目录

1 串行通信接口背景知识

2 STM32F4串口框图

3 STM32F4串口常用寄存器和库函数

4 串口配置方法(手把手写简单的通信实例）

5 串口通信实验讲解

✓ 串口通信原理与配置

■ 参考资料：

● 探索者STM32F4开发板：

《STM32F4开发指南-库函数版本》- 5.3小节 usart文件夹介绍

第9章 串口通信实验

□ STM32F4xx官方资料：

《STM32F4xx中文参考手册》-第26章 通用同步异步收发器

✓ 1.通信接口背景知识

◆ 处理器与外部设备通信的两种方式：

- **并行通信**
  - -传输原理：数据各个位同时传输。
  - -优点：速度快
  - -缺点：占用引脚资源多

- **串行通信**
  - -传输原理：数据按位顺序传输。
  - -优点：占用引脚资源少
  - -缺点：速度相对较慢

✓ 1.通信接口背景知识

◆ 串行通信：

按照数据传送方向，分为：

◆单工：

数据传输只支持数据在一个方向上传输

◆半双工：

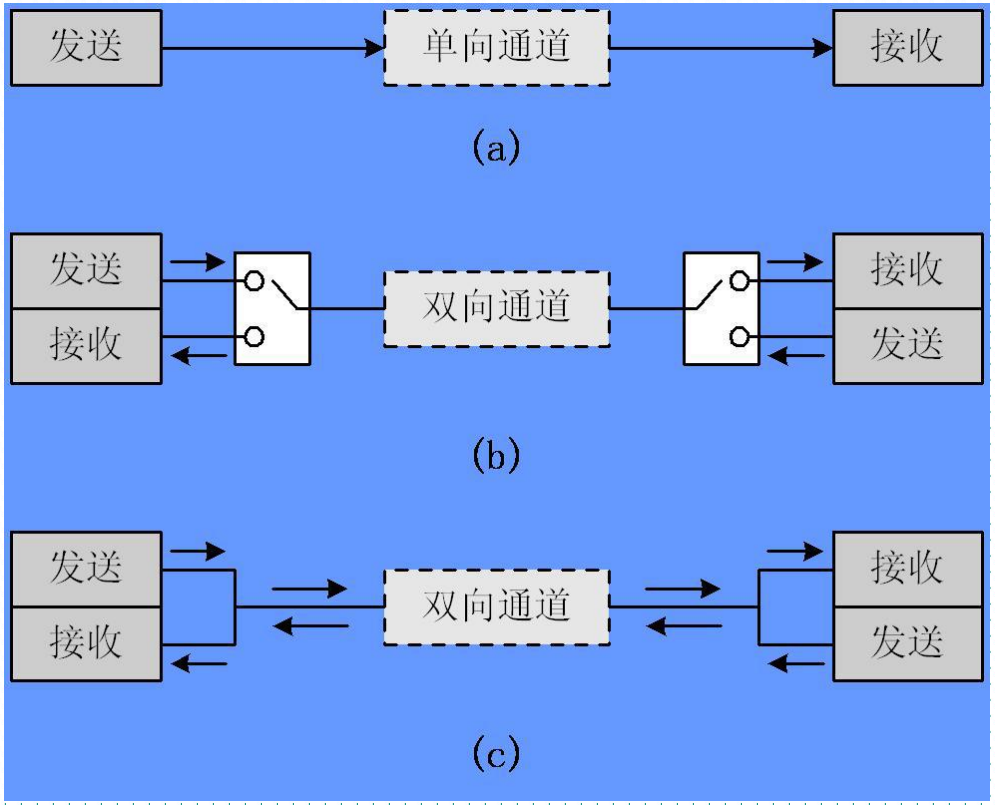
允许数据在两个方向上传输，但是，在某一时刻，只允许数据在一个方向上传输，它实际上是一种切换方向的单工通信；

◆全双工：

允许数据同时在两个方向上传输，因此，全双工通信是两个单工通信方式的结合，它要求发送设备和接收设备都有独立的接收和发送能力。

✓ 1.通信接口背景知识

◆ 串行通信三种传送方式：



✓ 1.通信接口背景知识

◆串行通信的通信方式

- **同步通信：** 带时钟同步信号传输。

  -SPI，IIC通信接口

- **异步通信：** 不带时钟同步信号。

  -UART(通用异步收发器),单总线

✓ 1.通信接口背景知识

◆常见的串行通信接口：

通信标准	引脚说明	通信方式	通信方向
UART (通用异步收发器)	TXD:发送端 RXD:接受端 GND:公共地	异步通信	全双工
单总线 （1-wire)	DQ:发送/接受端	异步通信	半双工
SPI	SCK:同步时钟 MISO:主机输入，从机输出 MOSI:主机输出，从机输入	同步通信	全双工
I2C	SCL:同步时钟 SDA:数据输入/输出端	同步通信	半双工

✓ 2.STM32串口通信基础

◆STM32的串口通信接口

- **UART:通用异步收发器**
- **USART:通用同步异步收发器**

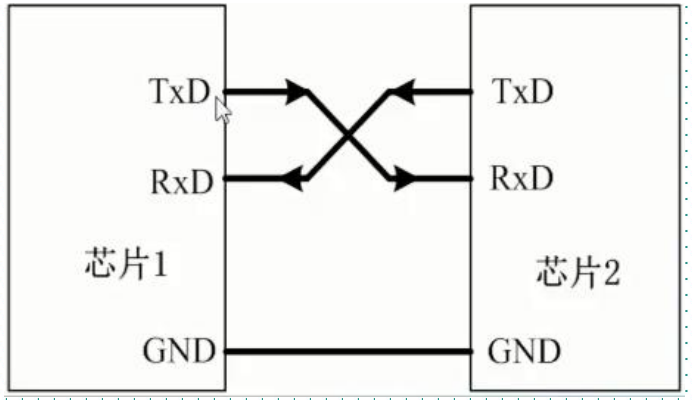
- **STM32F4XX目前最多支持8个UART,STM32F407一般是6个。具体可以对照选型手册和数据手册来看。**
- **STM32F103目前最多支持5个UART**

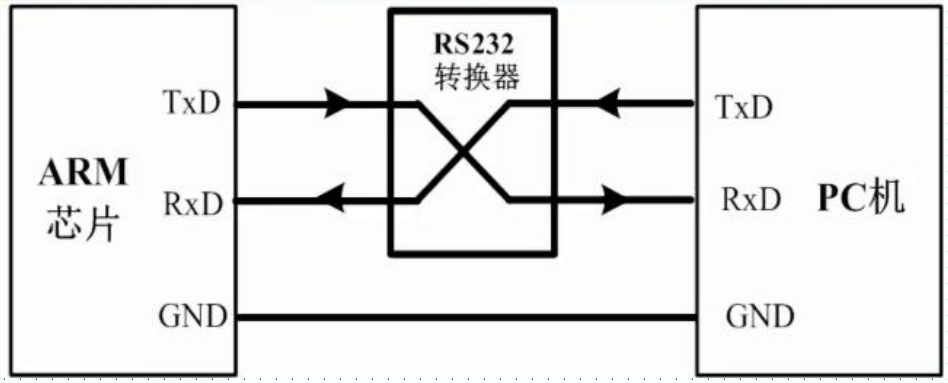
✓ 2.STM32串口通信基础

◆UART异步通信方式引脚连接方法：

-RXD:数据输入引脚。数据接受。

-TXD:数据发送引脚。数据发送。





■ 对于STM32F407，每个串口和引脚对应关系，可以查看数据手册引脚对应表。

✓ 2.STM32串口通信基础

◆UART异步通信方式引脚(STM32F407ZGT6):

串口号	RXD	TXD
1	PA10(PB7)	PA9（PB6)
2	PA3(PD6)	PA2(PD5)
3	PB11(PC11/PD9)	PB10(PC10/PD8)
4	PC11(PA1)	PC10(PA0)
5	PD2	PC12
6	PC7(PG9)	PC6(PG14)

STM32F4的芯片数据手册中芯片引脚功能中可以查看到。

✓ 2.STM32串口通信基础

◆UART异步通信方式特点：

- 全双工异步通信。
- 小数波特率发生器系统，提供精确的波特率。
- 可配置的16倍过采样或8倍过采样，因而为速度容差与时钟容差的灵活配置提供了可能。
- 可编程的数据字长度（8位或者9位）；
- 可配置的停止位（支持1或者2位停止位）；
- 可配置的使用DMA多缓冲器通信。
- 单独的发送器和接收器使能位。
- 检测标志：① 接受缓冲器 ②发送缓冲器空 ③传输结束标志
- 多个带标志的中断源。触发中断。
- 其他：校验控制，四个错误检测标志。

✓ 2.STM32串口通信基础

◆STM32串口通信过程

数据接收过程：

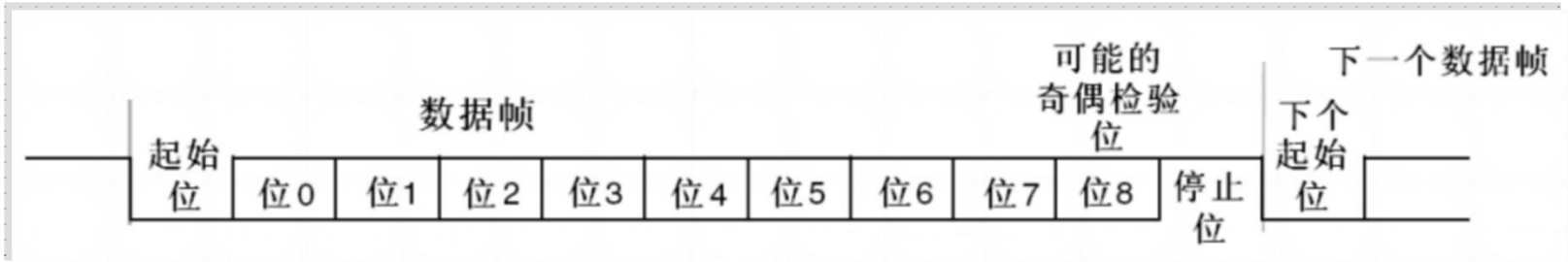
数据发送过程：

✓ 2.STM32串口通信基础

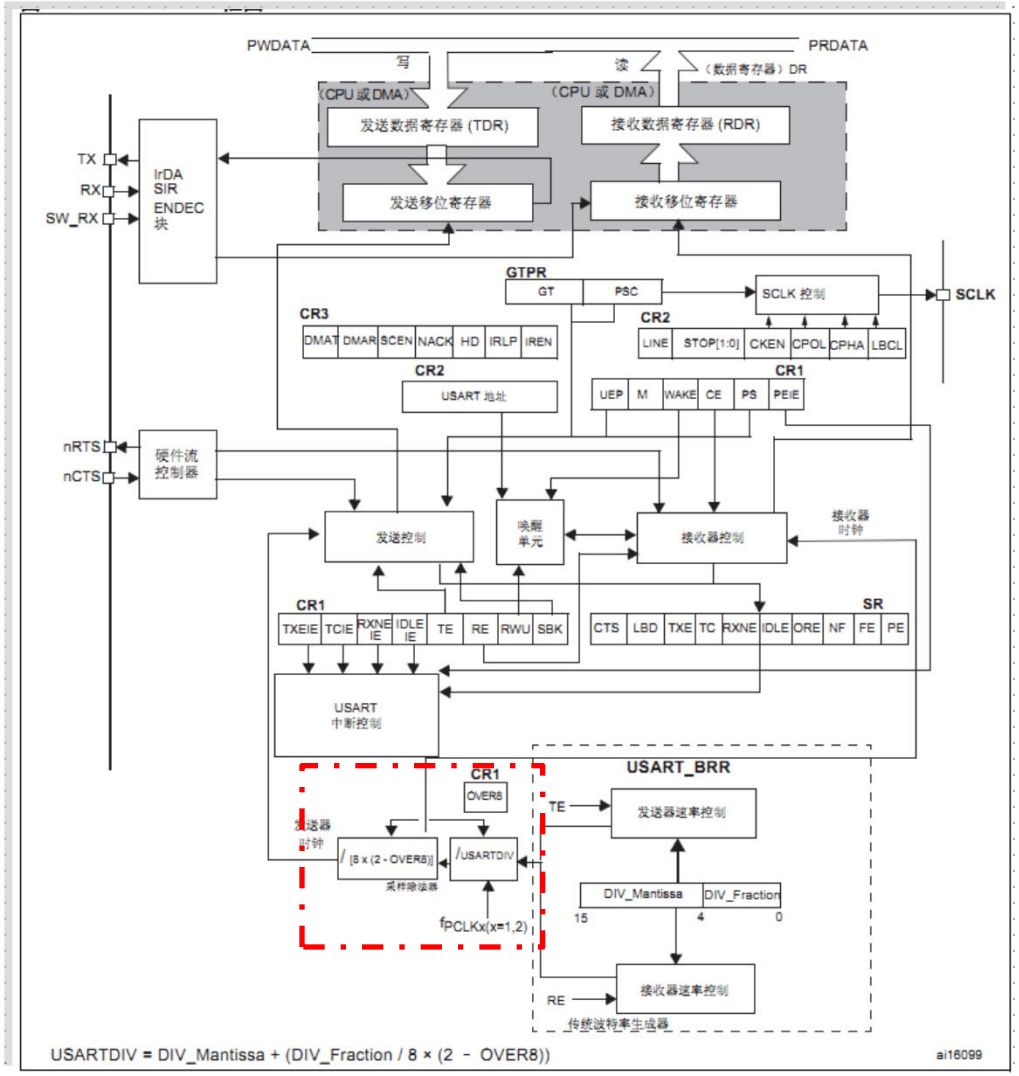
◆STM32串口异步通信需要定义的参数：

① 起始位
② 数据位（8位或者9位）
③ 奇偶校验位（第9位）
④ 停止位（1,15,2位）
⑤ 波特率设置

■ 范例：



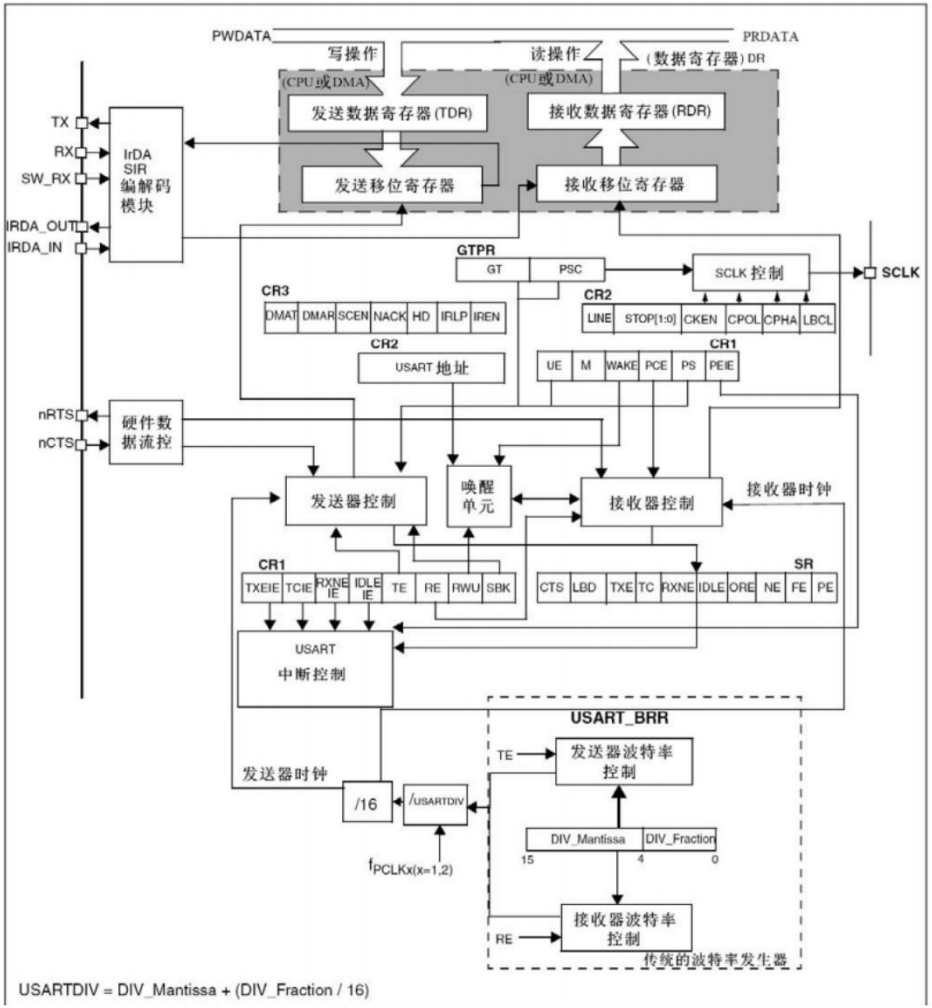
✓ 3.STM32串口框图



M4

✓ 2.STM32串口通信基础

图248 USART框图



M3

目录

1 STM32串口常用寄存器和库函数

2 串口配置一般步骤(手把手写串口实例)

✓ 串口通信基本原理

- 常用的串口相关寄存器
  - USART_SR状态寄存器
  - USART_DR数据寄存器
  - USART_BRR波特率寄存器

✓ 3.STM32串口常用寄存器和库函数

◆ 串口操作相关库函数（省略入口参数）：

void USART_Init(); //串口初始化：波特率，数据字长，奇偶校验，硬件流控以及收发使能
void USART_Cmd();//使能串口
void USART_ITConfig();//使能相关中断

*void USART_SendData();//发送数据到串口，**DR***
*uint16_t USART_ReceiveData();//接受数据，从**DR**读取接受到的数据*

FlagStatus USART_GetFlagStatus();//获取状态标志位
void USART_ClearFlag();//清除状态标志位
ITStatus USART_GetITStatus();//获取中断状态标志位
void USART_ClearITPendingBit();//清除中断状态标志位

✓ 3.STM32串口常用寄存器和库函数

状态寄存器(USART_SR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

位31:10	保留位，硬件强制为0
位9	**CTS**: CTS 标志 (CTS flag) 如果设置了CTSE位，当nCTS输入变化状态时，该位被硬件置高。由软件将其清零。如果USART_CR3中的CTSIE为'1'，则产生中断。 0：nCTS状态线上没有变化； 1：nCTS状态线上发生变化。 注：UART4和UART5上不存在这一位。
位8	**LBD**: LIN断开检测标志 (LIN break detection flag) 当探测到LIN断开时，该位由硬件置'1'，由软件清'0'(向该位写0)。如果USART_CR3中的LBDIE = 1，则产生中断。 0：没有检测到LIN断开；

FlagStatus USART_GetFlagStatus(USART_TypeDef USARTx, uint16_t USART_FLAG);*

✓ 3.STM32串口常用寄存器和库函数

数据寄存器(USART_DR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留							DR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:9	保留位，硬件强制为0
位8:0	**DR[8:0]**：数据值 (Data value) 包含了发送或接收的数据。由于它是由两个寄存器组成的，一个给发送用(TDR)，一个给接收用(RDR)，该寄存器兼具读和写的功能。TDR寄存器提供了内部总线和输出移位寄存器之间的并行接口(参见图248)。RDR寄存器提供了输入移位寄存器和内部总线之间的并行接口。 当使能校验位(USART_CR1中PCE位被置位)进行发送时，写到MSB的值(根据数据的长度不同，MSB是第7位或者第8位)会被后来的校验位该取代。 当使能校验位进行接收时，读到的MSB位是接收到的校验位。

void USART_SendData(USART_TypeDef USARTx, uint16_t Data);*
uint16_t USART_ReceiveData(USART_TypeDef USARTx);*

✓ 3.STM32串口常用寄存器和库函数

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV_Mantissa[11:0]												DIV_Fraction[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16 保留，必须保持复位值

位 15:4 **DIV_Mantissa[11:0]**：USARTDIV 的尾数
这 12 个位用于定义 USART 除数 (USARTDIV) 的尾数

位 3:0 **DIV_Fraction[3:0]**：USARTDIV 的小数
这 4 个位用于定义 USART 除数 (USARTDIV) 的小数。当 OVER8 = 1 时，不考虑 DIV_Fraction3 位，且必须将该位保持清零。

void USART_Init(USART_TypeDef USARTx, USART_InitTypeDef* USART_InitStruct);*

✓ 3.STM32串口常用寄存器和库函数

这里，我们简单介绍一下波特率的计算，STM32F4 的串口波特率计算公式（OVER8=0）如下：

$$\text{Tx / Rx 波特率} = \frac{f_{PCLKx}}{(16 * USARTDIV)}$$

上式中，f_{PCLKx} 是给串口的时钟（PCLK1 用于 USART2~5，PCLK2 用于 USART1 和 USART6）；USARTDIV 是一个无符号定点数。我们只要得到 USARTDIV 的值，就可以得到串口波特率寄存器 USART1->BRR 的值，反过来，我们得到 USART1->BRR 的值，也可以推导出 USARTDIV 的值。但我们更关心的是如何从 USARTDIV 的值得到 USART_BRR 的值，因为一般我们知道的是波特率，和 PCLKx 的时钟，要求的就是 USART_BRR 的值。

下面我们来介绍如何通过 USARTDIV 得到串口 USART_BRR 寄存器的值。假设我们的串口 1 要设置为 115200 的波特率，而 PCLK2 的时钟（即 APB2 总线时钟频率）为 84M。这样，我们根据上面的公式有：

USARTDIV=84000000/(115200*16)=45.572

那么得到：

DIV_Fraction=16*0.572=9=0X09;

DIV_Mantissa=45=0X2D;

✓ 4.串口配置一般步骤

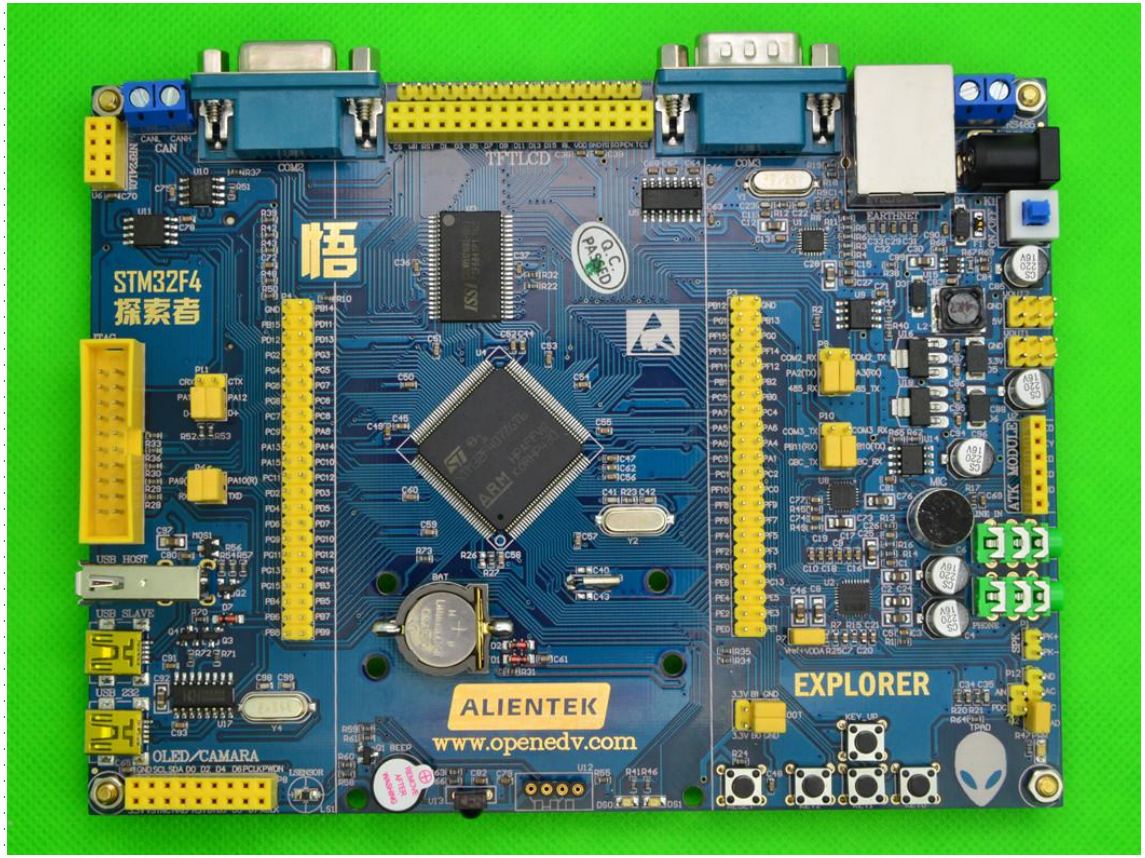
◆准备知识

需要先了解 **STM32F4**的端口复用映射相关知识，请参考前面端口复用映射视频。

✓ 4.串口配置一般步骤

◆硬件连接

PA9,PA10（串口1）连接到了USB串口电路。



✓ 4.串口配置一般步骤

◆串口配置的一般步骤

① 串口时钟使能：*RCC_APBxPeriphClockCmd();*
GPIO时钟使能：*RCC_AHB1PeriphClockCmd();*
② 引脚复用映射：
GPIO_PinAFConfig();
③GPIO端口模式设置:*GPIO_Init(); 模式设置为GPIO_Mode_AF*
④串口参数初始化：*USART_Init();*
⑤开启中断并且初始化NVIC（如果需要开启中断才需要这个步骤）
NVIC_Init();
USART_ITConfig();
⑥使能串口:*USART_Cmd();*
⑦编写中断处理函数：*USARTx_IRQHandler();*
⑧串口数据收发：
void USART_SendData();//发送数据到串口，DR
uint16_t USART_ReceiveData();//接受数据，从DR读取接受到的数据
⑨串口传输状态获取：
FlagStatus USART_GetFlagStatus();
void USART_ClearITPendingBit();