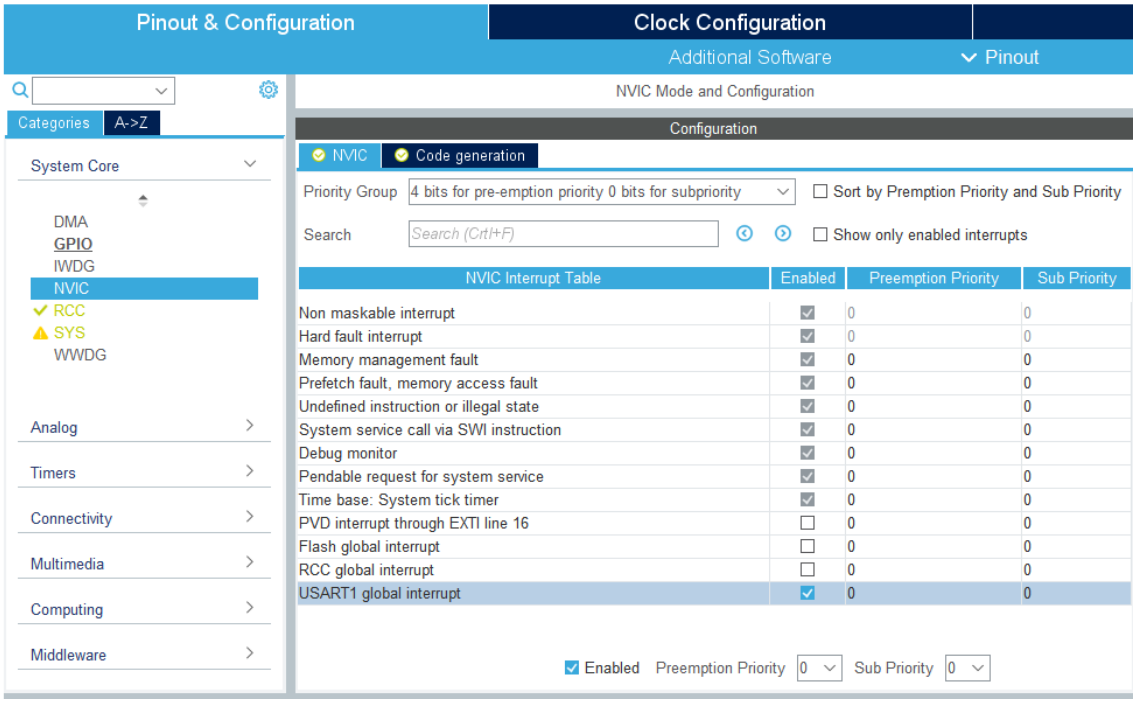
- 在“Configeration”配置页面的“NVIC Settings”中勾选“USART1 global interrupt”。



8.4.2 USART的HAL库应用实例

软件设计——STM32CubeMX功能参数配置

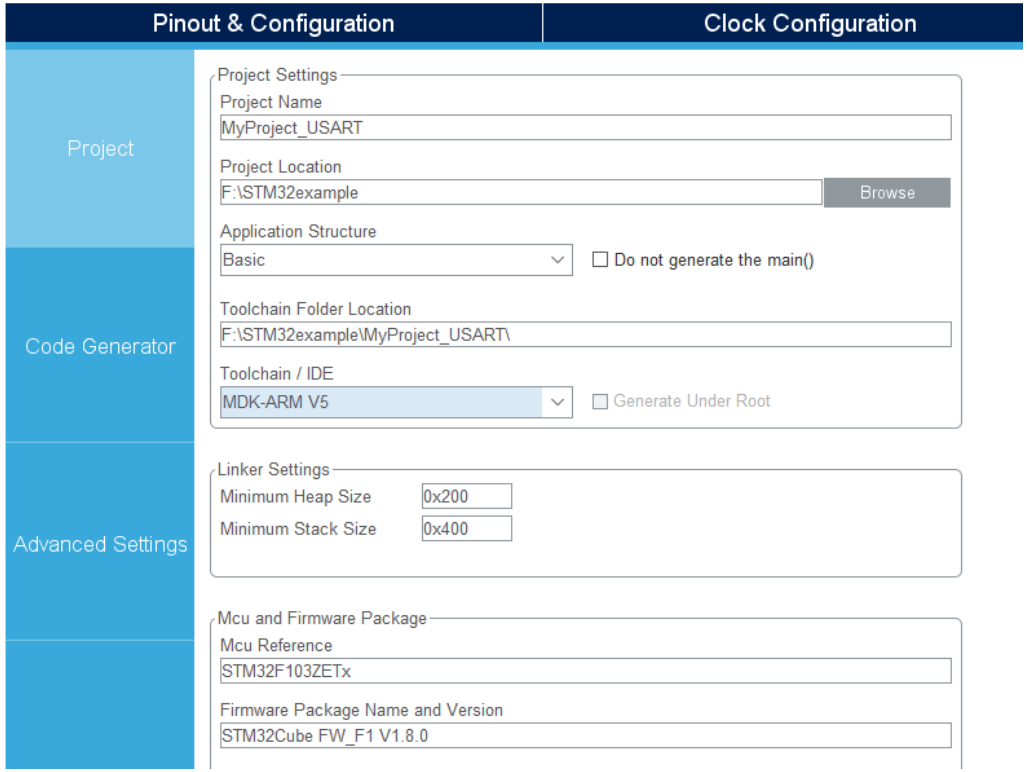
选择NVIC中断优先级分组



- 在“System Core”的“NVIC”中可选择NVIC中断优先级分组，默认为“4bits for pre-amption priority”，勾选“USART1 global interrupt”。

8.4.2 USART的HAL库应用实例

软件设计——生成工程代码

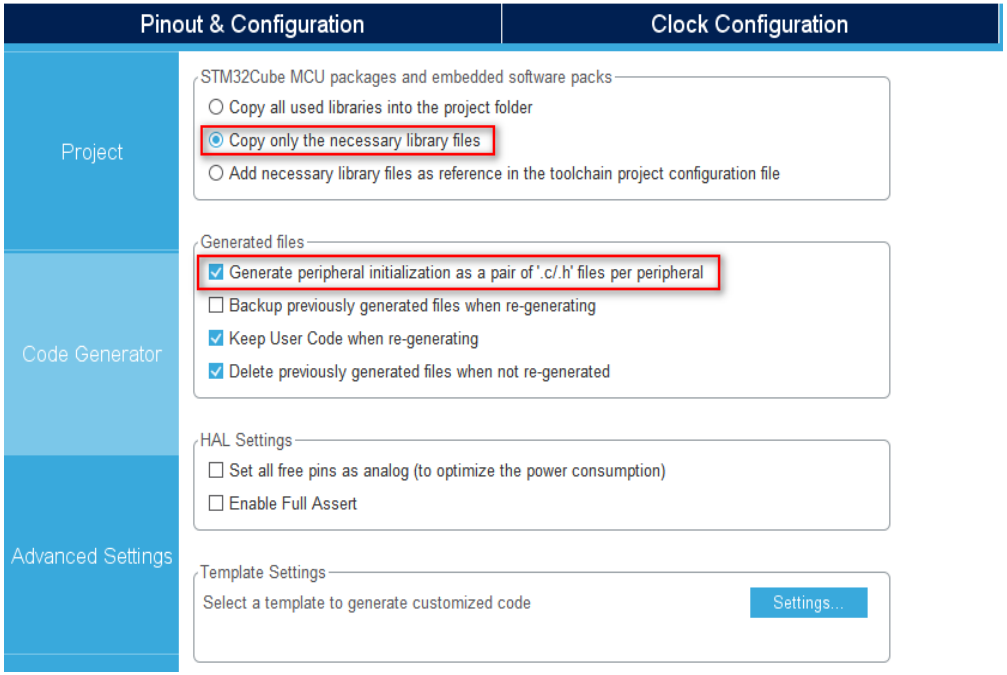


配置keil工程名称和存放位置

配置工程名称、工程保存位置、选择“MDK-ARM V5”编译器等

8.4.2 USART的HAL库应用实例

软件设计——生成工程代码



在“Code Generator”选项栏中找到“Generated files”框，勾选“Generate peripheral initialization as a pair of ’.c/.h’ files per IP”，将外设初始化的代码设置生成为独立的.c源文件和.h头文件。

8.4.2 USART的HAL库应用实例

软件设计——编写应用程序

在main.c文件中添加代码：

- 在main.c文件中添加代码。先定义两个数组RxBuffer和TxBuffer，用于接收数据和发送数据；
- 在main.c文件的/* USER CODE BEGIN PV */和/* USER CODE END PV */之间添加代码

```
/* USER CODE BEGIN PV */
uint8_t RxBuffer[10];
uint8_t TxBuffer[10];
/* USER CODE END PV */
```

8.4.2 USART的HAL库应用实例

软件设计——编写应用程序

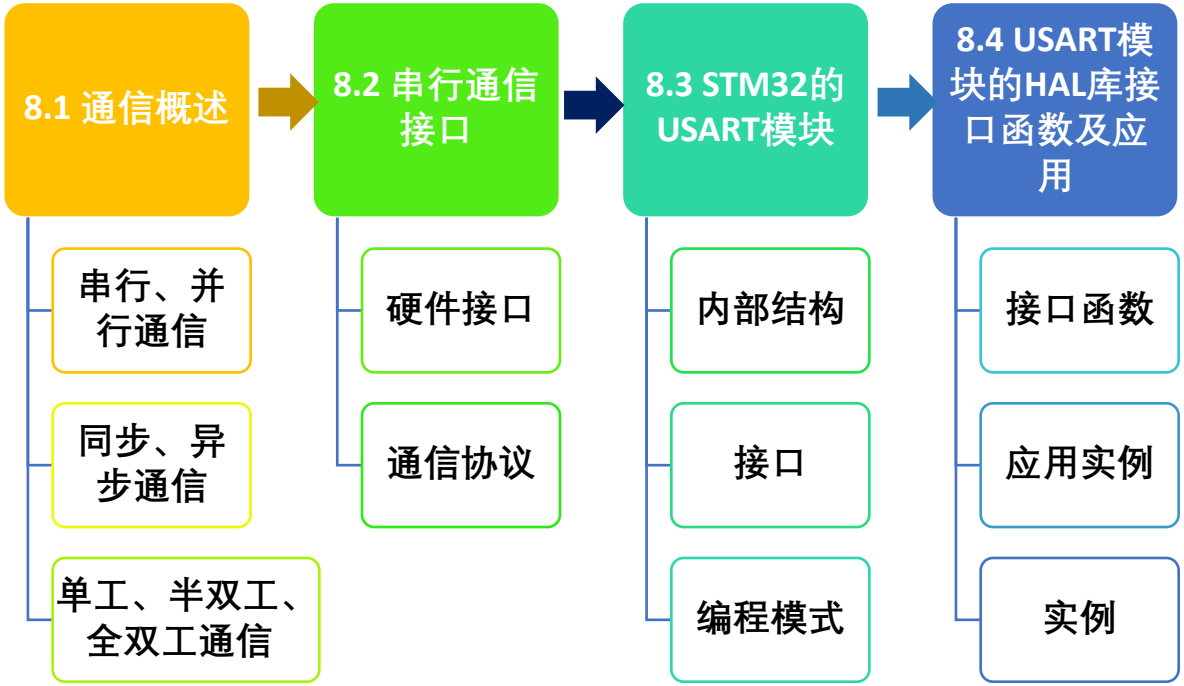
在int main(void)函数中的/* USER CODE BEGIN 2 */和/* USER CODE END 2 */之间添加代码：

```
/* USER CODE BEGIN 2 */
/*  使能USART1接收，调用中断处理函数 */
HAL_UART_Receive_IT(&huart1,(uint8_t *)&RxBuffer,1);
/* USER CODE END 2 */
```

中断处理函数进行相应功能程序的处理，在/* USER CODE BEGIN 4 */和/* USER CODE END 4 */之间添加代码：

```
/* USER CODE BEGIN 4 */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{   HAL_UART_Transmit(&huart1,(uint8_t *)&RxBuffer,1,0);
   HAL_UART_Receive_IT(&huart1,(uint8_t *)&RxBuffer,1);
}
/* USER CODE END 4 */
```

本章小结


8.1 通信概述
8.2 串行通信接口
8.3 STM32的USART模块
8.4 USART模块的HAL库接口函数及应用
串行、并行通信
同步、异步通信
单工、半双工、全双工通信
硬件接口
通信协议
内部结构
接口
编程模式
接口函数
应用实例
实例