МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Национальный исследовательский университет «МЭИ»

Кафедра «Автоматизированного электропривода»

Д. И. Савкин, Д. М. Шпак

МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА В ЭЛЕКТРОПРИВОДЕ

Практикум
для студентов, обучающихся по направлению
13.04.02 «Электроэнергетика и электротехника»

ISBN 978-5-7046-3136-1

УДК 621.34 + 621.398
ББК 31.291
С 135

Утверждено учебным управлением НИУ «МЭИ»
в качестве учебного издания

Подготовлено на кафедре автоматизированного электропривода

Рецензенты: д. т. н., проф. А. С. Анучин;
к. т. н., доц. Г. Л. Демидова

Савкин, Д. И.

С 135 Микропроцессорные средства в электроприводе [Электронный ресурс]: практикум / Д. И. Савкин, Д. М. Шпак. – Электрон. дан. – М.: Издательство МЭИ, 2024. – 1 электрон. опт. диск CD-ROM.

Практикум содержит примеры вопросов и заданий, рассматриваемых в курсе «Микропроцессорные средства в электроприводе» и выполняемых в рамках лабораторных работ.

Настоящее издание предназначено для самостоятельной проработки изучаемых разделов курса студентами, обучающимися по направлению 13.04.02 «Электроэнергетика и электротехника» по программе «Электропривод и автоматика», а также может быть использовано студентами других направлений и программ.

Минимальные системные требования:
Компьютер: процессор x86 с тактовой частотой 500 МГц и выше;
ОЗУ 512 Мб; 20 Мб на жестком диске;
Видеокарта: SVGA 1280x1024 High Color (32 bit);
Операционная система: Windows XP/7/8 и выше;
Дополнительные программные средства: Adobe Acrobat Reader версии 6 и выше.

ISBN 978-5-7046-3136-1

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Настоящий практикум является дополнением к учебному пособию «МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА В ЭЛЕКТРОПРИВОДЕ НА БАЗЕ МИКРОКОНТРОЛЛЕРОВ TMS320F28035: ЛАБОРАТОРНЫЙ ПРАКТИКУМ» [1]. Указанное учебное издание доступно для скачивания по ссылке в Приложении 4.

Практикум по содержанию соответствует учебной программе курса «Микропроцессорные средства в электроприводе», читаемого в НИУ «МЭИ» кафедрой Автоматизированного электропривода по направлению 13.04.02 «Электроэнергетика и электротехника». В издании собраны контрольные вопросы и задания для лучшего освоения материалов курса и для самостоятельной подготовки. Прежде, чем решать задачи и отвечать на вопросы каждого из разделов, рекомендуется изучить соответствующий материал учебного издания [1], выполнить задания на лабораторные работы, а также воспользоваться материалами лекций.

Сборник контрольных вопросов и заданий окажет помощь студентам в самостоятельном изучении курса, подготовке к экзамену, защите лабораторных работ и успешном выполнении индивидуальных заданий.

Для оформления отчётов к выполненным индивидуальным заданиям по лабораторным работам авторы рекомендуют пользоваться шаблоном, представленным в Приложении 1 и доступным для скачивания по ссылке из приложения.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ №1

Цель работы

Знакомство с аппаратными возможностями оценочной платы, на базе которой будут выполняться все лабораторно-практические работы, особенностями организации памяти, интерфейса с оборудованием пользователя, порядком подключения к компьютеру. Освоение технологии распределения памяти.

Материалы для подготовки: Лабораторная работа №1, стр. 7 – 34 [1].

Контрольные вопросы

1. Каков размер переменной типа "unsigned long" в битах?
2. Что означает ключевое слово "unsigned" при описании переменной?
3. Как объявить в программе на языке Си переменную, которая может принимать любое из следующих значений: 0, 159, -1124, 40000?
4. Сколько бит содержится в одной ячейке памяти микроконтроллера TI F28035?
5. Какие значения может принимать переменная типа "int"?
6. Какие значения может принимать переменная типа "unsigned char"?
7. Сколько ячеек памяти микроконтроллера TI F28035 потребуется для хранения 18 переменных типа "long"? Почему именно столько?
8. Сколько ячеек памяти микроконтроллера TI F28035 потребуется для хранения 24 переменных типа "int"? Почему именно столько?
9. Какие функции выполняет файл компоновки памяти?
10. Каково назначение секции ".cinit"?
11. Каково назначение секции ".stack"?
12. Каково назначение секции ".text"?
13. Каково назначение секции ".ebss"?

14. Что означает следующая запись в файле управления компоновкой?

```
SECTIONS {.ebss: > RAML2}
```

15. Что означает следующая запись в файле управления компоновкой?

```
MEMORY {RAM :origin = 0x0800, length = 0x0240}
```

Контрольные задания

1. Имеется следующий файл управления компоновкой (лист. 1). Какие в нём допущены ошибки и как их исправить?

Листинг 1. Пример файла управления компоновкой

```
MEMORY
{
PAGE 0 :
   RAMM0      : origin = 0x000050, length = 0x0003B0
   RAML0L1    : origin = 0x008000, length = 0x000C00

PAGE 1 :
   RAMM1       : origin = 0x000480, length = 0x000380
   RAML2       : origin = 0x008C00, length = 0x000400
   RAML3       : origin = 0x009000, length = 0x001000
}
SECTIONS
{
   .text            : > RAML0L1,   PAGE = 0
   .cinit           : > FLASH,     PAGE = 0
   .stack           : > RAMM1,     PAGE = 1
   .ebss            : > RAML2,     PAGE = 1
}
```

2. Дана следующая информация о памяти и секциях программы:

   - ✓ доступен блок FLASH-памяти размером 200 ячеек, который начинается с адреса 800;
   - ✓ доступен блок RAM-памяти размером 200 ячеек, который начинается с адреса 1000;
   - ✓ секции с текстом программы и константами расположены в блоке FLASH-памяти;
   - ✓ стек и секция глобальных переменных расположены в блоке RAM-памяти.

   Приведите пример файла управления компоновкой, который соответствует приведённому выше описанию.

3. В файле управления компоновкой определены области памяти согласно лист. 2. Заполните раздел SECTIONS так, чтобы правильно разместить все секции программы при условии, что в вашей программе:

   ✓ 1000 переменных типа unsigned int, из них 100 переменных проинициализированы значениями по умолчанию;

   ✓ текст программы занимает 110 ячеек памяти;

   ✓ размер стека составляет 600 ячеек.

Листинг 2. Пример файла управления компоновкой

```
MEMORY
{
   MEM0      : origin = 0x000000, length = 0x000100
   MEM1      : origin = 0x001000, length = 0x000400
   MEM2      : origin = 0x005000, length = 0x000200
   MEM3      : origin = 0x005000, length = 0x000300
}
SECTIONS
{
      .data  ...
      .cinit ...
      .stack ...
      .text  ...
}
```

4. В файле управления компоновкой определены следующие области памяти (см. лист. 3)

Листинг 3. Пример файла управления компоновкой

```
MEMORY
{
   MEM0      : origin = 0x000000, length = 0x000100
   MEM1      : origin = 0x001000, length = 0x000400
   MEM2      : origin = 0x005000, length = 0x000200
   MEM3      : origin = 0x006000, length = 0x000300
}
SECTIONS
{
      .data  ...
      .cinit ...
      .stack ...
      .text  ...
}
```

Заполните раздел SECTIONS так, чтобы правильно разместить все секции программы при условии, что в вашей программе:

   ✓ 400 переменных типа unsigned int, из них 200 переменных проинициализированы значениями по умолчанию;

- ✓ текст программы занимает 800 ячеек памяти;
- ✓ размер стека составляет 50 ячеек.

Вернуться к началу раздела

Вернуться к содержанию

КОНТРОЛЬНОЕ ЗАДАНИЕ К ЛАБОРАТОРНОЙ РАБОТЕ №2

Цель работы

Получение начальных сведений по технологии разработки и отладки простых программ в среде Code Composer Studio [2] на языке высокого уровня Си. Создание проекта, создание командного файла компоновщика, создание исходного файла с программой на C/C++, установка опций компилятора и компоновщика. Трансляция и загрузка программы на выполнение. Отладка программы в пошаговом режиме с использованием окон памяти и окон наблюдаемых переменных.

Материалы для подготовки: Лабораторная работа №2, стр. 35 – 66 [1].

Контрольное задание

1. Создать *.cmd файл отличный от примеров из учебника [1]. Ориентируйтесь на карту памяти ([1] – рис. 1.2, с. 12). Помимо стандартных секций (.text, .bss и т.д.) создайте собственную секцию (например с именем .mysect). Создайте блок памяти с именем INTBLOCK, начиная с адреса 0x8500 с минимальным размером, достаточным для размещения одной переменной типа int. Учтите, что новый блок не должен пересекаться в адресном пространстве с уже существующими блоками. Включите текст файла *.cmd в отчёт.

2. Создать несколько глобальных и локальных, знаковых и беззнаковых переменных разных типов (short, long, float) и выполнить с ними простые арифметические действия. Несколько переменных должны размещаться в вашей новой секции. Проследить за преобразованиями типов. Включить текст программы в отчёт.

3. После компиляции проанализировать *.map файл и просмотреть память контроллера в окне "Expressions". Убедиться, что переменные в памяти расположены соответственно файлу *.map. Поместить в блок INTBLOCK переменную типа int с именем "fixedInt" (пример приведён ниже). В таблице отобразить список переменных с указанием их типов и адресов. Вставить скриншоты окна "Expressions" для подтверждения правильности данных таблицы. Обязательно отобразить переменную "fixedInt".

4. В тексте программы присвойте переменной типа int значение 40000 и выведите её значение в окно "Expressions". Объясните результат. Присвойте этой же переменной значение 30000. Сдвиньте его на один бит влево, затем на один бит вправо. В отчёт включите значение переменной после каждой операции (присваивание, сдвиг влево, сдвиг вправо) и комментарии к результатам.

5. Выполните операцию присвоения переменной значения суммы, разности и умножения двух других переменных. Сравните получившиеся ассемблерные коды программ. Сделайте комментарий к каждой ассемблерной операции для всех вариантов.

Пояснения по выполнению контрольного задания

Для создания собственной секции достаточно добавить её название в описании SECTIONS cmd–файла (см. лист. 4).

Листинг 4. Пример cmd-файла с новой секцией

```
SECTIONS
{
      .text    :> FLASH PAGE 0
      .cinit   :> FLASH1 PAGE 0
      .ebss    :> M1SARAM PAGE 1
      .stack   :> M0SARAM PAGE 1
      newsect :> M1SARAM
}
```

В примере из лист. 4 создана пользовательская секция "newsect" в блоке памяти M1SARAM. Чтобы поместить какую-либо переменную в новую секцию, необходимо перед объявлением переменной добавить директиву #pragma DATA_SECTION (см. лист. 5).

Листинг 5. Пример размещения переменной i в пользовательской секции “newsect

```
#pragma DATA_SECTION(i, "newsect")
float i=0;
```

Таким образом, переменная окажется в блоке памяти M1SARAM, так как именно в нём расположена секция "newsect". Для побитового сдвига влево используется оператор "<<", а вправо – ">>".

Пример показан в лист. 6.

Листинг 6. Пример применения операторов сдвига

```
int a;              // Объявление переменной
void main (void){
      a = 10;       // Присвоение переменной значения «10»
      a = a << 2;   // Сдвиг значение на два бита влево
      a = a >> 3;   // Сдвиг значения на три бита вправо
}
```

В процессе выполнения в двоичном представлении 8-битная переменная будет выглядеть так, как показано в таблице ниже.

Таблица 1

Значение переменной в двоичном виде после выполнения операций сдвига

Операция	Двоичное представление до выполнения операции	Двоичное представление после выполнения операции	Десятичное представление после выполнения операции
a = 10;	0000 0000	0000 1010	10

Окончание таблицы 1

Операция	Двоичное представление до выполнения операции	Двоичное представление после выполнения операции	Десятичное представление после выполнения операции
a = a << 2;	0000 1010	0010 1000	40
a = a >> 3;	0010 1000	0000 0101	5

Вернуться к началу раздела

Вернуться к содержанию

КОНТРОЛЬНЫЕ ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ №3

Цель работы

Общие сведения о языке программирования Си. Изучение основных типов данных языка Си для микроконтроллеров TMS320F28035. Освоение новых возможностей по отладке программ в среде Code Composer Studio с использованием функций стандартной библиотеки ввода/вывода. Пошаговое выполнение и отладка программы на Си и Ассемблере с использованием окон регистров процессора, как эффективное средство изучения архитектуры и системы команд процессора.

Материалы для подготовки: Лабораторная работа №3, стр. 66 – 90 [1]

Контрольные задания

Задание 3.1

Рассчитать значение *i*-члена арифметической прогрессии с начальным значением *a* и разностью *d*.

Требования к программе.

Расчёт значения должен быть реализован в виде функции, принимающей два аргумента типа "int" – начальное значение и разность, один

аргумент типа "unsigned int" – порядковый номер искомого члена, и возвращающей значения типа "int" – значение искомого члена (лист. 7).

Листинг 7. Описание функции задания 3.1

```
int function(int a, int d, unsigned int i);
```

Пример вызова функции см в лист. 8.

Листинг 8. Вызов функции задания 3.1

```
int result;
int a = 5;
int d = -3;
result = function(a, d, 10);
// В соответствии с заданными параметрами в переменной «result»
// должно оказаться значение «-22» - значение десятого
// члена арифметической прогрессии
```

Задание 3.2

Дано значение *x*. Требуется выполнить следующее:

1. Рассчитать площадь круга с радиусом *x*, и площадь квадрата со стороной *x* и сохранить эти значения в переменных *Sc* и *Ssq*.

2. Рассчитать такое значение стороны квадрата, при котором его площадь была бы равна *Sc*. Сохранить значение в переменной *a*.

3. Рассчитать такое значение радиуса круга, при котором его площадь была бы равна *Ssq*. Сохранить значение в переменной *b*.

Требования к программе.

Расчёты всех значений должны быть реализованы в виде отдельных функций:

- функция расчёта площади круга должна принимать аргумент типа "float" (радиус круга) и возвращать результат типа "float" (площадь круга);
- функция расчёта площади квадрата должна принимать аргумент типа "float" (сторона квадрата) и возвращать результат типа "float" (площадь квадрата);
- функция расчёта стороны квадрата должна принимать аргумент типа "float" (площадь квадрата) и возвращать результат типа "float" (сторона квадрата);
- функция расчёта радиуса круга должна принимать аргумент типа "float" (площадь круга) и возвращать результат типа "float" (радиус круга) (см. лист. 9).

Листинг 9. Описание функции задания 3.2

```
float circle_square(float radius);
```

Пример вызова функции в лист. 10.

Листинг 10. Вызов функции задания 3.2

```
int rad = 12;
Sc = function(rad);
// В соответствии с заданными параметрами в переменной «Sc»
// должно оказаться значение «452.16» - площадь круга с
// радиусом 12
```

Задание 3.3

Рассчитать объёмы шара с радиусом *r* и конуса с радиусом основания *r* и высотой *h*. Сравнить их объёмы и записать результат сравнения в переменную *result*, где 1 – объем шара больше объёма конуса; 2 – объём шара меньше объёма конуса; 3 – объёмы равны.

Требования к программе.

Расчёты всех значений должны быть реализованы в виде отдельных функций:

- функция расчёта объёма шара должна принимать аргумент типа "float" (радиус шара) и возвращать результат типа "float" (объём шара);
- функция расчёта объёма конуса должна принимать два аргумента типа "float" (радиус основания и высота) и возвращать результат типа "float" (объём конуса);
- функция сравнения объёмов должна принимать два аргумента типа "float" (объём шара и объём конуса) и возвращать результат типа "unsigned int" (результат сравнения согласно заданию) (см. лист. 11).

Листинг 11. Описание функции задания 3.3

```
int compare(float v1, float v2);
```

Пример вызова функции см. в лист. 12.

Листинг 12. Вызов функции задания 3.3

```
float v1 = 20;
float v2 = 10;
int compare_res = compare(v1, v2);
// В переменной «compare_res» должно оказаться значение «1»,
// если мы считаем, что v1 – это объём шара, v2 - конуса
```

Задание 3.4

Рассчитать площадь треугольника ABC по значениям трех его сторон (формула Герона); затем рассчитать три высоты треугольника по известной площади и длинам сторон; затем рассчитать три площади треугольника по разным комбинациям высот и сторон (3.1).

$$\begin{cases} S = AB \cdot h_1 \\ S = BC \cdot h_2 \\ S = CA \cdot h_3 \end{cases} \tag{3.1}$$

Требования к программе.

Расчёты всех значений должны быть реализованы в виде отдельных функций:

- функция расчёта площади по трём сторонам должна принимать три значения типа "float" (длины сторон) и возвращать значение типа "float" (площадь треугольника);
- функция расчёта высоты треугольника должна принимать два аргумента типа "float" (площадь треугольника и длины стороны, противолежащей углу, из которого проведена высота) и возвращать значение типа "float" (длина высоты треугольника);
- функция расчёта площади треугольника по высоте и длине стороны должна принимать два аргумента типа "float" (длина высоты и длина стороны) и возвращать значение типа "float" (площадь треугольника) (см. лист. 13).

Листинг 13. Описание функции задания 3.4

```
// Прототип функции расчёта площади по высоте и стороне
float squareHeight(float sideLen, float height);
```

Пример вызова функции см. в лист. 14.

Листинг 14. Вызов функции задания 3.4

```
float S2 = squareHeight (4, 7);
// В переменной «squareHeight» должно оказаться значение «14»,
// так как площадь треугольника равна «0.5 * 4 * 7»
```

Задание 3.5

Определить, находится ли точка с координатами (x, y) внутри окружности, на окружности или вне окружности, центр которой находится в точке с координатами ($x1$, $y1$) и радиусом r.

Требования к программе.

Определение местоположения точки должно быть реализовано в виде отдельной функции, которая в качестве аргументов принимает два аргумента типа "float" (координаты *x* и *y*) и возвращает значение типа "unsigned int", где 0 – точка лежит внутри окружности; 1 – точка лежит на окружности; 2 – точка лежит вне окружности (см. лист. 15).

Листинг 15. Описание функции задания 3.5

```
int checkPoint(float x, float y);
```

Пример вызова функции см. в лист. 16.

Листинг 16. Вызов функции задания 3.5

```
float x1 = 0, x2 = -4.2, r = 1;
int pointPosition = checkPoint(1, 8);
// В переменной «compare_res» должно оказаться значение «1»,
// если мы считаем, что v1 – это объём шара, v2 - конуса
```

Задание 3.6

Дан прямоугольный параллелепипед с длинами рёбер *a*, *b*, *c*.
Разработать следующие функции:

- функция расчёта объёма параллелепипеда $V = \sqrt{a \cdot b \cdot c}$;
- функция расчёта площади поверхности $S = 2(a \cdot b + b \cdot c + a \cdot c)$;
- функция расчёта диагонали $d = \sqrt{a^2 + b^2 + c^2}$;
- функция, проверяющая, можно ли вписать в данный параллелепипед сферу. Сферу можно вписать в том случае, если площади всех граней параллелепипеда равны. Функция должна возвращать 1, если сферу вписать можно, и 0, если нельзя.

Требования к программе.

Каждая функция должна принимать три аргумента типа "float" – длины рёбер параллелепипеда – и возвращать одно значение: для функций расчёта объёма, площади поверхности и длины диагонали – типа "float", для функции проверки возможности вписывания сферы – типа "int".

Вернуться к началу раздела

Вернуться к содержанию

КОНТРОЛЬНЫЕ ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ №4

Цель работы

Продолжение знакомства с базовыми возможностями языка программирования Си. Написание и отладка в среде Code Composer Studio простых программ обработки данных. Работа со стандартными математическими библиотеками, в том числе с тригонометрическими вычислениями. Изучение технологии работы с массивами данных, указателями и структурами.

Материалы для подготовки: Лабораторная работа №4, стр. 91 – 117 [1].

Контрольные задания

Задание 4.1

Дана тригонометрическая функция (см. табл. 2), где k – произвольное число.

1. Рассчитать значения заданной тригонометрической функции в нескольких точках через равные отрезки (см. табл. 2). Заполнить произвольный массив результатами расчета.

2. Выполнить дополнительное задания для полученного массива данных. *Дополнительное задание необходимо выполнить в виде отдельной функции, в которую по указателю передается полученный массив.*

Таблица 2

Таблица вариантов задания 4.1

k	Функция	Приращение x	Дополнительно
1	$k \cdot \cos(2x + k) + 4k$	$1/6 \cdot \pi$; 10 точек	Подсчитать количество положительных результатов
2	$k \cdot \sin(4x + 2k) - 1$	$1/8 \cdot \pi$; 10 точек	Подсчитать количество отрицательных результатов
3	$4k \cdot \sin(x / k) + 1$	$1/5 \cdot \pi$; 10 точек	Подсчитать количество результатов большие 1 по модулю
4	$3 \cdot \cos(x + 2)$	$2/7 \cdot \pi$; 10 точек	Найти наибольшее значение
5	$4k \cdot \sin(x / k) + 1$	$2/9 \cdot \pi$; 10 точек	Найти наименьшее значение
6	$k \cdot \sin(4x + 2k)$	$1/6 \cdot \pi$; 10 точек	Найти наибольшее значение
7	$k \cdot \cos(2x + k) + 4k$	$1/7 \cdot \pi$; 10 точек	Подсчитать количество элементов меньше k
8	$\cos(k \cdot x - k) + k$	$3/11 \cdot \pi$; 10 точек	Найти наибольшее значение
9	$3 \cdot cos(x + 2)$	$2/11 \cdot \pi$; 10 точек	Найти наименьшее значение
10	$\cos(k \cdot x - k) + k$	$3/11 \cdot \pi$; 10 точек	Найти наименьшее значение

Задание 4.2

1. Задать функцию $F(x)$ в соответствии с вариантом (см. табл. 3). Найти корень уравнения $F(x) = 0$ методом бисекции (половинного деления) на отрезке заданном отрезке с заданной точностью. Сохранить корень функции в переменной и вывести переменную в окно "Expressions".

2. Подсчитать количество шагов, за которое был найден корень функции. Сохранить количество шагов в переменной и вывести переменную в окно "Expressions".

3. Занести в массив 10 значений функции $F(x)$ на заданном отрезке (x_n, x_k) через равные промежутки. Вывести массив в окно "Expressions".

Таблица 3

Таблица вариантов задания 4.2

№	Уравнение	Отрезок	Точность (*e)*
1	$x^4 - 100\sqrt{x+4}$	[0;5]	0,01
2	$x^3 - 5\sqrt{x+3}$	[0;5]	0,01
3	$\sqrt{x+4} - \frac{1}{0{,}005x^2}$	[0;20]	0,001
4	$2\sqrt{x^3+4} + \frac{1}{-0{,}005x^3}$	[1;6]	0,01
5	$0{,}1x^5 - 0{,}05x^3 - 7\sqrt{\frac{1}{x+1}}$	[-0.4;4]	0,01
6	$(0{,}1x)^2 - 15 + \frac{1}{\sqrt{x+50}}$	[30;40]	0,01
7	$\frac{10}{x+5} - \frac{2}{(x+6)^3}$	[-10;-6]	0,01

Окончание таблицы 3

№	Уравнение	Отрезок	Точность (*e*)
8	$0{,}01(x+4)^3-(x-7)^2+x+4$	[0;10]	0,01
9	$0{,}01(0{,}1x+5)^2-0{,}1(x-10)-\sqrt{x+10}$	[1000;1500]	0,1
10	$(-0{,}1x+4)^3+0{,}1(x-2)$	[55; 60]	0,01

Для проверки можно воспользоваться приведёнными в Приложении 2 графиками заданных функций.

Примечание.

Формулировка «найти корень функции с точностью *e*» в данном случае означает, что при подстановке в функцию найденного значения её результат должен находится в диапазоне (–*e*; +*e*).

Алгоритм решения методом половинного деления представлен ниже.

Задание 4.3

1. Решить уравнение (4.1) методом половинного деления с точностью e. Найти корень на интервале [0; *T*), где *T* – период функции (в радианах). Заданная точность *e* должна определяться по выражению (4.2).

2. Подсчитайте количество итераций, которое потребовалось для нахождения решения.

$$k \cdot \sin(k + 10 \cdot k \cdot x) - \frac{k}{100} = 0, \qquad (4.1)$$

где *k* – номер варианта.

$$e = \frac{1}{100 \cdot k} \qquad (4.2)$$

Для решения могут понадобиться следующие элементы языка Си:

- ✓ оператор условия if () {…} else if {…} else {…}
- ✓ циклы: while(){…} / do{...}while() / for(…){…}

- ✓ описание некоторых функций из файла "math.h" (не забудьте подключить его в тексте программы директивой #include)

Алгоритм решения методом половинного деления представлен ниже.

Алгоритм метода бисекции (половинного деления)

Входные данные:

- ✓ $F(x)$ – заданная функция;
- ✓ x_n, x_k – начало и конец отрезка, на котором нужно найти корень уравнения;
- ✓ x_i, dx – переменные для определения новых границ отрезка;
- ✓ Sign$(F(x_i))$ – функция определения знака $F(x)$ в заданной точке. *В стандартных библиотеках данная функция отсутствует, её надо составить самостоятельно;*
- ✓ e – заданная точность решения.

Алгоритм решения представлен в лист. 17.

Листинг 17 Алгоритм метода бисекции

```
1. Начало;
2. Ввод xn, xk, e;
3. Если F(xn) = 0, то Вывод (корень уравнения – xn);
4. Если F(xk) = 0, то Вывод (корень уравнения – xk);
5. Пока (|F(xi)| > e) повторять
      5.1 dx = dx / 2;
      5.2 xk = xn + dx;
      5.3 если sign(F(xn)) ≠ sign(F(xi)), то xk = xi;
      5.4 иначе xn := xi;
      5.5 конец повторения;
6. Вывод (Найден корень уравнения – xi с точностью e);
7. Конец.
```

Вернуться к началу раздела

Вернуться к содержанию

КОНТРОЛЬНЫЕ ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ №5

Цель работы

Получение навыков по использованию стандартных средств обращения к регистрам периферийных устройств, по подключению к проекту заголовочных файлов с описаниями встроенной периферии. Изучение технологии распределения памяти данных под встроенные периферийные устройства в командном файле компоновщика. Обучение программированию периферийных устройств. Получение навыков работы с дискретными портами ввода/вывода. Получение навыков программирования таймера общего назначения с использованием прерываний.

Материалы для подготовки: Лабораторная работа №5, стр. 118 – 159 [1].

Контрольные задания

Реализовать управление светодиодами на плате ControlCARD F28035 согласно указанному в задании алгоритму.

Во время работы должно подсчитываться количество переключений каждого светодиода. Значения переменных необходимо вывести в окно "Expressions".

В данном задании значения, обозначенные буквами *k*, *n* и так далее, в программе должны быть представлены переменными. При изменении значений этих переменных во время работы программы через окно "Expressions", поведение программы должно меняться соответствующим образом.

Светодиоды управляются выводами процессора GPIO31 и GPIO34.

Задание 5.1

Первый светодиод светится *k* миллисекунд, затем отключается на 1 с. Включается второй светодиод и светится *n* миллисекунд, затем отключается на 1 с. Затем снова светится первый и т.д.

Задание 5.2

Первый и второй светодиоды *k* раз по очереди включаются и выключаются на 500 мс, затем выжидается пауза *n* миллисекунд. После этого процесс повторяется.

Задание 5.3

Первый диод горит *n* миллисекунд, затем выключается; затем второй диод мигает *k* раз. Снова включается первый диод и так по кругу.

Задание 5.4

Первый светодиод мигает *k* раз с частотой 10 Гц, затем второй светодиод мигает *n* раз с частотой 5 Гц.

Задание 5.5

Реализовать следующий алгоритм переключения светодиодов. Сначала только первый диод мигает *n* раз. Затем оба диода синхронно мигают *k* раз. Затем только второй диод мигает *m* раз. И так по кругу.

Задание 5.6

Реализовать следующий алгоритм переключения светодиодов. Первый диод горит 1 с. Затем выключается. После этого второй диод мигает *k* раз. Снова включается первый на 1 с. Затем второй мигает *n* раз.

Задание 5.7

Реализовать следующий алгоритм мигания диодами.

Если $k = 1$, то первый светодиод мигает *n* раз, затем второй светодиод мигает *m* раз.

Если $k = 0$, то наоборот (первый – *m* раз, второй – *n* раз).

Задание 5.8

Если $k = 1$, то первый светодиод постоянно мигает с периодом равным *y* мс. Если $k = 0$, то не мигает.

Если $n = 1$, то второй светодиод постоянно мигает с периодом равным *y* мс. Если $n = 0$, то не мигает.)

Задание 5.9

Два светодиода должны мигать синхронно с одинаковой частотой.

Частота мигания должна определяться числом k в Герцах. При этом k должно автоматически ограничиваться в диапазоне от 1 до 10 включительно (т.е. если пользователь вводит $k > 10$, то оно становится равным 10).

Число n определяет скважность светодиодов и может меняться в диапазоне от 1 до 3 включительно. При значении 1 скважность должна быть 15%, при значении 2 – 50%, при значении 3 – 90%.

Задание 5.10

Реализовать следующую логику переключения светодиода:

- светодиод загорается на 200 мс и гаснет на 200 мс 1 раз;
- пауза 500 мс;
- светодиод загорается на 200 мс и гаснет на 200 мс 2 раза;
- пауза 500 мс;
- светодиод загорается на 200 мс и гаснет на 200 мс 3 раза;
- пауза 500 мс;
- ...
- светодиод загорается на 200 мс и гаснет на 200 мс k раз;
- пауза 500 мс;
- процесс повторяется с первого шага.

Аналогичный алгоритм для второго светодиода, но максимальное число миганий для него определяется переменной n, а не k.

Вернуться к началу раздела

Вернуться к содержанию

КОНТРОЛЬНЫЕ ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ №6

Цель работы

Знакомство с машинной арифметикой: форматы представления знаковых и дробных чисел. Знакомство с библиотекой IQmath. Знакомство с принципами построения цифровых фильтров и расчётом типовых звеньев систем автоматического управления (САУ).

Материалы для подготовки: Лабораторная работа №6, стр. 160 – 192 [1].

Контрольные задания

Задание 6.1

Задание выполняется с использованием IQ–математики.

1. Настроить прерывание от заданного таймера с заданной частотой.
2. Реализовать функцию инерционного звена с передаточной функцией (6.1):

$$W_{\text{ор}}(p) = \frac{k_{\text{ор}}}{T_{\text{ор}} \cdot p + 1}, \qquad (6.1)$$

где $T_{\text{ор}}$ – постоянная времени объекта регулирования (ОР); $k_{\text{ор}}$ – коэффициент усиления ОР.

3. Получить график реакции этой переходной функции на скачок амплитудой «1.0».
4. Реализовать функцию релейного регулятора $W_{\text{р}}(p)$:
   - если ошибка на входе регулятора больше нуля, выход регулятора равен «2»;
   - иначе выход регулятора равен «–2».
5. Используя функцию $W_{\text{ор}}(p)$ в качестве передаточной функции объекта регулирования и $W_{\text{р}}(p)$ в качестве регулятора, получить график реакции замкнутой системы на скачок с амплитудой «1.0». Кроме того, получить графики входного сигнала и выхода регулятора.

Код передаточной функции должен быть оформлен в виде отдельной функции, которая принимает один аргумент – вход передаточной функции, и возвращает значение выхода.

Аналогично, код регулятора должен быть оформлен в виде функции, которая принимает два аргумента – задание и обратную связь, и возвращает значение выхода регулятора.

Расчёт регулятора и объекта регулирования должен производиться в прерывании таймера с заданной частотой согласно варианту задания (см. табл. 4).

Таблица 4

Варианты исходных данных задания 6.1

Вариант	Таймер	Частота прерывания	Постоянная времени ОР (T_{op})	Коэффициент усиления ОР (k_{op})
1	CpuTimer0	10 кГц	0,2 с	2
2	CpuTimer1	12 кГц	0,3 с	3
3	CpuTimer2	5 кГц	0,1 с	4
4	CpuTimer1	8,5 кГц	0,15 с	8
5	CpuTimer2	15 кГц	0,23 с	1
6	CpuTimer0	4 кГц	0,08 с	2
7	CpuTimer1	9 кГц	0,12 с	6
8	CpuTimer2	11 кГц	0,04 с	7
9	CpuTimer0	7 кГц	0,25 с	9
10	CpuTimer2	8 кГц	0,11 с	4

Ниже приведены рисунки, поясняющие цель работы.

Структура для получения реакции передаточной функции на скачок и результирующий график показаны на рис. 1 и 2.

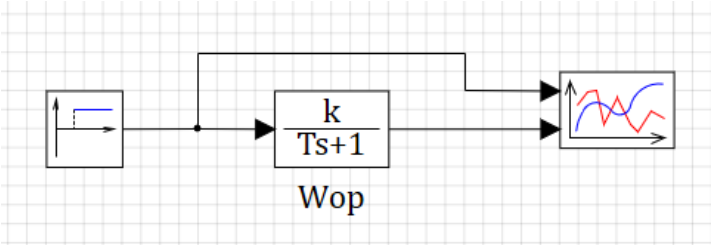


Рис. 1. Структурная схема системы для подачи скачка на объект управления в разомкнутой системе

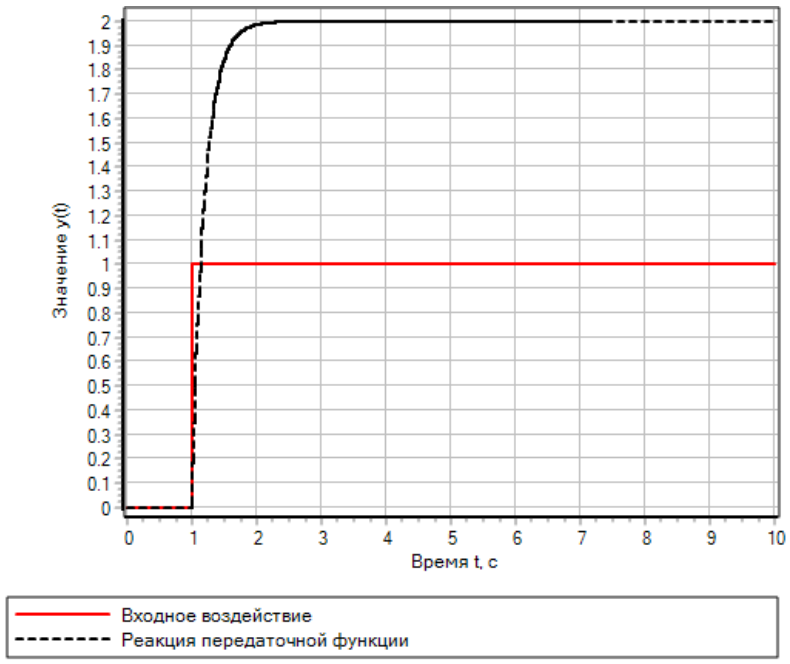


Рис. 2. График отклика объекта управления на скачок задания в разомкнутой системе

Структура для получения реакции замкнутой системы (регулятор + передаточная функция) на скачок и результирующий график показаны на рис. 3 и 4.

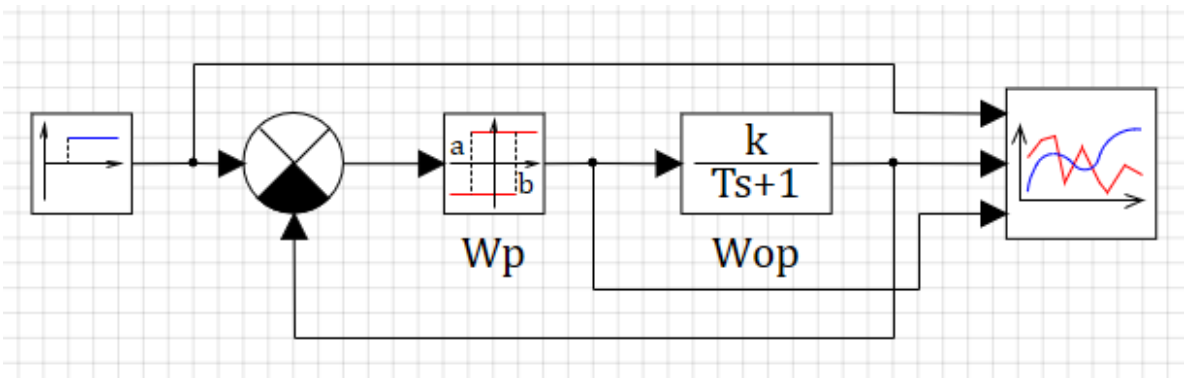


Рис. 3. Структурная схема системы для подачи скачка на объект управления в замкнутой системе

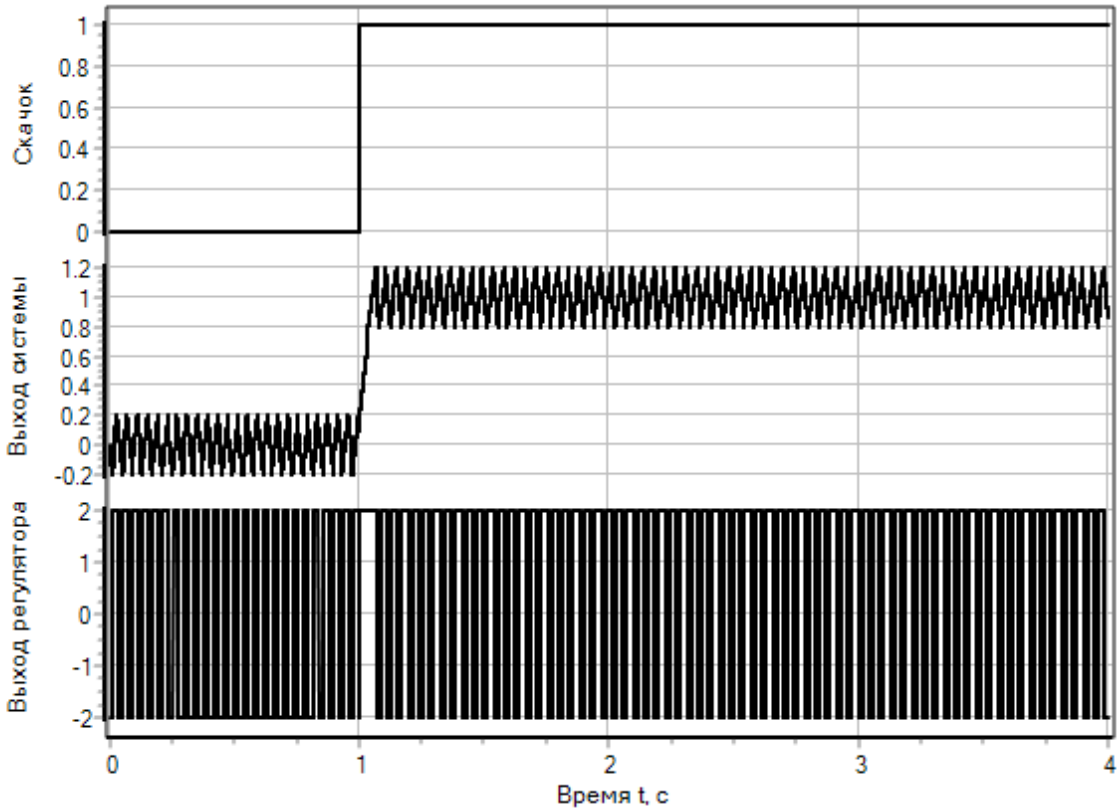


Рис. 4. График отклика объекта управления на скачок задания в замкнутой системе

Задание 6.2

В Приложении 3 заданы варианты структурных схем для выполнения задания.

Пример одного из вариантов схемы представлен на рис. 5

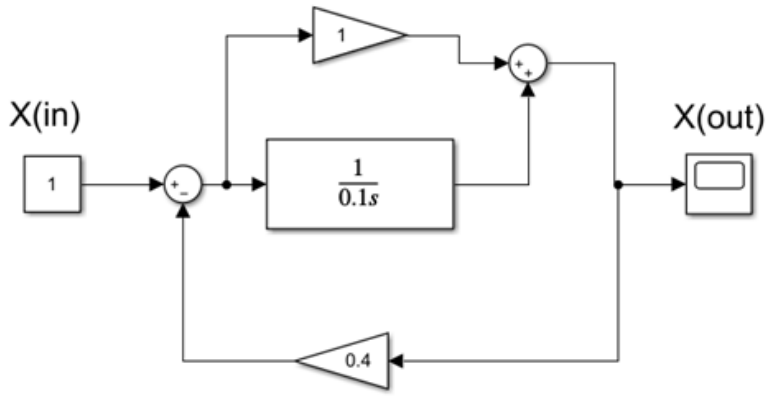


Рис. 5. Пример структурной схемы для задания 6.2

Задание.

1. Рассчитать заданную структурную схему (Приложение 3 – номер варианта согласно списку группы).
2. Вывести график изменения выходной переменной от времени – *Xout*(*t*).
3. Все расчеты выполнять с использованием IQ–математики.
4. Все расчеты выполнять обязательно в прерывании.
5. Частота вызова прерывания должна быть равна номеру варианта в кГц (например, 11 вариант – частота прерывания 11 кГц).
6. Расчет передаточной функции должен длиться ровно 1 секунду.
7. По результату выполнения задания подготовить отчёт (см. Приложение 1).

Пример выполнения.

Для удобства реализации программного кода добавим на структурную схему (рис. 5) обозначение переменных, которые будут связаны со входами и выходами элементов схемы (см. рис. 6).

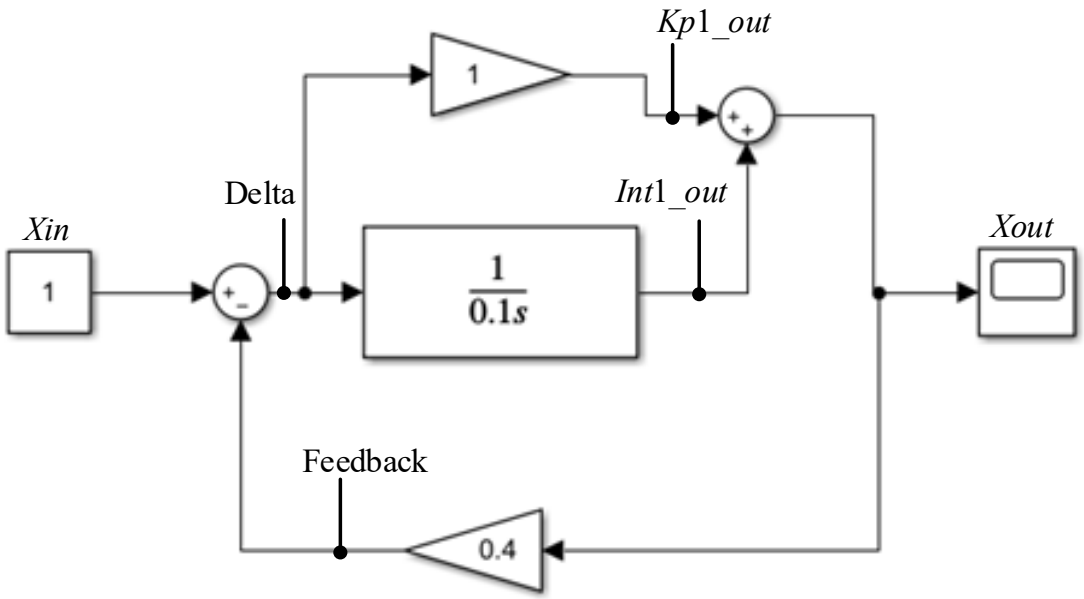


Рис. 6. Пример структурной схемы с обозначенными переменными

Обозначения на рис. 6:

- *Xin* – входной сигнал;
- *Feedback* – сигнал обратной связи;
- *Delta* – разница между входным сигналом и сигналом обратной связи;
- *Kp1_out* – выход пропорционального звена;
- *Int1_out* – выход интегрального звена;
- *Xout* – рассчитанный выходной сигнал.

При расчете структурных схем рекомендуется использовать следующие разностные уравнения, описывающие типовые звенья.

1. Инерционное звено – уравнение (6.2)

$$y_k = y_{k-1} + k_ф \cdot (x_{in} - y_{k-1}), \tag{6.2}$$

где y_k – значение выходного сигнала звена на текущем шаге расчёта; y_{k-1} – значение выходного сигнала звена на предыдущем шаге расчёта; x_{in} – значение входного сигнала; $k_ф$ – коэффициент инерционного звена.

2. Интегральное звено – уравнение (6.3)

$$y_k = y_{k-1} + k_u \cdot x_{in}, \tag{6.3}$$

где k_u – коэффициент интегрального звена.

Для схемы на рис. 6 $k_u = 1/(0{,}1 \cdot f)$, где f – заданная частота дискретизации (Гц), зависящая от частоты вызова прерывания, в котором производится расчёт. Например, при частоте прерывания 1 кГц, $f = 1000$, а следовательно, $k_u = 10/1000 = 0{,}01$.

В листинге 18 представлен пример расчёта заданной схемы (рис. 6).

Для расчёта структурной схемы рекомендуется выбрать подходящий формат дробных вычислений – float или IQ. Так же для расчёта интегральных и инерционных звеньев рекомендуется реализовать отдельные программные модули с соответствующими структурами.

Листинг 18. Пример расчёта заданной структурной схемы

```
Delta = Xin – Feedback;
Int1_out = Int1_out + ки * Delta; // Расчёт интегрального звена
Kp1_out = Delta * 1; // Расчёт пропорционального звена
Xout = Kp1_out + Int1_out; // Расчёт выходного сигнала
Feedback = Xout * 0,4; // Расчёт сигнала обратной связи
```

Вернуться к началу раздела

Вернуться к содержанию

ЭКЗАМЕНАЦИОННЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Общая характеристика параметров и набора периферийных устройств семейства специализированных микроконтроллеров управления двигателями фирмы Texas Instruments TMS320x28xx. Приведите пример рационального использования контроллеров этого типа с распределением задач между отдельными периферийными устройствами для управления автоматом по продаже шоколадок и снеков с выдачей товара из отдельных ячеек с номерами.

2. Каковы общие принципы организации гарвардской и модифицированной гарвардской архитектур сигнальных микроконтроллеров ‘C28xx? Назовите основные особенности и преимущества. Как организованы система шин адреса и данных и конвейер команд?

3. Какие типы встроенной памяти поддерживаются в процессорах семейства ‘C28xx? В каких областях памяти должен располагаться код программы, в каких данные? Каким образом обеспечивается защита программного обеспечения от несанкционированного копирования? Дайте понятие секций (неинициализированных переменных, стека, инициализированных переменных, кода программы).

4. Как в сигнальных микроконтроллерах ‘C28xxx организован стек? Зачем он нужен? Как используется при работе с подпрограммами и прерываниями?

5. Работа с указателями и массивами на языке Си. Назначение указателей. Инициализация, применение. Использование указателей для работы с массивами. Проиллюстрируйте на примере программного модуля очистки массива (массив из заданного количества элементов заполняется нулями)

6. Понятие функции в языке СИ. Технология передачи данных (параметров) в функцию и из нее обратно в вызывающую программу. Передача

непосредственных данных и указателей. Приведите пример любой простой функции и объясните способ ее описания и вызова.

7. Директивы и квалификаторы языка Си. "extern", "const", "volatile", "#include", "#define". Назначение и использование. Использования оптимизатора.

8. Реализация структур на языке Си. Исполняемые и заголовочные файлы. Инициализация переменных и функций внутри структуры, значения «по умолчанию». Покажите оформление структуры на примере простейшего программного модуля расчета ПИД-регулятора.

9. Базовые принципы модульной разработки программного обеспечения в среде Code Composer Studio. Основные понятия: модуль программы в исходном коде, в перемещаемом объектном коде, командный файл компоновщика, библиотечный файл, заголовочный файл, выходной исполняемый файл. Транслятор? Компоновщик? Что такое проект? Что входит в проект? Как должен быть оформлен программный модуль, если он вызывает процедуру, расположенную в другом модуле?

10. Таймеры общего назначения. Особенности применения и настройки. Использования таймеров общего назначения модулем ШИМ. Покажите на примере настройки центрированной ШИМ с несущей частотой 500 Гц: какие значения необходимо записать в регистры периода и делителя частоты?

11. Система прерываний. Какие возможности у контроллера периферийных прерываний? Что такое таблица векторов периферийных прерываний? Укажите последовательность операций при обработке прерываний, возможные источники прерываний.

12. Организация дискретного ввода/вывода микроконтроллеров TMS320xx28xx. Особенности применения и настройки. Покажите на примере программного модуля, который принимает и обрабатывает сигнал с внешней кнопки (GPIOA0) и по её нажатию «зажигает» светодиод (GPIOA1).

13. Назначение, принцип действия, режимы работы и примеры рационального использования в электроприводе каналов сравнения процессора событий. Какие возможности по организации каналов сравнения имеет менеджер событий микроконтроллеров TMS320xx28xx? Как организованы и применяются модули сравнения таймеров общего назначения, 3-канальный модуль полного сравнения?

14. Предложите технологию использования модуля захвата менеджера событий для прямой цифровой обработки сигналов с датчика положения ротора на трех элементах Холла. Какова структура программного обеспечения, какие функции предполагается выполнить в основной программе (фоновой), какие в процедурах обслуживания прерываний? К чему может привести несимметрия расположения датчиков?

15. Что такое модуль захвата менеджера событий? Назначение, принцип действия. Проиллюстрируйте технологию использования модуля захвата на примере задачи идентификации скорости привода по сигналу импульсного датчика положения, установленного на вал двигателя. Датчик имеет один канал и разрешение 256 меток на оборот.

16. Квадратурный декодер для сопряжения с импульсными датчиками положения. Какие основные функции, настройки? Назовите способы оценки положения и скорости.

17. Какие из периферийных устройств микроконтроллера ‘C28 можно использовать для прямого цифрового управления транзисторным преобразователем напряжения в цепи обмотки возбуждения двигателя постоянного тока? Нарисуйте структуру силовой части и структуру системы управления для реализации алгоритма ослабления магнитного поля системы с двузонным регулированием. Из каких функциональных блоков будет состоять программа управления (фоновая и обслуживания прерываний)?

18. Что такое векторная ШИМ применительно к мостовому инвертору напряжения для управления асинхронными двигателями от преобразователей частоты? Какой эффект дает применение этого метода управления по сравнению с обычной синусоидальной центрированной ШИМ? Что означает привязка к верхней или нижней шине?

19. Какие вам известны типы АЦП. Каковы принцы их реализации? Какие преимущества, недостатки у разных типов АЦП?

20. Как организован интерфейс передачи данных SPI? Каков принцип действия, вид посылки, области применения?

21. Как организован интерфейс передачи данных CAN на физическом уровне? Как организована линия передачи данных, каков формат посылок?

22. Как организован интерфейс передачи данных RS-485? Как организована линия передачи данных, каков формат посылок?

23. Каково назначение библиотеки IQMath? Как использовать функции библиотеки для решения задач цифрового управления приводами? В чем состоит основное преимущество этого подхода? Почему, например, не воспользоваться альтернативными функциями библиотеки с плавающей точкой? Продемонстрируйте технологию применения средств IQMath на любом примере.

24. Реализация ПИ-регуляторов и инерционных фильтров на языке Си. Приведите пример программного кода для моделирования заданной структуры.

25. Реализация типовых звеньев САУ (пропорционального, интегрального, дифференциального и инерционного) на языке Си. Приведите пример программного кода для моделирования заданной структуры.

Вернуться к началу раздела

Вернуться к содержанию

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Микропроцессорные средства в электроприводе на базе микроконтроллеров TMS320F28035: лабораторный практикум / Д. И. Савкин, Д. М. Шпак, А. С. Анучин и др. – М.: Издательство МЭИ, 2019. – 1 электрон. опт. диск CD-ROM.

2. Описание и техническая документация на среду разработки Code Composer Studio. URL: http://software-dl.ti.com/ccs/esd/documents/ccs_documentation-overview.html

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Онищенко, Г. Б. SimInTech: Системы управления и моделирования электропривода: учебное пособие / Г. Б. Онищенко, Ю. Н. Калачев. – М.: ДМК Пресс, 2022.

2. Перри, Грег. Программирование на Си для начинающих / Грег Перри, Дин Миллер. – 3-е изд. – М.: Эксмо, 2015.

3. Керниган, Брайан. Язык программирования Си / Брайан Керниган, Деннис Ритчи. – 2-е изд., перераб. и доп. – М.: Диалектика, 2020.

Приложение 1

Требования к оформлению отчётов по индивидуальным заданиям

Файл с шаблоном для оформления курсовых проектов/работ представлен по ссылке на QR–коде. Далее в Приложении представлены пояснения по правилам оформления, применяемым в шаблоне.

Возврат к содержанию

П1.1. ОФОРМЛЕНИЕ ТЕКСТА

Оформление заголовков

Заголовок раздела должен быть оформлен стилем «Заголовок 1». Для применения стиля нужно выделить текст заголовка и выбрать в разделе «Стили» стиль «Заголовок 1», как показано на рисунке П1.1. Параметры стиля: Times New Roman, 14 pt, полужирный, без отступов и выступов, выравнивание по левому краю.

Нумерация проставится автоматически, когда вы примените стиль.

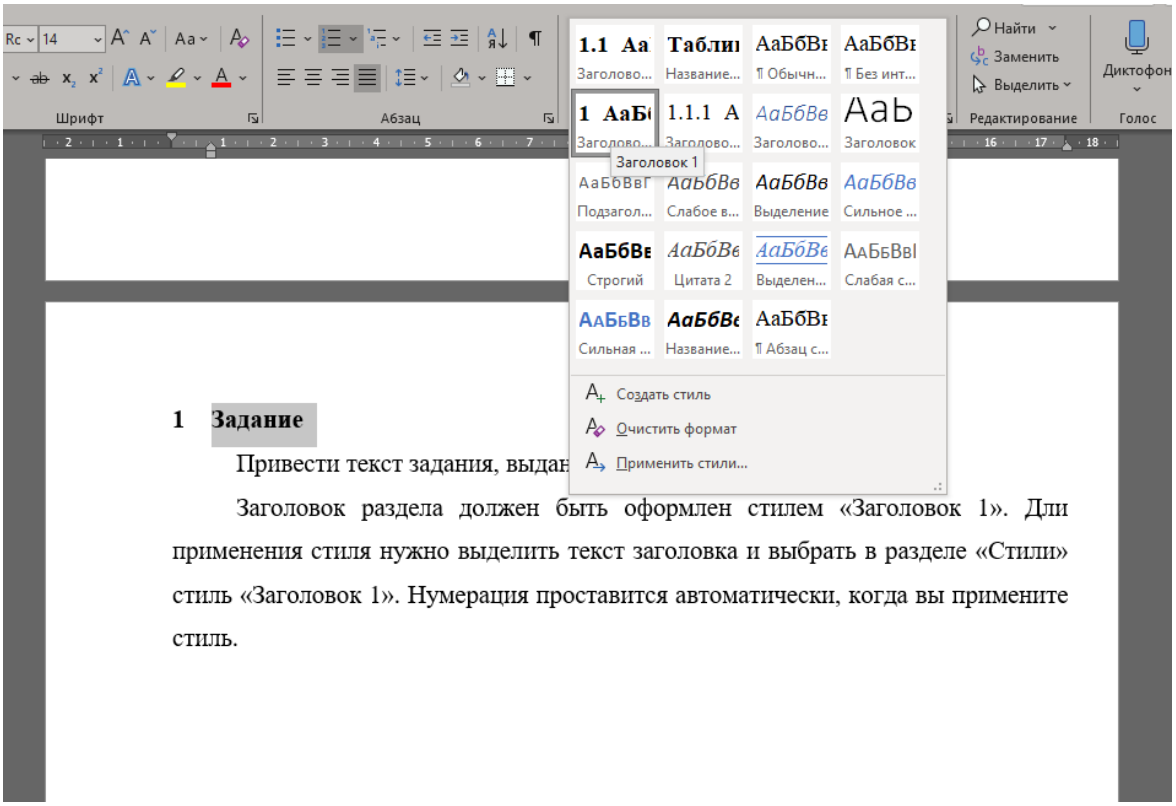


Рисунок П1.1 – Применение стиля к тексту

Заголовки второго уровня

Если требуется создать вложенный раздел (1.1, 1.2, 1.3 и т.д.), следует аналогичным образом применить к тексту заголовка стиль «Заголовок 2». Параметры стиля: Times New Roman, 14 pt, без отступов и выступов, выравнивание по левому краю.

Использовать более глубокую вложенность (1.1.1, 1.2.1.1 и т.д.) не рекомендуется.

Оформление текста

Оформлять текст следует также с использованием стилей – для этого нужно использовать стиль «Обычный». Параметры стиля: Times New Roman, 14 pt, отступ первой строки абзаца на 1,25 см, междустрочный интервал 1,5 строки, выравнивание по всей ширине.

П1.2. ОФОРМЛЕНИЕ РИСУНКОВ

Требования к содержимому рисунков

Если на рисунке (скриншоте) есть какие-либо надписи или числовые значения, то они должны быть читаемы. Все осциллограммы должны быть оформлены так, чтобы линии не сливались с фоном, т. е. линии должны иметь контрастный цвет и достаточную толщину.

Рисунок должен быть выровнен по центру страницы, без отступов/выступов.

Пример неправильно оформленного рисунка приведён на рис. П1.2, а пример правильно оформленного – на рисунке П1.3.

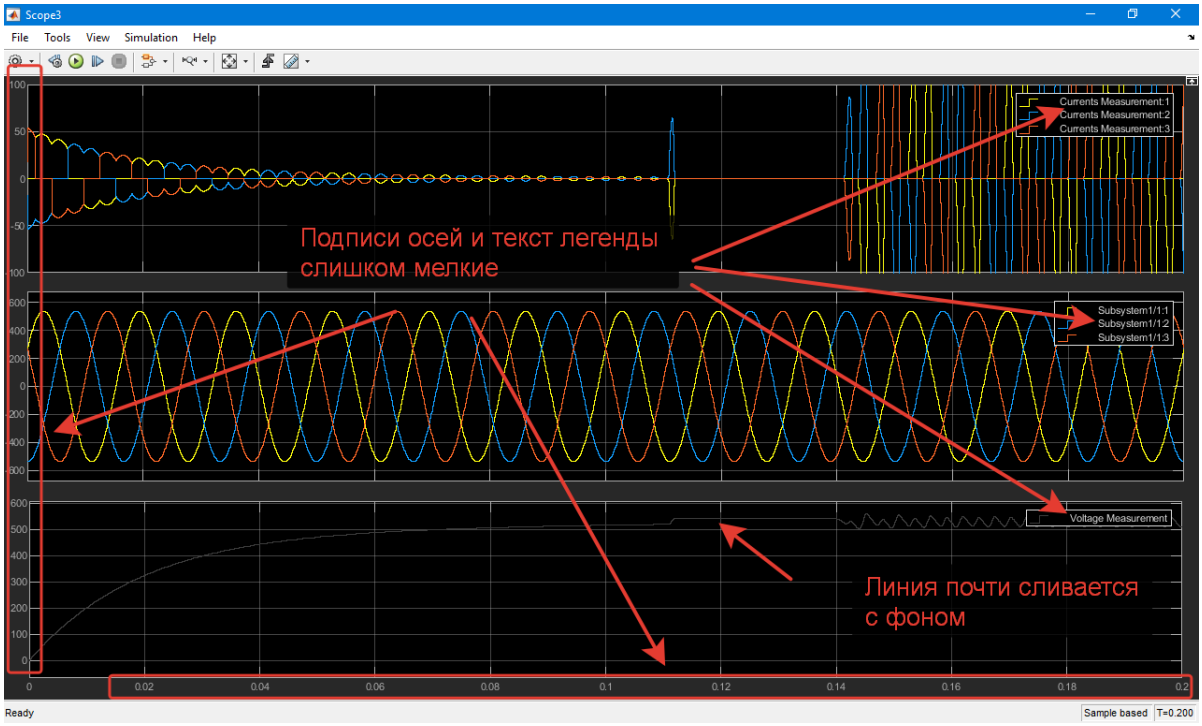


Рисунок П1.2 – Неправильно оформленный рисунок

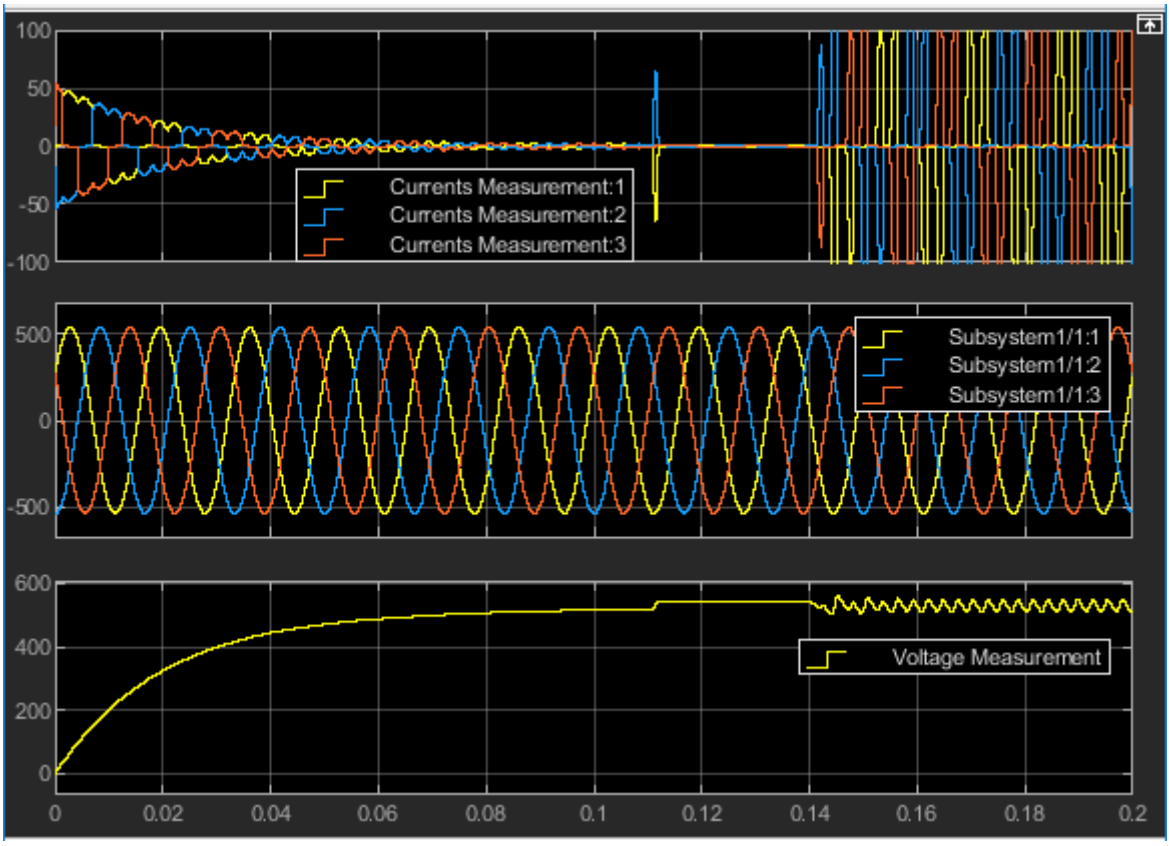


Рисунок П1.3 – Правильно оформленный рисунок

Оформление названий рисунков

Под каждым рисунком должна присутствовать подпись «Рисунок N (или Рис. N) – Название рисунка».

Название может быть сделано вручную; в таком случае к нему следует применить стиль «Название объекта». Параметры стиля: Times New Roman, 14pt, без отступов и выступов, выравнивание по центру.

Также можно оформлять название с использованием функции «Вставить название» из меню «Ссылки», как это показано на рис. П1.4. В таком случае номер рисунка будет проставлен автоматически.

Нумерация рисунков может быть сквозной («Рисунок 1/2/3…»), или включать номер раздела, если рисунков много («Рисунок 1.1/1.2/2.1/2.2 …»).

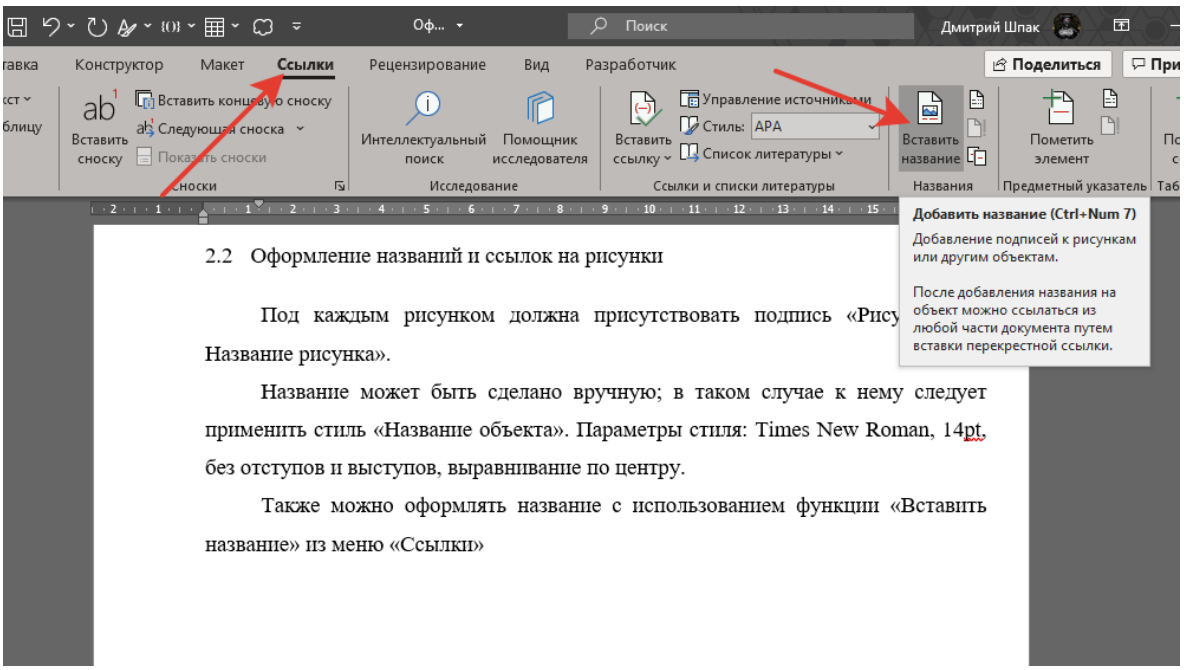


Рисунок П1.4 – Оформление названия рисунка

Ссылки на рисунки

На **каждый** рисунок в тексте должна быть ссылка с пояснением, что изображено на рисунке. Например, «На рисунке 4 показаны осциллограммы тока двигателя при работе в режиме…».

Ссылаться на рисунок следует до того, как он встречается в тексте: т. е. сначала пишется, что на каком-то рисунке что-то изображено, а затем вставляется сам рисунок.

П1.3. ОФОРМЛЕНИЕ КОДА ПРОГРАММЫ

Требования к оформлению листингов

Код программы должен оформляться в виде листингов. Для создания листинга следует вставить таблицу размера 1x1.

Если текст внутри листинга скопирован из среды моделирования, то его следует вставлять в режиме «Сохранить форматирование», чтобы сохранить цвета и настройки шрифтов, как это показано в листинге П1.1.

Иначе к содержимому листинга необходимо применить стиль «Код», как это показано в листинге П1.2. Параметры стиля: Consolas, 10pt, без отступов/выступов, междустрочный интервал 1.

Каждый листинг должен иметь название «Листинг N – Название листинга». Название должно располагаться над листингом, применяемый стиль – обычный.

Листинг П1.1. Текст, скопированный из среды моделирования с сохранением форматирования

```
int main(void) {
	int i = 0;
	while (1) {
		i = i + 2;
	}
}
```

Листинг П1.2. Текст, набранный вручную

```
   FLASHH      : origin = 0x3E8000, length = 0x002000     /* on-chip FLASH */
   FLASHE      : origin = 0x3EE000, length = 0x002000     /* on-chip FLASH */
   FLASHC      : origin = 0x3F2000, length = 0x002000     /* on-chip FLASH */
   FLASHA      : origin = 0x3F6000, length = 0x001F80     /* on-chip FLASH */
```

П1.4. ОФОРМЛЕНИЕ ТАБЛИЦ

Содержимое таблиц

Текст ячеек таблиц должен быть оформлен без отступов/выступов, с междустрочным интервалом 1.

Для этого можно использовать стиль «Текст таблицы». Названия столбцов таблиц необходимо дополнительно выровнять по центру. В зависимости от содержания ячеек, допускается выравнивать их либо по левому краю, либо по центру. При необходимости допускается уменьшать размер текста в ячейках до 12pt. Пример оформления таблицы приведён в таблице П1.1.

Таблица П1.1. Пример оформления таблицы

Имя файла	Размер файла	Дата создания файла
funcs.c	2,3 кБ	11.09.20
user_data.c	4.0 кБ	11.09.20
changes.h	1,7 кБ	11.09.20

П1.5. ССЫЛКИ НА ЛИСТИНГИ И ТАБЛИЦЫ

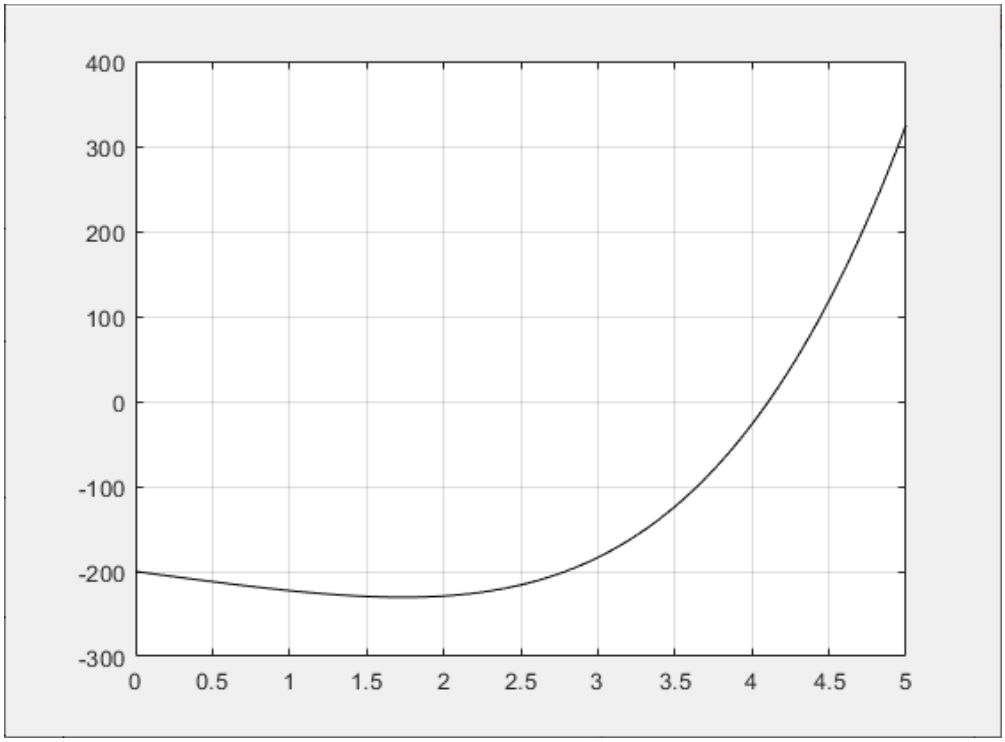
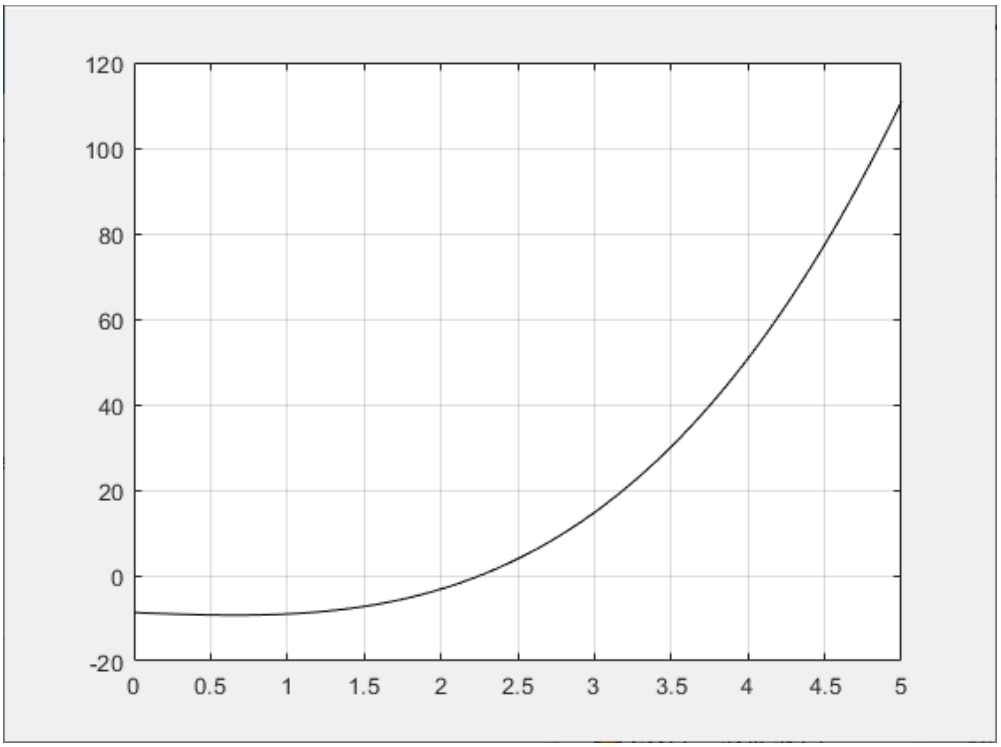
Так же, как и в случае с рисунками, на каждый листинг должна присутствовать ссылка в тексте, и эта ссылка должна предшествовать появлению листинга в документе. К ссылкам на таблицы предъявляются те же требования, что и к ссылкам на рисунки и листинги.

Приложение 2

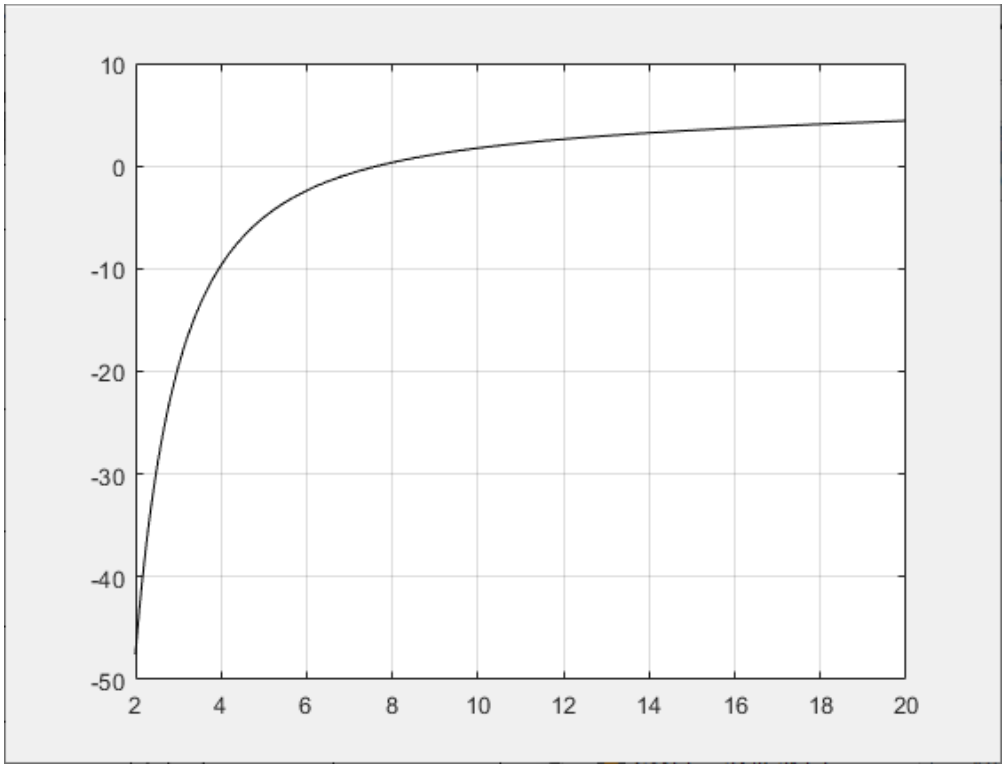
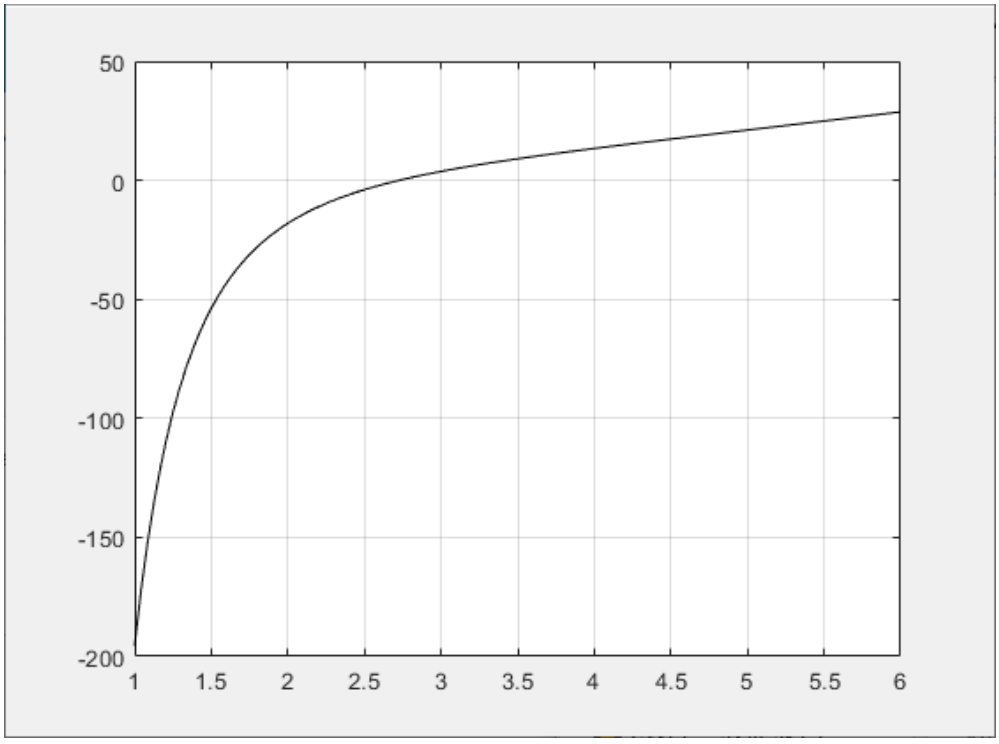
Графики функций к заданию 4.2.

Таблица П2.1

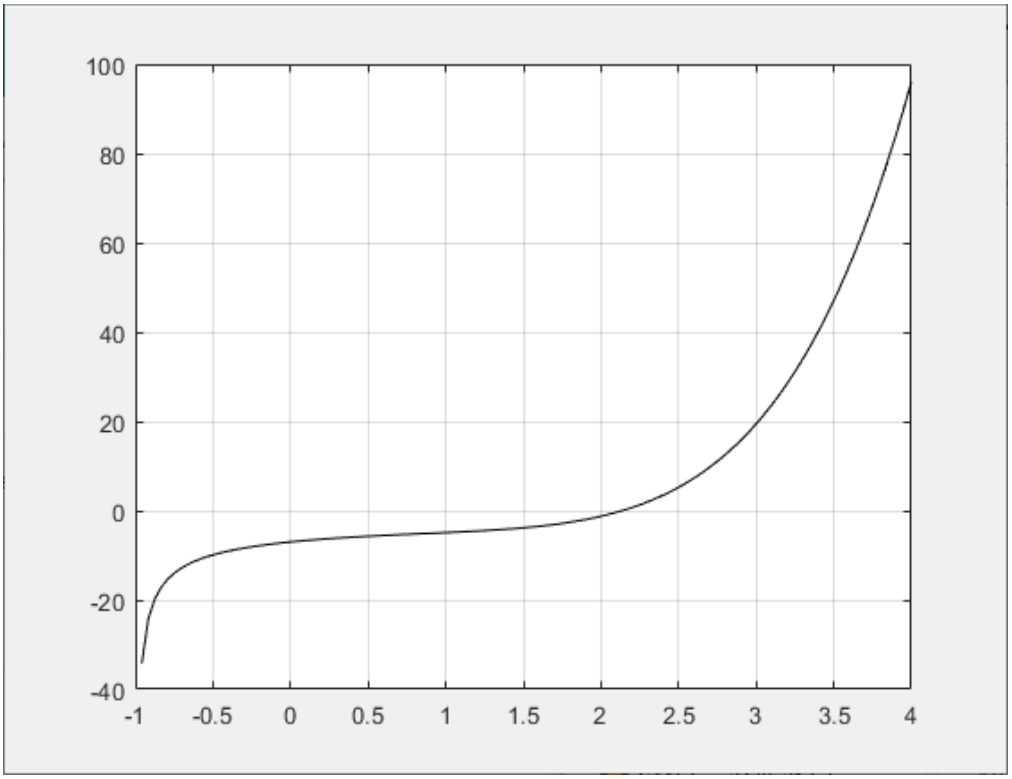
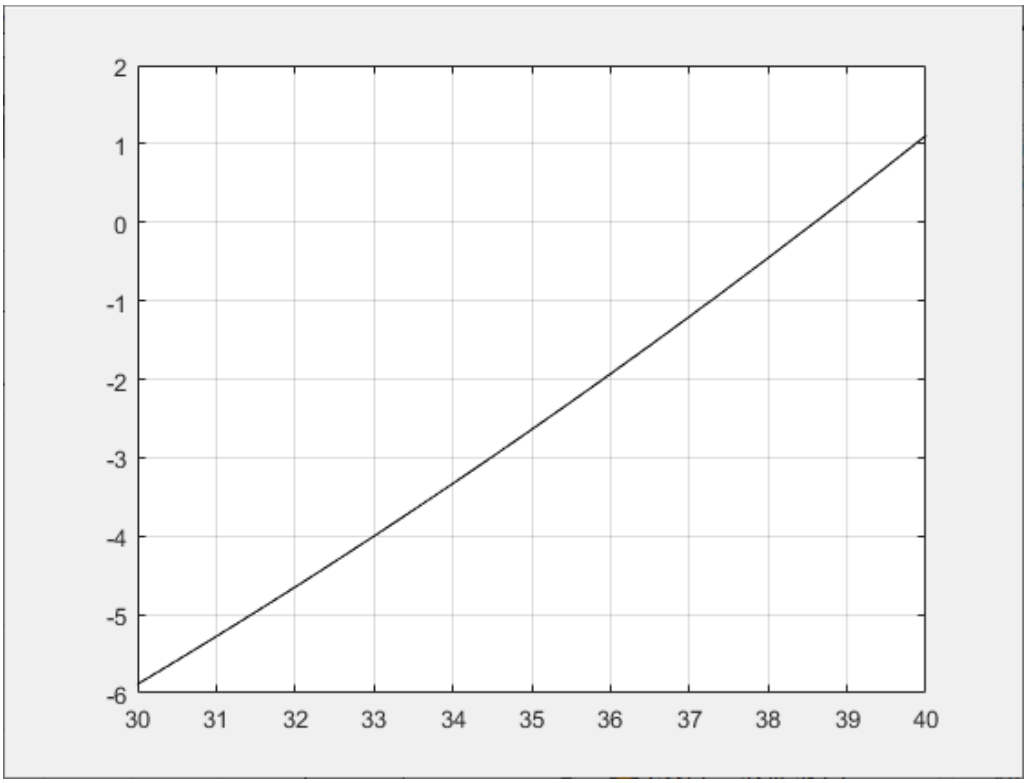
Варианты задания 4.2

1	
2	

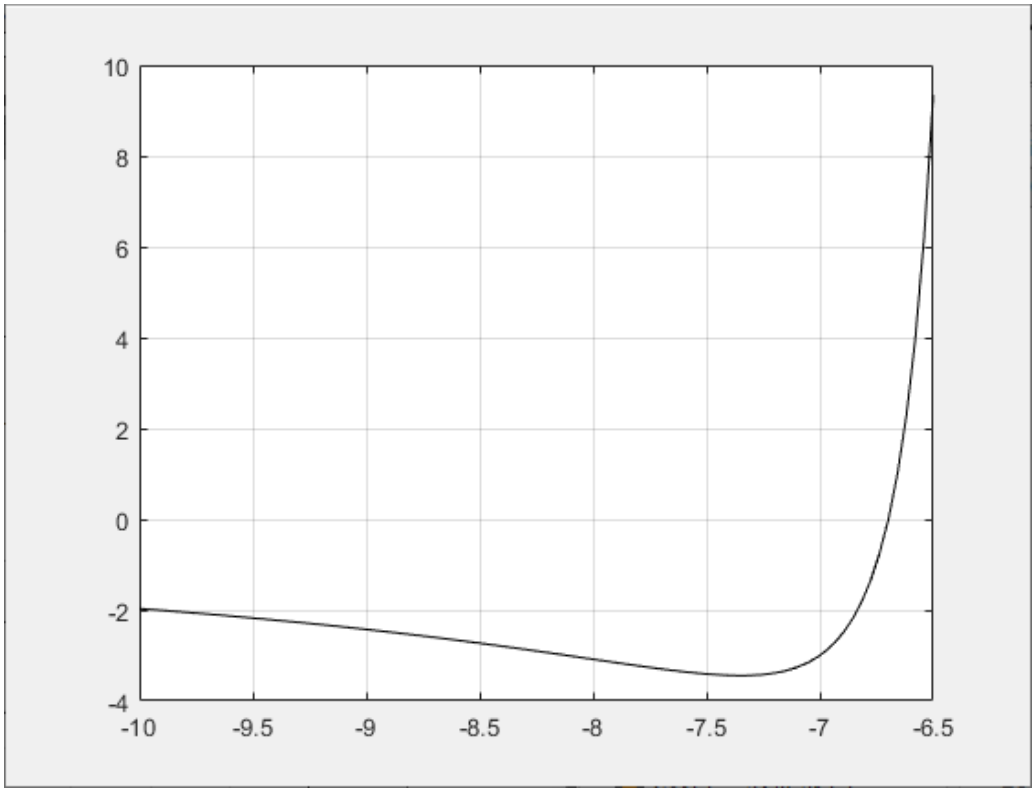
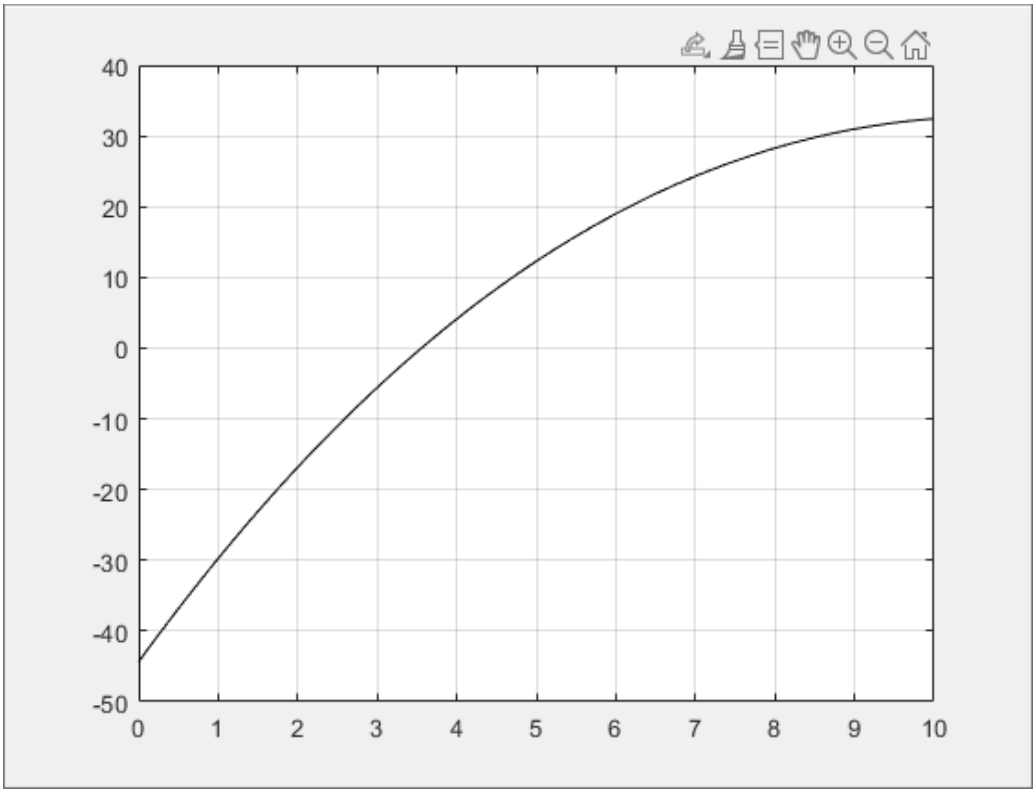
Продолжение таблицы П2.1

3	
4	

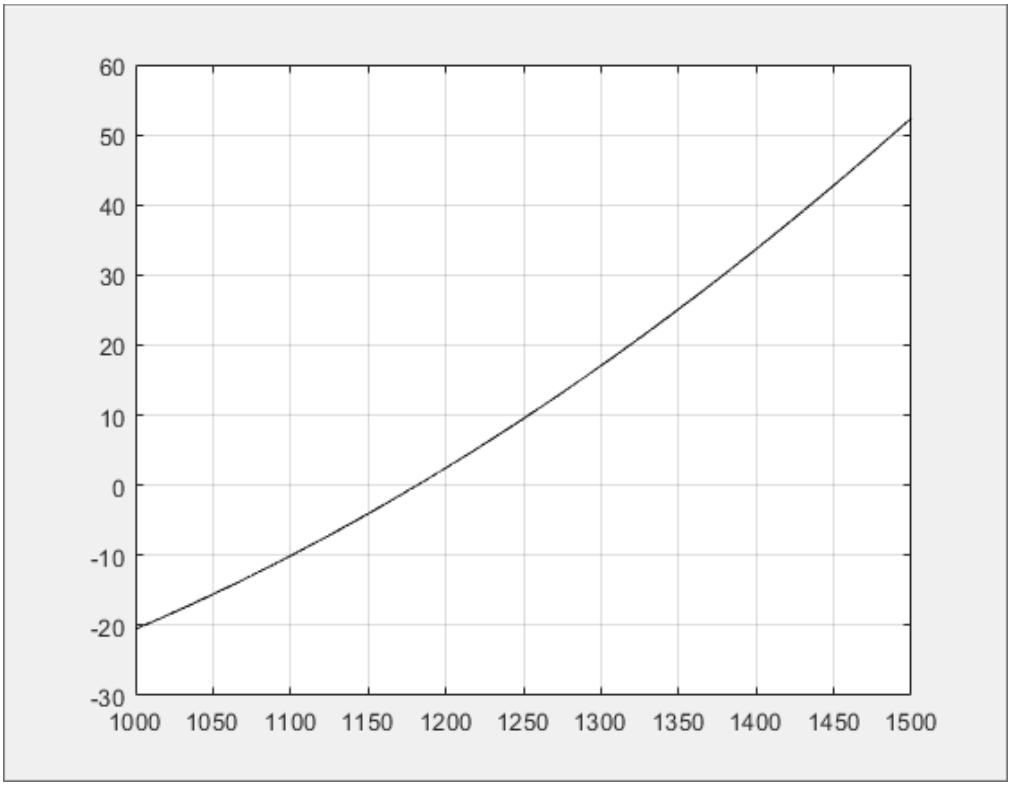
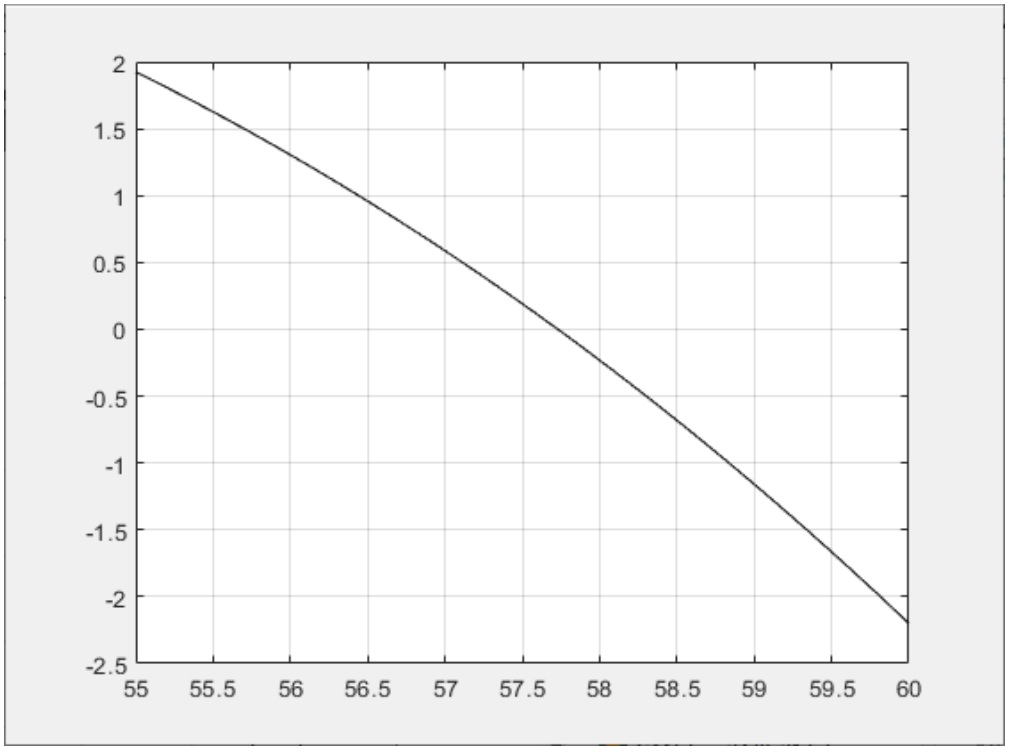
Продолжение таблицы П2.1

5	
6	

Продолжение таблицы П2.1

7	
8	

Окончание таблицы П2.1

9	
10	

Возврат к заданию 4.2

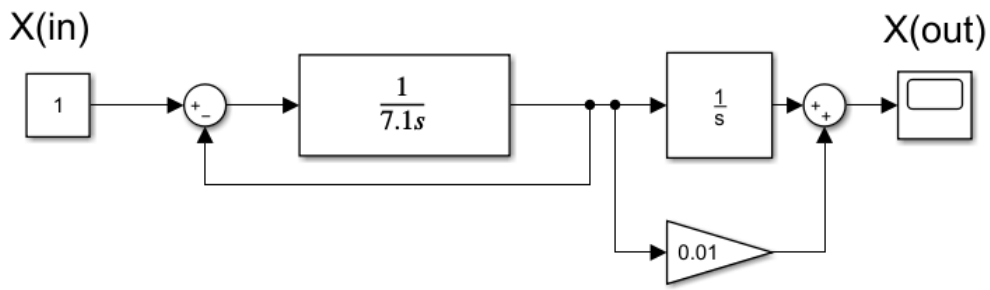
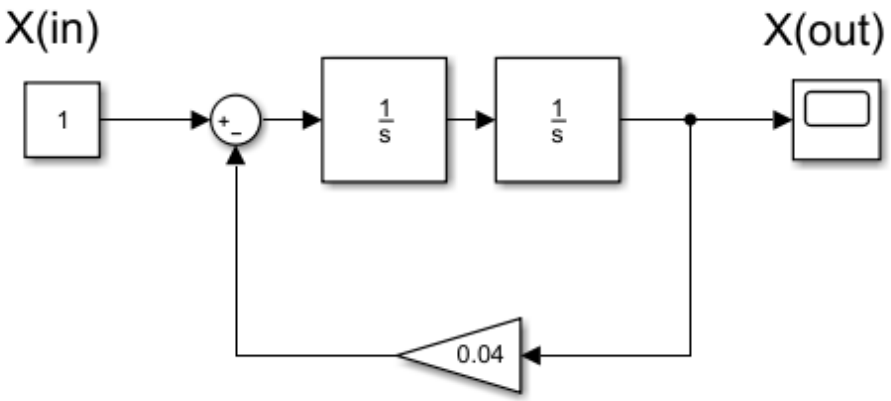
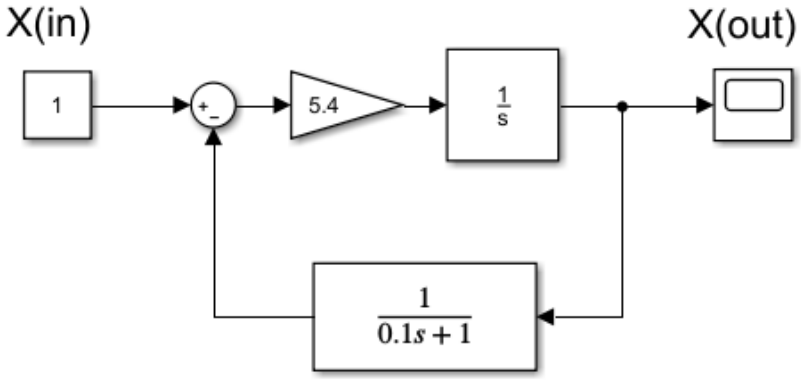
Возврат к содержанию

Приложение 3

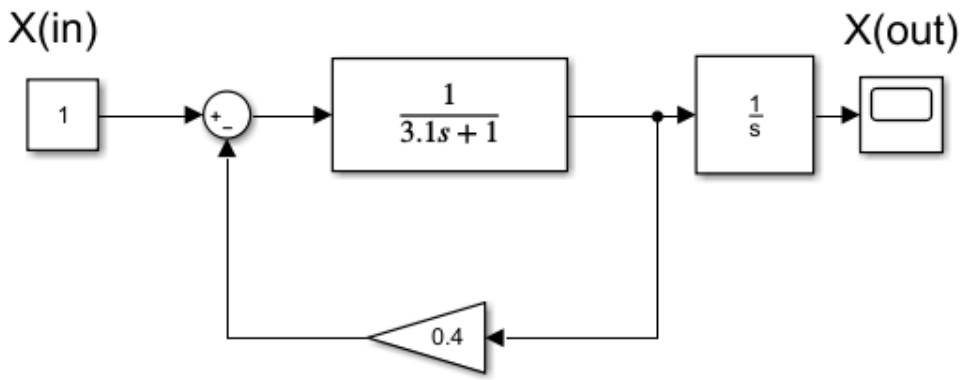
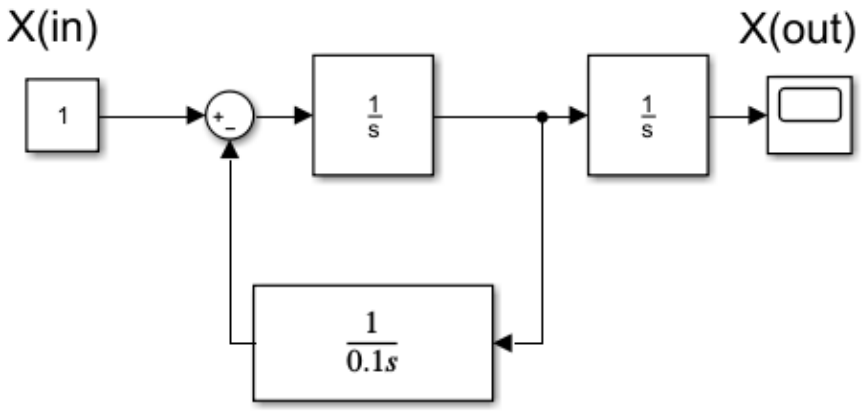
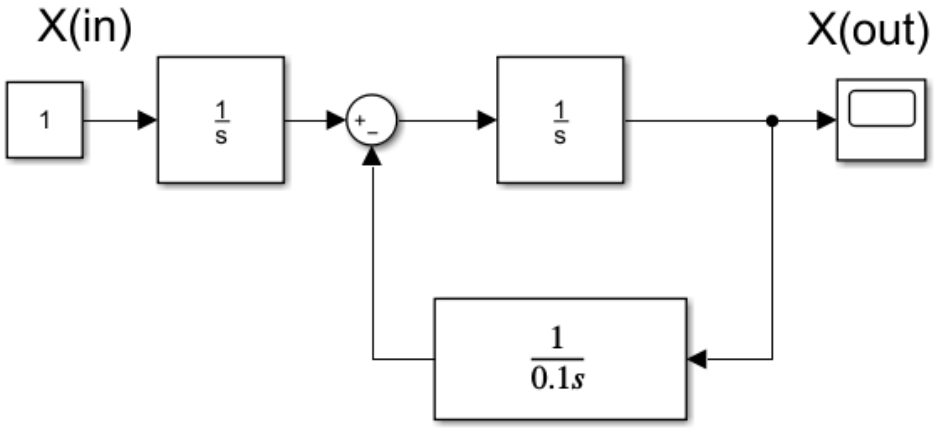
Варианты структурных схем к заданию 6.2.

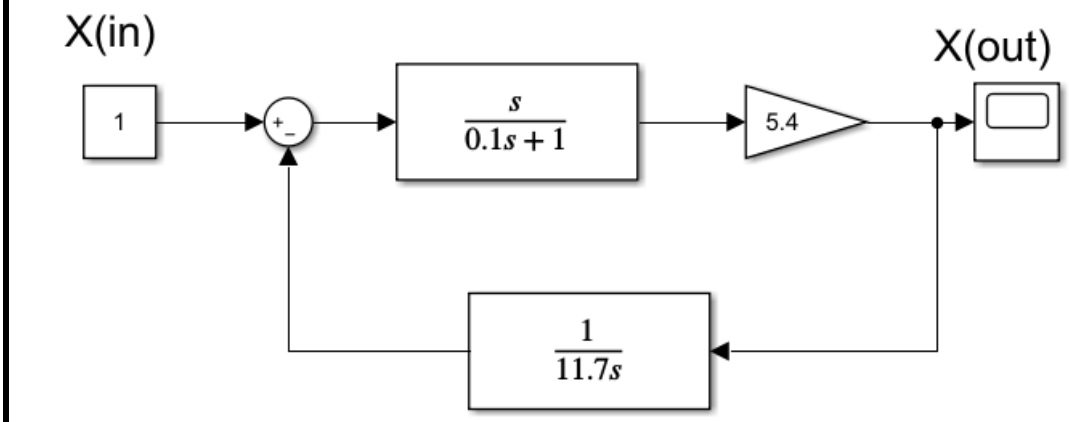
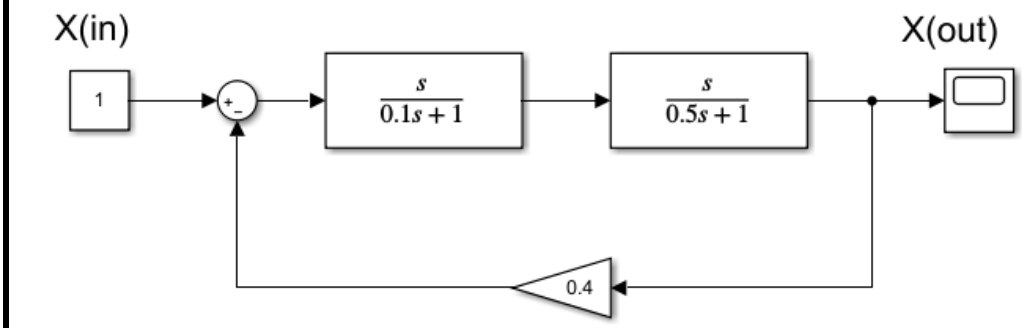
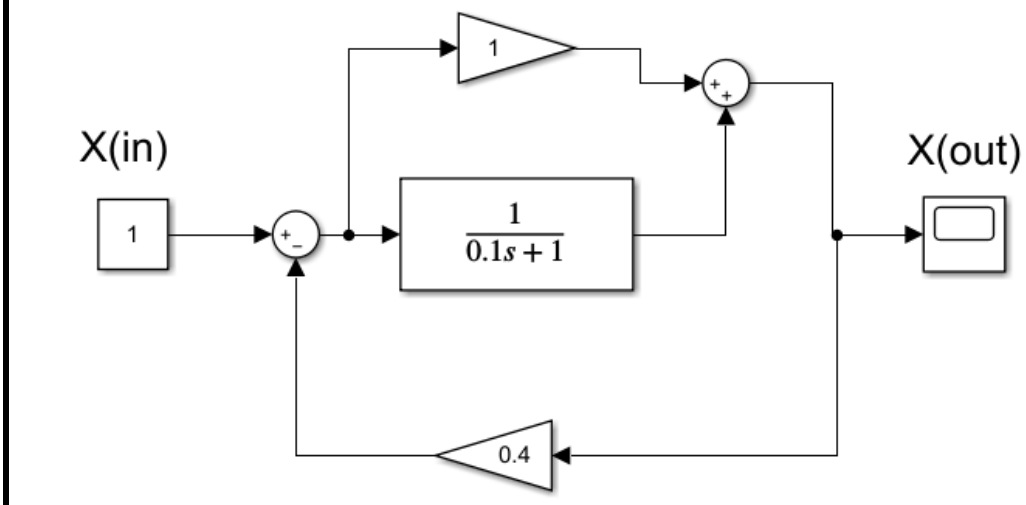
Таблица П3.1

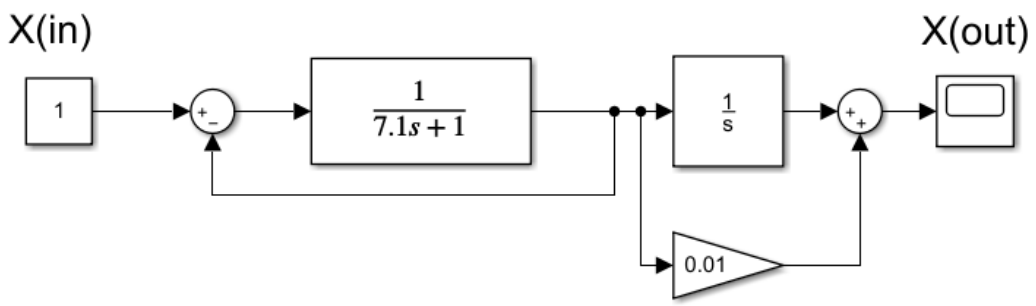
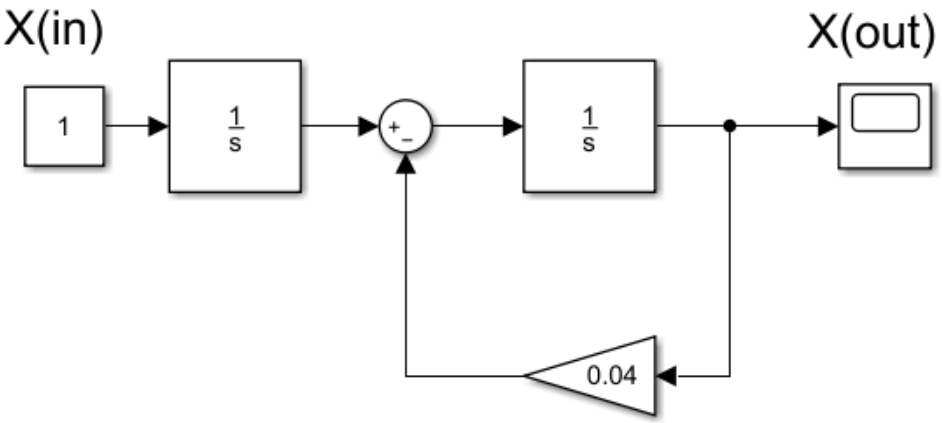
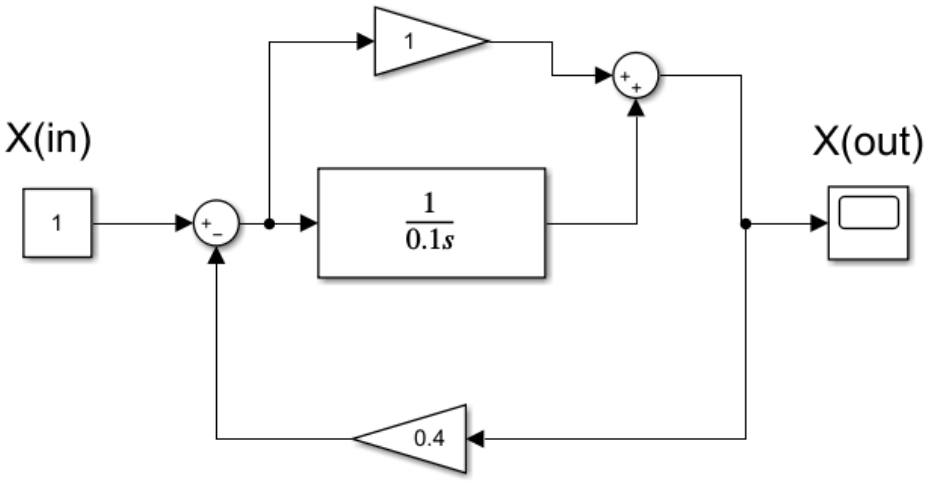
Варианты задания 6.2

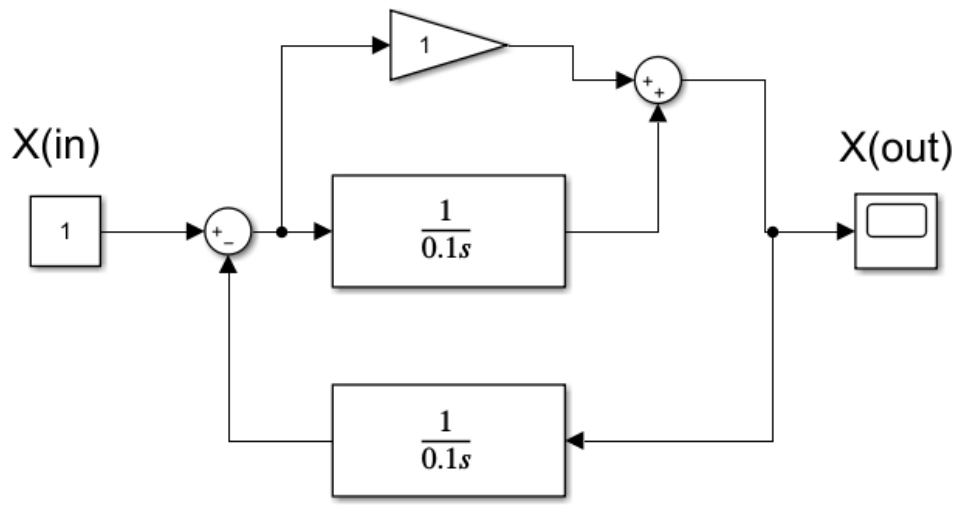
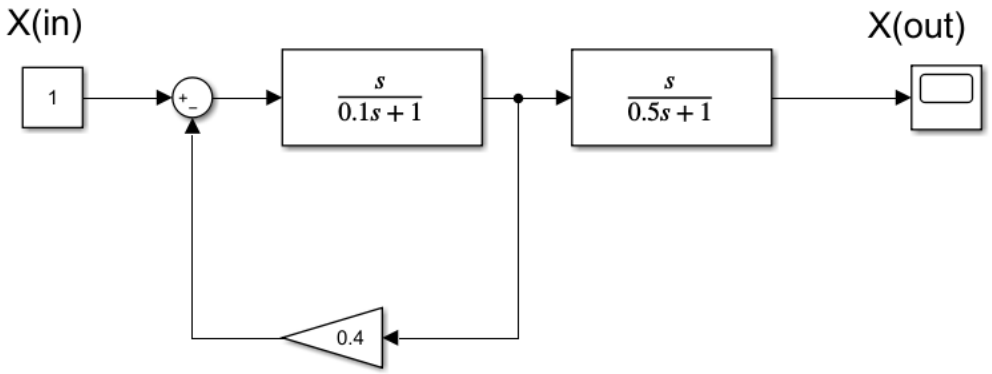
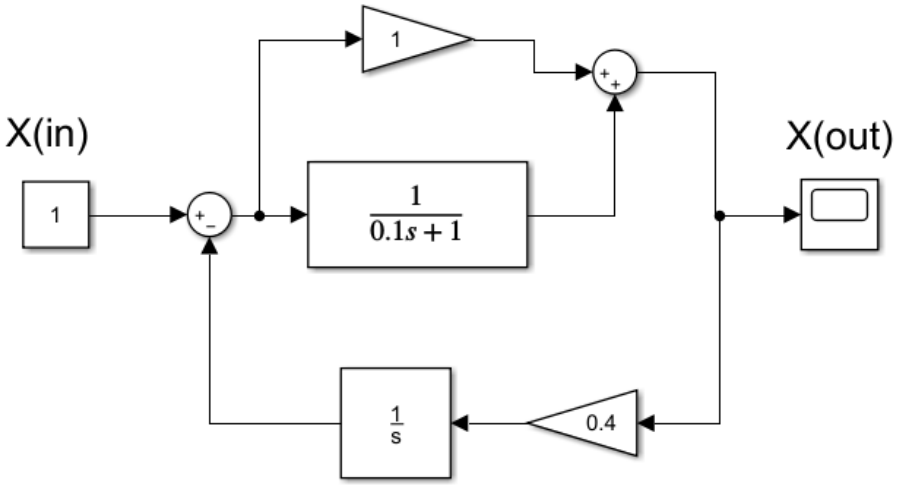
1	
2	
3	

Продолжение таблицы П3.1

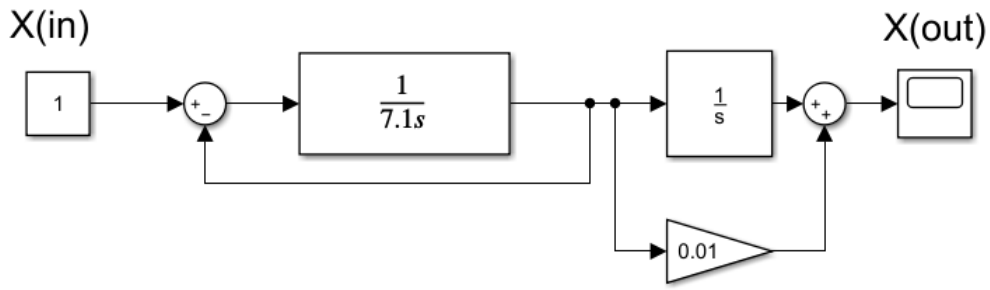
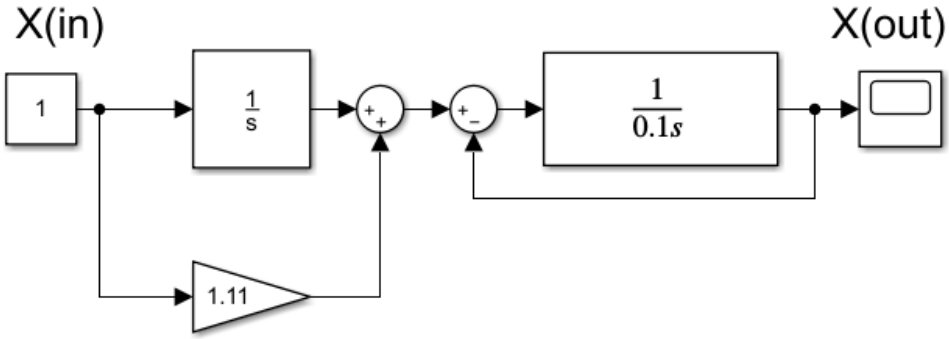
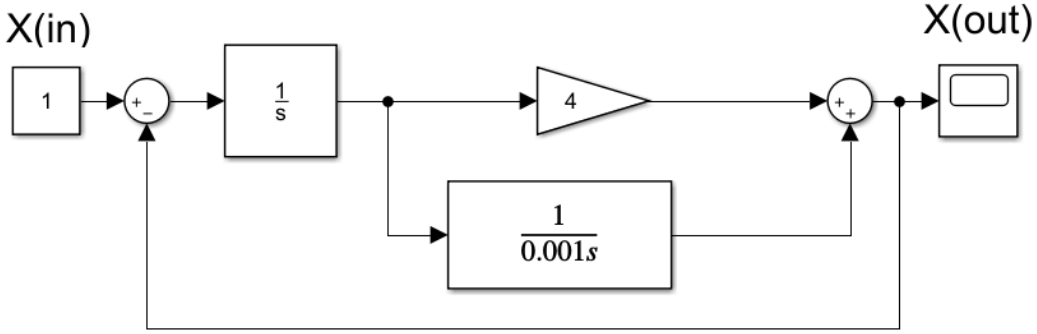
4	
5	
6	

7	 X(in) 1 $\frac{s}{0.1s+1}$ 5.4 X(out) $\frac{1}{11.7s}$
8	 X(in) 1 $\frac{s}{0.1s+1}$ $\frac{s}{0.5s+1}$ X(out) 0.4
9	 1 X(in) 1 $\frac{1}{0.1s+1}$ X(out) 0.4

10	 X(in) X(out) 1 $\frac{1}{7.1s+1}$ $\frac{1}{s}$ 0.01
11	 X(in) X(out) 1 $\frac{1}{s}$ $\frac{1}{s}$ 0.04
12	 1 X(in) X(out) 1 $\frac{1}{0.1s}$ 0.4

13	 X(in) 1 1 $\frac{1}{0.1s}$ X(out) $\frac{1}{0.1s}$
14	 X(in) 1 $\frac{s}{0.1s+1}$ $\frac{s}{0.5s+1}$ X(out) 0.4
15	 X(in) 1 1 $\frac{1}{0.1s+1}$ X(out) $\frac{1}{s}$ 0.4

Окончание таблицы П3.1

16	
17	
18	

Возврат к заданию 6.2

Возврат к содержанию

Приложение 4

Учебное издание «МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА В ЭЛЕКТРОПРИВОДЕ НА БАЗЕ МИКРОКОНТРОЛЛЕРОВ TMS320F28035: ЛАБОРАТОРНЫЙ ПРАКТИКУМ» доступно для скачивания по ссылки на QR-коде ниже.

Возврат к содержанию

Учебное электронное издание

Савкин Дмитрий Игоревич
Шпак Дмитрий Михайлович

Редактор Е. Б. Бурдюкова

При разработке практического электронного издания были использованы: пакет программ Microsoft Office с предустановленным компонентом MathType, векторный графический редактор Visio, а также web-технологии HTML, pdf.

Дата подписания – 16.12.2024
Объём издания – 2,01 МБ
Тираж – 10 электронных оптических дисков CD-ROM

Национальный исследовательский университет «МЭИ»
111250, Москва, Красноказарменная, д. 14, стр.1