南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Embedded System and Microcomputer Principle

LAB6 USART Conmmunication

2023 Fall
wangq9@mail.sustech.edu.cn

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

CONTENTS

01

USART Function Description

1. USART Function Description
-- Basic Description of Serial Port

- According to data transmission direction, serial port is divided into 3 types
  - **Simplex**: data are transmitted in one direction
  - **Half duplex**: data is allowed to be transmitted in two directions. However, at a certain time, data is only allowed to be transmitted in one direction.
  - **Full duplex**: it allows data to be transmitted in two directions at the same time. Therefore, full duplex communication is the combination of two simplex communication modes. It requires that both transmitting equipment and receiving equipment have independent receiving and transmitting capabilities.

1. USART Function Description
-- Basic Description of Serial Port

- Communication mode of serial communication
  - **Synchronous communication**: synchronous signal transmission with clock
    - SPI（Serial Peripheral Interface）
    - I^{2}C（Inter-Integrated Circuit）
  - **Asynchronous communication**: synchronous signal without clock
    - UART（Universal Asynchronous Receiver/Transmitter）

1. USART Function Description
-- Basic Description of Serial Port

- Comparison of common serial communication interfaces

Communication standard	Pin description	Communication mode	Communication direction
UART	TXD: transmit data out RXD: receive data in GND: common GND	Asynchronous communication	Full duplex
1-WIRE	DQ: sender / receiver	Asynchronous communication	Half duplex
SPI	SCK: synchronous clock MISO: host input, slave output MOSI: host output, slave input	Synchronous communication	Full duplex
I^2C	SCL: synchronous clock SDA: data input / output	Synchronous communication	Half duplex

1. USART Function Description
-- STM32 serial port

- Serial communication interface of STM32
  - 3 USART
    - USART (Universal Synchronous/Asynchronous Receiver/ Transmitter) supports both synchronous and asynchronous transmission. In asynchronous mode, a USART bidirectional communication needs two pins: receive data in(RX) and transmit data out(TX). In synchronous mode, an addition pin(SCLK) for clock synchronous is needed. In most cases, we use USART in asynchronous mode, which is UART, and only UART is involved in the following lab.
  - 2 UART

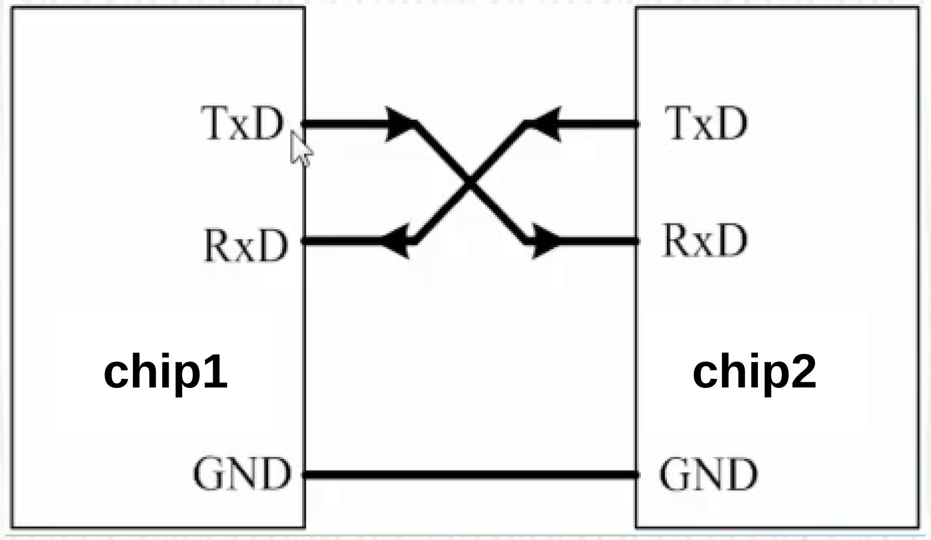
1. USART Function Description
-- STM32 serial port

- STM32 UART pin

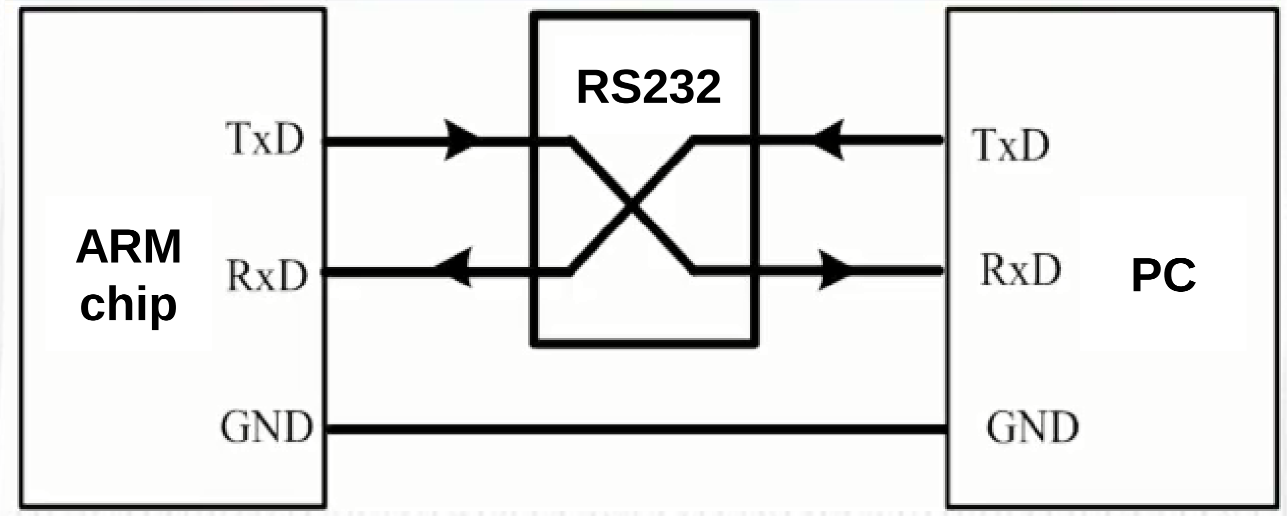
Serial port number	RXD	TXD
1	PA10	PA9
2	PA3	PA2
3	PB11	PB10
4	PC11	PC10
5	PD2	PC12

1. USART Function Description -- UART

- UART asynchronous communication mode pin connection method



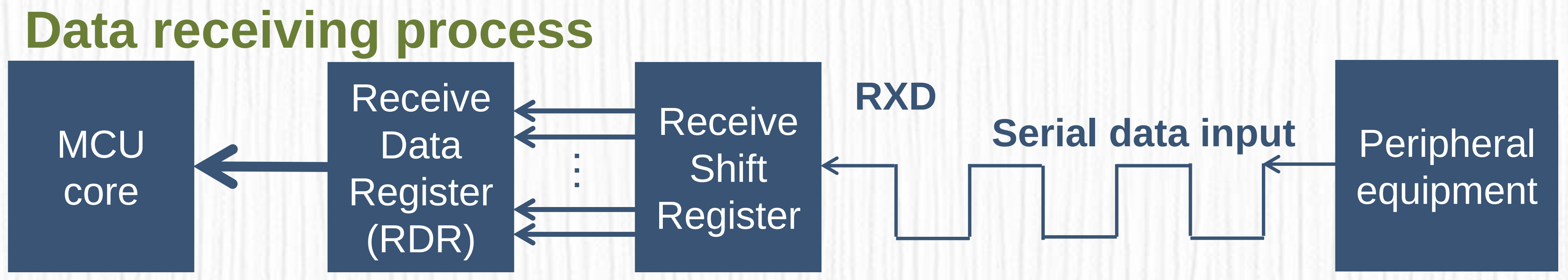
(a)
communication between two chips with the same electrical characteristics

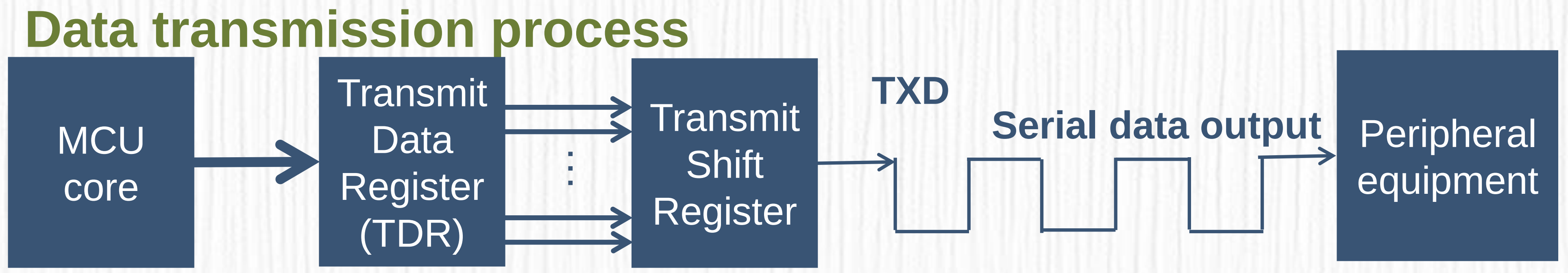


(b)
communication between computer and chip with different electrical characteristics

1. USART Function Description -- UART

- UART serial asynchronous communication process

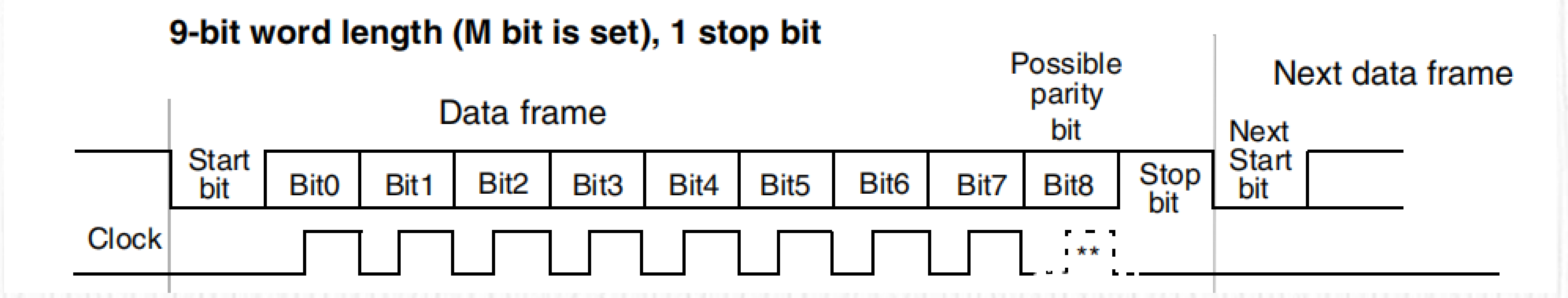




1. USART Function Description -- UART

- Parameters to be defined for STM32 serial asynchronous communication
  - Start bit
  - Data bits (8 bits or 9 bits)
  - Parity bit (odd or even)
  - Stop bit
  - Baud rate setting

An example



南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

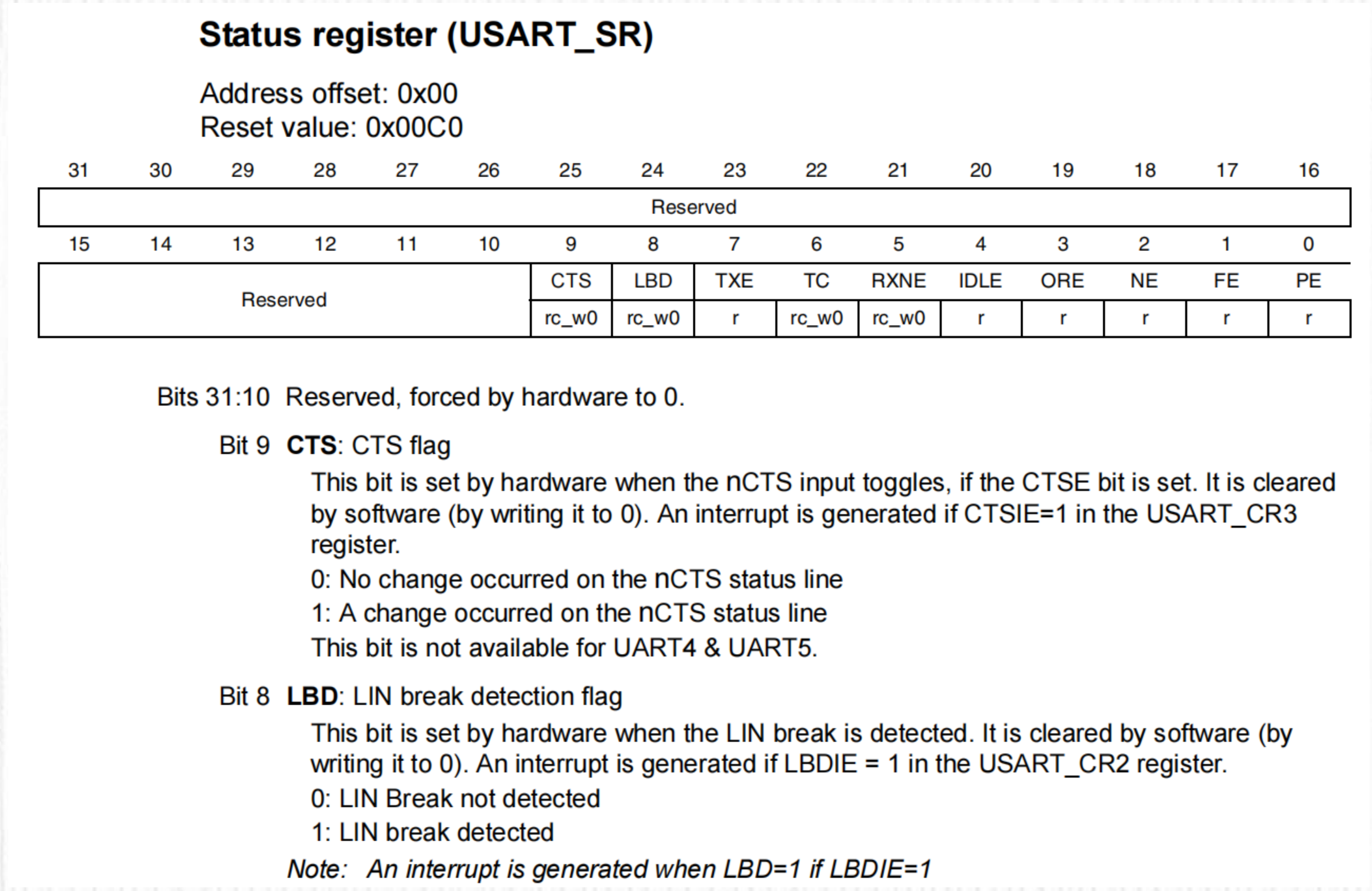
02

STM32 Serial Port Register

2. STM32 Serial Port Register

- USART_SR Status register
- USART_DR Data register
- USART_BRR Baud rate register
- USART_CR Control register

2. STM32 Serial Port Register
-- STM32 USART_SR



Status register (USART_SR)

Address offset: 0x00

Reset value: 0x00C0

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

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

Bits 31:10 Reserved, forced by hardware to 0.

Bit 9 **CTS**: CTS flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software (by writing it to 0). An interrupt is generated if CTSIE=1 in the USART_CR3 register.

0: No change occurred on the nCTS status line

1: A change occurred on the nCTS status line

This bit is not available for UART4 & UART5.

Bit 8 **LBD**: LIN break detection flag

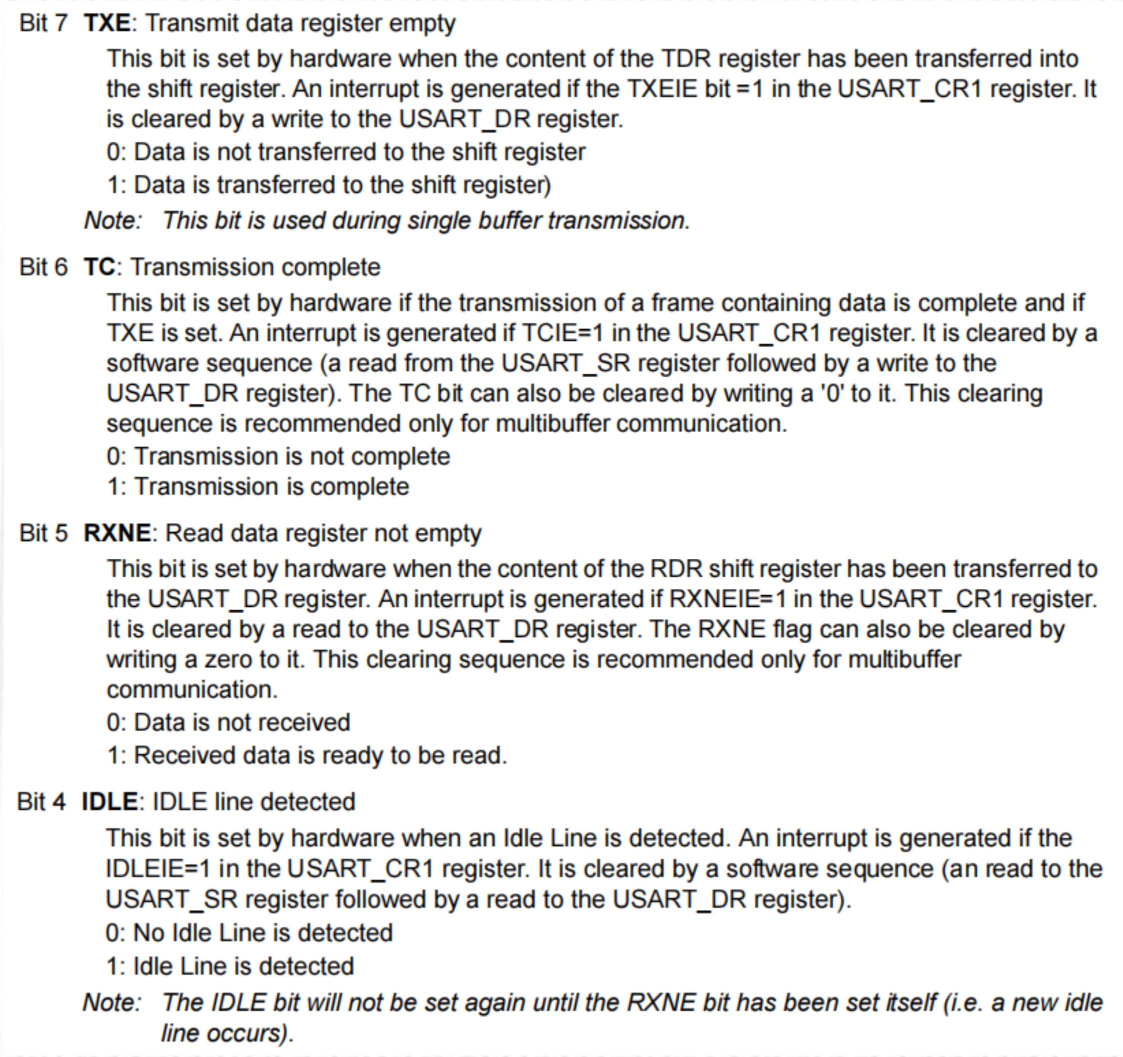
This bit is set by hardware when the LIN break is detected. It is cleared by software (by writing it to 0). An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: An interrupt is generated when LBD=1 if LBDIE=1

2. STM32 Serial Port Register -- STM32 USART_SR(continued)



Bit 7 **TXE**: Transmit data register empty

This bit is set by hardware when the content of the TDR register has been transferred into the shift register. An interrupt is generated if the TXEIE bit =1 in the USART_CR1 register. It is cleared by a write to the USART_DR register.

0: Data is not transferred to the shift register

1: Data is transferred to the shift register)

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit is set by hardware if the transmission of a frame containing data is complete and if TXE is set. An interrupt is generated if TCIE=1 in the USART_CR1 register. It is cleared by a software sequence (a read from the USART_SR register followed by a write to the USART_DR register). The TC bit can also be cleared by writing a '0' to it. This clearing sequence is recommended only for multibuffer communication.

0: Transmission is not complete

1: Transmission is complete

Bit 5 **RXNE**: Read data register not empty

This bit is set by hardware when the content of the RDR shift register has been transferred to the USART_DR register. An interrupt is generated if RXNEIE=1 in the USART_CR1 register. It is cleared by a read to the USART_DR register. The RXNE flag can also be cleared by writing a zero to it. This clearing sequence is recommended only for multibuffer communication.

0: Data is not received

1: Received data is ready to be read.

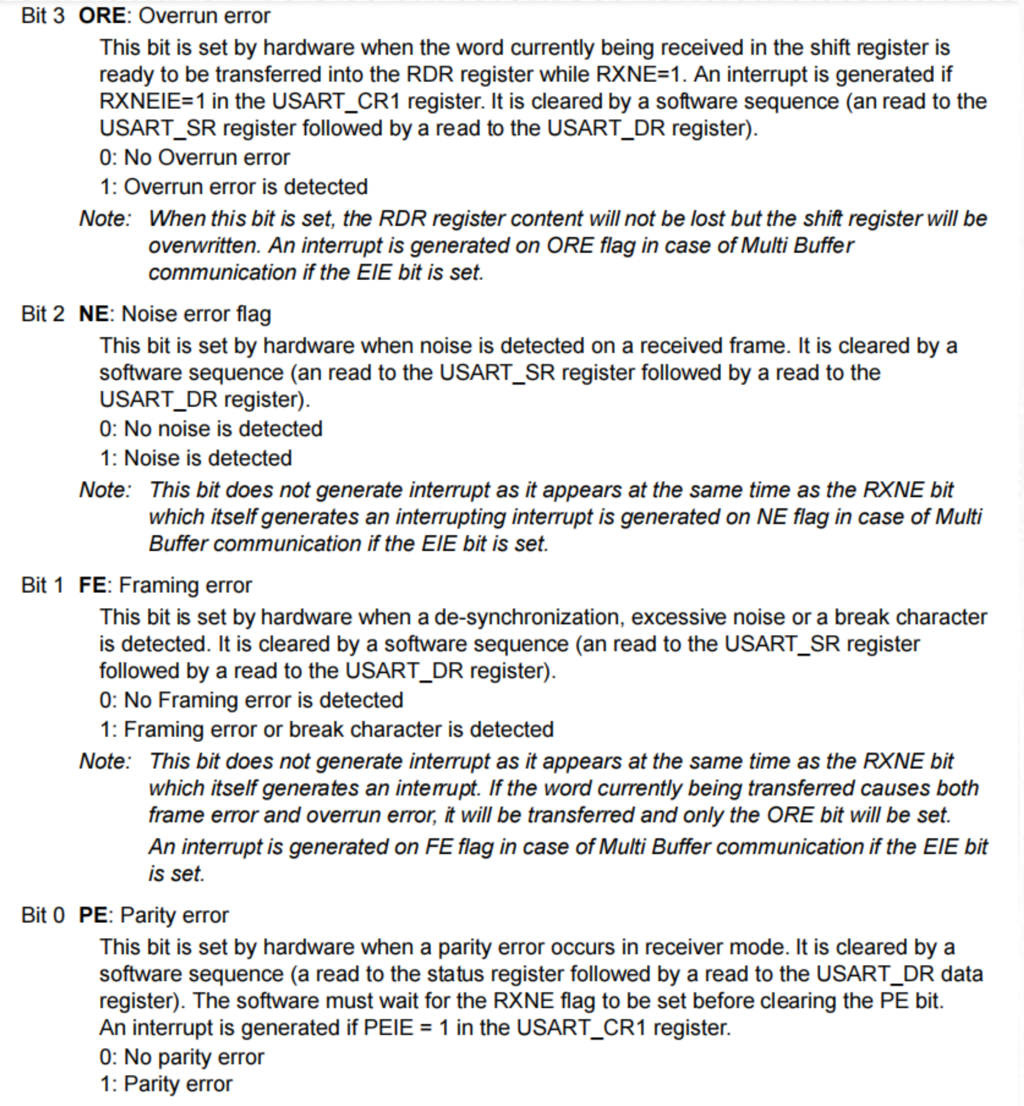
Bit 4 **IDLE**: IDLE line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if the IDLEIE=1 in the USART_CR1 register. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).

0: No Idle Line is detected

1: Idle Line is detected

Note: The IDLE bit will not be set again until the RXNE bit has been set itself (i.e. a new idle line occurs).



Bit 3 **ORE**: Overrun error

This bit is set by hardware when the word currently being received in the shift register is ready to be transferred into the RDR register while RXNE=1. An interrupt is generated if RXNEIE=1 in the USART_CR1 register. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).

0: No Overrun error

1: Overrun error is detected

Note: When this bit is set, the RDR register content will not be lost but the shift register will be overwritten. An interrupt is generated on ORE flag in case of Multi Buffer communication if the EIE bit is set.

Bit 2 **NE**: Noise error flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).

0: No noise is detected

1: Noise is detected

Note: This bit does not generate interrupt as it appears at the same time as the RXNE bit which itself generates an interrupting interrupt is generated on NE flag in case of Multi Buffer communication if the EIE bit is set.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).

0: No Framing error is detected

1: Framing error or break character is detected

Note: This bit does not generate interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. If the word currently being transferred causes both frame error and overrun error, it will be transferred and only the ORE bit will be set.

An interrupt is generated on FE flag in case of Multi Buffer communication if the EIE bit is set.

Bit 0 **PE**: Parity error

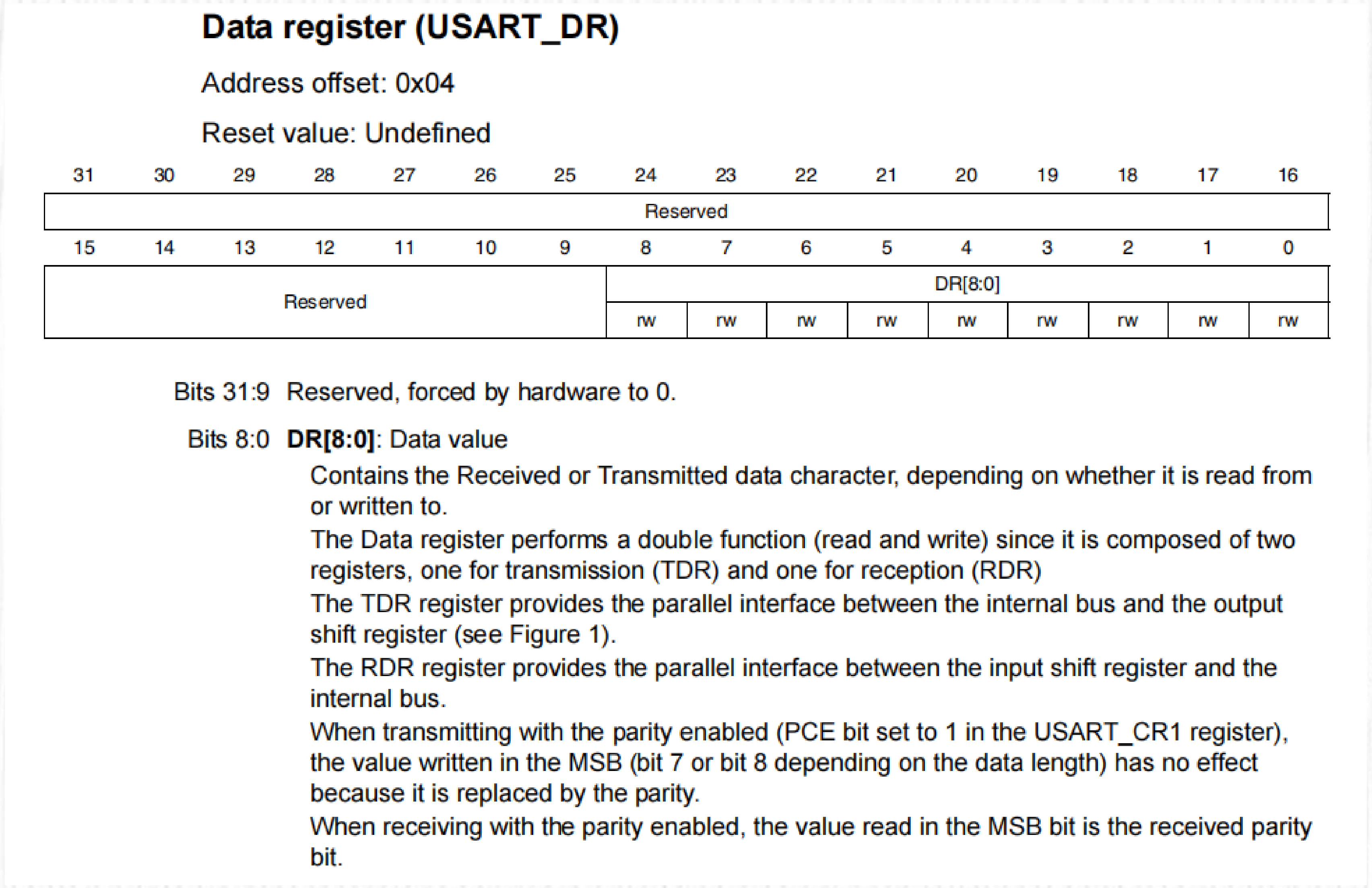
This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by a software sequence (a read to the status register followed by a read to the USART_DR data register). The software must wait for the RXNE flag to be set before clearing the PE bit.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error

1: Parity error

2. STM32 Serial Port Register
-- STM32 USART_DR



Data register (USART_DR)

Address offset: 0x04

Reset value: Undefined

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

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

Bits 31:9 Reserved, forced by hardware to 0.

Bits 8:0 **DR[8:0]**: Data value

Contains the Received or Transmitted data character, depending on whether it is read from or written to.

The Data register performs a double function (read and write) since it is composed of two registers, one for transmission (TDR) and one for reception (RDR)

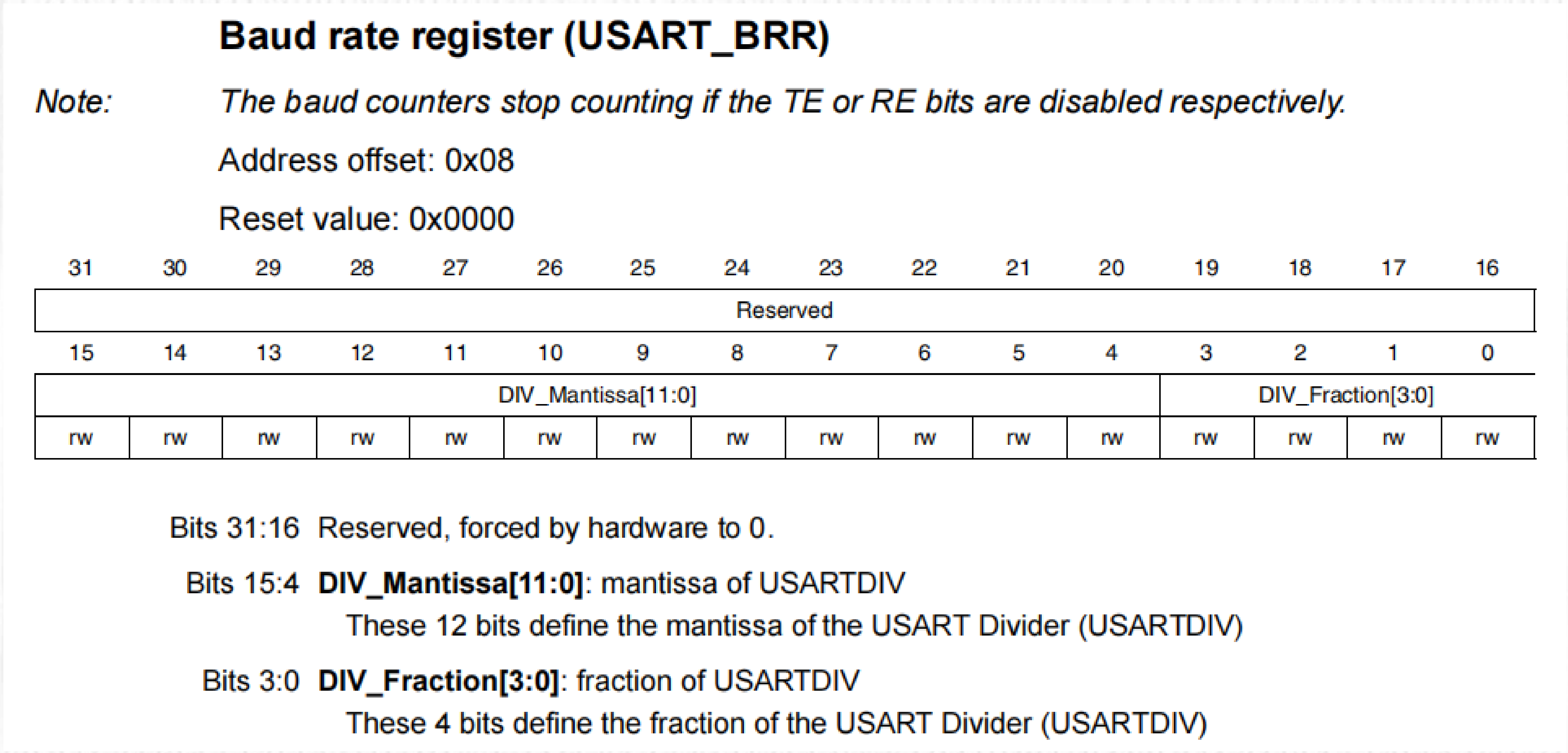
The TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 1).

The RDR register provides the parallel interface between the input shift register and the internal bus.

When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

2. STM32 Serial Port Register -- STM32 USART_BRR



Baud rate register (USART_BRR)

Note: The baud counters stop counting if the TE or RE bits are disabled respectively.

Address offset: 0x08

Reset value: 0x0000

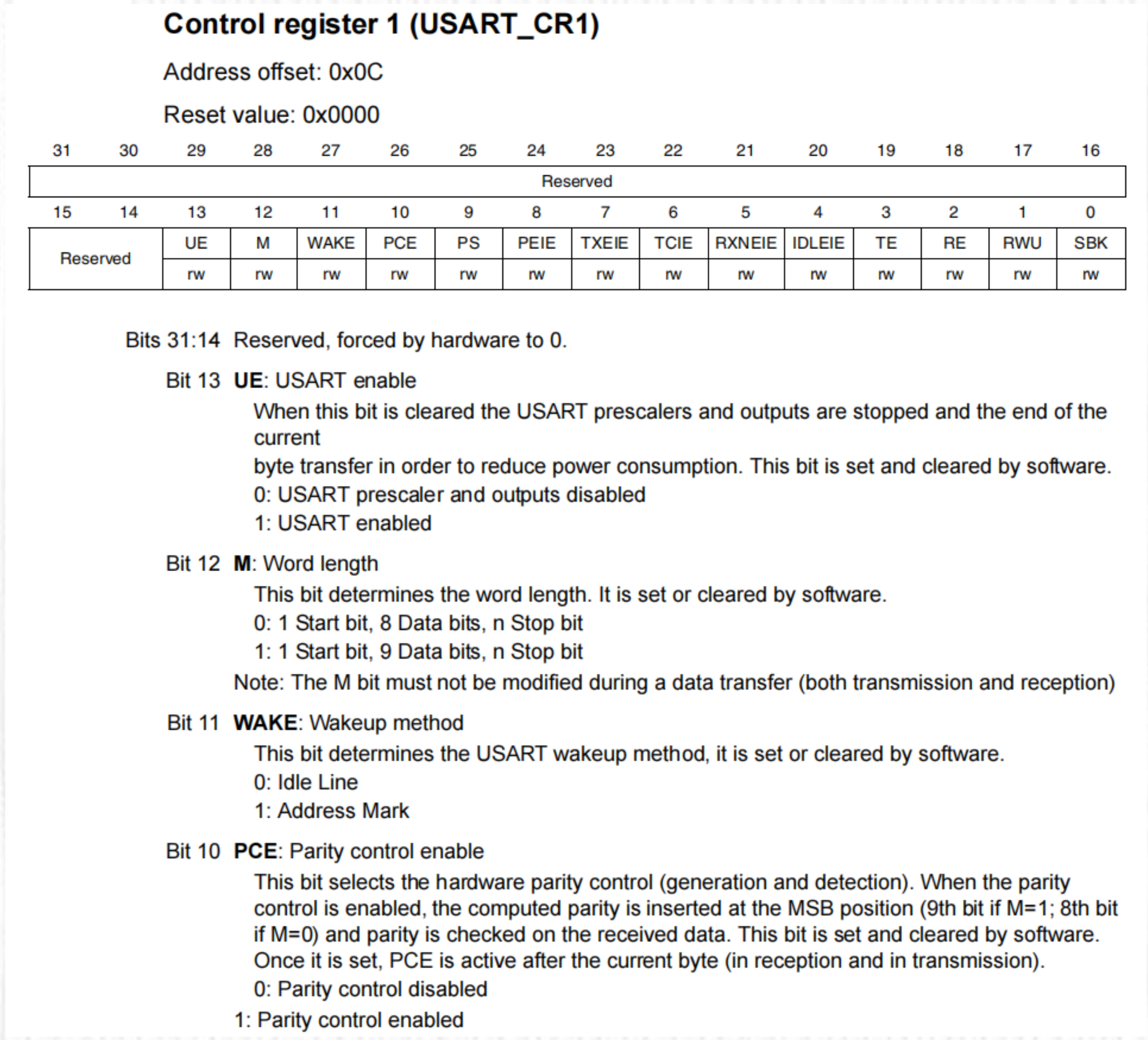
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DIV_Mantissa[11:0]												DIV_Fraction[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, forced by hardware to 0.

Bits 15:4 **DIV_Mantissa[11:0]**: mantissa of USARTDIV
These 12 bits define the mantissa of the USART Divider (USARTDIV)

Bits 3:0 **DIV_Fraction[3:0]**: fraction of USARTDIV
These 4 bits define the fraction of the USART Divider (USARTDIV)

2. STM32 Serial Port Register
-- STM32 USART_CR1



Control register 1 (USART_CR1)

Address offset: 0x0C

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, forced by hardware to 0.

Bit 13 **UE**: USART enable

When this bit is cleared the USART prescalers and outputs are stopped and the end of the current

byte transfer in order to reduce power consumption. This bit is set and cleared by software.

0: USART prescaler and outputs disabled

1: USART enabled

Bit 12 **M**: Word length

This bit determines the word length. It is set or cleared by software.

0: 1 Start bit, 8 Data bits, n Stop bit

1: 1 Start bit, 9 Data bits, n Stop bit

Note: The M bit must not be modified during a data transfer (both transmission and reception)

Bit 11 **WAKE**: Wakeup method

This bit determines the USART wakeup method, it is set or cleared by software.

0: Idle Line

1: Address Mark

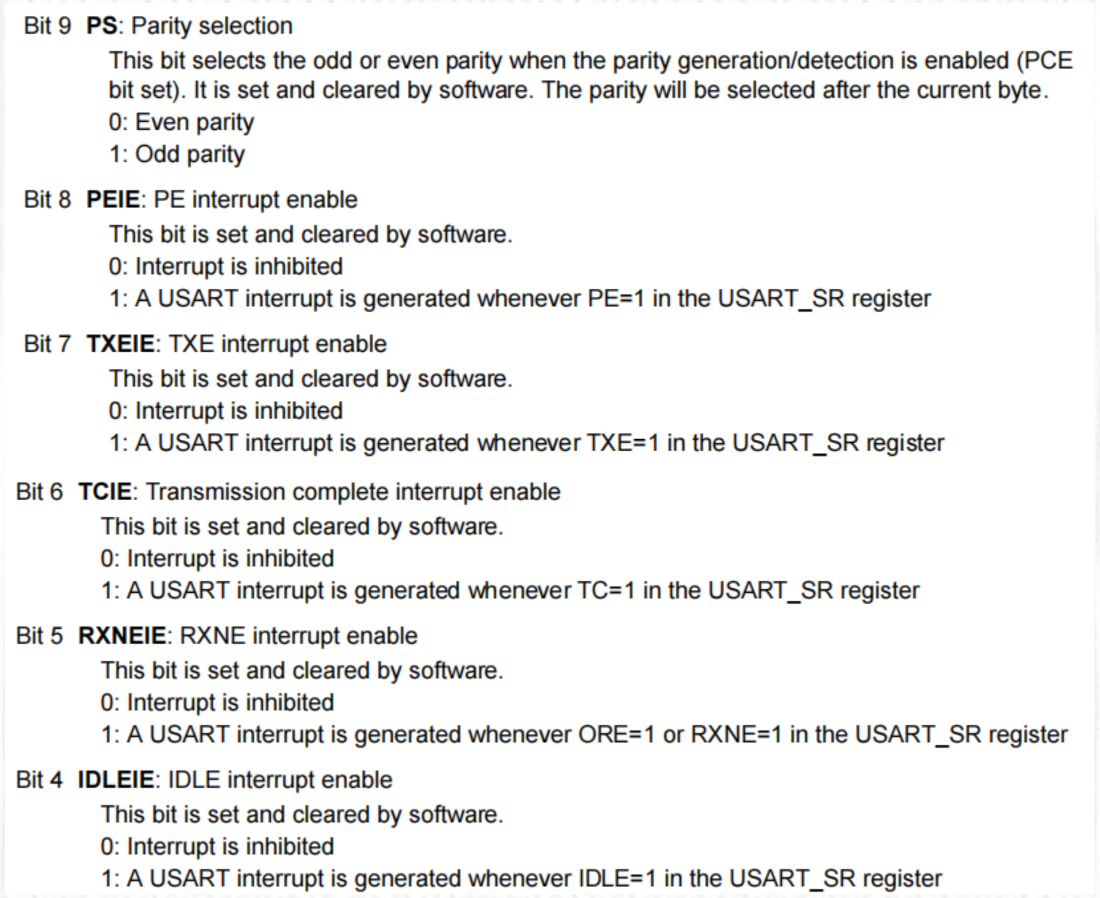
Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

2. STM32 Serial Port Register
-- STM32 USART_CR1(continued)



Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity will be selected after the current byte.

0: Even parity

1: Odd parity

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated whenever PE=1 in the USART_SR register

Bit 7 **TXEIE**: TXE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated whenever TXE=1 in the USART_SR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated whenever TC=1 in the USART_SR register

Bit 5 **RXNEIE**: RXNE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

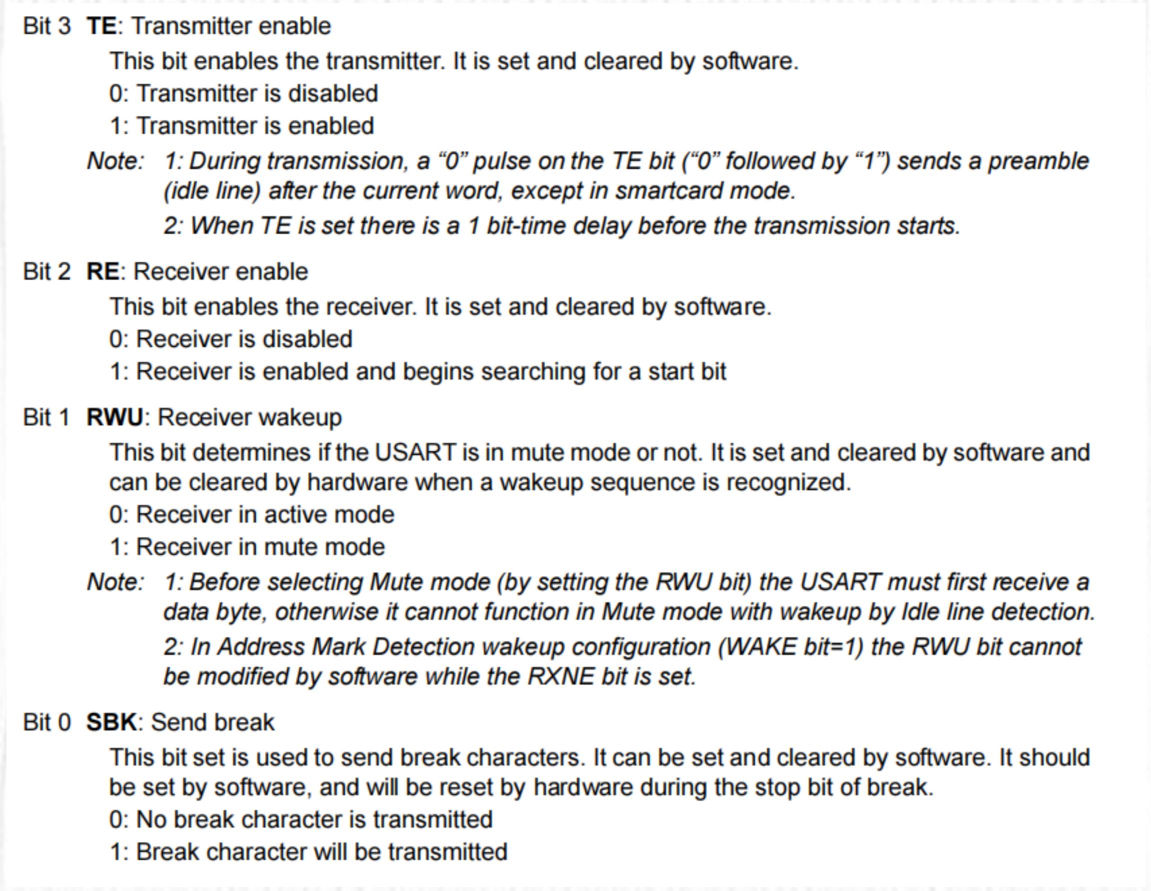
1: A USART interrupt is generated whenever ORE=1 or RXNE=1 in the USART_SR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated whenever IDLE=1 in the USART_SR register



Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: 1: During transmission, a "0" pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word, except in smartcard mode.

2: When TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **RWU**: Receiver wakeup

This bit determines if the USART is in mute mode or not. It is set and cleared by software and can be cleared by hardware when a wakeup sequence is recognized.

0: Receiver in active mode

1: Receiver in mute mode

Note: 1: Before selecting Mute mode (by setting the RWU bit) the USART must first receive a data byte, otherwise it cannot function in Mute mode with wakeup by Idle line detection.

2: In Address Mark Detection wakeup configuration (WAKE bit=1) the RWU bit cannot be modified by software while the RXNE bit is set.

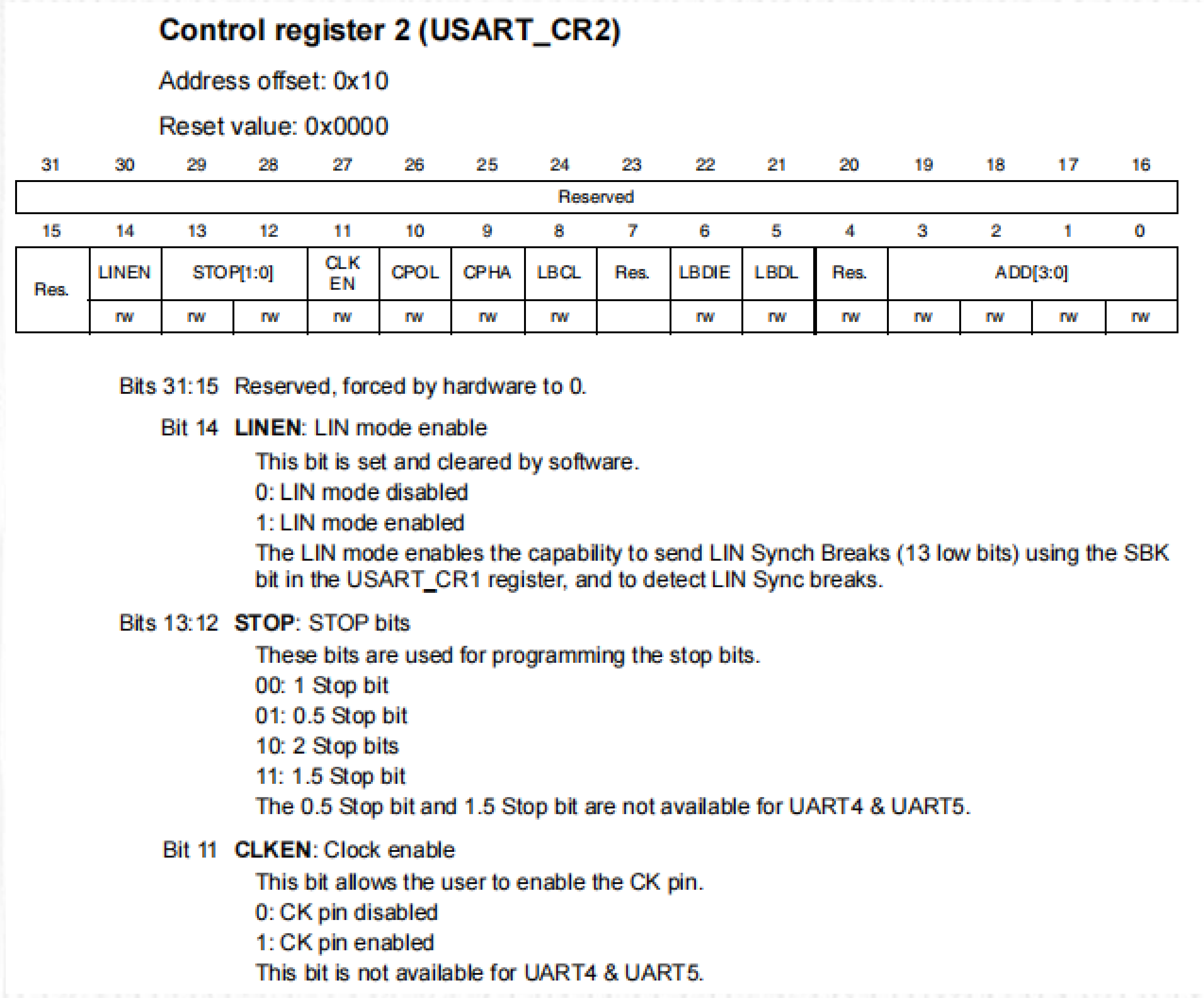
Bit 0 **SBK**: Send break

This bit set is used to send break characters. It can be set and cleared by software. It should be set by software, and will be reset by hardware during the stop bit of break.

0: No break character is transmitted

1: Break character will be transmitted

2. STM32 Serial Port Register
-- STM32 USART_CR2



Control register 2 (USART_CR2)

Address offset: 0x10

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LINEN	STOP[1:0]		CLK EN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	Res.	ADD[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

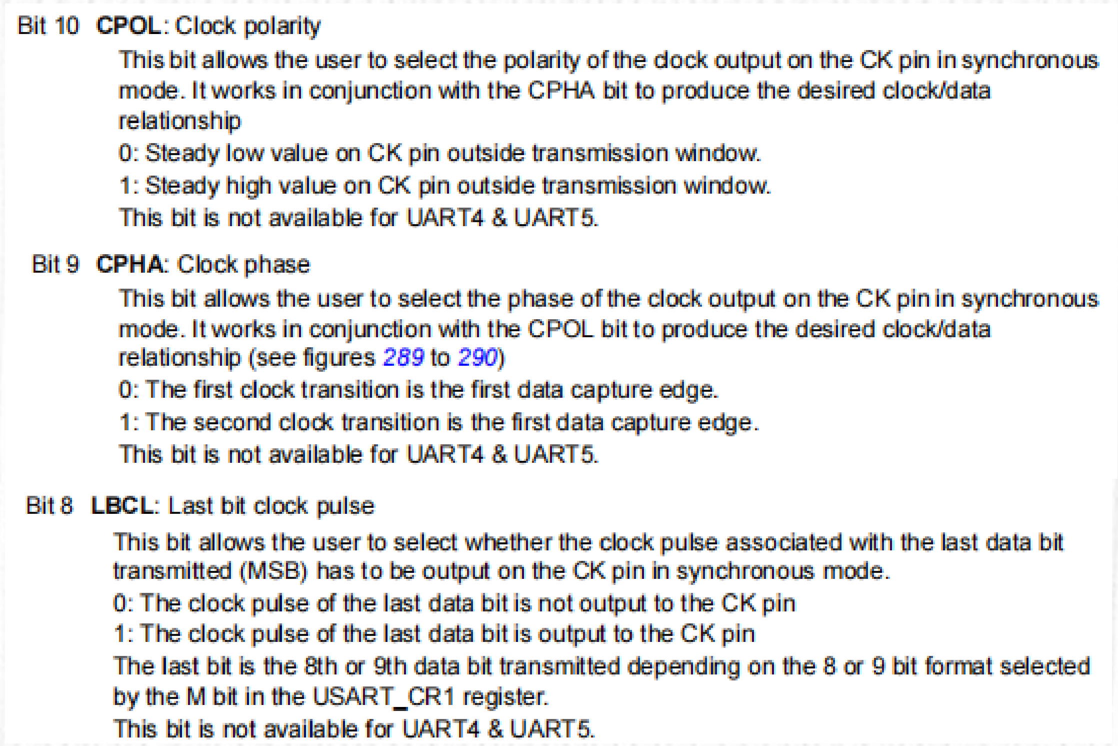
Bits 31:15 Reserved, forced by hardware to 0.

Bit 14 **LINEN**: LIN mode enable
This bit is set and cleared by software.
0: LIN mode disabled
1: LIN mode enabled
The LIN mode enables the capability to send LIN Synch Breaks (13 low bits) using the SBK bit in the USART_CR1 register, and to detect LIN Sync breaks.

Bits 13:12 **STOP**: STOP bits
These bits are used for programming the stop bits.
00: 1 Stop bit
01: 0.5 Stop bit
10: 2 Stop bits
11: 1.5 Stop bit
The 0.5 Stop bit and 1.5 Stop bit are not available for UART4 & UART5.

Bit 11 **CLKEN**: Clock enable
This bit allows the user to enable the CK pin.
0: CK pin disabled
1: CK pin enabled
This bit is not available for UART4 & UART5.

2. STM32 Serial Port Register
-- STM32 USART_CR2(continued)



Bit 10 **CPOL**: Clock polarity

This bit allows the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on CK pin outside transmission window.

1: Steady high value on CK pin outside transmission window.

This bit is not available for UART4 & UART5.

Bit 9 **CPHA**: Clock phase

This bit allows the user to select the phase of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see figures *289* to *290*)

0: The first clock transition is the first data capture edge.

1: The second clock transition is the first data capture edge.

This bit is not available for UART4 & UART5.

Bit 8 **LBCL**: Last bit clock pulse

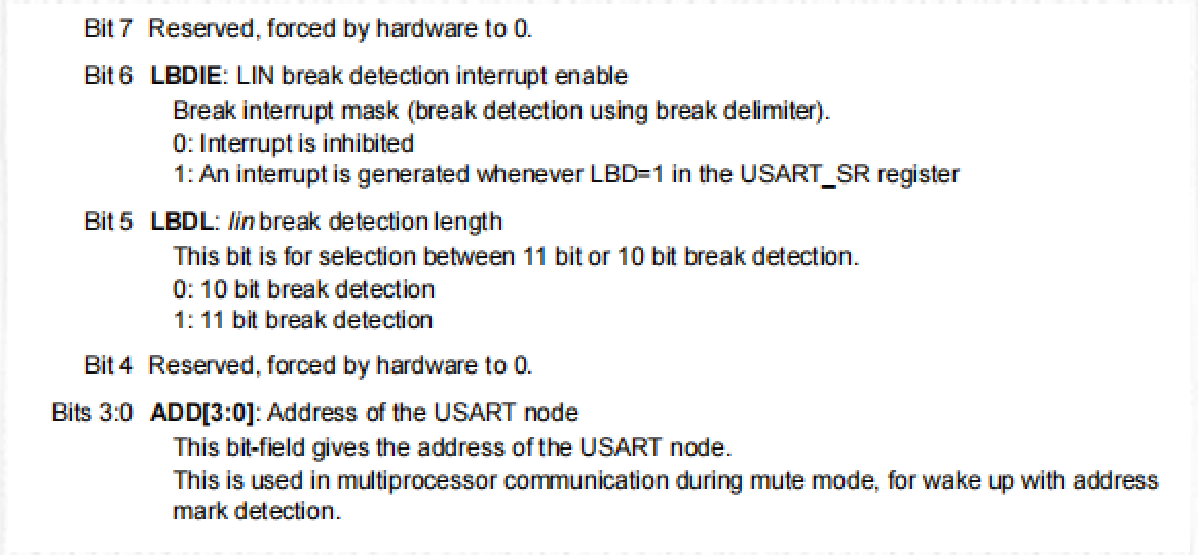
This bit allows the user to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the CK pin

1: The clock pulse of the last data bit is output to the CK pin

The last bit is the 8th or 9th data bit transmitted depending on the 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit is not available for UART4 & UART5.



Bit 7 Reserved, forced by hardware to 0.

Bit 6 **LBDIE**: LIN break detection interrupt enable

Break interrupt mask (break detection using break delimiter).

0: Interrupt is inhibited

1: An interrupt is generated whenever LBD=1 in the USART_SR register

Bit 5 **LBDL**: *lin* break detection length

This bit is for selection between 11 bit or 10 bit break detection.

0: 10 bit break detection

1: 11 bit break detection

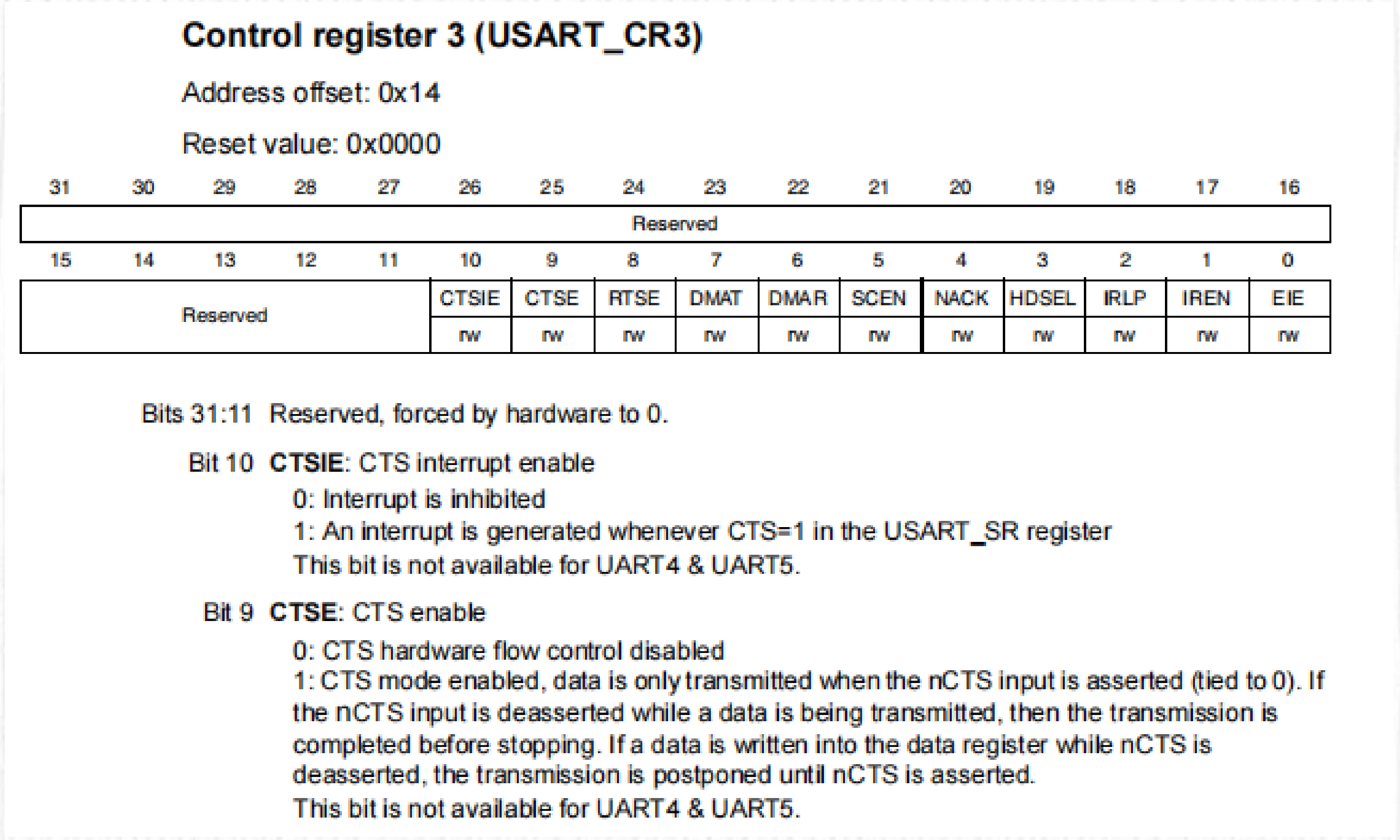
Bit 4 Reserved, forced by hardware to 0.

Bits 3:0 **ADD[3:0]**: Address of the USART node

This bit-field gives the address of the USART node.

This is used in multiprocessor communication during mute mode, for wake up with address mark detection.

2. STM32 Serial Port Register -- STM32 USART_CR3



Control register 3 (USART_CR3)

Address offset: 0x14

Reset value: 0x0000

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

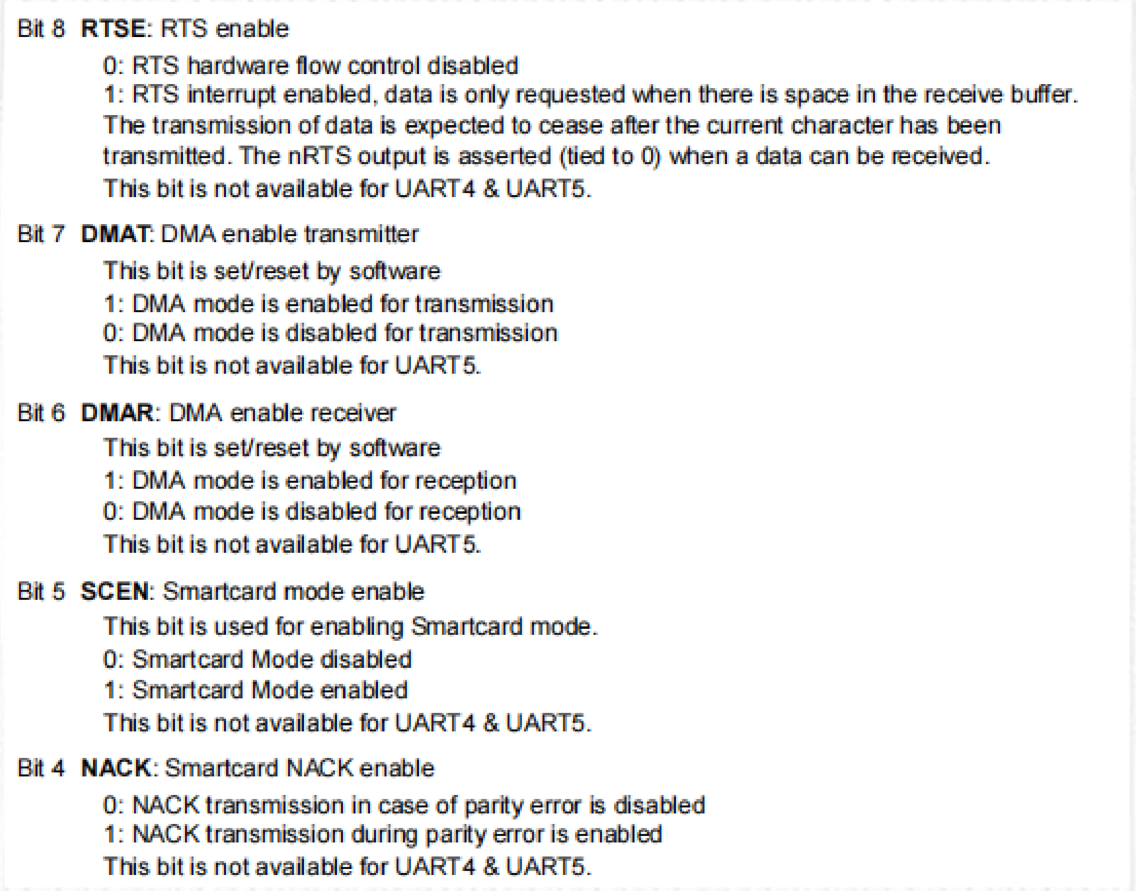
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, forced by hardware to 0.

Bit 10 **CTSIE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTS=1 in the USART_SR register
This bit is not available for UART4 & UART5.

Bit 9 **CTSE**: CTS enable
0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0). If the nCTS input is deasserted while a data is being transmitted, then the transmission is completed before stopping. If a data is written into the data register while nCTS is deasserted, the transmission is postponed until nCTS is asserted.
This bit is not available for UART4 & UART5.

2. STM32 Serial Port Register -- STM32 USART_CR3(continued)



Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled
1: RTS interrupt enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (tied to 0) when a data can be received.
This bit is not available for UART4 & UART5.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission
This bit is not available for UART5.

Bit 6 **DMAR**: DMA enable receiver

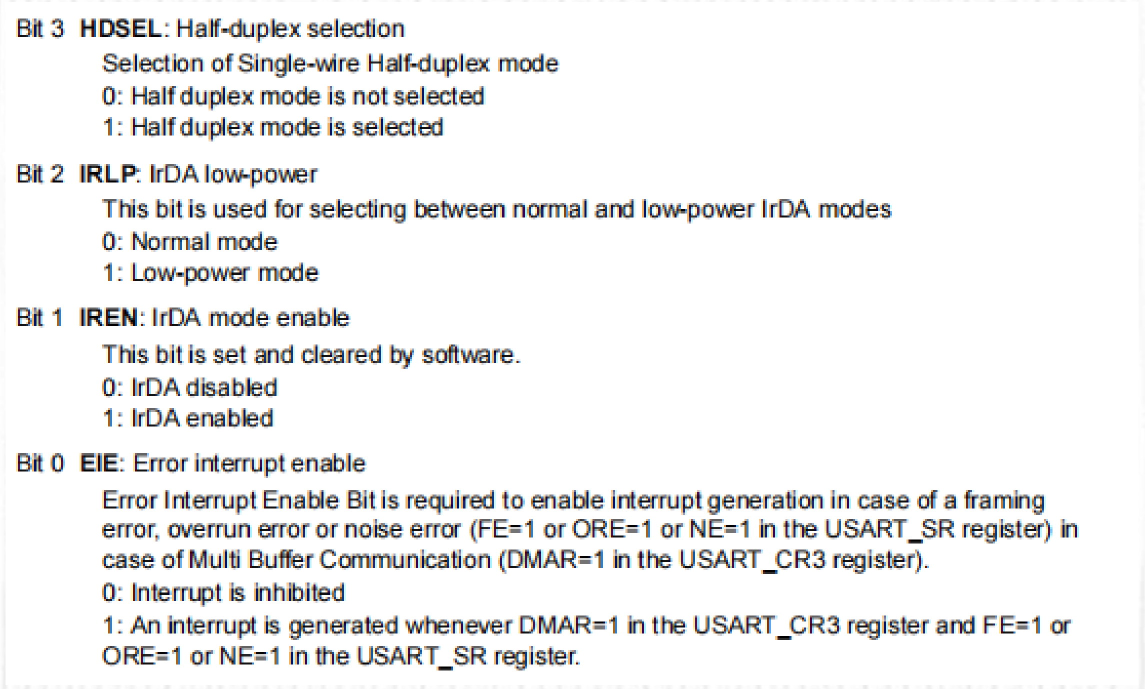
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception
This bit is not available for UART5.

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling Smartcard mode.
0: Smartcard Mode disabled
1: Smartcard Mode enabled
This bit is not available for UART4 & UART5.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled
1: NACK transmission during parity error is enabled
This bit is not available for UART4 & UART5.



Bit 3 **HDSEL**: Half-duplex selection

Selection of Single-wire Half-duplex mode
0: Half duplex mode is not selected
1: Half duplex mode is selected

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes
0: Normal mode
1: Low-power mode

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.
0: IrDA disabled
1: IrDA enabled

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise error (FE=1 or ORE=1 or NE=1 in the USART_SR register) in case of Multi Buffer Communication (DMAR=1 in the USART_CR3 register).
0: Interrupt is inhibited
1: An interrupt is generated whenever DMAR=1 in the USART_CR3 register and FE=1 or ORE=1 or NE=1 in the USART_SR register.

2. STM32 Serial Port Register -- Serial Port Configuration Steps

- Enable serial port clock and GPIO clock
- Reset serial port (this step is not necessary)
- GPIO port mode setting
- Initialize Serial port parameters
- Enable interrupt and initialize NVIC
- Enable serial port
- Write interrupt handling functions
- Serial port data transmission and sender
- Serial port transmission status acquisition

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

03

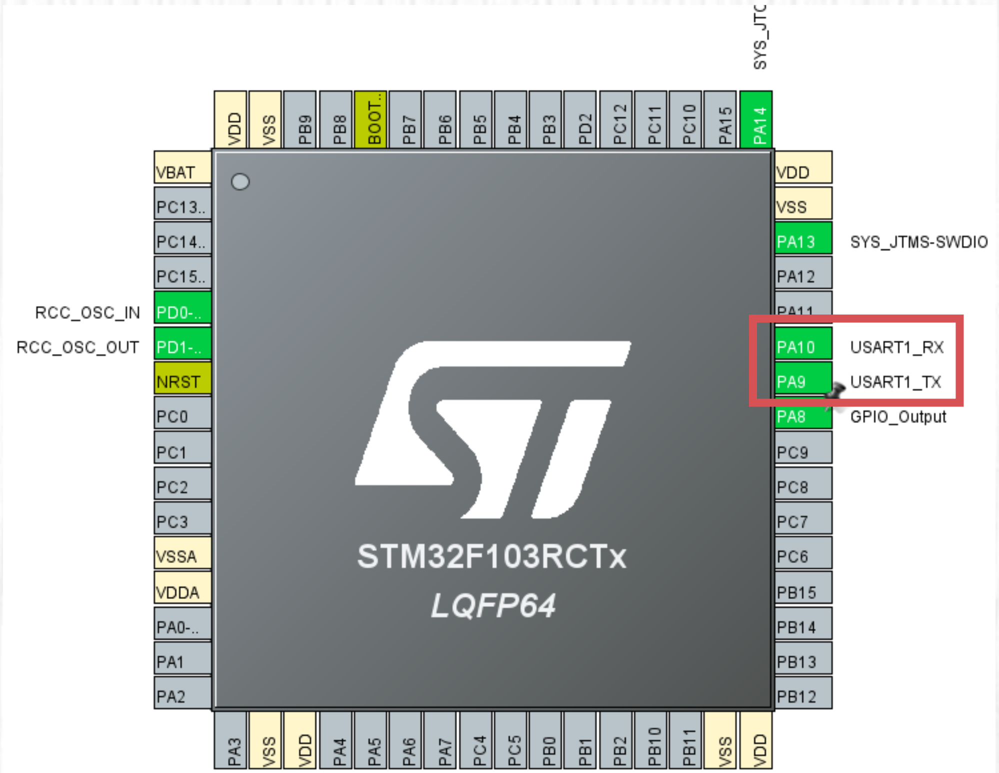
How to Program

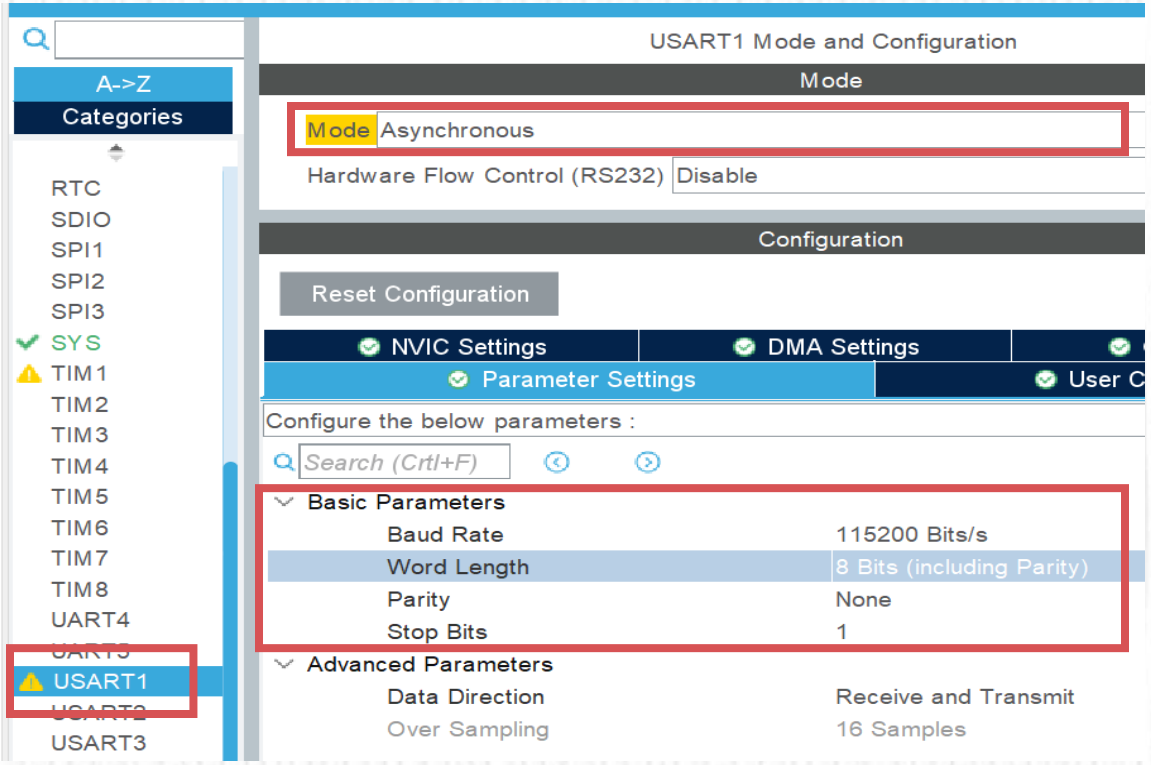
3. How to Program

- Our Goal
  - Make MiniSTM32 board communicate with the computer through the serial port.
  - PC sends a message to MiniSTM32 board, and then MiniSTM32 board sends the same message back to PC.

3. How to Program

- Configure USART in STM32CubeIDE
  - Choose any USART/UART and set its mode as **asynchronous**.
  - Two UARTs only work well when these four parameter are the same: baud rate, word length, parity and stop bits.





USART1 Mode and Configuration

Mode

Mode Asynchronous

Hardware Flow Control (RS232) Disable

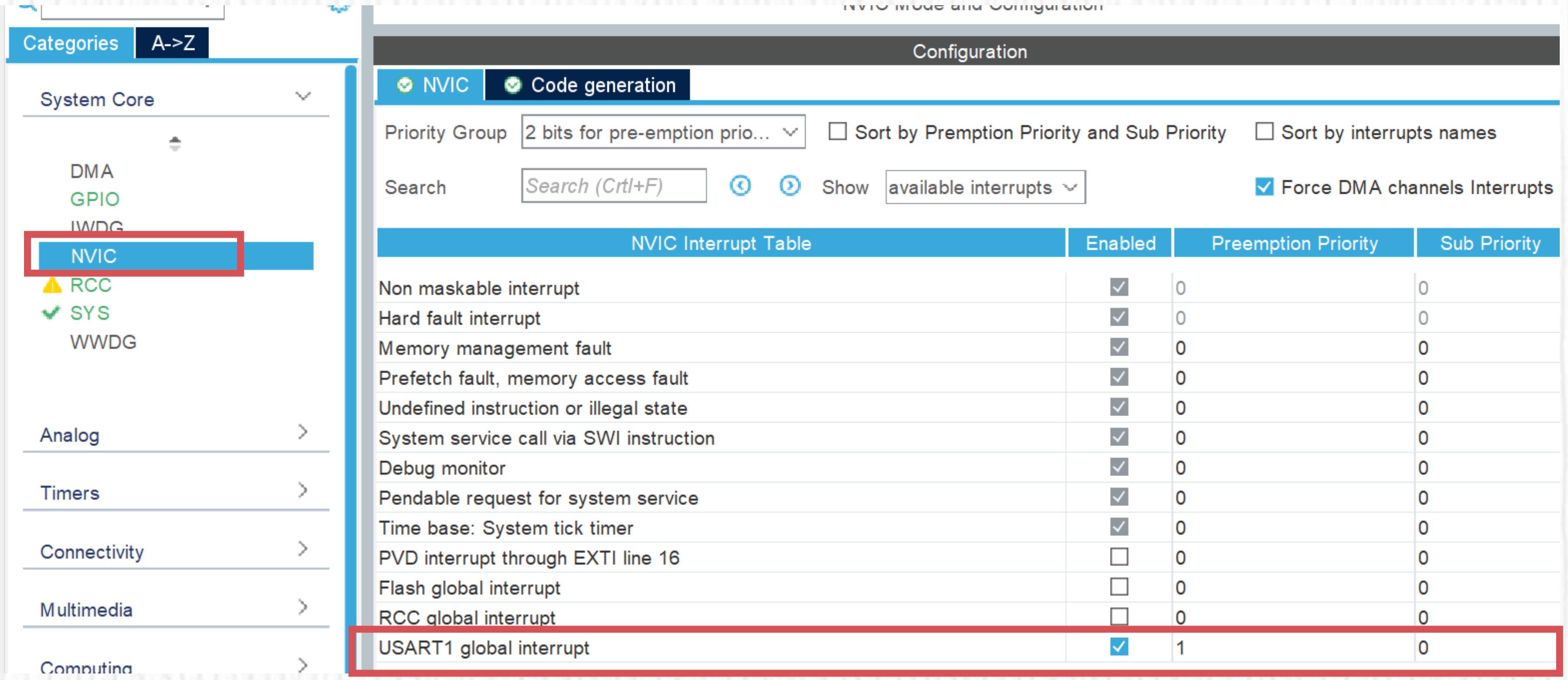
Configuration

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

3. How to Program

- Receive Data in interruption Mode
  - In most case, we transmit data in **blocking mode**, but receive the data by **interruption**, so we should enable the interruption for the corresponding USART.

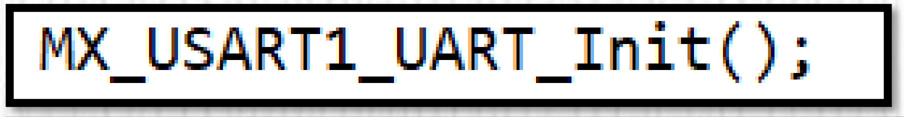


Priority Group: 2 bits for pre-emption prio...
☐ Sort by Premption Priority and Sub Priority
☐ Sort by interrupts names
Search: Search (Crtl+F)
Show: available interrupts
✓ Force DMA channels Interrupts

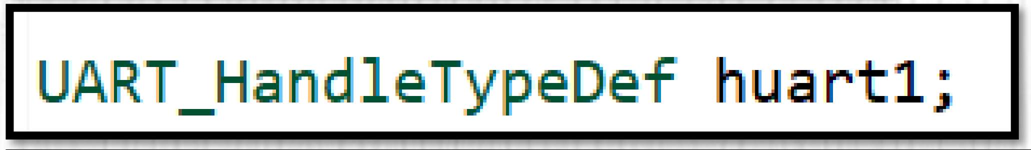
NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	✓	0	0
Hard fault interrupt	✓	0	0
Memory management fault	✓	0	0
Prefetch fault, memory access fault	✓	0	0
Undefined instruction or illegal state	✓	0	0
System service call via SWI instruction	✓	0	0
Debug monitor	✓	0	0
Pendable request for system service	✓	0	0
Time base: System tick timer	✓	0	0
PVD interrupt through EXTI line 16	☐	0	0
Flash global interrupt	☐	0	0
RCC global interrupt	☐	0	0
USART1 global interrupt	✓	1	0

3. How to Program

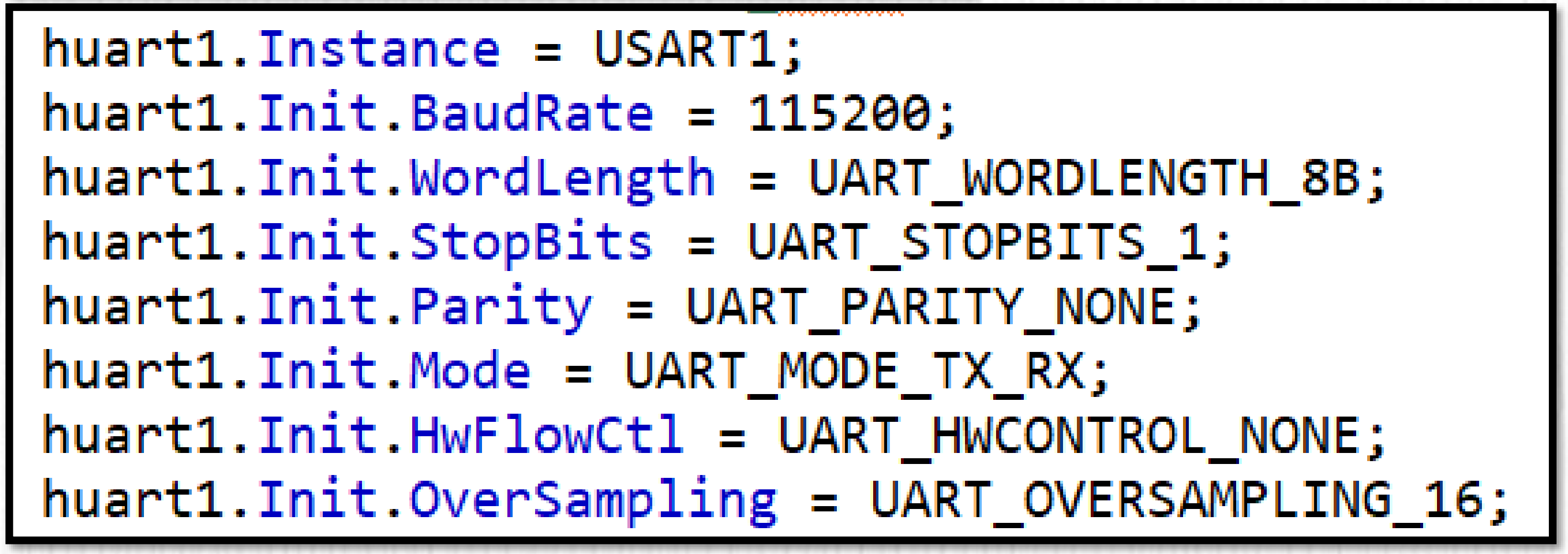
- Some structures and functions we used



```c
MX_USART1_UART_Init();
```



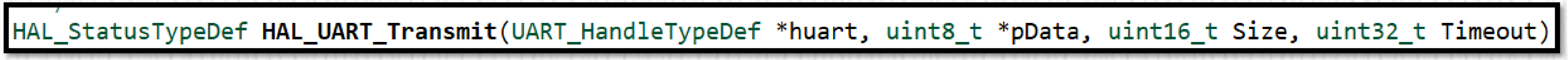
```c
UART_HandleTypeDef huart1;
```



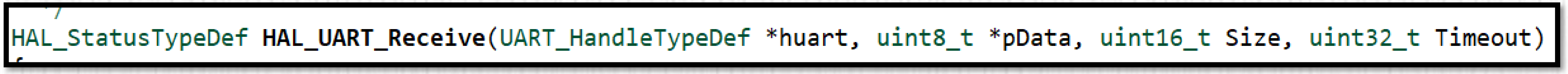
```c
huart1.Instance = USART1;
huart1.Init.BaudRate = 115200;
huart1.Init.WordLength = UART_WORDLENGTH_8B;
huart1.Init.StopBits = UART_STOPBITS_1;
huart1.Init.Parity = UART_PARITY_NONE;
huart1.Init.Mode = UART_MODE_TX_RX;
huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart1.Init.OverSampling = UART_OVERSAMPLING_16;
```

3. How to Program

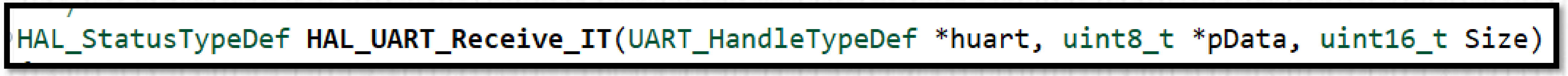
- Some structures and functions we used
  - HAL_UART_Transmit() and HAL_UART_Receive() functions are used to transmit/receive data in blocking mode. Which means, unless transmit/receive is completed or timeout, the programs will stop here.
  - But in most case, we transmit data in **blocking mode**, but receive the data by **interruption**, so we use HAL_UART_Receive_IT() function instead**.**
  - These three functions are in stm32f1xx_hal_uart.c



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



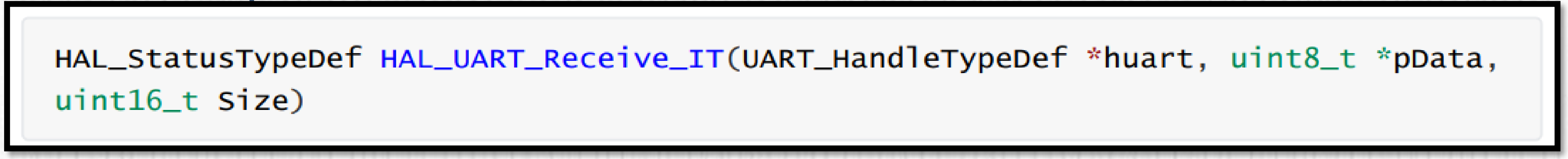
```
HAL_StatusTypeDef HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)
```



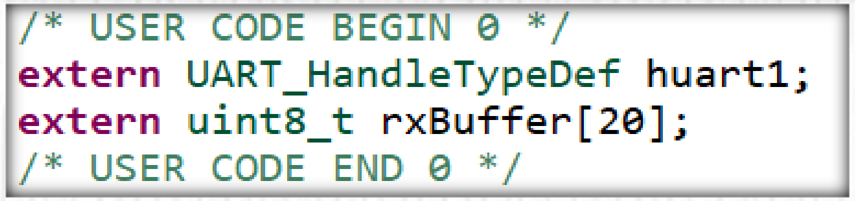
```
HAL_StatusTypeDef HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)
```

3. How to Program

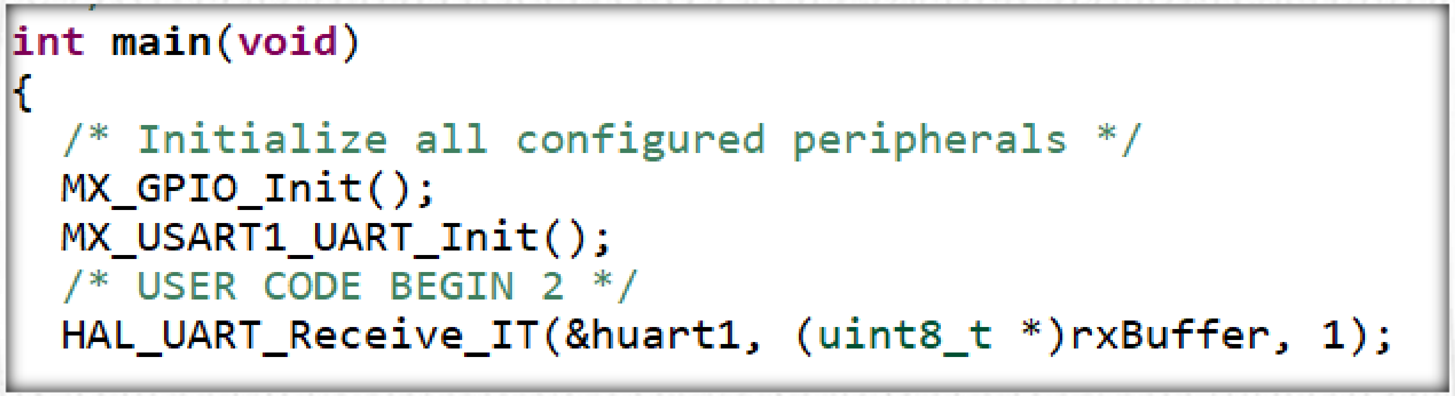
- Some structures and functions we used
  - In main.c, we need to call HAL_UART_Receive_IT() function, which can receive data in non-blocking mode and trigger an interruption.



```c
HAL_StatusTypeDef HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData,
uint16_t Size)
```



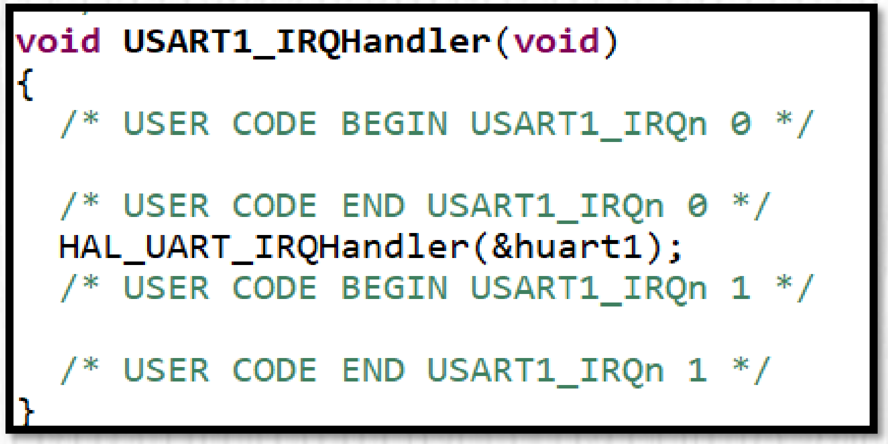
```c
/* USER CODE BEGIN 0 */
extern UART_HandleTypeDef huart1;
extern uint8_t rxBuffer[20];
/* USER CODE END 0 */
```



```c
int main(void)
{
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART1_UART_Init();
  /* USER CODE BEGIN 2 */
  HAL_UART_Receive_IT(&huart1, (uint8_t *)rxBuffer, 1);
```

3. How to Program

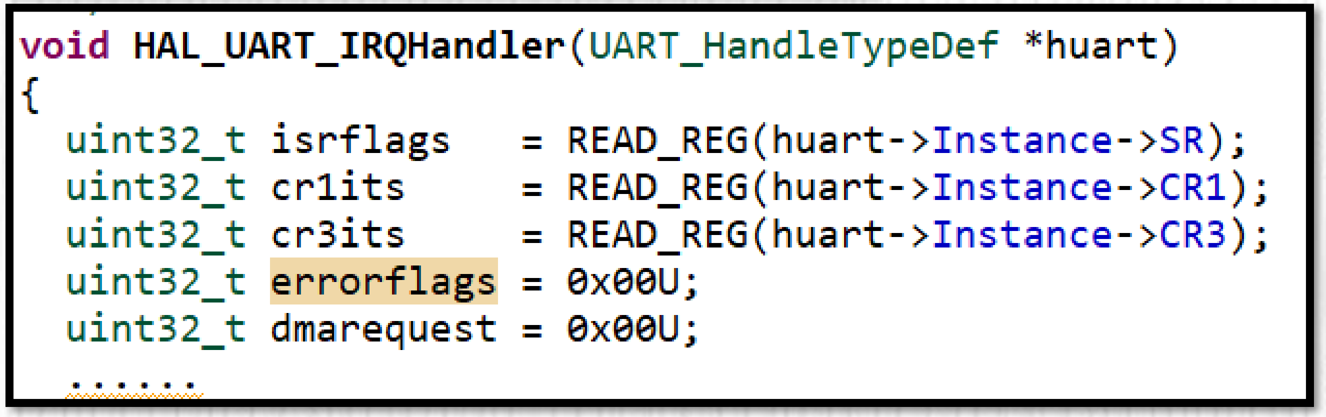
- Some structures and functions we used
  - When there is a interruption of UART coming, the program will go into function USART1_IRQHandler(), it is in file stm32f1xx_it.c



```
void USART1_IRQHandler(void)
{
  /* USER CODE BEGIN USART1_IRQn 0 */

  /* USER CODE END USART1_IRQn 0 */
  HAL_UART_IRQHandler(&huart1);
  /* USER CODE BEGIN USART1_IRQn 1 */

  /* USER CODE END USART1_IRQn 1 */
}
```



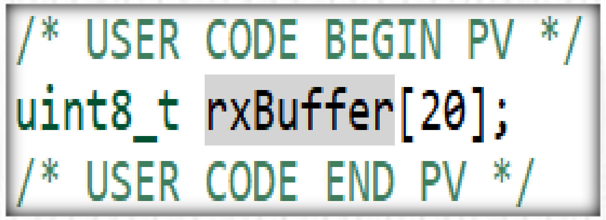
```
void HAL_UART_IRQHandler(UART_HandleTypeDef *huart)
{
  uint32_t isrflags   = READ_REG(huart->Instance->SR);
  uint32_t cr1its     = READ_REG(huart->Instance->CR1);
  uint32_t cr3its     = READ_REG(huart->Instance->CR3);
  uint32_t errorflags = 0x00U;
  uint32_t dmarequest = 0x00U;
  ......
```

3. How to Program

- Some structures and functions we used
  - When receive interruption is triggered, the program will call the function UART_Receive_IT() , which stores the character into the buffer pRxBuffPtr and call the receive complete callback function HAL_UART_RxCpltCallback() when it receive RxXferCount amounts of characters.
  - So we can re-implement the HAL_UART_RxCpltCallback() function if we want to receive data in non-blocking mode. For example, we can store the data in an array and sent it out when receive a line.

3. How to Program

- Re-implement HAL_UART_RxCpltCallback() in stm32f1xx_it.c, and **uint8_t rxBuffer[20]** is also defined in this file.

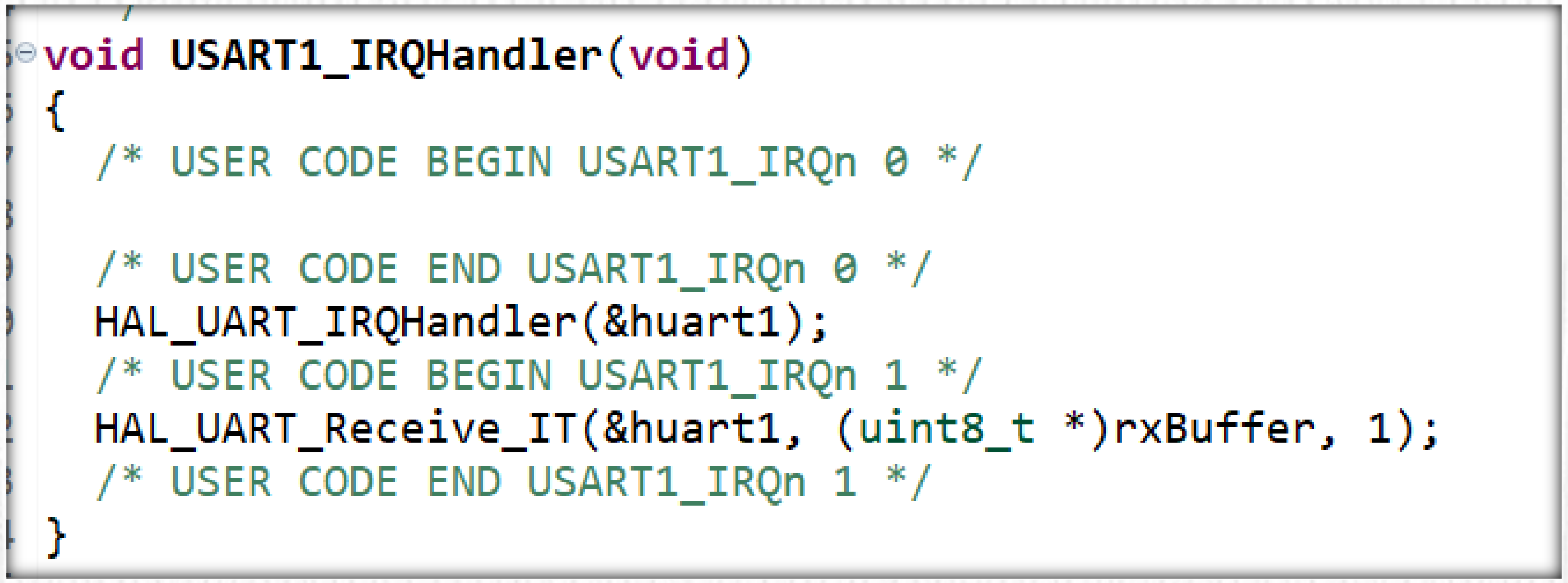


```
/* USER CODE BEGIN PV */
uint8_t rxBuffer[20];
/* USER CODE END PV */
```

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
        if(huart->Instance==USART1){
                static unsigned char uRx_Data[1024] = {0};
                static unsigned char uLength = 0;
                if(rxBuffer[0] == '\n'){
                        HAL_UART_Transmit(&huart1, uRx_Data, uLength, 0xffff);
                        uLength = 0;
                }else{
                        uRx_Data[uLength] = rxBuffer[0];
                        uLength++;
                }
        }
}
```

3. How to Program

- Re-call HAL_UART_Receive_IT() to receive data continuously in USART1_IRQHandler() function.

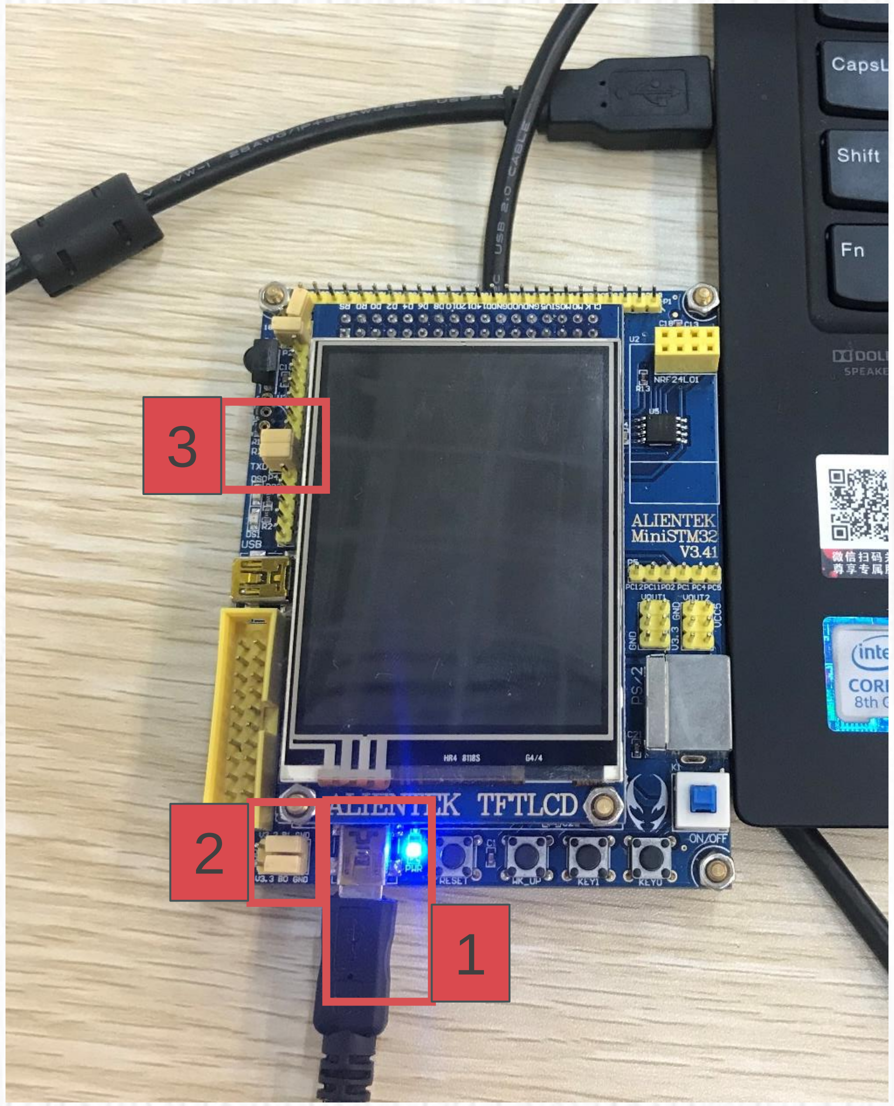


```
void USART1_IRQHandler(void)
{
  /* USER CODE BEGIN USART1_IRQn 0 */

  /* USER CODE END USART1_IRQn 0 */
  HAL_UART_IRQHandler(&huart1);
  /* USER CODE BEGIN USART1_IRQn 1 */
  HAL_UART_Receive_IT(&huart1, (uint8_t *)rxBuffer, 1);
  /* USER CODE END USART1_IRQn 1 */
}
```

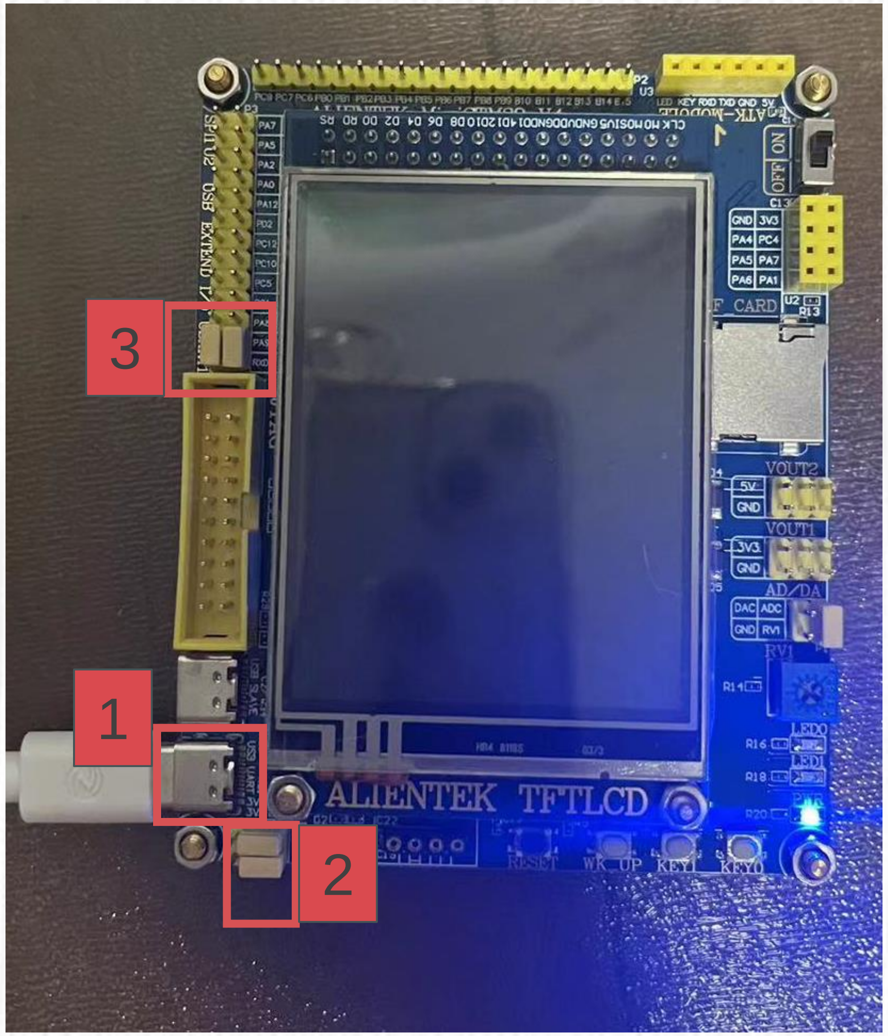
3. How to Program

- Communicate with PC - hardware wire connection
  - 1: use USB_232 port
  - 2: both BOOT0 and BOOT1 connect to GND
  - 3: if choose USART1, connect PA9 and PA10 to TXD and RXD



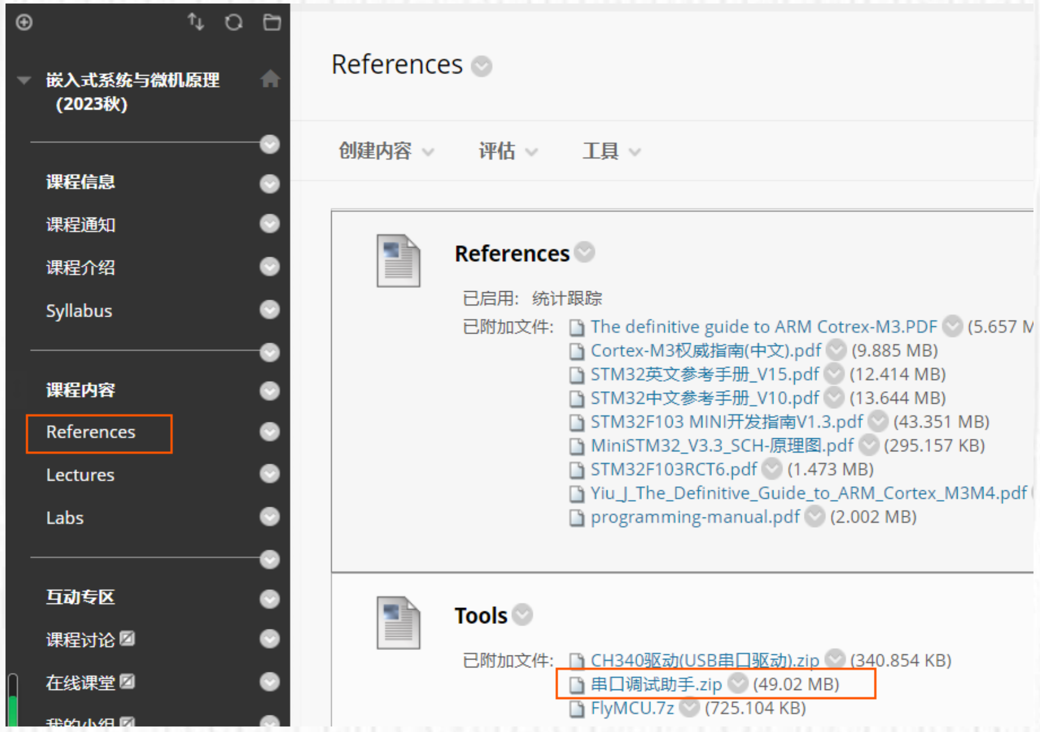
3. How to Program

- Communicate with PC - hardware wire connection (V4)
  - 1: use USB_UART port
  - 2: both BOOT0 and BOOT1 connect to GND
  - 3: if choose USART1, connect PA9 and PA10 to TXD and RXD



3. How to Program

- Communicate with PC – PC serial port configuration



References

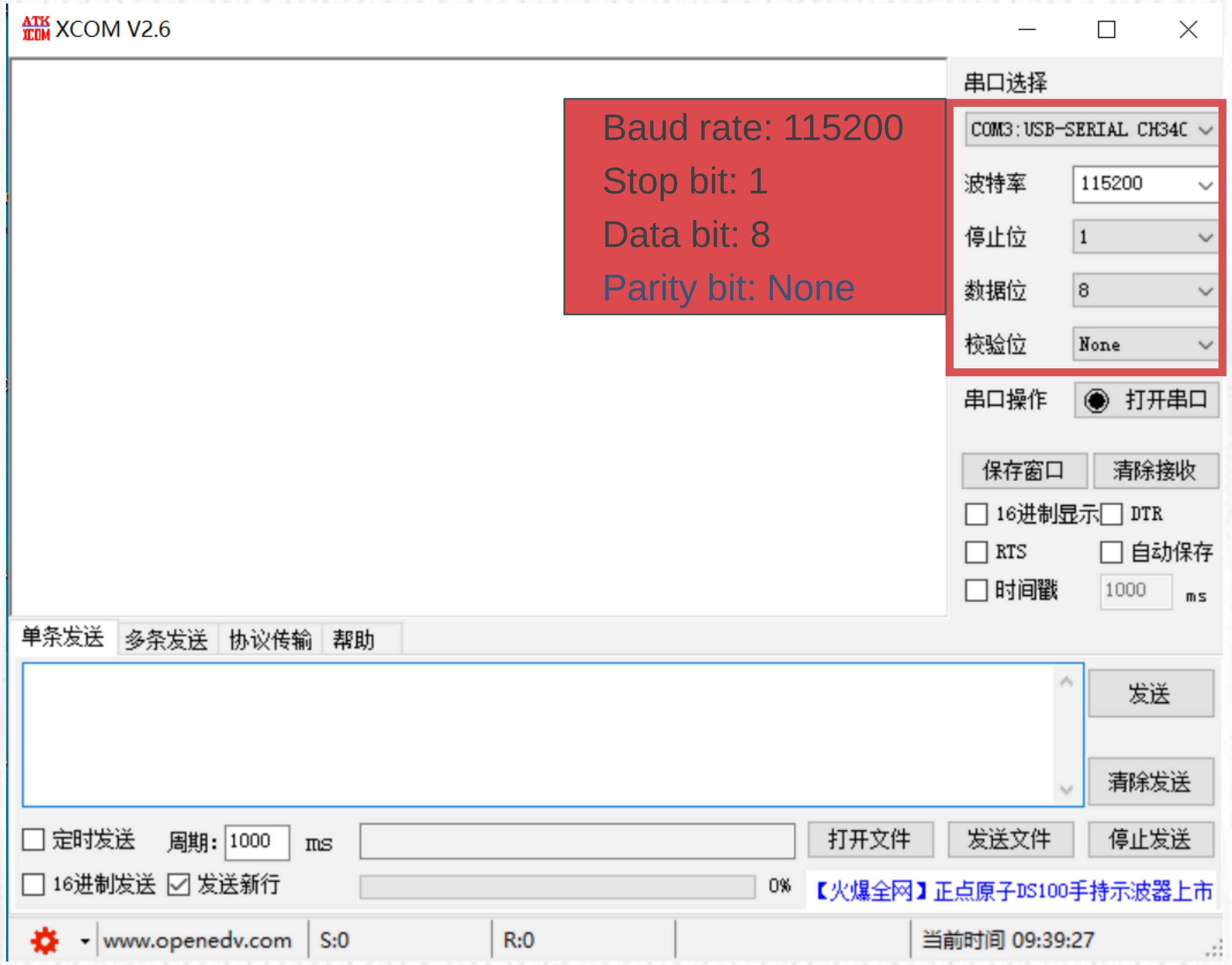
References

已启用: 统计跟踪

已附加文件:
- The definitive guide to ARM Cotrex-M3.PDF (5.657 M
- Cortex-M3权威指南(中文).pdf (9.885 MB)
- STM32英文参考手册_V15.pdf (12.414 MB)
- STM32中文参考手册_V10.pdf (13.644 MB)
- STM32F103 MINI开发指南V1.3.pdf (43.351 MB)
- MiniSTM32_V3.3_SCH-原理图.pdf (295.157 KB)
- STM32F103RCT6.pdf (1.473 MB)
- Yiu_J_The_Definitive_Guide_to_ARM_Cortex_M3M4.pdf
- programming-manual.pdf (2.002 MB)

Tools

已附加文件:
- CH340驱动(USB串口驱动).zip (340.854 KB)
- 串口调试助手.zip (49.02 MB)
- FlyMCU.7z (725.104 KB)

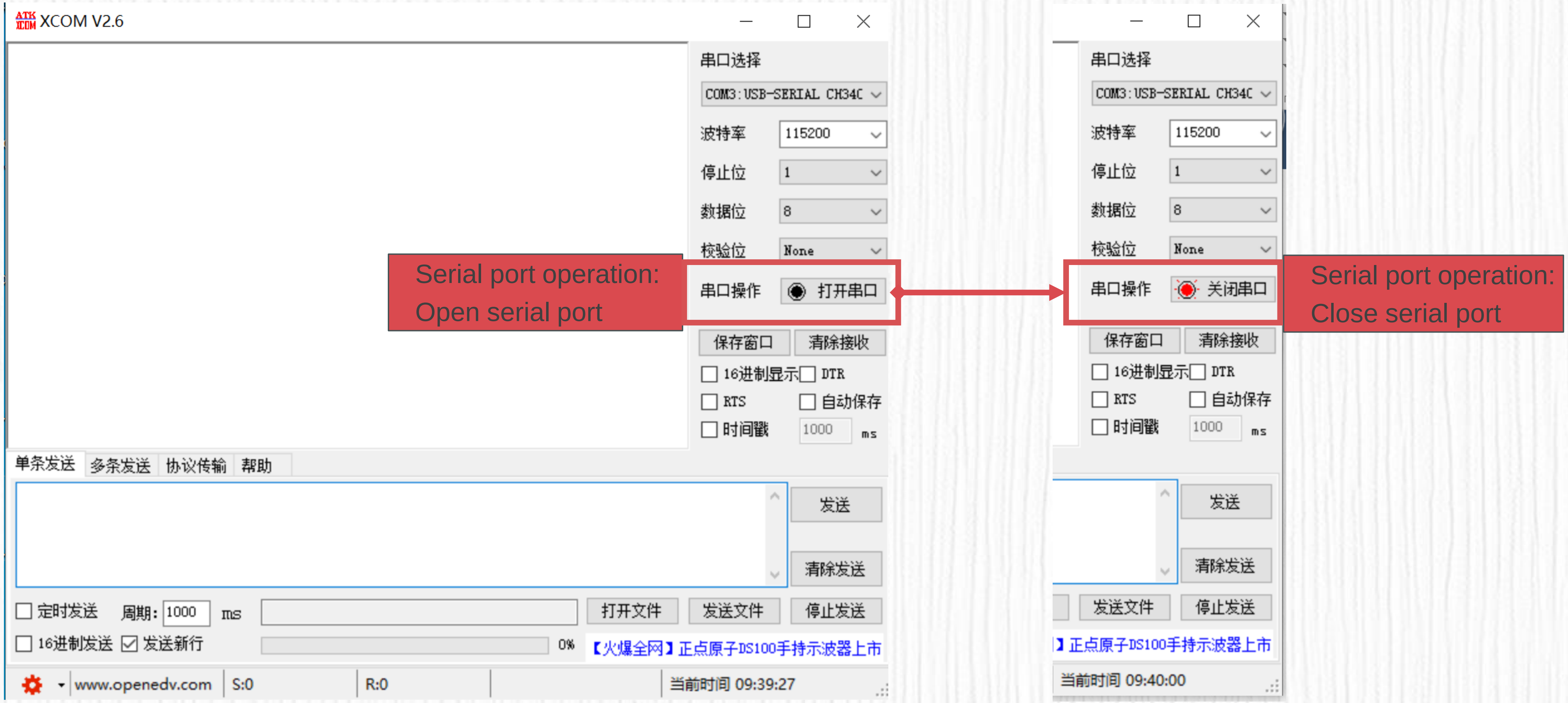


Baud rate: 115200
Stop bit: 1
Data bit: 8
Parity bit: None

串口选择: COM3: USB-SERIAL CH340
波特率: 115200
停止位: 1
数据位: 8
校验位: None

3. How to Program

- Communicate with PC – open serial port of PC


XCOM V2.6

串口选择: COM3:USB-SERIAL CH34C
波特率: 115200
停止位: 1
数据位: 8
校验位: None

Serial port operation: Open serial port

串口操作 ● 打开串口

Serial port operation: Close serial port

串口操作 ● 关闭串口

☐ 16进制显示 ☐ DTR
☐ RTS ☐ 自动保存
☐ 时间戳 1000 ms

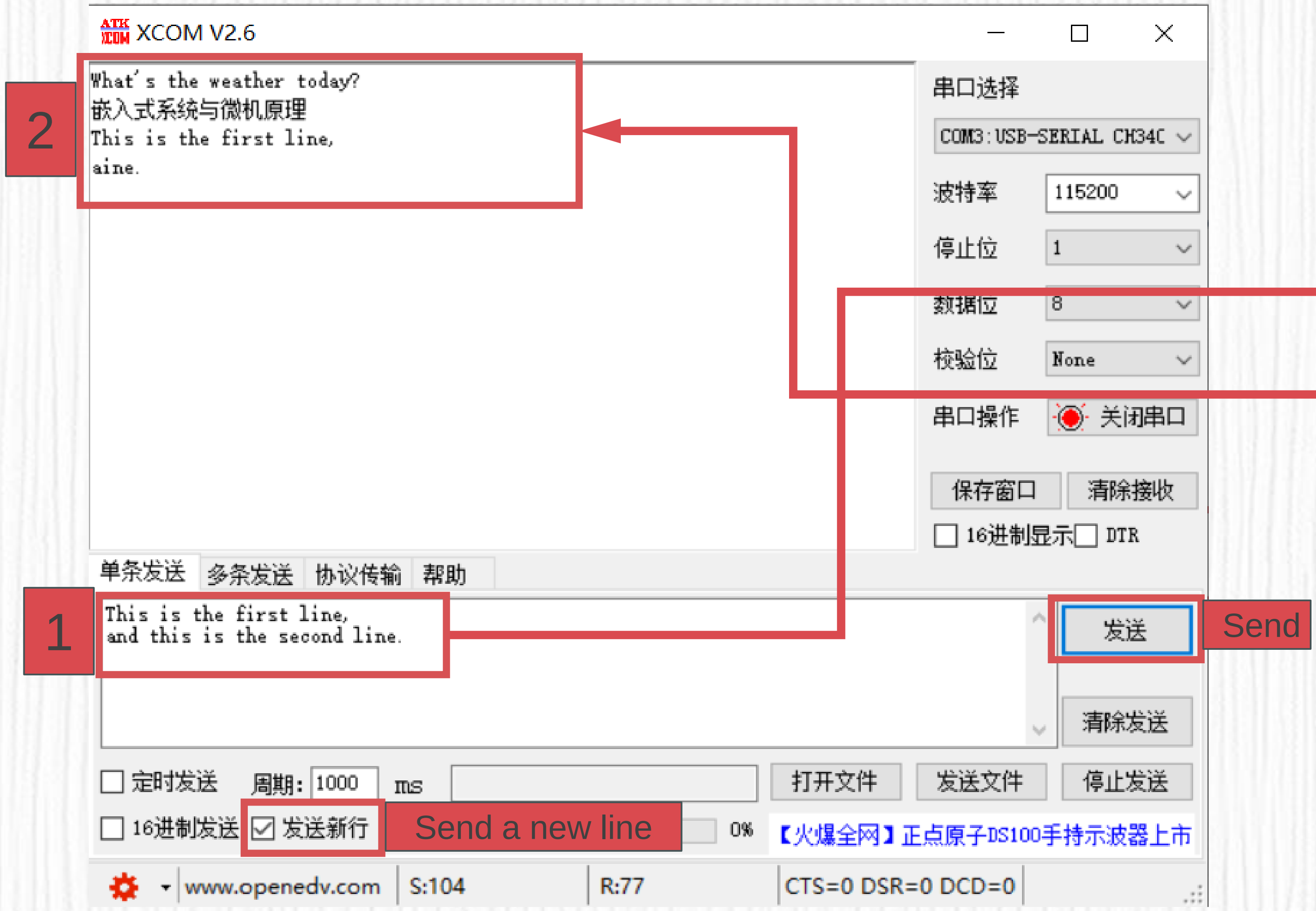
☐ 定时发送 周期: 1000 ms
☐ 16进制发送 ☑ 发送新行
0%
【火爆全网】正点原子DS100手持示波器上市

www.openedv.com S:0 R:0 当前时间 09:39:27

当前时间 09:40:00

3. How to Program

- Communicate with PC – send messages



XCOM V2.6

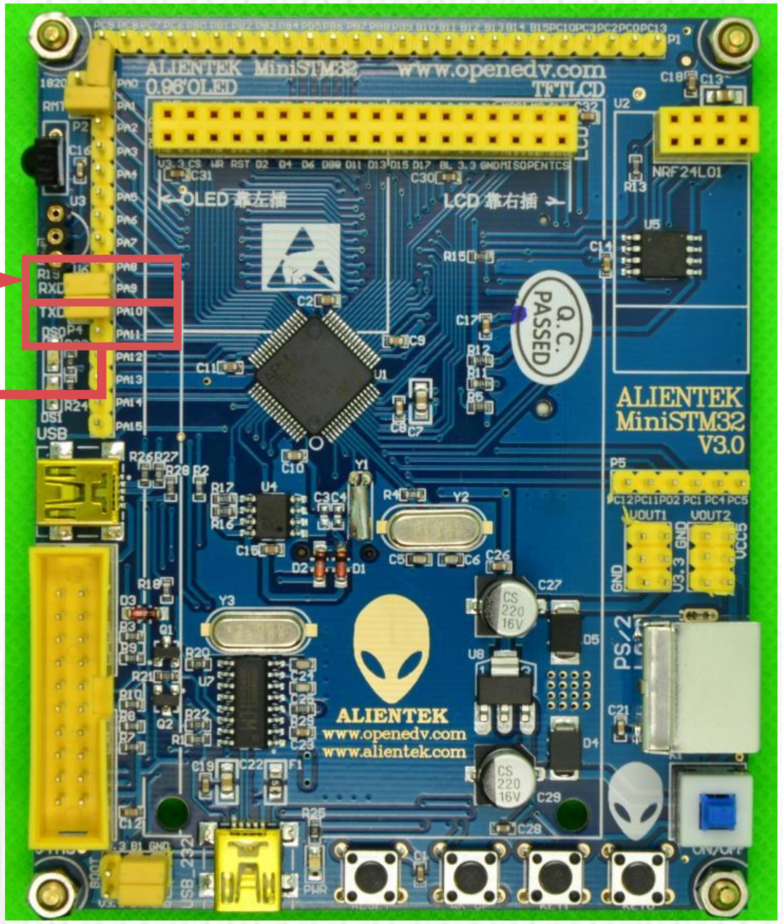
```
What's the weather today?
嵌入式系统与微机原理
This is the first line,
aine.
```

串口选择: COM3:USB-SERIAL CH34C
波特率: 115200
停止位: 1
数据位: 8
校验位: None
串口操作: 关闭串口

```
This is the first line,
and this is the second line.
```

☐ 定时发送 周期: 1000 ms
☐ 16进制发送 ☑ 发送新行

S:104 R:77



4. Practice

- Run the demo on MiniSTM32 board.
- Design a simple communication system.
  - When sending a message from PC to MiniSTM32 board, show this message on LCD.
  - When KEY0 is pressed, send a message from MiniSTM32 board to PC, and show the string “KEY0 is pressed” on PC.
  - When KEY1 is pressed, send a message from MiniSTM32 board to PC, and show the string “KEY1 is pressed” on PC.