Организация последовательной передачи информации. Интерфейс UART

1 Введение

Цель работы – исследовать возможности передачи данных между устройствами с помощью последовательного интерфейса UART, организовать связь между двумя микроконтроллерами через оптопару с использованием шифрования.

Оборудование: одноплатный компьютер MSTN-M100 на базе микроконтроллера К1986ВЕ92QI; светодиод; фототранзистор; персональный компьютер.

2 Краткие теоретические сведения

UART – универсальный асинхронный приемо-передатчки (англ. Universal Asynchronous Receiver-Transmitter, UART) – один из наиболее распространенных интерфейсов, который встречается в микроконтроллерах. Распространенность объясняется простотой и нетребовательностью к ресурсам: для организации передачи данных используются всего 2 вывода (обычно, они называются Rx и Tx). Интерфейс UART радиальный, т.е. по линиям связи одного интерфейса подключаются только 2 устройства. Выход TX (Transmit) одного устройства подключается к входу RX (Receive) второго. Сигнал TX второго UART устройства соединяется с входом RX первого (рисунок 1).

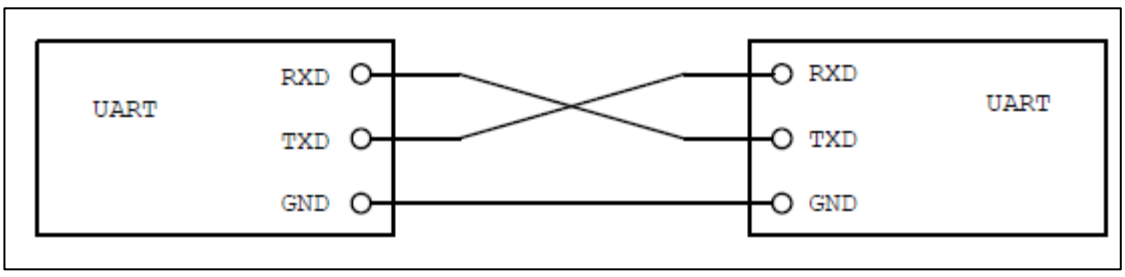

Рисунок 1 – Схема соединения интерфейсов UART

Свое развитие получил в том, что некоторые реализации интерфейса в периферии поддерживают синхронный режим передачи – в таком случае модуль называют USART. Он использует третью линию для подачи тактового сигнала.

Кроме того, могут подключаться другие выводы для выполнения служебных функций. В данной работе синхронный режим не рассматривается.

Последовательная передача данных - передача бит за битом.

Асинхронная передача - передача данных, при которой интервалы времени между направляемыми блоками данных не являются постоянными. Для выделения в потоке данных блоков в начале и конце каждого из них записываются старт/стопные биты (наличие и количество проверочного и стопового бита опциональны). При асинхронной передаче передатчик и приемник данных работают не зависимо друг от друга. Сигнал синхронизации отсутствует.

Синхронная передача – подразумевается передача сигнала синхронизации.

Устройство UART выполняет следующие функции:

– преобразование данных, полученных от периферийного устройства, из последовательной в параллельную форму;

– преобразование данных, передаваемых на периферийное устройство, из параллельной в последовательную форму.

Процессор читает и записывает данные, а также управляющую информацию и информацию о состоянии модуля. Прием и передача данных буферизуются с помощью внутренней памяти FIFO, позволяющей сохранить до 16 байтов независимо для режимов приема и передачи.

Протокол. Вначале передатчик переводит линию в низкий уровень — это старт бит. Зафиксировав, что состояние линии в низком уровне, приемник выжидает интервал T1 и считывает первый бит, потом через интервалы T2 считывает остальные биты. Последний бит это стоп бит. Говорящий о том, что передача этого байта завершена (рисунок 2).

В конце байта, перед стоп битом, может быть и бит четности (рисунок 3). Который используется для контроля качества передачи. Также может быть два стоповых бита, опять же для надежности. Битов может быть не 8, а 9. Все эти параметры задаются до начала отправки данных на передающей и принимающей

стороне. Обычно используется настройка: 8 бит, один старт-бит один стоп-бит, без бита четности.

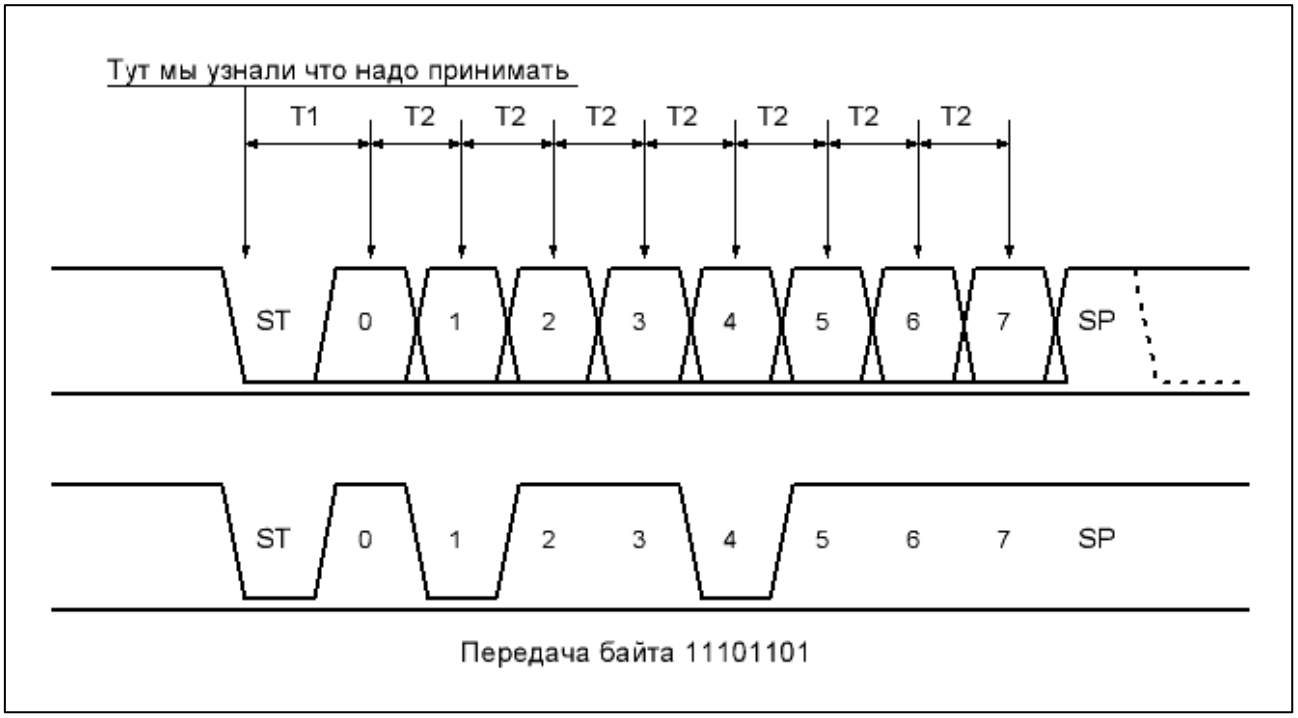

Рисунок 2 – Формат последовательных данных, формируемых UART (1)

Рисунок 3 - Формат последовательных данных, формируемых UART (2)

Обмен данными через UART между платами MSTN-M100 – очень полезная функция для многих проектов. Например, одна MSTN-M100 управляет моторами, а вторая используется для подключения сенсоров и передачи управляющих сигналов на первый микроконтроллер.

3 Работа с UART на одной плате MSTN-M100

Для работы с UART интерфейсом на одной плате, необходимо соединить одним проводом контакты 0 и 1 (0 – SERIAL1-TX, 1 – SERIAL1-RX) (рисунок 4).

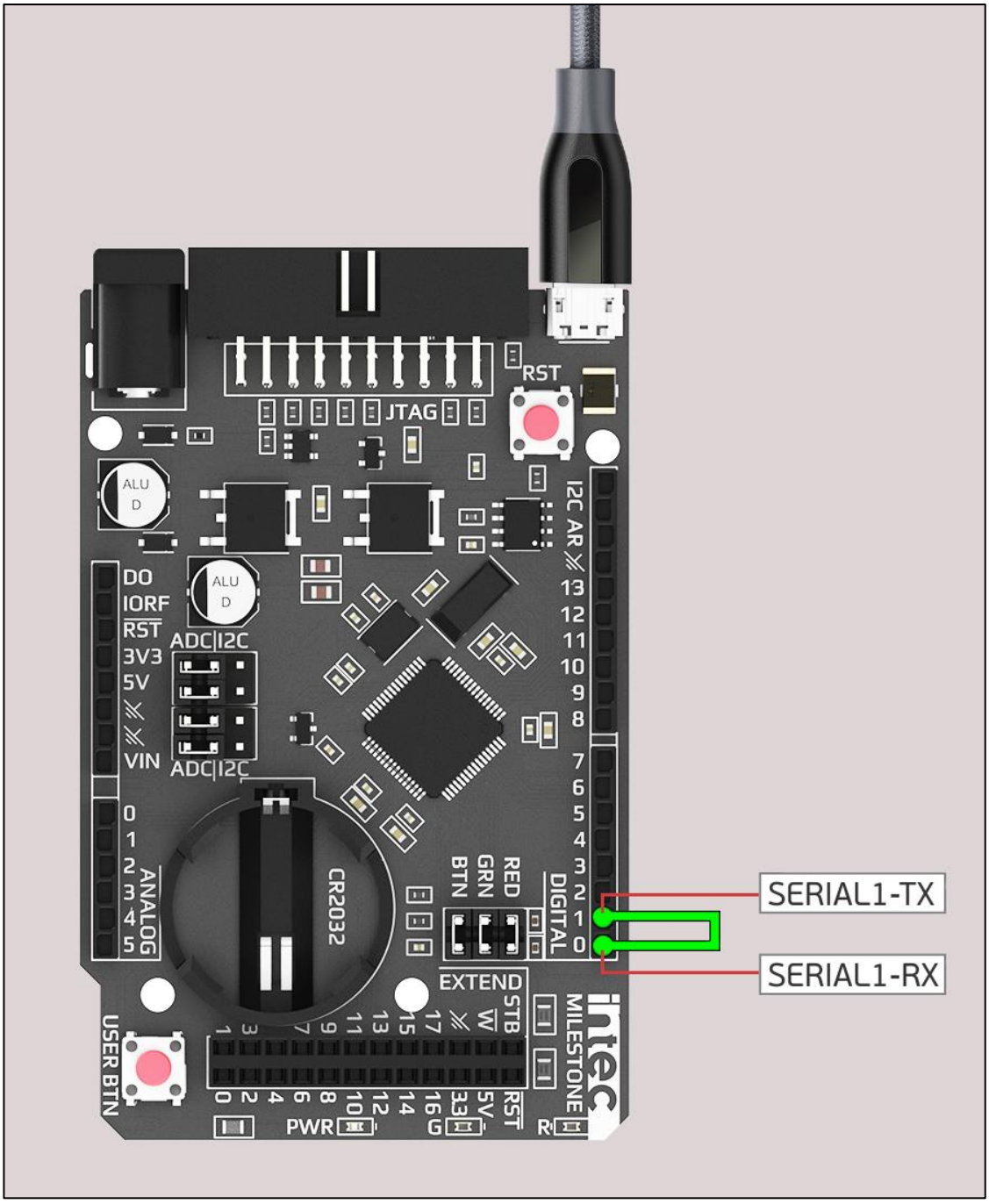

Рисунок 4 – Схема подключения для одной платы через провод

Вместо провода можно использовать **оптопару** (светодиод-фототранзистор), что позволяет передавать данные на некотором расстоянии. Электрическая принципиальная схема для использования оптопары приведена на рисунке 5.

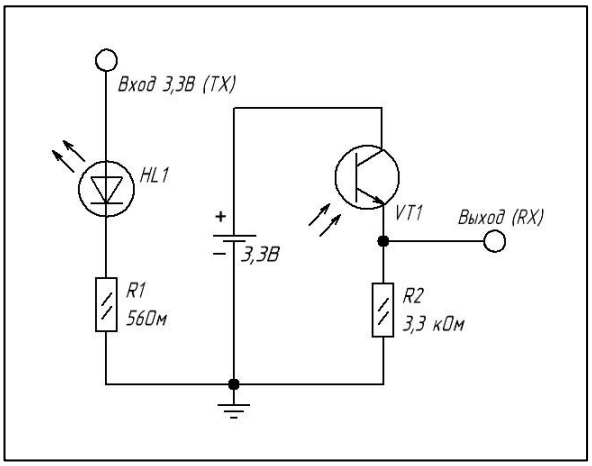

Рисунок 5 – Схема для оптопары

Рекомендуется использовать светодиод BL-L314IRCB и фототранзистор BPW17N. Важно использовать резистор R1 с сопротивлением 56±20 Ом для ограничения тока протекающего через светодиод HL1. Самая большая интенсивность свечения светодиода BL-L314IRCB приходится на ИК диапазон, поэтому при правильном подключении будет видно только **очень слабое** свечение в красном диапазоне видимого спектра. Расстояние между светодиодом и фототранзистором должно составлять не более 5 см.

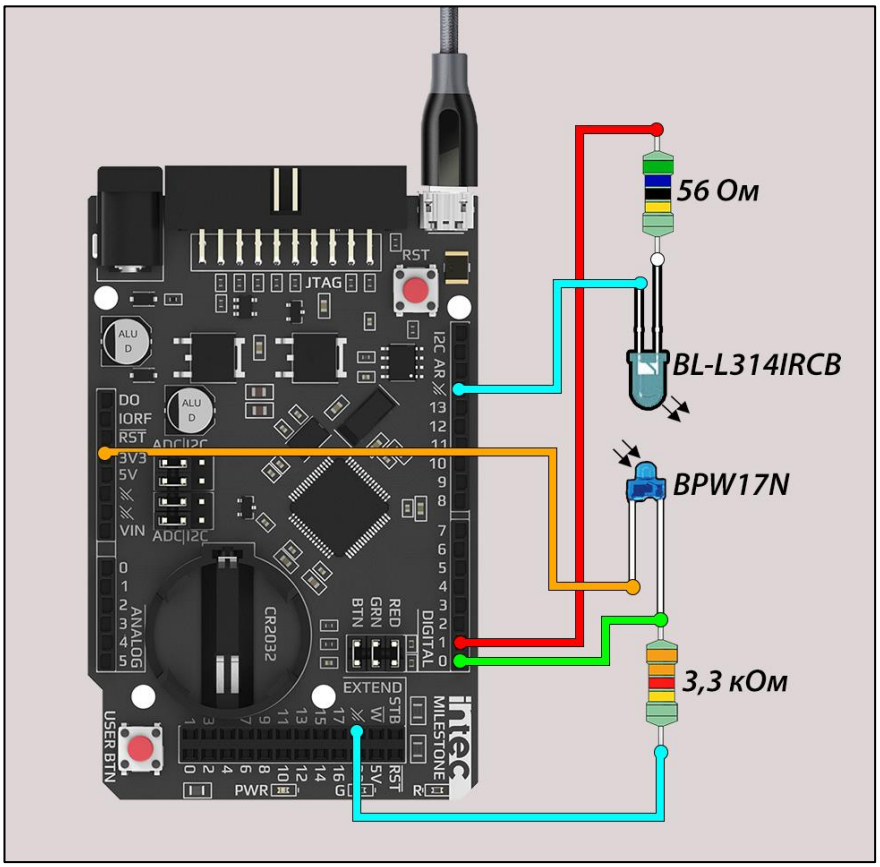

Рисунок 6 – Схема подключения оптопары для одной платы

4 Работа с UART на двух платах MSTN-M100

На рисунке 7 показана схема подключения двух плат MSTN-M100. Обратите внимание, что необходимо объединить контакты Gnd. В противном случае, обмен данными происходить не будет! При подключении контакт TX подключается к RX.

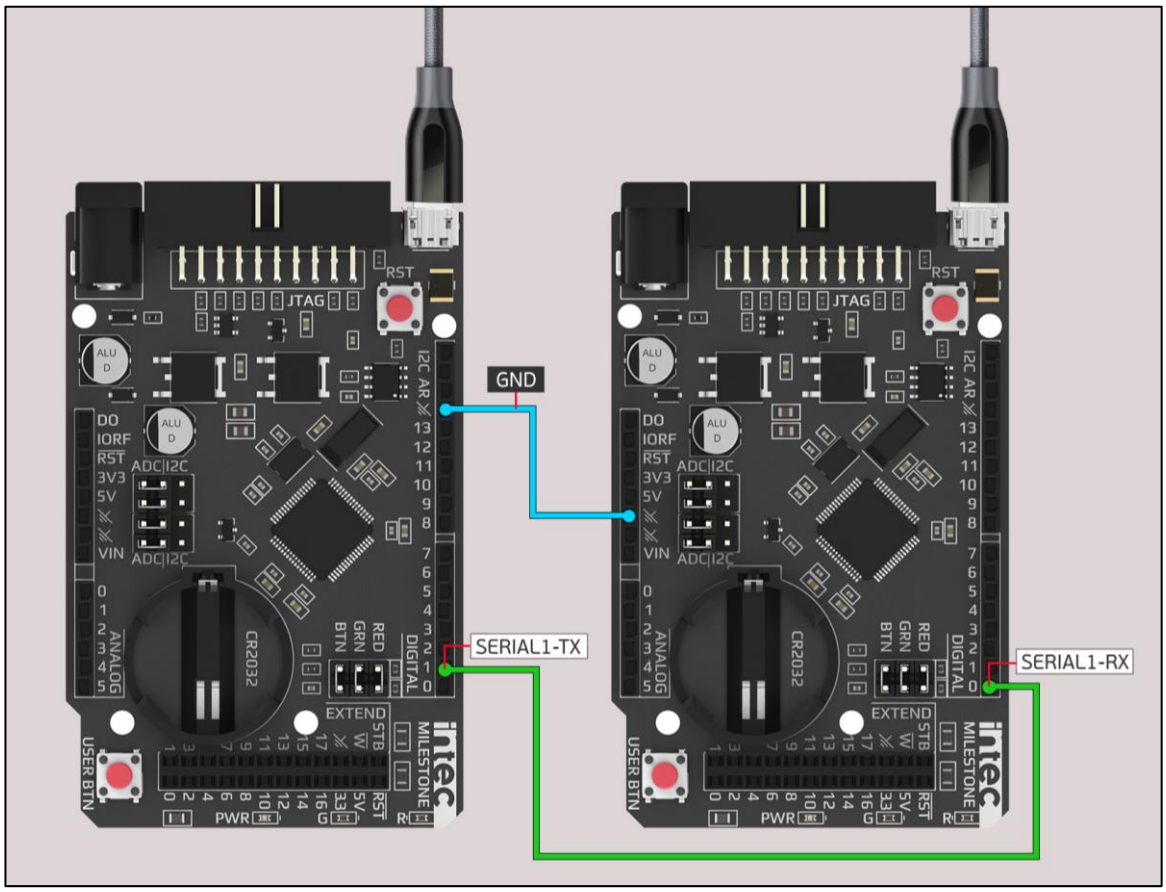

Рисунок 7 – Схема подключения для двух плат через провода

Каждая из плат подключается к разным компьютерам («MSTN-M100(1) – ПК-1»; «MSTN-M100(2) – ПК-2»). Одновременное использование двух плат в среде разработки NetBeans не предусмотрено производителем одноплатного компьютера MSTN-M100.

Чтобы использовать **оптопару** для передачи информации через UART на **вторую плату**, необходимо подключить все компоненты в соответствии со схемой на рисунке 8. Обратите внимание, что необходимо объединить контакты Gnd.

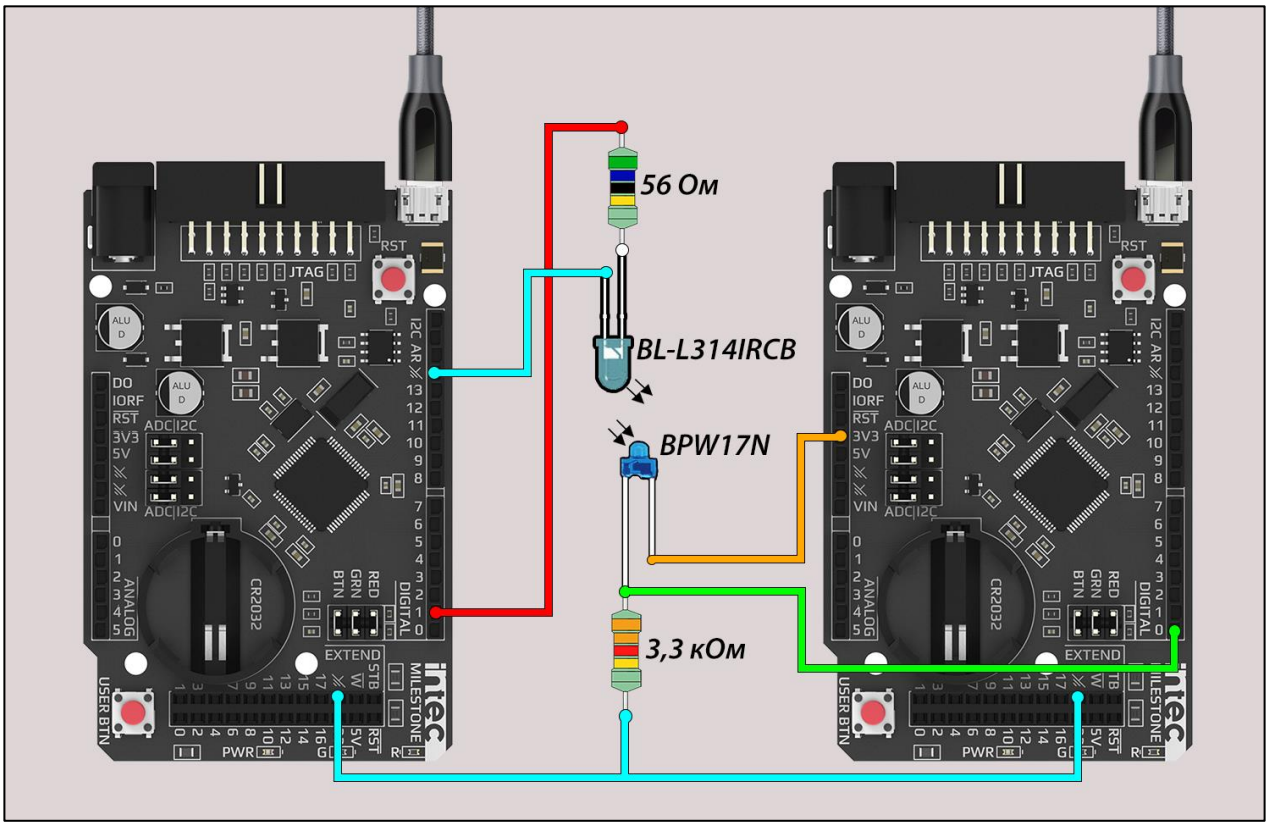

Рисунок 8 – Схема подключения оптопары для двух плат

5 Программирование UART на плате MSTN-M100

Необходимо создать новый проект в среде программирования NetBeans. Воспользоваться примером программы.

```
#include "main.h"
#include <stdlib.h>
#include "mstn_uart.h"
#include <stdio.h>
#include "mstn_clk.h"
#include <cstring>
#include <mstn.h>

int main(int argc, char *argv[])
{

    Delay(4000);
    char msg_transmit[128]; //Сообщение для отправки
    char msg_receive[128];  //Принятое сообщение
    int res;                //Для хранения одного байта из буфера и последующего
преобразования его в символ
    int k;                  //Порядковый номер элемента массива
    int count;              //Число символов в буфере

    while(1)
    {
        UART_Begin(SERIAL1, 300);                //Инициализация последоватлеьного
соединения
        //Отправка

        printf("Enter message:\n");
```

```
        scanf("%s", msg_transmit);                  //Ввод сообщения
        printf("Your msg: %s\n", msg_transmit);     //Вывод этого сообщения на экран

        printf("Ready... \n");
        Delay(1000);

        UART_PrintStr(SERIAL1, msg_transmit);       //Передача строки через интерфейс UART



        //Приём

        UART_Wait(SERIAL1, 1000);                   //Ожидание окончания приема посылки

        count = UART_Available(SERIAL1);            //Число символов в буфере
        printf("Number of characters in buffer: %d\n", count);

        k = 0;
        while(UART_Available(SERIAL1) != 0) //Пока буфер не будет пуст
        {
            res = UART_Read(SERIAL1);
            msg_receive[k] = (char)res; //Преобразование типа данных
            k++;
        }

        printf("Received message: %s\n", msg_receive);

        memset(msg_receive, 0, sizeof(msg_receive)); //Очищение массива
        UART_End(SERIAL1);
    }
    return EXIT_SUCCESS;
       }
```

Листинг 1 – Пример программы для работы с UART на одной плате

Выполнить сборку проекта (рисунок 9).

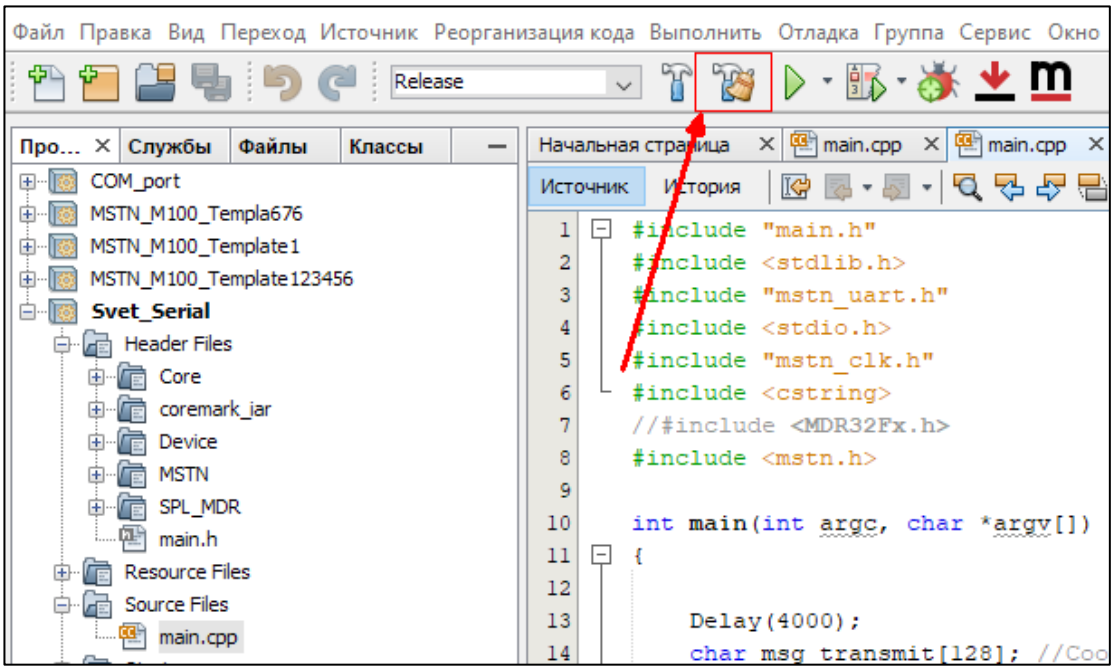

Рисунок 9 – Сборка проекта

Подключить заголовочные файлы (рисунок 10). Подключение заголовочных файлов описано в «C:\Intec\MSTN\M100\Doc\MSTN-M100_Quick_Start.pdf» на страницах 24-27.

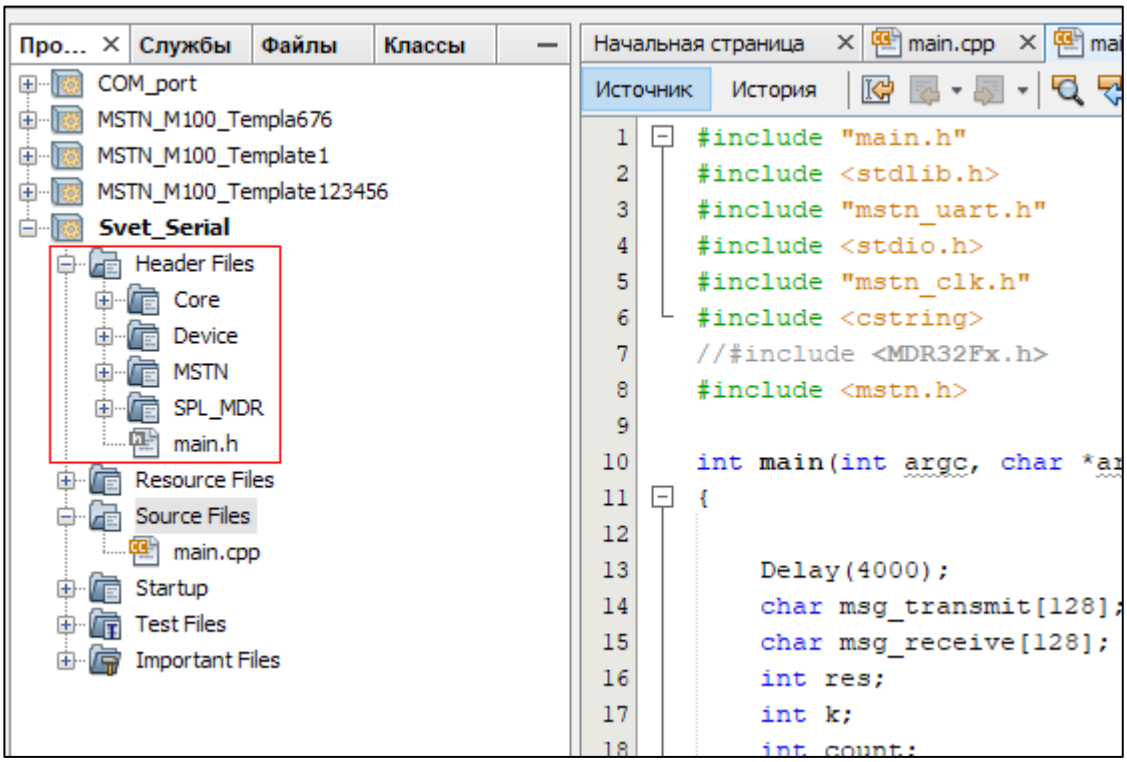

Рисунок 10 – Заголовочные файлы

После успешной сборки проекта необходимо загрузить программу на микроконтроллер (рисунок 11).

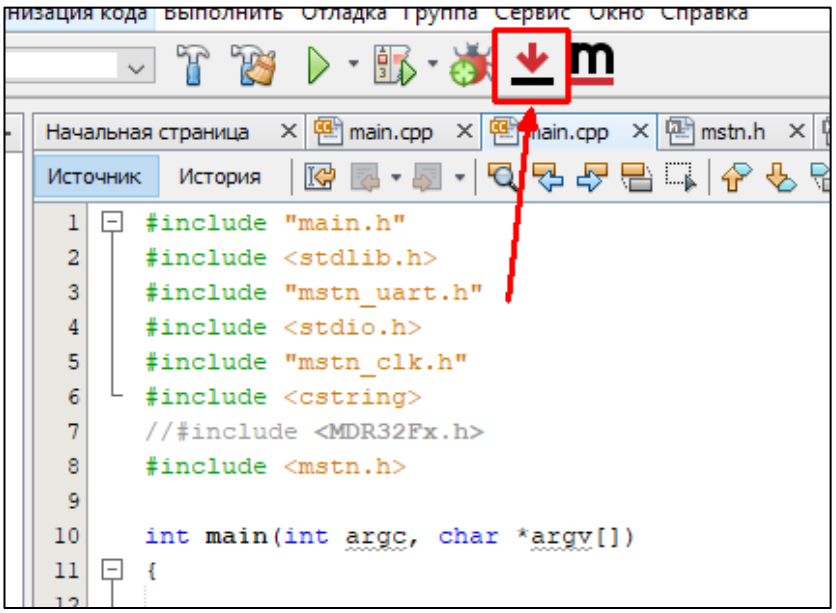

Рисунок 11 – Загрузка программы на микроконтроллер

После завершения загрузки программы необходимо установить связь с MSTN-M100 (рисунок 12).

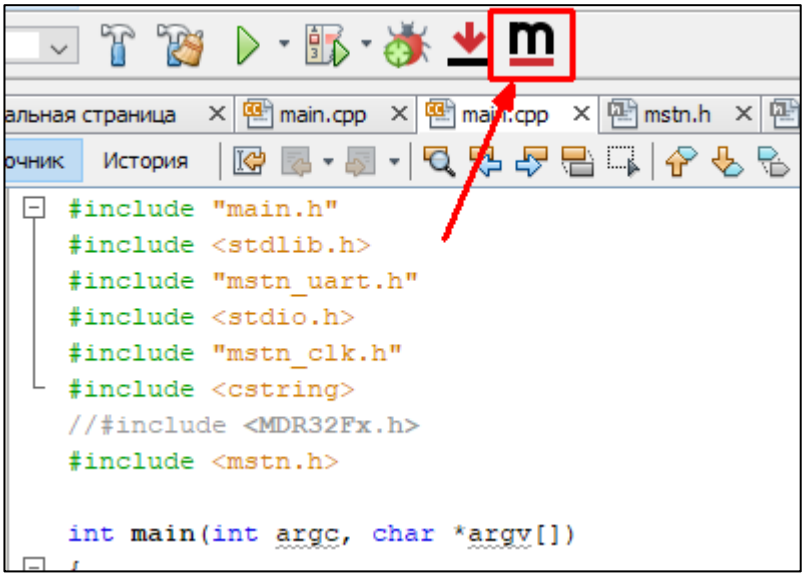

Рисунок 12 – Связь с платой

После установки связи с платой необходимо нажать кнопку «RST» (на плате возле micro-USB). В окне консоли следует выполнить отправку сообщения и убедиться, что сообщение было получено (рисунок 13).

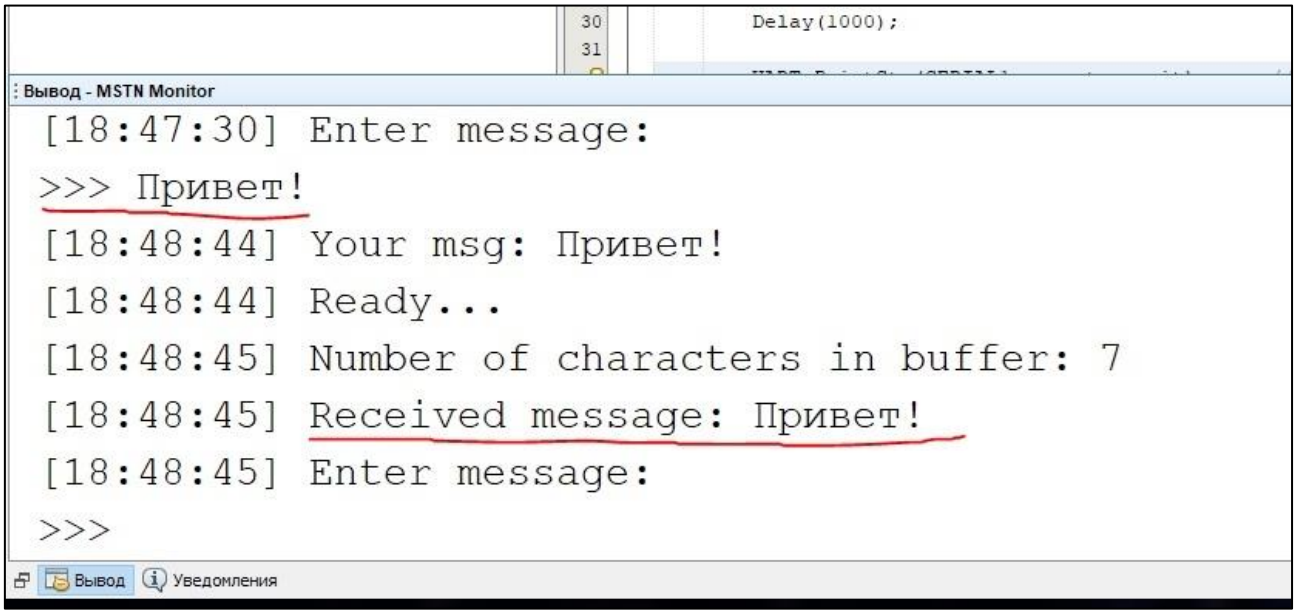

Рисунок 13 – Отправка-получение сообщения

Ниже приведён список функций, используемых в программе, для работы с UART из стандартной библиотеки MSTN.

Delay(int64_t msec) – останавливает выполнение программы в текущем контексте на заданное в параметре количество миллисекунд.

UART_Begin(_UART_InterfaceNum interfaceNum, uint32_t baud) – инициирует последовательное соединение и задает скорость передачи данных в бит/с (бод). Распространенные значения скорости: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 или 115200. Настройки выставляются по умолчанию: 8 значащих бит в кадре, без контроля четности, 1 стоп-бит.

interfaceNum – номер интерфейса UART;

baud – скорость передачи.

UART_PrintStr(_UART_InterfaceNum interfaceNum, const char*, ...) – printf() для интерфейса UART (печатает форматированные выходные данные в стандартный выходной поток).

UART_Wait(_UART_InterfaceNum interfaceNum, uint32_t maxWaitTime) – ожидает окончания приема посылки. Событие окончания приема возникает, если на вход приемника не поступало новых данных в течение периода времени, необходимого для передачи 32 бит. Если в данный момент времени прием не ведется, то функция будет ожидать начала и окончания приема. Максимальное время ожидания (в мкс.) задается во втором аргументе.

Внимание! В случае, если длина принимаемой посылки будет больше длины буфера - "лишние" байты в начале посылки будут утеряны.

int UART_Available(_UART_InterfaceNum interfaceNum) – функция получает количество байт(символов) доступных для чтения из последовательного интерфейса связи. Это те байты, которые уже поступили и записаны в буфер последовательного порта. Буфер может хранить до 128 байт, однако функция вернет общее количество байт, принятых после последнего обращения к функциям UART_Read() или UART_Peek(), даже если количество принятых байт превышает длину буфера (при этом доступными будут только последние UARTBUFFLEN принятых байт). Таким образом можно отследить событие переполнения буфера.

int UART_Read(_UART_InterfaceNum interfaceNum) – функция считывает очередной доступный байт из буфера последовательного соединения. Если при обращении к функции буфер приема переполнен, значение "количество доступных байт" сбрасывается к длине буфера (т.е. следующее за этим обращение к функции UART_Available() вернет значение (UARTBUFFLEN - 1)).

UART_End(_UART_InterfaceNum interfaceNum) – деинициализирует интерфейс (линии вывода переключаются в состояние по умолчанию).

Обо всех остальных функциях стандартной библиотеки для работы с UART интерфейсом можно прочитать в заголовочном файле «Header Files\MSTN\ mstn_uart.h» (рисунок 14).

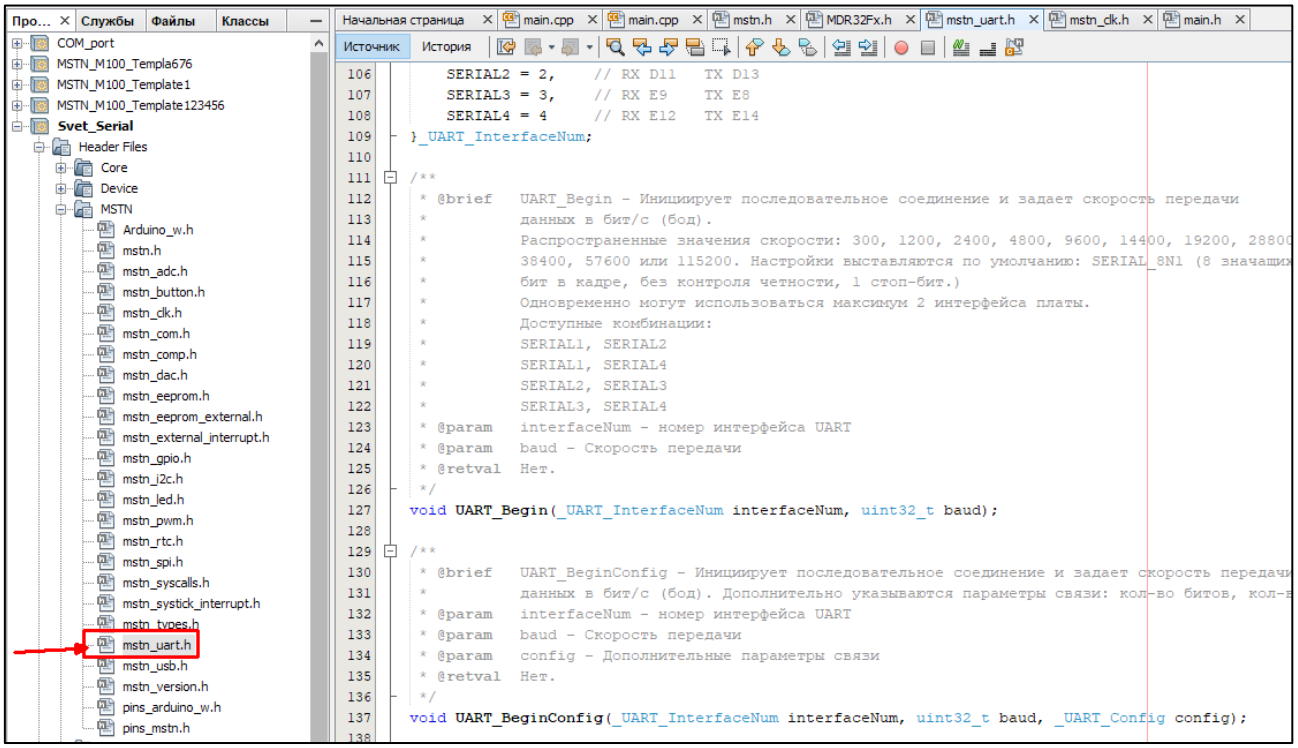

Рисунок 14 – Описание функций для работы с UART

6 Задание

1. Передать-получить сообщение на **одной** плате через UART интерфейс, Соединение через **провод**.

2. Передать-получить сообщение на **одной** плате через UART интерфейс. Соединение через **оптопару**.

3. Передать-получить сообщение через UART интерфейс, используя **две** платы. Соединение через **провод**. Задание выполняется на двух компьютерах в паре. Одна плата выполняет отправку сообщения, вторая получение. (Пример программы следует разделить и доработать).

4. Передать-получить сообщение через UART интерфейс, используя **две** платы. Соединение через **оптопару**. Задание выполняется на двух компьютерах в паре. Одна плата выполняет отправку сообщения, вторая получение. (Пример программы следует разделить и доработать).

Реализовать шифрование в одном из заданий, используя любой исторический или современный шифр шифра (шифр Цезаря, шифр Виженера, аффинный шифр и т.п.). Полный цикл: ввести сообщение - зашифровать сообщение - передать данные - принять данные – расшифровать сообщение - вывести сообщение.