Всероссийская конференция «Юные техники и изобретатели»

Государственная Дума Федерального Собрания

Российской Федерации

«Устройство терморегуляции помещений на базе технологии 1-Wire»

Выполнил:
Боровченко Эдуард Николаевич
ученик 9 класса НОУ «Лицей № 36 ОАО «РЖД»

Руководитель:
Ретивых Владимир Викторович
педагог дополнительного образования
НОУ «Лицей № 36 ОАО «РЖД»

Москва, 2015

ОГЛАВЛЕНИЕ

АННОТАЦИЯ

Проект относится к сфере инженерных разработок в области микроэлектроники. В данной работе описаны этапы проектирования одного из элементов системы «умного дома» – электронного терморегулятора на базе микроконтроллера, цифрового термодатчика и сигнального реле. В статье дано подробное описание каждого этапа проектирования, которое включает также основные сведения об архитектуре микроконтроллера PIC18F2550, работе с семисегментным индикатором, сдвиговым регистром 74HC595, интерфейсом 1-Wire, необходимым программным обеспечением. Изначальная версия проекта выполнена на демонстрационной плате на базе микроконтроллера PIC18F2550. На ней и в среде-симуляторе отлажены необходимые программные коды для работы с индикатором и термодатчиком DS18B20. Все листинги кодов и иллюстрации приведены по ходу изложения.

ВВЕДЕНИЕ

Бурное развитие информационных технологий и интеллектуальных систем в современном мире открывает для разработчиков электроники множество решений для широкого круга задач. Большую популярность в последние годы приобретают системы, так называемого «умного» дома. Они позволяют пользователю оперативно обслуживать множество современных бытовых приборов, управлять распределением электроэнергии, водных ресурсов, контролировать теплопотребление. Экономия энергоресурсов сегодня является не только общемировой практикой, но и позволяет конкретному потребителю более эффективно вести хозяйственную деятельность.

Системы «умного» дома, представленные на рынке, крайне разнообразны по своему исполнению и функционалу: имеются как «монолитные», так и рассредоточенные приборы и сети приборов. Наиболее часто можно встретить приборы, сочетающие следующие функции:

- управление освещением помещений (диммеры);
- контроль утечек воды и газа;
- контроль теплопотерь;
- охранно-пожарная сигнализация.

Это типичные системы, используемые как в городских, так и в загородных домах. Помимо этих функций можно встретить и такие: управление звуком акустических систем в помещении (системы «Мультирум»), удалённое управление бытовыми приборами через Интернет или посредством мобильной связи, контроль доступа на территорию объекта (видеонаблюдение, домофон, распознавание лиц и др.) и многие другие.

Данная работа посвящена разработке одного из наиболее популярных элементов системы «умного» дома – электронного терморегулятора. Вариантов его реализации существует несколько, например, с помощью термопары или терморезистора. Мы же в своей разработке используем технологию, предложенную американской компанией Dallas Semiconductors (ныне часть корпорации Maxim Integrated) в 1971 году, однопроводной связи 1-Wire®.

АКТУАЛЬНОСТЬ РАБОТЫ

Актуальность работы состоит в приобретении навыков работы с электронными устройствами и их моделированием в средах программирования и автоматического проектирования, а также знакомстве с проблемами обеспечения комфорта в жилых помещениях и сопутствующими их решению нормами и правилами.

ЦЕЛИ И ЗАДАЧИ

Целью работы является создание простого терморегулятора на базе микроконтроллера PIC фирмы Microchip. Для реализации цели поставлены следующие задачи:

- приобрести общие сведения о программировании и архитектуре микроконтроллеров семейства PIC;
- освоить язык программирования C в необходимом для программирования микроконтроллеров объёме;
- освоить протокол 1-Wire;
- написать программу для работы с индикатором и датчиком температуры;
- отладить программу в симуляторе Proteus 8.0;
- ознакомиться с действующими нормами и правилами, касающимися климат-контроля жилых помещений.

ОСНОВНОЕ СОДЕРЖАНИЕ

1. Общие сведения.

Устройство терморегуляции мы построим из трёх основных частей: управляющего микроконтроллера (далее МК), набора температурных датчиков и блока реле. В качестве вычислительного ядра возьмём микросхему PIC18F2550, производства компании Microchip, а в качестве датчика – DS18B20, производства компании Maxim. Здесь мы приведём основные сведения о работе с МК и языке С, в отдельной главе опишем протокол передачи данных 1-Wire®. Также в работе содержится информация о разработке печатной платы (разводке, травлении и монтаже), соответствующем программном обеспечении, программе-симуляторе. В работе представлены и блок-схемы программ, и листинги кодов.

Первой задачей, поставленной для достижения целей проекта, было знакомство с МК и изучение азов его программирования. Для наглядного изучения и отладки, первым делом, была спаяна демо-плата, содержащая МК и связанный с ним через сдвиговый регистр семисегментный светодиодный индикатор. Подробный рассказ о схемотехнике платы выходит за рамки вводной части. Об этом речь пойдёт в отдельной главе.

Микроконтроллер представляет собой микросхему, состоящую из вычислительного ядра и набора периферийных модулей, функционал которых выведен на ножки (рис. 1). Выбранный нами контроллер имеет следующие характеристики: частота – до 48 МГц (48 млн операций в секунду), память программ (FLASH-память) – 32 кбайт, память данных (ОЗУ) – 2 кбайт. Периферийные модули PIC18F2550: энергонезависимая память (EEPROM), универсальный синхронно-асинхронный приёмопередатчик (USART), интерфейс USB, аналого-цифровой преобразователь (АЦП), модуль захвата и сравнения, широтно-импульсный модулятор, расширенный последовательный и параллельный порты, компаратор, блок таймеров.

Ядро и периферия управляются *регистрами* – элементарными ячейками памяти, хранящими 1 байт информации. Все регистры управления (*Service Function Registers, SFR*) подробно описаны в технической документации к контроллеру.

Настройка ядра осуществляется *конфигурационными словами*, которые прописываются в контроллер прежде всей остальной инициализации. В них содержится информация о частоте ядра, источнике этой частоты (внутренняя RC-цепь, внешний кварцевый резонатор и т.д.), разрешении внешнего сброса, защите данных и доступе к памяти и др.

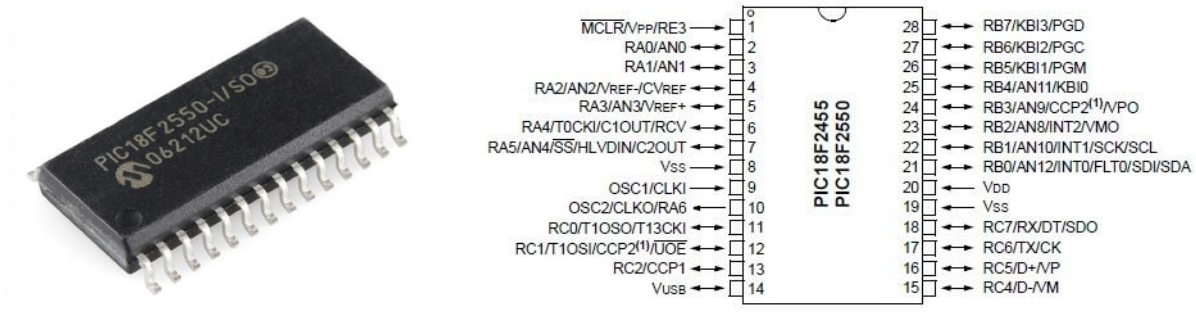

Рисунок 1. Микроконтроллер PIC18F2550 и карта его выводов.

Для работы с периферией сначала настраиваются *порты ввода-вывода*, т.е. модули, обрабатывающие информацию с ножек МК. Как видно из рисунка 1, ножки имеют названия RA0 – RA5 (порт A), RB0 – RB7 (порт B), RC0 – RC7 (порт C), RE3 (порт E). Через слэш перечислены функции, доступные этой ножке. Направление ввода-вывода задаётся с помощью регистров TRIS*x*, где *x* принимает значения A, B, C, E: если бит регистра, соответствующий ножке (например, TRISA1 управляет ножкой RA1), установлен (логическая единица), то ножка настроена на вход (приём данных), если очищен (логический нуль), – на выход (передача). Для данного контроллера входы бывают *аналоговыми* и *цифровыми*, выходы – только *цифровыми*.

Под *аналоговым сигналом* понимают сигнал, у которого каждый параметр описывается непрерывным во времени множеством значений. Для *цифрового* сигнала имеется два возможных значения (логические нуль и единица), которые являются функцией дискретного времени (т.е. захват значения производится через определённый интервал времени, на протяжении которого это значение остаётся неизменным). На рисунке 2 приведены графики описанных сигналов.

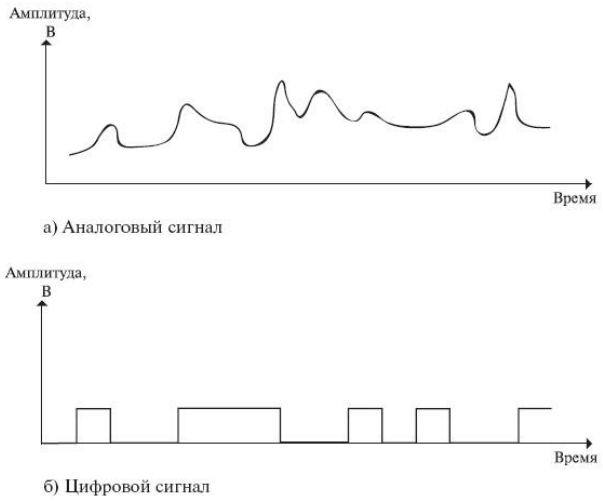

Рисунок 2. Пример аналогового и цифрового сигналов.

По умолчанию все порты настроены на вход, а ножки, поддерживающие функции AN0 – AN12, – на приём аналогового сигнала. Это позволяет экономить потребляемый контроллером ток. Чтобы перестроить эти ножки в цифровой режим, используется регистр ADCON1 (ADC Control Register 1 – регистр управления аналого-цифровым преобразователем, АЦП).

В цифровом режиме порта работают ещё два вида регистров: приёмные – PORT*x*, и передающие – LAT*x*.

Другие регистры SFR мы рассмотрим по мере необходимости в ходе работы. Сейчас же поговорим о семисегментном индикаторе. Его внешний вид и принципиальная схема представлены на рисунке 3.

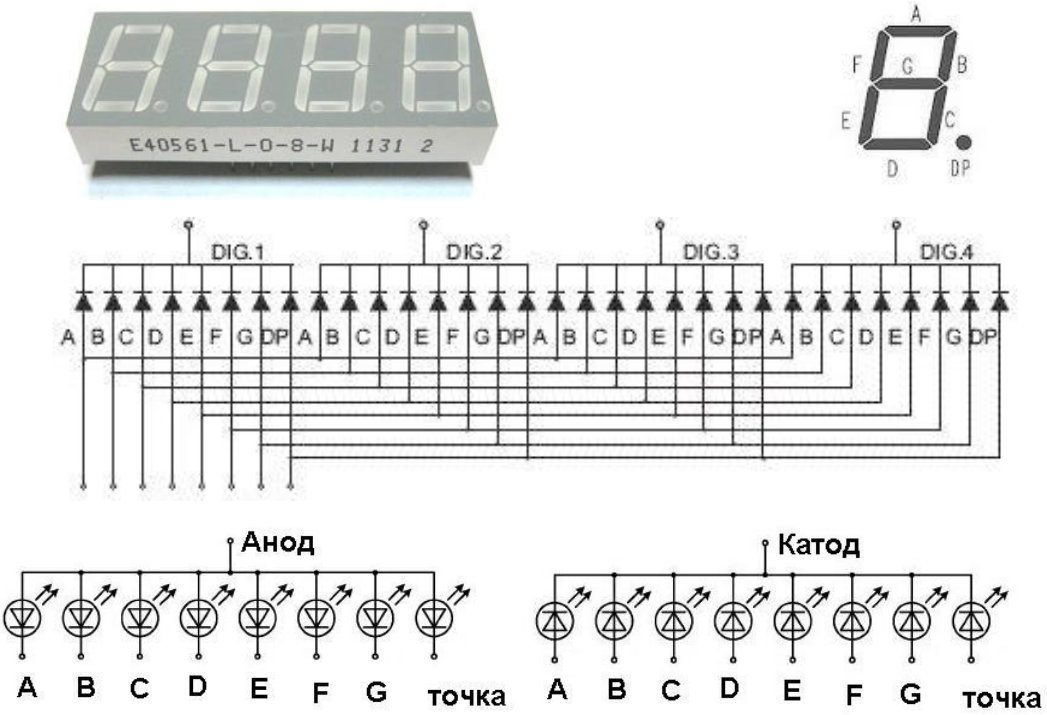

Рисунок 4. Внешний вид семисегментного индикатора и названия сегментов (сверху), принципиальная схема индикатора с общим катодом (в центре), типы светодиодных цепей (снизу): с общим анодом (слева) и с общим катодом.

В нашей работе используется индикатор с общим катодом. Как видно из рисунка, сегменты всех знаков соединены параллельно, а чтобы зажечь требуемый светодиод необходимо через токоограничивающий резистор подать напряжение на соответствующий сегмент и замкнуть на общий провод одну из линий DIG*x* ($x = 1, 2, 3, 4$). Максимальный ток ножки МК составляет 25 мА, а ток светодиода – до 10 мА (в зависимости от требуемой яркости). Поскольку общий ток знака может доходить до 80 мА, замыкание цепи следует произвести через электронный ключ – *транзистор*.

Переключение знаков осуществляется с помощью *динамической индикации*, т.е. последовательной подачи напряжения на сегменты и замыкания соответствующего ключа. Чтобы это переключение не было заметно, нужно отпирать транзистор с частотой, превышающей 24 Гц (24 кадра в секунду). При этом будет казаться, что светодиоды теряют часть яркости. Сегментами же можно управлять ножками контроллера.

Для управления четырёхзначным индикатором требуется 12 сигнальных линий, поэтому, чтобы не использовать такого количества ножек МК, мы использовали *сдвиговый регистр* 74HC595D (рис. 4). Он представляет собой микросхему, которая способна хранить 1 байт информации. Значение каждого бита выведено на отдельную ножку (параллельный выход). Данные же заносятся последовательно по линии SER (Serial Data Input) по каждому фронту сигнала тактовой частоты (линия SRCLK, Shift Register Clock). По этому же фронту предыдущее значение сдвигается вправо на один бит, отсюда и название устройства. Чтобы в регистре запомнилась введённая комбинация, необходимо подать напряжение высокого уровня (логическую единицу) на линию RCLK (Register Clock). Микросхема также имеет линию сброса (обнуления значения) – nSRCLR (Shift Register Clearing), который активируется низким уровнем напряжения (логическим нулём). Поэтому в процессе работы регистра этот вывод должен быть притянут к питанию. Подача логического нуля на вывод nOE (Operation Enable) включает регистр. Выводы информации: QA – QH. Вывод QH’ позволяет подключать несколько регистров каскадом. Заметим, что горизонтальная черта в названиях сигналов эквивалентна букве «*n*» перед ними. Эти символы означают, что активным уровнем является нуль. Если их нет, – то единица.

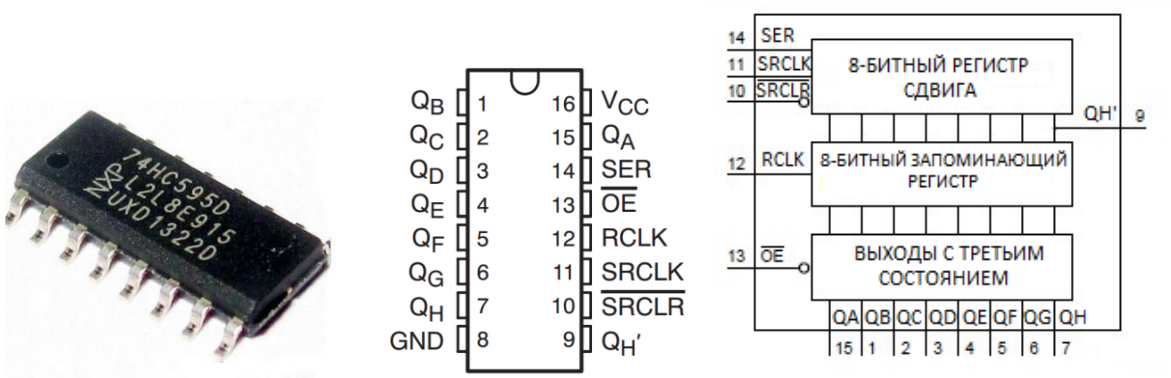

Рисунок 4. Сдвиговый регистр 74HC595D (слева), его карта выводов (в центре) и блок-схема (справа).

Очевидно, что для управления сдвиговым регистром требуется всего четыре вывода МК.

2. Демонстрационная плата. Работа с индикатором.

В этом разделе мы поговорим о демонстрационной плате на базе PIC18F2550. Здесь же приведём листинги кодов для работы со сдвиговым регистром и сегментным индикатором.

Демо-плата представляет собой печатную плату размером 57 х 50 мм, на которой расположены микроконтроллер, сдвиговый регистр, микросхема интерфейса RS-232, индикатор, клеммные колодки и разъём MicroUSB. Принципиальная схема представлена на рис. 5. На рис. 6 приведена 3D-модель.

Составим коды для индикатора, соответствующие цифрам и буквам A – F. Согласно принципиальной схеме, в сдвиговый регистр значения сегментов следует заносить так: E, D, DP, C, G, F, A, B. В таблице 1 приведены требуемые нам символы, их рисунки на индикаторе, бинарный и шестнадцатеричный коды для регистра. В листинге 1 приведен код управления индикатором.

Таблица 1. Таблица кодов для сдвигового регистра.

Символ	Рисунок	BIN	HEX	Символ	Рисунок	BIN	HEX
0		1101 0111	D7	8		1101 1111	DF
1		0001 0001	11	9		0101 1111	5F
2		1100 1011	CB	A		1001 1111	9F
3		0101 1011	5B	B		1101 1100	DC
4		0001 1101	1D	C		1100 0110	C6
5		0101 1110	5E	D		1101 1001	D9
6		1101 1110	DE	E		1100 1110	CE
7		0001 0011	13	F		1000 1110	8E

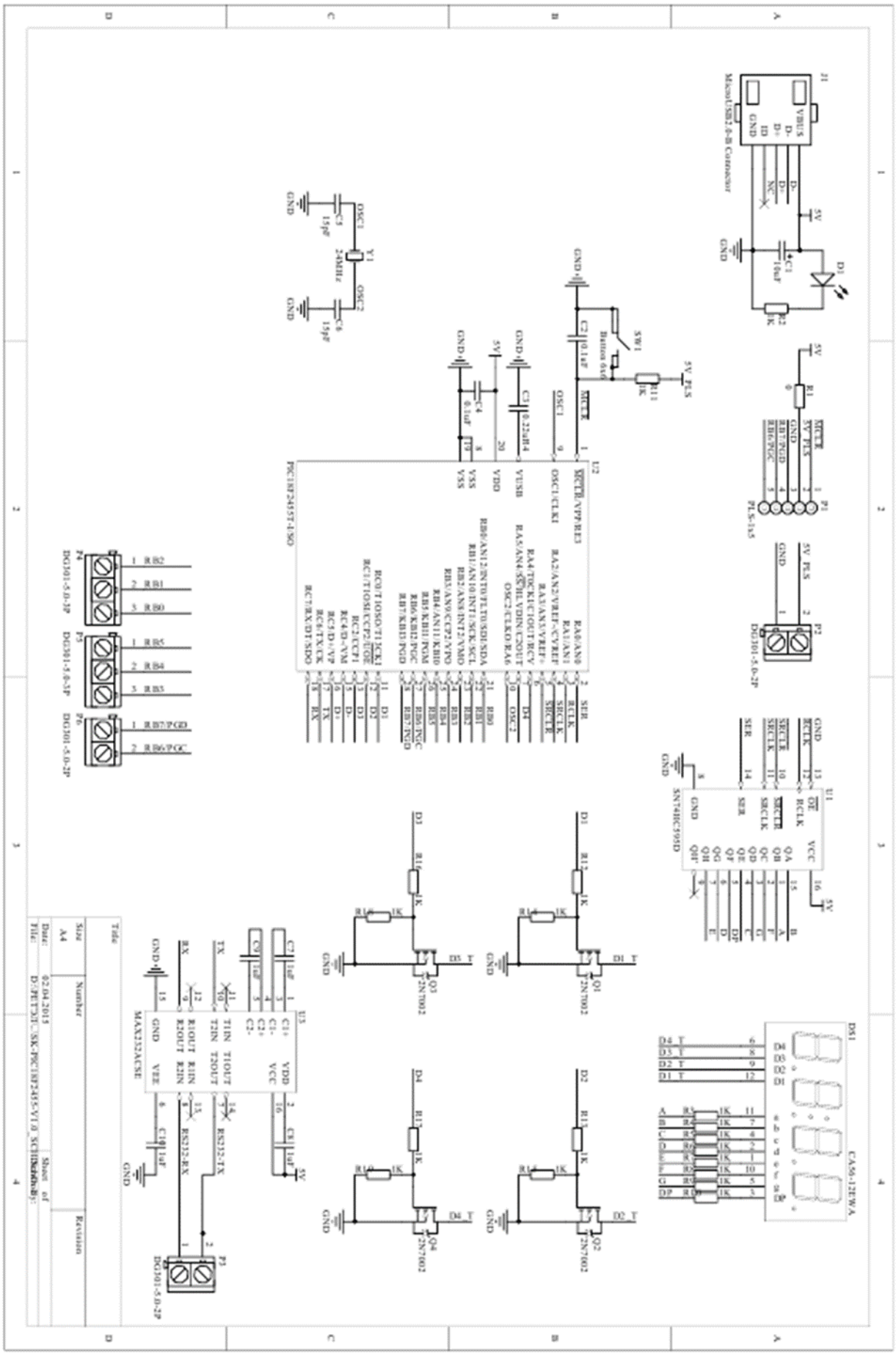

Рисунок 5. Принципиальная электрическая схема демо-платы.

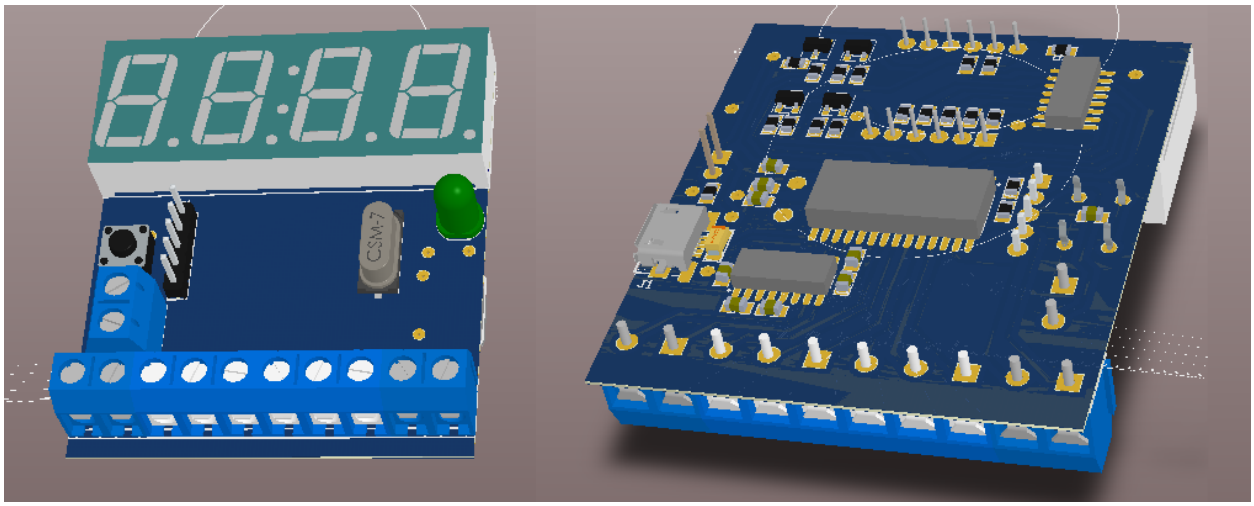

Рисунок 6. Трёхмерная модель демонстрационной платы.

Листинг 1. Функции для работы индикатором: вывод слова и бегущая строка.

```
#define DIG1    LATCbits.LATC0
#define DIG2    LATCbits.LATC1
#define DIG3    LATCbits.LATC2
#define DIG4    LATAbits.LATA5
#define SER     LATAbits.LATA0
#define CLK     LATAbits.LATA2
#define RCLK    LATAbits.LATA1
#define nSRCLR  LATAbits.LATA3

unsigned char Data[]={0xD7, 0x11, 0xCB, 0x5B, 0x1D, 0x5E, 0xDE, 0x13, 0xDF,
0x5F, 0x9F, 0xdc, 0xc6, 0xd9, 0xce, 0x8e};

void shift_reg_write(unsigned char data){
    unsigned char temp = data;
    CLK = 0;
    RCLK = 0;
    for(unsigned char i = 0; i < 8; i++){
        if(temp & 0x80) SER = 1;
        else SER = 0;
        asm("nop");
        CLK = 1;
        asm("nop");
        CLK = 0;
        temp <<= 1;
    }
    SER = 0;
    RCLK = 1;
}

void pause(unsigned long time){
    while(time-- > 0);
```

```
}

void Indicator_SetWord(unsigned char byte1, unsigned char byte0){
    time = 1000;
    while(time-- > 0){
        DIG1 = DIG2 = DIG3 = DIG4 = 0;
        shift_reg_write(Data[byte0 & 0x0F]);
        DIG4 = 1;
        pause(100);

        DIG1 = DIG2 = DIG3 = DIG4 = 0;
        shift_reg_write(Data[(byte0 >> 4) & 0x0F]);
        DIG3 = 1;
        pause(100);

        DIG1 = DIG2 = DIG3 = DIG4 = 0;
        shift_reg_write(Data[byte1 & 0x0F]);
        DIG2 = 1;
        pause(100);

        DIG1 = DIG2 = DIG3 = DIG4 = 0;
        shift_reg_write(Data[(byte1 >> 4) & 0x0F]);
        DIG1 = 1;
        pause(100);
    }
}

void Indicator_ByteString(unsigned char *buf, unsigned char length){
    for(unsigned char j = 0; j < length; j++){
        time = 1000;
        while(time-- > 0){
            if(j != 0){
                DIG1 = DIG2 = DIG3 = DIG4 = 0;
                shift_reg_write(Data[(buf[j-1] >> 4) & 0x0F]);
                DIG1 = 1;
                pause(100);

                DIG1 = DIG2 = DIG3 = DIG4 = 0;
                shift_reg_write(Data[buf[j-1] & 0x0F]);
                DIG2 = 1;
                pause(100);
            }
            DIG1 = DIG2 = DIG3 = DIG4 = 0;
            shift_reg_write(Data[(buf[j] >> 4) & 0x0F]);
            DIG3 = 1;
            pause(100);

            DIG1 = DIG2 = DIG3 = DIG4 = 0;
            shift_reg_write(Data[buf[j] & 0x0F]);
            DIG4 = 1;
            pause(100);
        }
        DIG1 = DIG2 = DIG3 = DIG4 = 0;
    }
}
```

Программирование МК производилось в среде MPLABX®, компилятор XC8. Для отладки кода использовалась программа-симулятор Proteus 8. О ней мы поговорим в разделе, где будем описывать отладку готового устройства.

3. Протокол 1-Wire.

В предыдущей главе для общения со сдвиговым регистром нам требовались три сигнальных линии: CLK, DATA и RCLK. В интерфейсе 1-Wire линии тактовой частоты и данных совмещены в одну, которая в технической документации обычно называется DQ, а сигнал ввода значения RCKL отсутствует. Его роль выполняет *протокол обмена данными*.

Как совместить сигнал тактирования с сигналом данных? Разработчики интерфейса предложили следующее решение. Состояние свободной линии – это высокий уровень (3,3 или 5 В), спад сигнала до низкого уровня (0 В) расценивается всеми устройствами как сигнал тактирования, затем передающее устройство должно выдержать линию в низком состоянии в течение определённого времени, а затем отпустить. Причём время удержания для передачи нуля больше (не менее 15 мкс), чем время удержания для передачи единицы (не менее 1 мкс). Принимающее устройство в интервале от 2-ой до 15-ой мкс, в зависимости от микросхемы, должно считать значение. Время же передачи одного бита информации должно находиться в пределах от 60 до 120 мкс. Такой фрагмент информации во времени называют *тайм-слотом*.

Как передаются данные? Сеть 1-Wire – это сеть, в которой есть только одно ведущее устройство (*Master*) и много ведомых (*Slave*). Все транзакции – акты приёма-передачи информации – инициирует Мастер. Сперва, чтобы узнать, есть ли устройства на линии, он выдаёт сигнал RESET (сброс) – удерживает линию в состоянии 0 В на протяжении не менее 480 мкс, но не более 960 мкс, а затем отпускает. Через 15 – 60 мкс все ведомые устройства должны выдать ответный импульс PRESENCE, длительностью от 60 до 240 мкс. После отпускания линии всеми устройствами, Мастер должен выдержать паузу такую, чтобы в сумме с PRESENCE она давала менее 480 мкс. После такой процедуры инициализации ведущее устройство выдаёт команду и передаёт или получает данные от ведомого, не забывая при этом выдавать тактовые импульсы и учитывая описанные выше временные интервалы. Иллюстрации приведены на рисунке 7.

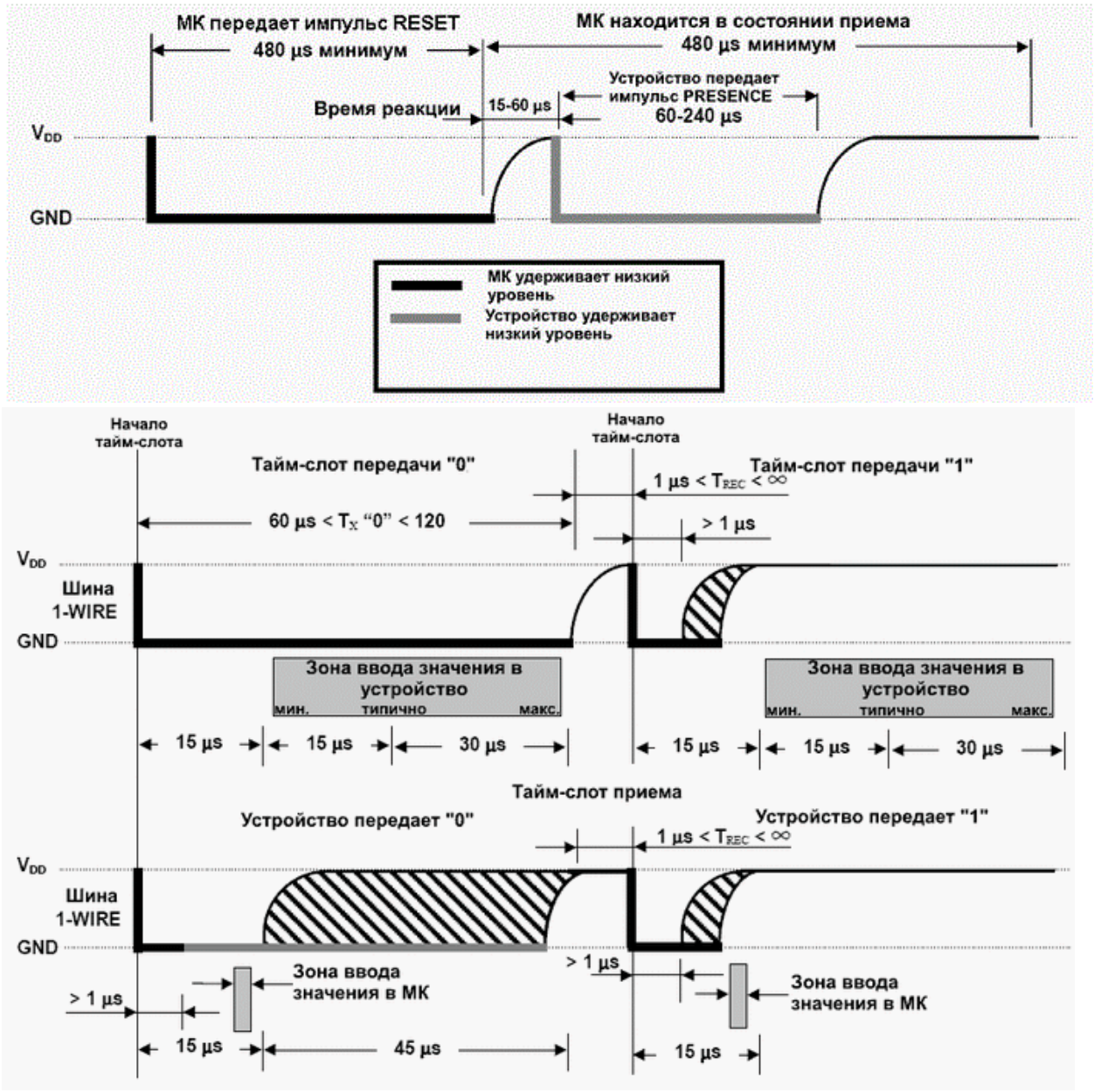

Рисунок 7. Иллюстрации к описанию физического уровня интерфейса 1-Wire.

Команды 1-Wire. Поговорим теперь о логическом уровне – протоколе. Любая транзакция начинается с команды. Спецификацией 1-Wire предусмотрены общие команды и команды, специфичные для конкретного устройства. К общим командам относятся:

- READ ROM (0x33): считать адрес единственного устройства на линии; после этой команды устройство отправляет свой адрес;
- MATCH ROM (0x55): выбор адреса – используется для обращения к конкретному устройству из многих подключенных; следом за командой Мастер отправляет адрес устройства;

- SKIP ROM (0xCC): игнорировать адрес – используется для обращения к единственному устройству на шине, при этом адрес устройства игнорируется (можно обращаться к неизвестному устройству или ко всем сразу);
- SEARCH ROM (0xF0): поиск адресов; после этой команды активируется алгоритм поиска устройств на линии.

Адрес устройства представляет собой 64-битный код, разбитый на три поля (рис. 8): код семейства (Family Code, 1 байт), уникальный идентификатор (Serial Number, 6 байт) и циклический код (CRC, 1 байт).

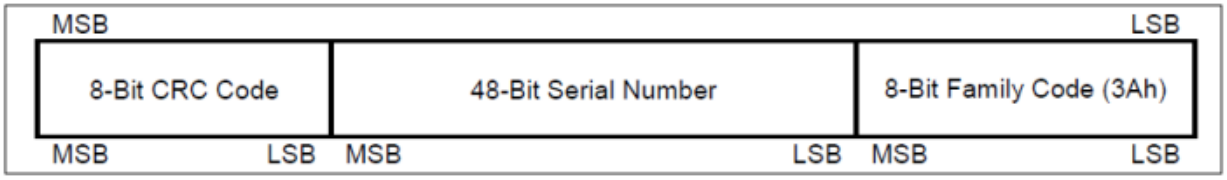

Рисунок 8. Формат адреса устройств 1-Wire.

После обработки адреса Мастер выдаёт на линию специфическую команду. Работа посвящена созданию терморегулятора на базе датчика температуры DS18B20, поэтому рассмотрим только его команды.

- CONVERT T (0x44): запуск конверсии температуры; следом датчик выдаёт бит состояния: 1 – конверсия завершена, 0 – в процессе.
- WRITE SCRATCHPAD (0x4E): записать конфигурацию; Мастер отправляет 3 байта настроек;
- READ SCRATCHPAD (0xBE): считать данные; устройство отправляет 9 байт, среди которых находятся настройки и значение температуры;
- COPY SCTRATCHPAD (0x48): копировать записи в энергонезависимую память;
- RECALL E2 (0xB8): вызвать данные из энергонезависимой памяти; устройство отправляет Мастеру 3 байта скопированных данных;
- READ POWER SUPPLY (0xB4): считать статус питания; устройство отправляет бит состояния: 1 – независимое питание, 0 – паразитное (от линии данных).

4. Обработка информации с термодатчика.

Этот раздел, в основном, состоит из листингов функций, предназначенных для работы с DS18B20. Также здесь коротко описана среда отладки Proteus 8 и приведены скриншоты.

Традиционная схемотехника 1-Wire основана на применении схем с открытым стоком. Все устройства контролируют линию с помощью порта ввода-вывода, изображённого на рис. 9. Используя МК, можно сделать порт как на двух ножках по такой же схеме, так и на одной. Первый вариант универсальный, здесь порт МК настраивается один раз: вывод OWTXPIN – на выход, OWRXPIN – на вход. Второй же прекрасно подходит для работы с одним дат-

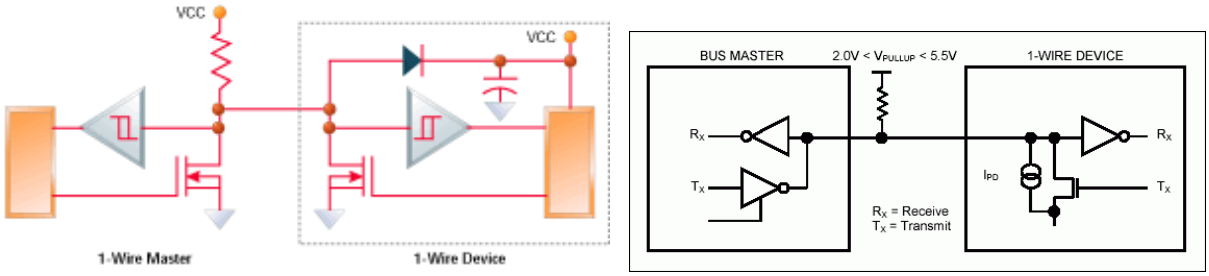

Рисунок 9. Схемотехника 1-Wire: универсальный порт (слева) и однопиновый вариант (справа).

чиком, но при этом ножка будет постоянно перестраиваться между работой на вход и на выход. Кроме того, поскольку транзистор с открытым стоком по сути является инвертором, уровни на линии в обоих случаях будут задаваться противоположными логическими значениями ножек. Мы реализовали программы для обоих вариантов.

Поясним подробнее работу с одной ножкой. В программе мы ввели определение SINGLE_PIN_VERSION. В этом режиме мы задаём бит направления порта TRISBbits.TRISB1 двумя определениями OWTXPIN_IO и OWRXPIN_IO, а чтобы не возникало коллизий с двухпиновой версией, по мере необходимости на выход настраиваем OWTXPIN_IO, а на вход – OWRXPIN_IO. При отправке данных ножка настраивается только на выход, а контроллер действует в соответствии с таймингом. При приёме бита ножка настраивается на выход, микроконтроллер удерживает линию в нуле в течение 1 мкс, затем отпускает её и перестраивает ножку на вход. До 15-ой мкс принимает значение с линии, а затем перестаёт её контролировать до начала следующего бита.

Выбор режима работы осуществляется с помощью директивы предкомпилятора #ifndef – #elif – #endif. Листинги приведены ниже.

Листинг 2. Определения и функции для работы с линией 1-Wire.

```
#define SINGLE_PIN_VERSION

#ifndef (SINGLE_PIN_VERSION)
#define OWTXPIN         LATBbits.LATB0
#define OWTXPIN_IO      TRISBbits.TRISB0
#define TX_HIGH         0
#define TX_LOW          1
#elif
#define OWTXPIN         LATBbits.LATB1
#define OWTXPIN_IO      TRISBbits.TRISB1
#define TX_HIGH         1
#define TX_LOW          0
#endif

#define OWRXPIN         PORTBbits.RB1
#define OWRXPIN_IO      TRISBbits.TRISB1

#define OUTPUT          0
#define INPUT           1

#define NO_DEVICES      0x00
#define READY           0x01
#define SHORT_CIRCUIT   0x02

#define PROCESS         0x00
#define DONE            0x01

#define MATCH_ROM       0x55
#define READ_ROM        0x33
#define SKIP_ROM        0xCC
#define SEARCH_ROM      0xF0

#define CONVERT_T       0x44
#define READ_SCRATCHPAD 0xBE

#define _480us          240
#define _116us          58
#define _10us           5
#define _4us            2
#define _2us            1

unsigned char OW_Reset(void){
     unsigned char reset_to_presence_timeout = 30;
     OWTXPIN_IO = OUTPUT;
     OWTXPIN = TX_LOW;
     pause(_480us);
     OWTXPIN = TX_HIGH;
     OWRXPIN_IO = INPUT;
     while(reset_to_presence_timeout-- > 0){
          if(OWRXPIN == 0){
               pause(_480us);
               if(OWRXPIN == 1){
                    return READY;
               }
```

```
                return SHORT_CIRCUIT;
            }
        }
        return NO_DEVICES;
}

void OW_Write(unsigned char command){
    unsigned char byte = command, i;
    OWTXPIN_IO = OUTPUT;
    for(i = 0; i < 8; i++){
        if(byte & 0x01){
            OWTXPIN = TX_LOW;
            pause(_4us);
            OWTXPIN = TX_HIGH;
            pause(_116us);
        }else{
            OWTXPIN = TX_LOW;
            pause(_116us);
            OWTXPIN = TX_HIGH;
            pause(_4us);
        }
        byte >>= 1;
    }
}

unsigned char OW_Read_Byte(void){
    unsigned char byte = 0, i;
    for(i = 0; i < 8; i++){
        OWTXPIN_IO = OUTPUT;
        OWTXPIN = TX_LOW;
        pause(_2us);
        OWTXPIN = TX_HIGH;
        pause(_10us);
        byte >>= 1;
        OWRXPIN_IO = INPUT;
        if(OWRXPIN){
            byte |= 0x80;
        }
        pause(_116us);
    }
    OWTXPIN_IO = OUTPUT;
    return byte;
}

unsigned char OW_Read_Status(void){
    OWTXPIN_IO = OUTPUT;
    OWTXPIN = TX_LOW;
    pause(_2us);
    OWTXPIN = TX_HIGH;
    pause(_10us);
    OWRXPIN_IO = INPUT;
    return OWRXPIN;
}
```

Листинг 3. Программа для работы с DS18B20.

```
// onewire.h ***********************************************************************
typedef union{
    unsigned char All;
    struct{
        unsigned Line       :2;
        unsigned Device     :1;
        unsigned Counter    :5;
    };
}ow_status_t;

typedef union{
    unsigned char All[8];
    struct{
        unsigned char Family;
        unsigned char Number[6];
        unsigned char CRC;
    };
}ow_serial_t;

typedef union{
    unsigned char All[9];
    struct{
        unsigned char Temperature_LSB;
        unsigned char Temperature_MSB;
        unsigned char User_Byte_1;
        unsigned char User_Byte_2;
        unsigned char Configuration;
        unsigned char Reserved[3];
        unsigned char CRC;
    };
}ow_scratc_t;

typedef struct{
    ow_status_t Status;
    ow_serial_t Serial;
    ow_scratc_t Scratchpad;
}one_wire_t;

// main.c **************************************************************************
one_wire_t u1;
int main(){
   while(1){
       // Импульс Reset
       u1.Status.All = 0;
       u1.Status.Line = OW_Reset();

       if(u1.Status.Line == READY){
           OW_Write(READ_ROM);
           for(u1.Status.Counter=0; u1.Status.Counter<8; u1.Status.Counter++){
               u1.Serial.All[u1.Status.Counter] = OW_Read_Byte();
           }
       }

       // Вывод идентификатора на индикатор
```

```
        Indicator_ByteString(u1.Serial.All, 8);


        // Начинает измерение температуры
        u1.Status.Line = OW_Reset();

        if(u1.Status.Line == READY){
            OW_Write(MATCH_ROM);
            for(u1.Status.Counter=0; u1.Status.Counter<8; u1.Status.Counter++){
                OW_Write(u1.Serial.All[u1.Status.Counter]);
            }

            while(u1.Status.Device == PROCESS){
                OW_Write(CONVERT_T);
                u1.Status.Device = OW_Read_Status();
            }
        }

        if(u1.Status.Device == DONE){
            u1.Status.Device = PROCESS;
            u1.Status.Line = OW_Reset();
            if(u1.Status.Line == READY){
                OW_Write(SKIP_ROM);
                OW_Write(READ_SCRATCHPAD);
            for(u1.Status.Counter=0; u1.Status.Counter<9; u1.Status.Counter++){
                    u1.Scratchpad.All[u1.Status.Counter] = OW_Read_Byte();
                }
            }
        }
        // Вывод скрэтчпада
        Indicator_ByteString(u1.Scratchpad.All, 9);
    }
}
```

Программа отлажена в среде-симуляторе Proteus 8. Пакет представляет собой систему схемотехнического моделирования на основе моделей электронных компонентов, принятых в PSpice. Отличительной чертой пакета PROTEUS VSM является возможность моделирования работы программируемых устройств: микроконтроллеров, микропроцессоров, DSP и проч. Библиотека компонентов содержит справочные данные. Дополнительно в пакет PROTEUS VSM входит система проектирования печатных плат. Пакет Proteus состоит из двух частей, двух подпрограмм: ISIS — программа синтеза и моделирования непосредственно электронных схем и ARES — программа разработки печатных плат. Вместе с программой устанавливается набор демонстрационных проектов для ознакомления. [1]

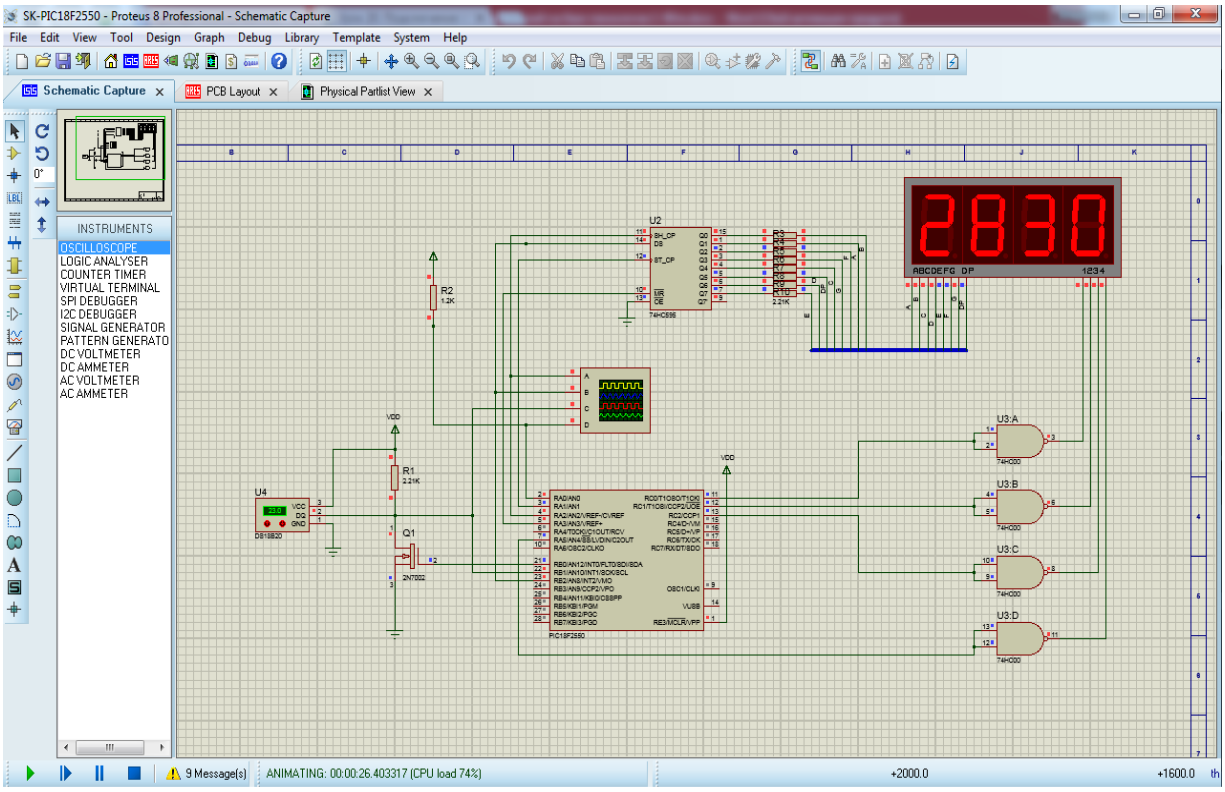

Рисунок 10. Скриншот окна Proteus 8 во время симуляции.

ВЫВОДЫ И ПРАКТИЧЕСКИЕ РЕКОМЕНДАЦИИ

На данном этапе проектирования терморегулятора изготовлен прототип устройства «в железе» на базе отладочной платы, а также написаны и отлажены программные коды, необходимые для работы с цифровым термодатчиком.

Рекомендации для дальнейшего развития проекта таковы:

- произвести пересчёт температуры из выходных данных термодатчика в градусы Цельсия;
- разработать алгоритм управления сигнальным реле, включающим и выключающим электрический обогреватель, с помощью которого регулируется температура помещения;
- разработать принципиальную схему готового устройства, развести и спаять плату;
- провести испытания устройства в жилой комнате и дать рекомендации по его эксплуатации.

ЗАКЛЮЧЕНИЕ

В процессе работы над проектом я сделал следующее:

- спаял демо-плату;
- в необходимом объёме освоил архитектуру микроконтроллера PIC18F2550 и язык C для его программирования;
- ознакомился с электронным термометром DS18B20, сдвиговым регистром и индикатором;
- разобрался с интерфейсом 1-Wire и написал для него необходимые функции;
- отладил программу в симуляторе Proteus 8.0;
- ознакомился с некоторыми действующими нормами и правилами, касающимися климат-контроля жилых помещений.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. PIC18F2455/2550/4455/4550 Data Sheet. 28/40/44-Pin, High-Performance, Enhanced Flash, USB Microcontrollers with nanoWatt Technology. // Электронный ресурс.
2. Ю. С. Магда «Микроконтроллеры PIC: архитектура и программирование». М.: ДМК Пресс, 2009. – 240 стр.: ил.
3. Википедия «Умный дом». www.ru.wikipedia.org/wiki/Умный_дом // Электронный ресурс.
4. DS18B20 Programmable Resolution 1-Wire Digital Thermometer. // Электронный ресурс.
5. Радиокот «Интерфейс 1-Wire». http://radiokot.ru/articles/13/ // Электронный ресурс.
6. СхемНет «Интерфейс 1-Wire». http://cxem.net/comp/comp53.php // Электронный ресурс.
7. РобоКрафт «1-Wire». http://robocraft.ru/blog/communication/117.html // Электронный ресурс.
8. Атерлюкс «Программная реализация протокола 1-Wire (iButton, MicroLAN) на микроконтроллерах AVR». www.aterlux.ru/index.php?page=article&art=1wire // Электронный ресурс.
9. Википедия «Proteus (система автоматизированного проектирования)». www.ru.wikipedia.org/wiki/Proteus_(система_автоматизированного_проектирования) // Электронный ресурс.
10. LabKit «Моделирование работы микроконтроллеров в Proteus или как зашить ПИК в Протеусе». http://www.labkit.ru/html/tutorial?id=383 // Электронный ресурс.

ИСПОЛЬЗОВАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

1. MPLABX® IDE – среда программирования микроконтроллеров PIC;
2. XC8 – компиллятор кода для 8-разрядных микроконтроллеров;
3. Altium Designer 14 – среда автоматического проектирования печатных плат;
4. ISIS Proteus 8.0 – среда автоматического проектирования и моделирования электронных устройств;
5. Microsoft Word 2013 – текстовый редактор.