DMA：USART

This tutorial demonstrates **Serial (USART1)** communication via **DMA**

1、software-hardware

- **STM32F103CubeIDE**
- **STM32 robot expansion board**

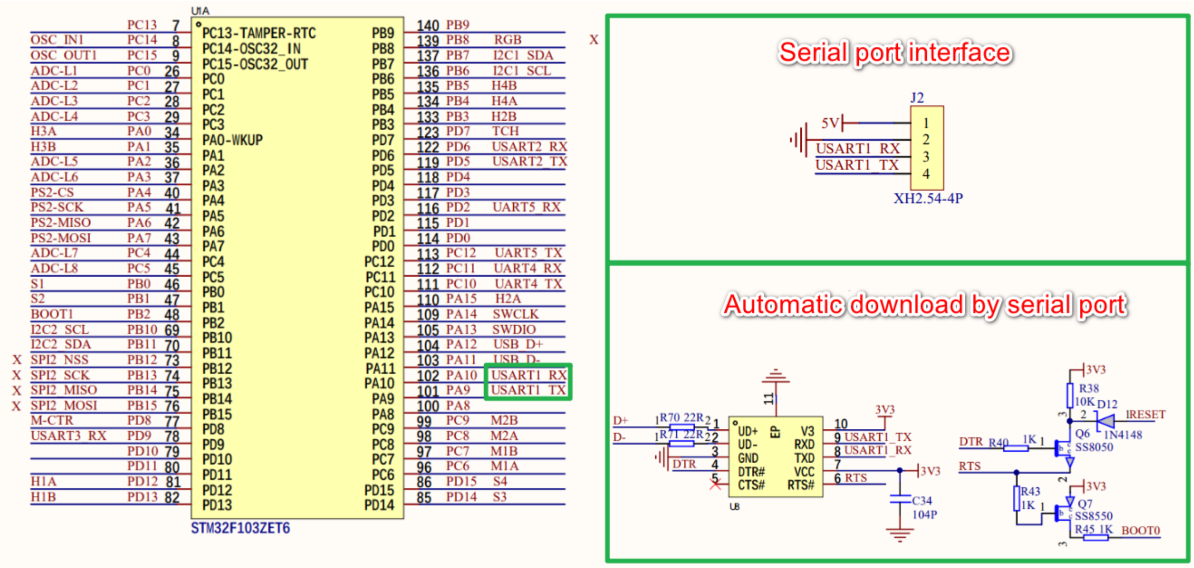
  USART1, DMA: chip internal peripheral
- **Type-C cable or ST-Link**

  Download or simulate the program of the development board

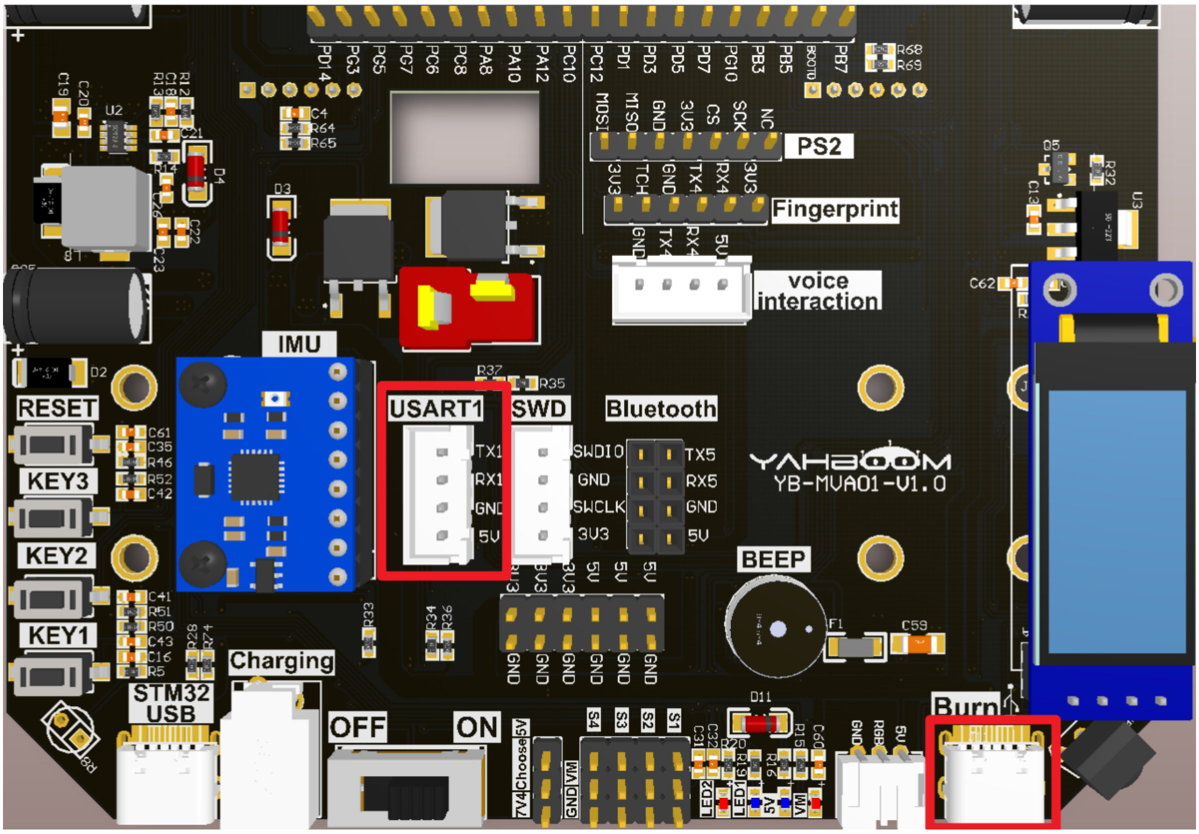
2、Brief principle

2.1、Hardware schematic diagram

The schematic only shows the serial interface used in the tutorial (USART1)



2.2、Physical connection diagram



3.3、Principle of control

- **USART1**

The knowledge related to serial port will not be introduced, you can see before [Chapter 3:3.4 Serial communication]

- **DMA（Direct Memory Access)**

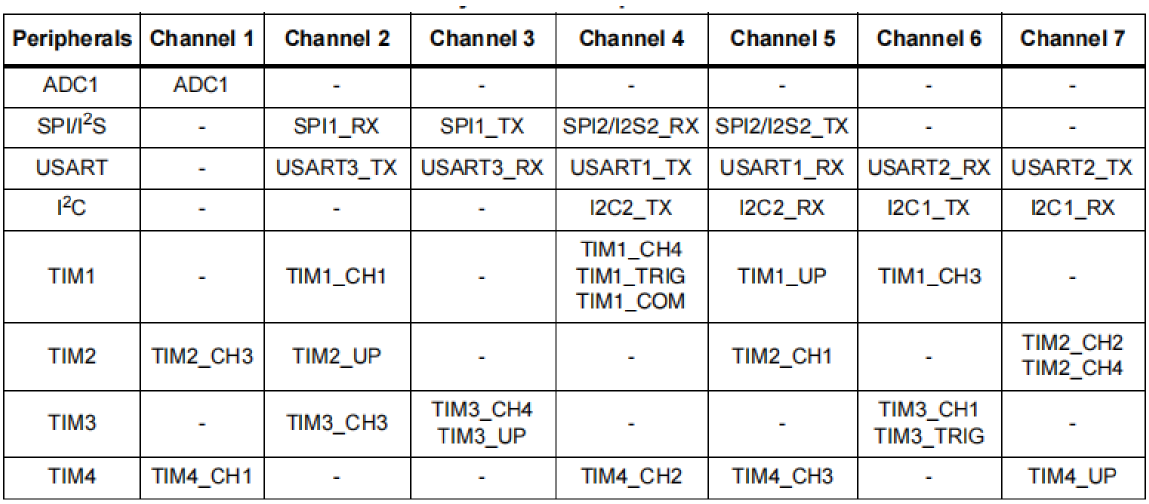
STM32F103ZET6 has a total of two DMA controllers, DMA1 has 7 channels, DMA2 has 5 channels;

It is used for high-speed data transfer between peripheral equipment and memory and between memory and memory.

DMA features

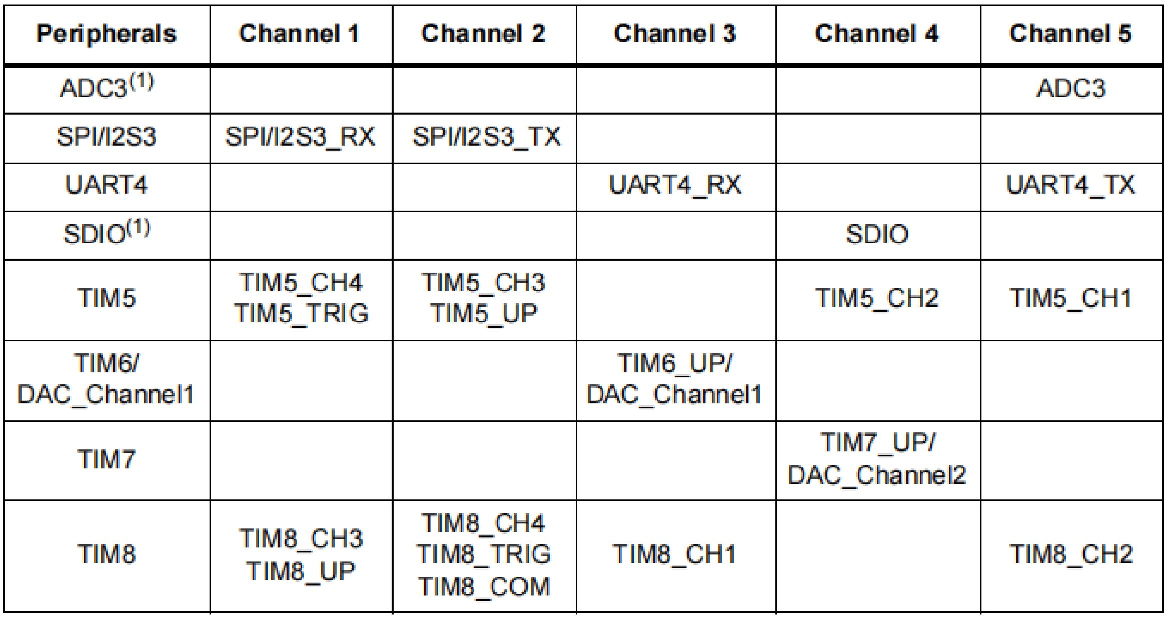
The initialization and start of DMA are completed by the CPU, and the transfer process is executed by the DMA controller without the participation of the CPU, so that the CPU resources are saved to do other operations

DMA1 requests per channel



Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
ADC1	ADC1	-	-	-	-	-	-
SPI/I²S	-	SPI1_RX	SPI1_TX	SPI2/I2S2_RX	SPI2/I2S2_TX	-	-
USART	-	USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I²C	-	-	-	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 per 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

DMA1 channels 4 and 5 are used in this tutorial

3、Engineering configuration

Project Configuration: Prompts for configuration options in the STM32CubeIDE project configuration process

3.1、Notes

Omitted project configuration: **New project, chip selection, project configuration, SYS for pin configuration, RCC configuration, clock configuration, and project configuration** content

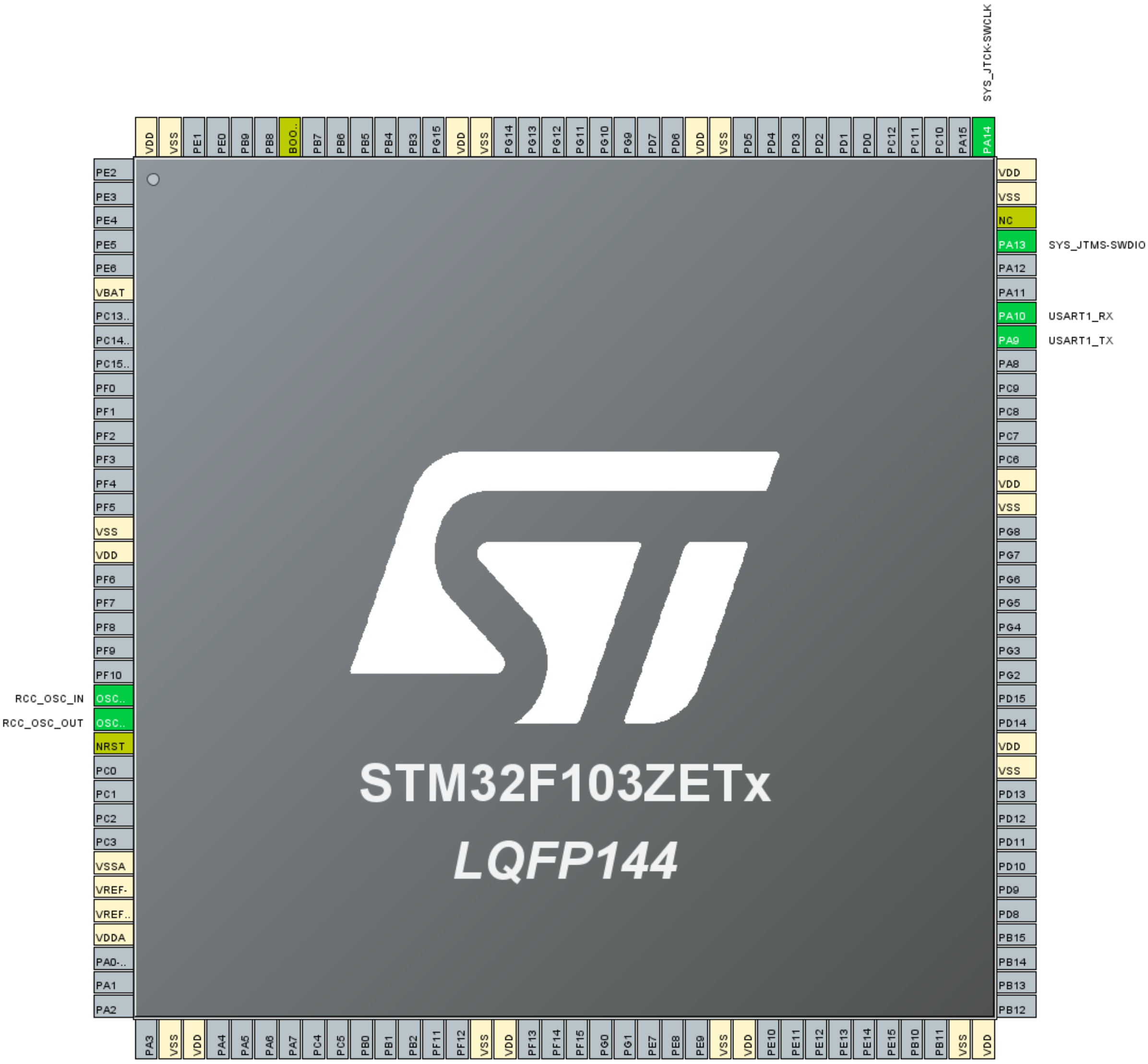
The project configuration part, which is not omitted, is the key point to configure in this tutorial.

```
Please refer to [2. Development environment construction and use: STM32CubeIDE
installation and use] to understand how to configure the omitted parts of the
project.
```

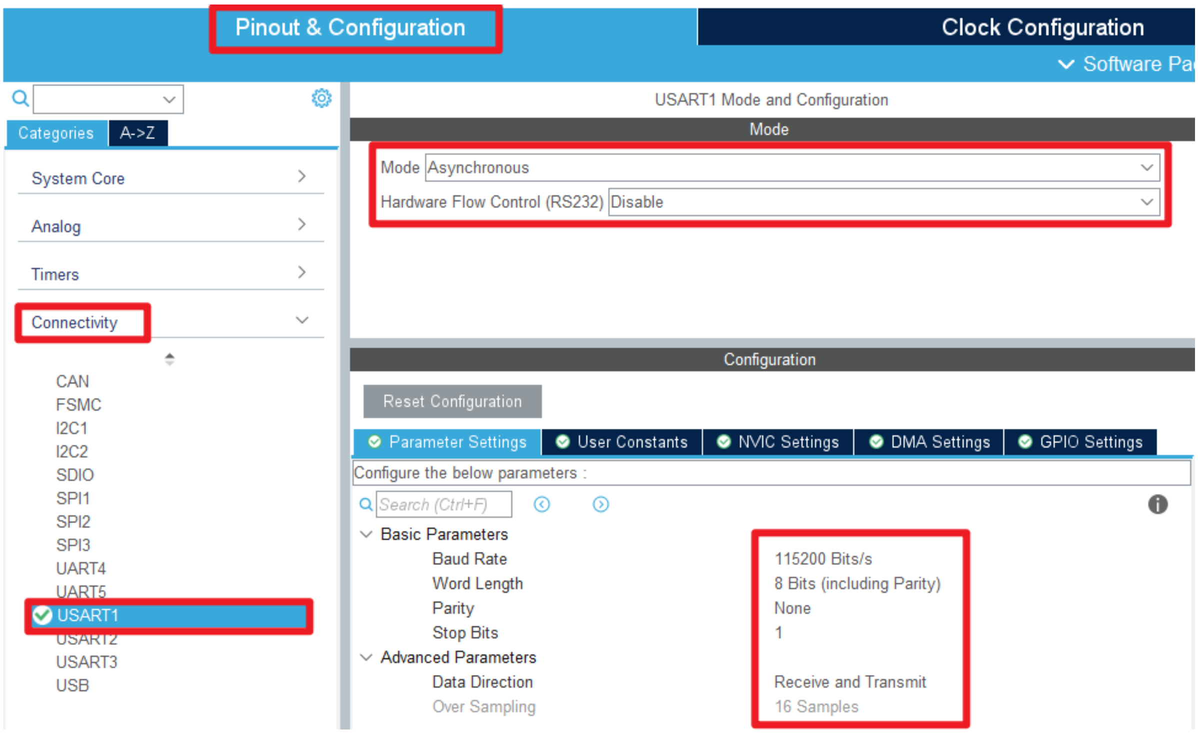
3.2、Pin configuration

- **Configure the specified pin function**

You can directly select the corresponding pin number in the pin view, and the corresponding option will appear when the mouse is left clicked



- **USART1**



USART1 Mode and Configuration

Mode

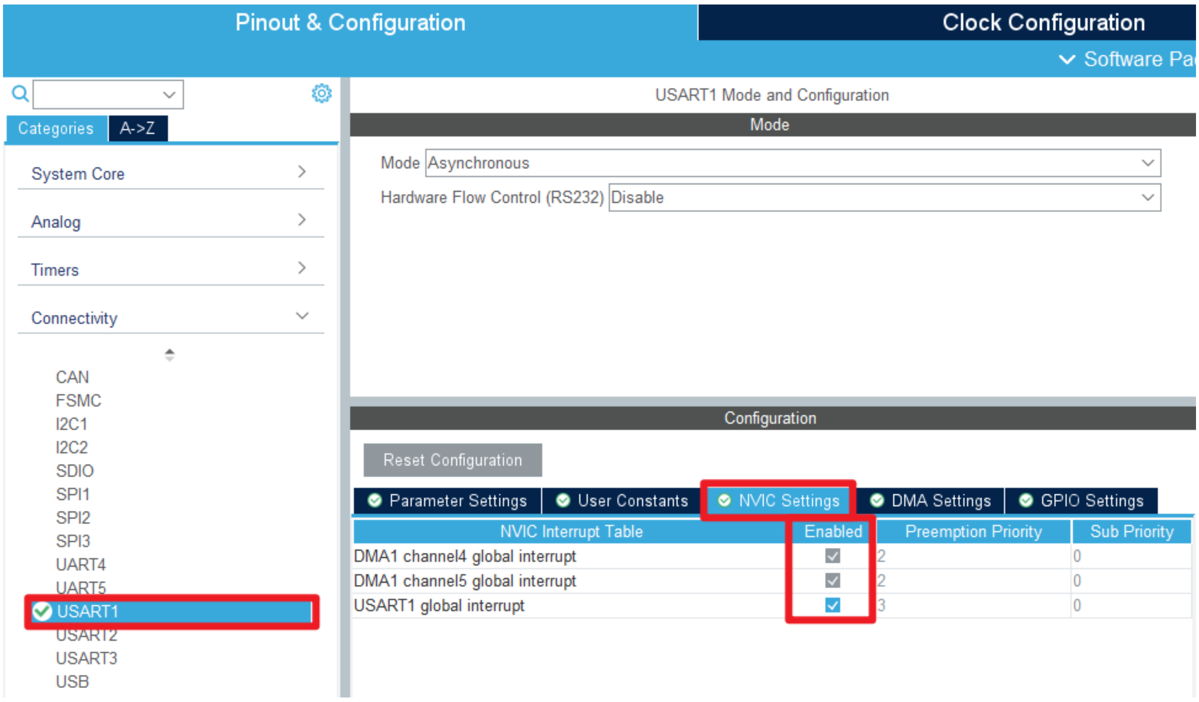
Mode: Asynchronous

Hardware Flow Control (RS232): Disable

Configuration

Configure the below parameters :

Basic Parameters	
Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1
Advanced Parameters	
Data Direction	Receive and Transmit
Over Sampling	16 Samples



USART1 Mode and Configuration

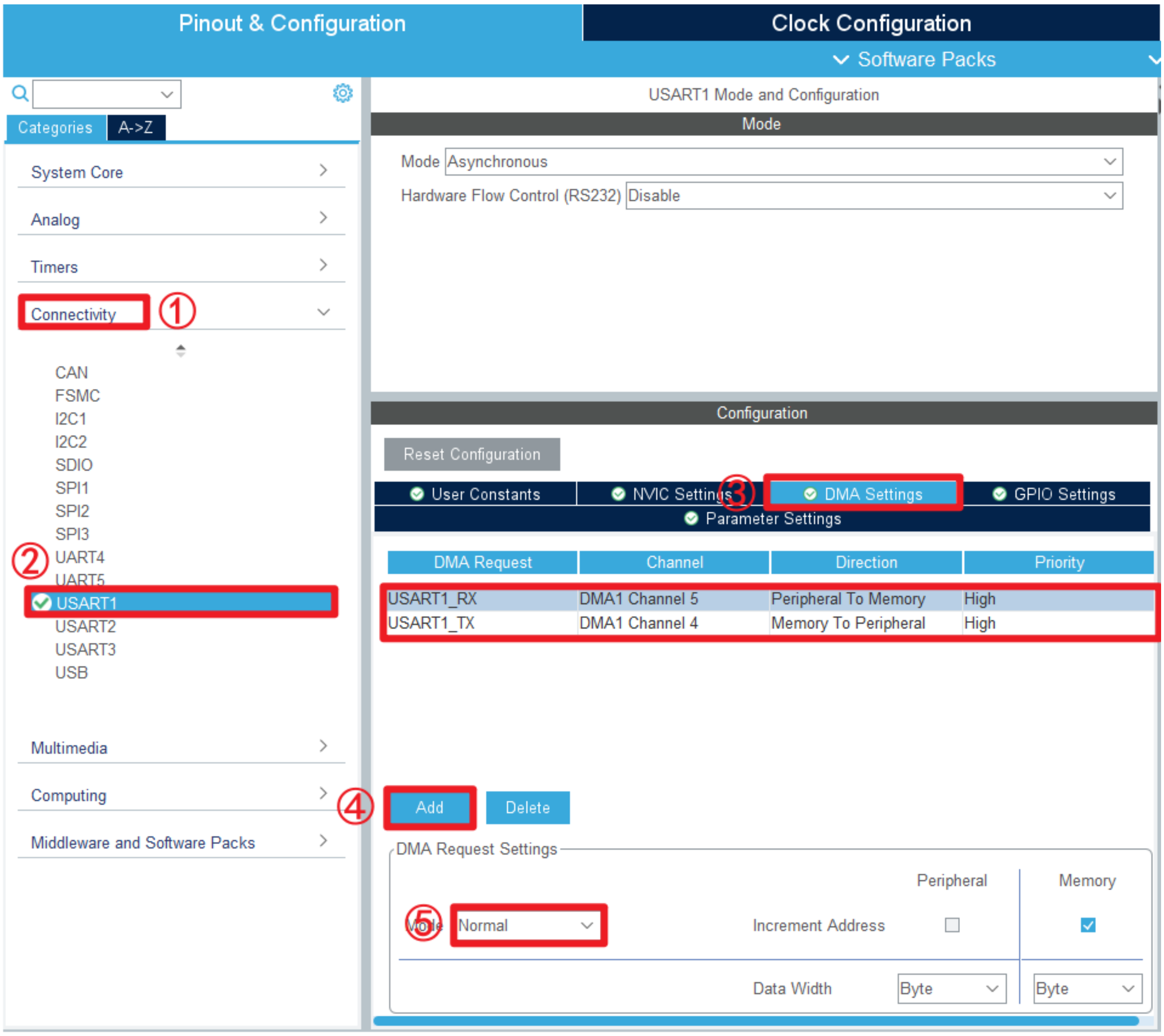
Mode

Mode: Asynchronous

Hardware Flow Control (RS232): Disable

Configuration

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
DMA1 channel4 global interrupt	☑	2	0
DMA1 channel5 global interrupt	☑	2	0
USART1 global interrupt	☑	3	0



Mode Asynchronous

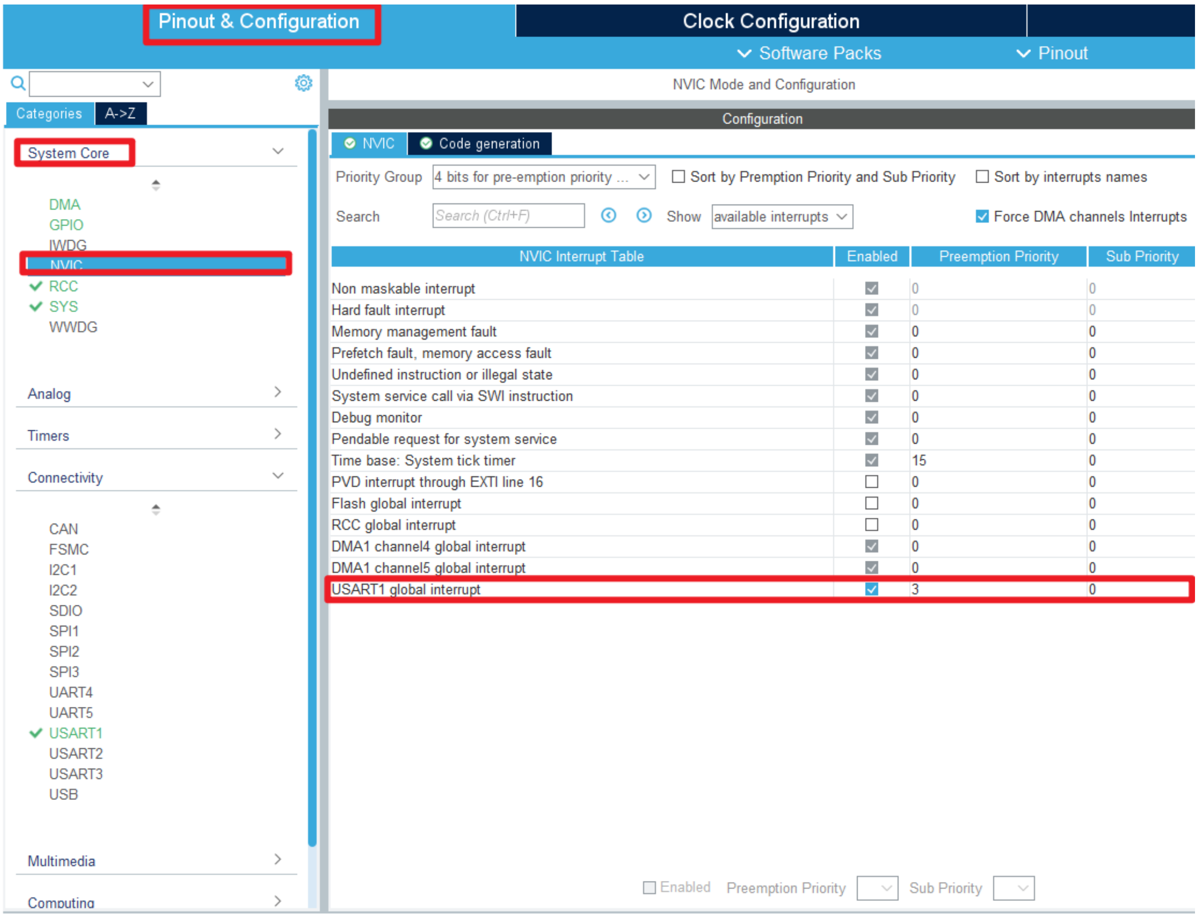
Hardware Flow Control (RS232) Disable

DMA Request	Channel	Direction	Priority
USART1_RX	DMA1 Channel 5	Peripheral To Memory	High
USART1_TX	DMA1 Channel 4	Memory To Peripheral	High

		Peripheral	Memory
Mode Normal	Increment Address	□	☑
	Data Width	Byte	Byte

- **NVIC**

  NVIC can modify interrupt priority



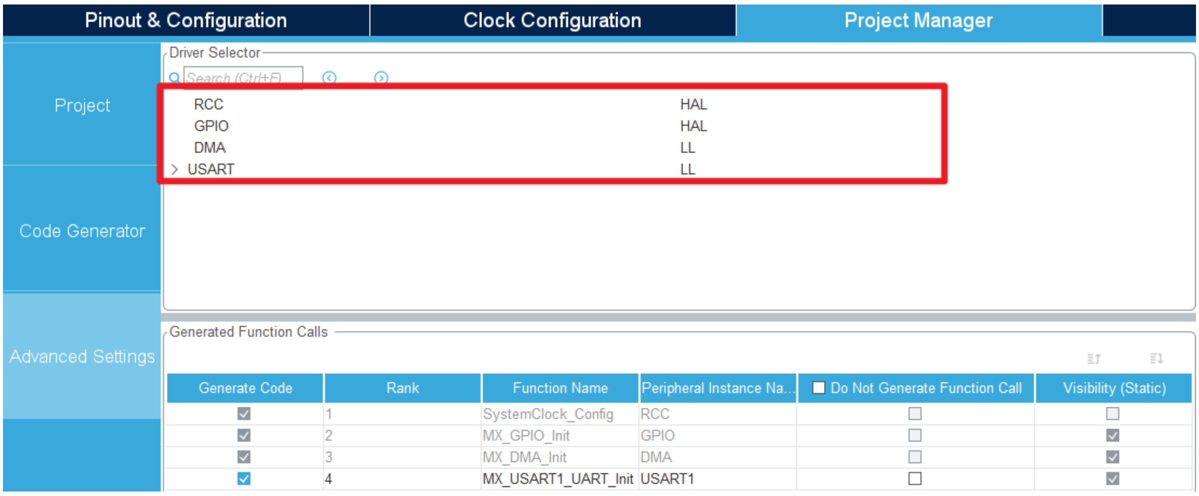
Priority Group: 4 bits for pre-emption priority ...   ☐ Sort by Premption Priority and Sub Priority   ☐ Sort by interrupts names

Search: Search (Ctrl+F)   Show: available interrupts   ☑ Force DMA channels Interrupts

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	☑	0	0
Hard fault interrupt	☑	0	0
Memory management fault	☑	0	0
Prefetch fault, memory access fault	☑	0	0
Undefined instruction or illegal state	☑	0	0
System service call via SWI instruction	☑	0	0
Debug monitor	☑	0	0
Pendable request for system service	☑	0	0
Time base: System tick timer	☑	15	0
PVD interrupt through EXTI line 16	☐	0	0
Flash global interrupt	☐	0	0
RCC global interrupt	☐	0	0
DMA1 channel4 global interrupt	☑	0	0
DMA1 channel5 global interrupt	☑	0	0
USART1 global interrupt	☑	3	0

☐ Enabled   Preemption Priority   Sub Priority

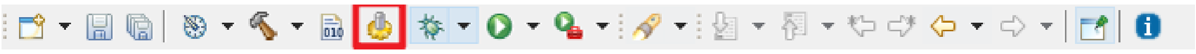
- **Advanced Settings**



RCC	HAL
GPIO	HAL
DMA	LL
> USART	LL

Generate Code	Rank	Function Name	Peripheral Instance Na...	☐ Do Not Generate Function Call	Visibility (Static)
☑	1	SystemClock_Config	RCC	☐	☐
☑	2	MX_GPIO_Init	GPIO	☐	☑
☑	3	MX_DMA_Init	DMA	☐	☑
☑	4	MX_USART1_UART_Init	USART1	☐	☑

- **Generating code**



4、 Main Function

This section mainly introduces the functional code written by users. **Detailed code can be opened by yourself in the project file we provide, and enter the Bsp folder to view the source code.** .

4.1、User function

> **function：USART1_DataByte**

Function prototypes	**void USART1_DataByte(uint8_t data_byte)**
Functional Description	Sending a single byte
Input parameters	**data_byte**： Data to be sent
Return value	None
Tips	Define the DMA_USART macro to switch between DMA transfer and USART directly

> **function：USART1_DataString**

Function prototypes	**void USART1_DataString(uint8_t *data_str, uint16_t datasize)**
Functional Description	Sending strings
Input parameters1	**data_str**： The first address to send data to
Input parameters2	**datasize**： Data size
Return value	None
Tips	Define the DMA_USART macro to switch between DMA transfer and USART directly

4.2、LL library function analysis

LL library functions that have been covered in the previous tutorial will not be covered in the tutorial

```
If you want to find the HAL library and LL library function analysis involved in
the entire tutorial, you can view the documents in the folder [8. STM32 Manual:
STM32F1_HAL Library and LL Library_User Manual]
```

> **function：LL_DMA_SetDataTransferDirection**

Function prototypes	**void LL_DMA_SetDataTransferDirection(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t Direction)**
Functional Description	Set the direction of data transmission
Input parameters1	**DMAx**： DMA handle address

Function prototypes	void LL_DMA_SetDataTransferDirection(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t Direction)
Input parameters2	**Channel**： DMA channel number in the range 0 to 7
Input parameters3	**Direction**： Direction of data transmission
Return value	None

> **function：LL_DMA_SetChannelPriorityLevel**

Function prototypes	void LL_DMA_SetChannelPriorityLevel(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t Priority)
Functional Description	Set the priority of the DMA channel
Input parameters1	**DMAx**： DMA handle address
Input parameters2	**Channel**： DMA channel number in the range 0 to 7
Input parameters3	**Priority**： priority
Return value	None

> **function：LL_DMA_SetMode**

Function prototypes	void LL_DMA_SetMode(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t Mode)
Functional Description	Set DMA working mode
Input parameters1	**DMAx**： DMA handle address
Input parameters2	**Channel**： DMA channel number in the range 0 to 7
Input parameters3	**Mode**： Mode of work
Return value	None

> **function：LL_DMA_SetPeriphIncMode**

Function prototypes	void LL_DMA_SetPeriphIncMode (DMA_TypeDef *DMAx, uint32_t Channel, uint32_t PeriphOrM2MSrcIncMode)
Functional Description	Set DMA peripheral delta mode

Function prototypes	**void LL_DMA_SetPeriphIncMode (DMA_TypeDef *DMAx, uint32_t Channel, uint32_t PeriphOrM2MSrcIncMode)**
Input parameters1	**DMAx**：DMA handle address
Input parameters2	**Channel**：DMA channel number in the range 0 to 7
Input parameters3	**PeriphOrM2MSrcIncMode**：Incremental mode selection
Return value	None

> **function：LL_DMA_SetMemoryIncMode**

Function prototypes	**void LL_DMA_SetMemoryIncMode (DMA_TypeDef *DMAx, uint32_t Channel, uint32_t MemoryOrM2MDstIncMode)**
Functional Description	Set DMA memory increment mode
Input parameters1	**DMAx**：DMA handle address
Input parameters2	**Channel**：DMA channel number in the range 0 to 7
Input parameters3	**MemoryOrM2MDstIncMode**：Incremental mode selection
Return value	None

> **function：LL_DMA_SetPeriphSize**

Function prototypes	**void LL_DMA_SetPeriphSize(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t PeriphOrM2MSrcDataSize)**
Functional Description	Set the peripheral data size
Input parameters1	**DMAx**：DMA handle address
Input parameters2	**Channel**：DMA channel number in the range 0 to 7
Input parameters3	**PeriphOrM2MSrcDataSize**：Data size
Return value	None

> **function：LL_DMA_SetMemorySize**

Function prototypes	**void LL_DMA_SetMemorySize (DMA_TypeDef *DMAx, uint32_t Channel, uint32_t MemoryOrM2MDstDataSize)**
Functional Description	Set the memory data size
Input parameters1	**DMAx**： DMA handle address
Input parameters2	**Channel**： DMA channel number in the range 0 to 7
Input parameters3	**MemoryOrM2MDstDataSize**： Data size
Return value	None

> **function：LL_DMA_SetMemoryAddress**

Function prototypes	**void LL_DMA_SetMemoryAddress(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t MemoryAddress)**
Functional Description	Set the DMA memory address
Input parameters1	**DMAx**： DMA handle address
Input parameters2	**Channel**： DMA channel number in the range 0 to 7
Input parameters3	**MemoryAddress**： Memory address
Return value	None

> **function：LL_DMA_SetPeriphAddress**

Function prototypes	**void LL_DMA_SetPeriphAddress(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t PeriphAddress)**
Functional Description	Set the DMA peripheral address
Input parameters1	**DMAx**： DMA handle address
Input parameters2	**Channel**： DMA channel number in the range 0 to 7
Input parameters3	**PeriphAddress**： Peripheral device address
Return value	None

> **function：LL_DMA_SetDataLength**

Function prototypes	**void LL_DMA_SetDataLength(DMA_TypeDef *DMAx, uint32_t Channel, uint32_t NbData)**
Functional Description	Set the length of DMA transfer data
Input parameters1	**DMAx**： DMA handle address
Input parameters2	**Channel**： DMA channel number in the range 0 to 7
Input parameters3	**NbData**： Data length
Return value	None

> **function：LL_DMA_EnableChannel**

Function prototypes	**void LL_DMA_EnableChannel(DMA_TypeDef *DMAx, uint32_t Channel)**
Functional Description	Enable DMA channels
Input parameters1	**DMAx**： DMA handle address
Input parameters2	**Channel**： DMA channel number in the range 0 to 7
Return value	None

> **function：LL_DMA_DisableChannel**

Function prototypes	**void LL_DMA_DisableChannel(DMA_TypeDef *DMAx, uint32_t Channel)**
Functional Description	Closing the DMA channel
Input parameters1	**DMAx**： DMA handle address
Input parameters2	**Channel**： DMA channel number in the range 0 to 7
Return value	None

> **function：LL_USART_EnableDMAReq_RX**

Function prototypes	**void LL_USART_EnableDMAReq_RX(USART_TypeDef *USARTx)**
Functional Description	Enable DMA serial port reception
Input parameters	**USARTx**： USART handle address
Return value	None

> function：LL_USART_EnableDMAReq_TX

Function prototypes	void LL_USART_EnableDMAReq_TX(USART_TypeDef *USARTx)
Functional Description	Enable DMA serial port sending
Input parameters	**USARTx**：USART handle address
Return value	None

5、Experimental phenomenon

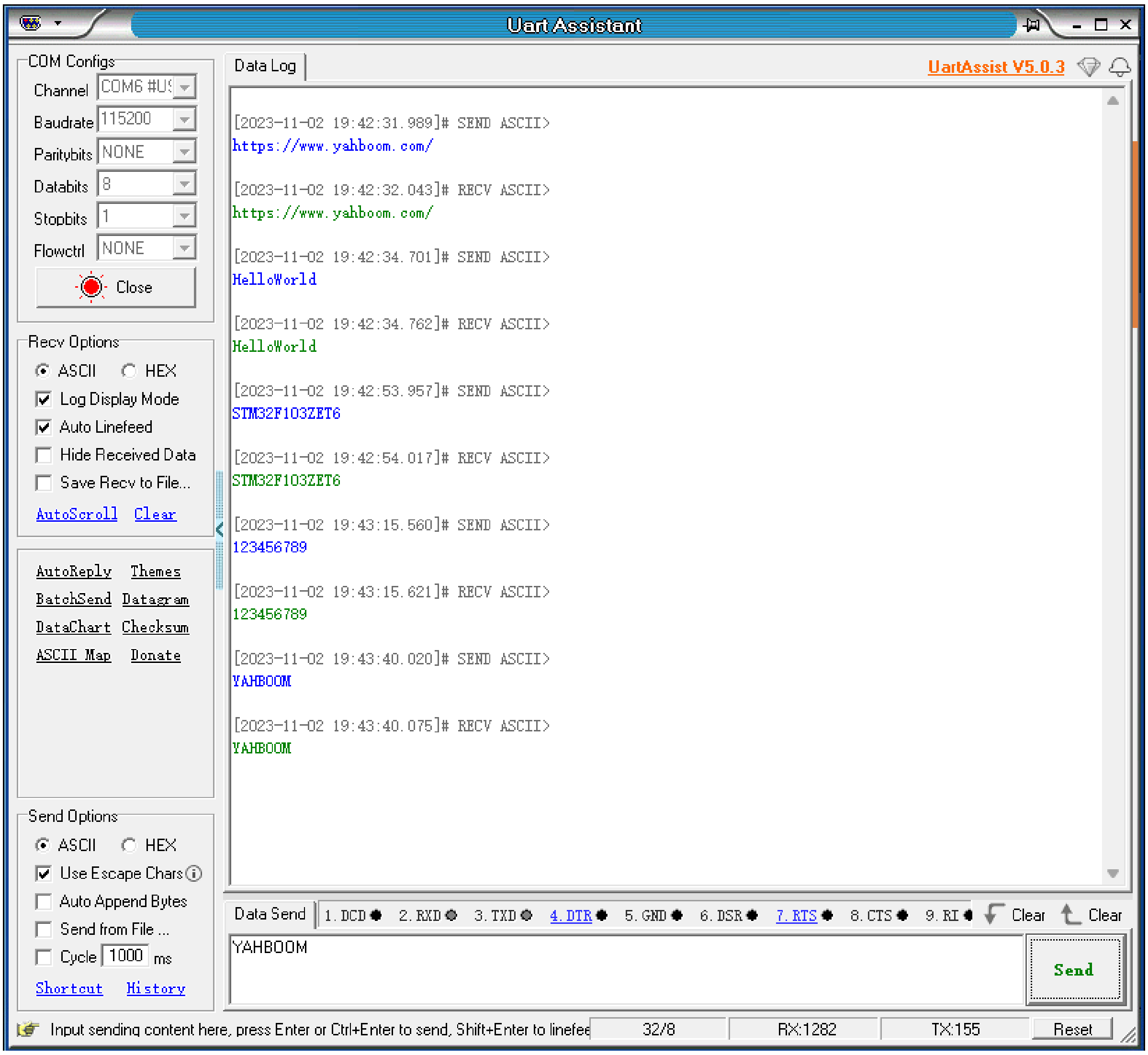
After downloading the program successfully, press the RESET button of the development board to open the serial debugging assistant to observe the phenomenon

```
For program download, please refer to [2. Development environment construction
and use: program download and simulation]
```

phenomenon:

Press RESET button, serial debugging assistant will print USRT+DMA Class! Character;

Then through the serial port debug assistant to send what information will print the corresponding information



Channel COM6 #U
Baudrate 115200
Paritybits NONE
Databits 8
Stopbits 1
Flowctrl NONE

● ASCII ○ HEX
☑ Log Display Mode
☑ Auto Linefeed
□ Hide Received Data
□ Save Recv to File...

● ASCII ○ HEX
☑ Use Escape Chars
□ Auto Append Bytes
□ Send from File ...
□ Cycle 1000 ms

```
[2023-11-02 19:42:31.989]# SEND ASCII>
https://www.yahboom.com/

[2023-11-02 19:42:32.043]# RECV ASCII>
https://www.yahboom.com/

[2023-11-02 19:42:34.701]# SEND ASCII>
HelloWorld

[2023-11-02 19:42:34.762]# RECV ASCII>
HelloWorld

[2023-11-02 19:42:53.957]# SEND ASCII>
STM32F103ZET6

[2023-11-02 19:42:54.017]# RECV ASCII>
STM32F103ZET6

[2023-11-02 19:43:15.560]# SEND ASCII>
123456789

[2023-11-02 19:43:15.621]# RECV ASCII>
123456789

[2023-11-02 19:43:40.020]# SEND ASCII>
YAHBOOM

[2023-11-02 19:43:40.075]# RECV ASCII>
YAHBOOM
```

YAHBOOM

Input sending content here, press Enter or Ctrl+Enter to send, Shift+Enter to linefee 32/8 RX:1282 TX:155