МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ»

Факультет информатики и
вычислительной техники

Кафедра ИиСП

Отчет
по лабораторной работе № 10

по дисциплине «Машинно-зависимые языки программирования»

Выполнил: студент группы ПС-11
Щеглов Г.С
Проверил: Баев А.А.

г. Йошкар-Ола
2024

Цель работы: научиться работать с бесконтактными считывателями RFID.

Задания на лабораторную работу:

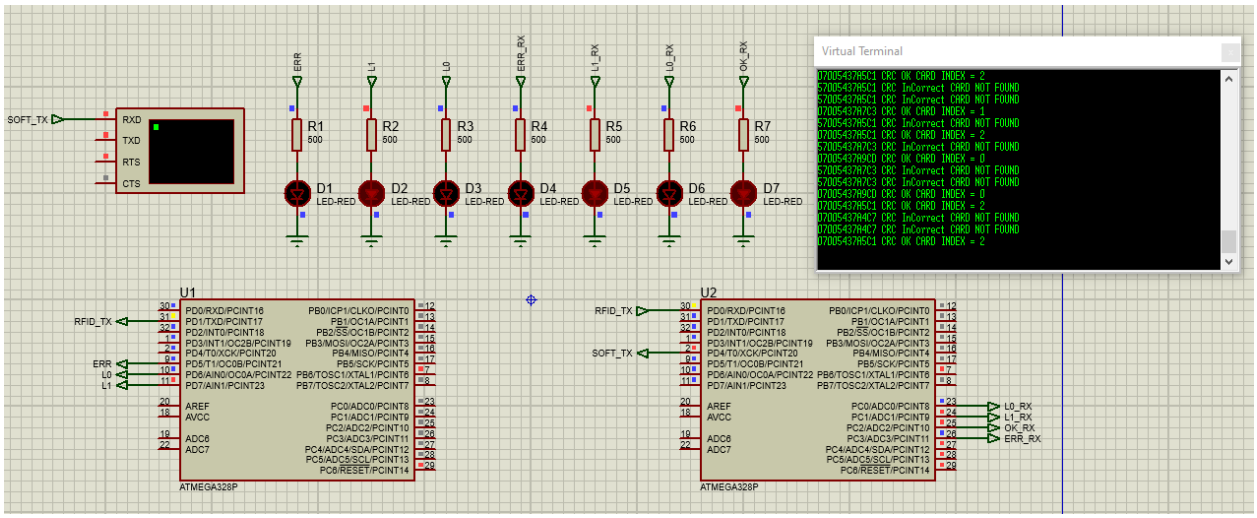
1. Реализовать схему с RFID

1. Теоретические сведения

Учебное пособие - ПРИМЕНЕНИЕ МИКРОКОНТРОЛЛЕРОВ В РАДИОТЕХНИЧЕСКИХ И БИОМЕДИЦИНСКИХ СИСТЕМАХ

2. Практическая часть

Схема, моделирующая RFID:



Код передатчика на C:

```
#define F_CPU 16000000UL
#define BAUDRATE 9600
#include <avr/io.h>
#include <util/delay.h>
#define CARD_DATA_SIZE 14
#define BASE_DELAY 100
#define LED_ERR 5
#define LED0 6
#define LED1 7
#define LEDS_OFF() PORTD &= ~((1 << LED1) | (1 << LED0) | (1 << LED_ERR))

//Массив данных для 4 карт
volatile char cardData[56] = {  0x02, 0x30, 0x37, 0x30, 0x30, 0x35,
0x34, 0x33, 0x37, 0x41, 0x39, 0x43, 0x44,
0x03,  0x02, 0x30, 0x37, 0x30, 0x30, 0x35, 0x34,
0x33, 0x37, 0x41, 0x37, 0x43, 0x33, 0x03,  0x02,
0x30, 0x37, 0x30, 0x30, 0x35, 0x34, 0x33, 0x37,
0x41, 0x35, 0x43, 0x31, 0x03,  0x02, 0x30, 0x37,
0x30, 0x30, 0x35, 0x34, 0x33, 0x37, 0x41, 0x34,
0x43, 0x37, 0x03};

void InitPorts(){
	DDRD = 0xFF;
	PORTD = 0x00; }

void InitUart(){
	UCSR0B = (1 << TXEN0);
	UCSR0C = (1 << UCSZ01 | 1 << UCSZ00);
	UBRR0H = 0;
	UBRR0L = F_CPU/BAUDRATE/16 - 1;
}
```

```
void SendChar(char symbol){
	while(!(UCSR0A & (1<<UDRE0)));
	UDR0 = symbol;
}
void SendPacket(char* dat){
	int i = 0;
	while(i < CARD_DATA_SIZE)
	SendChar(dat[i++]); }
void Leds_On(uint8_t value){
	PORTD |= (value >> 1) << LED1;
	PORTD |= (value & 0x01) << LED0;
}
void Delay_Func(uint8_t iteration){
	//Функция необходима для формирования изменяемой задержки  //так как
_delay_ms() не принимает изменяемые переменные в качестве параметра
	int i = 0;
	while(i++ < iteration)
	_delay_ms(BASE_DELAY);
}
int main(void) {
	InitPorts(); //Инициализация портов
	InitUart(); //Инициализация UART
	uint8_t cardIdx = 0;
	int shift = 0;
	uint8_t iterations =0;
	uint8_t doError = 0;
	char sendData[CARD_DATA_SIZE];
	while (1)  {
		LEDS_OFF(); //Вызов макроса выключения светодиодов
		cardIdx = rand() & 0x03; //Выбор случайной карты
		Leds_On(cardIdx); //Индикация номера кадра
		//Формирование кадра для отправки
		shift = cardIdx * CARD_DATA_SIZE;
		for (int i = 0; i < CARD_DATA_SIZE; i++)
		sendData[i] = cardData[i + shift];
		//Внесение ошибки в кадр
		doError = rand() & 0x01; //
		if(doError) sendData[1] = 0x35;
		PORTD |= doError << LED_ERR; //Индикация наличия ошибки
		//Отправка данных
		SendPacket(sendData);     //Случайная задержка отправки сигнала
		//от 100мс до 1.7 сек
		iterations = rand() & 0x0F;
		Delay_Func(++iterations);
	}
}
```

Код приемника на С:

```
#define F_CPU 16000000UL
#define BAUDRATE 9600
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#define CARD_DATA_SIZE 5
#define BASE_DELAY 100
#define LED0 0
#define LED1 1
#define LED_OK 2
#define LED_ERR 3
#define LEDS_OFF PORTC = ~((1 << LED1) | (1 << LED0) | (1 << LED_ERR) | (1 <<
LED_OK))
#define READY 1
#define RECEIVING 2
#define RECEIVED 3
#define UART_RCV_ON UCSR0B = (1 << RXEN0) | (1 << RXCIE0)
#define UART_RCV_OFF UCSR0B = 0x00
#define TX_SOFT 4
#define SET_0_SOFT PORTD &= ~(1 << TX_SOFT)
#define SET_1_SOFT PORTD |= (1 << TX_SOFT)
volatile char cardData[20] = {0x07, 0x00, 0x54, 0x37, 0xA9,
0x07, 0x00, 0x54, 0x37, 0xA7,
0x07, 0x00, 0x54, 0x37, 0xA5,
0x07, 0x00, 0x54, 0x37, 0xA3};
char arr[15] = {0x02, 0x35, 0x37, 0x30, 0x30, 0x35, 0x34,
0x33, 0x37, 0x41, 0x35, 0x43, 0x31, 0x03, 0x00}; //Буфер приема
volatile uint8_t state = READY;
volatile uint8_t uCnt = 0;
volatile char uValue = 0;
volatile uint8_t cardCount = 4;
ISR(USART_RX_vect){
	uValue = UDR0;
	if(uValue == 0x02)   {
		uCnt = 0;
		state = RECEIVING;
	}
if(state == RECEIVING)   {
	arr[uCnt++] = uValue;
	if(uCnt > 14) uCnt = 0; //Зацикливание записи в массив приема
	}	//Условие окончания приема данных
	if(uValue == 0x03 && uCnt > 0)   {
		state = RECEIVED;
	}
}

void InitPorts(){
	DDRD = 0xFF;
	PORTD = 0x00;
	DDRC = 0xFF;
	PORTC = 0x00; }
void InitUart(){
	UCSR0B = (1 << RXEN0) | (1 << RXCIE0);
	UCSR0C = (1 << UCSZ01 | 1 << UCSZ00);
	UBRR0H = 0;    UBRR0L = F_CPU/BAUDRATE/16 - 1;
}
```

```
void SendCharSoftUART(char symbol)   {
	//BAUDRATE_SOFT = 57600 bps
	//t = 1 / BAUDRATE = 17.36 us
	uint8_t i = 0x01;
	SET_0_SOFT; // Старт бит
	_delay_us(17.36);
	while(i > 0){
		// 8 бит данных
		if(symbol & i) SET_1_SOFT;
		else SET_0_SOFT;
		i <<= 1; // Младший бит первый (LSB)
		_delay_us(17.36);
	  }
	SET_1_SOFT; // Стоп бит
	_delay_us(17.36);
}
void SendStringSoftUART(char * buffer)   {
	while(*buffer != 0){
	SendCharSoftUART(*buffer++);
	}
}
void Leds_On(uint8_t value){
	PORTC |= (value >> 1) << LED1;
	PORTC |= (value & 0x01) << LED0;
}

uint8_t ConvertCharToByte(char val)   {
	switch(val)   {
		case '0': return 0x00;
		case '1': return 0x01;
		case '2': return 0x02;
		case '3': return 0x03;
		case '4': return 0x04;
		case '5': return 0x05;
		case '6': return 0x06;
		case '7': return 0x07;
		case '8': return 0x08;
		case '9': return 0x09;
		case 'A': return 0x0A;
		case 'B': return 0x0B;
		case 'C': return 0x0C;
		case 'D': return 0x0D;
		case 'E': return 0x0E;
		case 'F': return 0x0F;
		default: return 0x00;
	}
}
int main(void) {
	//Инициализация
	InitPorts();
	InitUart();
	SET_1_SOFT;
	SendStringSoftUART("Hello\r\n");
	sei();
	int cardIdxEst = -1; // Вычисленный индекс карты
	int curMatch = -1;
	char tempArr[CARD_DATA_SIZE + 1];
	uint8_t CRC = 0; // Вычисленное CRC
	uint8_t rCRC = 0; // Принятое CRC
	int tmp = 0;
```

```
	int shift;
	while (1)    {
		//Ожидание приема кадра
		if(state == RECEIVED)   {
			state = READY; // Сброс состояния в режим ожидания
			// Проверка CRC, извлечение информации из принятого массива и
запись во временный массив
			rCRC = (ConvertCharToByte(arr[11]) << 4);
			rCRC |= ConvertCharToByte(arr[12]);
			CRC = 0;
			for (int i = 0; i < CARD_DATA_SIZE; i++)   {
			tmp = (ConvertCharToByte(arr[2 * i + 1]) << 4) |
ConvertCharToByte(arr[2 * i + 2]);
			tempArr[i] = tmp & 0xFF;
			CRC ^= tempArr[i];
					}
		LEDS_OFF;
		//Отключаем аппаратный UART, чтобы не влияло на тайминг программного
UART
		UART_RCV_OFF;
		SendStringSoftUART(arr);
		//Проверка контрольной суммы
		if((rCRC == CRC) & (rCRC != 0))   {
			PORTC |=  1 << LED_OK;
			SendStringSoftUART(" CRC OK ");
	}else {
		PORTC |=  1 << LED_ERR;
		SendStringSoftUART(" CRC InCorrect ");
		}
		   //Поиск совпадения по номеру карты
		cardIdxEst = -1;
		shift = 0;
		for(int i = 0; i < cardCount; i++)   {
			curMatch = tempArr[CARD_DATA_SIZE - 1] ^ cardData[shift +
CARD_DATA_SIZE - 1];
			for (int j = CARD_DATA_SIZE - 2; j >= 0; j--)   {
				curMatch += tempArr[j] ^ cardData[shift + j];
				if(curMatch > 0) break;
			}
			shift += CARD_DATA_SIZE;
			if(curMatch == 0) {
				cardIdxEst = i; break;
				}
				}
					//Индикация индекса карты
			Leds_On(cardIdxEst & 0x03);
			if(cardIdxEst >= 0)   {
				SendStringSoftUART("CARD INDEX = ");
				SendCharSoftUART(cardIdxEst + 0x30);
				}
			else SendStringSoftUART("CARD NOT FOUND ");
			SendStringSoftUART("\r\n");
			UART_RCV_ON; //Включаем аппаратный UART
		}
	}
}
```

Выводы: мы научились работать с RFID