#include "src/U8glib2.h"

#include <OneWire.h>

#include "pitches.h"

#define iButtonPin A3 // Линия data ibutton

#define ACpin 6 // Вход аналогового компаратора 3В для Cyfral/Metacom

#define btnPin 10 // Кнопка переключения режима чтение/запись

#define buttonNextPin 4 // Кнопка вперед

#define buttonPrevPin 5 // Кнопка

#define buttonMorePin 8 // Кнопка

#define buttonLessPin 7 // Кнопка

#define speakerPin 9 // Спикер, он же buzzer, он же beeper

// ----------------------------- переменные для работы с ключом -----------------------------

OneWire ibutton (iButtonPin);

byte addr[8]; // временный буфер

byte keyID[8] = {255,255,255,255,255,255,255,255};// ID ключа для записи

bool readflag = true; // флаг сигнализируе, что данные с ключа успечно прочианы в ардуино

bool writeflag = false; // режим запись/чтение

enum EmRWType {TM01, RW1990\_1, RW1990\_2, TM2004}; // тип болванки

enum EmkeyType {keyUnknown, keyDallas, keyTM2004, keyCyfral, keyMetacom}; // тип оригинального ключа

EmkeyType keyType;

// выбранный номер регистра подсвечиваемый на диплее

uint8\_t selectedRegisterIndex = 0;

// -----------------------------------------------------------------------------------

// ----------------------------- УСТАНОВКА И ГЛАВНЫЙ ЦИКЛ ----------------------------

// -----------------------------------------------------------------------------------

void setup() {

pinMode(btnPin, INPUT\_PULLUP); // включаем чтение и подягиваем пин кнопки режима к +5В

pinMode(buttonNextPin, INPUT\_PULLUP);

pinMode(buttonPrevPin, INPUT\_PULLUP);

pinMode(buttonMorePin, INPUT\_PULLUP);

pinMode(buttonLessPin, INPUT\_PULLUP);

pinMode(speakerPin, OUTPUT);

pinMode(ACpin, INPUT); // Вход аналогового компаратора 3В для Cyfral

Sd\_StartOK();

prepeareDisplay();

// выводим надпись режим чтения (todo и возможно что кода еще нет)

out\_titleString("READ");

out\_redraw();

}

// переменные кнопок

bool btnBuffer;

bool btnClick;

bool preBtnPinSt = true;

bool buttonNextPressState = false;

uint16\_t buttonNextPreviousState = false;

bool buttonPrevPressState = false;

uint16\_t buttonPrevPreviousState = false;

bool buttonMorePressState = false;

uint16\_t buttonMorePressTime = 0;

uint16\_t buttonMorePreviousState = false;

bool buttonLessPressState = false;

uint16\_t buttonLessPressTime = 0;

uint16\_t buttonLessPreviousState = false;

void loop() {

// основная кнопка

btnBuffer = digitalRead(btnPin);

btnClick =((btnBuffer == LOW) && (preBtnPinSt == HIGH));

preBtnPinSt = btnBuffer;

// кнопка вперёд

// считываем состояние

btnBuffer = !digitalRead(buttonNextPin);

buttonNextPressState = ((btnBuffer == HIGH) && (buttonNextPreviousState == LOW));

buttonNextPreviousState = btnBuffer;

// если нажата

if(buttonNextPressState ){

selectedRegisterIndex =

(selectedRegisterIndex == 7)? (0):(selectedRegisterIndex + 1);

out\_redraw();

}

// кнопка назад

// считываем состояние

btnBuffer = !digitalRead(buttonPrevPin);

buttonPrevPressState = ((btnBuffer == HIGH) && (buttonPrevPreviousState == LOW));

buttonPrevPreviousState = btnBuffer;

// если нажата

if(buttonPrevPressState ){

selectedRegisterIndex =

(selectedRegisterIndex == 0)? (7):(selectedRegisterIndex - 1);

out\_redraw();

}

if(!digitalRead(buttonMorePin) && !digitalRead(buttonLessPin)){

if (readflag == true) write2iBtn();

else { // сюда испонение не должно попасть

out\_titleString("ERROR");out\_redraw();

Sd\_ErrorBeep();

out\_titleString("READ");out\_redraw();

}

}

// кнопка Больше

// считываем состояние

btnBuffer = !digitalRead(buttonMorePin);

buttonNextPressState = ((btnBuffer == HIGH) && (buttonNextPreviousState == LOW));

buttonNextPreviousState = btnBuffer;

// если нажата

if(buttonNextPressState ){

keyID[selectedRegisterIndex] =

(keyID[selectedRegisterIndex] == 255)? (0):(keyID[selectedRegisterIndex] + 1);

out\_redraw();

}

// кнопка меньше

// считываем состояние

btnBuffer = !digitalRead(buttonLessPin);

buttonNextPressState = ((btnBuffer == HIGH) && (buttonNextPreviousState == LOW));

buttonNextPreviousState = btnBuffer;

// если нажата

if(buttonNextPressState ){

keyID[selectedRegisterIndex] =

(keyID[selectedRegisterIndex] == 0)? (255):(keyID[selectedRegisterIndex] - 1);

out\_redraw();

}

// если кнопка нажата

if ((Serial.read() == 't') || btnClick) { // переключаель режима чтение/запись

// если чуть раньше код был успешно прочтен

if (readflag == true) {

writeflag = !writeflag;

// выводим на экран и в порт, текущий режим

if (writeflag) out\_titleString("WRITE");

else out\_titleString("READ");

out\_redraw();

} else {

// если предыдущий код не был прочтен и была нажата кнопка возвращаем прибор в режим

// без кода, готовый к чтению (точнее просто выводим об этом информацию)

out\_titleString("NO CODE"); out\_redraw();

Sd\_ErrorBeep();

out\_titleString("READ"); out\_redraw();

}

}

// если режим чтения и НАЙДЕН и успешно ПРОЧИТАН ключ cyfral

if ((!writeflag) && (searchCyfral())) { // запускаем поиск cyfral

readflag = true;

out\_titleString("SUCESS!");out\_redraw();

Sd\_ReadOK();

out\_titleString("READ");out\_redraw();

}

// если найден ключ dallas

if (ibutton.search(addr)) { // запускаем поиск dallas

// если режим чтения

if (!writeflag){

out\_titleString("PROCESSING...");out\_redraw();

readflag = readiBtn(); // чиаем ключ dallas

if (readflag) {

out\_titleString("SUCESS!");out\_redraw();

Sd\_ReadOK();

}else{

Serial.println("CRC is not valid!");

out\_titleString("ERROR");out\_redraw();

Sd\_ErrorBeep();

}

out\_titleString("READ");out\_redraw();

}else{

if (readflag == true) write2iBtn();

else { // сюда испонение не должно попасть

out\_titleString("ERROR");out\_redraw();

Sd\_ErrorBeep();

out\_titleString("READ");out\_redraw();

}

}

}else{

ibutton.reset\_search();

}

}

// -----------------------------------------------------------------------------------

// ----------------------------- БАЗЗЕР -----------------------------

// -----------------------------------------------------------------------------------

void Sd\_ReadOK() { // звук ОК

for (int i=400; i<6000; i=i\*1.5) { tone(speakerPin, i); delay(20); }

noTone(speakerPin);

}

void Sd\_WriteStep(){ // звук "очередной шаг"

for (int i=2500; i<6000; i=i\*1.5) { tone(speakerPin, i); delay(10); }

noTone(speakerPin);

}

void Sd\_ErrorBeep() { // звук "ERROR"

for (int j=0; j <3; j++){

for (int i=1000; i<2000; i=i\*1.1) { tone(speakerPin, i); delay(10); }

delay(50);

for (int i=1000; i>500; i=i\*1.9) { tone(speakerPin, i); delay(10); }

delay(50);

}

noTone(speakerPin);

}

void Sd\_StartOK(){ // звук "Успешное включение"

tone(speakerPin, NOTE\_A7); delay(100);

tone(speakerPin, NOTE\_G7); delay(100);

tone(speakerPin, NOTE\_E7); delay(100);

tone(speakerPin, NOTE\_C7); delay(100);

tone(speakerPin, NOTE\_D7); delay(100);

tone(speakerPin, NOTE\_B7); delay(100);

tone(speakerPin, NOTE\_F7); delay(100);

tone(speakerPin, NOTE\_C7); delay(100);

noTone(speakerPin);

}

// -----------------------------------------------------------------------------------

// ----------------------------- ДИСПЛЕЙ -----------------------------

// -----------------------------------------------------------------------------------

// переменная дисплея

U8GLIB\_SSD1306\_128X64 u8g(U8G\_I2C\_OPT\_NONE|U8G\_I2C\_OPT\_DEV\_0); // I2C / TWI

// flip screen, if required

//u8g.setRot180();

char lastTitle[20] = "";

char lastCodeString[40] = "";

void prepeareDisplay(){

u8g.setFont(u8g\_font\_6x10);

u8g.setFontRefHeightExtendedText();

u8g.setDefaultForegroundColor();

u8g.setFontPosTop();

}

void out\_titleString(const char\* str){// вывод надписи в заголовок. максимум 11 символов

strncpy(lastTitle, str, strlen(str));

\*(lastTitle + strlen(str)) = '\0';

//out\_redraw();

}

uint8\_t resultStartPos = 0;

uint8\_t resultEndPos = 1;

uint8\_t abc = 0;

void out\_redraw(){

// экран состоит из нескольких страниц, на каждой из которых надо выполнять одни и те же команды

// большой символ помещается на 4х страницах, маленький на 2х

u8g.firstPage();

// вывод текущего заголовка

u8g.setScale2x2();

u8g.firstPage();

for(uint8\_t page = 0; page < 2; page++){

u8g.drawStr(0, 0, lastTitle);

u8g.nextPage();

}

u8g.undoScale();

// если есть корректный код

if(readflag){

// создание строки кода

\*(lastCodeString) = '\0';

for(uint8\_t keyPart = 0; keyPart < 8; keyPart++){

itoa(\*(keyID + keyPart), lastCodeString + strlen(lastCodeString), DEC);

if(keyPart != 7)

strcat(lastCodeString, ":");

}

// вывод кода

char firstHalf[40] = "";

char secondHalf[40] = "";

char buff[10] = "";

uint8\_t firstLen = 0;

\*(firstHalf) = '\0';

for(uint8\_t keyPart = 0; keyPart < 8; keyPart++){

itoa(\*(keyID + keyPart), buff, DEC);

if(keyPart >= 4){

strcat(secondHalf, buff);

if(keyPart != 7)

strcat(secondHalf, ":");

}else{

firstLen += strlen(buff)+1;

strcat(firstHalf, buff);

strcat(firstHalf, ":");

}

}

u8g.nextPage();

u8g.drawStr(0, 24, firstHalf);

boolean outOfBounds = false;

int8\_t a;

// вичисление позиции подчеркивания

char temp[sizeof(lastCodeString)] = "";

strcpy(temp, lastCodeString);

\*(temp+sizeof(lastCodeString)) = '\0';

resultStartPos = 0;

resultEndPos = 1;

for(uint8\_t numberPoz = 0; numberPoz < selectedRegisterIndex && !outOfBounds; numberPoz++){

a = whatApos(temp);

if(a == -1){

outOfBounds = true;

}

resultStartPos += a +1;

strncpy(temp, (lastCodeString + resultStartPos), strlen(lastCodeString));

\*(temp + strlen(lastCodeString) - resultStartPos) = '\0';

}

u8g.nextPage();

if(!outOfBounds){

a = whatApos(temp);

if(a == -1){

resultEndPos = resultStartPos+1;

}else{

resultEndPos = a + resultStartPos;

}

if(resultStartPos < firstLen){

// выводим его

abc = 0;

while(abc < resultStartPos){

temp[abc] = ' ';

abc++;

}

while(abc < resultEndPos){

temp[abc] = '\_';

abc++;

}

\*(temp+abc) = '\0';

u8g.drawStr(0, 26, temp);

}

}

u8g.drawStr(0, 36, secondHalf);

u8g.nextPage();

u8g.drawStr(0, 36, secondHalf);

if(!outOfBounds && resultStartPos >= firstLen){

resultStartPos -= firstLen;

resultEndPos -= firstLen;

// выводим его

abc = 0;

while(abc < resultStartPos){

temp[abc] = ' ';

abc++;

}

while(abc < resultEndPos){

temp[abc] = '\_';

abc++;

}

\*(temp+abc) = '\0';

u8g.drawStr(0, 38, temp);

}

}else{

u8g.nextPage();

u8g.drawStr(0, 24, "No Code");

}

// очищаем содержимое остальных страниц

while(u8g.nextPage()){}

}

int8\_t whatApos(char\* str){

int8\_t i = 0;

while (str[i] && str[i] != ':') ++i;

return ':' == str[i] ? i : -1;

}

// -----------------------------------------------------------------------------------

// ----------------------------- МЕТОДЫ РАБОТЫ С КЛЮЧАМИ -----------------------------

// -----------------------------------------------------------------------------------

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* dallas \*\*\*\*\*\*\*\*\*\*\*\*\*\*

EmRWType getRWtype(){

byte answer;

// TM01 это неизвестный тип болванки, делается попытка записи TM-01 без финализации для dallas или c финализацией под cyfral или metacom

// RW1990\_1 - dallas-совместимые RW-1990, RW-1990.1, ТМ-08, ТМ-08v2

// RW1990\_2 - dallas-совместимая RW-1990.2

// TM2004 - dallas-совместимая TM2004 в доп. памятью 1кб

// пробуем определить RW-1990.1

ibutton.reset(); ibutton.write(0xD1); // проуем снять флаг записи для RW-1990.1

ibutton.write\_bit(1); // записываем значение флага записи = 1 - отключаем запись

delay(10); pinMode(iButtonPin, INPUT);

ibutton.reset(); ibutton.write(0xB5); // send 0xB5 - запрос на чтение флага записи

answer = ibutton.read();

//Serial.print("\n Answer RW-1990.1: "); Serial.println(answer, HEX);

if (answer == 0xFE){

Serial.println(" Type: dallas RW-1990.1 ");

return RW1990\_1; // это RW-1990.1

}

// пробуем определить RW-1990.2

ibutton.reset(); ibutton.write(0x1D); // проуем установить флаг записи для RW-1990.2

ibutton.write\_bit(1); // записываем значение флага записи = 1 - включаем запись

delay(10); pinMode(iButtonPin, INPUT);

ibutton.reset(); ibutton.write(0x1E); // send 0x1E - запрос на чтение флага записи

answer = ibutton.read();

//Serial.print("\n Answer RW-1990.2: "); Serial.println(answer, HEX);

if (answer == 0xFE){

ibutton.reset(); ibutton.write(0x1D); // возвращаем оратно запрет записи для RW-1990.2

ibutton.write\_bit(0); // записываем значение флага записи = 0 - выключаем запись

delay(10); pinMode(iButtonPin, INPUT);

Serial.println(" Type: dallas RW-1990.2 ");

return RW1990\_2; // это RW-1990.2

}

//}

// пробуем определить TM-2004

ibutton.reset(); ibutton.write(0x33); // посылаем команду чтения ROM для перевода в расширенный 3-х байтовый режим

for ( byte i=0; i<8; i++) ibutton.read(); //читаем данные ключа

ibutton.write(0xAA); // пробуем прочитать регистр статуса для TM-2004

ibutton.write(0x00); ibutton.write(0x00); // передаем адрес для считывания

answer = ibutton.read(); // читаем CRC комманды и адреса

//Serial.print("TM2004 CRC: "); Serial.println(answer, HEX);

byte m1[3] = {0xAA, 0,0}; // вычисляем CRC комманды

if (OneWire::crc8(m1, 3) == answer) {

answer = ibutton.read(); // читаем регистр статуса

//Serial.print(" status: "); Serial.println(answer, HEX);

Serial.println(" Type: dallas TM2004");

ibutton.reset();

return TM2004; // это Type: TM2004

}

ibutton.reset();

Serial.println(" Type: dallas unknown, trying TM-01! ");

return TM01; // это неизвестный тип DS1990, нужно перебирать алгоритмы записи (TM-01)

}

bool write2iBtnTM2004(){ // функция записи на TM2004

byte answer; bool result = true;

ibutton.reset();

ibutton.write(0x3C); // команда записи ROM для TM-2004

ibutton.write(0x00); ibutton.write(0x00); // передаем адрес с которого начинается запись

boolean isLableShowed = true;

for (byte i = 0; i<8; i++){

// мигание экраном

if(isLableShowed){

out\_titleString("DONT REMOVE");out\_redraw();

}else{

out\_titleString("");out\_redraw();

}

isLableShowed = !isLableShowed;

ibutton.write(keyID[i]);

answer = ibutton.read();

//if (OneWire::crc8(m1, 3) != answer){result = false; break;} // crc не верный

delayMicroseconds(600); ibutton.write\_bit(1); delay(50); // испульс записи

pinMode(iButtonPin, INPUT);

Serial.print('\*');

Sd\_WriteStep();

if (keyID[i] != ibutton.read()) { result = false; break;} //читаем записанный байт и сравниваем, с тем что должно записаться

}

if (!result){

ibutton.reset();

Serial.println(" The key copy faild");

out\_titleString("ERROR");out\_redraw();

Sd\_ErrorBeep();

out\_titleString("WRITE");out\_redraw();

return false;

}

ibutton.reset();

Serial.println(" The key has copied successesfully");

Sd\_ReadOK();

out\_titleString("SUCESS!");out\_redraw();

delay(500);

out\_titleString("WRITE");out\_redraw();

return true;

}

bool write2iBtnRW1990\_1\_2\_TM01(EmRWType rwType){ // функция записи на RW1990.1, RW1990.2, TM-01C(F)

byte rwCmd, rwFlag = 1;

switch (rwType){

case TM01: rwCmd = 0xC1; break; //TM-01C(F)

case RW1990\_1: rwCmd = 0xD1; rwFlag = 0; break; // RW1990.1 флаг записи инвертирован

case RW1990\_2: rwCmd = 0x1D; break; // RW1990.2

}

ibutton.reset(); ibutton.write(rwCmd); // send 0xD1 - флаг записи

ibutton.write\_bit(rwFlag); // записываем значение флага записи = 1 - разрешить запись

delay(10); pinMode(iButtonPin, INPUT);

ibutton.reset(); ibutton.write(0xD5); // команда на запись

boolean isLableShowed = true;

for (byte i = 0; i<8; i++){

// мигание экраном

if(isLableShowed){

out\_titleString("DONT REMOVE");

}else{

out\_titleString("");

}

out\_redraw();

isLableShowed = !isLableShowed;

if (rwType == RW1990\_1) BurnByte(~keyID[i]); // запись происходит инверсно для RW1990.1

else BurnByte(keyID[i]);

Serial.print('\*');

Sd\_WriteStep();

}

ibutton.write(rwCmd); // send 0xD1 - флаг записи

ibutton.write\_bit(!rwFlag); // записываем значение флага записи = 1 - отключаем запись

delay(10); pinMode(iButtonPin, INPUT);

if (!dataIsBurningOK()){ // проверяем корректность записи

Serial.println(" The key copy faild");

out\_titleString("ERROR");out\_redraw();

Sd\_ErrorBeep();

out\_titleString("WRITE");out\_redraw();

return false;

}

Serial.println(" The key has copied successesfully");

if ((keyType == keyMetacom)||(keyType == keyCyfral)){ //переводим ключ из формата dallas

ibutton.reset();

if (keyType == keyCyfral) ibutton.write(0xCA); // send 0xCA - флаг финализации Cyfral

else ibutton.write(0xCB); // send 0xCA - флаг финализации metacom

ibutton.write\_bit(1); // записываем значение флага финализации = 1 - перевезти формат

delay(10); pinMode(iButtonPin, INPUT);

}

Sd\_ReadOK();

out\_titleString("SUCESS!");out\_redraw();

delay(500);

out\_titleString("WRITE");out\_redraw();

return true;

}

void BurnByte(byte data){

for(byte n\_bit=0; n\_bit<8; n\_bit++){

ibutton.write\_bit(data & 1);

delay(5); // даем время на прошивку каждого бита до 10 мс

data = data >> 1; // переходим к следующему bit

}

pinMode(iButtonPin, INPUT);

}

bool dataIsBurningOK(){

byte buff[8];

if (!ibutton.reset()) return false;

ibutton.write(0x33);

ibutton.read\_bytes(buff, 8);

byte Check = 0;

for (byte i = 0; i < 8; i++)

if (keyID[i] == buff[i]) Check++; // сравниваем код для записи с тем, что уже записано в ключе.

if (Check != 8) return false; // если коды совпадают, ключ успешно скопирован

return true;

}

bool write2iBtn(){

int Check = 0, CheckSumNewKey = 0;

Serial.print("The new key code is: ");

for (byte i = 0; i < 8; i++) {

Serial.print(addr[i], HEX); Serial.print(":");

CheckSumNewKey += keyID[i];

if (keyID[i] == addr[i]) Check++; // сравниваем код для записи с тем, что уже записано в ключе.

}

if (Check == 8) { // если коды совпадают, ничего писать не нужно

Serial.println(" it is the same key. Writing in not needed.");

out\_titleString("ALREADY WRT");out\_redraw();

Sd\_ErrorBeep();

delay(500);

out\_titleString("WRITE");out\_redraw();

return false;

}

byte rwType = getRWtype(); // определяем тип RW-1990.1 или 1990.2 или TM-01

Serial.print("\n Burning iButton ID: ");

if (rwType == TM2004) return write2iBtnTM2004(); //шьем TM2004

else return write2iBtnRW1990\_1\_2\_TM01(rwType); //пробуем прошить другие форматы

}

bool readiBtn(){

for (byte i = 0; i < 8; i++) {

Serial.print(addr[i], HEX); Serial.print(":");

keyID[i] = addr[i]; // копируем прочтенный код в ReadID

}

if (addr[0] == 0x01) { // это ключ формата dallas

keyType = keyDallas;

if (getRWtype() == TM2004) {

//Serial.println(" Type: dallas TM2004");

keyType = keyTM2004;

} //else Serial.println(" Type: dallas RW1990.x");

if (OneWire::crc8(addr, 7) != addr[7]) {

return false;

}

Sd\_ReadOK();

return true;

}

if ((addr[0]>>4) == 0x0E) Serial.println(" Type: unknown family dallas. May be cyfral in dallas key.");

else Serial.println(" Type: unknown family dallas");

keyType = keyUnknown;

return true;

}

//\*\*\*\*\*\*\*\*\*\*\*\* Cyfral \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned long pulseAComp(bool pulse, unsigned long timeOut = 20000){ // pulse HIGH or LOW

bool AcompState;

unsigned long tStart = micros();

do {

AcompState = ACSR & \_BV(ACO); // читаем флаг компаратора

if (AcompState == pulse) {

tStart = micros();

do {

AcompState = ACSR & \_BV(ACO); // читаем флаг компаратора

if (AcompState != pulse) return (long)(micros() - tStart);

} while ((long)(micros() - tStart) < timeOut);

return 0; //таймаут, импульс не вернуся оратно

} // end if

} while ((long)(micros() - tStart) < timeOut);

return 0;

}

void ACsetOn(){

ADCSRA &= ~(1<<ADEN); // выключаем ADC

ADCSRB |= (1<<ACME); //включаем AC

ADMUX = 0b00000011; // подключаем к AC Линию A3

}

bool read\_cyfral(byte\* buf, byte CyfralPin){

unsigned long ti; byte j = 0;

digitalWrite(CyfralPin, LOW); pinMode(CyfralPin, OUTPUT); //отклчаем питание от ключа

delay(200);

ACsetOn(); //analogComparator.setOn(0, CyfralPin);

pinMode(CyfralPin, INPUT\_PULLUP); // включаем пиание Cyfral

for (byte i = 0; i<36; i++){ // чиаем 36 bit

ti = pulseAComp(HIGH);

if ((ti == 0) || (ti > 200)) break; // not Cyfral

//if ((ti > 20)&&(ti < 50)) bitClear(buf[i >> 3], 7-j);

if ((ti > 50) && (ti < 200)) bitSet(buf[i >> 3], 7-j);

j++; if (j>7) j=0;

}

if (ti == 0) return false;

if ((buf[0] >> 4) != 0b1110) return false; /// not Cyfral

byte test;

for (byte i = 1; i<4; i++){

test = buf[i] >> 4;

if ((test != 1)&&(test != 2)&&(test != 4)&&(test != 8)) return false;

test = buf[i] & 0x0F;

if ((test != 1)&&(test != 2)&&(test != 4)&&(test != 8)) return false;

}

return true;

}

bool searchCyfral(){

for (byte i = 0; i < 8; i++) addr[i] = 0;

bool rez = read\_cyfral(addr, iButtonPin);

if (!rez) return false;

keyType = keyCyfral;

for (byte i = 0; i < 8; i++) {

Serial.print(addr[i], HEX); Serial.print(":");

keyID[i] = addr[i]; // копируем прочтенный код в ReadID

}

Serial.println(" Type: Cyfral ");

return true;

}