Laboratory work No. 4

**Goal of research:**

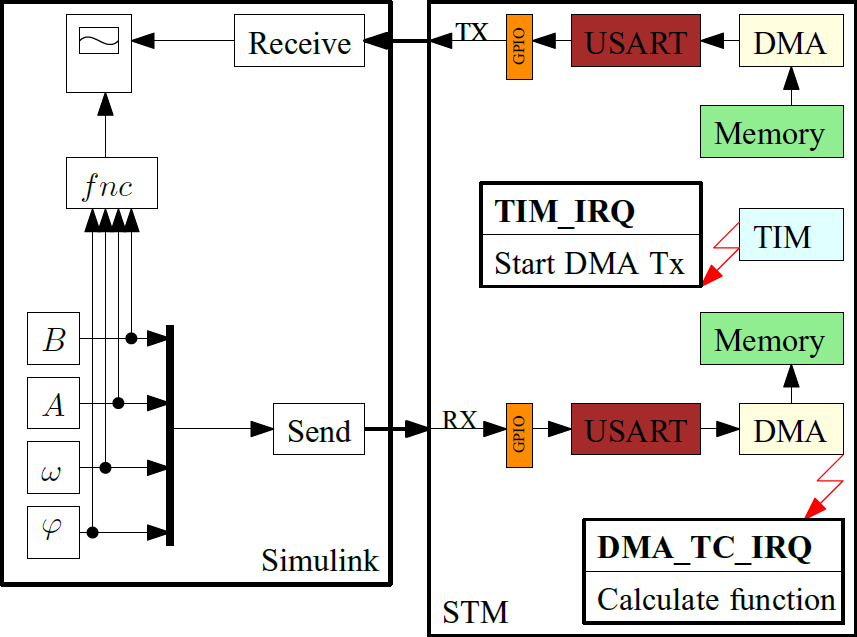
Data transfer and reception in Simulink.

**Software:**

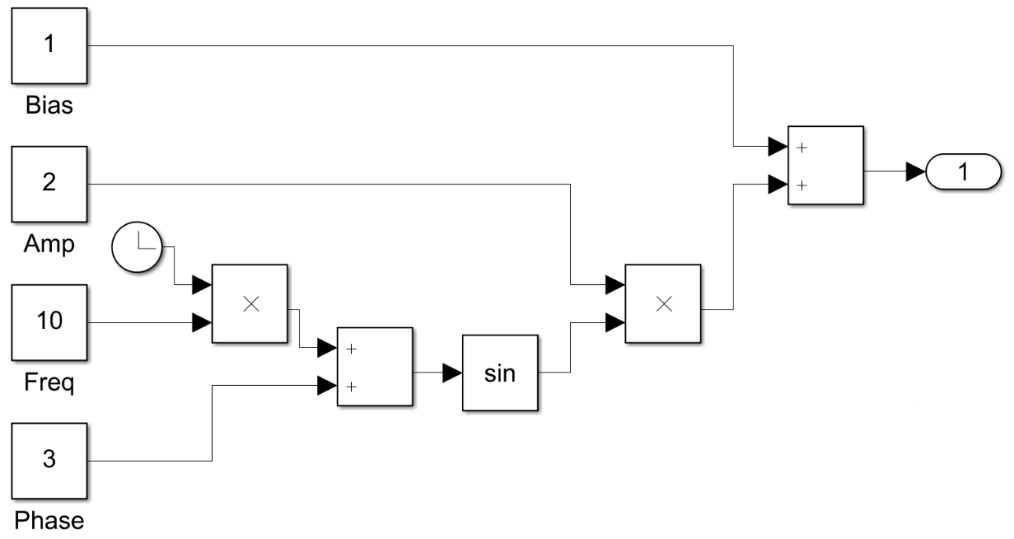
STM32CubeIDE, Matlab.

**General information:**

Schematic of the system being developed:

****

To generate a signal in Matlab, you can build a circuit like this:



For correct data exchange with Simulink in the Receive block it is necessary to specify Header and Terminator. Respectively in the sent packet from the microcontroller must be bytes responsible for the beginning and end of the packet.

*Required HAL functions*

Start the timer in the interrupt generation mode.

*HAL\_TIM\_Base\_Start\_IT TIM\_HandleTypeDef\* htim)*

* *htim* – the pointer to a configuration structure of type TIM\_HandleTypeDef.

Data transfer using DMA:

*HAL\_UART\_Transmit\_DMA (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 of data to be transferred.
* *Size* – the amount of data to be transmitted.

Data reception using an interrupt:

*HAL\_UART\_Receive\_DMA (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 of processing an interrupt from the timer:

*void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef\* htim)*

* *htim* – the pointer to a configuration structure of type TIM\_HandleTypeDef.

**The order of work:**

*Part I. Program development 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 documentation, select the timer that will be used to generate the periodic interrupt.

3. In the graphical initialization window of the controller (name.IOC), for the USART, with which you worked in the last work, add data transfer via DMA for receiving and transmitting. For the stream to receive data set the cyclic mode of operation.

4. In the graphical initialization window of the controller (name.IOC), set the Prescaller, Counter Period for the Timer defined in the previous step. Interrupt frequency should be equal to 100Hz. Take the bus frequency timer equal to 16MHz to calculate the value of dividers. Enable interrupts from the timer.

4. In the main file delete the callback function USART, and add a function to handle interrupts from the timer:

/\* USER CODE BEGIN 4 \*/

void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef\* htim)

{

}

/\* USER CODE END 4 \*/

5. In the appropriate area of the main function, start receiving data via UART using DMA and start the timer to operate in interrupt mode.

6. Implement the following algorithm of the program:

- Send the result of calculations to Simulink by interrupt from the timer;

- Form a packet in Simulink to send function parameters to the controller;

- Process the received packet in the controller and calculate the new value of the periodic function;

- Form a packet to send the result of calculations to the computer;

- Repeat the algorithm again

*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 and select a timer to generate a periodic interrupt.

3. Initialize the UART to receive and transmit data using DMA, enable interrupt on data reception. When setting the register BRR, take the frequency of the bus, which is connected to the UART, equal to 16MHz.

4. Initialize the timer to generate a periodic interrupt with a frequency of 100Hz.

6. Implement the algorithm in the interrupt handler from Part I p.6.

**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. | Frequency | Amplitude |
| 1 | 1 | 6 |
| 2 | 2 | 5 |
| 3 | 3 | 4 |
| 4 | 4 | 3 |
| 5 | 5 | 2 |
| 6 | 6 | 1 |
| 7 | 6 | 6 |
| 8 | 5 | 5 |
| 9 | 4 | 4 |
| 10 | 3 | 3 |
| 11 | 2 | 2 |
| 12 | 1 | 1 |