Laboratory work No. 3

**Goal of research:**

Gain skills in working with USART, exchanging data with a computer.

**Software:**

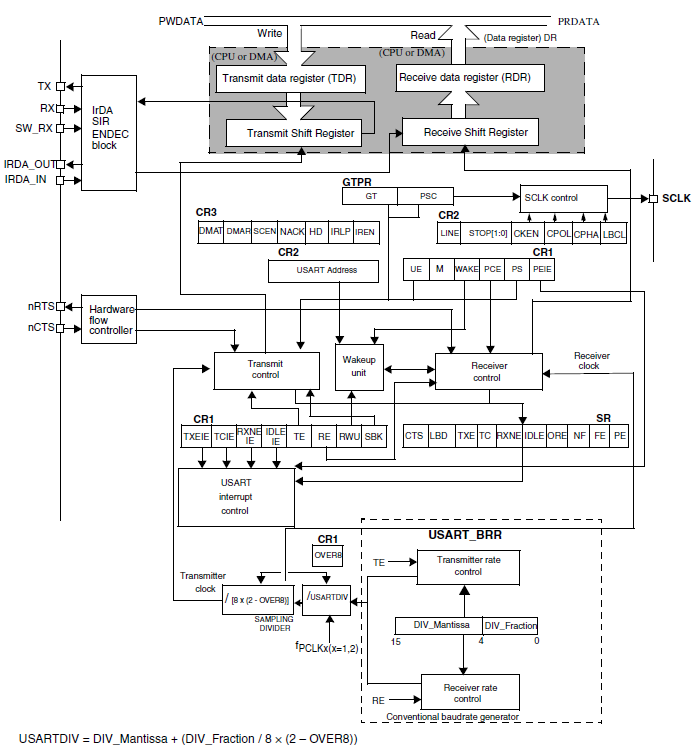
STM32CubeIDE, Terminal.

**General information:**

Universal Asynchronous Receiver-Transmitter (UART) is a node of computing devices designed to organize communication with other digital devices. It converts the transmitted data into serial form so that it is possible to transmit them over a physical digital line to another similar device. The conversion method is well standardized and widely used in computer technology (especially in embedded devices and systems-on-chip (SoC)).

It is a logic circuit with one side connected to the bus of a computing device and the other side having two or more pins for external connection.

The UART can be a separate chip (e.g. Intel I8251, I8250) or part of a larger integrated circuit (e.g. microcontroller). It is used to transfer data through the serial port of a computer.



Data between devices is usually transmitted in packets of a certain fixed length. The length of the receiving and transmitting packet can be different. For example, a data packet transmitted by a humidity and temperature sensor:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BYTE COUNT** | **HEX** | **DECIMAL** | **CHARACTER DISPLAYED** | **Details** |
| 1 | 0x0A | 10 | ‘\n’ = LF | New Line Character |
| 2 | 0x48 | 72 | ‘H’ | Fixed Character H |
| 3 | 0x3A | 58 | ‘:’ | Fixed Character : |
| 4 | 0x30 | 48 | ‘0’ | Humidity Character-Hundreds |
| 5 | 0x35 | 53 | ‘5’ | Humidity Character-Tens |
| 6 | 0x38 | 56 | ‘8’ | Humidity Character-Ones |
| 7 | 0x20 | 32 | ‘ ‘ | Space |
| 8 | 0x54 | 84 | ‘T’ | Fixed Character T |
| 9 | 0x3A | 58 | ‘:’ | Fixed Character : |
| 10 | 0x30 | 48 | ‘0’ | Temperature Character-Hundreds |
| 11 | 0x32 | 50 | ‘2’ | Temperature Character-Tens |
| 12 | 0x34 | 52 | ‘4’ | Temperature Character-Ones |
| 13 | 0x0D | 13 | ‘\r’ = CR | New Line Character |

In the above example, bytes 1 and 13 are the beginning and end of the message.

*Algorithm of data transmission from the controller side using interrupts:*

1. The buffer for sending is formed and the length of the array to be sent is set.
2. An interrupt is triggered on the empty data sending register, and the counter of the sent data is reset to zero.
3. In the interrupt, on the event written in the previous step, the data byte from the array is written to the USART data register, the value of the counter of the sent data is increased.
4. After sending all required bytes from the buffer, the interrupt on the empty data sending register is disabled.

*The algorithm for receiving data from the controller side using interrupts* is similar to the algorithm for receiving data transmission with a few changes:

* Interrupt on non-empty register of received data is always enabled.
* A fixed number of bytes is always received.
* The received data counter is reset to zero when the end-of-packet character is received.

*HAL functions for UART operation*

Data transfer using interrupts:

*HAL\_UART\_Transmit\_IT (UART\_HandleTypeDef \* huart, uint8\_t \* pData, uint16\_t Size)*

* *huart* – the pointer to the configuration structure of type UART\_HandleTypeDef.
* *pData* – the pointer to the buffer of data to be transferred.
* *Size* – the amount of data to be transferred.

Data reception using an interrupt:

*HAL\_UART\_Receive\_IT (UART\_HandleTypeDef \* huart, uint8\_t \* pData, uint16\_t Size)*

* *huart* – the pointer to the configuration structure of UART\_HandleTypeDef type.
* *pData* – the pointer to the buffer for the received data.
* *Size* – the amount of data to be accepted.

Callback function for receiving data:

*void HAL\_UART\_RxCpltCallback(UART\_HandleTypeDef \*huart)*

* *huart* – the pointer to the configuration structure of type

**The order of work:**

*Part I. Developing a program using a code generator.*

1. Start STM32CubeIDE, in the opened window choose the path to your working folder. There should be no Russian letters in the path to the working folder and the project name. In this folder should be stored all laboratory works.

2. Based on the circuit diagram, determine which UART is used to send data to the computer.

3. In the window of graphical initialization of the controller (name.IOC), set the UART, defined in the previous step, asynchronous mode of operation. Set the baud rate according to the option. Enable global interrupts from the UART.

4. In the main file add callback-function:

/\* USER CODE BEGIN 4 \*/

void HAL\_UART\_RxCpltCallback(UART\_HandleTypeDef \*huart) {

}

/\* USER CODE END 4 \*/

In the main function in the appropriate area, start receiving data via UART using an interrupt.

5. Develop a protocol for data exchange with the computer. The data packet from the computer should contain bytes identifiers of the beginning and end of the message, 2 bytes of number. Example:

N25E - packet 4 bytes; N,E - bytes identifiers of the beginning and end of the message respectively; 25 - decimal number written in 2 bytes, "2" in one byte "5" in the other. The data packet to the computer must be formed in the same way.

To receive and transmit data to the computer, you must use the program Terminal, or similar. Note that this program sends data in the form of ACSII codes:

Изображение выглядит как текст, стол

Автоматически созданное описание

The packet "N25E" in the sending string will be converted to an array [78;50;53;69].

6. Implement the following program algorithm:

- Receive a data packet from the computer;

- Extract a number from the packet;

- Perform a mathematical operation on this number;

- Display the result on the 7-segment indicator;

- Generate a packet to send the result to the computer;

- Send the data packet to the computer.

You can use the *sprintf* function to easily generate a packet of data to send.

*Part II. Program development using registers.*

1. Start STM32CubeIDE, in the opened window choose the path to your working folder. There should be no Russian letters in the path to the working folder and the project name. In this folder all laboratory works should be stored.

2. Based on the circuit diagram, determine which UART is used to send data to the computer.

3. Initialize the UART to receive and transmit data, enable interrupts to receive data. When setting the register BRR, take the frequency of the bus, which is connected to the UART, equal to 16MHz. Enable interrupt handler from the UART in the interrupt controller. Create a function for interrupt handler from the UART.

4. Implement the algorithms for receiving and transmitting data described in the "General Information" to the laboratory work in the interrupt handler.

5. Develop a protocol for data exchange with the computer. The data packet from the computer should contain bytes identifiers of the beginning and end of the messsage, 2 bytes of number. Example:

N25E - packet 4 bytes; N,E - bytes identifiers of the beginning and end of the message respectively; 25 - decimal number written in 2 bytes, "2" in one byte "5" in the other. The data packet to the computer must be formed in the same way.

To receive and transmit data to the computer, you must use the program Terminal, or similar. Note that this program sends data in the form of ACSII codes:

Изображение выглядит как текст, стол

Автоматически созданное описание

The packet "N25E" in the send string will be converted to an array [78;50;53;69].

6. Implement the following program algorithm:

- Receive a data packet from the computer;

- Extract a number from the packet;

- Perform a mathematical operation on this number;

- Display the result on the 7-segment indicator;

- Generate a packet to send the result to the computer;

- Send the data packet to the computer.

You can use the *sprintf* function to easily generate a packet of data to send.

**Tasks:**

1. Perform all of the steps in Part I.

2. If possible, complete all activities in Part II.

Demonstrate all assignment items one by one to the instructor.

**Variants:**

|  |  |  |
| --- | --- | --- |
| Variant no. | Baudrate | Function |
| 1 | 9600 |  |
| 2 | 19200 |  |
| 3 | 38400 |  |
| 4 | 57600 |  |
| 5 | 115200 |  |
| 6 | 9600 |  |
| 7 | 19200 |  |
| 8 | 38400 |  |
| 9 | 57600 |  |
| 10 | 115200 |  |
| 11 | 9600 |  |
| 12 | 19200 |  |