МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И.Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)

Учебное пособие по дисциплине

«Компьютерная и микропроцессорная техника»

Практикум по МК AVR-8 на симуляторах

Бондаренко Д.Н.

Санкт-Петербург 2016

УДК 621.316.544.1 (035.5)

ББК 32.844.1_04я2

Автор: Бондаренко Д. Н.

Практикум по МК AVR8 на симуляторах: Электронное учеб. пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2016. – 56 с.

ISBN 978_5_94120_220_1

Учебное пособие предназначено для проведения лабораторного практикума в процессе начального изучения встраиваемых микроконтроллеров. Рекомендуется использовать при выполнении лабораторных работ по дисциплине «Компьютерная и микропроцессора техника» плана подготовки бакалавров по направлению 13.03.02 «Электроэнергетика и электротехника» и профилю 13.03.02-11 «Электротехнологические установки и системы». В комплекте с.предыдущим учебным пособием [1] помогает в самостоятельном изучении встраиваемых 8 разрядных микроконтроллеров семейства AVR (Atmel).

Данное учебное пособие содержит готовые примеры алгоритмов, схем, программ и наборы индивидуальных заданий по четырём темам:

- порты ввода/вывода и принцип программного управления,
- таймеры/счётчики и прерывания в задачах генерации импульсов,
- АЦП и контактные датчики температуры в терморегуляторе,
- последовательный интерфейс для подключения внешнего ЦАП к терморегулятору.

Практикум предусматривает получение навыков по тестированию готовых примеров и отладке в процессе разработки индивидуальных заданий.

Особенностью данного курса является ориентация примеров на программные симуляторы, это позволяет при знакомстве с примерами и разработке индивидуальных заданий пользоваться только персональным компьютером. Используются свободно распространяемые программные среды проектирования.

IDE AVR Studio (Atmel) с компилятором AVR-GCC (пакет WinAVR, GNU-C) используются для редактирования текста программ, компиляции с языка С в машинные коды и отладки в цифровом симуляторе. Схемный редактор ISIS с системой моделирования Proteus VSM (Labcenter Electronics) используется для аналого-цифрового моделирования электрических схем, включающих программируемую модель МК, модели пассивных компонентов, микросхем, интерактивных компонентов и пр.

Примеры в тексте учебного пособия имеют ссылки на папки и файлы с готовыми проектами для AVR-Studio и Proteus VSM, они размещены на диске.

Содержание

Введение

Данное пособие содержит лишь краткое описание изучаемых ресурсов МК и алгоритмов. Более полное описание дается в курсе лекций по дисциплине «Компьютерная и микропроцессора техника» и в учебных пособиях [1, 2] и в справочных изданиях [3].

При проведении лабораторных работ по дисциплине «Компьютерная и микропроцессора техника» аудиторное время в компьютерном классе ограничено (34 ч), причём данный практикум охватывает только 4 из 6 работ. Поэтому для успешного выполнения программы обучения следует грамотно использовать время для самостоятельной подготовки.

Каждая из 4 работ состоит из следующих разделов:

- *название и цели работы* со ссылками на разделы учебного пособия [1],
- *основные понятия* содержат краткое описание изучаемого материала и ссылки на разделы учебного пособия [1], где размещено подробное описание,
- *готовый пример*: задание, алгоритм, схема, текст программы, методика тестирования,
- *индивидуальные задания* для закрепления понимания материала и примера с рекомендациями по разработке и требованиями по оформлению отчёта.

Работа в компьютерном классе (аудитории) состоит из:

- тестирования готовых примеров,
- консультаций по разработке индивидуальных заданий,
- демонстрации тестирования индивидуальных заданий преподавателю,
- защиты работы в виде ответов на контрольные вопросы (из текста [1] или сформулированных преподавателем по данной работе).

Читать и разбирать материал работ надо вне аудитории, это же относится и к рутинным операциям по разработке и оформлению индивидуальных заданий.

Рассмотрим подробнее методику выполнения работ.

1. Знакомство с целями работ, основными положениями и готовым примером рекомендуется выполнять самостоятельно (вне аудитории). Затем рекомендуется познакомиться с текстом УП из разделов, указанных в «Целях работ».
2. Тестирование готового примера рекомендуется выполнять аудиторно на подготовленных компьютерах в присутствии преподавателя, при необходимости консультируясь. Важно понять принципы реализации алгоритмов и освоить рекомендуемые приёмы тестирования. Тестирование преследует разные цели: проверка общего хода выполнения алгоритма, проверка численных преобразований, измерение длительностей выполнения частей программы, в первую очередь, основного цикла. При работе транслятора формируются данные по расходования памяти программ и данных. Временные характеристики и расход памяти важны при сравнении эффективности алгоритмов.
3. Разработку индивидуального задания рекомендуется выполнять по большей части самостоятельно (вне аудитории). С первого шага важно научиться документировать свои действия по разработке алгоритма, схемы и программы, составить таблицу входных и расчётных данных.

4. Отладка и тестирование разработанных программ (и схем) также можно выполнять самостоятельно, для этого необходимо установить на свой ПК или ноутбук дистрибутивы используемых программных средств (см. выше). Процесс тестирования рекомендуется начинать с проверки работы алгоритма в целом, далее тестируется вариация входных данных по ранее подготовленным данным из таблицы, запись измеренных значений и значений по оценке погрешности, тестирование логических функций, измерения по оценке времени реакции.

5. Демонстрация тестирования индивидуального задания преподавателю выполняется аудиторно. Важно иметь набор данных для тестирования, уметь пользоваться рекомендованными приёмами тестирования. Тестирование предполагает понимание схемы и алгоритма работы, умение продемонстрировать общую работу алгоритма и важные детали, проверку численных данных, оценку получаемого отклонения, измерения времени выполнения важных частей программы.

6. Подготовка отчета выполняется самостоятельно (см. «Требования к отчету»), преподаватель обычно проверяет отчет вне аудитории.

7. Подготовка к ответам на контрольные вопросы предполагает самостоятельный просмотр разделов УП, указанных в заголовке работы.

Содержание отчета

1. Индивидуальное задание в виде строки таблицы с номером варианта и в виде текста.
2. Алгоритм работы:
   - текстовое описание решения по выбору узлов МК и режима их работы, частоты тактирования fclk,, выбор МК,
   - схема модели в Proteus VSM (если использована);
   - программа на С – блок-схема или текстовое описание структуры программы и самых важных функций.
3. Текст разработанной программы с комментариями.
4. Описание процедуры тестирования (AVR-Studio или/и Proteus VSM) и результатов тестирования алгоритма (оценка точности и дискретности, закона регулирования/управления).
5. Время выполнения основных элементов программы и использованные ресурсы МК (количество ячеек памяти программ и данных). Время реакции алгоритма – минимальное и максимальное значения.

Лабораторная работа 1 [1, p.1, 2]
Принцип программного управления и порты ввода/вывода

Цели работы:

1. Изучение архитектуры ядра МК серии AVR [1, p.1.2].
2. Приобретение навыков в использовании портов ввода/вывода [1, p.1.6].
3. Знакомство с принципом и свойствами программного управления [1, p.1.6].
4. Знакомство со средствами программирования и отладки [1, p.2.3, 2.4].

1. Основные понятия

1.1. Архитектура МК

Это описание элементов, необходимых для работы пользователя, то есть программиста и схемотехника:

- система команд и способы адресации (ассемблер, оптимальный код на С);
- организация памяти и регистров;
- организация встроенных периферийных устройств и системы прерываний;
- типовые схемы включения с описанием назначения и расположения выводов и особенностей внутренней схемотехники выводов.

1.2. Порты ввода/вывода

Это основной элемент системы ввода/вывода дискретных сигналов. Выводы портов имеют наименование *Pxy*, где *x* – имя порта (*A, B,* …), *y* – номер вывода порта (0, 1, … , 7). При старте МК все выводы портов настроены на ввод, уровень сигнала определяется внешней схемой и может быть программно считан как Высокий (логическая единица, 1) или Низкий (логический ноль, 0) уровень. Программа может настроить любой вывод как выход и задать Высокий или Низкий уровень напряжения.

Программа работает с портами через три регистра, каждый из восьми битов связан с соответствующим по номеру выводом, то есть PA0 – DDRA.0 – PIRTA.0 – PINA.0.:

DDRx [Data Direction Register x] – регистр направления порта *x*, доступен по записи и чтению, обеспечивает индивидуальную настройку восьми выводов в состояние входа (0) или выхода (1);

PORTx – регистр данных порта *x,* доступен по записи и чтению, определяет уровень выводимого сигнала при операциях вывода, при операциях ввода определяет тип входа – высокоомное *Hi-Z* состояние (0, после рестарта) или подключение к ножке внутреннего подтягивающего *[pull-up]* резистора (1);

PINx – регистр состояния вывода (ножки), доступен только по чтению, связан с ножкой через триггер Шмита.

После рестарта МК регистры *DDRx* и *PORTx* обнулены, все выводы находятся в режиме высокоомного входа. Запись 1 в *PORTx.y* (при *DDRx.y* = 0) подключает ко входу *Pxy* внутренний резистор (сопротивление 30…70 кОм), подтягивающий вверх, то есть к напряжению питания *VCC* = 5 В.

Для перевода вывода *Px.y* в режим выхода необходимо записать 1 в бит *DDRx.y*. Для перевода выхода в ВЫСОКОЕ состояние необходимо записать 1 в бит *PORTx.y*.

Для изменения состояния одного бита или группы бит регистра ввода/вывода на языке С используются либо операции присвоения новых данных всему регистру (2 машинных

цикла), либо операции «чтение-модификация-запись» (3 м.ц.), изменяющие состояние только определённого бита или бит.

В первом случае 8-битная константа или переменная записывается в регистр:

```
DDRB = (1<<0); // = 0b0000 0001 = 0x01 – установка младшего бита, остальные сброшены.
```

Во втором случае операция присвоения константы объединяется с побитовыми логическими операциями:

```
DDRB |= (1<<0); //ИЛИ с присвоением, установка 0-го бита без изменения остальных
DDRB &= ~(1<<0); //И с присвоением инвертир. константы, обнуление 0-го бита без изменения остальных
```

Чтение бита регистра входа выполняется с использованием операции побитового И:

```
if ( PINB & (1<<0) ) //Условие истинно, если установлен бит 0 регистра PINB
```

Регистры *DDRx, PORTx, PINx* находятся в адресном пространстве регистров ввода/вывода *[Memory I/O]*, также доступны в пространстве памяти данных *[Memory Data]*, но с более длинными адресами.

1.3. Принцип программного управления

Состоит в *циклическом* выполнении программных действий: чтении входов, вычислении управляющих воздействий и их выводе.

После рестарта МК однократно выполняет инициализацию, то есть настройку регистров ввода/вывода и переменных. Затем начинается бесконечное выполнение команд *основного цикла*.

Время реакции выхода на изменение входного сигнала варьируется от одного до двух периодов основного цикла. Длительность основного цикла зависит от числа машинных команд, используемых в этом цикле и производительности МК.

На языке С программа состоит из в функции main() {…}, выход из неё не предусмотрен, основной цикл обычно ограничен оператором while() {…}. Внутри функции main могут вызываться другие функции, в том числе вложенные. Тексты всех функций при трансляции преобразуются в машинные коды, размещаемые в памяти программ. Глобальные переменные при трансляции размещаются в памяти данных.

1.4. Среда программирования и отладки AVR Studio с транслятором AVR GCC

Обеспечивают написание и редактирование текста программы, трансляцию текста программы в машинный код в различных форматах, загрузку машинного кода в целевой МК, поддерживают процесс отладки в целевом МК или в программной модели (симуляторе), сохраняют все настройки в проекте.

В данной работе вы познакомитесь с проектом, редактором текста, отладкой в симуляторе, освоите простейшие команды пошаговой отладки, работу с окнами процессора, устройств ввода/вывода и переменных [1, p.2.3, 2.4].

2. Пример 1: «Регулятор с дискретным выходом»

2.1. *Задание*

Регулятор получает численное значение через порт *A* (диапазон $x = 50…255$), линейно преобразует его для сравнения в диапазон $y = 500…1200$ и переключает выход по результатам сравнения y с уставной и гистерезисом. Уставка сравнения $set = 850$, гистерезис $hys = 10$, выход $out = PD1$ реализует закон 1 «нагреватель»: *out* переходит в «1», если $y < Set - his$ и в «0», если $y > Set + his$.

Табл. 1.1. Задание в краткой форме

№	*X*	*Y*	*Set*	*hys*	*Reg_out*
00	*PA0-7*, 50-255	500…1200	850	10	*PD1*, Зак.1

2.2. Разработка алгоритма и схемы

Построение схемы начинается с выбора МК. Число ног МК для данной задачи: 8 – для ввода числа, 1 – выход регулятора, итого – 9 входов/выходов портов. Еще необходимы выводы для цепи питания (0 В к контакту GND, и +5 В к контакту VCC) и вход сброса Reset. Итого требуется МК с числом выводов не менее 9 + 3 = 12, выбрана одна из первых моделей семейства AVR AT90S8515 с числом выводов 40.

Схема электрическая регулятора представлена на рис. 1.1. Схема разработана в редакторе схем *ISIS* пакета *Proteus VSM* с использованием готовых моделей компонентов и отображает момент сеанса моделирования.

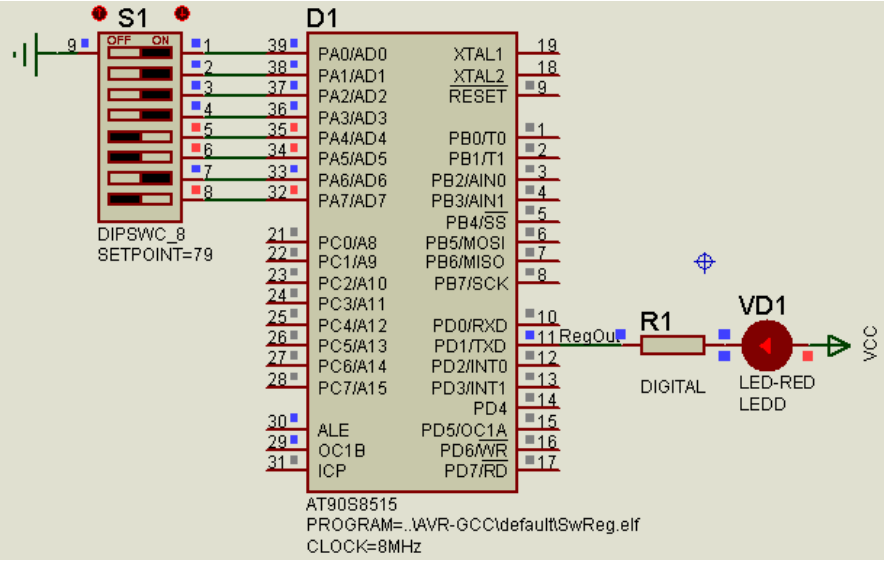


Рис. 1.1

Основной элемент регулятора – микроконтроллер D1 (AT90S8515), его модель в среде ISIS Proteus VSM имеет параметры настройки (частота тактирования <CLOCK=8MHz>, файл машинного кода <PROGRAM=..\SwReg.hex>). На схеме не показаны выводы питания МК: цепь +5 В подключается к выводу с именем VCC (ножка [pin] 40), цепь 0 В – к GND (20).

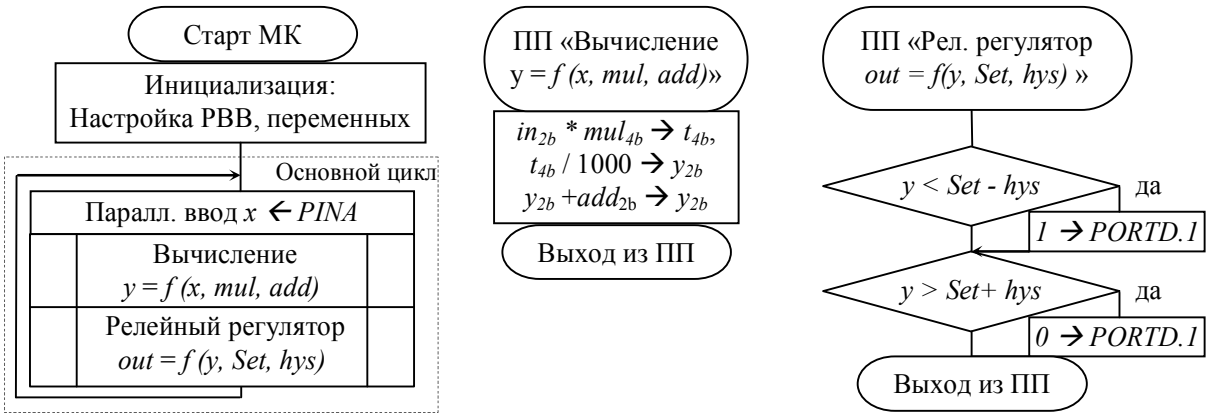
К выводам порта *A* (*PA0…PA7*) подключен движковый переключатель *S1*, он задает значение числа *X*, пропорциональное сигналу обратной связи регулятора, например, температуре. Переключатель состоит из 8 ключей, левые по схеме контакты объединены и подключены к нулю питания. Замыкание ключа гарантирует низкий уровень на соответствующем входе порта *PA0…PA7*. Для формирования высокого уровня в разомкнутом состоянии используются подтягивающие (к +5 В) резисторы порта, программно это задаётся выражением PORTA = 0xff. (Модель переключателя поддерживает интерактивную симуляцию).

Объект регулирования, например, электропечь, имитируется красным светодиодом VD1. Он связан «проводом» *RegOut* с выходом PD1. В данном варианте задания светодиод с токоограничивающим резистором R1 подключены между выходом МК и плюсом питания МК VCC. Для включения «нагрева», то есть свечения светодиода необходимо перевести выход PD1 в НИЗКОЕ состояние и наоборот.

На рис. 1.2 приведены блок-схемы алгоритма работы программы терморегулятора: общая (а), функции вычисления y (б), функции ключевого терморегулятора (в). На этапе

инициализации (рис. 1.2, а) задаются начальные состояния регистров ввода/вывода (порты) и переменных. В основном цикле последовательно выполняются 3 задачи – ввод числа *x* с порта *A*, вычисление значения y по заданному закону преобразования x («измерение»), собственно вычисление состояния выхода регулятора по «измеренному» значению *y* и заданному закону регулирования.

Рассмотрим подробнее каждую из этих задач.



а б в

Рис. 1.2

Ввод численного значения *x* с порта A сводится к копированию содержимого регистра PINA в переменную *x* в формате однобайтного беззнакового целого *[unsigned char*

Преобразование входных значений *x* в «измеренные» значения y выполняется по закону *л*инейной аппроксимации $y(x) = x * k + add$. Для определения значений *k* и *add* составим систему уравнений.

1) $ymin = xmin * k + add,$
2) $ymax = xmax * k + add$, отсюда

$k = (ymax – ymin) / (xmax – xmin) = (1200 – 500) / (255 – 50) \approx 3.415,$

$add = ymin – k*xmin = 500 – 3.415*50 = 500 – 170.75 = 329.25.$

Итак, $y(x) = x * 3.415 + 329.25$.

По это формуле рассчитаем значения Y для значений X в начале (50), в конце интервала (255) и в точках переключения выхода регулятора, занесем их в Табл. 1.2. Эти данные (две первые колонки) пригодятся при тестировании алгоритма.

Табл. 1.2. Расчетные и измеренные значения

X (PINA)	Yрасч	Yизмер (meas)	Погрешн Y		Вых PD1
			абс	отн, %	
50	500,00	499	1,00	0,08	1
51	503,42	503	0,41	0,03	1
…	…	…	…	…	…
155	858,58	858	0,58	0,05	1
156	861,99	861	0,99	0,08	0
157	865,41	865	0,40	0,03	0
…	…	…	…	…	…
254	1196,66	1196	0,66	0,05	0
255	1200,08	1199	1,08	0,09	0

…	…	…	…	…	…
151	844,92	844	0,91	0,08	0
150	841,50	841	0,50	0,04	0
149	838,09	837	1,09	0,09	1

Реализация программного вычисления требует использования переменных с плавающей запятой (float).

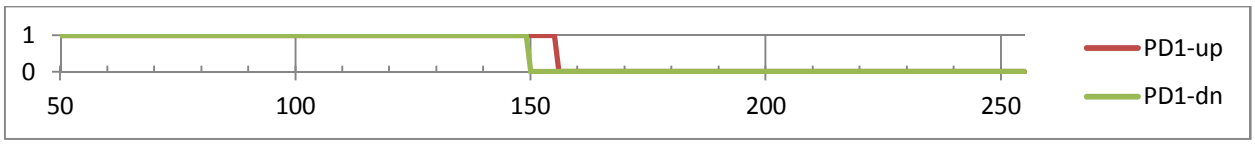
Другой способ заключается в использовании только целочисленной арифметики, тогда умножение на дробный множитель заменяется на две последовательные операции: сначала умножить на целое число, например, mul = 3415, затем разделить на div = 1000, либо mul = 341, div = 100. Этот способ выполняется быстрее и использует меньше памяти. При этом можно получить снижение точности результата.

Диапазон входных значений (50…255) предполагает разрешение 1/205, что примерно соответствует 0.5 % от 205, поэтому выбираем *div* = 1000. Слагаемое 329.25 округляем до ближайшего целого. На блок-схеме множитель обозначен *mul*, делитель задан числом 1000.

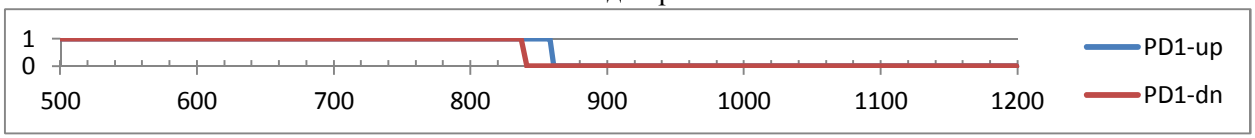
Для представления численных значений *y* (500…1200) в виде целого требуется двухбайтное число (***unsigned int*** 0…65565). Результат умножения x_max * mul = 3415 * 255 = 870825 больше 65535, поэтому один из сомножителей – mul объявим четырехбайтным (***unsigned long*** $0…2^{32} – 1$). После вычислений результат преобразуем к двухбайтному целому.

Функция дискретного (релейного) регулятора (рис. 1.2, в) реализуется двумя последовательными условными операторами. Первый по условию *y* < (*Set* – *Hys*) устанавливает выход *PD1* в Высокое состояние записью единицы в бит 1 регистра PORTD, второй по условию *y* > *(Set* + *Hys*) сбрасывает выход в Низкое состояние записью нуля.

Для регуляторов типа «Нагреватель» или «Охладитель» значение выхода PD1 в диапазоне Y от (Set – hys) до (Set + hys) зависит от «направления» изменения входа, то есть проявляется «гистерезис» (рис. 1.3), это снижает частоту переключения.



Состояние выхода при изменении *x*



Состояние выхода при изменении *y*

Рис. 1.3

Для смены типа регулятора надо только поменять знаки и/или границы сравнений в соответствии с законом управления.

2.3. Текст программы на языке С для AVR_GCC.

```
/*****************************************************
Chip type               : AT90S8515
AVR Core Clock frequency: 8.000000 MHz
*****************************************************/
#include <avr/io.h> //Файл выбора файла описания символьных констант io8515.h

//Глобальные переменные
unsigned char in; //беззнаковое целое форматом 1 байт
```

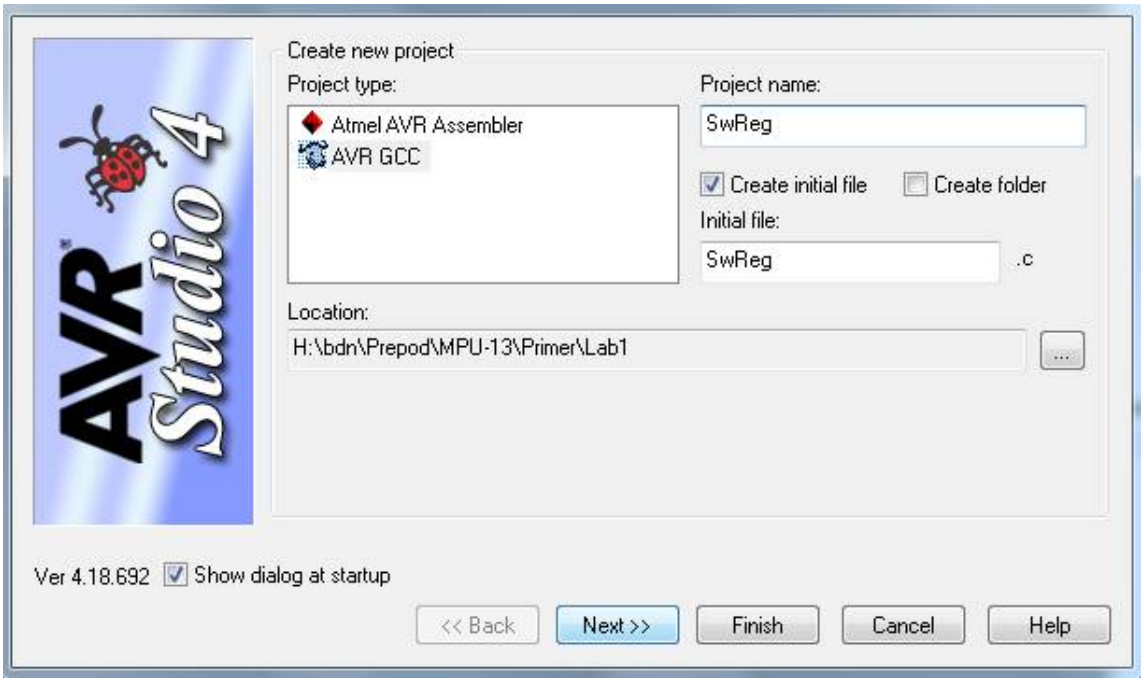
```
unsigned int meas; //беззнаковое целое форматом 2 байта

unsigned int calc(unsigned char in) {//Вычисление y=f(x)
  long mul=3415; //множитель форматом 4 байта
  unsigned int add=329;//351;//смещение
  unsigned int out;//возвращаемое значение
  mul = mul * in;
  mul = mul / 1000;
  mul = mul + add;
  out = mul;
  //out = in*mul/1000 + add; //Можно и так!
  return out; //Возврат результата вычислений
} // End of calc() ---------------------------------

void reg_sw(unsigned y) {
  int Set = 850;//Уставка
  char hys = 10;//Гистерезис
    if(y < Set - hys)
      PORTD |= (1<<1);//Уст-ть бит 1 порта D в 1, остальные без изменения
    if(y > Set + hys)
      PORTD &= ~(1<<1);//Обнулить бит 1 порта D, остальные без изменения
} // End of reg_sw() ----------------------------

int main(void) {
  // Настройка регистров портов ввода/вывода
  PORTA=0xFF; //Порт A – 8-битный вход с подтягивающими резисторами
  DDRD=(1<<PD1);  //Порт D бит 1 – выход, остальные - входы
  PORTD=(1<<PD1); //Уст-ть бит 1 порта D в 1, остальные в 0

  while (1) { //Основной цикл
    in = PINA; //Чтение 8-битного входа Порта A в переменную in
    meas = calc(in); //Вызов функции calc со входным параметром in,
присвоение результата переменной meas
    reg_sw(meas); //Функция reg_sw управляет выходом и ничего не возвращает
  } // Конец основного цикла
} // Конец функции main() ========================================
```

2.4. Проект, редактор текста и транслятор среды AVR Studio

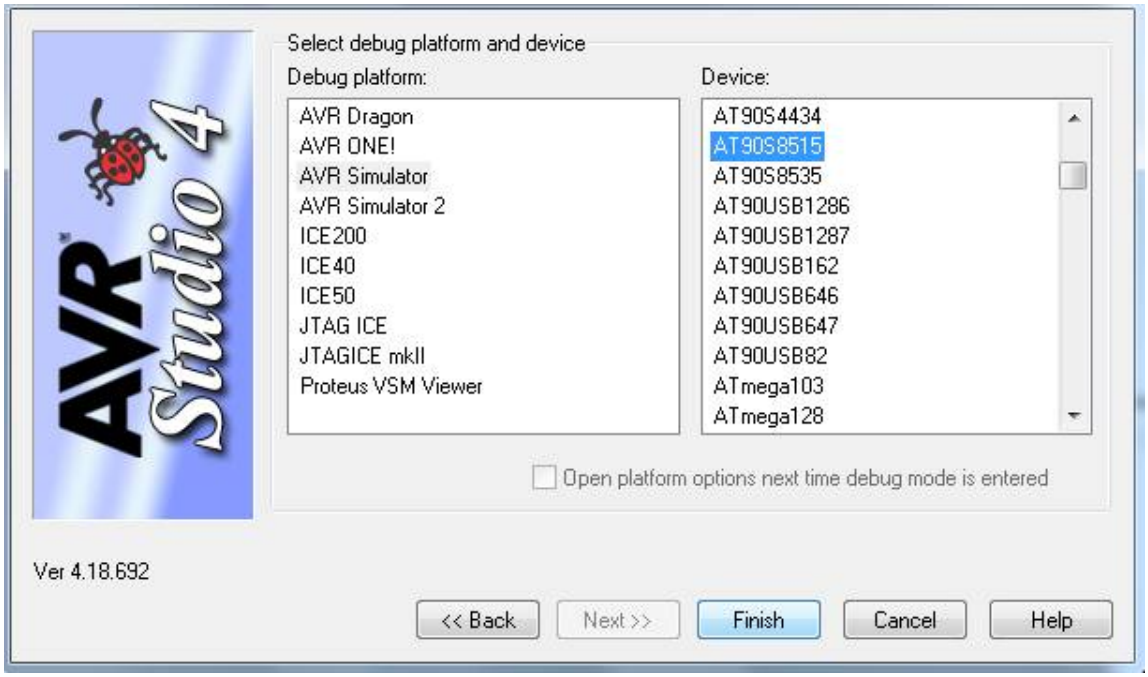
2.4.1. Создайте или откройте проект с готовым примером <..\Lab1\SwReg.aps>

Для создания проекта запустите среду разработки AVR Studio (Start →Atmel AVR Tools → AVR Studio 4).

Выберите во всплывающем окне New Project или выберите в меню Project → New Project, в обоих случаях появится всплывающее окно Create New Project (рис. 1.4, а).



а



б

Рис. 1.4

В окошке Project type кликните AVR GCC, в строке Location с помощью браузера укажите путь к вашему проекту, в строке Project name введите имя проекта (например, SwReg), оно автоматически дублируется в строке Initial File, нажмите кнопку Next. *Путь и имя файла должны состоять только из латиницы!*

Появится следующее окно диалога Select debug platform and device (рис. 1.4, б), в окошке Debug Platform в качестве отладчика по умолчанию выбран симулятор (AVR Simulator), в окошке Device выберите модель МК (AT90S8515), нажмите кнопку Finish, проект создан.

Теперь окно среды AVR Studio кроме меню и панелей пиктограмм содержит окно редактора исходного текста программы (пока пустое, но с именем), слева – окно файлов проекта, снизу – список окон, используемых на различных этапах проектирования.

Через меню Project → Configuration Options откройте окно настройки опций проекта (рис. 1.5), не забудьте задать значение частоты тактирования (Frequency: 8000000 Hz) и минимальный уровень оптимизации (Optimization: -O0), познакомьтесь с другими параметрами настройки. Для сохранения изменений нажмите кнопку «ОК», затем в меню Project → Save Project.

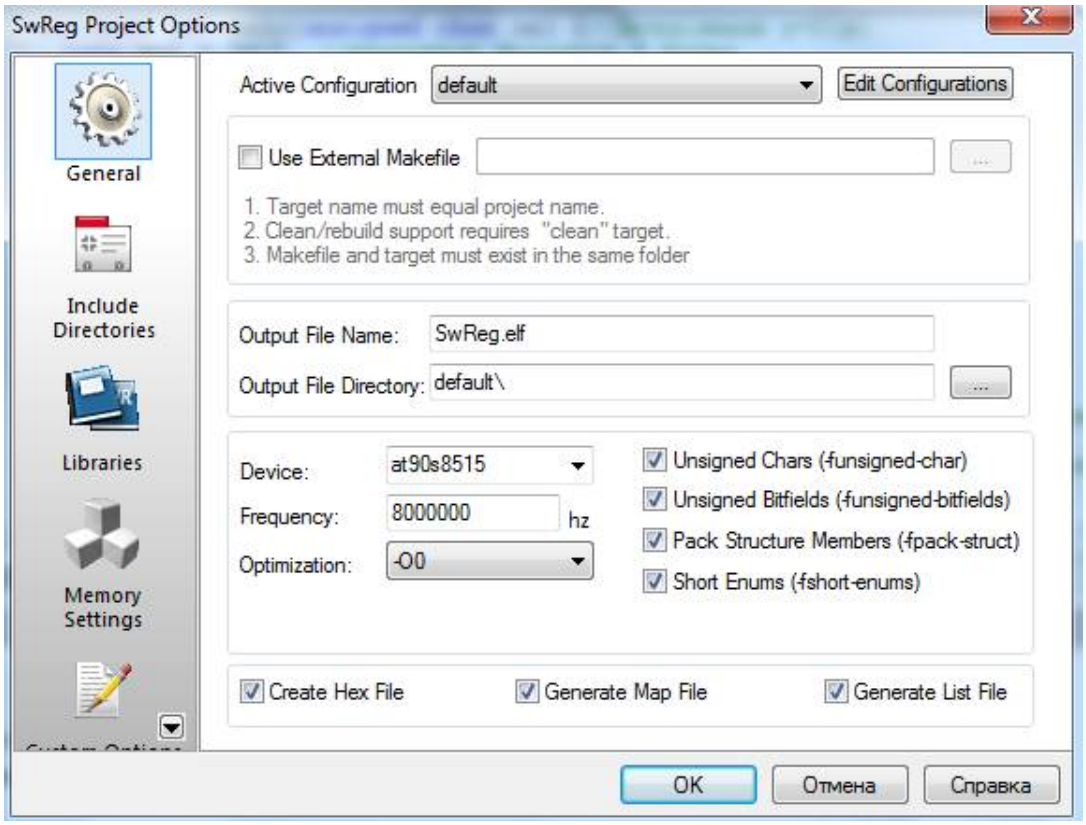


Рис. 1.5

В следующий раз для открытия проекта в любом браузере достаточно кликнуть на имени файла ..\SwReg.aps.

2.4.2. Текстовый редактор

В окне текстового редактора (..\SwReg.c) можно писать или редактировать ранее написанный исходный текст программы. Скопируйте в это окно текст программы на языке С из методички и сохраните.

2.4.3. Транслятор (компилятор)

Создание модуля с машинным кодом для загрузки или отладки выполняется через меню Build → Build (или [F7]).

В отсутствии синтаксических ошибок и ошибок выбора вспомогательных файлов в нижнем окне Build появится сообщение «Build succeeded with 0 warnings…» (рис. 1.6), что говорит об успешном завершении трансляции, построении нового загрузочного модуля и других вспомогательных файлов.

В предыдущих строках находится информация об использовании памяти программ (Program) и памяти данных (Data), что позволяет оценить свободные ресурсы, качество транслятора и влияние параметров оптимизации.

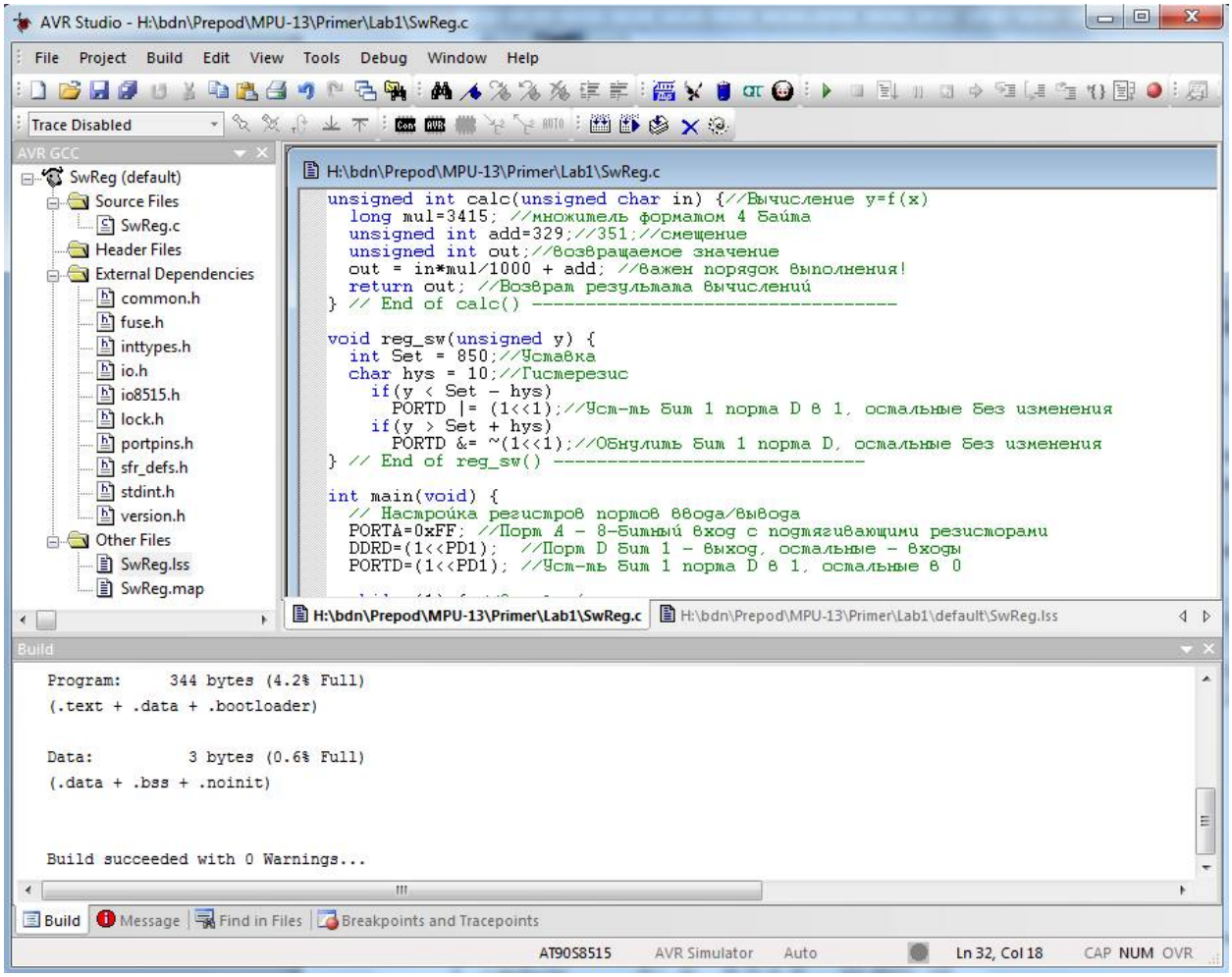


Рис. 1.6

Не нулевое число предупреждений, например, «Build succeeded with 1 warnings…», предупреждает о наличии допустимых отклонений в синтаксисе. В списке сообщений транслятора в окне Build строки предупреждений помечены желтым маркером. При двойном клике на эту строку в окне исходного текста появляется синий указатель на строку, вызвавшую предупреждение. Лучше добиваться отсутствия предупреждений.

При обнаружении недопустимых ошибок транслятор выдает сообщение «Build failed with N errors and M warnings..» и не обновляет ранее созданный загрузочный модуль и другие файлы. Строки сообщений об ошибках подсвечены красным маркером, двойной клик также помогает найти строки исходного текста с ошибкой. Начинать разбор всегда следует с первой ошибки.

При успешной трансляции и указанных параметрах настройки проекта в папке ..\default транслятор AVR_GCC создает следующие файлы:

SwReg.hex – двоичный файл с машинными кодами для загрузки в память программ МК (во FlashROM),

SwReg.elf – двоичный файл с машинными кодами и исходным текстом программы для символьной отладки (в отладчике AVR Studio или в системе моделирования Proteus VSM),

SwReg.eep – двоичный файл только с машинными кодами для загрузки в энергонезависимую память данных МК (в EEPROM) – в данном случае пустой,
SwReg.lss – текстовый файл листинга с машинными кодами, исходным текстом программы, сообщениями об ошибках и предупреждениях транслятора при их наличии;
SwReg.map – текстовый файл распределения памяти МК между модулями программы и данных.

2.5. Тестирование программы в отладчике-симуляторе AVR Studio

2.5.1. Отладчик и его инструменты

Вызов отладчика выполняется из меню Debug → Start Debugging, естественно, при наличии загрузочного модуля. Можно объединить процессы трансляции проекта и запуск отладчика командой меню Build → Build and Run.

В результате на первой исполняемой строке программы (в функции main) появляется указатель отладчика в виде желтой стрелки (рис. 1.7). Это признак работы отладчика, другими признаками является изменение видимости пунктов меню и пиктограмм используемых при отладке, появление дополнительных окон для просмотра ресурсов МК.

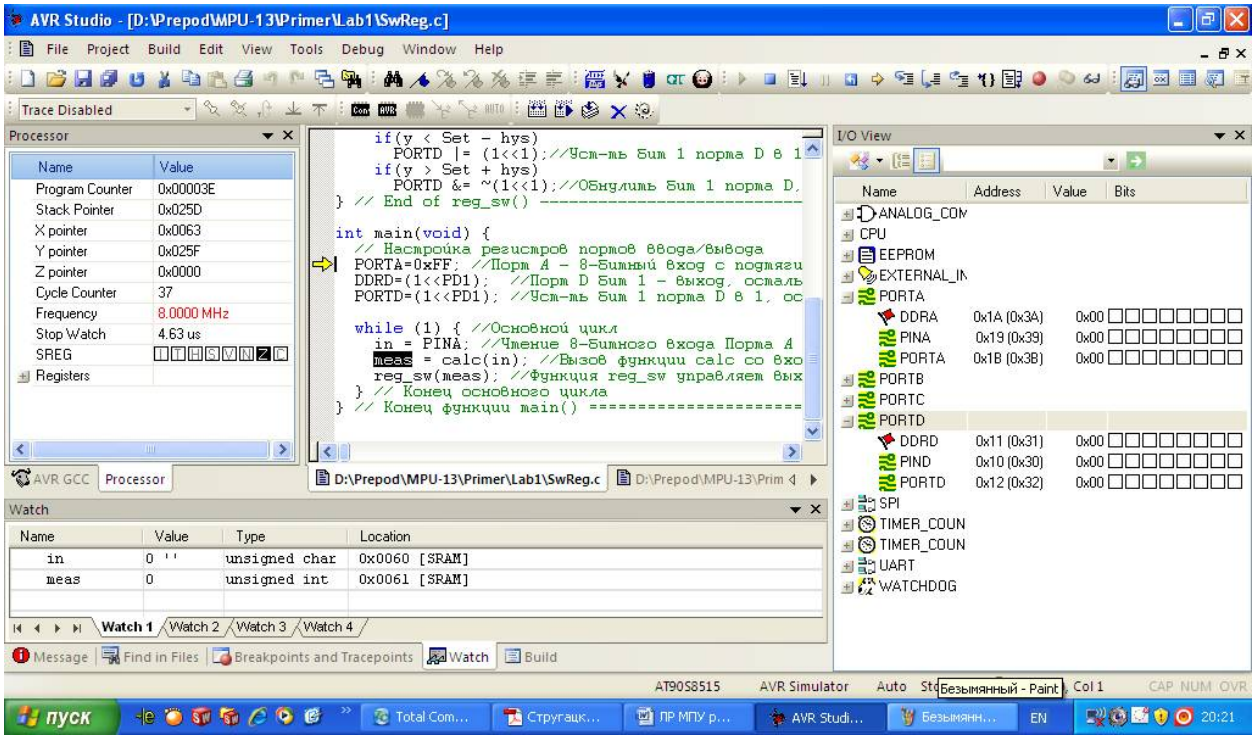


Рис. 1.7

Левое окно проекта AVR GCC называется окном Processor. Счетчик команд Program Counter указывает текущий адрес ячейки памяти программ. Счетчик машинных циклов Cycle Counter отсчитывает время после рестарта, в строке Stop Watch время в микросекундах [us] или миллисекундах [ms], для его расчета используется значение частоты тактирования Frequency: Stop Watch = Cycle Counter / Frequency.

Значение частоты Frequency можно изменить на требуемое (8 МГц) через команду меню Debug → AVR Simulator Options → Device Selection → Frequency.

Для измерения интервала времени между командами/строками программы текущие значения Stop Watch, Cycle Counter можно обнулять на любом шаге через всплывающее контекстное меню окна Processor.

Обратите внимание на ненулевое значение времени после рестарта программы на языке С, 37 машинных циклов ушло на выполнение настройки переменных и системных

функций языка. Длительность этого интервала зависит от числа переменных и используемых системных функций, для больших программ может составлять тысячи и даже десятки тысяч машинных циклов.

Пошаговое (построчное) выполнение программы выполняется командами Debug → Step Into (<F11>) и Debug → Step Over (<F10>), последняя при встрече функции выполняет ее без показа построчного выполнения.

Справа от окна с текстом программы появилось окно регистров ввода/вывода I/O View, на рис. 1.7 это окно имеет вид Tree View (выбор вверху окна). Кликом по крестикам возле PORTA и PORTD откроем выпадающие списки связанных с ними по смыслу регистров. Поля Value и Bits отображают текущие значения регистров в виде числа (шестнадцатеричное или десятичное) и двоичных бит (белый квадратик – 0, черный – 1).

Вы можете произвольно менять их значения (кликом), но лучше это делать осмысленно – менять только те значения, которые позволяют имитировать воздействия на входы. При работе с портами это биты регистров PINx, выводов, настроенных на ввод, в нашем примере, PINA. При выполнении программы выводы, настроенные как выходы, будут менять свое состояние в соответствии с алгоритмом. В нашем случае это выход PD1, программа устанавливает или очищает бит PORTD.1, с отставанием на один такт его значения копируются битом PIND.1.

2.5.2. Тестирование функции main

В первую очередь освоим тестирование готового примера «по крупному» – без захода внутрь функций, проверим соответствие входных и выходных данных. Для этого удобно использовать команду отладки Step Over <F10>, если случайно зайдете в функцию (подпрограмму), то для выхода используйте команду Step Out <Shift + F11>.

Выполните три строки инициализации и одну строку основного цикла. Обратите внимание на изменение состояния наблюдаемых регистров портов A и D. Что заставило «почернеть» PIND1?

Просмотр значений именованных переменных удобно выполнять с помощью окна Watch. Выделите мышью переменную для наблюдения, нажмите правую кнопку мыши и во всплывающем контекстном меню выберите пункт «Add Watch». При запросе первой переменной появится плавающее окно «Watch» в котором отображаются имя переменной (Name), ее значение (Value), тип (Type) и адрес размещения (Location). Значение отображается в десятичном или шестнадцатеричном (Hex) формате, для переменных типа char в режиме Hex дополнительно в апострофах индицируется значение в коде ASCII. С помощью контекстного меню можно выбрать удобный формат («галка» возле строки «Display … Value as Hex»). Значение переменной можно не только наблюдать, но и редактировать.

Значения глобальных переменных доступны всегда, локальные переменные не доступны для просмотра вне зоны их видимости – «Not in scope».

Тестирование функции calc заключается в проверке погрешности преобразования входных данных (50…255) в данные для индикации и регулирования (500…1200). Для этого в окне регистров ввода/вывода с помощью контекстного всплывающего меню убирается «галка» возле «Hexadecimal Display», после этого в регистр PINA (вход данных) можно вручную вводить любое десятичное число. Следует проверить выдачу функции calc

(переменную y), вводя данные в начале диапазона (50), в конце диапазона (255) и в середине диапазона. Вспоминаем о Табл. 1.1, заполняем графу «Yизмер meas». По полученным данным рассчитываем абсолютную и относительную погрешность, делаем выводы о характере относительной погрешности: укладывается ли в задание, с чем связана погрешность.

Теперь можно тестировать функцию reg_sw. Следует снять передаточную характеристику, то есть зависимость изменения состояния выхода регулятора Reg_Sw (выход PD1, бит POTRD.1) от входного «измеренного» значения *y* как при росте, так и при спаде последнего в области значений *Set* ± hys. Результат поместим в графу «Вых PD1» Табл. 1.1, построим графики (рис. 3.2).

2.5.3. Тестирование остальных функций

calc: Посмотрите на порядок вычислений внутри функции calc. Обратите внимание на порядок вычислений и последовательность значений локальных переменных. Есть ли способы уменьшить время выполнения за счет оптимизации порядка вычисления?

reg_sw: последовательность выполнения условных операторов и операции с битами.

2.5.4. Тестирование времени реакции алгоритма и его составных частей

Для оценки времени выполнения следует обнулить значения Stop Watch и Cycle Counter, выполнить одну или более строк программы, записать новые значения Stop Watch и Cycle Counter.

Проверьте, зависит ли время выполнения функций от входных данных.

3. Индивидуальное задание

3.1 Задание

Таблицы 1.3 и 1.4 содержит лишь параметры вариантов, задание в полной форме предполагает подстановку этих параметров в текст, находящийся в разделе «2.1. Задание» данной работы.

Табл. 1.3. Варианты первого индивидуального задания

№	*X*	*y*	*Set*	*hys*	*Reg_out*
00	*PA0-7*, 50-255	500…1200	850	10	*PD1*, Зак.1
01	PB0-7,0-100	-50…+50	-25	5	PC0, Зак.2
02	PD0-7, 0-255	0…500	250	50	PA3, Зак.2
03	PC0-7, 0-200	0.0…100.0	50	0.5	PB0, Зак.1
04	PA0-7,50-255	5.0…60.0	30	15	PD5, Зак.4
05	PB7-0, 0-200	150…400	250	70	PD0, Зак.3
06	PC7-0, 0-100	-80.0…-30	-55	2.5	PA4, Зак.4
07	PD0-7, 0-255	0…1000	750	5	PB7, Зак.1
08	PC0-7, 0-200	1000-2000	1400	100	PA1, Зак.3
09	PA0-7, 0-255	200…500	350	50	PB0, Зак.2
10	PB0-7, 50-255	-100…0	-60	10	PD7, Зак.2
11	PD0-7,50-255	-40…+70	25	5	PA0, Зак.1
12	PB0-7, 25-225	500…2500	1800	50	PA2, Зак.2
13	PC0-7, 250-25	0…250	175	5	PB3, Зак.3
14	PD0-7, 255-0	50…700	200	25	PC1, Зак.4
15	PB0-6, 5-125	0…1200	500	100	PA0, Зак.1
16	PA1-7, 2-254	50…250	150	50	PD7, Зак.2
17	PB0-6, 0-99	-40…+60	20	15	PA6, Зак.3
18	PC2-7, 4-200	2500…3500	3000	100	PB5, Зак.4

19	PD0-5, 4-64	-100…+100	0	25	PC4, Зак.1
20	PA1-6, 10-110	600…2600	1500	100	PD3, Зак.2

Табл. 1.4. Законы ключевого (релейного) регулирования

Наименование	*Алгоритм*
Закон 1 «Нагреватель»	Если y < Set - hys, out=1; если y > Set + hys, out=0
Закон 2 «Охладитель»	Если y < Set - hys, out=0; если y > Set + hys, out=1
Закон 3 «В коридоре»	Если Set-hys < y < Set + hys, out=1, иначе, out=0
Закон 4 «Вне коридора»	Если Set-hys < y < Set + hys, out=0, иначе, out=1

3.2 Разработка алгоритма и схемы

Нарисуйте эскиз схемы от руки или в схемном редакторе ISIS Proteus. Схема позволяет наглядно отобразить назначение выводов МК в данном проекте.

Разработка алгоритма сводится к коррекции блок-схемы программы, выбора входов и выхода, зависимости Y от X (составить таблицу), выбору значений констант преобразования, выбору формата основных и промежуточных переменных, коррекции алгоритма регулирования.

Для работы с отрицательными значениями Y требуется использовать типы signed. Для работы с дробными значениями уставки Set или гистерезиса his требуется значения Y увеличить в 10 раз (фиксированная запятая в формате xxx.x).

3.3 Разработка программы

Начинается с коррекции текста исходной программы с учётом изменений в алгоритме.

Далее выполняется трансляция, удаляются возможные ошибки трансляции (всегда начиная с первой!).

3.4 Тестирование программы

Тестирование логики и точности вычислительных операций выполняется аналогично тестированию готового примера. На этом этапе исправляются возможные логические ошибки, измеряются времена выполнения.

После успешного самостоятельного тестирования выполняется демонстрация тестирования преподавателю.

3.5 Оформление отчета

Общие требования см. Введение. Для отчета фиксируются вводимые данные и результаты работы программы с оценкой погрешности, времена выполнения этапа инициализации и основного цикла, расходы памяти программ (ПП) и памяти данных (ДП).

Лабораторная работа 2 [1, p. 3-5]

Генерация импульсов с варьируемыми параметрами с использованием прерывания по переполнению таймера/счетчика

Цели и задачи

- Знакомства с вводом/выводом по прерываниям [1, p. 3.5, 4]
- Знакомство с устройством таймеров/счётчиков [1, p. 5.3, 5.4]
- Знакомство с алгоритмами генерации импульсов [1, p. 5.3, 5.4]
- Изучение алгоритма генерации импульсов в прерывании по переполнению таймера [1, p. 5]

1. Основные понятия

1.1. Ввод/вывод по прерываниям

Позволяет прервать выполнение основного цикла для выполнения малого количества процедур чтения или записи и возвращения в основной цикл. Это обеспечивает время реакции много меньшее периода основного цикла. Естественно, при средней длительности обработки всех прерываний много меньшей длительности основного цикла.

Сами процедуры чтения/записи выполняются программным способом и оформляются в виде подпрограммы обслуживания прерывания [ISR – Interrupt SubRoutine]. Каждое прерывание имеет свои вектор (адрес ISR), флаг запроса (xxIF, регистры GIFR, TIFR) и бит маскирования, то есть индивидуального разрешения (xxIE, регистры GIMSK, TIMSK). Команды SEI/CLI выполняют общее разрешение/запрет прерываний.

1.2. Таймеры/счётчики

Состоят из аппаратного счетчика TCNTx (x – номер счетчика от 0 до 5) разрядностью N = 8 (TC0, 2, 4) или N = 16 (TC1, 3, 5) бит, дополнительных узлов прерывания, сравнения и захвата, специальных входов и выходов, регистров данных, состояния и управления.

Счетчик тактируется либо от внешних импульсов (режим *счетчика*), либо от внутренних импульсов (режим *таймера*). Внутренние импульсы формируются делением частоты системного генератора fclk на числа из ряда 1, 8, 64, 256, 1024, настройка тактирования выполняется тремя битами CSxy регистра управления TCCR(B)x, здесь x – номер таймера/счетчика (0, 1…), y – номер бита.

Счет в регистре TCNTx обычно идет на увеличение. При достижении максимального значения TOP = 2^N – 1 (255 или 65535) происходит автоматический сброс в ноль, этот момент называется *переполнением* (фиксируется установкой бита TOVx), счет продолжается. Счетный регистр TCNTx доступен по чтению и записи. Запись позволяет уменьшать интервал времени между переполнениями. При общем разрешении прерываний и установленном бите TOIEx регистра маскирования TIMSK, переполнение формирует запрос прерывания по адресу TIMERx_OVF_vect.

1.3. Алгоритмы генерации импульсов

1.3.1. Разновидности импульсов с варьируемыми параметрами

Импульсы одноканальные (рис. 2.1, а-в) состоят из интервала с высоким уровнем (импульс длительностью $t_и$) и с низким (пауза длительностью $t_п$), период сигнала $T = t_и + t_п$, частота $f = 1/T$. Варьируемым параметром выступает период T или частота f, это частотная модуляция (ЧМ). Вариацию длительности импульса ti или паузы tп при постоянной

длительности периода называют широтно-импульсной модуляцией (ШИМ). Задача МК состоит в периодическом формировании фронта и среза импульса через заданные интервалы времени, два события на период.

Импульсы двухканальные (рис. 2.1, г-ж) имеют одинаковую частоту, каждый канал может иметь собственную длительность импульса / паузы, фронты импульсов имеют во времени сдвиг, называемый задержкой $t_з$ (задается в секундах) или сдвигом фаз $t_з/T$ (задается в долях периода) – фазовая модуляция (ФМ). Рис. 2.1, е, ж – управление ключами Н-моста (или полумоста) требует равных по длительности сигналов ($t_{и1} = t_{и2}$, $t_{п1} = t_{п2}$), сдвинутых ровно на половину периода. Здесь четыре события на период.

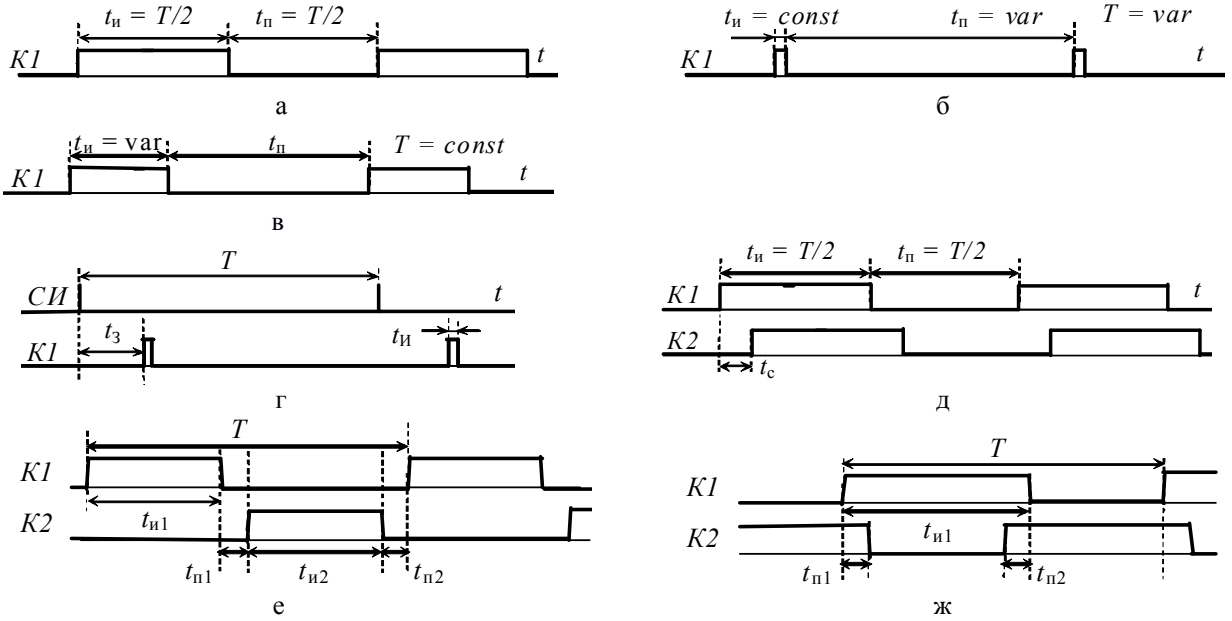


Рис. 2.1

1.3.2. Способы генерации импульсов с варьируемыми параметрами

При вариации длительности импульса, периода или задержки шаг варьирования постоянный по времени, назовем его квантом времени $\Delta t = const$. При вариации частоты $\Delta f = const$ шаг варьирования длительности переменный $\Delta t = 1/f - 1/(f + \Delta f) = var$. Задача формирования варьируемого интервала времени сводится к отсчету квантов времени. В зависимости от способа формирования кванта, способа счета квантов и способа формирования фронтов и срезов импульсов различают три подхода.

1. Самый простой способ – чисто программный. Кванты времени формируются из задержек и частных циклов, счет квантов, формирование фронтов и срезов программные [1, р. 5.2].

2. Использование таймера/счетчика в режиме таймера с прерыванием по переполнению позволяет разгрузить процессор от задач формирования и счета квантов времени. Частота тактирования таймера ftcx определяет длительность кванта времени $\Delta t = Ttcx = 1/ftcx$. Значение, загружаемое в счетный регистр TCNTx определяет длительность очередного интервала переполнения $ti = nti * \Delta t$, $nti \le TOP$, здесь i – номер события.

Подпрограмма прерывания должна иметь число ветвей, соответствующее числу событий на периоде повторения генерируемых импульсов, ветви выполняются по одной за прерывание с автоматическим переходом к следующей. В каждой ветви программно формируется фронт или срез на требуемом выводе, в счетный регистр TCNTx загружается число –nti (эквивалентно «TOP – nti») для формирования длительности следующего интервала. Структура подпрограммы будет рассмотрена в Примере 2.

Простой пример генерации одноканального ШИМ сигнала с постоянным периодом 512 мкс (1953 Гц) и варьируемой длительностью n * 64 мкс, где n = 1, 2 … 7. Выберем 8 битный таймер/счетчик TC0. Для удобства пересчёта выберем частоту тактирования процессора fclk = 1 МГц (1 такт – 1 мкс). Для получения длительности кванта 64 мкс «подходит» коэффициент деления k = fclk/ftc0 = 64. На рис. 2.2 приведена временная диаграмма ШИМ сигнала на выходе PB0, сигнала переполнения TOV0 и значений счетного регистра TCNT0 при заданной длительности импульса 64 мкс, паузы 448 мкс.

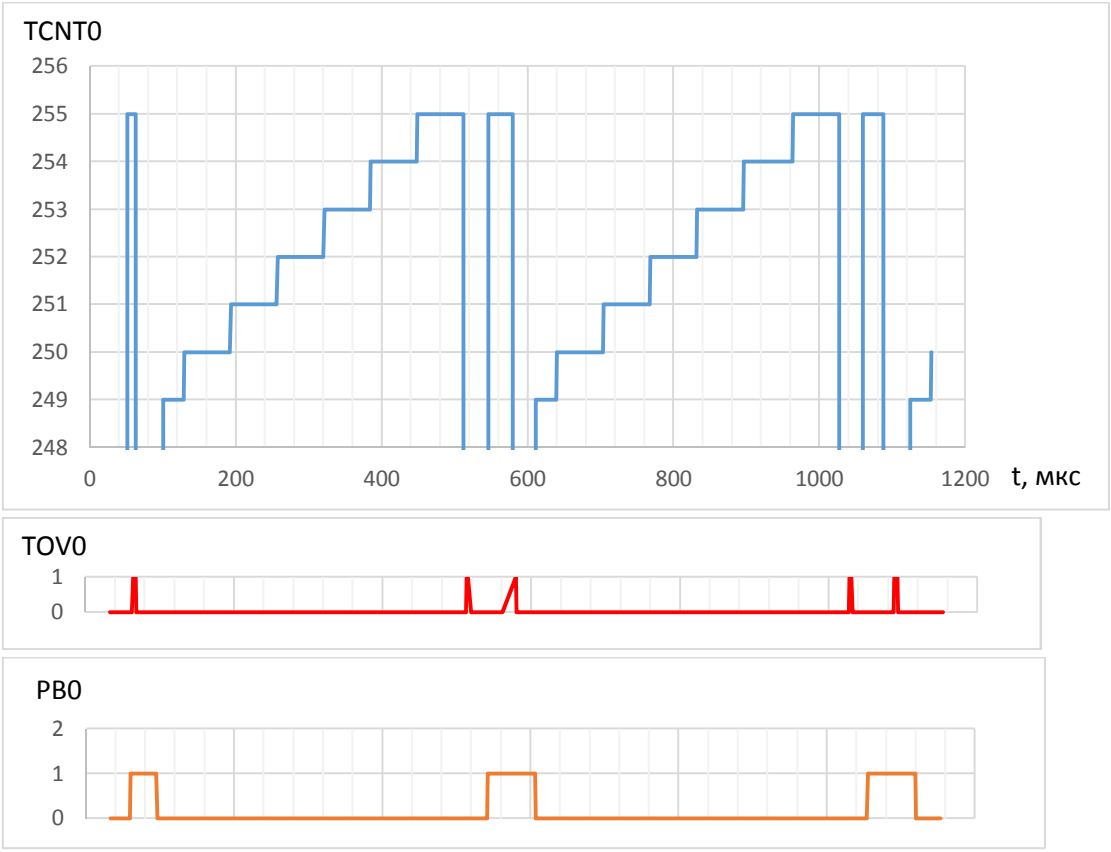


Рис. 2.2

График TCNT0 показывает значительную погрешность при формировании первого такта после записи нового значения в счетный регистр. Это приводит к плавающей задержке фронтов и срезов импульса PB0 по отношению к сигналу переполнения TOV0 (32, 29, 27, 27, 32 мкс). Анализ погрешности длительности интервалов (-28, -2, 1, 0, 1 мкс) показывает, что джиттер сравнительно мал, кроме первого импульса.

Такой способ особенно удобен при формировании многоканальных импульсов. Для минимизации времени обслуживания прерывания операции по расчету длительностей интервалов лучше выполнять в основном цикле.

3. Наименьшую погрешность в формировании импульсов обеспечивает использование выхода сравнения OCx таймера/счетчика. Здесь таймер/счетчик формирует импульсы аппаратно, процессор нужен только для настройки и изменения варьируемого параметра импульсов.

Выбор этого способа ограничивается числом выходов сравнения и составом предусмотренных режимов таймера/счетчика.

2. Пример 2: «Генерация импульсов с варьируемыми параметрами»

2.1. Задание к Примеру 2 (Вариант 0)

Генерация одноканальных импульсов (рис. 2.1, а) на дискретном выходе с варьируемой частотой f(d) = 5000 + 50*d [Гц], ширина импульса равна паузе ($t_и = t_п$), параметр вариации d = 0, 1, ... 100 (число шагов вариации D = 100) вводится через параллельный порт.

2.2. Схема и алгоритм

Схема (рис. 2.3) использует 7 входов порта A (PA0…PA6) для ввода числа d и выход PB0 для генерации импульсов.

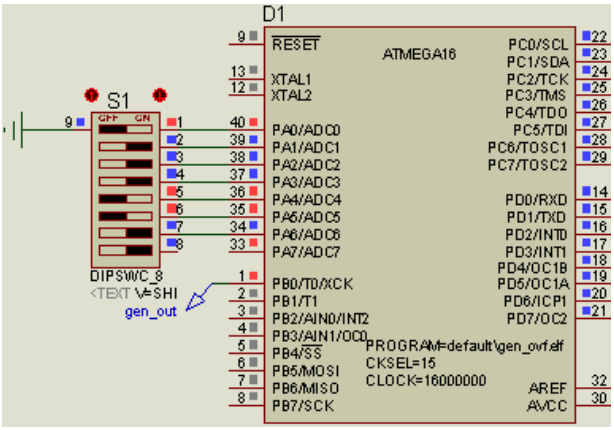


Рис. 2.3

Блок-схема обобщенного алгоритма генерации импульсов управления с варьируемыми параметрами с использованием прерывания по переполнению таймера/счетчика представлена на рис. 2.3.

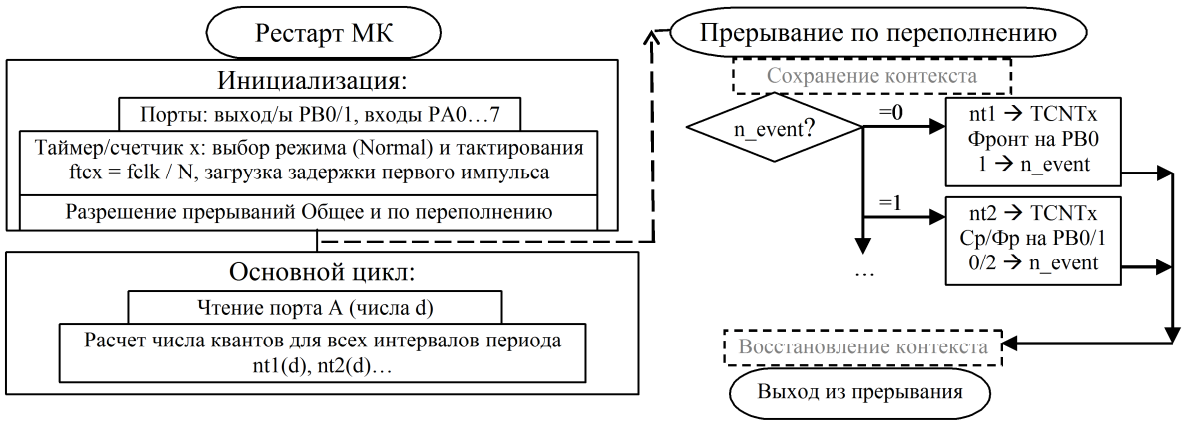


Рис. 2.4

Выберем в качестве выхода для формирования импульсов вывод PB0 (и PB1 для импульсов второго канала), значение варьируемого параметра d будем считывать через параллельный порт *PA*.

В подпрограмме прерывания по переполнению будут формироваться фронты и срезы импульсов. Для одноканальных импульсов требуется формировать по одному фронту и срезу за период T, то есть два переполнения (события) за период, для двухканальных – до четырех переполнений (по два фронта и среза) и т.д.

Формирование постоянных и малых по длительности интервалов (до сотни машинных циклов) лучше выполнять программно, не выходя из прерывания, заполняя интервал времени требуемым количеством «пустых» операторов `__asm ("nop")`. В этом случае уменьшается число переполнения за период.

В этом же прерывании выполняется загрузка счетного регистра TCNTx числом, пропорциональным длительности следующего интервала. Для одноканальных импульсов это интервалы импульса (число nt1) и паузы (число nt2), для двухканальных – интервалы между фронтами (число nt1), между фронтом второго и срезом первого (nt2), между срезами (nt3), между срезом второго и фронтом первого (nt4). В ряде вариантов часть длительностей совпадает (nt1 = nt2 или nt1 = nt3 и nt2 = nt4), число переменных уменьшается.

Для выбора требуемого набора действий в прерывании требуется объявить переменную-счетчик событий n_event статического или глобального типа с начальным нулевым значением. В каждом прерывании выбирается ветвь с текущим номером действия, выполняется требуемое действие (фронт или срез на PB0 или PB1), регистру TCNTx присваивается требуемое значение (nti), номер события меняется на следующий. Для одноканальных импульсов вместо переменной-счетчика n_event можно использовать значение бита порта PORTB.0.

В основном цикле выполняется считывание числа *d* с порта входа PA и перерасчет целочисленных значений, определяющих длительности варьируемых интервалов (nt1…nt4).

На этапе инициализации настраиваются:

1) выход/ы порта PB0 (PB1) для генерации импульса/ов, вход PA для ввода числа d (подтягивающие резисторы),

2) таймер/счетчик:

- задать частоту тактирования ftc битами CSxy регистра управления TCCRxB (и частотой тактирования процессора fclk),
- задать интервал времени до первого переполнения записью числа в счетный регистр TCNTx,
- разрешить прерывания по переполнению установкой бита TOIEx регистра TIMSK,

3) выполнить общее разрешение прерывания установкой бита I регистра состояния SREG, выполняется командой SEI.

2.3. Расчет параметров алгоритма

• Определить число неперекрывающихся интервалов (от N = 2 до N = 4), из которых состоит период генерируемых импульсов.

Написать для каждого интервала закон изменения длительности ti(d) [с] или [мс] или [мкс], i = 0, ... N – 1. Для варьируемого интервала ti(d) = ti_min + d*Di, где Di = (ti_max – ti_min)/100. Для постоянных это константа :). Нарисовать временную диаграмму формируемых импульсов и значений счетного регистра TCNTx, фронты и срезы импульсов должны совпадать с моментами переполнений.

Пример: диаграмма аналогична рис. 2.2 (переполнение по фронту и по срезу), но длительность импульса равна длительности паузы t1(d) = t2(d) = T(d)/2 = 1 / f(d) / 2 = 1 / (5000 + d * 50) / 2 = 1/(10000 + 100*d) [с] (численные значения в Табл. 2.1).

• По закону изменения длительности определить минимальное значение кванта времени dt и минимальную частоту тактирования таймера ftc_min = 1/dt. При заданной вариации временного параметра (T, ti и пр.) величина шага постоянная, при вариации частоты зависит от d.

Пример: с ростом частоты период уменьшается, поэтому минимальный шаг изменения длительности импульса (и паузы) будет при d = 99 и 100: dt = t1(100) – t1(99) = = 1/(10000 + 100*99) – 1/(10000 + 100*100) = 0.25 мкс, то есть ftc_min = 4 МГц (Табл. 2.2).

• Выбрать частоты тактирования таймера ftc и процессора fclk:

1) ftc = fclk / K, K = 1, 8, 64, 256, 1024 - предделитель таймера (выбор битами CSxy),
2) fclk в диапазоне от 1 до 16 МГц, выгоднее выбирать ближе к 16 МГц,
3) ftc = ftc_min * M, M – положительное целое.

Пример: при fclk = 16 МГц можем выбирать тактирование из ряда 16 МГц, 2 МГц, 250 кГц… , выбираем ftc = fclk = 16 МГц, то есть K = 1, M = 4.

- Записать выражения для длительностей интервалов в виде числа тактов выбранной ftc: nti(d) = ti(d) / ftc, частота и время должны быть [с] и [Гц] или [мс] и [кГц] или [мкс] и [МГц]. Выражения должны содержать только целые числа (см. Лаб.1).

Пример: выражение для числа тактов при выбранном тактировании nt1(d) = t1(d) * ftc = 16000000 / (10000 + 100*d) = 160000 / (100 + d), см. nt1 и nt1/ftc в Табл. 2.1.

- Выбрать разрядность таймера по максимальным значениям nti, TC0 – 8, TC1 – 16.

Пример: максимальное число тактов nt1(0) = 160000 / 100 = 1600; это больше 255, но меньше 65535, поэтому выбираем 16-битный таймер/счетчик TC1.

2.4. Текст программы на языке С

```
#include<avr/io.h> /* Символьные константы ввода/вывода */
#include<avr/interrupt.h> /* Символьные константы прерываний */

charn_event= 0; //Переключатель события
unsigned int nt1 = 0; //Число квантов
static char d = 100; //Уставка частоты

// ***************************************
ISR(TIMER1_OVF_vect) { //Обработчик прерывания
  TCNT1 = -nt1; //Загрузка числа квантов
if(n_event == 0) { //Выбор события
    PORTB |= (1<<0); //Фронт
n_event= 1; //Следующее событие
  } else {
    PORTB &= ~(1<<0); // Срез
n_event= 0; //Следующее событие
  }
} // Выход из прерывания

unsigned int calc_nt1(char d) { //Расчет числа квантов
unsigned long nti_l;
nti_l = 160000L /(100 + d) - 47;
return (unsigned int) nti_l;
}//**************************

int main() {
  nt1 = calc_nt1(d);//для тестирования и расчета задержки первого импульса
  PORTA = 0xff; //Включить притяжку входов PA0…7
  DDRB = (1<<0);//Настройка выхода импульсов
  // T/C1 init
TCCR1A = 0x00; //Mode: Normal top=FFFFh
  TCCR1B = (1<<CS10); //ftc1 = fclk
  TCNT1 = -nt1;//загрузка для первогоимпульса
TIMSK =(1<<TOIE1); //разрешение прерывания переполнения TC1
sei();// Global enable interrupts – общее разрешение прерываний
while (1) {
    d = PINA; //Чтение параметра вариации
    nt1 = calc_nt1(d); //Вычисление числа квантов
  };
} //-O0 – нет оптимизации – основной выбор для тестирования всех Примеров (!);
```

Табл. 2.1 Расчетные и измеренные параметры алгоритма

	Расчет	Измерения	Оценка погрешности

d	fset, Hz	Tset, us	t1, us	dt, us	nt1	nt1/ftc, us	t1, us	T_pr, us	f_pr, Hz	ef, Hz	ef, %
0	5000	200.00	100.00		1600	100.00	102.94	205.75	4860	140	1.40
1	5050	198.02	99.01	0.99	1584	99.00	101.94	203.75	4908	142	1.42
2	5100	196.08	98.04	0.97	1568	98.00	100.94	201.75	4957	143	1.43
49	7450	134.23	67.11		1073	67.06	70	139.88	7149	301	3.01
50	7500	133.33	66.67	0.45	1066	66.63	69.56	139.019	7193	307	3.07
99	9950	100.50	50.25		804	50.25	53.19	106.31	9406	544	5.44
100	10000	100.00	50.00	**0.25**	800	50.00	52.94	105.75	9456	544	5.44

Столбец «dt, us» получен как разность значений «t1» при изменении d на 1 и наглядно позволяет выбрать Ttc_max = MIN(dt).
Столбец «nt1» получен при тестировании функции calc_nt1() в отладчике-симуляторе AVR Studio.
Столбец «dt, us» получен как разность значений «t1» при изменении d на 1

2.5. Тестирование генерации в симуляторе AVR-Studio

После входа в режим отладки не забудьте проверить частоту симуляции в окне Processor в строке Frequency, она должна соответствовать выбранной fclk (Пример: 16 MHz). Для изменения частоты используйте меню *Debug AVR Simulator Options Device Selection Frequency*.

Для ввода очередного значения d используйте окно I/O View (PORTA Value), поместите в окно Watch переменные d, nt1. Так удобно тестировать функцию вычисления calc_nt1.

Для быстрого прогона большого количества строк программы в отладчике существует несколько команд. В данном случае удобно использовать предварительную расстановку точек останова (Breakpoints<F9>) и команду запуска выполнения (Run - <F5>). После очередного запуска выполнения программа выполняется, пока не встретит одну из точек останова. В процессе выполнения не индицируется изменения ресурсов МК, из команд отладки активны только «Break» <Ctrl+F5> (останов программы) и «Reset» <Shift+F5> (рестарт программы).

Поставьте точки останова отладчика на команды изменения состояния выхода PB0 в функции обработчика прерывания по переполнению таймера 1 и измерьте длительность импульса, паузы и периода на первых 2-3 импульса (сравните с Табл. 2.1).

Время симуляции отображается в окне *Processor* в виде машинных тактов (*CycleCounter*) и в мкс/мс (*StopWatch*). После очередной остановки эти значения можно обнулять командами всплывающего меню (*ResetCycleCounter, ResetStopwatch*).

Сравните с расчетным значением для n = 0, стабильна ли длительность?

Для оценки характера разброса погрешности значения частоты (периода) от значения d повторите измерения при различных значениях уставки вариации. Обратите внимание, что после ввода нового d следует дождаться выполнения расчета nti в основном цикле.

В табл. 2.1 приведены рекомендуемые для тестирования значения d: на краях диапазона и в середине, плюс на ближайших значениях – 1, 2, 49, 50, 99, 100.

Сравнение расчетных и измеренных значений t1 Примера 2 показывает постоянное превышение формируемой длительности на 2.94 мкс. Подобную постоянную погрешность легко скорректировать уменьшением числа, загружаемого в счетный регистр. Для Примера это 2.94 мкс * 16 МГц = 47 тактов, то есть nti_1 = 160000L / (100 + d) – 47.

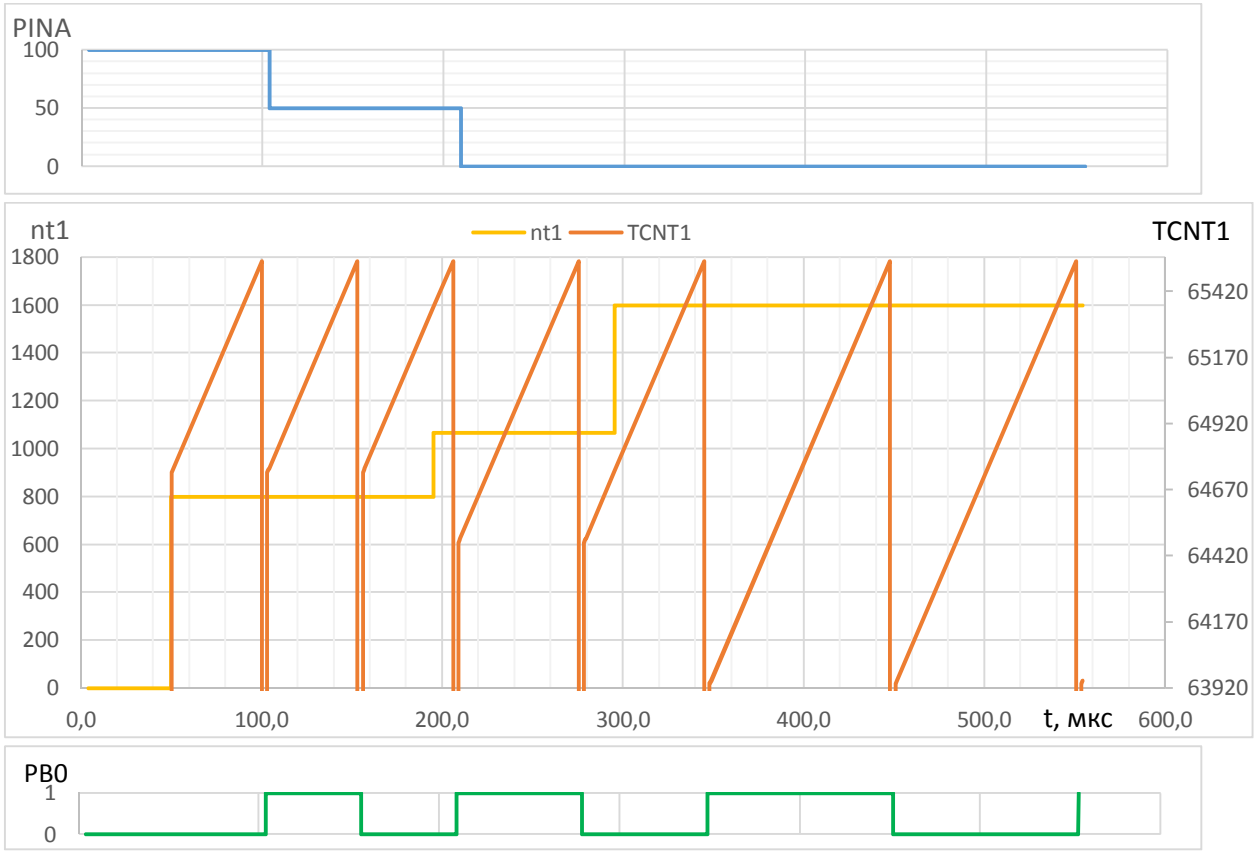


Рис. 2.5

На рис. 2.5 приведена временная диаграмма генерируемых импульсов PB0, числа тактов nt1, счетного регистра TCNT1, при вариации d (PINA) по трем значениям: 100, 50 и 0. Данные получены в симуляторе AVR-Studio, «вручную» перенесены в MS_Excel, построены графики.

Теперь измерьте период ввода и расчета значения длительности *nt1*, то есть период основного цикла в секундах и циклах.

Поставьте дополнительные точки прерывания на первую и последнюю строку подпрограммы прерывания ISR(TIMER… и измерьте задержки формирования фронта и среза (2.75 и 2.69 мкс – основная часть корректируемой погрешности), полное время обработки прерывания (3.44 и 3.56 мкс) – это «добавка» к времени реакции алгоритма.

3. Индивидуальное задание

3.1. Типовое задание

Генерации на выходе/ах МК импульсов заданной формы и длительности, один из параметров варьируется числом d (от 0 до 100), которое вводится через порт МК. Закон вариации, параметры и ссылка на рисунок приведены в Табл. 2.1.

Табл. 2.2 Варианты второго индивидуального задания

№	Параметры импульсов, d = 0, 1, … 100	№	Параметры импульсов, d = 0, 1, … 100
0	f = 5000 + 50*d Гц, $t_и = t_п$, рис. а	13	$t_и$/ T = 0.25 + 0.005*d, f = 20 кГц, рис. в
1	f = 1 + d Гц, $t_и = t_п$, рис. а	14	tз = 0 + 100*d мкс, f = 50 Гц, tи = 10 мкс, г
2	T = 100 + d мкс, tи = 20 мкс, рис. б	15	tи = 1 + d мкс, f = 5 кГц, рис. в
3	tс = 0 + 0.25*d мкс, T = 1000 мкс, рис. д	16	$t_з$ = 0 + 2.5*d мкс, f = 1 кГц, рис. г
4	tи = 10 + d мкс, T = 1000 мкс, рис. в	17	f = 400 + d Гц, tп = 5 мкс, рис. е
5	tз = 1 + 0.5*d мкс, f = 10 кГц, рис. г	18	T = 1000 + 10*d мкс, tи = 1 мкс, рис. б
6	f = 100 + d Гц, tи = 1 мс, рис. б	19	tс = 1 + d мкс, T = 400 мкс, рис. д
7	tи = 5*d мкс, T = 2000 мкс, рис. г	20	T = 50 + 0.5*d мкс, tи = 1 мкс, рис. б

8	f = 500 + 5*d Гц, tп = 5 мкс, рис. ж	21	tи = 1 + d мс, tп = 10 мс, рис. в
9	tи = 1 + d мс, T = 200 vc, рис. в	22	tи = 100 + 48*d мкс, f = 100 Гц, рис. е
10	tп = 200 + 10*d мкс, tи = 100 мкс, рис. б	23	f = 10000 + 100*d Гц, tи = 1 мкс, рис. е
11	f = 5000 + 10*d Гц, tп = 3мкс, рис. ж	24	tп = 200 + 12*d мкс, tи = 100 мкс, рис. б
12	tи = 5 + 0.4*d мкс, T = 100 мкс, рис. е	25	f = 1000 + 400*d Гц, tп = 1 мкс, рис. ж

3.1. Рекомендации по разработке и тестированию

Состав глобальных переменных для передачи числа тактов должен соответствовать количеству интервалов переполнения на периоде, разрядность должна совпадать с разрядностью таймера/счётчика.

Число интервалов также определяет структуру обработчика прерывания (цепочки if-else или switch) и число состояний соответствующей переменной.

Для формирования коротких постоянных интервалов (до сотни машинных циклов) удобно использовать программную задержку (командами asm(“nop”) длительностью один машинный цикл) без выхода из прерывания, в этом случае переполнение должно быть только в начале короткого интервала.

Настройки выбранного таймера отличаются цифрой в именах регистров и вектора прерывания (0 или 1), выбором коэффициента деления.

3.2. Требования к содержанию отчета

- Задание в полной и краткой форме.
- Временная диаграмма формируемых импульсов, счетчика TCNTx, прерываний, последовательности действий.
- Блок-схема программы
- Расчет параметров алгоритма: частота тактирования ftc, выражения и константы длительностей переполнения nti(d).
- Текст программы с комментариями.
- Тестирование и коррекция (если требуется и возможна). Таблица расчётных и измеренных параметров алгоритма генерации.
- Результаты измерений по оценке времени реакции алгоритма: инициализация, основной цикл, прерывания – задержки формирования события и фоновой задачи.
- Статистика использования памяти программ и памяти данных.

Лабораторная работа 3 [1 p. 6]
Аналоговый ввод в терморегуляторе

Цели работы:

1) Знакомство с алгоритмом работы многоканального терморегулятора [1 p. 6.5]
2) Изучение работы встроенного аналого-цифрового преобразователя [1 p. 6.3]
3) Знакомство с контактными датчиками температуры и схемами их подключения ко входам АЦП [1 p. 6.5]
4) Знакомство с системой моделирования Proteus VSM [1 p. 2.4].

1. Основные понятия

1.1. Терморегулятор [1 p. 6.5]

Устройство для измерения и поддержания на заданном уровне температуры нагреваемого или охлаждаемого объекта, может иметь от одного до нескольких каналов измерения и регулирования, общий пульт управления – цифровой индикатор и кнопки (рис. 3.1). Регулятор (нагреватель или охладитель) воздействует на объект управления через исполнительный орган ключевого или пропорционального типа. Дискретный регулятор (в данной работе) может только включать и выключать объект, например, коммутировать питание электронагревателя или компрессора охладителя. Аналоговый регулятор (в следующей работе) может плавно регулировать воздействие на объект, например, увеличивать или уменьшать мощность нагрева.

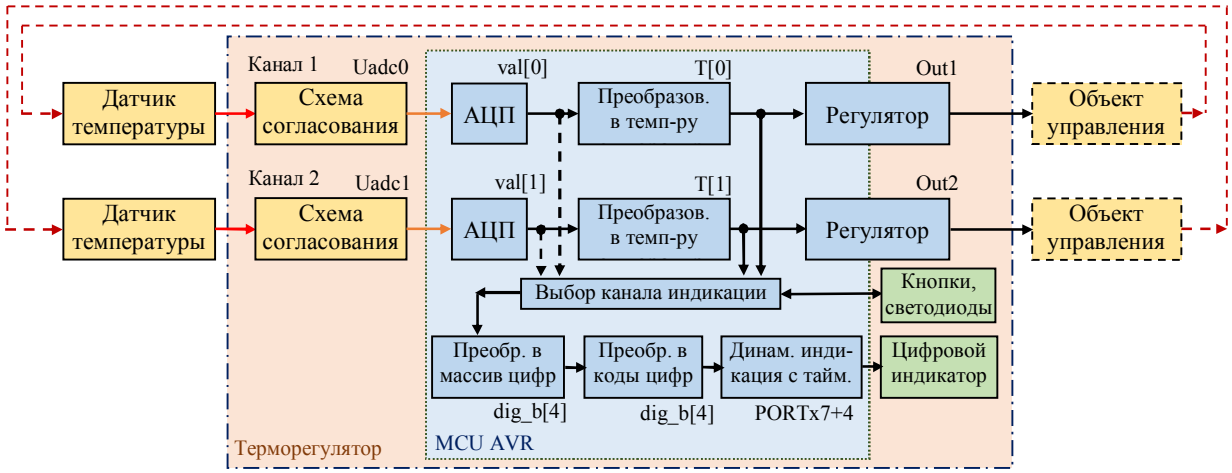


Рис. 3.1 Функциональная схема терморегулятора

Канал измерения состоит из датчика температуры, схемы согласования, аналого-цифрового преобразователя (АЦП), преобразователя кода АЦП в температуру. Обеспечивает требуемую точность измерения.

Датчик температуры располагается непосредственно на объекте, напряжение (ЭДС) или сопротивление датчика пропорционально температуре. Схема согласования обеспечивает заданный диапазон напряжения на входе АЦП.

Встроенный в МК многоканальный АЦП с выбором параметров настройки и канала измерения. Это аппаратный узел МК с программным управлением.

Преобразование кода АЦП *val[x]* в код температуры *T[x]* с учетом нормальной статической характеристики (НСХ) датчика, схемы согласования и требований точности. Программа МК, выполняющая линейное или кусочно-линейное преобразование.

Регулятор ключевого типа поддерживает заданную температуру с использованием дискретного выхода. Регулятор сравнивает измеренное значение *nT* с уставкой температуры *Set* (с учетом величины гистерезиса *His*) и переводит выход в состояние 1 или 0. Для каждого канала свой регулятор, выход «OUT1» со светодиодом и пр. Программа, аналогичная регулятору из лабораторной работы 1.

Пульт управления, состоящий из цифрового индикатора на 7-сегментных светодиодах, отдельных светодиодов и кнопки. Цифровой индикатор отображает измеренные значения температуры. Кнопка позволяет выбрать канал измерения для индикации, выбор канала индицируется светодиодами «CHx». Устройство и алгоритмы программного обслуживания цифрового индикатора и кнопок см. в УП п.2.6 и в Примерах в папке p1-2 Human Machine Interface.

1.2. Аналогово цифровой преобразователь (АЦП) в МК AVR [УП р. 6.3]

Предназначен для преобразования напряжения на входе/ах ADCx в число n, закон преобразования n = UadcX / Uref * 2^N, N = 10 – разрядность, Uref = 2…Vcc – опорное напряжение.

В данном примере программно выбран Uref = AVCC = 5 В, фильтруемое конденсатором на ноге AREF, здесь AVCC – вход питания аналоговых узлов МК. Измеряемое напряжение подается на один из 8 входов ADC0…ADC7, выбор канала преобразования программный. Результат преобразования выбранного канала (число от 0 до 1023) помещается в регистр данных ADC_{16}. Одно преобразование требует 13 или 14 тактов, частота тактирования АЦП fadc должна находиться в диапазоне от 50 до 200 кГц, получаем длительность преобразования от 65 до 280 мкс. Запуск преобразования программный или аппаратный (по одному из доступных прерываний) в однократном или циклическом режимах. Настройка и управление через регистры ADCSR(A) и ADMUX.

В данной работе удобно использовать однократный режим запуска АЦП с предварительным выбором канала. Подробнее см. УП р.6.3.

1.3. Датчики температуры (контактного типа) и их согласование с АЦП:

а) термоэлектрический преобразователь (ТЭП) является источником термоЭДС весьма малой амплитуды (десятки мВ), почти линейно растущей с температурой; конструктивно состоит из пары проволок разных сплавов металлов, спаянных в одной точке, поэтому его называют «термопара» [thermocouple], средняя точность и стабильность, большой диапазон измерений (-270 ... +1330 °C);

б) термосопротивление (ТС) – резистивный датчик из металлической проволоки (платина или медь) с положительным температурным коэффициентом сопротивления (ТКС), в обозначении указывается значение сопротивления при нормальной температуре (например, [RTD-Pt] 100 Ом, от -200 до +850 °C), высокая точность и стабильность (прецизионные), значительные габариты и стоимость;

в) полупроводниковые датчики температуры имеют боле узкий диапазон (от -55 до +155 °C, отдельные до +300 °C), собственно термочувствительный элемент обычно имеет нелинейную статическую характеристику, но часто снабжен встроенной схемой линеаризации; часть выпускаются как двухполюсники с переменным сопротивлением, другие – как трехполюсники с выходом по напряжению, цена обычно ниже других типов.

Датчик характеризуется диапазоном измерения температуры [*Tmin*, *Tmax*] и диапазоном соответствующего изменения выходного сигнала: напряжения [*Umin*, U*max*] для термопары и сопротивления [Rmin, Rmax] для термосопротивления.

Схема подключения датчика должна обеспечить на входе АЦП диапазон напряжений максимально близкий к диапазону АЦП, то есть источника опорного напряжения (от 0 до 2…5 В). Для упрощения задачи будем во всех случаях использовать Uref = VCC = 5 В, поэтому напряжение датчика должно входить в диапазон от 0 до 5 В.

Термопара требует предварительного усиления сигнала (*предусиление*), при измерении отрицательных температур дополнительно требуется смещение или инверсия.

Резистивные датчики требуют преобразования изменения сопротивления в изменение напряжения. Самая простая схема представляет собой резистивный делитель напряжения (рис. 3.2, б), состоит из термосопротивления RT, дополнительного резистора *R* и источника напряжения *VCC = 5 B*, средняя точка делителя напряжения Us подключается к синфазному входу АЦП, например, к *ADC0*. Использование единого опорного напряжения для датчика и АЦП снижает погрешности дрейфа и разброса напряжения питания. Номинал резистора R должен обеспечить максимальный диапазон напряжений, обычно он равен номинальному сопротивлению термосопротивления.

1.4. Система моделирования Proteus VSM [УП р. 2.4]

Пакет аналого-цифрового моделирования *Proteus VSM* из среды сквозного проектирования электронных устройств *Proteus* (программа *ISIS)* фирмы *Labcenter Electronics*.

Модель МК AVR кроме загрузки и исполнения программы, разработанной в AVR_Studio имеет полнофункциональные модели периферии, в том числе встроенного АЦП. Наличие интерактивных моделей контактных датчиков температуры с кнопками «задания» температуры (рис. 3.2) позволяет имитировать работу реальных датчиков. Анимационная модель цифрового семисегментного индикатора позволяет вести отладку и тестирование в интерактивном режиме (имитация работы «железа» и программы в реальном времени).

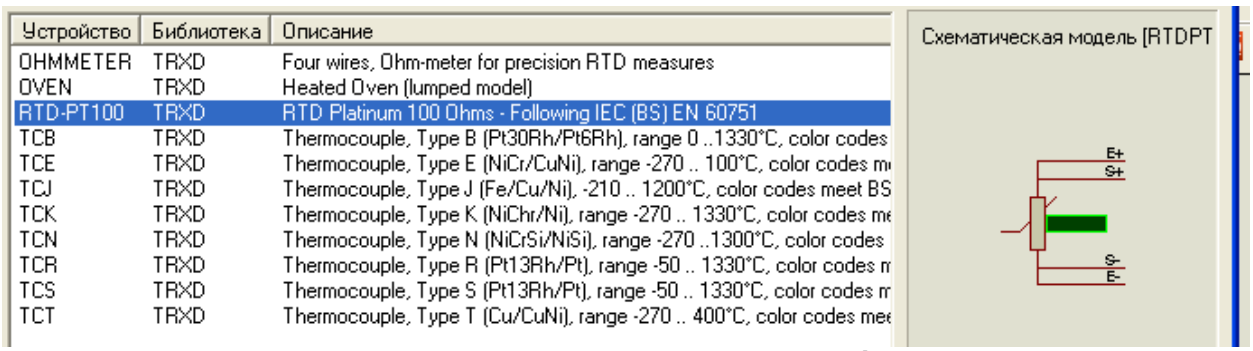
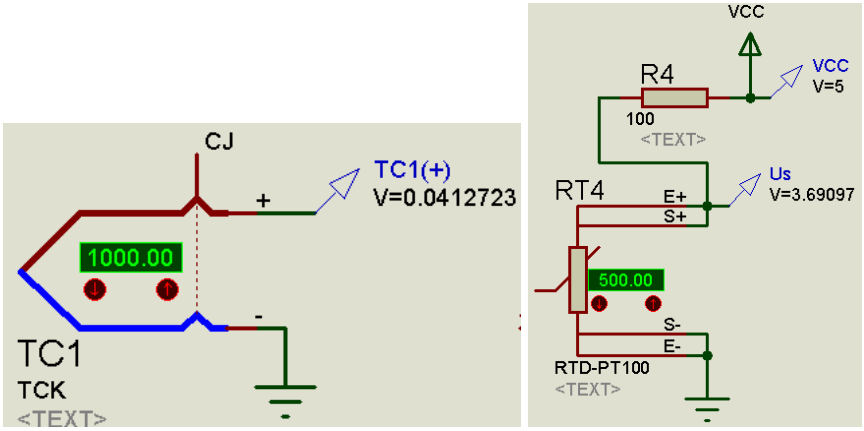


Рис. 3.1. Окно выбора модели датчика температуры *Transducers* → *Temperature (Proteus VSM)*

Справочная информация и примеры тестирования моделей датчиков находятся в папке TermoSensors. На рис. 3.1 в библиотеке моделей датчиков температур виден один платиновый терморезистор RTD-PT100 и несколько термопар (TCK, TCJ и др.).



а б

Рис. 3.2

В папке Primer\Lab3\TermoSensors в файле TermoSensorModels.DSN найдёте выборку всех используемых ниже датчиков температуры.

2. Пример 3: «Дискретный терморегулятор» 0-1 и 0-2

2.1. Задание

Разработать схему и программу терморегулятора с цифровым индикатором температуры на N каналов с релейным выходом для тестирования в системе моделирования Proteus VSM. Тип датчика, диапазон и точность измерения, число каналов измерения/регулирования, параметры регулятора – тип, уставка и гистерезис заданы в табл. 3.1. Первый пример – двухканальный терморегулятор с термопарами K типа, второй – одноканальный с платиновым терморезистором номинальным сопротивлением 100 Ом.

Табл. 3.1. Параметры задания к примерам

№	Датчик	Число кан.	Измерения		Регулирование		
			Диапазон, ºС	Точность, %	Закон	Уставка, ºС	Гист., ºС
0-1	ТЭП К	2	0…999	1	Нагреватель	800	3
0-2	ТС Pt 100 Ом	1	0…500	0.5	Охладитель	400	10

2.2. Схема и модель Proteus VSM

На рис. 3.1 приведен скриншот окна схемного редактора ISIS Proteus VSM в процессе моделирования примера 0-1 (папка TermoRegTC, файл TR-2TC-m16.dsn). Электрические связи («провода») представлены зелеными линиями, стрелками выходящими и входящими с именами цепей (одно имя – одна цепь). Перечень элементов схемы:

C (100 нФ) – конденсатор 100 нФ для фильтрации опорного напряжения,

D (ATMEGA16) – выбранный МК (22 + 5 pins),

E1, E2 (VCVS 121) – идеальные усилители (см. ниже),

HG (7SEG-MPX4-CA-BLUE) – анимационная модель 7-сегментного цифрового индикатора на 4 цифры с Общим Анодом,

SW (SW-SPST-MOM) – интерактивная модель кнопки,

TC1, TC2 (TCK) – интерактивная модель термопары K (XA) типа,

VD1, VD3 (LED-GREEN), VD2, VD4 (LED-BLUE) – анимационные модели светодиодов, зелёные – выбор канала индикации, синие – имитаторы выходов регуляторов.

Синие косые стрелки с надписями ADC0, ADC1, PB7 – указатели узлов схемы для наблюдения напряжений [Voltage Probe], рядом на рисунке видны численные значения в Вольтах. Цветные квадратики возле выводов компонентов обозначают уровень цифрового сигнала: красный – «Высокий», синий – «Низкий», серый – неопределённый (высокоомные входы).

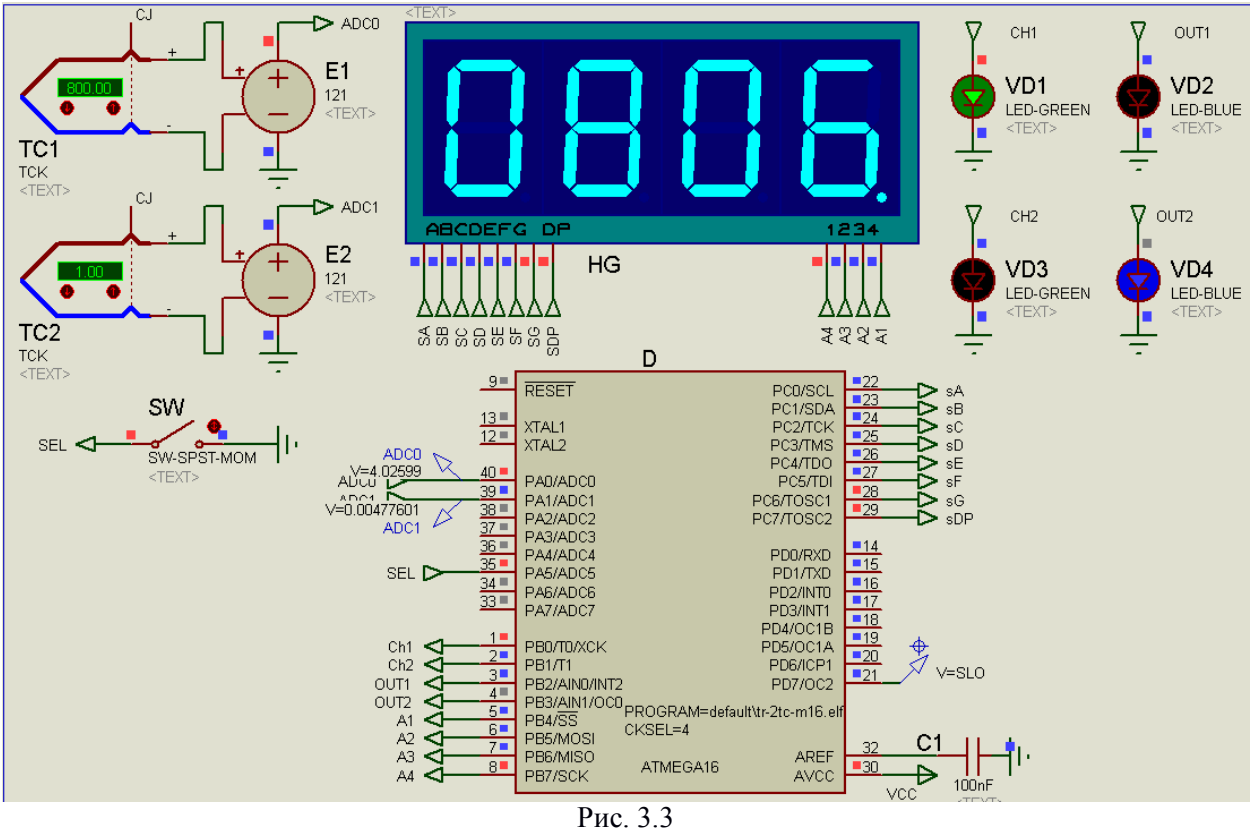


Рис. 3.3

Вверху слева расположена аппаратная часть двух каналов измерения: модели термопар TC1, TC2 и усилителей согласования E1, E2. Выходы усилителей обозначены ADC0, ADC1 и подключены ко входам АЦП (PA0/ADC0, PA1/ADC1), встроенного в МК D.

Модель МК настроена на частоту тактирования 8 МГц от встроенного источника тактирования, программа разработана в среде AVR-Studio и загружена в виде файла прошивки <*.hex> или прошивки с отладочной информацией <*.elf>. Вход питания аналоговых узлов AVCC подключен к источнику VCC (+5 В).

Цифровой индикатор HG программно управляется МК, требует использования алгоритма динамической индикации. Выводы 1…4 – это общие аноды каждой из «цифр», выводы A…G, DP – катоды одноименных сегментов (7 сегментов цифр и 1 сегмент – десятичная точка). Момент на рисунке соответствует формированию «изображения» лидирующего (левого) нуля (высокий уровень на аноде 1 и низкие уровни на сегментах A…F), остальные три «цифры» погашены. Программа обеспечивает частоту смены изображения 100 Гц, человеческий глаз не замечает этого мигания. Алгоритм динамической индикации использует таймер с переполнением на частоте 400 Гц.

В программе примера 0-1 цифровой индикатор отображает численное значение температуры первого или второго канала в градусах. Выбор канала индицируется зелеными светодиодами CHx, смена выбора – нажатием кнопки SEL.

Синие светодиоды OUT1, OUT2 зажигаются, когда выход регулятора соответствующего канала находится в высоком состоянии.

На рис. 3.3 «горит» светодиод CH1, цифровой индикатор отображает температуру 806 °C близкую к значению 800 °C, установленному на индикаторе термопары TC1. Светодиод регулятора первого канала OUT1 «не горит» - 806 °C больше 800 + 3 °C, светодиод второго канала OUT2 «горит» - 1 °C меньше 800 – 3 °C.

Схема примера 0-2 находится в папке TermoRegRTD, файл TR-RTD-m16.DSN. Здесь один канал измерения (схема попроще, нет кнопки выбора канала) и резистивный датчик температуры.

Термопара. Для согласования использован идеальный усилитель на базе источника напряжения, управляемого напряжением E1 [Modelling Primitives → Analog[SPICE] → VCVS – Voltage Controlled Voltage Source], его единственный параметр – коэффициент усиления называется в свойствах модели Voltage gain. Для выбора коэффициента усиления следует узнать или измерить в модели значения напряжений в начале и в конце диапазона, затем разделить напряжение опоры (например, VCC = 5 В) на максимальное напряжение датчика. Для термопары хромель-алюмель К типа (ТСК) при 1000 ºС напряжение 0.04127 В, то есть ku = Uref / $U_{ТСК}$(Tmax) = 5 / 0.04127 ≈ 121 (рис.3.2, а).

При смене знака измеряемой температуры меняется знак напряжения, в этих случаях выход усилителя требуется смещать источником напряжения Vx [VSOURCE].

Термосопротивление. Номинал резистора делителя R (рис. 3.2) подбирается по результатам измерения в симуляторе напряжения средней точки делителя Us (Табл. 3.2). Целью подбора является максимальное значение разности Us(Tmax) – Us(Tmin). При RTnom = 100 Ом лучше использовать R = 100 Ом.

Табл. 3.2. Выбор дополнительного резистора

	Us, В				
T, °C	R=50	R=100	R=300	R=500	R=1000
0	3.344	2.512	1.259	0.84	0.459
500	4.247	3.691	2.422	1.803	1.1
ΔUs, В	0.903	1.179	1.163	0.963	0.641

Полупроводниковые датчики температуры в большинстве имеют выход по напряжению, если диапазон выходит за границы 0…+5 В, требуется использовать схему смещения.

2.3. Алгоритм

На этапе инициализации настраиваются переменные, порты ввода/вывода (PB0…PB7, PC0…PC7, PD7 – выходы, PA5 – вход), таймер 0 для тактирования цифрового индикатора (8 МГц / 400 Гц = 20000, 20000 / 256 = 78.125, делитель таймера k = 256, период переполнения 78 тактов), встроенный АЦП (выбор тактирования 8 МГц / 64 = 125 кГц, опоры AVCC+C(AREF) и однократного режима работы).

Основной цикл двухканального варианта:

1) запуск преобразования первого канала АЦП и запись результата в переменную val[0],
2) запуск преобразования второго канала АЦП и запись результата в переменную val[1],

3) преобразование значения переменной val[0] (кода АЦП) в значение температуры T[0] в соответствии с требованиями точности,
4) преобразование значения переменной val[1] в значение температуры T[1],
5) выявление нажатия кнопки SEL для смены выбора канала (channel = 0 или 1),
6) преобразование значения Температуры выбранного канала T[channel] в коды для цифрового индикатора с учётом требуемой точности (целые и или десятые доли градуса) и знака (только положительные, только отрицательные или со сменой знака); коды для цифрового индикатора размещаются в четырех однобайтовых ячейках dig_b[4],
7) управление выходом ключевого Регулятора 1 (OUT1 ≡ PB2) по Температуре 1 (и заданным значениям уставки Set и гистерезиса hys),
8) управление выходом ключевого Регулятора 2 (OUT2 ≡ PB3) по Температуре 2,
9) задача обслуживания алгоритма динамической индикации цифрового индикатора синхронизируется с переполнением таймера сканированием флага переполнения таймера.

По установке флага выполняется его сброс, запись в счетный регистр TCNT0 числа -78, далее выполняются действия по выключению одной из четырех «цифр» и включению следующей. Статическая переменная dig_n ведёт циклический счет цифр от 0 до 3. Выключение цифры выполняется низким уровнем на выходе Ai (PB4…PB7), включение – высоким уровнем на выходе Ai+1 и выводом на порт C (PC0…7) кода цифры из переменной dig_b[i] с инверсией (управление катодами).

Для одноканального терморегулятора отпадает необходимость в дублировании действий измерения, регулирования и поддержки механизма выбора канала индикации.

Для увеличения числа каналов достаточно повторить требуемое число раз операции измерения и регулирования.

Задача преобразования заданного канала АЦП требует записи номера канала в младшие биты регистра ADMUX, запуска преобразования установкой бита ADSC регистра ADCSRA. По завершении преобразования АЦП устанавливает флаг запроса прерывания ADIF в регистре ADCSRA. Программа должна сбросить этот бит «единицей» и скопировать содержимое регистра результата ADC в двухбайтную переменную val[x].

Задача преобразования кода АЦП в Температуру решается линейным преобразованием либо кусочно-линейной аппроксимацией. Выбор определяется характером зависимости напряжения на входе АЦП от температуры и заданной точностью.

Пользуясь моделью выбранного датчика удобно построить зависимость напряжения или сопротивления датчика от температуры, затем рассчитать или измерить коды АЦП и построить график, например, в программе MS Excel.

Для согласования термопары (Табл. 3.3) столбец значений ЭДС <E> получен в с выхода модели TCK, столбец напряжений на входе АЦП Us по формуле E * 121, столбец значений АЦП adc в получен по формуле ЦЕЛОЕ(Us * 1024 / 5), столбец «индицируемых» значений температуры <Tp> по формуле линейного масштабирования ЦЕЛОЕ(adc * 978 / 1000). Закон линейного преобразования значений АЦП в температуру получен делением максимальной температуры на соответствующее значение АЦП 1000 / 1022 = 0,978, смещение в данном случае отсутствует.

Табл. 3.3. Термопара ТСК (ХА)

T, °C	E, V	Us, В	adc	Tp, °C
0	0,000000	0	0	0
100	0,004096	0,496	101	98
200	0,008138	0,985	201	196
300	0,012210	1,477	302	295
400	0,016396	1,984	406	397
500	0,020640	2,497	511	499
600	0,024900	3,013	617	603
700	0,029126	3,524	721	705
800	0,033270	4,026	824	805
900	0,037320	4,516	924	903
1000	0,041270	4,994	1022	999

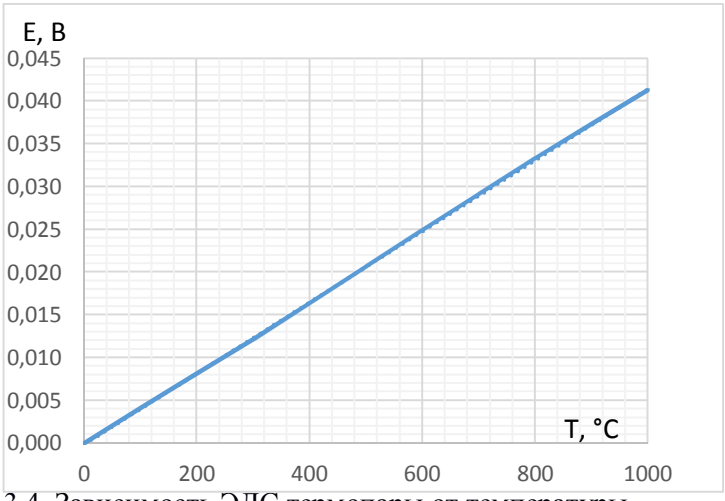


Рис. 3.4. Зависимость ЭДС термопары от температуры

Видно, что ожидаемая погрешность не превышает 5 °C, что составляет 0.5 % от диапазона 0…1000 °C, то есть удовлетворяет заданию.

Сопротивление платинового терморезистора почти линейно зависит от температуры (Табл. 3.4, рис. 3.5), напряжение Us(T) изменяется по нелинейному закону Us(T) = VCC * RT(T) / (R + RT(T)), пропорционально меняется код АЦП. В большинстве случаев разбиение нелинейной характеристики на 10 участков с постоянным шагом по температуре достаточно для решения задачи линейной аппроксимации с заданной точностью.

Табл. 3.4 Терморезистор RTD-Pt 100 Ом

T, °C	R, Ом	Us, В	adc
0	100,00	2,512	514
50	119,39	2,731	559
100	138,50	2,912	596
150	157,32	3,064	627
200	175,85	3,194	654
250	194,09	3,306	677
300	212,04	3,403	696
350	229,70	3,488	714
400	247,08	3,563	729
450	264,16	3,631	743
500	280,96	3,691	755

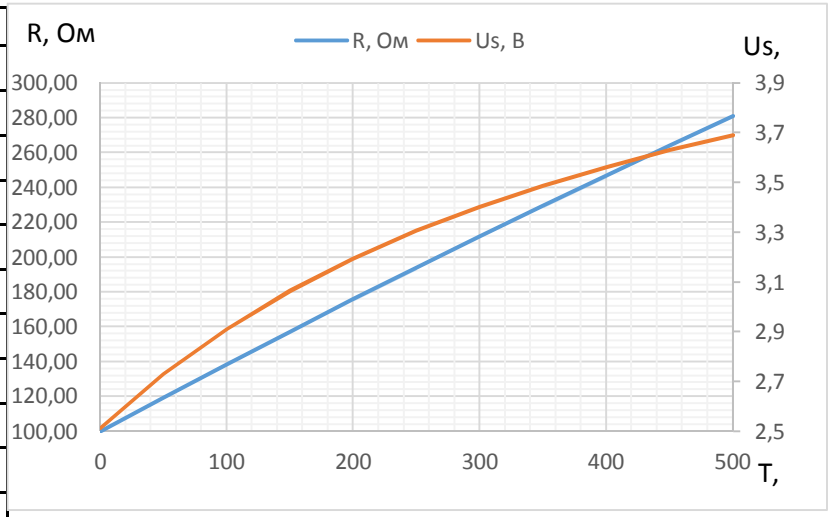


Рис. 3.5. Зависимости сопротивления и напряжения от температуры

Общая формула линейной зависимости y = kx + y0, по координатам двух точек (x1, y1), (x2, y2):

y1 = kx1 + y0, y2 = kx2 + y0

y1 – y2 = k (x1 – x2), k = (y1 – y2) / (x1 – x2),

y0 = y1 – k x1 = y1 – x1 (y1 – y2) / (x1 – x2)

Результат: y = (x – x1) (y1 – y2) / (x1 – x2) + y1

Здесь y – вычисляемая температура, x – текущее значение АЦП, y[i] , x[i] – константы из первого и четвёртого столбцов Табл. 3.4. По заданию требуется точность 0.5 % от 500 °C, то есть 2.5 °C. Выбираем индикацию до десятых долей градуса в формате с

фиксированным положением десятичной точки <xxx.x>. Для этого значения констант температуры из первой колонки Табл. 3.4 следует умножить на 10.

Программа должна по коду АЦП «выбрать» ближайшие значения АЦП из таблицы и по соответствующим значениям температуры рассчитать линейную аппроксимацию температуры.

Замечание. Альтернативным способом получения значений АЦП является индикация этих значений в процессе отладки программы в Протеусе.

2.4. Проект и текст программ

Пример с термопарой находится в папке TermoRegTC, схема модели Proteus VSM с двумя каналами измерения/управления в файле TR-2TC-m16.dsn, проект программы AVR-Studio в TR-2TC-m16.aps, текст программы на языке С в TR-2TC-m16.c.

Пример с термосопротивлением RTD-PT100 находится в папке TermoRegRTD, схема с одним каналом измерения/управления в файле TR-RTD-m16.dsn, проект программы в TR-RTD-m16.aps, текст программы на языке С в TR-RTD-m16.c.

Тексты программ на языке С соответствуют вышеописанному алгоритму и снабжены достаточным количеством комментариев.

2.5. Тестирование примеров 0-1 и 0-2

В симуляторе AVR-Studio моделируется только цифровая часть АЦП (тактирование, процедура преобразования), аналоговых сигналы и цепи не моделируется. Поэтому отладку программы удобно вести в AVR-Studio, тестирование работы схемы с аналоговыми сигналами возможно только в Proteus VSM,

Тестирование в системе моделирования Proteus VSM выполняется либо в интерактивном режиме, либо в режиме расчёта временной диаграммы.

Модель терморегулятора удобно тестировать в интерактивном режиме:

1) задачу измерения – заданием «температуры» на цифровом индикаторе датчика (TCx, RTx, …) и индикацией на 7-сегментном цифровом индикаторе HG;

2) регулятор – наблюдением переключения выхода при вариации температуры датчика в области уставки с учетом гистерезиса.

Запуск сеанса интерактивного моделирования выполняется кнопкой Play, останов – кнопкой Stop, кнопки Pause и Step позволяют войти в режим символьного отладчика, похожего на отладчик AVR-Studio, но с меньшим уровнем удобств.

Модели датчиков температуры имеют настраиваемые параметры (доступные во всплывающем меню Edit Properties в режиме Stop): заданная температура [Actual Temperature] и шаг изменения температуры [Temperature Step]. Изменение температуры выполняется кнопками «+» и «-» датчика от заданной температуры с выбранным шагом.

Проверка точности каналов измерения выполняется на краях и в середине диапазона, для термосопротивления – в середине каждого участка аппроксимации, шаг должен соответствовать дискретности.

Аналогичный шаг требуется при проверке регулятора.

Для оценки динамики интерактивный режим не всегда удобен. Период основного цикла удобно наблюдать по временной диаграмме выхода (PD7), переключаемого в конце каждого периода.

Временные диаграммы строятся для напряжений узлов или токов ветвей схемы, помеченных специальными указателями (щупами) с косой стрелкой и именем. Окно временной диаграммы может быть цифровым [Digital analysis], аналоговым [Analog analysis], смешанным [Mixed analysis] и пр. Состав графиков [Add Traces…], длительность расчета [Edit Graph…], запуск расчета [Simulate Graph…] задаются во всплывающем контекстном меню. Для подробного анализа полученного графика следует выбрать в контекстном меню [Maximize (Show Window)], появляются возможности изменения масштаба времени, два курсора для измерений и пр.

3. Индивидуальное задание

3.1. Полный текст задания приведен в п. 2.1, в Табл. 3.5. приведены параметры вариантов заданий.

Индивидуальные задания для закрепления понимания данного примера требуют:

1) анализа характеристики датчика,
2) разработки схемы подключения датчика,
3) выбора способа и расчета параметров программного масштабирования кодов АЦП в температуру,
4) коррекции алгоритма индикации для отрицательных значений температур,
5) корректировки числа каналов измерения/регулирования в схеме и программе.

Табл. 3.5. Параметры третьего индивидуального задания

№	Датчик	Число кан.	Измерения		Регулирование		
			Диапазон, ºС	Точность, %	Закон	Уставка, ºС	Гист., ºС
0-1	ТЭП K	2	0…999	1	Нагреватель	800	3
0-2	ТС Pt 100 Ом	1	0…500	0.5	Охладитель	400	10
1	ТЭП B	2	500…1300	1	Нагреватель	950	20
2	ТЭП E	2	-150…0	1	Охладитель	-100	30
3	ТС Pt 100 Ом	1	0…500	0.5	Нагреватель	250	4
4	ТЭП J	2	-50…450	1	Охладитель	50	5
5	ТЭП N	2	100…600	1	Нагреватель	400	50
6	ТС Pt 100 Ом	2	250…850	1	Охладитель	700	2
7	ТЭП R	2	300…800	1	Нагреватель	500	25
8	ТС Pt 100 Ом	2	-100…+200	1	Охладитель	100	20
9	ТЭП S	2	600…1100	1	Нагреватель	850	40
10	ТС Pt 100 Ом	2	-100…+400	1	Охладитель	-20	3
11	ТЭП T	1	-100…+100	1	Нагреватель	0	20
12	LM20	2	-50…+100	2	Охладитель	50	10
13	LM35	1	-50…+150	2	Нагреватель	100	20
14	MCP9701	2	-40...+125	2	Охладитель	0	5
15	PTC_NICKEL	2	-55...+300	2	Нагреватель	200	20
16	KTY81	2	-55...+300	2	Охладитель	-10	10
17	ТЭП K	1	-100…+100	1	Охладитель	0	10
18	ТЭП K	3	-100…+900	1	Нагреватель	750	10
19	ТЭП K	4	300…1300	1	Нагреватель	800	10
20	KTY81	3	-45...+80	2	Охладитель	25	2

3.2. Анализ характеристики датчика

Выполняется по модели датчика в Proteus VSM в файле TermoSensorModels.DSN. Строится зависимость напряжения для термопар, сопротивления для резистивных датчиков (интерактивным омметром [OHMMETR]) от температуры в заданном диапазоне, разбитом на 10 интервалов. Удобно оформить как таблицу в MS Excel и построить график для оценки линейности.

3.3. Согласование датчика со входом АЦП

Выбор типовой схемы, расчет элементов схемы. Напряжение на входе АЦП должно строго входить в диапазон 0…Uref, по возможности максимально заполняя его. Сразу проверка в Proteus VSM.

Для датчиков с выходом напряжения требуются усиление и/или смещение [], для резистивных датчиков – выбор схемы и параметров резистивного делителя.

3.4. Преобразование кода АЦП в температуру

Первое действие – оценка линейности кодов АЦП для выбора вида преобразования (lin_scale() или calc_lin_appr_T()), удобно выполнить сразу в модели Протеуса. Для этого в среде AVR-Studio корректируем текст основного цикла: в строке prn7_4_ud4(T[channel],0); заменить T[channel] на val[0], компилируем. Для 5 – 10 значений температуры измеряются значения кодов АЦП, строится график для оценки погрешности линейной аппроксимации (по крайним точкам, или по начальной и средней и пр.).

Если линейная аппроксимация не обеспечивает требуемой точности, используем полученные точки для кусочно-линейной аппроксимации.

Возвращаем индикацию (текст программы основного цикла) в исходное состояние.

3.5. Коррекция алгоритма индикации

Для отображения отрицательных температур требуется переделка функции void prn7_4_ud4(unsigned int udval, char dp_pozition), которая может отображать только беззнаковые целые до четырёх десятичных цифр в виде целого или числа с десятичной точкой (позиция задаётся dp_pozition)). Идея состоит в том, чтобы выделить попадание в отрицательный диапазон, вывести изображения «минуса» (код 0x40), преобразовать число в положительное и вывести только три(!) младшие цифры.

Проверяем погрешность оцифровки в Proteus VSM, особенно в середине участков аппроксимации, записываем наибольшее отклонение.

3.6. Дискретный регулятор

Разработка аналогично Лабе 1, тестирование в Proteus VSM вариацией температуры датчика с минимальным шагом индикации в окрестностях точек переключения по значениям цифрового индикатора (а не датчика!).

3.7. Измерение времени реакции

Длительность основного цикла и инициализации по временной диаграмме выхода PD7, времени выполнения функции чтения АЦП в симуляторе AVR-Studio.

3.8. Содержание отчета

- задание,
- характеристика датчика, выбор и расчет схемы согласования датчика со входом АЦП,
- схема электрическая терморегулятора с перечнем элементов,
- текстовое описание настройки АЦП,
- алгоритм преобразования кода АЦП в значения температуры заданной точности,
- текст программы: переменные, main с комментариями, работа с АЦП и пр.,
- результаты тестирования по оценке точности измерения (таблица значений температур датчика, напряжений датчика, кодов АЦП, измеренных температур и оценки погрешности), проверке регулятора (с гистерезисом),
- оценка времени реакции регулятора и расхода времени на АЦ-преобразование,
- расход памяти ПП и ПД, выводов МК.

Лабораторная работа 4 [1, p. 7]
Последовательный интерфейс в пропорциональном терморегуляторе

Цели работы:

1) Изучение алгоритма работы пропорционального регулятора температуры
2) Изучение протоколов и принципов работы встроенных контроллеров синхронных последовательных интерфейсов (ПИ) SPI, I2C [1, p.7.3, 7.4]
3) Изучение алгоритмов вывода аналоговых сигналов через ЦАП с ПИ [1, p.7.5]

1. Основные понятия

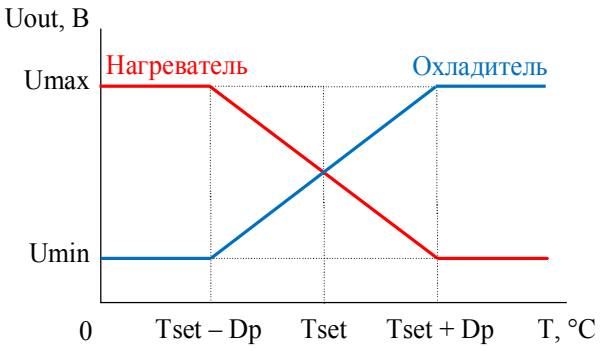
1.1. Пропорциональный регулятор температуры

Данная работа развивает пример из предыдущей работы, посвященный изучению терморегулятора с использованием моделей Proteus VSM. Теперь вместо регулятора ключевого типа мы рассмотрим пропорциональный регулятор с аналоговым выходом. Регулятор вычисляет разность уставки температуры и измеренного значения ΔT = Tset – T, умножает эту разность на значение коэффициента пропорциональности ku, смещает и ограничивает вычисленное значение диапазоном Umin…Umax и выводит в виде аналогового сигнала напряжения Uout (рис. 4.1, Табл. 4.1).

Табл. 4.1

Диапазон температур	Нагреватель	Охладитель
T < Tset – Dp	Uout = Umax	Uout = Umin
(Tset – Dp) < T < (Tset + Dp)	Uout = Umid + ku * ΔT	Uout = Umid – ku * ΔT
T > Tset + Dp	Uout = Umin	Uout = Umax

где Dp – диапазон пропорциональности, ku = (Umin – Umax) / (Tset + Dp – Tset + Dp) = (Umin – Umax) / 2 Dp – коэффициент пропорционального усиления, Umid = Umin + (Umax – Umin) / 2 – среднее значение Uout.



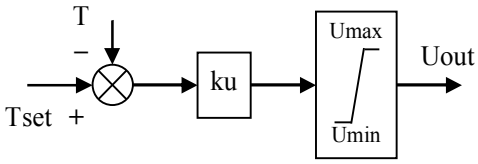


Рис. 4.1

Использование такого регулятора оправдано, когда объект управления имеет возможность плавного изменения мощности нагревателя или охладителя. В этом случае появляется возможность точнее и с меньшим «разбросом» стабилизировать температуру.

МК AVR не имеют встроенного цифро-аналогового преобразователя (ЦАП), поэтому используется внешний ЦАП, подключенный к МК по последовательному интерфейсу (ПИ) синхронного типа. Такие микросхемы выпускаются различными фирмами и позволяют экономить выводы МК.

1.2. Протоколы и контроллеры синхронных последовательных интерфейсов [7.3, 7.4]

В синхронных интерфейсах по одной из линий передаются такты от ведущего к ведомому/мым.

Для передачи данных достаточно записать байт в регистр данных передатчика. По завершении приёма полученный байт считывается из регистра данных приёмника. В контроллерах интерфейсов эти два разных регистра имеют один адрес.

Передача цепочки двух и более логически связанных байт называется *кадром*.

Трехпроводный интерфейс SPI является самым простым и скоростным из встраиваемых в МК. Три основных сигнала: такты SCK, данные от ведущего к ведомому MOSI, данные от ведомого к ведущему MISO, подключаются по схеме общей шины. За 8 тактов ведущий и ведомый обмениваются байтами. Четвёртый сигнал – выбор ведомого SS является индивидуальным, обеспечивает аппаратную адресацию и синхронизацию при передаче цепочки связанных байтов.

Встроенный контроллер SPI в МК AVR может работать как в роли ведущего, так и ведомого с частотой тактирования fsck не менее чем в 2 или 4 раза ниже системной fclk. При работе в роли ведущего выборка ведомого/ых выполняется программно через любой вывод/ы порта. Регистры: данных SPDR, управления SPCR, состояния SPSR. Настраивается порядок бит (прямой или обратный), фаза и полярность тактов (4 режима).

Обмен данными как в ведущем, так и в ведомом начинается с записи байта в SPDR, далее ожидание флага готовности (бит SPIF в регистре SPSR), завершается чтением принятого байта из регистра SPDR.

Двухпроводный интерфейс I2C имеет шину тактов SCL и данных SDA, обе с притяжкой к VCC (в состоянии покоя обе линии в Высоком состоянии); обеспечивает связь между ведущим и до 127 ведомыми на частоте 100 или 400 кГц, поддерживает смену ведущего. По сравнению с SPI менее производительный, более сложный в программном обслуживании, но более «сетевой» при меньшем количестве проводников.

Каждый кадр начинается с последовательности Start (срез SDA, затем срез SCL), затем следует передача адреса ведомого (7 бит), бита Чтение или Запись, прием бита ответа ведомого (Ack), итого 9 тактов. Далее передаются байты в заданном направлении (Запись – от ведущего к ведомому, Чтение – наоборот), после каждого байта бит ответа приемника (на 9-ом такте). Для завершения последовательности либо ведущий выдаёт последовательность Stop (фронт SCL, затем фронт SDA), либо приемник данных не передаёт бита ответа Ack.

Встроенный контроллер I2C в МК AVR называется TWI, выполняет функции как ведущего так и ведомого, fscl ≤ fclk / 16, имеет 5 регистров: выбора частоты (ведущего) f_{SCL} TWBR, управления TWCR, состояния TWSR, данных TWDR, адреса ведомого TWAR.

Последовательности Start и Stop запускаются установкой соответствующих бит регистра управления, передача байта – записью в регистр данных, прием байта – чтением регистра данных после его заполнения. Регистр состояния TWAR имеет пять старших битов, определяющих 32 состояния конечного автомата. Любое действие завершается установкой флага TWINT, есть запрос прерывания по этому событию, затем требуется анализ кода состояния. Сброс флага TWINT программный записью «единицы».

Выполнение элементарных действий протокола и констант выделено в файлы<i2c.c>, <i2c.h>, ниже приведен текст трех функций.

```
void i2c_start(void) {
  TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN); //Формирование START
  while (!(TWCR & (1<<TWINT))) ;//Ожидание флага TWINT
  if((TWSR & 0xF8) != START) error_i2c();//Проверка кода состояния START
}
void i2c_write(char byte) {
  TWDR = (byte);//Загрузка байта в регистр данных
  TWCR=(1<<TWEN)|(1<<TWINT);//Посылка байта
  while (!(TWCR & (1<<TWINT))) ;// Ожидание флага TWINT
  if((TWSR & 0xF8) != BYTE_W_ACK) error_i2c();//Проверка подтверждения приемника
}
void error_i2c(void) { PORTC |= (1<<6); }//Индикация ошибки интерфейса
```

1.3. Микросхемы ЦАП с последовательным синхронным интерфейсом [1, p.7.5]

1.3.1. Модели ЦАП в Proteus VSM и их характеристики

Во всех рассматриваемых ниже примерах роль ведущего в интерфейсе отводится МК, внешние ЦАП являются ведомыми и получают данные (число n), преобразуемые в уровень аналогового сигнала напряжения по закону Uout = n * Uref / 2^N, N – разрядность, Uref – опорное напряжение.

В среде моделирования Proteus VSM имеется достаточное количество моделей внешних ЦАП, в том числе с последовательным интерфейсом. Они расположены в разделе Data Converters → D/A Converters, различаются прежде всего разрядностью (8, 10, 12 разрядов) и типом интерфейса (SPI, I2C).

В Табл. 4.2 приведена выборка микросхем для ознакомления, технические описания находятся в папке DataSheets \ Serial-DAC-Proteus. В первой графе таблицы приведено их наименование, в следующей – тип последовательного интерфейса и его максимальная частота тактирования.

Таблица 4.2. ЦАП с ПИ из библиотеки Proteus VSM

Наимен.	ПИ, МГц	Разр.	Опора	Кан.	Выв.	Формат данных, биты от старш. к младш.
ad5310	SPI, 33	10	VCC	1	6	xxxx$d_9d_8d_7d_6$ $d_5d_4d_3d_2d_1d_0$xx
ad5320	SPI, 33	12	VCC	1	6	xxxx$d_{11}d_{10}d_9d_8$ $d_7d_6d_5d_4d_3d_2d_1d_0$
ad5601	SPI, 30	8	VCC	1	6	xx$d_7d_6d_5d_4d_3d_2$ $d_1d_0$xxxxxx
ad5611	SPI, 30	10	VCC	1	6	xx$d_9d_8d_7d_6d_5d_4$ $d_3d_2d_1d_0$xxxx
ad5621	SPI, 30	12	VCC	1	6	xx$d_{11}d_{10}d_9d_8d_7d_6$ $d_5d_4d_3d_2d_1d_0$xx
Max504	SPI, 10	10	VCC	1	14	xxxx$d_9d_8d_7d_6$ $d_5d_4d_3d_2d_1d_0$xx
Max515	SPI, 10	10	2.048, ±5	1	8	xxxx$d_9d_8d_7d_6$ $d_5d_4d_3d_2d_1d_0$xx
Max5258/9	SPI, 10	8	Внеш.<5	8	16	xx$c_2c_1c_0$110 $d_7d_6d_5d_4d_3d_2d_1d_0$
Mcp4821/2	SPI, 20	12	2.048*2	½	8	$c_0$0g1$d_{11}d_{10}d_9d_8$ $d_7d_6d_5d_4d_3d_2d_1d_0$**
Mcp4725	I2C, 3.4	12	VCC	1	6	110000$a_0$0 0000$d_{11}d_{10}d_9d_8$ d_7…d_0
Max5820	I2C, 0.4	8	Внеш.<5	2	8	011100$a_0$0 000$c_0d_7d_6d_5d_4$ d_3…$d_0$xxxx*
Max5821		10				011100$a_0$0 000$c_0d_9d_8d_7d_6$ d_5…$d_0$xx*
Max5822		12				011100$a_0$0 000$c_0d_{11}d_{10}d_9d_8$ d_7…d_0*
						01110000 11110000 00001100
Max517	I2C, 3.4	8	Внеш.<5	1	8	01011$a_1a_0$0 000000xxx d_7…d_0
Max518			VCC	2	8	01011$a_1a_0$0 000000xxc_0 d_7…d_0
Max519			Внеш.<5	2	16	010$a_3a_2a_1a_0$0 000000xxc_0 d_7…d_0

*Требуется инициализация для выхода из режима PowerDown, см. строку ниже

** Бит g управляет усилением. При VCC = 5 В выбирайте g = 0, Vout = D * 2.048*2 / 4096.

Разрядность (графа «Разр.») варьируется от 8 до 12 бит, диапазон выходного напряжения у всех однополярный, верхний предел (графа «Опора») у большинства задан напряжением питания VCC = 5 В, у других не превышает 5 В. Ряд микросхем объединяют несколько ЦАПов в одном корпусе (графа «Кан.»), в этом случае используется аппаратная или программная адресация канала. В графе «Выв.» указано число ног микросхемы, наименование выводов видно на условном графическом изображении в Протеусе. В графе «Формат данных, биты от старшего к младшему» указан формат кадра в виде последовательности двух или трех байт (в двоичном формате) для записи нового значения выбранного канала ЦАП, обозначения: x – не значащие биты (0), d_x – данные, a_x – аппаратный адрес микросхемы ЦАП, c_x – программный адрес канала в многоканальных ЦАП.

1.3.2. ЦАП с SPI (3-Wire) интерфейсом

Для передачи данных в ЦАП контроллер SPI в МК AVR использует выходы SCK, MOSI, вывод MISO не используется. Для формирования сигнала выборки каждой микросхемы ЦАП (активный уровень – 0) используется отдельный выход порта.

Наименование выводов микросхем ЦАП с интерфейсом SPI отличается разнообразием. Как правило, вход тактов называется CLK или SCK или SCLK, вход данных – DI или DIN или SDIN, выход данных (если имеется) – DO или SDO, вход выборки кристалла – CS или SYNC (со знаком инверсии) или LATCH. Аналоговый выход носит название OUT или VOUT (если выходов несколько, они нумеруются), вход опорного напряжения – REF или REFIN, при его отсутствии в качестве опорного напряжения используется напряжение питания (обычно +5 В, VCC или VDD).

Схема подключения двух 10 разрядных ЦАП AD5310 к МК ATmega16 в редакторе схем Proteus ISIS приведена на рис. 4.2, связи выполнены через значки выходящих и входящих темно-зеленых стрелок. Модель AD5310 параметров настройки не имеет, слева входы SPI интерфейса, справа аналоговый выход, выводы питания (VCC и GND.) скрыты.

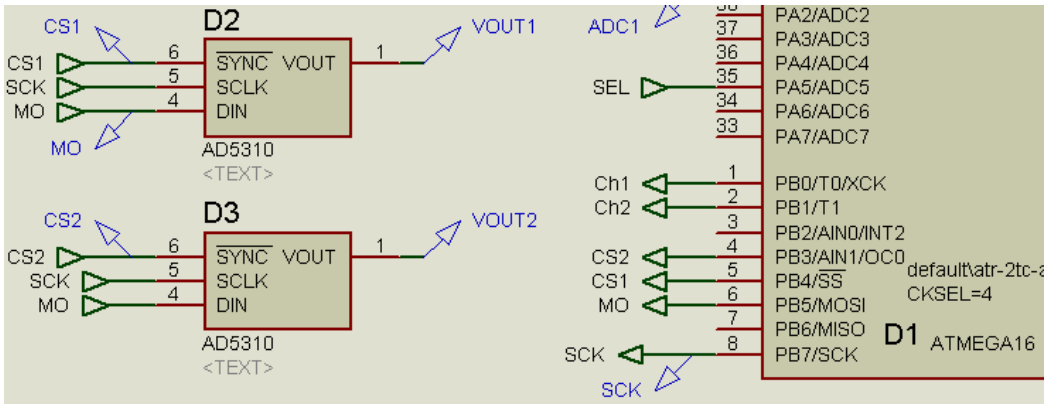


Рис. 4.2

Временная диаграмма, формат данных, программа.

Временная диаграмма посылки 10 битного числа к D2 AD5310 (рис. 4.3) состоит из двух «пачек» по 8 тактов на линии SCK, объединённых низким уровнем сигнала на линии CS1 и двух байт на линии MO.

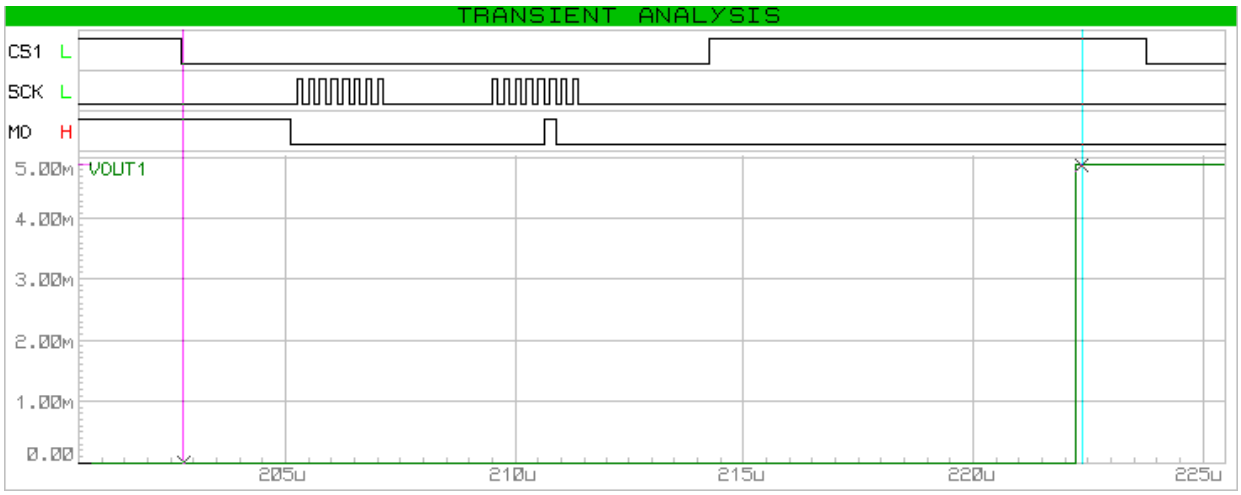


Рис. 4.3

Порядок бит данных «от старшего к младшему», смотрим формат данных в Табл. 4.1 (xxxx$d_9d_8d_7d_6$ $d_5d_4d_3d_2d_1d_0$xx), записываем двоичное значение числа (d_9…d_0) и преобразуем в десятичное 0b0000 0000 01 = 0x001 = 1. По формуле цифроаналогового преобразования рассчитываем значение напряжения на выходе ЦАП VOUT1 = 1 * 5 / 1024 = 4.88 мВ.

Частота тактирования МК fclk = 8 МГц, тактирование SPI в два раза ниже fspi = 4 МГц (0,25 мкс), длительность выборки (нулевого уровня CS1) составила 11.5 мкс. Через 8.1 мкс виден результат ЦА-преобразования – фронт напряжения VOUT1 амплитудой 4.88 мВ.

Некоторые микросхемы имеют более одного канала ЦАП, для выборки используются дополнительные биты в посылке, обозначенные в Табл. 4.1 c_x (двоичная адресация).

Программное обслуживание для передачи данных по SPI состоит из настройки регистров управления SPCR (включение модуля, выбор роли Ведущего и настроек временной диаграммы) и состояния SPSR (выбор тактирования на максимально доступной частоте) на этапе инициализации и функции void put_DAC_ad5310 (unsigned int value1, char ch) в основном цикле. Первый параметр (value1) передает значение для ЦАП, второй – номер канала (0 или 1), он преобразуется в соответствующий сигнал выборки (CS1 или CS2).

```
#define CS1 PB4
#
void put_DAC_ad5310(unsigned int value1, char ch) {
  unsigned char bbyte;
  if(ch==0) PORTB &= ~(1<<CS1); else PORTB &= ~(1<<CS2);//Срез CSx

  SPDR = value1>>6;//Загрузка первого байта данных
  while(!(SPSR &(1<<SPIF)));//Ожидание завершения передачи
  bbyte = SPDR;//Фиктивное чтение для сброса флага завершения

  SPDR = value1<<2;//Загрузка второго байта данных
  while(!(SPSR &(1<<SPIF)));
  bbyte = SPDR;

  if(ch==0) PORTB |= (1<<CS1); else PORTB |= (1<<CS2);//Фронт CSx
```

Структура функции внятно соответствует временной диаграмме (рис. 4.3). Операции сдвига с двухбайтным значением переменной value обеспечивают формирование формата данных при загрузке в регистр данных SPDR

Для работы с другими микросхемами ЦАП с SPI интерфейсом требуется некоторая коррекция схемы подключения (в том числе, выбор соответствующей модели в Proteus VSM) и двух байт временной диаграммы (по Табл. 4.1).

1.3.3. ЦАП с I2C (2-Wire) интерфейсом

Название линии микросхем ЦАП с интерфейсом I2C стандартное – такты SCL и данные SDA.

На рис. 4.5 приведена схема подключения 12-битного ЦАП MCP4725 (U2) к МК ATmega16. Резисторы R1, R2 притяжки к питанию обеспечивают высокий уровень на линиях в отсутствии тактов и данных. Вход A0 занулен и определяет значение младшего из 7 бит адреса Ведомого. Модель MCP4725 в Proteus VSM имеет параметр Address Option: A0, на рисунке 4.4 это отражено надписью «DVCADR = $C0», число «$C0» означает шестнадцатеричный адрес. Если вход A0 подключить к VCC, то адрес станет 0xC2, таким образом можно подключить сразу две микросхемы MCP4725. Выводы питания VDD и VSS подключены к источнику питания VCC (5 В), это же напряжение служит для опоры Uref.

Для увеличения диапазона аналогового сигнала до 10 В к выходу ЦАП VOUT подключена модель идеального усилителя с ku = 2, выполненная на базе источника напряжения управляемого напряжением E1 (число 2.0 индицирует значение параметра Voltage Gain, то есть ku).

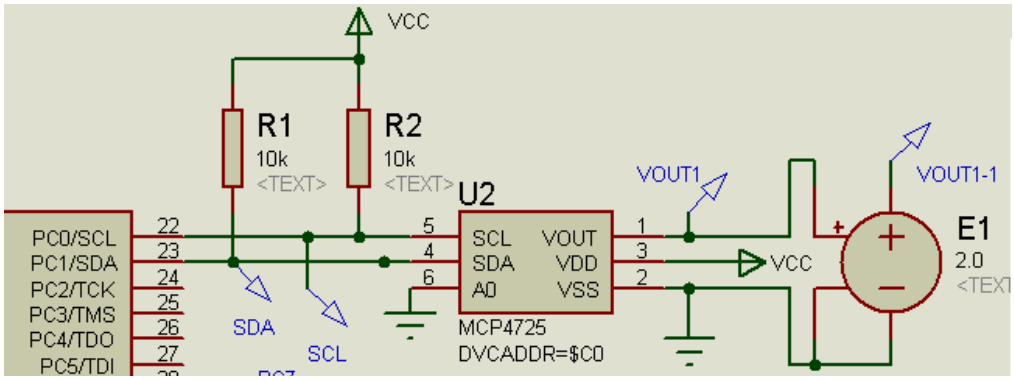


Рис. 4.4

На рис. 4.5 представлена временная диаграмма записи 12-битного числа $D_{11}…D_0$ в MCP4725 по адресу 0xC0 в соответствии с форматом из Табл. 4.1. Ведущий формирует комбинацию Start, затем 7 бит адреса Ведомого (1100 $00A_0$) и бит команды записи W = 0, на девятом такте ведомый (ЦАП) подтверждает свое наличие и готовность принять данные нулевым уровнем (Ack). Далее передаются два байта данных, каждый завершается ответом ведомого (Ack), заканчивается обмен комбинацией Stop.

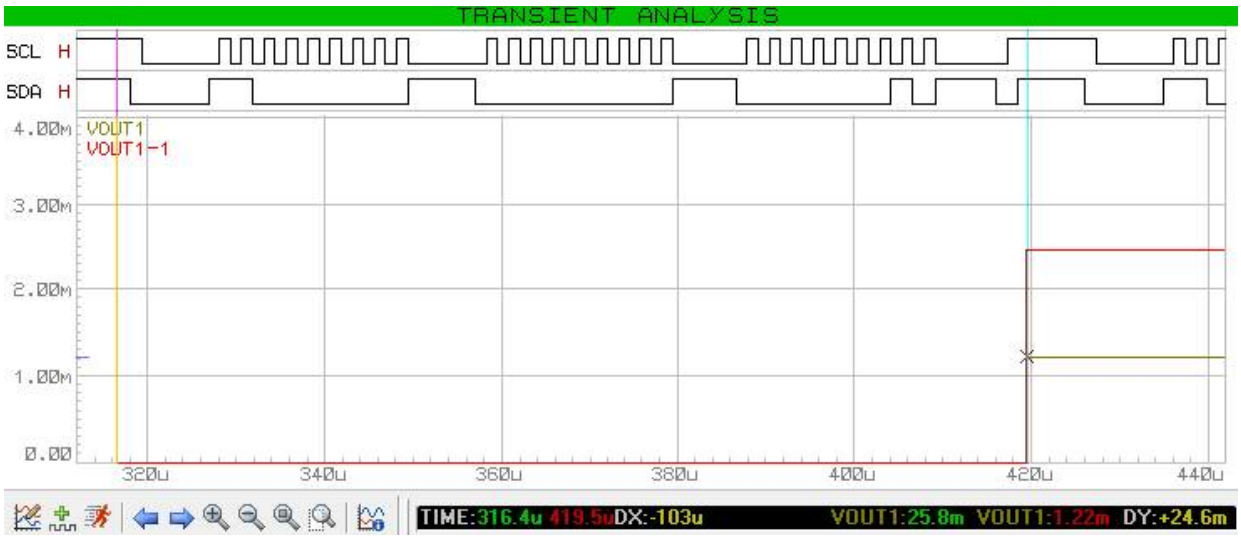


Рис. 4.5

Данные по формату выровнены по младшему биту второго байта, поэтому легко выделяются и преобразуются в число 0b0000 0000 0001 = 0x001 = 1. Этому числу соответствует напряжение на выходе VOUT = 1 * 5 / 4096 = 1.22 мВ. При частоте тактирования fscl = 400 кГц интервал между комбинациями Start и Stop занял 101 мкс. С задержкой в 1 мкс виден результат ЦА-преобразования – фронт напряжения VOUT1 амплитудой 1.22 мВ.

Программное обслуживание передачи данных по I2C состоит из функции настройки void init_i2c_Master(void), в основном цикле используется функция void put_DAC_mcp4725 (val[0], 0), первый параметр – передаваемое в ЦАП значение, второй – номер канала.

```
#define SLA_MCP4725 0b11000000
void put_DAC_mcp4725(unsigned int value, char ch) {
    i2c_start();//Формирование последовательности Start
    i2c_SLA_W(SLA_MCP4725 |(ch<<1));//Посылка адреса и команды записи
    i2c_write((unsigned char)(value>>8));//Посылка старших 4 бит данных
    i2c_write((unsigned char)value);//Посылка младшего байта данных
    i2c_stop();//Посылка последовательности Stop
} //************************
```

2. Готовые примеры

2.1. Задание

Разработать схему и программу терморегулятора с пропорциональным (аналоговым) выходом и цифровым индикатором температуры для тестирования в системе моделирования Proteus VSM.

Параметры измерителя и регулятора приведены в Табл. 4.3, параметры измерителя повторяют значения из лабораторной № 3. Отличия начинаются в группе «Регулирование»: графа «Dp, °C» задает диапазон пропорциональности, «ПИ» – тип интерфейса к внешнему ЦАП, «Uout, В» – диапазон напряжения аналогового выхода, «dU, мВ» – шаг изменения напряжения.

Табл. 4.3. Варианты четвертого индивидуального задания

№	Датчик	Кан.	Измерения		Регулирование					
			ºС	%	°Закон	Tset, ºС	Dp, ºС	ПИ.	Uout, В	dU, мВ
0-1	ТЭП ХА(К)	2	0…999	1	Нагреватель	700	100	SPI	0.5…4.5	5.0
0-2	ТС 100 Ом	1	0…500	0.5	Охладитель	250	200	I2C	2.5…7.5	2.5
1	ТЭП В	2	500…1300	1	Нагреватель	950	200	SPI	2.5…7.5	25.0
2	ТЭП Е	2	-150…0	1	Охладитель	-75	50	SPI	0.0…2.5	5.0
3	ТС 100 Ом	1	0…500	0.5	Нагреватель	250	40	I2C	5.5…9.5	10.0
4	ТЭП J	2	-50…450	1	Охладитель	200	100	SPI	2.5…5.0	10.0
5	ТЭП N	2	100…600	1	Нагреватель	400	90	SPI	1.0…4.0	2.5
6	ТС 100 Ом	2	0…500	1	Охладитель	250	150	I2C	2.5…7.5	10.0
7	ТЭП R	2	300…800	1	Нагреватель	500	75	SPI	1.5…9.5	20.0
8	ТС 100 Ом	2	-100…+200	1	Охладитель	100	20	SPI	0.1…4.9	15.0
9	ТЭП S	2	600…1100	1	Нагреватель	850	40	I2C	0.0…1.5	2.0
10	ТС 100 Ом	2	-100…+400	1	Охладитель	-20	30	SPI	1.0…3.0	5.0
11	ТЭП T	1	-100…+100	1	Нагреватель	0	40	SPI	4.5…9.5	5.0
12	LM20	2	-50…+100	2	Охладитель	50	40	I2C	-10…+10	100.0
13	LM35	1	-50…+150	2	Нагреватель	100	20	SPI	-5…+5	50.0
14	MCP9701	2	-40...+125	2	Охладитель	0	35	I2C	0…-5	40.0
15	PTC_NICKEL	2	-55...+300	2	Нагреватель	200	60	SPI	-2.5…+2.5	50.0
16	KTY81	2	-55...+300	2	Охладитель	-10	50	I2C	4.5…9.5	50.0
17	ТЭП К	1	-100…+100	1	Охладитель	0	90	SPI	-10…0	10
18	ТЭП К	3	-100…+900	1	Нагреватель	750	200	SPI	-5…+5	10
19	ТЭП К	4	300…1300	1	Нагреватель	800	400	I2C	-10…+10	20
20	KTY81	3	-45...+80	2	Охладитель	25	50	SPI	-2.5…+2.5	25

2.2. Схема электрическая

Большая часть схемы идентична схеме из Лаб.3, отличия только в выходе/ах регулятора: вместо светодиода, имитирующего «Нагреватель» или «Охладитель» ключевого типа, здесь к выводам SPI или I2C(TWI) контроллера МК подключена требуемая микросхема ЦАП, а к ее выходу при необходимости, цепи усиления и/или смещения аналогового сигнала.

Выбор ЦАП определяется типом интерфейса (SPI или I2C), разрядностью, в ряде случаев и числом каналов. Разрядностью выбирается в соответствии с числом квантов, 8, 10 и 12 разрядам соответствует число квантов N, равное 256, 1024 и 4096. Число квантов выбирается по значениям диапазона и шага выходного напряжения $N \geq (Umax - Umin) / dU$.

Так для примера 0-1 (Umax – Umin) / dU = (4.5 – 0.5) / 0.005 = 800, выбираем ближайшее большее 1024, то есть 10 разрядов. Диапазон напряжений почти всех представленных 10 разрядных ЦАПов 0…5 В, шире заданного, поэтому удобно ограничить значения программно. В этом случае шаг напряжения dUreal = Uref / N = 5 / 1024 = 4.88 мВ меньше заданного, значения пределов нижнего Nmin = 0.5 * 1024 / 5 = 102.4, верхнего Nmax = 4.5 * 1024 / 4.5 = 921.6, дробная часть отбрасывается.

Первый 10 разрядный ЦАП с SPI интерфейсом AD5310, выше в примере раздела 1.3.2 приведены схема и программа его обслуживания.

Для примера 0-2 диапазон напряжения (2.5…7.5 В) выходит за пределы диапазона 0…5 В, выбрано усиление в два раза и программное ограничение. Поэтому при выборе разрядности (Umax – Umin) / dU = (10 – 0) / 0.0025 = 4000, выбираем ближайшее большее 4096, то есть 12 разрядов. (При выборе смещения 2.5 В (7.5 – 2.5) / 0.0025 = 2000, тоже 12!)

Схема и программа при выборе 12 разрядного ЦАП с I2C интерфейсом MCP4725 подробно рассмотрены выше в разделе 1.3.3.

2.3. Алгоритм и программа

Большая часть алгоритма Лаб.3 сохраняется, только вместо функции ключевого регулятора появляется пара новых функций: вычисления значения пропорционального регулятора и посылки этого значения на выбранный ЦАП посредством встроенного контроллера последовательного интерфейса. Формат значения – без знаковое двухбайтное число, выбран для универсальности, в основном цикле продолжаем использовать val[x]

```
val[0] = reg_P_heater(T[0]);// или reg_P_cooler…
put_DAC_ad5310(val[0], 0); // или для 0-2: put_DAC_mcp4725(val[0], 0);
```

Диапазон чисел на выходе функции пропорционального регулятора зависит от разрядности ЦАП и наличия/отсутствия программных ограничений. Для удобства использован механизм макровычислений. Вариант 0-1:

```
#define TSET 700L /* Задание, °C */
#define DP 100 /* Задание, °C */
#define Umin 500L /* Задание, мВ */
#define Umax 4500L /* Задание, мВ */
#define Nmax 1024L /* Число квантов */
#define Uref 5000L /* Опорное напряжение, мВ */

#define nRegMIN (Umin * Nmax / Uref) /*102 нижнее значение регулятора */
#define nRegMAX (Umax * Nmax / Uref) /*921 верхнее значение регулятора*/
#define nRegMID ((nRegMAX + nRegMIN)/2) /*511 среднее значение регулятора */
#define KU100 ((nRegMAX - nRegMIN)*100 / DP / 2) /*409 к-нт передачи рег-ра*/

unsigned int reg_P_heater(unsigned int x) {
  signed int y;
  if(x < (TSET - DP))      y = nRegMAX;//Верхнее ограничение
  else if(x > (TSET + DP)) y = nRegMIN;//Нижнее ограничение
  else { /* Зона пропорциональности */
    y = TSET - x;//Рассогласование
    y = ((signed long) y * KU100) / 100;//Усиление
    y += nRegMID;//Смещение
  }
  return (unsigned int)y;
} // ***************************************
```

Функция охладителя отличается только знаком при вычислении смещения: `y -= nRegMID;`

На этапе инициализации добавляется настройка контроллера последовательного интерфейса: выбор режима работы, тактирования.

Удобно предусмотреть на этапе инициализации возможность отладки новых функций.

Тестирование регулятора нагреватели или охладителя удобно выполнять в среде AVR-Studio, используя окно Watch для наблюдения и редактирования переменных.

Тестирование вывода числа на внешний ЦАП возможно только Proteus VSM, для этого организуется цикл посылки чисел 0, 1, 2, 3, 4 на ЦАП.

2.4. Состав файлов примеров

В папке Lab4\atr-2tc-spi находится готовый пример 2-канального терморегулятора по заданию 0-1 из Табл.4.3, он состоит из файла модели Proteus VSM <ATR-2TC-SPI-m16.DSN>, файла проекта AVR-Studio <atr-2tc-ad5310-m16.aps>, файла текста программы <atr-2tc-ad5310-m16.c>.

В папке Lab4\atr-rtd-i2c находится готовый пример 1-канального терморегулятора по заданию 0-2 из Табл.4.3, он состоит из файла модели Proteus VSM <ATR-RTD-I2C-m16.DSN>, файла проекта AVR-Studio <atr-rtd-mcp4725-m16.aps>, файлов текстов программы <atr-rtd-mcp4725-m16.c>, <i2c.c>, <i2c.h>.

2.5. Тестирование примеров

Тестирование готовых примеров целесообразно выполнять в Proteus VSM. Для этого кликните на файл проекта <*.dsn>.

Сначала надо убедиться в исправной работе последовательного интерфейса и аналогового выхода. Это удобно сделать по временным диаграммам цифровых сигналов интерфейса (CS, SCK, MOSI для SPI и SCL, SDA для I2C) и аналогового сигнала на выходе ЦАП VOUT1 в окне «Mixed analysis» (заголовок «Transient analysis»).

В примере 0-1 подготовлено окно с сигналами CS1, SCK, MO в цифровом виде (отдельной «строкой» каждый) и VOUT1 в аналоговом, длительность расчета 300 мкс (рис. 4.6).

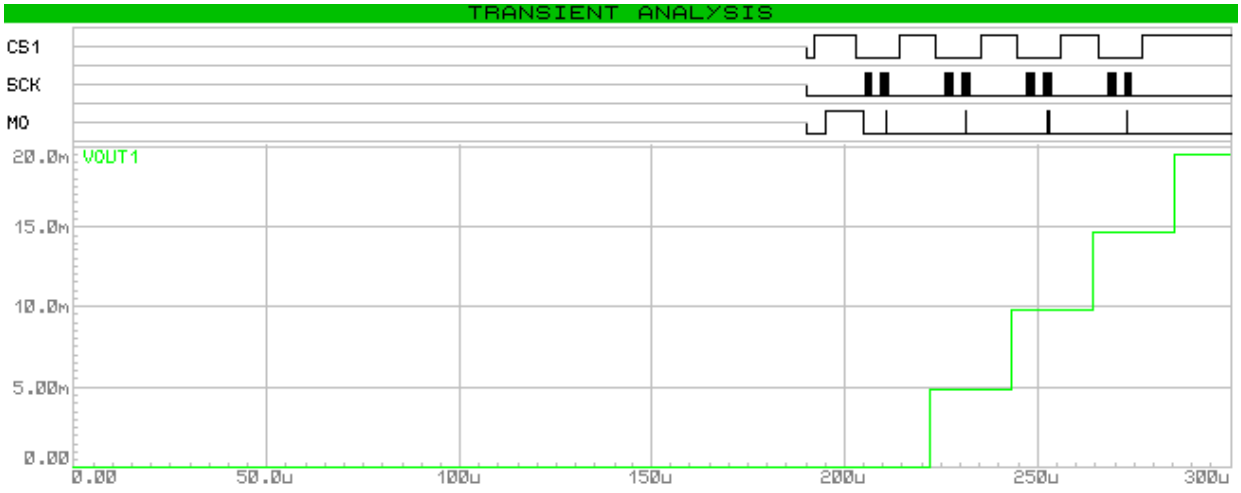


Рис. 4.6

После 200 мкс виден цикл из 4 примерно одинаковых периодических сигналов интерфейса, сопровождающихся ступенчатым ростом напряжения на выходе ЦАП AD5310 до уровня чуть ниже 20 мВ, то есть шаг около 5 мВ – соответствует ожидаемому.

Для детального анализа работы последовательного интерфейса удобно с помощью контекстного меню перейти в режим Maximize (Show Window), затем кнопками выбора масштаба и интервала времени можно получить изображение, похожее на рис. 4.3 и самостоятельно выполнить измерения временных интервалов и уровней напряжения с использование пары курсоров.

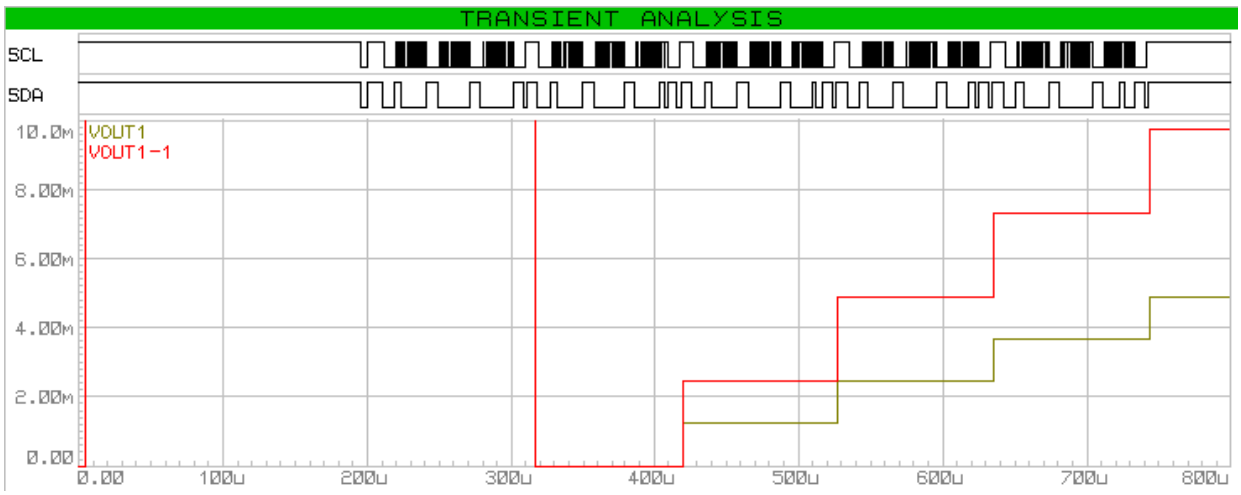


Рис. 4.7

Аналогичные наблюдения можно выполнить и в примере 0-2 для сигналов SCL, SDA в цифровом виде и VOUT1, VOUT1-1 в аналоговом, длительность расчета 800 мкс (рис. 4.7 и 4.5).

В интерактивном режиме (рис. 4.8) удобно тестировать статическую регулировочную характеристику, варьируя значение температуры датчика (TC1 610 °C), наблюдая численные значения индицируемой температуры (614 °C) и напряжения на аналоговом выходе регулятора (VOUT1 V=4.20898).

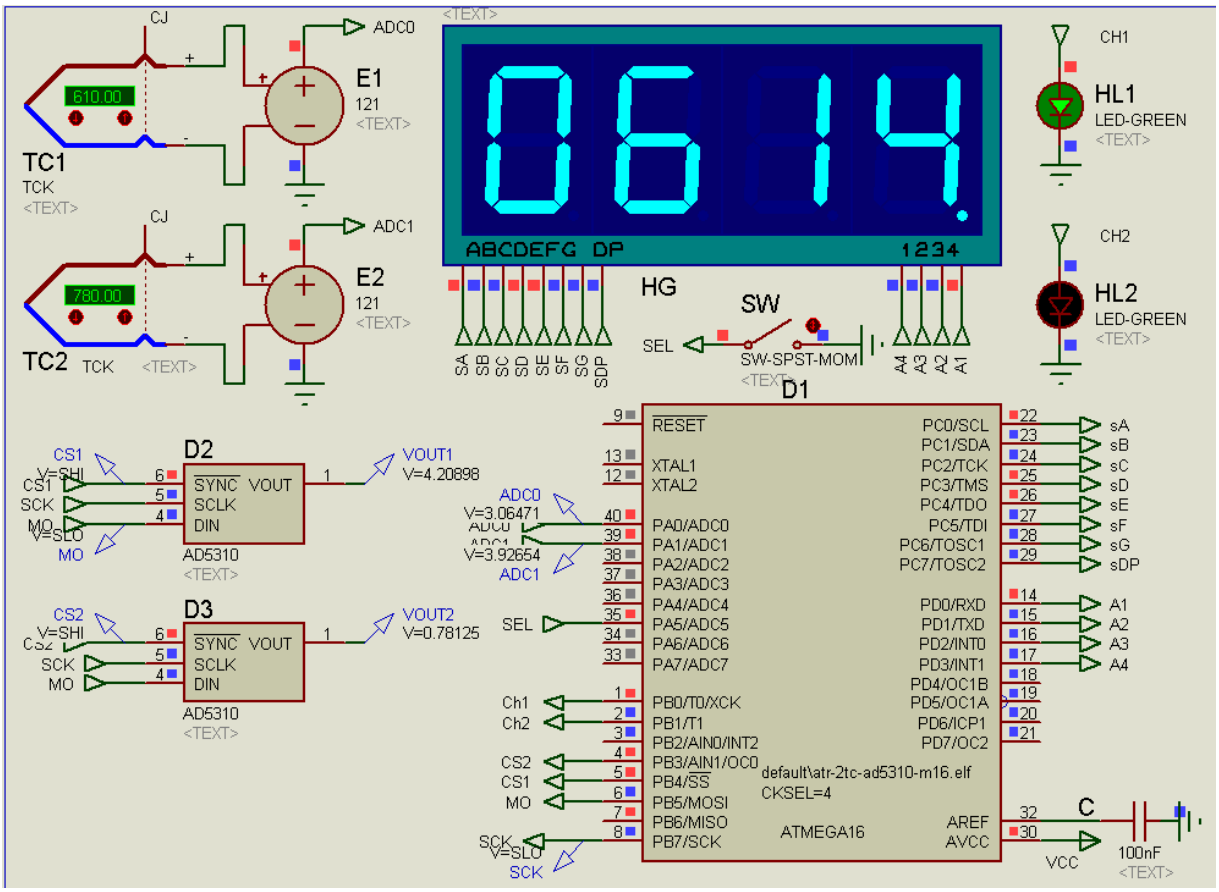


Рис. 4.8

Важно пройти переходы от пропорционального участка к участкам насыщения с малым шагом, например, TC1 = 0, 590, 600, 610, … 790, 800, 810, 1000, записать значения Тинд и Vout1 в таблицу и построить график (рис. 4.9).

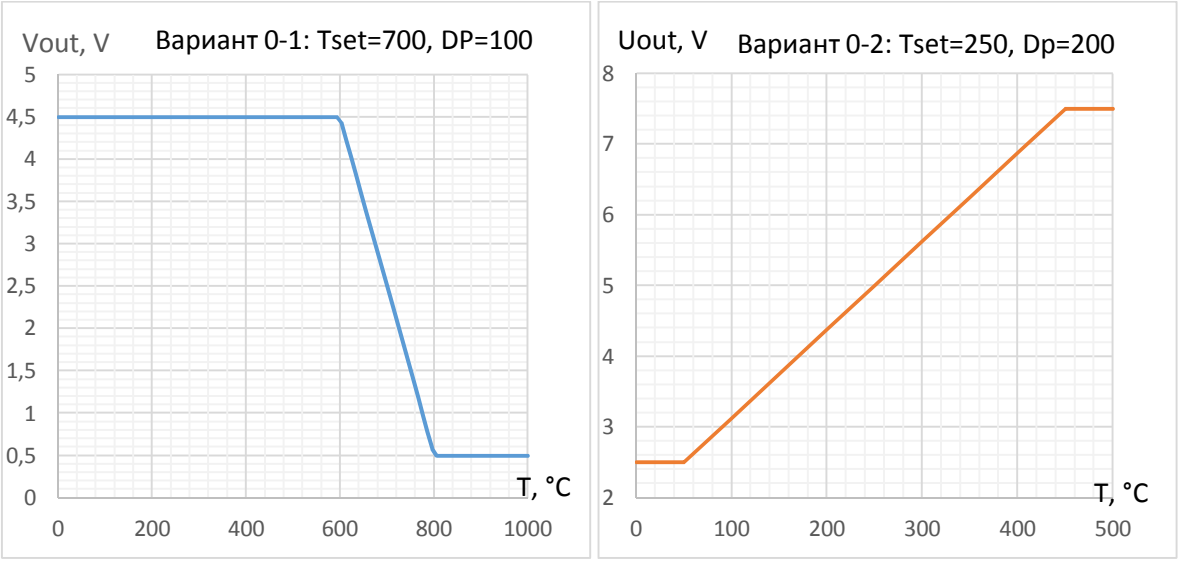


Рис. 4.9

Рис. 4.10

Для оценки времени реакции требуется измерить период основного цикла. Это удобно сделать по диаграмме сигналов на входе и выходе ЦАПа/ов. Интерактивная диаграмма рис. 4.10 показывает значение периода 768 мкс (вариант 0-1), интерактивность позволяет менять значения датчиков в процессе ее построения.

3. Разработка индивидуального задания

3.1. Полный текст задания приведен в п. 2.1, в Табл. 4.3. приведены параметры вариантов заданий.

3.2. Выбор ЦАП

Из Табл. 4.2 по типу интерфейса, разрядности и числу каналов. Разрядность n задаёт число квантов $N = 2^n$, минимально необходимое число квантов определяется диапазоном и шагом выходного напряжения $N\min = U / dU$, $N \geq N\min$..

3.3. Схема подключения ЦАП

Схема подключения определяется типом интерфейса, особенностями выбранной микросхемы ЦАП (адресация, питание, опорное напряжение). При необходимости выходной аналоговый сигнал схемно усиливается и/или смещается.

3.4. Функция вывода данных в ЦАП

Функция вывода данных через встроенный контроллер последовательного интерфейса put_DAC_name (value, number) разрабатывается по аналогии с функциями из 1.3.2, 1.3.3 для выбранного ЦАП (name), оперирует с регистрами данных и управления/состояния. Различия между ЦАП сводятся к типу интерфейса, особенностям адресации и формату данных. Тестирование проводить аналогично (временная диаграмма в Proteus VSM) в частном программном цикле для наблюдения и сохранения временной диаграммы интерфейса и проверки уровней аналогового сигнала.

3.5. Пропорциональный регулятор

Выбора типа пропорционального регулятора, коррекция данных с учётом требований диапазона и выбранного ЦАП. Тестирование в интерактивном режиме в Proteus VSM вариацией температуры датчика для построения передаточной характеристики Uout (Тинд).

3.6. Измерение времени реакции алгоритма

По временным диаграмма сигналов последовательного интерфейса измерить длительность инициализации, основного цикла. Оценить долю времени последовательного интерфейса.

3.7. Содержание отчета

1. Задание в краткой и полной форме.
2. Функциональная схема регулятора.
3. Выбор ЦАП и других элементов аналогового выхода
4. Схема электрическая терморегулятора с перечнем элементов.
5. Текст программы: функции ЦАП, терморегулятора, переменные, main с комментариями.
6. Временная диаграмма тестирования интерфейса и аналогового выхода с описанием способа тестирования.
7. Диаграмма статической характеристики регулятора Uout(T, Tset, Dp).
8. Тайминг программы и оценка времени реакции.
9. Расходы ПП, ПД и выводов МК.

Список литературы

1. Бондаренко Д. Н. Компьютерная и микропроцессорная техника. Часть 1. Встраиваемые микроконтроллеры AVR8: Электронное учеб. пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2015. – 205 с.

2. Голик С.Е. Микроконтроллеры для систем управления: учеб. Пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2015. 160 с.

3. Евстифеев А.В. Микроконтроллеры AVR семейств Tiny и Mega фирмы “ATMEL”. – М.: Издательский дом «Додэка-XXI», 2008. – 560 с.

Приложение

Текст программы генерации ШИМ на языке С

```
// Пример генерации ШИМ сигнала длительностью 64, 128, … 448, период 512 мкс
#include <avr/io.h> /* Символьные константы ввода/вывода */
#include <avr/interrupt.h> /* Символьные константы прерываний */

unsigned char nti, ntp;//число квантов
// ***************************************
ISR(TIMER0_OVF_vect) { //Обработчик прерывания переполнения таймера 0
  if(!(PORTB & (1<<0))) { //Выбор действия
    PORTB |= (1<<0); //Фронт
    TCNT0 = -nti; //Загрузка числа квантов
  } else {
    PORTB &= ~(1<<0); // Срез
    TCNT0 = -ntp; //Загрузка числа квантов
  }
} // Выход из прерывания

int main() {
char d; //параметр вариации
  PORTA = 0xff; //Включить притяжку входов PA0…7
  DDRB = (1<<0);//Настройка выхода импульсов
  DDRC = 0xff;  DDRD = 0xff;
  d = PINA & 0x07;
  nti = d;   ntp = 8 - d;
  // T/C0 init
  TIMSK =(1<<TOIE0); //разрешение прерывания переполнения TC1
  sei();// Global enable interrupts - общее разрешение прерываний
  TCNT0 = -nti;//загрузка для первого импульса
  TCCR0 = (1<<CS01)|(1<<CS00);//Пуск тактир-я ftc0 = fclk/64 = 1/64MHz = 64us
  PORTB |= (1<<0); //Фронт
  while (1) {
    d = PINA & 0x07; //& 0x0f; //Чтение параметра вариации
    { nti = d;  ntp = 8 - d;  }
  };
} //-O1 - первый уровень оптимизации
```

Листинг программы генерации ШИМ

```
gen_ovf.elf:     file format elf32-avr
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000104  00000000  00000000  00000074  2**1
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .bss          00000002  00800060  00800060  00000178  2**0
                  ALLOC
  2 .debug_aranges 00000020  00000000  00000000  00000178  2**0
                  CONTENTS, READONLY, DEBUGGING
  3 .debug_pubnames 0000003a  00000000  00000000  00000198  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_info   000000c6  00000000  00000000  000001d2  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_abbrev 00000076  00000000  00000000  00000298  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_line   000000e6  00000000  00000000  0000030e  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_frame  00000030  00000000  00000000  000003f4  2**2
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    00000082  00000000  00000000  00000424  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .debug_loc    00000013  00000000  00000000  000004a6  2**0
                  CONTENTS, READONLY, DEBUGGING
```

```
Disassembly of section .text:

00000000 <__vectors>:
   0: 0c 94 2a 00       jmp   0x54  ; 0x54 <__ctors_end>
   4: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>
   8: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>
   c: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>
  10: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>
  14: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>
  18: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>
  1c: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>
  20: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>
  24: 0c 94 3e 00       jmp   0x7c  ; 0x7c <__vector_9>
  28: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>
  2c: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>
  30: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>
  34: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>
  38: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>
  3c: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>
  40: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>
  44: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>
  48: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>
  4c: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>
  50: 0c 94 3c 00       jmp   0x78  ; 0x78 <__bad_interrupt>

00000054 <__ctors_end>:
  54: 11 24             eor   r1, r1
  56: 1f be             out   0x3f, r1    ; 63
  58: cf e5             ldi   r28, 0x5F   ; 95
  5a: d4 e0             ldi   r29, 0x04   ; 4
  5c: de bf             out   0x3e, r29   ; 62
  5e: cd bf             out   0x3d, r28   ; 61

00000060 <__do_clear_bss>:
  60: 10 e0             ldi   r17, 0x00   ; 0
  62: a0 e6             ldi   r26, 0x60   ; 96
  64: b0 e0             ldi   r27, 0x00   ; 0
  66: 01 c0             rjmp  .+2         ; 0x6a <.do_clear_bss_start>

00000068 <.do_clear_bss_loop>:
  68: 1d 92             st    X+, r1

0000006a <.do_clear_bss_start>:
  6a: a2 36             cpi   r26, 0x62   ; 98
  6c: b1 07             cpc   r27, r17
  6e: e1 f7             brne  .-8         ; 0x68 <.do_clear_bss_loop>
  70: 0e 94 63 00       call  0xc6  ; 0xc6 <main>
  74: 0c 94 80 00       jmp   0x100 ; 0x100 <_exit>

00000078 <__bad_interrupt>:
  78: 0c 94 00 00       jmp   0     ; 0x0 <__vectors>

0000007c <__vector_9>:
#include <avr/io.h> /* Символьные константы ввода/вывода */
#include <avr/interrupt.h> /* Символьные константы прерываний */

unsigned char nti, ntp;
// ***************************************
ISR(TIMER0_OVF_vect) { //Обработчик прерывания
  7c: 1f 92             push  r1
  7e: 0f 92             push  r0
  80: 0f b6             in    r0, 0x3f    ; 63
  82: 0f 92             push  r0
  84: 11 24             eor   r1, r1
```

```
  86: 8f 93             push  r24
  88: ef 93             push  r30
  8a: ff 93             push  r31
  if(!(PORTB & (1<<0))) { //Выбор действия
  8c: c0 99             sbic  0x18, 0     ; 24
  8e: 0a c0             rjmp  .+20        ; 0xa4 <__vector_9+0x28>
    PORTB |= (1<<0); //Фронт
  90: e8 e3             ldi   r30, 0x38   ; 56
  92: f0 e0             ldi   r31, 0x00   ; 0
  94: 80 81             ld    r24, Z
  96: 81 60             ori   r24, 0x01   ; 1
  98: 80 83             st    Z, r24
    TCNT0 = -nti; //Загрузка числа квантов
  9a: 80 91 60 00       lds   r24, 0x0060
  9e: 81 95             neg   r24
  a0: 82 bf             out   0x32, r24   ; 50
  a2: 09 c0             rjmp  .+18        ; 0xb6 <__vector_9+0x3a>
  } else {
    PORTB &= ~(1<<0); // Срез
  a4: e8 e3             ldi   r30, 0x38   ; 56
  a6: f0 e0             ldi   r31, 0x00   ; 0
  a8: 80 81             ld    r24, Z
  aa: 8e 7f             andi  r24, 0xFE   ; 254
  ac: 80 83             st    Z, r24
    TCNT0 = -ntp; //Загрузка числа квантов
  ae: 80 91 61 00       lds   r24, 0x0061
  b2: 81 95             neg   r24
  b4: 82 bf             out   0x32, r24   ; 50
  }
} // Выход из прерывания
  b6: ff 91             pop   r31
  b8: ef 91             pop   r30
  ba: 8f 91             pop   r24
  bc: 0f 90             pop   r0
  be: 0f be             out   0x3f, r0    ; 63
  c0: 0f 90             pop   r0
  c2: 1f 90             pop   r1
  c4: 18 95             reti

000000c6 <main>:

int main() {
char d; //Число квантов
  PORTA = 0xff; //Включить притяжку входов PA0…7
  c6: 8f ef             ldi   r24, 0xFF   ; 255
  c8: 8b bb             out   0x1b, r24   ; 27
  DDRB = (1<<0);//Настройка выхода импульсов
  ca: 21 e0             ldi   r18, 0x01   ; 1
  cc: 27 bb             out   0x17, r18   ; 23
  DDRC = 0xff;  DDRD = 0xff;
  ce: 84 bb             out   0x14, r24   ; 20
  d0: 81 bb             out   0x11, r24   ; 17
  d = PINA & 0x07;
  d2: 99 b3             in    r25, 0x19   ; 25
  d4: 97 70             andi  r25, 0x07   ; 7
  nti = d;   ntp = 8 - d;
  d6: 90 93 60 00       sts   0x0060, r25
  da: 88 e0             ldi   r24, 0x08   ; 8
  dc: 89 1b             sub   r24, r25
  de: 80 93 61 00       sts   0x0061, r24
  // T/C0 init
  TIMSK =(1<<TOIE0); //разрешение прерывания переполнения TC1
  e2: 29 bf             out   0x39, r18   ; 57
  sei();// Global enable interrupts - общее разрешение прерываний
```

```
  e4: 78 94             sei
  TCNT0 = -nti;//загрузка для первого импульса
  e6: 91 95             neg   r25
  e8: 92 bf             out   0x32, r25   ; 50
  TCCR0 = (1<<CS01)+(1<<CS00); //Пуск тактирования ftc0 = fclk/64 = 1/64MHz =
64us
  ea: 83 e0             ldi   r24, 0x03   ; 3
  ec: 83 bf             out   0x33, r24   ; 51
  PORTB |= (1<<0); //Фронт
  ee: e8 e3             ldi   r30, 0x38   ; 56
  f0: f0 e0             ldi   r31, 0x00   ; 0
  f2: 80 81             ld    r24, Z
  f4: 81 60             ori   r24, 0x01   ; 1
  f6: 80 83             st    Z, r24
  while (1) {
    d = PINA & 0x07; //& 0x0f; //Чтение параметра вариации
  f8: e9 e3             ldi   r30, 0x39   ; 57
  fa: f0 e0             ldi   r31, 0x00   ; 0
  fc: 80 81             ld    r24, Z
  fe: fe cf             rjmp  .-4         ; 0xfc <main+0x36>

00000100 <_exit>:
 100: f8 94             cli

00000102 <__stop_program>:
 102: ff cf             rjmp  .-2         ; 0x102 <__stop_program>
```