**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**



Звіт до лабораторної роботи №5

з навчальної дисципліни: «Мікроконтролери ч.1»

**РОЗРОБЛЕННЯ ЗАВАДОСТІЙКОГО ПРОТОКОЛУ ОБМІНУ**

**ДАНИХ В МЕРЕЖІ МІКРОКОНТРОЛЕРІВ НА БАЗІ RS-485**

Варіант №38

**Виконав:**

ст. гр. ІР-22

Матвійчук А.І.

**Прийняв:**

Ст. викладач кафедри КСА

Лагун І.І.

Львів – 2021

**Порядок виконання роботи:**

Порядок виконання роботи

1. Згідно варіанту завдання (таблиця 1) зібрати у пакеті симуляції

Proteus схему та написати програми для реалізації вказаного

завдання.

2. Розробити клієнтську програму для керування МК-системою.

3. Протестувати роботу клієнтської програми зі схемою в Proteus.

4. Залити програми в Arduino-Kits.

5. Отримані результати представити викладачу.

**Завдання:**

1. Спроектувати принципову схему системи обміну даними

між мікроконтролерною мережею (один ATmega2560 та два

ATmega328P) на основі інтерфейсу RS-485 та персональним

комп’ютером;

2. Розробити протокол обміну між мікроконтролерами

«master» та «slave» мережі на основі RS-485, а також між

клієнтською програмою Windows та МК «master» (RS-232);

У протоколі мають бути наявні контрольні суми CRC

(таблиця 2).

3. Для МК «master» та «slave» написати програмний код, що

реалізовуватиме поставлену задачу:

 1й Slave надсилає «Прізвище Ім’я По-батькові» студента;

 2й Slave надсилає «Дату народження» студента;

Продублювати 5 раз ці повідомлення з внесеними в них

спотвореннями згідно варіанту завдання.

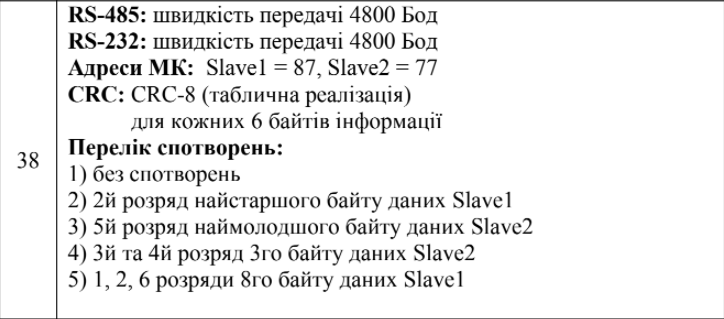
4. Розробити клієнтську програму (С++, C#, JavaScript) та

написати для неї програмний код, що реалізовуватиме

зв’язок між комп’ютером та МК «master» і відображатиме

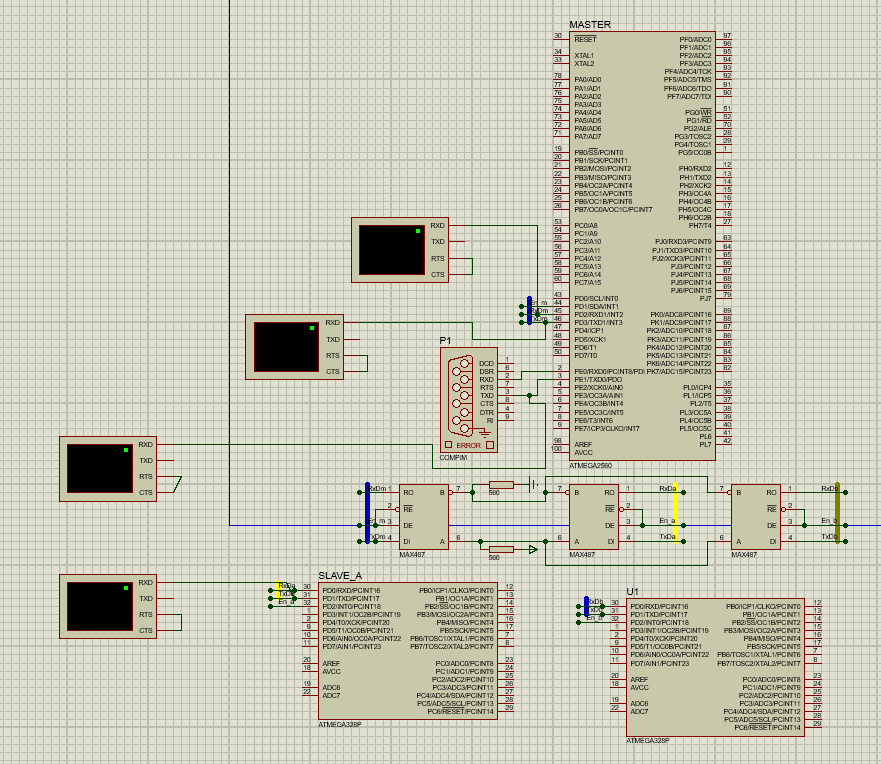
та аналізуватиме отримані дані з МК «slave» на наявність

помилок передачі.

****

****

**Схема у Proteus:**



**Відео презентація лабораторної роботи:**

**https://drive.google.com/file/d/1nuP-UuLilp9CkiXwsIgllWXaEVm-F0B2/view**

**Код прошивки:**

master.ino

#define BAUD 4800 //Baud rate of rs485

#define MODE 20

#define TRANSMIT HIGH

#define RECEIVE LOW

#define ID 1 //Master ID

#define SLAVE1\_ID 87

#define SLAVE2\_ID 77

char symbol;

int i = 0;

char\* bufferPtr;

int lastMessageFrom;

int sendTo;

long crcRecieved;

void setup(void) {

Serial1.begin(BAUD); //Serial to read from python UI

Serial.begin(4800); //Serial between other MCU(RS485)

pinMode(MODE, OUTPUT);

}

void loop()

{

if (Serial.available()) {

int inByte = Serial.read();

if (inByte == 49) {

sendMessage(SLAVE1\_ID, "Give");

} else if (inByte == 50) {

sendMessage(SLAVE2\_ID, "Give");

}

}

i = 0;

crcRecieved = 0;

if (Serial1.available() ) {

bufferPtr = (char\*)malloc(50 \* sizeof(char));

delay(3);

while (Serial1.available()) {

symbol = Serial1.read();

if (symbol == ';') {

bufferPtr[i++] = '\0';

i = 0;

int c;

sscanf(bufferPtr, "%d", &c);

if (c != ID) {

while (Serial1.available()) {

Serial1.read();

}

free(bufferPtr);

return;

}

} else if (symbol == ':') {

bufferPtr[i++] = '\0';

i = 0;

int c;

sscanf(bufferPtr, "%d", &c);

lastMessageFrom = c;

} else if (symbol == '!') {

bufferPtr[i++] = '\0';

i = 0;

sscanf(bufferPtr, "%lx", &crcRecieved);

} else {

bufferPtr[i++] = symbol;

}

}

bufferPtr[i++] = '\0';

}

if (i > 0 && strlen(bufferPtr) > 0 && crcRecieved != 0) {

sendToUI(bufferPtr);

} else if (crcRecieved == 0 && strlen(bufferPtr) > 0 && i > 0){

Serial.print(bufferPtr);

}

free(bufferPtr);

}

void sendToUI(char str[]) {

char buffer[30];

sprintf(buffer, "#%lx;%s", crcRecieved, str);

Serial.print(buffer);

}

void sendMessage(int recevierId, char str[]) {

digitalWrite(MODE, TRANSMIT);

delay(60);

char buffer[20];

sprintf(buffer, "%d;%s", recevierId, str);

Serial1.print(buffer);

delay(60);

digitalWrite(MODE, RECEIVE);

}

slave1.ino

#define BAUD 4800

#define MODE 2

#define TRANSMIT HIGH

#define RECEIVE LOW

#define ID 87

char symbol;

int i = 0;

char\* bufferPtr;

void setup(void) {

Serial.begin(BAUD);

pinMode(MODE, OUTPUT);

digitalWrite(MODE, RECEIVE);

}

void loop() {

i = 0;

symbol = 0;

if (Serial.available()) {

bufferPtr = (char\*)malloc(50 \* sizeof(char));

delay(5);

while (Serial.available()) {

symbol = Serial.read();

if (symbol == ';') {

bufferPtr[i++] = '\0';

i = 0;

int c;

sscanf(bufferPtr, "%d", &c);

if (c != ID) {

while (Serial.available()){Serial.read();}

free(bufferPtr);

return;

}

} else {

bufferPtr[i++] = symbol;

}

}

if (i > 0){

bufferPtr[i++] = '\0';

}

}

if (i > 0) {

if (!strcmp(bufferPtr, "Give")) {

char message[] = "06.03.2002";

sendMessage(1, message, sizeof(message));

}

}

free(bufferPtr);

//delay(200);

}

void sendMessage(int recevierId, char str[], long sizeOf) {

char buffer[20];

digitalWrite(MODE, TRANSMIT);

long crc = calculateCrcWithTable(str, sizeOf / sizeof(char) - 1, 0x7, 0x00, false, false, 0x00, 8);

delay(60); //1010 ^ 1000 = 0010 => 0010 ^ 1000 = 1010

sprintf(buffer, "%d;%d:%lx!%s\*", recevierId, ID, crc, str);

Serial.print(buffer); // Без спотворень

delay(50);

str[0] ^= (1<<2); // 2й розряд найстаршого байту даних Slave1

sprintf(buffer, "%d;%d:%lx!%s\*", recevierId, ID, crc, str);

Serial.print(buffer);

delay(50);

str[0] ^= (1<<2); //фікс попередньої операції

/\*

str[strlen(str) - 1] ^= (1<<5); //5й розряд наймолодшого байту даних Slave2

sprintf(buffer, "%d;%d:%lx!%s\*", recevierId, ID, crc, str);

Serial.print(buffer);

delay(50);

str[strlen(str) - 1] ^= (1<<5); //фікс попередньої операції

str[3] ^= (1<<3) | (1<<4); //3й та 4й розряд 3го байту даних Slave2

sprintf(buffer, "%d;%d:%lx!%s\*", recevierId, ID, crc, str);

Serial.print(buffer);

delay(50);

str[3] ^= (1<<3) | (1<<4); //фікс попередньої операції

\*/

str[8] ^= (1<<1) | (1<<2) | (1<<6); //1, 2, 6 розряди 8го байту даних Slave1

sprintf(buffer, "%d;%d:%lx!%s\*", recevierId, ID, crc, str);

Serial.print(buffer);

delay(60);

digitalWrite(MODE, RECEIVE);

}

long calculateCrcWithTable(const char data[], const unsigned long length, unsigned long crc,

const unsigned long poly, const bool refIn, const bool refOut, const unsigned long xorOut,

const unsigned long crcBitLength) {

unsigned long greatestBit = 1 << (crcBitLength - 1);

unsigned long cutter = (1 << crcBitLength) - 1;

unsigned long table[256];

for (unsigned long i = 0; i < 256; i++) {

unsigned long res = i;

if (crcBitLength > 8) {

res <<= (crcBitLength - 8);

}

for (unsigned int j = 0; j < 8; j++) {

if ((res & greatestBit) != 0) {

res = (res << 1) ^ poly;

}

else {

res <<= 1;

}

}

table[i] = res & cutter;

}

unsigned long shift = (crcBitLength - 8 < 0) ? 0 : (crcBitLength - 8);

for (unsigned long i = 0; i < length; i++) {

unsigned long temp = data[i];

if (refIn) {

temp = reverseBits(temp, 8);

}

crc = table[(crc >> shift) ^ temp] ^ (crc << 8);

crc &= cutter;

}

if (refOut) {

crc = reverseBits(crc, crcBitLength);

}

return crc ^ xorOut;

}

unsigned long reverseBits(unsigned long value, const unsigned long size) {

unsigned long newValue = 0;

for (unsigned long i = size - 1; i >= 0; i--) {

newValue |= (value & 1) << i;

value >>= 1;

}

return newValue;

}

Slave2.ino

#define BAUD 4800

#define MODE 2

#define TRANSMIT HIGH

#define RECEIVE LOW

#define ID 77

char symbol;

int i = 0;

char\* bufferPtr;

void setup(void) {

Serial.begin(BAUD);

pinMode(MODE, OUTPUT);

digitalWrite(MODE, RECEIVE);

}

void loop() {

i = 0;

symbol = 0;

if (Serial.available()) {

bufferPtr = (char\*)malloc(50 \* sizeof(char));

delay(5);

while (Serial.available()) {

symbol = Serial.read();

if (symbol == ';') {

bufferPtr[i++] = '\0';

i = 0;

int c;

sscanf(bufferPtr, "%d", &c);

if (c != ID) {

while (Serial.available()){Serial.read();}

free(bufferPtr);

return;

}

} else {

bufferPtr[i++] = symbol;

}

}

if (i > 0){

bufferPtr[i++] = '\0';

}

}

if (i > 0) {

if (!strcmp(bufferPtr, "Give")) {

char message[] = "Andriy Matviichuk";

sendMessage(1, message, sizeof(message));

}

}

free(bufferPtr);

//delay(200);

}

void sendMessage(int recevierId, char str[], long sizeOf) {

char buffer[40];

digitalWrite(MODE, TRANSMIT);

long crc = calculateCrcWithTable(str, sizeOf / sizeof(char) - 1, 0x7, 0x00, false, false, 0x00, 8);

delay(60); //1010 ^ 1000 = 0010 => 0010 ^ 1000 = 1010

sprintf(buffer, "%d;%d:%lx!%s\*", recevierId, ID, crc, str);

Serial.print(buffer); // Без спотворень

delay(50);

/\*

str[0] ^= (1<<2); // 2й розряд найстаршого байту даних Slave1

sprintf(buffer, "%d;%d:%lx!%s\*", recevierId, ID, crc, str);

Serial.print(buffer);

delay(50);

str[0] ^= (1<<2); //фікс попередньої операції

\*/

str[strlen(str) - 1] ^= (1<<5); //5й розряд наймолодшого байту даних Slave2

sprintf(buffer, "%d;%d:%lx!%s\*", recevierId, ID, crc, str);

Serial.print(buffer);

delay(50);

str[strlen(str) - 1] ^= (1<<5); //фікс попередньої операції

str[3] ^= (1<<3) | (1<<4); //3й та 4й розряд 3го байту даних Slave2

sprintf(buffer, "%d;%d:%lx!%s\*", recevierId, ID, crc, str);

Serial.print(buffer);

/\*

delay(50);

str[3] ^= (1<<3) | (1<<4); //фікс попередньої операції

str[8] ^= (1<<1) | (1<<2) | (1<<6); //1, 2, 6 розряди 8го байту даних Slave1

sprintf(buffer, "%d;%d:%lx!%s\*", recevierId, ID, crc, str);

Serial.print(buffer);

\*/

delay(60);

digitalWrite(MODE, RECEIVE);

}

long calculateCrcWithTable(const char data[], const unsigned long length, unsigned long crc,

const unsigned long poly, const bool refIn, const bool refOut, const unsigned long xorOut,

const unsigned long crcBitLength) {

unsigned long greatestBit = 1 << (crcBitLength - 1);

unsigned long cutter = (1 << crcBitLength) - 1;

unsigned long table[256];

for (unsigned long i = 0; i < 256; i++) {

unsigned long res = i;

if (crcBitLength > 8) {

res <<= (crcBitLength - 8);

}

for (unsigned int j = 0; j < 8; j++) {

if ((res & greatestBit) != 0) {

res = (res << 1) ^ poly;

}

else {

res <<= 1;

}

}

table[i] = res & cutter;

}

unsigned long shift = (crcBitLength - 8 < 0) ? 0 : (crcBitLength - 8);

for (unsigned long i = 0; i < length; i++) {

unsigned long temp = data[i];

if (refIn) {

temp = reverseBits(temp, 8);

}

crc = table[(crc >> shift) ^ temp] ^ (crc << 8);

crc &= cutter;

}

if (refOut) {

crc = reverseBits(crc, crcBitLength);

}

return crc ^ xorOut;

}

unsigned long reverseBits(unsigned long value, const unsigned long size) {

unsigned long newValue = 0;

for (unsigned long i = size - 1; i >= 0; i--) {

newValue |= (value & 1) << i;

value >>= 1;

}

return newValue;

}

Main.py

from functools import reduce  
import threading  
import sys  
from PyQt5.QtWidgets import QApplication, QTableWidgetItem  
from PyQt5 import uic,QtGui,QtCore  
from PyQt5.QtGui import \*  
from PyQt5.QtGui import QPalette, QColor  
from PyQt5.QtCore import QThread,QTimer  
from PyQt5 import QtWidgets  
import time  
import os  
import ui\_py  
import serial  
  
  
  
ser = ''  
  
  
  
CRC\_BIT\_LENGTH = 8  
POLY = 0x7  
INIT\_CRC = 0x0  
XOR\_OUT = 0x0  
REF\_IN = False  
REF\_OUT = False  
  
  
roll\_text = ''  
  
selected\_slave = 0  
  
received = ''  
  
ser = serial.Serial("COM7", 4600, bytesize=8, parity='N', stopbits=1, timeout=0)  
  
  
if getattr(sys, 'frozen', False):  
 os.chdir(sys.\_MEIPASS)  
  
class Sensors(QThread):  
  
 def \_\_init\_\_(self):  
 QThread.\_\_init\_\_(self)  
  
  
  
  
 def run(self):  
  
 while True:  
 pass   
  
  
class App(QtWidgets.QMainWindow, ui\_py.Ui\_MainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
  
 self.sensor\_detect = Sensors()  
 self.sensor\_detect.start()  
  
 self.setupUi(self)  
 self.start()  
  
 self.timer = QTimer()  
 self.timer.timeout.connect(self.set\_info)  
 self.timer.start(0.1)  
  
 def set\_info(self):  
 check\_serial()  
  
  
 def start(self):  
  
 self.setWindowTitle("lab6")  
 self.show()  
  
 self.pushButton\_2.clicked.connect(lambda: slave\_1())  
 self.pushButton\_3.clicked.connect(lambda: slave\_2())  
  
def slave\_1():  
 global selected\_slave,roll\_text  
 roll\_text = ''  
 ser.write(b'1')  
 selected\_slave = 1  
  
def slave\_2():  
 global selected\_slave,roll\_text  
 roll\_text = ''  
 ser.write(b'2')  
 selected\_slave = 2  
  
  
  
def check\_serial():  
 global selected\_slave, received,roll\_text  
 if ser:  
 data = str(ser.read())[2:-1]  
 if data:  
  
 if received != '':  
 received += data  
 elif data == "#" and received == '':  
 received += data  
  
 if received.endswith("\*"):  
 received = received[1:-1]  
 received = received.split(';')  
  
 crc = int("0x" + received[0], 16)  
  
 result = check\_crc(received[1], crc, CRC\_BIT\_LENGTH, POLY, INIT\_CRC, REF\_IN, REF\_OUT, XOR\_OUT)  
  
 roll\_text = str(roll\_text) + received[1] + '\n' + "Crc check: " + str(result) + '\n'  
  
 if selected\_slave==1:  
 ex.label.setText(roll\_text)  
 elif selected\_slave==2:  
 ex.label\_2.setText(roll\_text)  
  
  
  
 received = ''  
  
  
def check\_crc(data, crc, crc\_bit\_length, poly, init\_crc, in\_ref, res\_ref, final\_xor):  
  
  
  
 if isinstance(data, str):  
  
 data = convert\_ascii\_str\_to\_int\_list(data)  
  
  
  
 poly\_msb = 1 << (crc\_bit\_length - 1)  
 cutter = (1 << crc\_bit\_length) - 1  
  
 new\_crc = init\_crc  
  
  
  
 for i in range(len(data)):  
 temp = data[i]  
 if in\_ref:  
 temp = reverse\_bits(temp, 8)  
 temp <<= crc\_bit\_length - 8  
 new\_crc ^= temp  
  
 for \_ in range(8):  
 if (new\_crc & poly\_msb) != 0:  
 new\_crc = (new\_crc << 1) ^ poly  
 else:  
 new\_crc <<= 1  
 new\_crc &= cutter  
  
 if res\_ref:  
 new\_crc = reverse\_bits(new\_crc, crc\_bit\_length)  
 new\_crc ^= final\_xor  
 return new\_crc == crc  
  
  
def reverse\_bits(number, bits\_quantity=None):  
 number = bin(number)[2:]  
  
 if bits\_quantity is not None:  
 return int(''.join(number[::-1] + '0' \* (bits\_quantity - len(number))), 2)  
 return int(''.join(number[::-1]), 2)  
  
  
def convert\_ascii\_str\_to\_int\_list(data):  
 return [convert\_ascii\_str\_to\_int(i) for i in data]  
  
  
def convert\_ascii\_str\_to\_int(data):  
  
 return reduce(lambda x, y: x | y, [ord(data[::-1][i]) \* ((2 \*\* 8) \*\* i) for i in range(len(data))])  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
  
  
  
 app = QApplication(sys.argv)  
  
 app.setStyle("Fusion")  
  
 # Now use a palette to switch to dark colors:  
  
 palette = QPalette()  
 palette.setColor(QPalette.Window, QColor(53, 53, 53))  
  
 palette.setColor(QPalette.Base, QColor(25, 25, 25))  
 palette.setColor(QPalette.AlternateBase, QColor(53, 53, 53))  
 palette.setColor(QPalette.ToolTipBase, QColor(0, 0, 0))  
 palette.setColor(QPalette.ToolTipText, QColor(255, 255, 255))  
 palette.setColor(QPalette.Text, QColor(255, 255, 255))  
 palette.setColor(QPalette.Button, QColor(53, 53, 53))  
 palette.setColor(QPalette.ButtonText, QColor(255, 255, 255))  
 palette.setColor(QPalette.BrightText, QColor(255, 0, 0))  
 palette.setColor(QPalette.Link, QColor(42, 130, 218))  
 palette.setColor(QPalette.Highlight, QColor(100, 0, 0))  
 palette.setColor(QPalette.HighlightedText, QColor(0, 0, 0))  
 app.setPalette(palette)  
  
  
 ex = App()  
 app.exec\_()

**Висновок:**

В данній лабораторній роботі я навчився розробляти завадостійкий протокол обміну даних в мережі мікроконтроллерів на базі RS-485 .