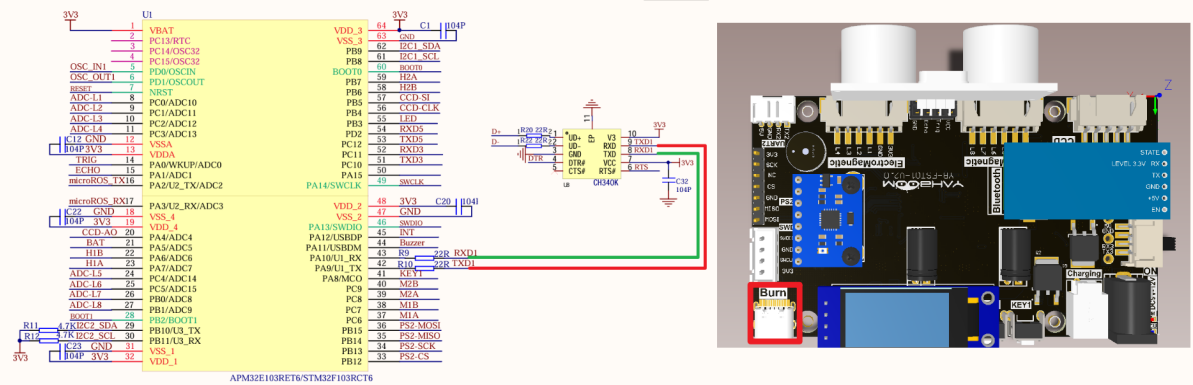
DMA：USART

The tutorial demonstrates serial port (USART1) communication via DMA.

```
The tutorial only introduces the standard library project code
```

Hardware connection



Peripherals	Development board	Description
USART1_TX	PA9	USART1 sends data
USART1_RX	PA10	USART1 receives data

Control principle

```
Serial port related knowledge will not be introduced, mainly DMA knowledge.
```

STM32F103RCT6 has two DMA controllers, DMA1 has 7 channels and DMA2 has 5 channels;

It is used for high-speed data transmission between peripherals and memory and between memory and memory.

DMA features

DMA initialization and startup are completed by the CPU, and the transfer process is performed by the DMA controller without CPU involvement, thus saving CPU resources for other operations.

DMA1 requests for each channel

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
ADC1	ADC1	-	-	-	-	-	-
SPI/I^2S	-	SPI1_RX	SPI1_TX	SPI2/I2S2_RX	SPI2/I2S2_TX	-	-
USART	-	USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I^2C	-	-	-	I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1	-	TIM1_CH1	-	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	-
TIM2	TIM2_CH3	TIM2_UP	-	-	TIM2_CH1	-	TIM2_CH2 TIM2_CH4
TIM3	-	TIM3_CH3	TIM3_CH4 TIM3_UP	-	-	TIM3_CH1 TIM3_TRIG	-
TIM4	TIM4_CH1	-	-	TIM4_CH2	TIM4_CH3	-	TIM4_UP

DMA2 requests for each channel

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
ADC3(1)	-	-	-	-	ADC3
SPI/I2S3	SPI/I2S3_RX	SPI/I2S3_TX	-	-	-
UART4	-	-	UART4_RX	-	UART4_TX
SDIO(1)	-	-	-	SDIO	-
TIM5	TIM5_CH4 TIM5_TRIG	TIM5_CH3 TIM5_UP	-	TIM5_CH2	TIM5_CH1
TIM6/ DAC_Channel1	-	-	TIM6_UP/ DAC_Channel1	-	-
TIM7	-	-	-	TIM7_UP/ DAC_Channel2	-
TIM8	TIM8_CH3 TIM8_UP	TIM8_CH4 TIM8_TRIG TIM8_COM	TIM8_CH1	-	TIM8_CH2

This tutorial uses DMA1 channel 4.

Pin definition

Main control chip	Pin	Main function (after reset)	Default multiplexing function	Redefine function
STM32F103RCT6	PA9	PA9	USART1_TX/TIM1_CH2	
STM32F103RCT6	PA10	PA10	USART1_RX/TIM1_CH3	

Software code

Since the default function of the PA9/10 pin is a normal IO pin function, we need to use the multiplexing function.

```
Product supporting data source code path: Attachment → Source code summary →
1.Base_Course → 15.USART+DMA
```

Control function

The tutorial only briefly introduces the code, and you can open the project source code to read it in detail.

> **uart_init**

```
void uart_init(u32 bound)
{
    //GPIO port settings
    GPIO_InitTypeDef GPIO_InitStructure;
    USART_InitTypeDef USART_InitStructure;
    NVIC_InitTypeDef NVIC_InitStructure;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1|RCC_APB2Periph_GPIOA, ENABLE);
//Enable USART1, GPIOA clock

    //USART1_TX GPIOA.9
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9; //PA.9
    GPIO_InitStructure.GPIO_Speed ••= GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP; //Multiplex push-pull output
    GPIO_Init(GPIOA, &GPIO_InitStructure);//Initialize GPIOA.9

    //Initialize USART1_RX GPIOA.10
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;//PA10
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;//Floating input
    GPIO_Init(GPIOA, &GPIO_InitStructure);//Initialize GPIOA.10
    //USART initialization settings

    USART_InitStructure.USART_BaudRate = bound;//Serial port baud rate
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;//Word length is
8-bit data format
    USART_InitStructure.USART_StopBits = USART_StopBits_1;//One stop bit
    USART_InitStructure.USART_Parity = USART_Parity_No; //No parity bit
    USART_InitStructure.USART_HardwareFlowControl =
USART_HardwareFlowControl_None; //No hardware data flow control
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx; //Transmit
and receive mode

    USART_Init(USART1, &USART_InitStructure); //Initialize serial port 1
    USART_ITConfig(USART1, USART_IT_RXNE, ENABLE); //Serial port receive
interrupt
    USART_Cmd(USART1, ENABLE); //Enable serial port 1

    NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
    }
```

> **USARTx_DMA_Config**

```
void USARTx_DMA_Config(void)
{
```

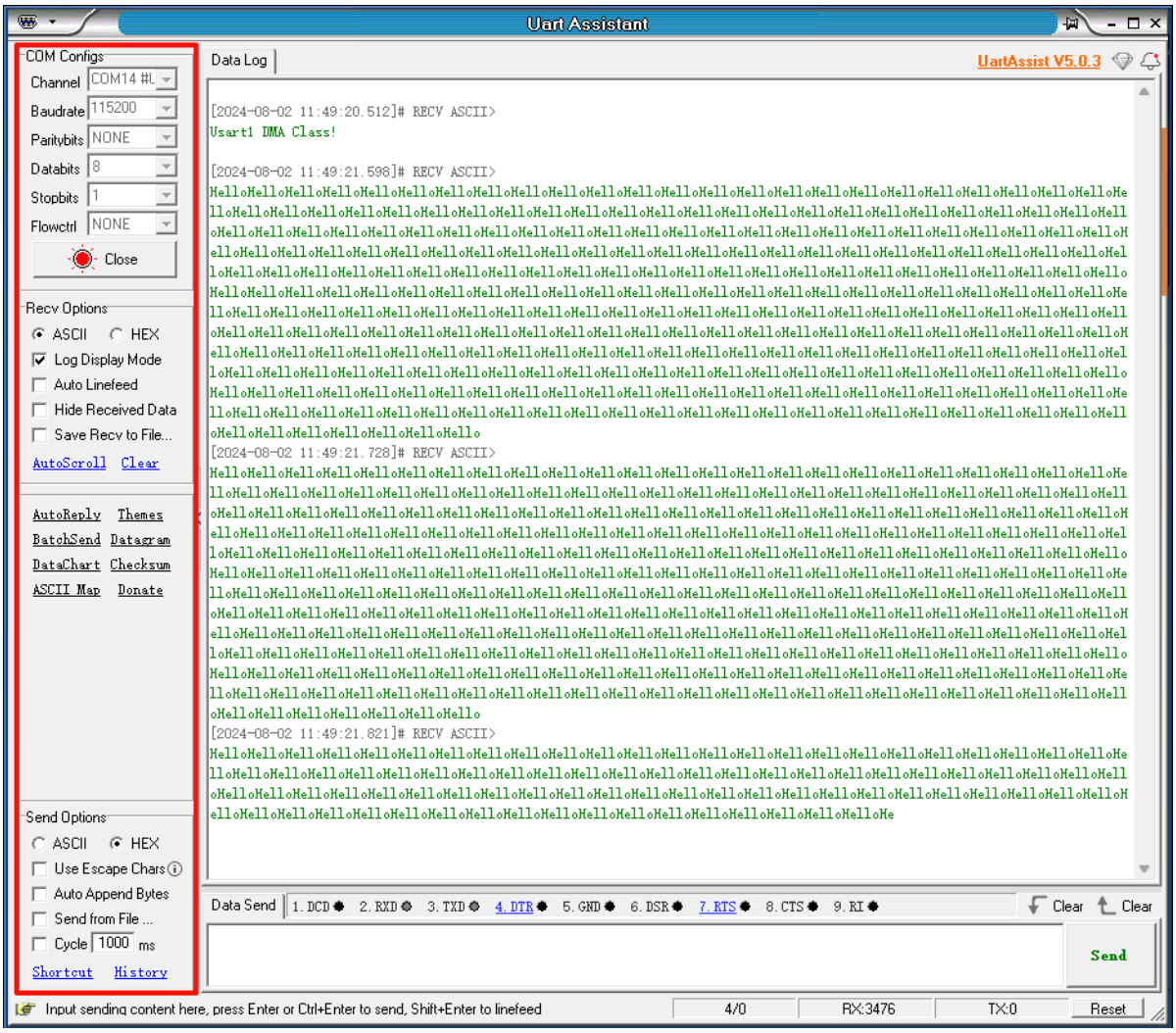
```
    DMA_InitTypeDef DMA_InitStructure;

    // Enable serial port DMA clock
    RCC_AHBPeriphClockCmd(USART_TX_DMA_CLK, ENABLE);
    // Set DMA source address: serial port data register address*/
    DMA_InitStructure.DMA_PeripheralBaseAddr = USART_DR_ADDRESS;
    // Memory address (pointer to the variable to be transferred)
    DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)SendBuff;
    // Direction: from memory to peripherals
    DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralDST;

    // Transfer size
    DMA_InitStructure.DMA_BufferSize = SENDBUFF_SIZE;
    // Peripheral address does not increase
    DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
    // Memory address increment
    DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
    // Peripheral data unit
    DMA_InitStructure.DMA_PeripheralDataSize =
    DMA_PeripheralDataSize_Byte;
    // Memory data unit
    DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;

    // DMA mode, one-time or circular mode
    DMA_InitStructure.DMA_Mode = DMA_Mode_Normal;
    //DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
    // Priority: Medium
    DMA_InitStructure.DMA_Priority = DMA_Priority_Medium;
    // Disable memory-to-memory transfer
    DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
    // Configure DMA channel
    DMA_Init(USART_TX_DMA_CHANNEL, &DMA_InitStructure);
    // Clear the conversion flag of DMA1 channel 4
    DMA_ClearFlag(USART_TX_DMA_FLAG_TC );
    // Enable DMA
    DMA_Cmd (USART_TX_DMA_CHANNEL,ENABLE);
}
```

Experimental phenomenon

The USART_DMA.hex file generated by the project compilation is located in the OBJ folder of the [USART+DMA] project. Find the USART_DMA.hex file corresponding to the project and use the FlyMcu software to download the program to the development board.

After the program is successfully downloaded: Use the serial port debugging assistant to view the information, and you can see that the serial port prints Usart1 DMA Class! and Hello (Hello prints for about 300ms).

```
When using the serial port debugging assistant, you need to pay attention to the
serial port settings. If the settings are wrong, the phenomenon may be
inconsistent
```


Uart Assistant
COM Configs
Channel COM14 #L
Baudrate 115200
Paritybits NONE
Databits 8
Stopbits 1
Flowctrl NONE
Close
Recv Options
ASCII HEX
Log Display Mode
Auto Linefeed
Hide Received Data
Save Recv to File...
AutoScroll Clear
AutoReply Themes
BatchSend Datagram
DataChart Checksum
ASCII Map Donate
Send Options
ASCII HEX
Use Escape Chars
Auto Append Bytes
Send from File ...
Cycle 1000 ms
Shortcut History
Data Log
UartAssist V5.0.3
[2024-08-02 11:49:20.512]# RECV ASCII>
Usart1 DMA Class!
[2024-08-02 11:49:21.598]# RECV ASCII>
[2024-08-02 11:49:21.728]# RECV ASCII>
[2024-08-02 11:49:21.821]# RECV ASCII>
Data Send
1. DCD 2. RXD 3. TXD 4. DTR 5. GND 6. DSR 7. RTS 8. CTS 9. RI
Clear
Clear
Send
Input sending content here, press Enter or Ctrl+Enter to send, Shift+Enter to linefeed
4/0
RX:3476
TX:0
Reset