3.6.Serial communication

1.learning target

1.Learn the basic use of the serial port of the STM32 motherboard.

2.Understand the basics of serial communication.

USART introduction of STM32F1:

Serial communication (Serial Communication) refers to a communication method that transmits data bit by bit between peripherals and computers through data signal lines, ground lines, etc., and belongs to the serial communication method.

USART is the Universal Synchronous Asynchronous Transceiver, which can flexibly perform full-duplex data exchange with external devices and meet the requirements of external devices for the industry standard NRZ asynchronous serial data format. UART is the Universal Asynchronous Transceiver, which cuts out the synchronous communication function on the basis of USART. Synchronization and asynchrony mainly depend on whether the clock needs to be provided externally. This chapter mainly uses the USART, and the STM32F103C8T6 chip contains 3 USART peripherals. The USART supports synchronous one-way communication and half-duplex single-wire communication, and also supports LIN (Interconnect Domain Network), smart card protocol and IrDA (Infrared Data Association) SIR ENDEC specification, and modem operation (CTS/RTS). Furthermore, it supports multiprocessor communication and DMA function, which enables high-speed data communication using DMA. The USART provides a variety of baud rates through a fractional baud rate generator. The most widely used USART in STM32 is printf to output debugging information. When we need to know some variable data information in the program, we can use the printf output function to print this information to the serial port and display it on the assistant, so that we can debug the program Bring great convenience.

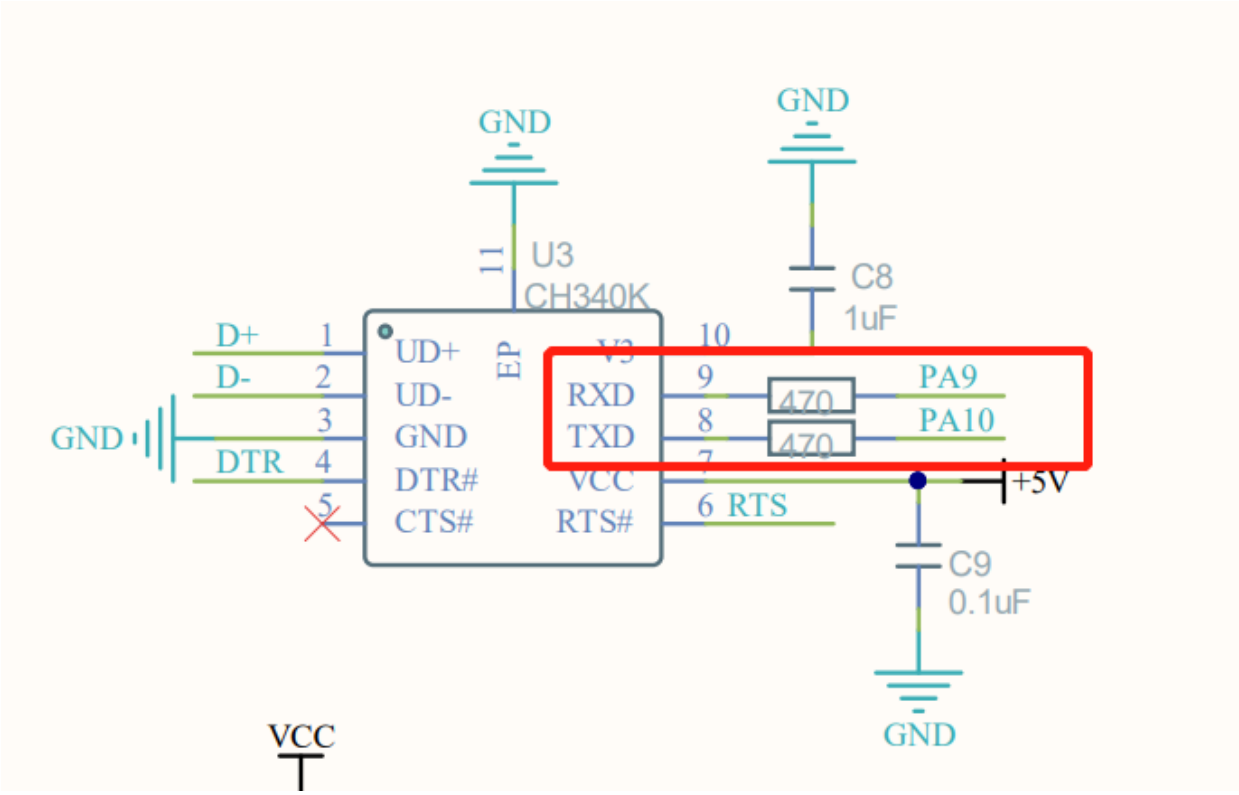
When in use, the corresponding pins of each serial port can be found in the STM32 data sheet.

Table 5. Medium-density STM32F103xx pin definitions (continued)

Pins							Pin name	Type(1)	I / O Level(2)	Main function(3) (after reset)	Alternate functions(4)	
LFBGA100	UFBG100	LQFP48/UFQFPN48	TFBGA64	LQFP64	LQFP100	VFQFPN36					Default	Remap
H9	J12	-	-	-	57	-	PD10	I/O	FT	PD10	-	USART3_CK
G9	J11	-	-	-	58	-	PD11	I/O	FT	PD11	-	USART3_CTS
K10	J10	-	-	-	59	-	PD12	I/O	FT	PD12	-	TIM4_CH1 / USART3_RTS
J10	H12	-	-	-	60	-	PD13	I/O	FT	PD13	-	TIM4_CH2
H10	H11	-	-	-	61	-	PD14	I/O	FT	PD14	-	TIM4_CH3
G10	H10	-	-	-	62	-	PD15	I/O	FT	PD15	-	TIM4_CH4
F10	E12	-	F6	37	63	-	PC6	I/O	FT	PC6	-	TIM3_CH1
E10	E11		E7	38	64	-	PC7	I/O	FT	PC7	-	TIM3_CH2
F9	E10		E8	39	65	-	PC8	I/O	FT	PC8	-	TIM3_CH3
E9	D12	-	D8	40	66	-	PC9	I/O	FT	PC9	-	TIM3_CH4
D9	D11	29	D7	41	67	20	PA8	I/O	FT	PA8	USART1_CK/ TIM1_CH1(9)/ MCO	-
C9	D10	30	C7	42	68	21	PA9	I/O	FT	PA9	USART1_TX(9)/ TIM1_CH2(9)	-
D10	C12	31	C6	43	69	22	PA10	I/O	FT	PA10	USART1_RX(9)/ TIM1_CH3(9)	-

2.Hardware Wiring Introduction

Since the core board comes with a ch340 serial port circuit, the serial port communication can be realized directly through the usb cable, without the need for an external usb to ttl module.



Partial circuit diagram of CH340

3.program analysis

Serial port initialization code:

```
void USART1_Init(u32 bound)
{

  GPIO_InitTypeDef GPIO_InitStructure;
  USART_InitTypeDef USART_InitStructure;
  NVIC_InitTypeDef NVIC_InitStructure;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);



  GPIO_InitStructure.GPIO_Pin=GPIO_Pin_9;//TX
  GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode=GPIO_Mode_AF_PP;
  GPIO_Init(GPIOA, &GPIO_InitStructure);  /*
  GPIO_InitStructure.GPIO_Pin=GPIO_Pin_10;//RX
  GPIO_InitStructure.GPIO_Mode=GPIO_Mode_IN_FLOATING;
  GPIO_Init(GPIOA, &GPIO_InitStructure);



  USART_InitStructure.USART_BaudRate = bound;
  USART_InitStructure.USART_WordLength = USART_WordLength_8b;
  USART_InitStructure.USART_StopBits = USART_StopBits_1;//
  USART_InitStructure.USART_Parity = USART_Parity_No;//
  USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
  USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
  USART_Init(USART1, &USART_InitStructure);

  USART_Cmd(USART1, ENABLE);

  USART_ClearFlag(USART1, USART_FLAG_TC);

  USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);


  NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority=3;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority =3;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
}
```

In the USART1_Init() function, first enable the USART1 serial port and port clock, and initialize the GPIO as an alternate function. Secondly, configure the serial port structure USART_InitTypeDef, enable the serial port and enable the receiving interrupt. In order to prevent the influence of the serial port sending status flag, we clear the serial port status flag (TC), and finally configure the corresponding NVIC and enable the corresponding interrupt channel. We will The preemption priority of USART1 is set to 3, and the response priority is set to 3.

serial interrupt code:

```
void USART1_IRQHandler(void)
{
  u8 r;
  if(USART_GetITStatus(USART1, USART_IT_RXNE) != RESET)
  {
    r =USART_ReceiveData(USART1);//(USART1->DR);
    USART_SendData(USART1,r);
    while(USART_GetFlagStatus(USART1,USART_FLAG_TC) != SET);
  }
  USART_ClearFlag(USART1,USART_FLAG_TC);
}
```

In order to confirm whether USART1 has a receiving interrupt, the function USART_GetITStatus, which reads the serial port interrupt status flag bit, is called. If a receiving interrupt event is indeed generated, then the statement in the if will be executed, and the data received by the serial port will be saved in the variable r, and then there will be Send out through the serial port, read the serial port status flag through the USART_GetFlagStatus function, if the data transmission is completed, exit the while loop statement, and clear the sending completion status flag USART_FLAG_TC.

main function code:

```
#include "system.h"
#include "SysTick.h"
#include "led.h"
#include "usart.h"

int main()
{
  u8 i=0;
  SysTick_Init(72);
  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
  LED_Init();
  USART1_Init(9600);

  while(1)
  {
    i++;
    if(i%20==0)
    {
      led1=!led1;
    }
    delay_ms(10);
  }
}
```

The function implemented by the main function is very simple. First, it calls the previously written hardware initialization function, including SysTick system clock, interrupt grouping, LED initialization, etc. Then call the USART1 initialization function we wrote earlier, here we set the serial communication baud rate to 9600. Finally, enter the while loop statement, and let the D1 indicator

flash at intervals of 200ms. If a receiving interrupt event occurs, it will enter the interrupt execution, and return to the main function to continue running after execution

4.Experimental phenomena

After the program download is complete, the led light will flicker, and after configuring the serial port assistant as shown in the figure below, you can view the returned sending content in the serial port assistant receiving window

Note: The baud rate must be 9600, otherwise the sent content cannot be received.