ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Дисциплина «Промышленные сети»

### ОТЧЕТ

по лабораторной работе

на тему

«Реализация Modbus RTU в среде FreeDOS»

Выполнил:

студенты группы 3540901/02001

Клюев А.М.

Бараев Д.Р.

Дроздов Н.Д.

Проверил:

Жуков Андрей Владимирович

«\_\_» \_\_\_\_\_\_\_\_\_\_ 2021г., \_\_\_\_\_\_\_\_\_\_

Санкт-Петербург 2021

# Цель работы:

Реализация функций 1,5 (чтение/запись) протокола Modbus RTU в среде FreeDOS.

# Ход работы:

Ниже представлен код программы с комментариями:

В строках с 110 по 118:

*for( iterInf1 = 0;; iterInf1++){//// infl*

*printf("iteraciya %x=========================================\n values of mOmory \n", iterInf1);*

*for(c=0; c<MOmoryLen; c++)*

*printf("%2x ", c);*

*printf("\n");*

*for(c=0; c<MOmoryLen; c++)*

*printf("%2x ", mOmory[c]);*

*printf("\n \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");*

На 110 строке запущен вечный цикл с подсчётом итераций *iterInf1*.

В строках 113–118 печатается на экран номера манипулируем ячеек памяти и их содержимое.

В строках с 120 по 133:

*chet=0;*

*for(;;){//////paus*

*#ifdef \_USE\_PT\_*

*c = rx(50000); /\* 50 ms \*/*

*#else*

*c = rx(3); /\* 3 sec \*/*

*#endif*

*if(c!= T\_OUT){*

*break;*

*}*

*}*

*massbuf[0]=c;*

В строке 122 запускается вечный цикл – пауза из него выход только по приходу сообщений в строках с 129–131.

В строках с 138 по 163 производится проверка того, что входное сообщение не превышает размера входного буфера (massbuf), и записывается в него входное сообщение, сообщение об ошибке обрабатывается при помощи переменных masserr2, masserr :

*for (;;) { /\* receiving \*/*

*#ifdef \_USE\_PT\_*

*c = rx(50000); /\* 50 ms \*/*

*#else*

*c = rx(3); /\* 3 sec \*/*

*#endif*

*chet++;*

*if (c == T\_OUT) {*

*printf("?-timeout\n");*

*break;*

*} else {*

*if (c == RCV\_ERR) {*

*masserr2=1;*

*masserr=2;*

*} else {*

*massbuf[chet]=c;*

*}*

*}*

*if(chet>=BUFsz){*

*masserr=1;*

*}*

*}*

В строках с 166 по 173 switch обрабатываются ошибки masserr==0 – нет ошибок, в цикле выводится содержимое буфера- входного сообщения:

*switch(masserr)*

*{*

*case 0:*

*printf(" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_print \n");*

*for(iter=0; iter< chet; iter++){*

*printf("%4x", massbuf[iter]);*

*}*

В строках с 175 по 184, в условии проверяется, что контрольная сумма- верная, функция rd\_seq считывает значение буфера в переменные node, func в строках 182 и 183:

*crcsumm=ModRTU\_CRC(massbuf, chet-2);*

*if((crcsumm<<8)/256 ==massbuf[chet-2] && massbuf[chet-1] == crcsumm>>8){*

*printf("\n kontrol summ == true\n");*

*n2=chet-2;*

*iter=0;*

*iter= rd\_seq(iter, 1, &node);*

*iter= rd\_seq(iter, 1, &func);*

*printf("\n funk number== %2x \n", func);*

В строках с 186 по 199 проверяется, что входное сообщение соответствует функции 1 (считывание данных). Далее rd\_seq считывается содержимое буфера в переменные

*if(node==1)*

*{*

*if(func==1 || func==5){*

*if(func==1)*

*{*

*iter= rd\_seq(iter, 2, &offset);*

*iter= rd\_seq(iter, 2, &number);*

*printf("\n return read last bits = %4x \n", iter);*

*printf("funk 1111111111111111111111111111111 \n");*

*printf("node= %2x \n", node);*

*printf("func= %2x \n", func);*

*printf("offset= %4x \n", offset);*

*printf("number= %4x \n", number);*

В строках с 201 по 210 проверяется что входные данные корректны и не выходят за границы считываемого массива. И начинает формироваться ответ в массив ans:

*if(node==1 && number>=2){*

*printf("success\n");*

*printf("last numb memory = %2x \n", number+ offset);*

*if(number+ offset< MOmoryLen){*

*ans[0]=0x01;*

*ans[1]=0x01;*

В строках с 213 по 222 вычисляется длина данных в ответе в байтах:

*corrans1\_iterator=3;*

*if(number%8==0) {*

*corrans1\_len=number / 8;*

*}*

*else {*

*corrans1\_len=number / 8 + 1;*

*}*

*ans[2]=corrans1\_len;*

В строках с 224 по 234 считывается побитово содержимое памяти *mOmory* компьютера в ответ ans:

*corrans1\_iterator2=0;*

*ans[corrans1\_iterator]=0;*

*for(c=offset; c<number+ offset; c++)*

*{*

*if(corrans1\_iterator2>=8){corrans1\_iterator2=0;*

*corrans1\_iterator++;*

*ans[corrans1\_iterator]=0;}*

*ans[corrans1\_iterator]=((mOmory[c])<<(corrans1\_iterator2))+ans[corrans1\_iterator];*

*corrans1\_iterator2++;*

*}*

В строках с 237 по 241 завершается формирование содержимого массива для ответа ans

*corrans1\_len=corrans1\_len+3;///massege len*

*crcsumm=ModRTU\_CRC(ans, corrans1\_len);*

*ans[corrans1\_len]=(crcsumm<<8)/256;*

*ans[corrans1\_len+1]=crcsumm>>8;*

*thisanslen=corrans1\_len+2;*

В строках с 243 по 255 формируется ответ при выходе за границы считываемого массива:

*}*

*else{*

*ans[0]=0x80+1;*

*ans[1]=0x02;*

*//ans[3]=0x04;*

*for(c=0; c<2; c++)*

*printf("%2x\n", ans[c]);*

*crcsumm=ModRTU\_CRC(ans, 2);*

*ans[3]=(crcsumm<<8)/256;*

*ans[4]=crcsumm>>8;*

*thisanslen=5;*

*}*

В строках с 261 по 273 сформированное сообщение сообщение:

*timerDelay(numberOfTimeout);*

*//c=1;*

*c = reset\_rs();*

*if (c){*

*puts("\nReceiver was not empty!");*

*}*

*else{*

*for(c=0; c<thisanslen; c++)*

*tx (ans[c]);*

*}*

*}*

В строках с 275 по 286 обрабатывается функция 5 из входного сообщения

*}*

*if(func==5)*

*{*

*iter= rd\_seq(iter, 2, &offset);*

*iter= rd\_seq(iter, 2, &number);*

*printf("\n return read last bits = %4x \n", iter);*

*printf("funk 5555555555555555555555555 \n");*

*printf("node= %2x \n", node);*

*printf("func= %2x \n", func);*

*printf("poz= %4x \n", offset);*

*printf("val= %4x \n", number);*

В строках с 288 по 303 формируется ответ и изменяется содержимое памяти (299–302)

*if(offset< MOmoryLen && (number==0x0 || number==0xFF00))*

*{*

*ans[0]=massbuf[0];*

*ans[1]=massbuf[1];*

*ans[2]=massbuf[2];*

*ans[3]=massbuf[3];*

*ans[4]=massbuf[4];*

*ans[5]=massbuf[5];*

*ans[6]=massbuf[6];*

*ans[7]=massbuf[7];*

*thisanslen=8;*

*if(number==0x0)*

*mOmory[offset]=0;*

*else*

*mOmory[offset]=1;*

*}*

В строках с 304 по 315 формируется ответ при некорректном запросе функции 5:

*else*

*{*

*ans[0]=0x01;*

*ans[1]=0x85;*

*ans[2]=0x01;*

*crcsumm=ModRTU\_CRC(ans, 3);*

*ans[3]=(crcsumm<<8)/256;*

*ans[4]=crcsumm>>8;*

*thisanslen=5;*

*}*

В строках с 320 по 336 отправляется ответ на функцию 5 и закрываются скобки обработки всех корректно полученных запросов:

*timerDelay(numberOfTimeout);*

*c = reset\_rs();*

*if (c) {*

*puts("\nReceiver was not empty!");*

*}*

*else{*

*for(c=0; c<thisanslen; c++)*

*tx (ans[c]);*

*}*

*}*

*}*

*}*

В строках с 366 по 384 обрабатывается некорректная контрольная сумма и некорректные входные сообщения:

*}*

*else{*

*printf("\n control summ == false \n");*

*}*

*break;*

*case 1:*

*printf("more leng buf");*

*break;*

*case 2:*

*break;*

*}*

*if(masserr2==1){*

*printf("ERRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRor mass");*

*}*

*}*

*return 0;*

*}*

# Вывод:

В ходе данной лабораторной работы было изучено:

* Эмулятор терминала ZOC (Windows);
* Эмулятор терминала Term90 (FreeDOS);

Было осуществлено:

* Проверка связи;
* Передача файлов;
* Программирование обмена через COM-порт;
* Имитатор ведущего Modbus Tester;
* Программирование приема запросов;
* Разбор принятого запроса;

В результате выполнения лабораторной работы были реализованы функции 1,5 (чтение/запись) протокола Modbus RTU в среде FreeDOS.