

# ОСНОВЫ МИКРО- ПРОЦЕССОРНОЙ ТЕХНИ- КИ

Руководство по выполнению  
базовых экспериментов

ОМТ.001 РБЭ (920.5)



*Гибкие решения для учебных лабораторий*

Чикота В.В. Основы микропроцессорной техники. Руководство по выполнению базовых экспериментов. ОМТ1 РБЭ (920.5). Под ред. П.Н. Сенигова. – Челябинск: ООО «ИПЦ «Учебная техника», 2014. – 92 с.

Представлены перечни используемой при выполнении базовых экспериментов аппаратуры ГалСен<sup>®</sup>, электрические схемы соединений и их описания, а также указания по проведению базовых экспериментов.

Руководство предназначено для использования при подготовке к проведению лабораторных работ по учебной дисциплине “Основы микропроцессорной техники” и смежным с ней дисциплинам в высших, средних и начальных профессиональных образовательных учреждениях.

© 2014 ООО «ИПЦ «Учебная техника» (г. Челябинск, РФ). Предприятие-изготовитель оставляет за собой право вносить изменения в технические и функциональные характеристики, а также в потребительские свойства и внешний вид представленного оборудования без предварительного уведомления. Никакая часть данного издания не может быть воспроизведена в какой бы то ни было форме без письменного разрешения правообладателя. Правообладателями зарегистрированных товарных знаков являются ООО ИПЦ «Учебная техника» (свидетельство №318023 от 12 декабря 2006 г.), ООО «Учебная техника-ГалСен» (свидетельство №587588 от 20 сентября 2016 г.). Все иные товарные знаки и зарегистрированные товарные знаки, логотипы и иные объекты интеллектуальной собственности, упомянутые в издании, принадлежат их владельцам. Официальный сайт предприятия-изготовителя и правообладателя: [WWW.GALSEN.RU](http://WWW.GALSEN.RU).

# Содержание

<b>ВВЕДЕНИЕ.....</b>	<b>1</b>
<b>1. ИЗУЧЕНИЕ БАЗОВЫХ ВОЗМОЖНОСТЕЙ ОБОРУДОВАНИЯ И СРЕДЫ ПРОГРАММИРОВАНИЯ.....</b>	<b>2</b>
1.1. Изучение программной оболочки AtmelStudio (мой первый проект).....	3
<b>2. ПРОГРАММИРОВАНИЕ ТАЙМЕРОВ МИКРОКОНТРОЛЛЕРА.....</b>	<b>17</b>
2.1. Формирование выдержки времени с помощью таймера .....	18
2.2. Формирование сигнала заданной частоты .....	23
2.3. Определение длительности внешних сигналов с помощью таймеров.....	27
2.4. Счетчик с программируемым коэффициентом деления на базе таймера.....	33
<b>3. ИСПОЛЬЗОВАНИЕ ПРЕРЫВАНИЙ ПРИ ПРОГРАММИРОВАНИИ МИКРОКОНТРОЛЛЕРА.....</b>	<b>38</b>
<b>4. ПРОГРАММИРОВАНИЕ АЦП МИКРОКОНТРОЛЛЕРА .....</b>	<b>45</b>
<b>5. ЦАП НА БАЗЕ ШИМ СИГНАЛОВ МИКРОКОНТРОЛЛЕРА .....</b>	<b>56</b>
<b>6. ПЕРЕДАЧА ДАННЫХ ПО ПОСЛЕДОВАТЕЛЬНЫМ КАНАЛАМ СВЯЗИ.....</b>	<b>62</b>
6.1. Передача данных с использованием канала SPI.....	63
6.2. Передача данных с использованием канала USART .....	71
<b>7. ИСПОЛЬЗОВАНИЕ МИКРОКОНТРОЛЛЕРА В ПРИКЛАДНЫХ ЗАДАЧАХ.....</b>	<b>81</b>
7.1. Секундомер .....	82
7.2. Термометр .....	88

## Введение

В настоящем руководстве описаны базовые эксперименты, выполняемые с использованием набора миниблоков «Микроконтроллеры» (600.25) и комплекте типового лабораторного оборудования «Основы цифровой техники». Набор миниблоков позволяет изучить программирование и использование микроконтроллеров Atmel для решения различных задач.

Комплект предназначен для проведения лабораторных работ в учреждениях начального, среднего и высшего профессионального образования.

Комплект может быть также использован на семинарах и курсах повышения квалификации электротехнического персонала предприятий и организаций.

Аппаратная часть комплекта выполнена по блочному (модульному) принципу и состоит:

- спроектированные с учебными целями блок испытания цифровых устройств (БИЦУ) содержащий в своем составе элементы коммутации и индикации, генераторы прямоугольных импульсов различной частоты и наборное поле, для размещения миниблоков и обеспечения их напряжением питания. На схемах соединений часть блока БИЦУ, содержащая наборное поле не показывается;

- набор миниблоков с логическими элементами (600.11);

- набор миниблоков «Микроконтроллеры» содержащий в своем составе:

- 1) миниблок микроконтроллера AT90usb162;

- 2) миниблок микроконтроллера Atxmega16A4U;

- 3) миниблок потенциометра;

- 4) миниблок набора резисторов и конденсаторов.

- мультиметр для измерения уровня сигналов;

- однофазный источник питания;

- настольную раму для установки необходимых в экспериментах функциональных блоков.

Питание комплекта осуществляется от однофазной электрической сети напряжением 220 В с нейтральным и защитным проводниками.

---

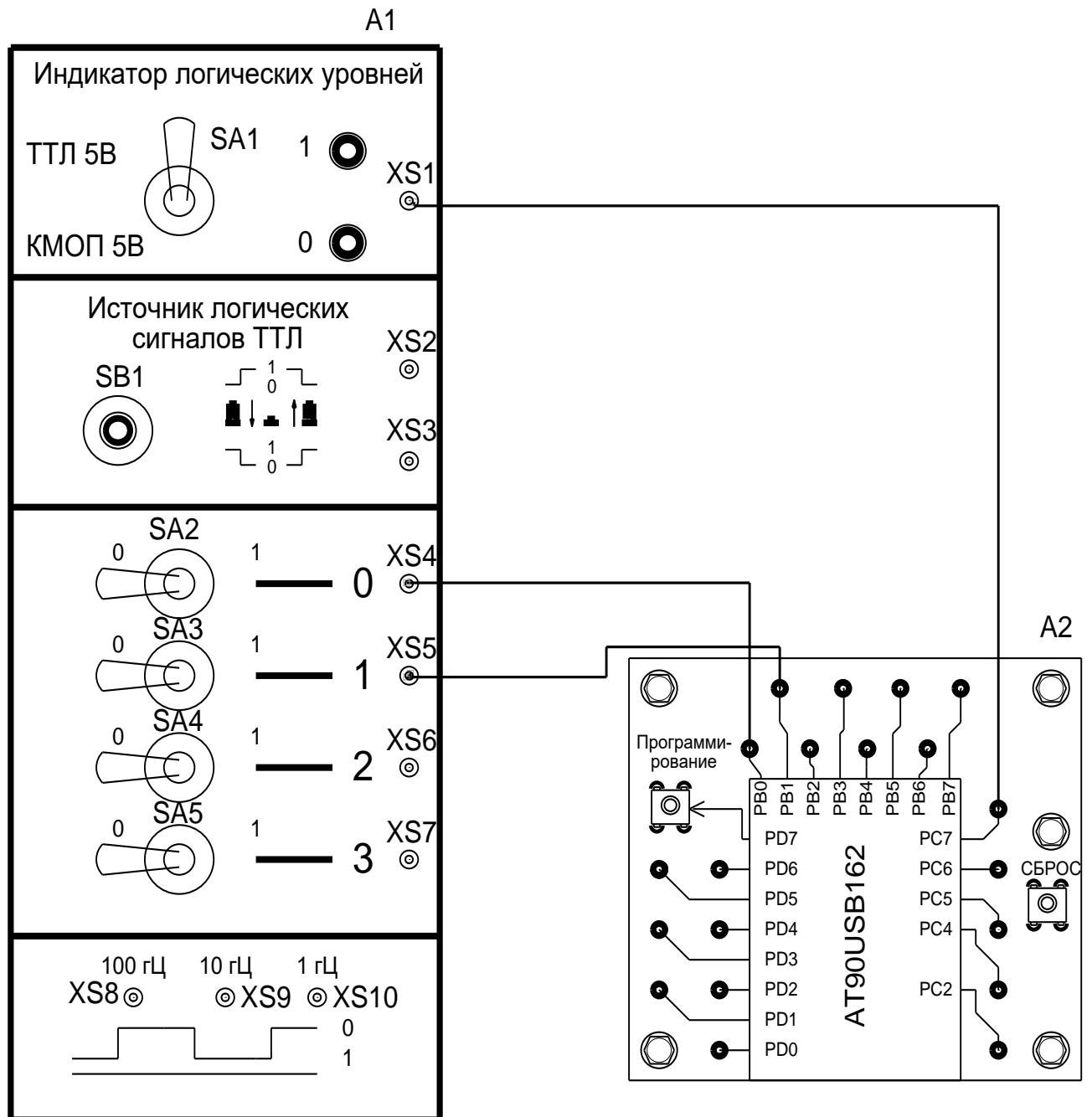
## **1. Изучение базовых возможностей оборудования и среды программирования**

Цель: Научиться создавать программу для управления контроллером или открывать и редактировать существующую программу в среде Atmel Studio, компилировать программу, загружать программу в контроллер, анализировать возникающие ошибки и исправлять их.

## **1.1. Изучение программной оболочки AtmelStudio (мой первый проект)**

Цель: Научиться создавать новую программу контроллера в программной оболочке или открывать и редактировать существующую программу. Научиться компилировать программу, анализировать и исправлять возникающие на этапе компиляции программы ошибки.

- Рекомендуемая схема соединений и ее описание
- Перечень аппаратуры и программных средств
- Указания по проведению эксперимента



**Рис.1.1.1. Рекомендуемая схема соединений**

С помощью тумблеров SA2, SA3 блока “Индикатора логических уровней” A1 задаются входные сигналы логического элемента, исследуемого в работе. Выход логического элемента подключен к выводу PC7 микроконтроллера и его состояние контролируется с помощью светодиодов блока A1.

---

## Перечень аппаратуры и программных средств

*Таблица 1.1.1.*

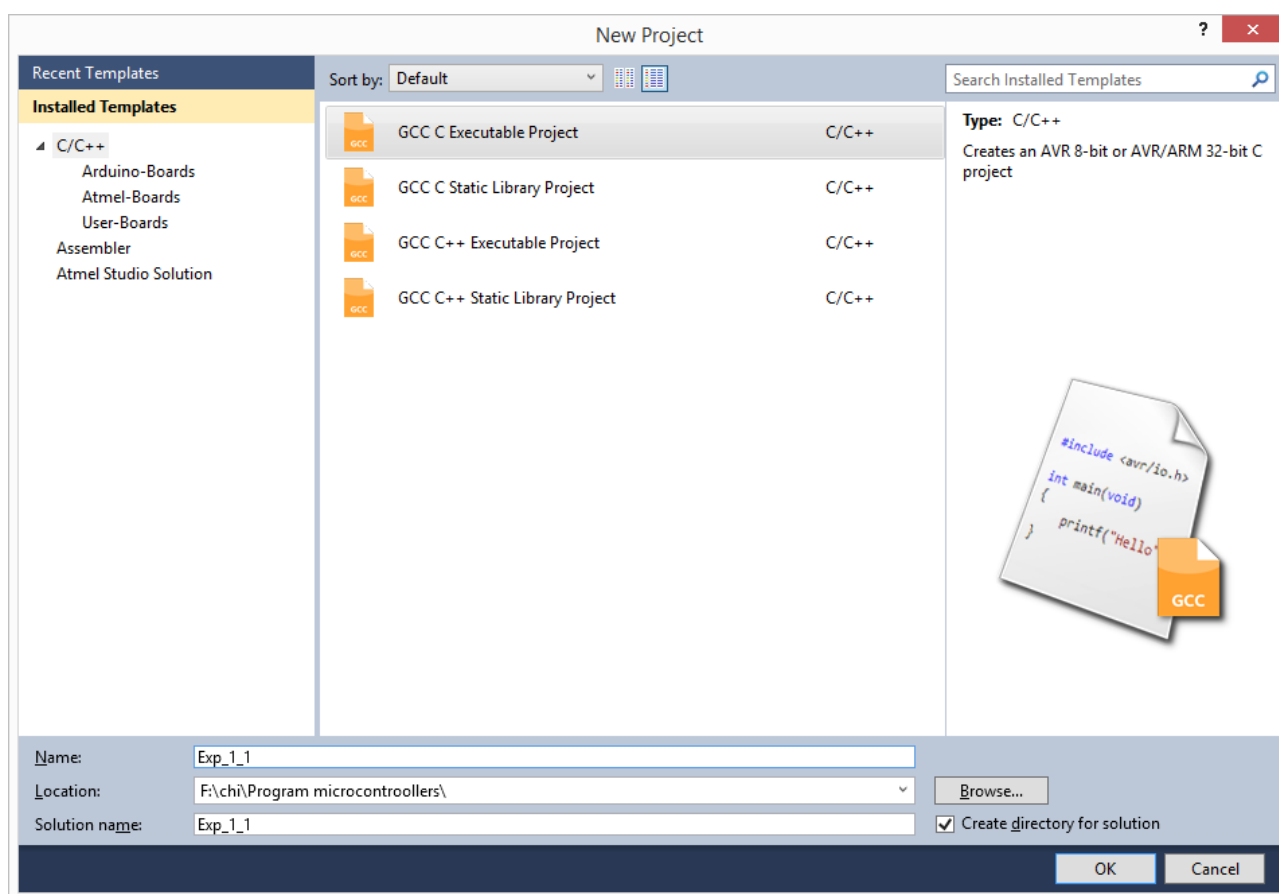
Обозначение	Наименование	Тип
A1	Блок БИЦУ	219
A2	Миниблок “Микроконтроллер AT90USB162”	Набор миниблоков 600.25
	Однофазный источник питания ОИП1	218.8
	Компьютер или ноутбук	
	Программа Atmel Studio	Версия 6.1



## Указания по проведению эксперимента

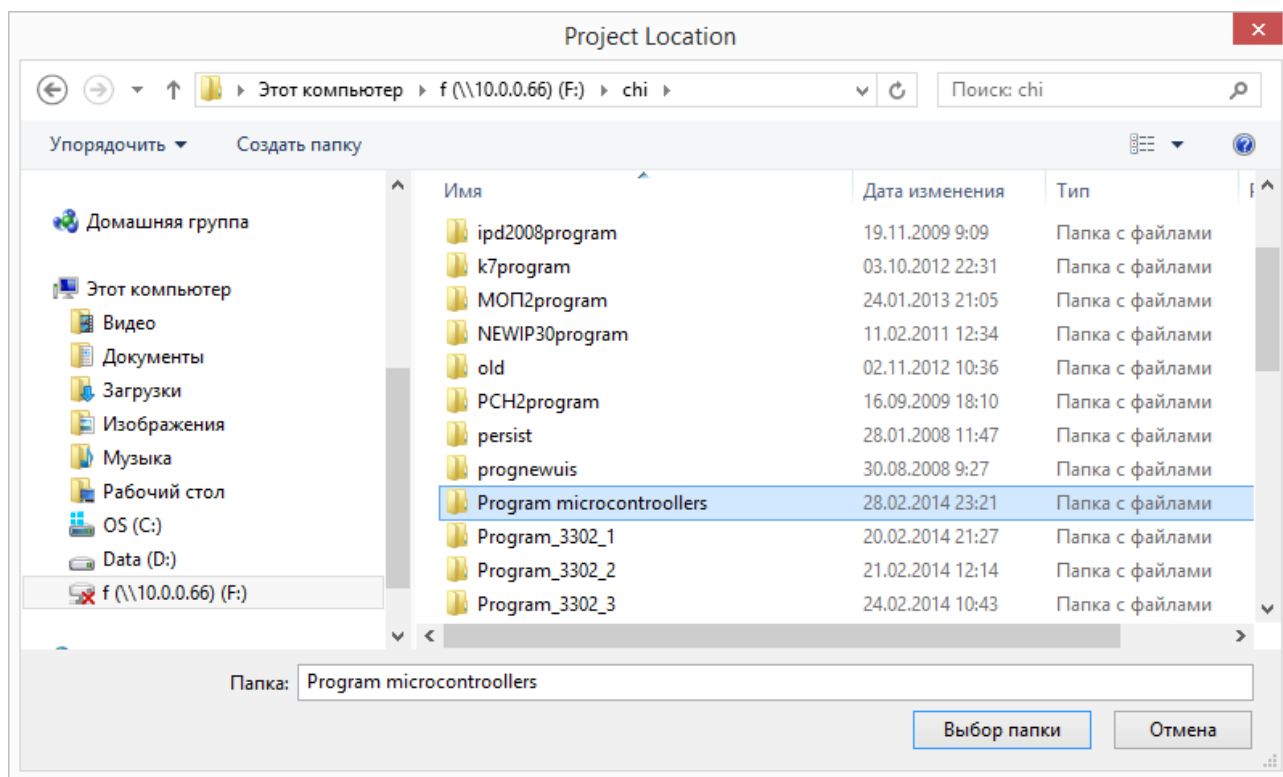
- Подключите компьютер и блок БИЦУ А1 к однофазному источнику питания ОИП, а сам источник к электрической сети.
- Подключите миниблок А2 к компьютеру, используя кабель miniUSB.
- Соберите схему в соответствии с рекомендуемой схемой соединений рис.1.1.1.
- Включите автомат на лицевой панели однофазного источника питания и выключатель СЕТЬ на лицевой панели блока БИЦУ.
- Загрузите в компьютер программу Atmel Studio.
- Создайте новый проект:
  - В открывшемся стартовом окне выберите пункт меню File(Файл) подпункт меню New(Новый) и подпункт меню Proect...(Проект...) (далее, File\New\Proect...);
  - В открывшемся блоке диалога New Project (Новый проект) рис.1.1.2 мышью выберите пункт GCC C Executable Project(GCC выполняемый проект) и заполните поля Name (Имя) и Solution Name (Имя решения).

! Оболочка Atmel Studio не всегда корректно работает с русскоязычными каталогами, именами файлов и проектов. По возможности используйте имена в английской кодировке.



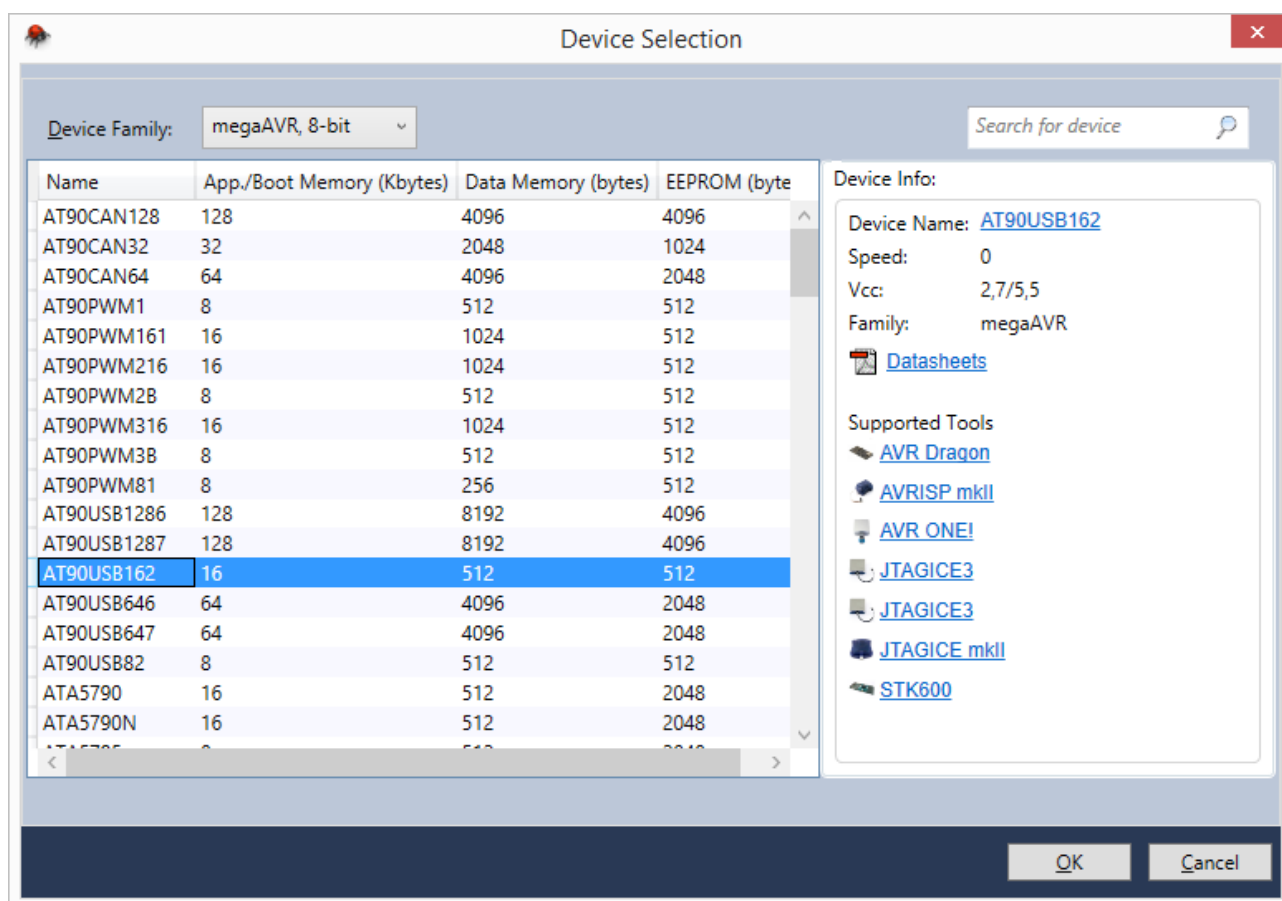
**Рис.1.1.2. Блок диалога New Project в программной оболочке Atmel Studio**

- Нажмите кнопку Browse...(Обзор...). В стандартном блоке диалога Windows (Положение проекта) рис. 1.1.3 выберите местоположение папки с вновь создаваемым проектом. Нажмите кнопку “Выбор папки” в блоке диалога Project Location и кнопку ОК в блоке диалога New Project;

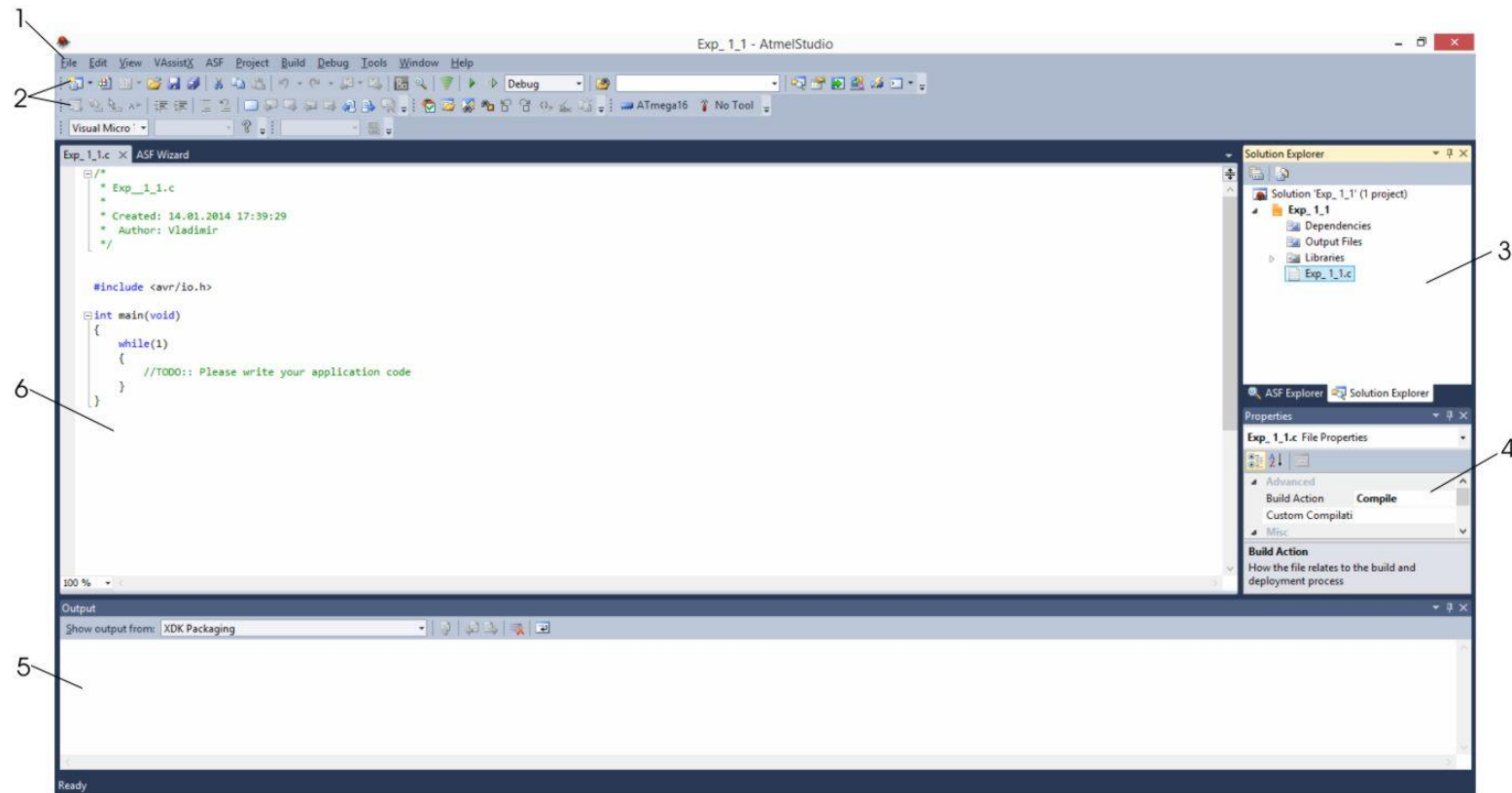


**Рис.1.1.3. Блок диалога по выбору местоположения проекта**

- Открывается блок диалога Device Selection (Выбор типа микроконтроллера) рис.1.1.4. Для облегчения поиска в окне Device Family (Семейство миккроконтроллеров) выберите пункт Mega AVR, 8-bit. Выберите микроконтроллер AT90USB162 и нажмите кнопку ОК;
- Если все сделано правильно, оболочка перестраивается и открывается с вновь созданным проектом рис. 1.1.5;

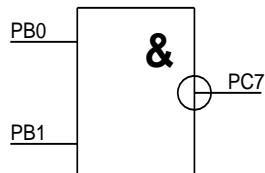


**Рис.1.1.4. Блок диалога по выбору типа микроконтроллера**



**Рис.1.1.5. Программная оболочка Atmel Studio с вновь созданным проектом**

1. Строка меню.
  2. Панель инструментов.
  3. Окно вывода данных о проекте.
  4. Окно свойств со свойствами объекта выделенного в окне вывода данных о проекте.
  5. Многофункциональное окно для вывода данных при компиляции проекта, поиске данных в проекте, вывода данных проекта.
  6. Окно редактирования файлов проекта.
- В окне редактирования файлов открывается файл, имя которого совпадает с именем проекта Exr\_1\_1.c. Файл содержит заготовку для начала нового проекта:
    - В начале файла, создается описание проекта с именем файла и датой создания проекта. К описанию добавляется имя автора проекта, совпадающее с именем пользователя компьютера; Любые описания не используются в дальнейшем компиляторами программы и могут быть изменены или удалены.
    - Директивой `#include` в программу добавляется заголовочный файл `<avr/io.h>`. На этапе создания проекта, был выбран микроконтроллер AT90USB162. При этом оболочка AtmelStudio автоматически создала `#define` определение `_AVR_AT90USB162_`. Файл `io.h` находит это определение и в соответствии с ним подключает `include` файл `iousb162.h`, который определяет имена всей периферии доступной в микроконтроллере;
    - Если по какой-либо причине, тип микроконтроллера в проекте не определен, файл `io.h` вызывает ошибку компиляции.
    - Автоматически создается программа `main` с бесконечным циклом `while`.
  - Напишем программу, которая реализует логический элемент 2И-НЕ рис. 1.1.6. Листинг соответствующей программы с комментариями представлен на рис. 1.1.7.







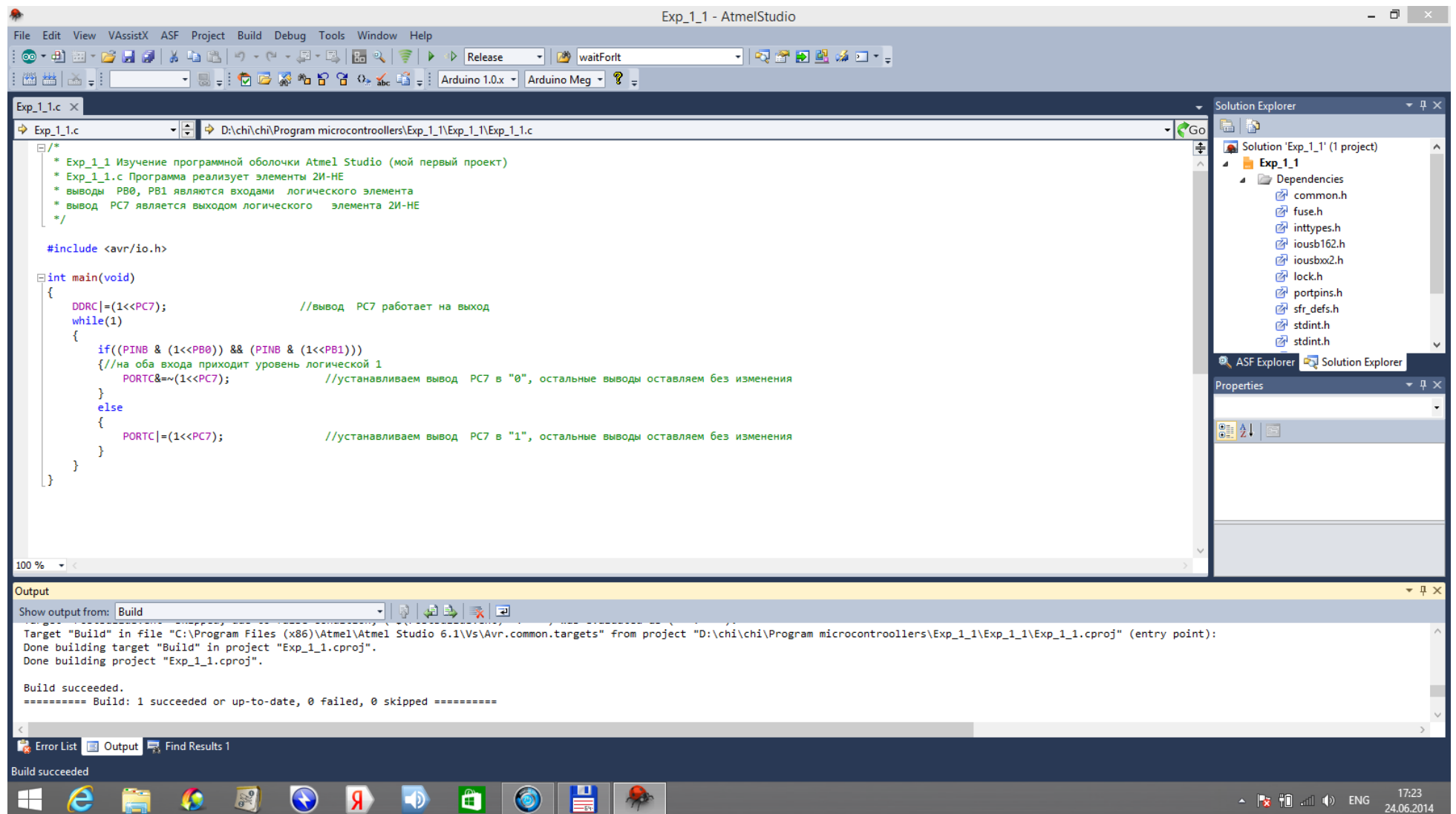
**Рис. 1.1.6. Логический элемент 2И-НЕ**

```
#include <avr/io.h>

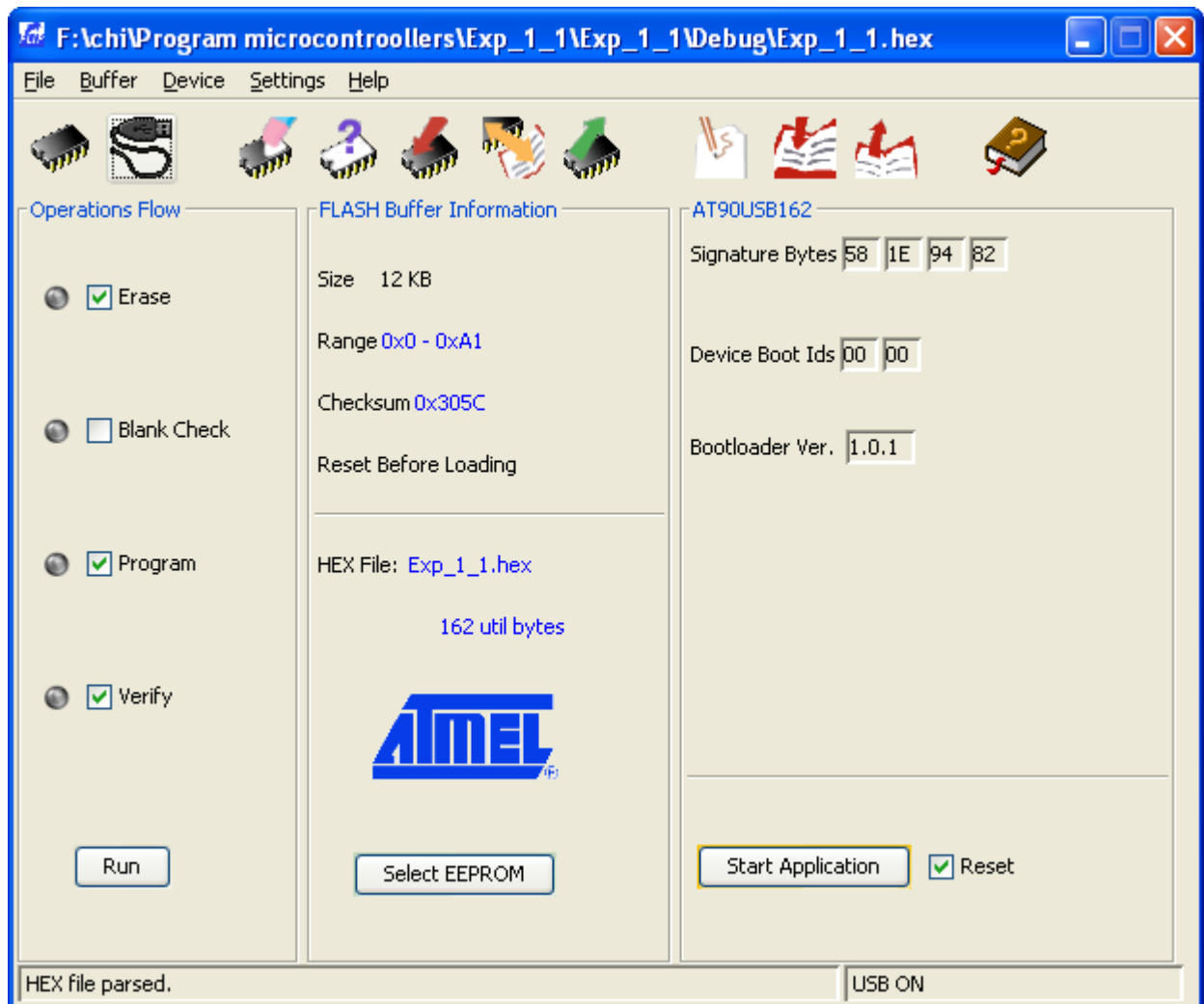
int main(void)
{
    DDRC|=(1<<PC7);           //вывод PC7 работает на выход
    while(1)
    {
        if((PINB & (1<<PB0)) && (PINB & (1<<PB1)))
        { //на оба входа приходит уровень логической 1
            PORTC&=~(1<<PC7);   //устанавливаем вывод PC7 в "0", остальные
                                //выводы оставляем без изменения
        }
        else
        {
            PORTC|=(1<<PC7);     //устанавливаем вывод PC7 в "1", остальные
                                //выводы оставляем без изменения
        }
    }
}
```

**Рис.1.1.7. Программа логического элемента 2И-НЕ**

- Измените настройки режима компилятора в режим работы без оптимизации кода:
  - В окне Solution Explorer (Управление решением) подведите курсор к названию проекта Exp\_1\_1 и нажмите правую кнопку мыши. Из контекстного меню выберите пункт Properties (Свойства).
  - В открывшемся окне свойств поле Configuration (Конфигурация) установите в значение Release. Переключитесь на страницу ToolChain (Цепь инструментов). Откройте пункт AVR/GNU C Compiler и выберите подпункт Optimisation (Оптимизация);
  - Установите пункт Optimisation Level (Уровень оптимизации) в значение None (-O0);
  - В оболочке Atmel Studio установите окно Solution Configuration (Решения конфигурация) в значение Release.
- Откомпилируйте написанную программу, нажав кнопку  или выберите соответствующий пункт меню Build/Build Exp\_1\_1.
- Если все сделано правильно, после компиляции, экран будет выглядеть подобно тому, как представлено на рис 1.1.8.
- В окне Output (в нижней части оболочки) выдается результат построения проекта. При успешном построении в нижней строке Build выводится надпись succeeded (успешно). Если пролистать окно Output вверх можно увидеть, какие файлы были откомпилированы и какие файлы были созданы. Дополнительно выводится информация об использованных ресурсах контроллера Program Memory Usage (Использование памяти программы) и Data Memory Usage (Использование памяти данных). Количество используемой памяти выводится в байтах и в процентах от общего количества памяти.
- Запишем полученную программу в микроконтроллер:
  - Подайте питание на блок БИЦУ A1.
  - Запустите программу FLIP;
  - Нажмите кнопку  или выберите пункт меню Device/Select... (Устройство/выбрать...) в открывшемся блоке диалога Device Selection (Выбор устройства) выберите микроконтроллер AT90USB162;
  - Нажмите кнопку  и в открывшемся блоке диалога Load HEX/A90 File войдите в папку, где расположен ваш проект. Затем в папку RES и выберите файл EXP\_1\_1.hex.
  - В окне Operation Flow выберите пункты
    - ✓ Erase (Стереть) - стирает старую программу в микроконтроллере;
    - ✓ Program (Программировать) – программирует микроконтроллер с записанной программой;
    - ✓ Verify (Проверить) – проверяет правильность записи программы.
  - Чтобы перевести миниблок в режим программирования, на плате миниблока AT90USB162 одновременно нажмите кнопки **СБРОС** и **Программирование**. Отпустите сначала кнопку **СБРОС**, затем кнопку **Программирование**.
  - В программе FLIP нажмите кнопку  Select a Communication Medium (Выбор способа соединения). В открывшемся списке выберите пункт USB. При успешном соединении в правом нижнем углу появляется надпись USB ON. Если связь установить не удастся выводится блок диалога с ошибкой (Проверьте подключение кабеля USB, наличие питания на блоке БИЦУ, повторно переведите плату в режим программирования и повторите попытку соединения);
  - Если все выполнено правильно внешний вид блока диалога будет иметь вид представленный на рис. 1.1.9;



**Рис.1.1.8. Внешний вид оболочки AtmelStudio после успешной компиляции проекта**



**Рис.1.1.9. Блок диалога при программировании микроконтроллера**

- Нажмите кнопку RUN, чтобы запрограммировать микроконтроллер;
- Выводится блок диалога, показывающий порядок выполнения операции. При успешном выполнении операции кнопки Erase, Program, Verify становятся зелеными. При возникновении ошибки кнопка, где произошла ошибка, становится красной;
- Запустите программу, нажав кнопку Start Application программатора Flip или кнопку СБРОС миниблока.
- Переключая тумблеры убедитесь, что микроконтроллер выполняет функции элемента 2И\_НЕ:
- Давайте допишем программу таким образом, чтобы реализовать элемент 2ИЛИ имеющего те же входы, но выход PC6.
- Чтобы оставить исходный код программы без изменения, сохраним программу Exp1\_1c под именем Exp1\_1mist, выбрав из меню File/Save as (Файл/ Сохранить как).
- Допишите программу, как показано на рис 1.1.10. Добавленный код выделен жирным шрифтом. Чтобы посмотреть, как компилятор находит ошибки в исходном коде в текст программы внесены две ошибки.



```

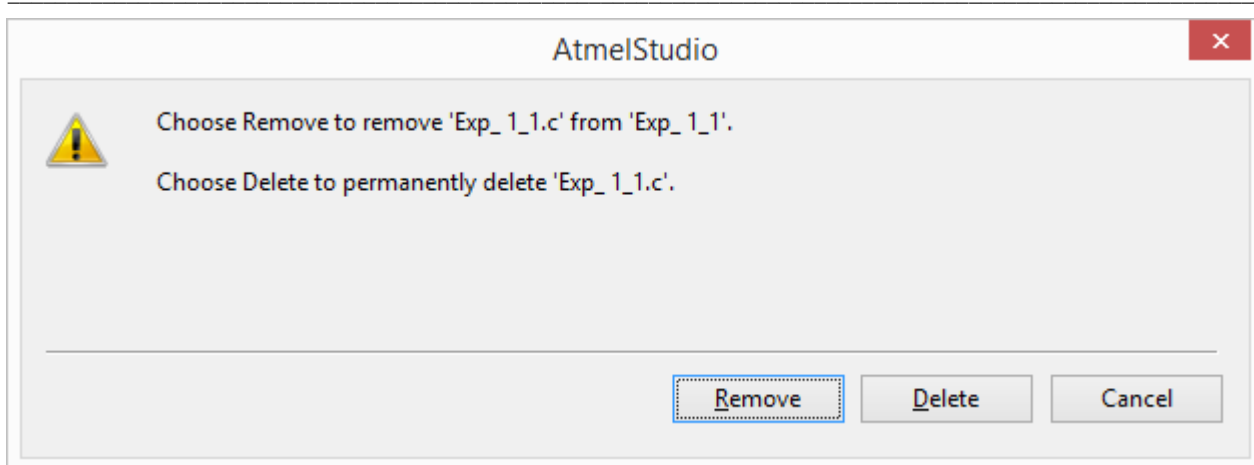
#include <avr/io.h>

int main(void)
{
    DDRC|=(1<<PC7)+ (1<<PC6);           //вывод PC7 работает на выход
    while(1)
    {
        if((PINB & (1<<PB0)) && (PINB & (1<<PB1)))
        { //на оба входа приходит уровень логической "1"
            PORTC&=~(1<<PC7);           //устанавливаем вывод PC7 в "0", остальные
                                         //выводы оставляем без изменения
        }
        else
        {
            PORTC|=(1<<PC7);             //устанавливаем вывод PC7 в "1", остальные
                                         //выводы оставляем без изменения
        }
        if((PINB & (1<<PB0))==0) && ((PINB & (1<<PB1))==0))
        { //на оба входа приходит уровень логического "0"
            PORTC &=~(1<<PC6)           //устанавливаем вывод PC6 в "0", остальные
                                         //выводы оставляем без изменения
                                         //Ошибка, нет точки с запятой после оператора
        }
        else
        {
            PORTC |= (1<<PC6);           //устанавливаем вывод PC6 в "1", остальные
                                         //выводы оставляем без изменения
                                         //!!!ошибка, в слове PORTC вместо английской
                                         //буквы C, русская буква
        }
    }
}


```

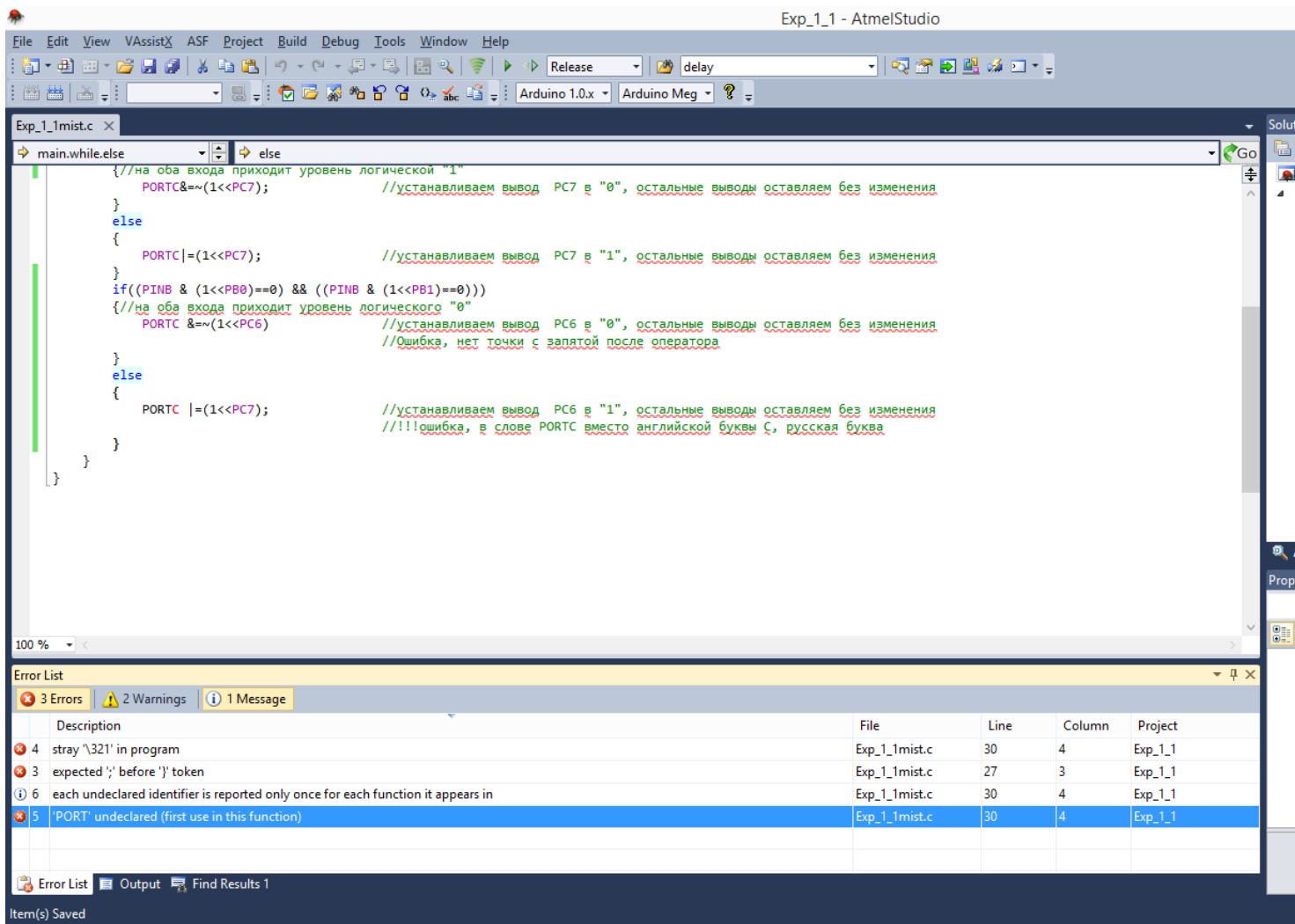
### Рис.1.1.10. Программа с ошибками

- Заменим в проекте название файла:
  - Переместите мышь на строку названия проекта Exp\_1\_1 в Solution Explorer и нажмите правую кнопку мыши;
  - В всплывающем меню выберите пункт ADD/Existing Item... (Добавить/Существующий элемент);
  - В открывшемся блоке диалога выберите файл exp\_1\_1mist. Теперь в проекте два файла. В Solution Explorer выберите мышью файл exp\_1\_1 и нажмите клавишу delete клавиатуры, чтобы удалить файл. Открывается блок диалога рис.1.1.11 с предложением :
    - Remove (Убрать) - убрать файл из проекта;
    - Delete (Удалить) - удалить файл из проекта и с диска;
    - Cancel (Отменить) – отменяет любые действия с файлом



**Рис. 1.1.11. Блок диалога при удалении файла из проекта**

- Откомпилируйте проект, нажав кнопку . После компиляции оболочка принимает вид представленный на рис. 1.1.12. Нижнее окно автоматически переключается на вкладку Error List (Список ошибок).
- Подведите курсор мыши к первой ошибке `stray '\321' in program` и дважды нажмите левую кнопку мыши. Курсор в окне программы перемещается на строку `PORTC |=(1<<PC6);` где неправильно набрана буква в названии порта PORTC. Дополнительно редактор выделяет эту букву красным цветом. Исправьте данную ошибку.
- Вторая ошибка “Expected ‘;’ before ‘}’ token” (Ожидается ';' перед символом '}'). Дважды щелкните мышью по ошибке. В окне редактирования курсор перемещается на строку с ошибкой. Но он оказывается не на том месте, где была совершена ошибка, а на следующей строке, где появился символ '}', который компилятор не ожидал. Поставьте символ ';' в конце строки `PORTC&=~(1<<PC6);`.
- Во второй строке окна Error List выводится предупреждение ‘each undeclared identifier is reported only once for each function it appears in’ (каждый недекларированный идентификатор показывается только один раз для каждой функции, где он используется). Дважды щелкните мышью по строке предупреждения. Курсор в окне редактирования укажет на несуществующий идентификатор PORTT.
- В третьей строке окна Error List выводится предупреждение, указывающее на то, что идентификатор PORTC с русской буквой будет отмечен ошибкой только в одном месте.
- Ошибка в четвертой строке указывает на то, что идентификатор PORTC с русской буквой не определен.
- Повторно откомпилируйте программу.
- Загрузите программу в контроллер
- Проверьте правильность работы программы.
- Измените программу для реализации других видов логических элементов.



**Рис. 1.1.12. Оболочка Atmel Studio после компиляции проекта с ошибкам**

## **2. Программирование таймеров микроконтроллера**

Цель: Научиться программировать таймеры для формирования выдержек времени, сигналов заданной частоты, измерения частот и длительностей импульсов внешних сигналов.

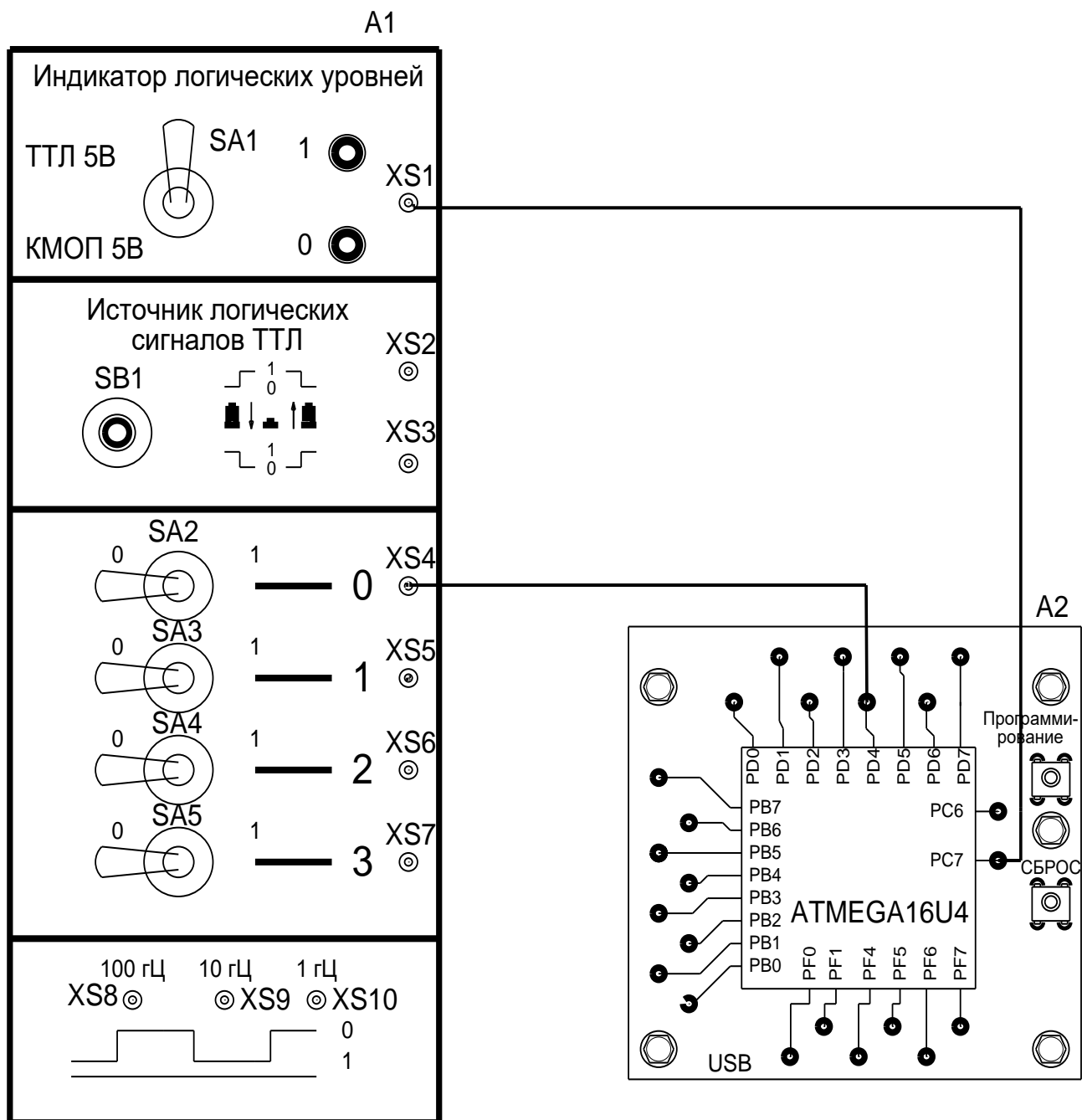
---

## **2.1. Формирование выдержки времени с помощью таймера**

Цель: Научиться использовать встроенный в микроконтроллер таймер для формирования заданной выдержки времени.

- Рекомендуемая схема соединений и ее описание.
- Перечень аппаратуры и программных средств
- Указания по проведению эксперимента

### Рекомендуемая схема соединений и ее описание



**Рис.2.1.1. Рекомендуемая схема соединений**

При переключении тумблера SA2 на уровень логической “1”, микроконтроллер формирует выдержку времени и зажигает красный светодиод индикатора логических уровней блока БИЦУ A1.

---

**Перечень аппаратуры и программных средств***Таблица 2.1.1.*

Обозначение	Наименование	Тип
A1	Блок БИЦУ	219
A2	Миниблок “Микроконтроллер ATMEGA16U4”	Набор миниблоков 600.25
	Однофазный источник питания ОИП1	218.8
	Компьютер или ноутбук	
	Программа Atmel Studio	Версия 6.1

## Указания по проведению эксперимента

- Подключите компьютер и блок БИЦУ А1 к однофазному источнику питания ОИП, а сам источник к электрической сети.
- Подключите миниблок А2 к компьютеру, используя кабель miniUSB.
- Соберите схему в соответствии с рекомендуемой схемой соединений рис.2.1.1.
- Включите автомат на лицевой панели однофазного источника питания и выключатель СЕТЬ на лицевой панели блока БИЦУ.
- Загрузите в компьютер программу Atmel Studio.
- Загрузите программы эксперимента Program microcontrollers\Exp\_2\_1\Exp\_2\_1.
- Откройте файл Exp\_2\_1\_1.c. Исходный текст представлен на рис. 2.1.2.

```

/*
 * Exp_2_1   Программирование таймеров микроконтроллера
 * Exp_2_1_1.c Программирование выдержки времени
 * При включении тумблера SA2 таймер формирует заданную выдержку времени и после ее
 * окончания программа зажигает светодиод индикатора логических уровней блока БИЦУ А1.
 */
#include <avr/io.h>
int main(void)
{
    DDRC=0x80;           //вывод C7, управления светодиодом VD7 работает на вывод
    TCCR1A=0x00;          //обычный режим работы таймера,
    TCCR1B=5;             //коэффициент прескалера 1024
    OCR1A=4883;           //задание выдержки времени 5с при частоте тактового генератора
                        //микроконтроллера 1 МГц. Величина OCR1A быть
                        //рассчитана по формуле:
                        // OCR1A=T*f/1024-1
                        //где
                        // T - заданная выдержка времени,
                        // 1024 -/коэффициент прескалера,
                        // f - частота тактового генератора
    unsigned char flag=0; //флаг начала новой выдержки времени
    while(1)
    {
        if(PIND & (1<<PD4) )
        { //Включен тумблер SA2
            if (flag==0)
            { //выдержка времени не начата
                TCNT1=0;
                TIFR1|=(1<<OCF1A); //сбрасываем флаг таймера
                flag=1;
            }
        }
        else
        { //Тумблер SA2 выключен
            flag=0;
            PORTC&=~(1<<PC7);
        }
        // здесь основной цикл программы
        if((TIFR1 & (1<<OCF1A)) && (flag==1) )
        { //закончилась выдержка времени
            PORTC|=(1<<PC7); //зажигаем светодиод
        }
    }
}

```

**Рис. 2.1.2. Текст программы эксперимента Exp\_2\_1\_1**



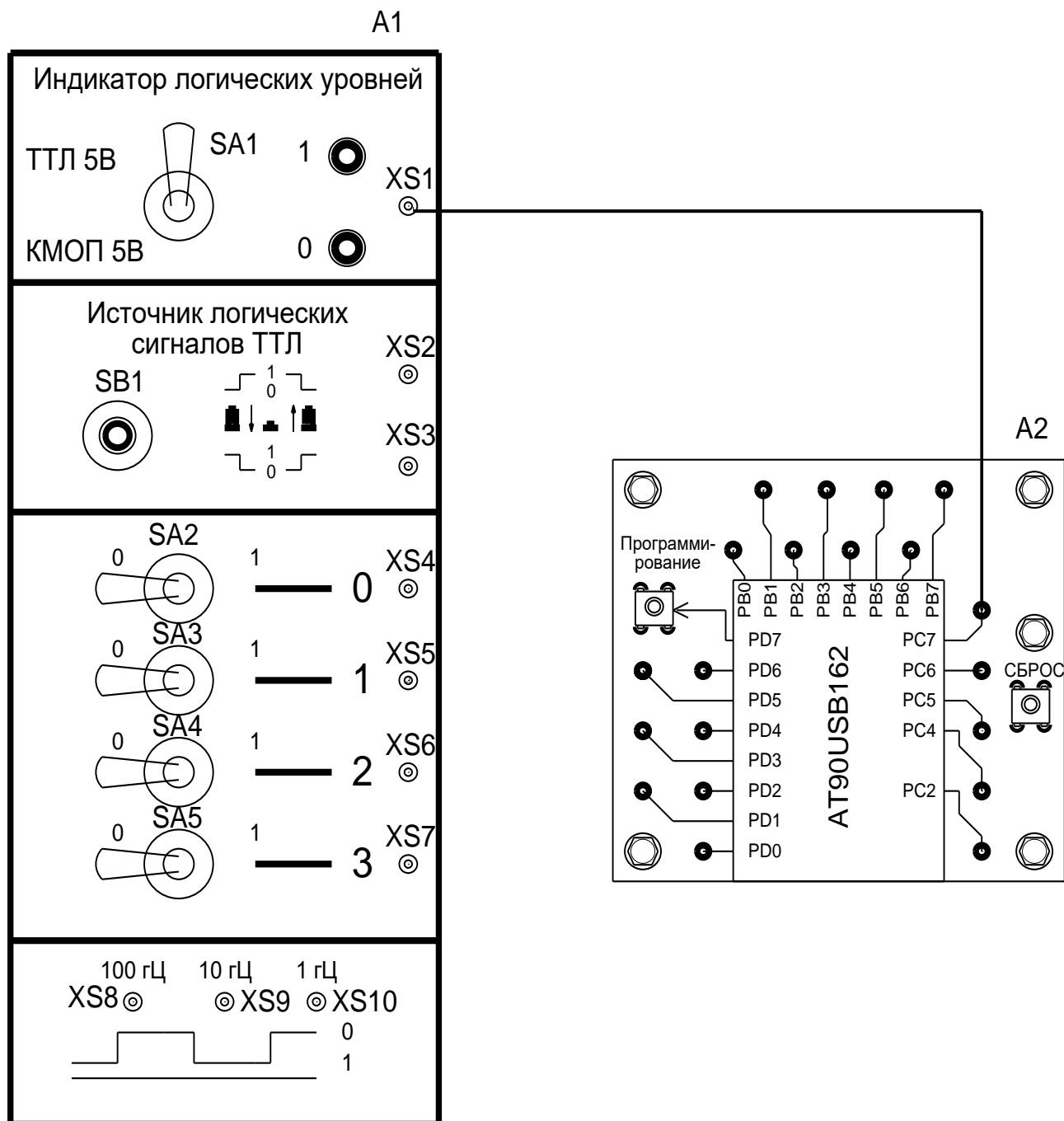
- 
- Откомпилируйте программу и загрузите программу в контроллер с помощью программатора FLIP (загрузочный файл программы Exr\_2\_1\_1.hex в каталоге Release проекта).
  - Запустите программу. Проверьте правильность формируемой выдержки времени от момента нажатия кнопки до момента зажигания светодиода.
  - Задайте другую выдержку времени и проверьте правильность ее формирования.
  - Добавьте после строки “здесь основной цикл программы”, какой либо код с большим временем выполнения, соизмеримым с выдержкой времени таймера (например, цикл for). Проверьте влияние длительности выполнения основного кода на правильность формирования выдержки времени.
  - Измените алгоритм программы к виду:
    - ввод данных с кнопки;
    - 1/3 от основного цикла;
    - ввод флага таймера и зажигание светодиода;
    - ввод данных с кнопки;
    - 1/3 от основного цикла программы;
    - ввод флага таймера и зажигание светодиода;
    - ввод данных с кнопки;
    - 1/3 от основного цикла программы;
    - ввод флага таймера и зажигание светодиода.
  - Проверьте точность формируемой выдержки времени при данном алгоритме работы контроллера.

## **2.2. Формирование сигнала заданной частоты**

Цель: Научиться использовать таймер микроконтроллера для формирования сигнала типа меандр с заданной регулируемой частотой.

- Рекомендуемая схема соединений и ее описание.
- Перечень аппаратуры и программных средств
- Указания по проведению эксперимента

### Рекомендуемая схема соединений и ее описание



**Рис.2.2.1. Рекомендуемая схема соединений**

Вывод PC6 при инициализации таймера 1 запрограммирован для выдачи сигнала непосредственно с этого таймера. По светодиодам индикатора логических уровней блока БИЦУ А1, при низкой частоте таймеров, можно судить о получаемой частоте и коэффициенте заполнения.

---

### Перечень аппаратуры и программных средств

*Таблица 2.2.1.*

Обозначение	Наименование	Тип
A1	Блок БИЦУ	219
A2	Миниблок “Микроконтроллер AT90USB162”	600.25
	Однофазный источник питания ОИП1	218.8
	Компьютер или ноутбук	
	Программа Atmel Studio	Версия 6.1

## Указания по проведению эксперимента

- Подключите компьютер и блок БИЦУ (А1) к однофазному источнику питания ОИП, а сам источник к электрической сети.
- Подключите миниблок А2 к компьютеру, используя кабель miniUSB.
- Соберите схему в соответствии с рекомендуемой схемой соединений рис.2.2.1.
- Включите автомат на лицевой панели однофазного источника питания и выключатель СЕТЬ на лицевой панели блока БИЦУ.
- Загрузите в компьютер программу Atmel Studio.
- Загрузите программы эксперимента Program microcontrollers\Exp\_2\_2\Exp\_2\_2.
- Откройте файл Exp\_2\_2.c. Исходный текст программы представлен на рис. 2.2.2.

```

/* Exp_2 "Программирование таймеров микроконтроллера"
 * Exp_2_2 "Формирование сигнала заданной частоты"
 * Exp_2_2.c В программе при инициализации таймер программируется для работы в
 * режиме CTC с вершиной по регистру OCR1A. Выход таймера OC1A (вывод PC6)
 * запрограммирован в режим переключения и подключен к светодиодам индикатора
 * логических уровней блока БИЦУ А1. При низкой частоте таймера,
 * по состоянию светодиодов можно судить о частоте и коэффициенте заполнения
 * формируемого сигнала.
 */

#include <avr/io.h>

int main(void)
{
    DDRC|=(1<<PC6);           //вывод PC6 работает на выход
    TCCR1A=0x40;              // режим CTC , с переключением вывода OCR1B
    TCCR1B=0xD;                //коэффициент прескалера 1024
    OCR1A=4883;                // OCR1A=1000000/(2*1024*f)-1
                               // где:
                               // 1000000 – тактовая частота генератора в Гц
                               // 1024 - коэффициент прескалера
                               // f – частота, которую планируется получить
                               // при f=0.1 Гц
                               // OCR1A=1000000/(2*1024*0.1)-1=4883

    while(1)
    {

    }
}

```

**Рис. 2.2.2. Текст программы эксперимента Exp\_2\_2.**

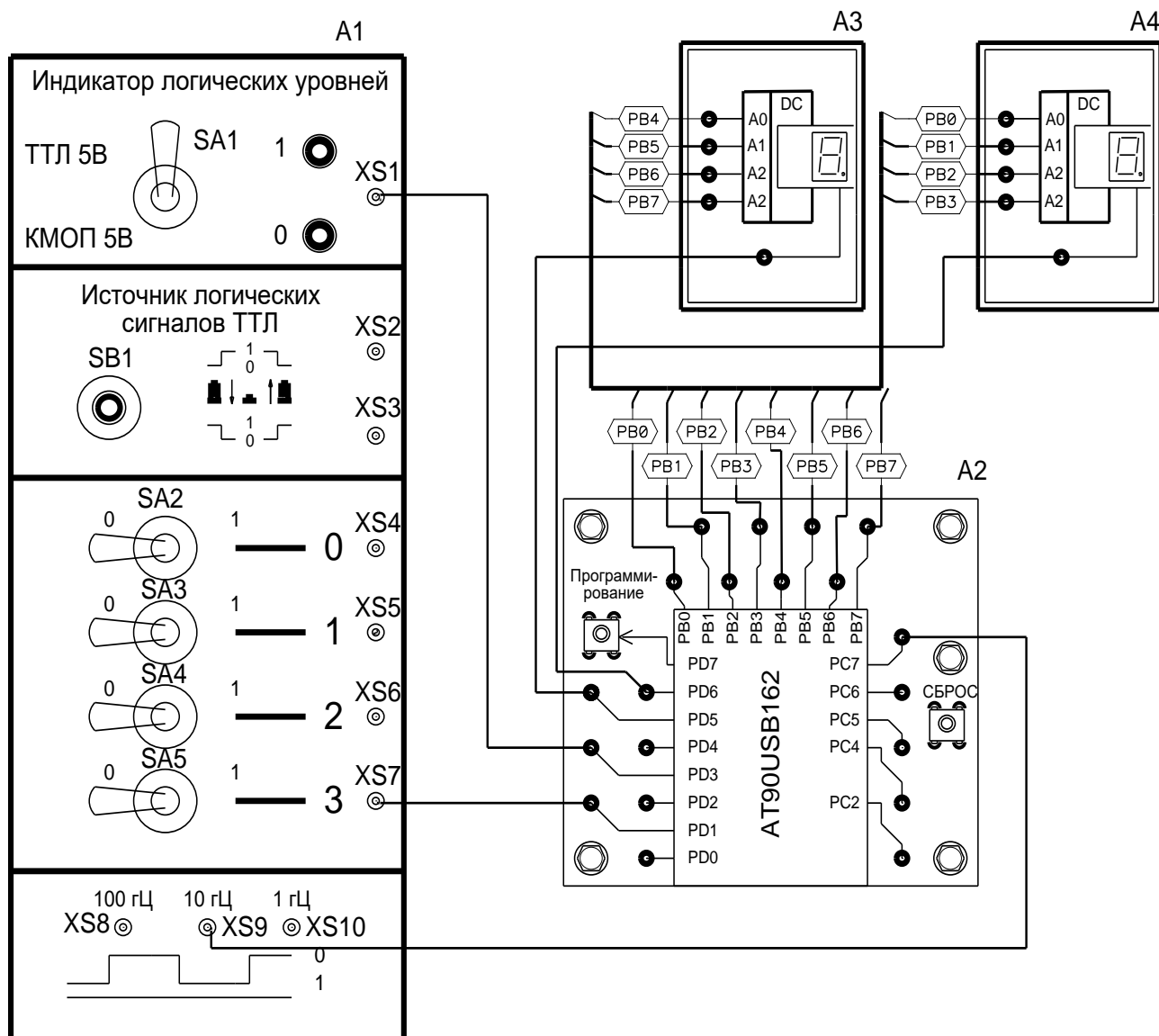
- По скорости переключения светодиодов определите частоту таймера и коэффициент заполнения. Сравните полученные величины с заданными.
- При наличии осциллографа (в состав стенда не входит), увеличьте частоту таймеров до нескольких кГц и проверьте полученный результат на этой частоте.
- Добавьте в цикл while, какой либо цикл, со временем выполнения большим, чем период заданной частоты. Проверьте, как эта программа повлияет на работу таймера.
- Подсоедините к выводам (предположим PD0, PD1) тумблеры SA2, SA3. Сформируйте с помощью таймера 2 выдержку времени. Обеспечьте, чтобы при каждом окончании выдержки времени и включенном состоянии тумблера SA2, формируемая на выводе PC7 частота увеличивалась, а при включенном состоянии тумблера SA3 уменьшалась.

## **2.3. Определение длительности внешних сигналов с помощью таймеров**

Цель: Научиться определять длительность импульсов и частоту внешних сигналов с использованием таймеров.

- Рекомендуемая схема соединений и ее описание.
- Перечень аппаратуры и программных средств
- Указания по проведению эксперимента

### Рекомендуемая схема соединений и ее описание



**Рис.2.3.1. Рекомендуемая схема соединений**

Вывод PC7 при инициализации таймера 1 запрограммирован для определения момента прихода фронта внешнего сигнала. Тумблер SA5 при установке в положение “0” переключает программу в режим измерения длительности импульса. При переключении в положение “1” определяется частота внешнего сигнала. Полученные значения периода сигнала в секундах и частоты сигнала в Гц выводятся на семисегментные индикаторы A3, A4. Красный светодиод блока БИЦУ сигнализирует о перегрузке (длительность импульса или частота слишком большие, чтобы вывести их на индикаторы).

### Перечень аппаратуры и программных средств

*Таблица 2.3.1.*

Обозначение	Наименование	Тип
A1	Блок БИЦУ	219
A2	Миниблок “Микроконтроллер AT90USB162”	Набор миниблоков 600.25
A3, A4	Миниблоки семисегментных индикаторов	Набор миниблоков 600.11
	Однофазный источник питания ОИП1	218.8
	Компьютер или ноутбук	
	Программа Atmel Studio	Версия 6.1



### Указания по проведению эксперимента

- Подключите компьютер и блок БИЦУ (А1) к однофазному источнику питания ОИП, а сам источник к электрической сети.
- Подключите миниблок А2 к компьютеру, используя кабель miniUSB.
- Соберите схему в соответствии с рекомендуемой схемой соединений рис.2.3.1.
- Включите автомат на лицевой панели однофазного источника питания и выключатель СЕТЬ на лицевой панели блока БИЦУ.
- Соберите схему в соответствии с рекомендуемой схемой соединений рис.2.3.1.
- Загрузите в компьютер программу Atmel Studio.
- Загрузите программы эксперимента Program microcontrollers\Exp\_2\_3\Exp\_2\_3.
- Откройте файл Exp\_2\_3.c. Исходный текст представлен на рис. 2.3.2.

```

/* Exp_2_3 "Определение длительности внешних сигналов с помощью таймеров"
 * Exp_2_3.c Программа в зависимости от состояния тумблера SA5 измеряет длительность
 * импульса (положение 0) или частоту сигнала (положение 1), подаваемого на вывод PC7.
 * Полученный результат выводится на семисегментный индикатор.
 */
#include <avr/io.h>
#include "Exp_2_3.h"
int main(void)
{
    unsigned int Tf;                //переменная для хранения времени захвата переднего фронта
    DDRB=0xFF;                      //выходы управления семисегментными индикаторами
    DDRD|=(1<<PD3)+(1<<PD5)+(1<<PD6); //выходы управления светодиодом и точками
                                     //семисегментных индикаторов
    TCCR1A=0x0;                    //инициализация таймера 0 с коэффициентом прескалера 1024
    TCCR1B=0xC5;                   //с захватом сигнала по переднему фронту, подавитель помех
                                     //от дребезга сигналов включен.

    //!!! переключение частоты генератора на 8 МГц
    unsigned char a=0x80;
    unsigned char b=0;
    unsigned int ptr=0x61;
    asm volatile(
        "st  %a0, %1"                "\n\t"
        "st  %a0, %2"
        :
        : "e"(ptr), "r"(a), "r"(b));

    //!!!
    while(1)
    {
        if(TIFR1 & (1<<ICF1))
        { //произошел захват сигнала
            if(PIND & (1<<PD1))
            { //режим определения частоты сигнала
                TCCR1B=0xC5;          //возвращаемся к захвату сигнала по переднему фронту
                Tf=ICR1;              //запоминаем полученный результат
                TCNT1=0;              //начинаем отсчет нового периода сигнала
                TIFR1 &=(1<<ICF1);
                //сбрасываем флаг захвата
                PORTD&=~(1<<PD3);      //сбрасываем светодиод переполнения
                CalcFreq(Tf);          //расчет частоты сигнала и вывод на семисегментный
                                     //индикатор полученного результата

                PORTD&=~(1<<PD6);      //тушим точки после индикаторов
                PORTD&=~(1<<PD5);
            }
            else
            { //режим определения длительности импульса
                if(TCCR1B & (1<<ICES1))
                { //захват по переднему фронту, начинаем отсчет импульса
                    TCNT1=0;          //начинаем отсчет новой длительности импульса
                    TCCR1B=0x85;      //переключаем таймер на захват по заднему
                                     //фронту
                    TIFR1 &=(1<<ICF1); //сбрасываем флаг захвата
                }
            }
        }
    }
}

```

```

else
{
    //задний фронт сигнала, можно определить длительность импульса
    Tf=ICR1; //время в тиках таймера
    TCNT1=0; //начинаем новый отсчет
    TCCR1B=0xC5; //переключаем на захват по переднему фронту
    TIFR1 &=(1<<ICF1); //сбрасываем флаг захвата
    PORTD&=~(1<<PD3); //сбрасываем светодиод переполнения
    CalcTimp(Tf); //расчет длительности импульса и вывод
    //полученного результата на индикатор
    PORTD&=~(1<<PD6); //тушим точку после младшего индикатора
    PORTD|=(1<<PD5); //зажигаем точку после старшего индикатора
}
}
}
if(TIFR1 &(1<<TOV1))
{
    TIFR1|=(1<<TOV1); //очищаем переполнение
    PORTD|=(1<<PD3); //зажигаем светодиод
}
//здесь основная программа
}

//расчет частоты сигнала и вывод на семисегментный индикатор полученного результата
//расчет частоты производится по формуле:
//f=8000000/(1024*nimp)
//где
// nimp - число импульсов подсчитанное таймером
// 1024 - установленный коэффициент прескалера
// 8000000 - тактовая частота микроконтроллера 8 МГц
//если полученная частота больше 99 Гц, зажигаем светодиод переполнения
void CalcFreq(unsigned int nimp)
{
    unsigned int f; //частота внешнего сигнала
    f=7813/nimp;
    if(f>99)
    {
        //переполнение
        PORTD|=(1<<PD3); //зажигаем красный светодиод переполнения
    }
    PORTB=PrintBCD99(f) ; //переводим полученное число к двоично-десятичному формату
    //и выдаем полученный результат на индикатор
}

//программа преобразует число к двоично-десятичному числу, упакованному в один байт
//n - преобразуемое число (в диапазоне от 0...99)
//выход: двоично-десятичное число
unsigned char PrintBCD99(unsigned char n)
{
    unsigned char nl; //младшая тетрада двоично-десятичного числа
    nl=n%10;
    n=n/10;
    n=(n<<4)+nl;
    return n;
}

//расчет длительности импульса и вывод полученного результата на индикатор
//расчет длительности импульса производится по формуле:
//T=nimp*1024*10/(8000000)
//где
// nimp - число импульсов, подсчитанное таймером
// 1024 - коэффициент прескалера
// 8000000 - тактовая частота генератора 8 МГц
// 10 - преобразование числа к виду #.#
void CalcTimp(unsigned int nimp)
{
    unsigned int T; //время импульса внешнего сигнала
    T=nimp/781;
    PORTB=PrintBCD99(T) ; //переводим полученное число к двоично-десятичному формату
    //и выдаем полученный результат на индикатор
}

```

Рис. 2.3.2. Текст программы эксперимента Exp\_2\_3.

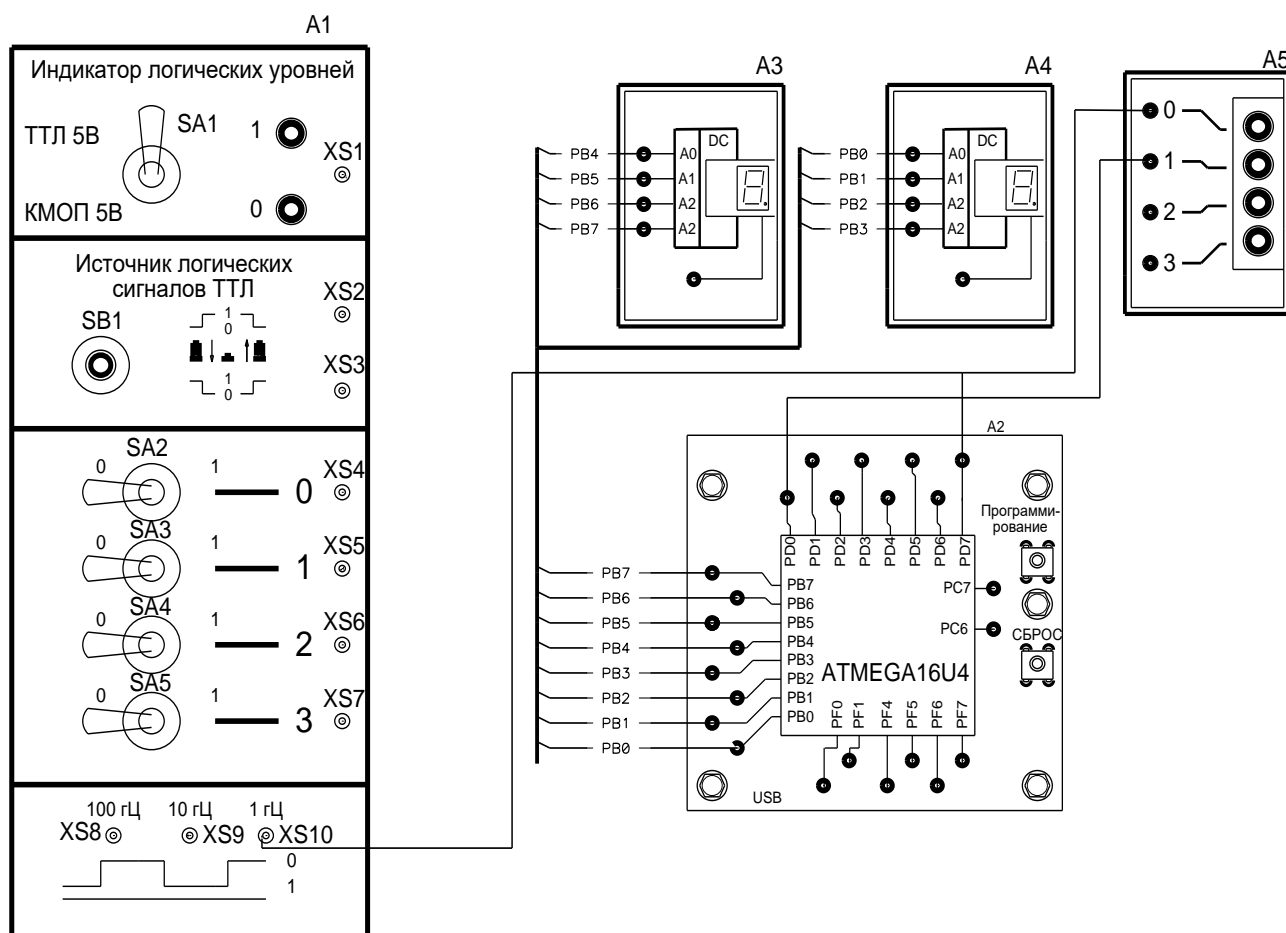
- Откомпилируйте программу
- Загрузите программу в микроконтроллер с помощью программатора FLIP.
- Переведите программу в рабочий режим, нажав кнопку **СБРОС**.
- Установите тумблер SA5 в “1”, режим измерения частоты сигнала.
- Микроконтроллер показывает частоту сигнала в Гц. Подавая сигналы частотой 1, 10, 100 Гц, проверьте правильность индицируемой частоты.
- Измените программу, таким образом, чтобы повысить точность индицируемой частоты (при индицировании частоты 1 Гц после старшего разряда, должна появляться десятичная точка). Обеспечьте, чтобы при превышении частоты 100 Гц не наступал режим перегрузки.
- Переключите тумблер SA5 в “0”. Проверьте правильность измерения длительности импульса на частотах 1 и 10 Гц.
- Измените программу таким образом, чтобы и на частоте 100 Гц, можно было определить длительность импульсов.
- Выключите блок БИЦУ тумблером **СЕТЬ**.

## **2.4. Счетчик с программируемым коэффициентом деления на базе таймера**

Цель: Научиться использовать таймер для подсчета числа импульсов внешнего сигнала.

- Рекомендуемая схема соединений и ее описание.
- Перечень аппаратуры и программных средств
- Указания по проведению эксперимента

### Рекомендуемая схема соединений и ее описание



**Рис.2.4.1. Рекомендуемая схема соединений**

На вывод PD7, запрограммированный как вход таймера 0, поступают импульсы с генератора. При низкой частоте импульсов (1 Гц) их можно также увидеть на 0 светодиоде. Таймер отсчитывает заданное число импульсов и после этого переключает вывод OC0V (вывод PD0) в противоположное состояние. Переключение вывода PD0 можно наблюдать с помощью светодиода 1. На индикатор выводится число импульсов в регистре таймера TCNT0, считающего приходящие импульсы.

## Перечень аппаратуры и программных средств

*Таблица 2.4.1.*

Обозначение	Наименование	Тип
A1	Блок БИЦУ	219
A2	Миниблок “Микроконтроллер ATmega16U4”	Набор миниблоков 600.25
A3, A4	Миниблоки семисегментных индикаторов	Набор миниблоков 600.11
A5	Миниблок светодиодов	Набор миниблоков 600.11
	Однофазный источник питания ОИП1	218.8
	Компьютер или ноутбук	
	Программа Atmel Studio	Версия 6.1

### Указания по проведению эксперимента

- Подключите компьютер и блок БИЦУ (А1) к однофазному источнику питания ОИП, а сам источник к электрической сети.
- Подключите миниблок А2 к компьютеру, используя кабель miniUSB.
- Соберите схему в соответствии с рекомендуемой схемой соединений рис.2.4.1.
- Включите автомат на лицевой панели однофазного источника питания и выключатель СЕТЬ на лицевой панели блока БИЦУ.
- Загрузите в компьютер программу Atmel Studio.
- Загрузите программы эксперимента Program microcontrollers\Exp\_2\_4\Exp\_2\_4.
- Откройте файл Exp\_2\_4.c. Исходный текст программы представлен на рис. 2.4.2.

```

/* Exp_2_4 Счетчик с программируемым коэффициентом деления на базе таймера
 * Exp_2_4.c Программа, формирующая на базе таймера 0, счетчик с заданным коэффициентом
 * деления. Счетные импульсы поступают на вход таймера T0 (вывод PD7). Таймер отсчитывает
 * заданное число импульсов, после чего переключает состояние своего выхода OC0B (вывод
 * PD0) на противоположное и сбрасывает счетчик в 0. Вывод PD7 (вход таймера T0) подключен
 * к светодиоду 0 миниблока светодиодов А5. Вывод PD0 (выход OC0B таймера T0) подключен к
 * светодиоду 1. При низкой частоте импульсов (1Гц) можно визуально наблюдать поступление
 * импульсов. На семисегментные индикаторы выводится реальное состояние счетного регистра
 * TCNT0.
 */
#include <avr/io.h>
#include "Exp_2_4.h"
int main(void)
{
    DDRB=0xFF; //выходы управления семисегментными индикаторами
    DDRD|=(1<<PD0); //выходы OC0B таймера 0
    TCCR0A=0x12; //переключение вывода OC0B в противоположное состояние по
                //окончании счета. Задание режима работы CTC
    TCCR0B=0x7; //подсчет сигналов с внешнего вывода T0 с переключением по
               //переднему фронту
    OCR0B=OCR0A=5; //задание коэффициента счета равным 6=5+1
                 //таймер будет считать с 0 до 5

    while(1)
    {
        IndNImp(TCNT0); //подпрограмма выводит на семисегментный индикатор текущее
                       //состояние счетчика
    }
    //подпрограмма выводит на семисегментный индикатор текущее состояние счетчика
    //n - число импульсов
    void IndNImp(unsigned char n)
    {
        PORTB=PrintBCD99(n); //переводим полученное число к двоично-десятичному формату
                             //и выдаем полученный результат на семисегментные
                             //индикаторы
    }
    //программа преобразует число к двоично-десятичному числу, упакованному в один байт
    //n – преобразуемое число (в диапазоне от 0...99)
    //выход: двоично-десятичное число
    unsigned char PrintBCD99(unsigned char n)
    {
        unsigned char n1; //младшая тетрада двоично-десятичного числа
        n1=n%10;
        n=n/10;
        n=(n<<4)+n1;
        return n;
    }
}

```

**Рис. 2.4.2. Текст программы эксперимента Exp\_2\_4.**

- Откомпилируйте программу
- Загрузите программу в микроконтроллер с помощью программатора FLIP.
- Переведите программу в рабочий режим, нажав кнопку **СБРОС**.
- Подсчитайте по светодиоду 0 число импульсов, приходящих в промежутке между переключениями светодиода 1. Сравните, полученное число импульсов, с заданным коэффициентом счета.
- По показаниям семисегментного индикатора определите диапазон счета таймера 0.
- Измените коэффициент счета таймера. Загрузите программу в микроконтроллер и проверьте работу таймера при новом коэффициенте.
- Подключите к свободному выводу микроконтроллера тумблер SA2 блока A1. Запрограммируйте микроконтроллер таким образом, чтобы при поступлении с тумблера SA2 сигнала логической "1" сбрасывался, а при достижении заданного коэффициента счета, таймер останавливался в этом состоянии.
- Выключите блок БИЦУ тумблером **СЕТЬ**.



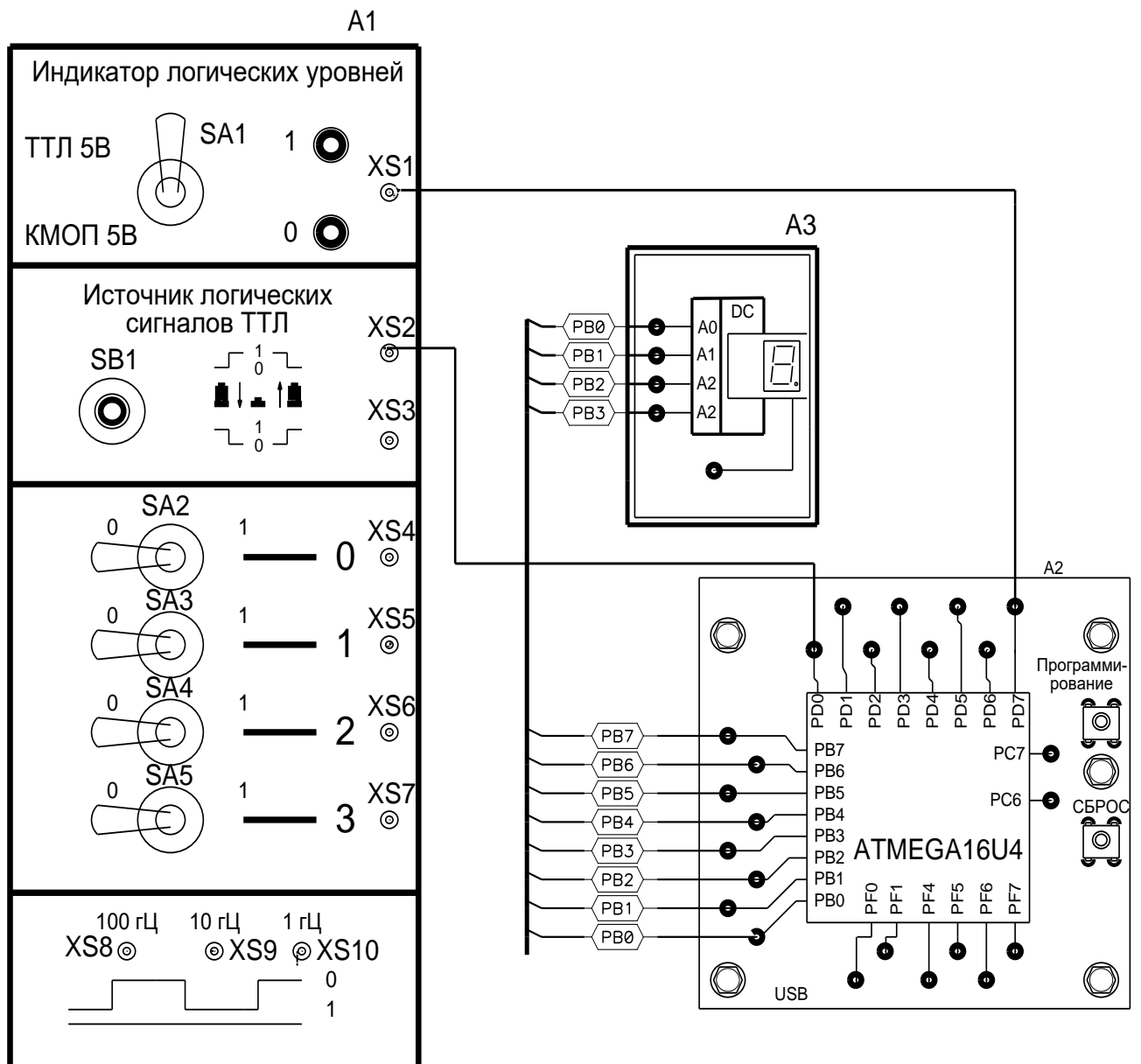
---

### **3. Использование прерываний при программировании микроконтроллера**

Цель: Научиться использовать прерывания для выполнения действий критичных к времени выполнения.

- Рекомендуемая схема соединений и ее описание
- Перечень аппаратуры и программных средств
- Указания по проведению эксперимента

### Рекомендуемая схема соединений и ее описание



**Рис.3.1. Рекомендуемая схема соединений**

В первой части эксперимента "Программирование внешних прерываний микроконтроллера" кроме микроконтроллера A2 используется кнопка SB1 из блока БИЦУ A1 для формирования прерывания и семисегментный индикатор A3 для вывода места расположения прерывания в основном цикле программы. При приходе очередного прерывания состояние светодиодов индикатора логических уровней меняется на противоположное.

Во второй части программы "Использование прерываний таймера", кроме микроконтроллера A2, используется светодиод "индикатора логических уровней" из блока БИЦУ A1 и миниблок семисегментного индикатора A3. Светодиод меняет свое состояние при каждом прерывании таймера. На семисегментный индикатор выводится расположение прерывания в основном цикле программы.

---

**Перечень аппаратуры и программных средств***Таблица 3.1.*

Обозначение	Наименование	Тип
A1	Блок БИЦУ	219
A2	Миниблок “Микроконтроллер ATMEGA16U4”	Набор миниблоков 600.25
A3	Миниблок семисегментного индикатора	Набор миниблоков 600.11
	Однофазный источник питания ОИП1	218.8
	Компьютер или ноутбук	
	Программа Atmel Studio	Версия 6.1

### Указания по проведению эксперимента

- Подключите компьютер и блок БИЦУ (А1) к однофазному источнику питания ОИП, а сам источник к электрической сети.
- Соберите схему в соответствии с рекомендуемой схемой соединений рис.3.1.
- Подключите миниблок А2 к компьютеру, используя кабель miniUSB.
- Включите автомат на лицевой панели однофазного источника питания и выключатель СЕТЬ на лицевой панели блока БИЦУ.
- Загрузите в компьютер программу Atmel Studio.
- Загрузите решение Program microcontrollers\Exp\_3\Exp\_3. Данное решение состоит из двух проектов:
  - 1) Exp\_3\_1 “Программирование внешних прерываний”
  - 2) Exp\_3\_2 “Использование прерываний таймера”
- Exp\_3\_1 “Программирование внешних прерываний” Программа эксперимента представлена на рис.3.2. При инициализации программы разрешается прерывание по приходу переднего фронта сигнала на выводе Int0 (вывод PD0) и инициализируются выходы управления семисегментным индикатором. Пока нет прерывания, программа выполняет бесконечный цикл while, имитирующий основной цикл программы. Этот цикл состоит из 3-х циклов for, внутри которых переменной n присваивается значение с номером цикла. При возникновении прерывания (при нажатии кнопки SB0) управление сразу передается программе прерывания ISR(INT0\_vect), которая выводит номер цикла for в котором произошло прерывание.
- Сделайте Exp\_3\_1 стартовым проектом, если он таковым не является (Переведите курсор на название проекта, нажмите правую кнопку мыши и выберите из всплывающего меню пункт Set as StartUp Project (Установить как стартовый проект)).
- Откомпилируйте программу и загрузите программу в контроллер с помощью программатора FLIP.
- Нажимайте кнопку SB0 и по цифре на экране семисегментного индикатора А3, определите, что прерывания могут происходить в любой точке программы. По переключению светодиода определите на каком фронте происходит прерывание.
- Измените условия возникновения прерывания (по заднему фронту, по уровню нуля или при переключении). Проконтролируйте, как поведет себя микроконтроллер в этих режимах.

```

/* Exp_3 Использование прерываний при программировании микроконтроллера
Exp_3_1.c Программа при приходе переднего фронта сигнала на вывод INT0 (PD0) вызывает прерывание.
Внутри прерывания состояние красного светодиода “индикатора логических уровней” меняется
на противоположное и на семисегментный индикатор выводится номер цикла, в котором произошло данное прерывание.
*/

#include <avr/io.h>
#include <avr/interrupt.h>

volatile char n=0; //контролирует часть программы в которой произошло прерывание

int main(void)
{
    DDRD|=(1<<PD7); //выводы управления светодиодом
    DDRB=0xff; //выводы управления семисегментными индикаторами
    EICRA=0x3; //прерывание INT0 будет происходить по переднему фронту
    EIMSK|=(1<<INT0); //разрешение прерывания INT0
    unsigned char i; //счетчик
    sei(); //разрешение прерывания
    while(1)
    {
        for(i=0;i<255;i++)
        { //первая часть программы
            n=1;
        }
        for(i=0;i<255;i++)
        { //вторая часть программы
            n=2;
        }
        for(i=0;i<255;i++)
        { //третья часть программы
            n=3;
        }
    }
}

//прерывание при приходе переднего фронта на вывод Int0 (PD0) [т.е. при нажатии кнопки SB0]
ISR(INT0_vect)
{
    PORTB=n; //вывод на семисегментный индикатор номера части из которой произошло прерывание

    if (PORTD & (1<<PD7))
    { //красный светодиод горит
        PORTD&=~(1<<PD7); //тушим красный светодиод
    }
    else
    { //красный светодиод погашен
        PORTD|=(1<<PD7); //зажигаем красный светодиод
    }
}

```

**Рис.3.2. Текст программы проекта 3 \_1.**

- Ехр\_3\_2 “Использование прерываний таймера”.
- При инициализации программа инициализирует таймер для работы в обычном режиме без вывода ШИМ сигнала. Разрешает переполнение таймера по прерыванию. Производит инициализацию выводов семисегментного индикатора и программирует вывод PD7 управления светодиодом, как вывод, работающий на выход. Аналогично предыдущей части, организует бесконечный цикл while с тремя циклами for. При наступлении прерывания таймера по переполнению вызывается программа прерывания таймера ISR(TIMER1\_OVF\_vect) в которой состояние светодиода меняется на противоположное и на индикатор выводится номер цикла for, в котором произошло прерывание.
- Сделайте Ехр\_3\_2 стартовым проектом (Переведите курсор на название проекта. Нажмите правую кнопку мыши и выберите из всплывающего меню пункт Set as StartUp Project (Установить как стартовый проект)).
- Откомпилируйте программу.
- Загрузите программу в контроллер с помощью программатора Flip.
- По миганию светодиода убедитесь, что прерывания происходят равномерно, через заданные промежутки времени.
- Измените длительности циклов for. Проверьте, как длительность циклов for влияет на равномерность прерываний и их длительность.
- По семисегментному индикатору убедитесь, что прерывания могут происходить в любом месте основного цикла программы.
- Измените программу, задав значение регистрам сравнения таймера OCR1A и OCR1B и разрешив прерывания по сравнению для таймера. Добавьте программы по обслуживанию прерываний ISR(TIMER1\_COMPA\_vect) и ISR(TIMER1\_COMPB\_vect), так, чтобы каждая из них зажигала свой светодиод (в миниблоке светодиодов) в момент прихода прерывания. Измените программу ISR(TIMER1\_OVF\_vect), чтобы она зажигала, а не переключала светодиод. Добавьте прерывание по выводу INTO (аналогично первой части), которое будет тушить светодиоды зажигаемые таймерами.
- Нажимайте кнопку SB0, и по светодиодам определите в какой последовательности, происходят прерывания таймеров. Сравните полученные результаты с теоретическими.

```

/* Exp_3 Использование прерываний при программировании микроконтроллера
Exp_3_2.c Программа инициализирует таймер 1 для работы в режиме с прерыванием по переполнению.
Внутри прерывания состояние красного светодиода “индикатора логических уровней” меняется
на противоположное и на семисегментный индикатор выводится номер цикла for основного цикла
программы, из которого произошло данное прерывание.
*/

#include <avr/io.h>
#include <avr/interrupt.h>

volatile char n=0;           //показывает часть программы, в которой произошло
                             //прерывание

int main(void)
{
    TCCR1A=0;                //таймер работает в обычном режиме, с коэффициентом
    TCCR1B=0b00000011;       //прескалера 64
    TIMSK1|=(1<<TOIE1);      //разрешение прерывания по переполнению таймера 1
    DDRB=0x0f;               //выводы управления семисегментным индикатором, работают
                             //на выход
    DDRD|=(1<<PD7);           //вывод управления светодиодом, работает на выход
    unsigned char i=0;        //счетчик циклов
    sei();                    //разрешение прерывания
    while(1)
    {
        for(i=0;i<255;i++)
        { //первая часть программы
            n=1;
        }
        for(i=0;i<255;i++)
        { //вторая часть программы
            n=2;
        }
        for(i=0;i<255;i++)
        { //третья часть программы
            n=3;
        }
    }
}

//прерывание при переполнении таймера 1
ISR(TIMER1_OVF_vect)
{
    PORTB=n;                  //вывод номера части из которой произошло прерывание
    if(PORTD & (1<<PD7))      //меняем состояние светодиода на противоположное
    {
        PORTD&=~(1<<PD7);
    }
    else
    {
        PORTD|=(1<<PD7);
    }
}

```

Рис. 3.3. Текст программы проекта 3\_2

---

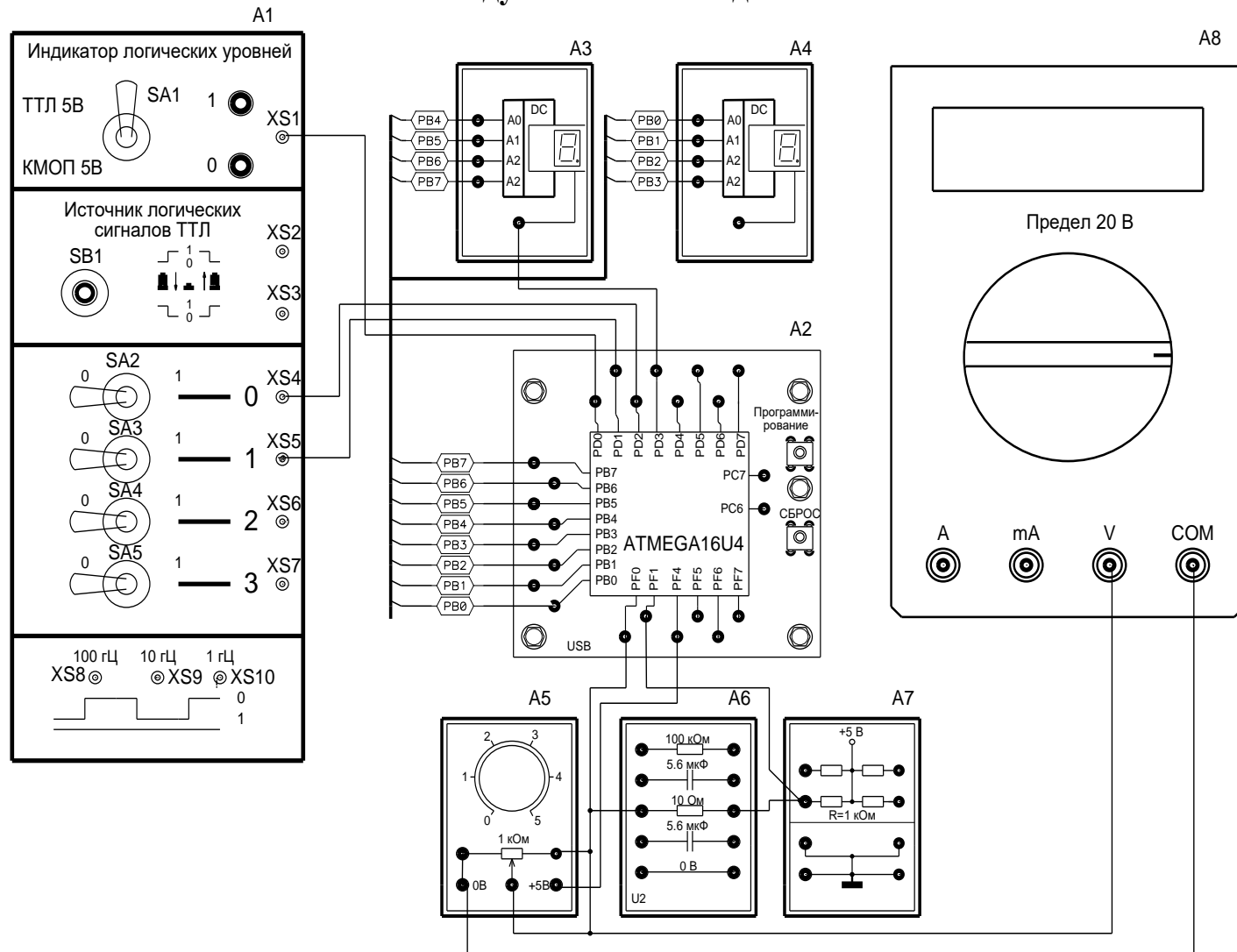
## 4. Программирование АЦП микроконтроллера

Цель: Научиться использовать встроенный в микроконтроллер АЦП для ввода значений аналоговых сигналов.

- Рекомендуемая схема соединений и ее описание
- Перечень аппаратуры и программных средств
- Указания по проведению эксперимента



### Рекомендуемая схема соединений и ее описание



**Рис.4.1. Рекомендуемая схема электрических соединений**

В первом проекте эксперимента “Программирование АЦП при вводе данных не дифференциального сигнала” кроме микроконтроллера А2, используются тумблеры SA2...SA3 из блока БИЦУ А1.1 для изменения режима работы АЦП, Семисегментные индикаторы А3, А4 предназначены для индикации напряжения, измеренного с помощью АЦП, в Вольтах. Схема из переменного резистора миниблока А5, служит регулятором, задающим режим работы резистора 1 кОм миниблока А7. Резистор 10 Ом миниблока А6 выполняет роль шунта, для измерения тока через нагрузку. В данном эксперименте измеряется величина напряжения, поступающая с движка потенциометра миниблока А5 относительно общего провода. Измеряемый сигнал поступает на канал ADC0 (вывод PF0) микроконтроллера А2. С помощью мультиметра производится проверка правильности вводимых с помощью АЦП данных. Тумблер SA2, через вывод PD2 микроконтроллера, позволяет изменять выравнивание полученного результата АЦП:

“0” – правое выравнивание результата;

“1” – левое выравнивание результата.

Тумблер SA3, через вывод PD1 микроконтроллера, позволяет изменять величину опорного напряжения АЦП:

“0” – опорное напряжение АЦП равно напряжению питания микроконтроллера на выводе AVcc;

“1” – опорное напряжение АЦП равно 2.56 В.

Во втором проекте эксперимента “Программирование АЦП при измерении дифференциального сигнала” кроме микроконтроллера А2 используются семисегментные индикаторы А3, А4 для индикации в Вольтах результатов, полученных с АЦП. Схема из резисторов миниблоков А5...А7 аналогична предыдущему проекту. Измеряемый сигнал (напряжение нагрузки) поступает на канал ADC4:ADC1 (выводы PF4:PF1) микроконтроллера А2. С помощью мультиметра производится проверка правильности вводимых с помощью АЦП данных. Красный светодиод индикатора логических уровней блока БИЦУ А1 индицирует отрицательный знак числа.

В третьем проекте эксперимента “Программирование АЦП с использованием предварительного усилителя сигнала”, измеряемый сигнал поступает с шунта на дифференциальный вход АЦП ADC1:ADC0 (выводы PF1:PF0). Роль шунта выполняет резистор номиналом 10 Ом из миниблока А6. Перед измерением АЦП сигнал усиливается с помощью усилителя микроконтроллера. Тумблер SA3 из блока БИЦУ А1 позволяет менять коэффициент усиления:

“0” – коэффициент усиления 10;

“1” – коэффициент усиления 40.

Тумблер SA2 из блока БИЦУ А1 меняет величину опорного напряжения АЦП:

“0” – опорное напряжение АЦП равно напряжению питания микроконтроллера на выводе AVcc ;

“1” – опорное напряжение АЦП равно 2.56 В.

Семисегментные индикаторы А3,А4 индицируют результаты АЦП в мВ. Схема из резисторов миниблоков А5...А7 аналогична первому проекту. С помощью мультиметра производится проверка правильности вводимых с помощью АЦП данных.

## Перечень аппаратуры и программных средств

*Таблица 4.1.*

Обозначение	Наименование	Тип
A1	Блок БИЦУ	219
A2	Миниблок “Микроконтроллер ATMEGA16U4”	Набор миниблоков 600.25
A3, A4	Миниблоки семисегментных индикаторов	Набор миниблоков 600.11
A5	Миниблок потенциометра	Набор миниблоков 600.25
A6	Миниблок RC элементов	Набор миниблоков 600.25
A7	Миниблок резисторов	Набор миниблоков 600.11
A8	Мультиметр Mastech UT-53 (или аналогичный)	1403
	Однофазный источник питания ОИП1	218.8
	Компьютер или ноутбук	
	Программа Atmel Studio	Версия 6.1

## Указания по проведению эксперимента

- Подключите компьютер и блок БИЦУ (А1) к однофазному источнику питания ОИП, а сам источник к электрической сети.
- Соберите схему в соответствии с рекомендуемой схемой соединений рис.4.1.
- Подключите миниблок А2 к компьютеру, используя кабель miniUSB.
- Включите автомат на лицевой панели однофазного источника питания и выключатель СЕТЬ на лицевой панели блока БИЦУ.
- Загрузите в компьютер программу Atmel Studio.
- Загрузите решение Program microcontrollers\Exp\_4\Exp\_4. Данное решение состоит из трех проектов:
  - 1) Exp\_4\_1: "Программирование АЦП при вводе данных не дифференциального сигнала";
  - 2) Exp\_4\_2: "Программирование АЦП при измерении дифференциального сигнала";
  - 3) Exp\_4\_3: "Программирование АЦП с использованием предварительного усилителя";
- Exp\_4\_1: "Программирование АЦП при вводе данных не дифференциального сигнала". Программа эксперимента и ее описание представлена на рис. 4.2.

```

/* Exp_4 "Программирование АЦП микроконтроллера"
 * Exp_4_1 "Программирование АЦП при вводе данных недифференциального сигнала"
 * АЦП микроконтроллера программируется для ввода данных по каналу ADC0 (вывод PF0)
 * с движка потенциометра A5, выполняющего роль линейного регулятора. Микроконтроллер
 * пересчитывает полученный результат в Вольты и выводит полученный результат на
 * семисегментный индикатор. Мультиметр позволяет судить о правильности полученного
 * результата.
 * Тумблер SA2 задает выравнивание полученного результата АЦП:
 * "0" - правое выравнивание результата;
 * "1" - левое выравнивание результата.
 * Тумблер SA3 задает величину опорного напряжения
 * "0"- опорное напряжение микроконтроллера равно напряжению питания микроконтроллера на
 * выводе AVcc;
 * "1" - опорное напряжение равно 2.56 В.
 */

#include <avr/io.h>
#include "Exp_4_1.h"

#define K_V_R_256 40 //коэффициент приведения результата к В*10 при при правом
//выравнивании и опорном напряжении 2.56 В
#define K_V_L_256 2560 //коэффициент приведения результата к В*10 при при левом
//выравнивании и опорном напряжении 2.56 В
#define K_V_R_500 20 //коэффициент приведения результата к В*10 при при правом
//выравнивании и опорном напряжении 5 В (напряжение питания)
#define K_V_L_500 1310 //коэффициент приведения результата к В*10 при при левом
//выравнивании и опорном напряжении 5 В (напряжение питания)

int main(void)
{
    DDRB=0xff; //выводы управления семисегментным индикатором работают на выход
    DDRD|=(1<<PD3); //выход управления точкой семисегментного индикатора
    PORTD|=(1<<PD3); //зажигаем точку семисегментного индикатора
    ADMUX=0xC0; //при инициализации АЦП выбираем опорное напряжение 2.56 В,
//правое выравнивание результата, и канал ADC0
    ADCSRA=0xD2; //первый запуск АЦП без разрешения прерываний и с тактовой
//частотой 250 кГц
    unsigned int k_trans; //коэффициент преобразования показаний АЦП в В;
    k_trans=K_V_R_256; //присваиваем значение коэффициенту преобразования в
//соответствии с установками АЦП;
    int ResADC; //введенные данные с ADC

```

```

while(1)
{
    if (ADCSRA & (1<<ADIF))
    { //закончилось очередное преобразование АЦП
        ResADC=ADC;
        PORTB=PrintBCD99(ResADC/k_trans);
        if(PIND & (1<<PD2))
        { //тумблер SA2 в состоянии логической "1" (левое выравнивание результата)
            if(PIND & (1<<PD1))
            { //тумблер SA3 в "1" (опорное напряжение 2.56 В)
                ADMUX=0xE0; //задаем выбранный режим АЦП
                k_trans=K_V_L_256;
            }
            else
            { //тумблер SA3 в "0" (опорное напряжение 5 В)
                ADMUX=0x60; //задаем выбранный режим АЦП
                k_trans=K_V_L_500;
            }
        }
        else
        { //тумблер SA2 в состоянии логического "0" (правое выравнивание результата)
            if(PIND & (1<<PD1))
            { //тумблер SA3 в "1" (опорное напряжение 2.56 В)
                ADMUX=0xC0; //задаем выбранный режим АЦП
                k_trans=K_V_R_256;
            }
            else
            { //тумблер SA3 в "0" (опорное напряжение 5 В)
                ADMUX=0x40; //задаем выбранный режим АЦП
                k_trans=K_V_R_500;
            }
        }
        ADCSRA=0xD2; //запускаем следующее преобразование АЦП
    }
}

//программа преобразует число к двоично-десятичному числу, упакованному в один байт
//n – преобразуемое число (в диапазоне от 0...99)
//выход: двоично-десятичное число
unsigned char PrintBCD99(unsigned char n)
{
    unsigned char n1; //младшая тетрада двоично-десятичного числа
    n1=n%10;
    n=n/10;
    n=(n<<4)+n1;
    return n;
}

```

**Рис. 4.2. Текст программы проекта Exp\_4\_1.**

- Сделайте проект стартовым проектом (Переведите курсор на название проекта. Нажмите правую кнопку мыши и выберите из всплывающего меню пункт Set as StartUp Project (Установить как стартовый проект)).
- Откомпилируйте программу.
- Загрузите программу в микроконтроллер с помощью программатора FLIP.
- Снимите данные, получаемые с помощью АЦП, заполнив таблицу 4.2. Измерение входного напряжения  $U_{вх}$  производите с помощью мультиметра.

Таблица 4.2

U <sub>ВХ</sub>	Показания семисегментного индикатора, В			
	U <sub>оп</sub> =5В, правое (SA3=0, SA2=0)	U <sub>оп</sub> =2.56 В, правое (SA3=1, SA2=0)	U <sub>оп</sub> =2.56 В, левое (SA3=1, SA2=1)	U <sub>оп</sub> =5В, левое (SA3=0, SA2=1)

- Оцените как величина опорного напряжения и выравнивание влияют на точность полученного результата и его стабильность. При необходимости подберите более точные коэффициенты преобразования.
- Эксп\_4\_2: “Программирование АЦП при вводе данных дифференциального сигнала”. Программа эксперимента и ее описание представлена на рис. 4.3.

```

/* Эксп_4 "Программирование АЦП микроконтроллера"
 * Эксп_4_2 "Программирование АЦП при вводе данных дифференциального сигнала"
 * АЦП программируется для ввода данных по каналу ADC4:ADC1 (выводы PF4:PF1)
 * с резистора 1 кОм миниблока А7, выполняющего роль нагрузочного резистора схемы.
 * Регулирование напряжения на резисторе осуществляет потенциометр миниблока А5.
 * Микроконтроллер пересчитывает полученный результат в Вольты и выводит полученный
 * результат на семисегментные индикаторы. Мультиметр позволяет судить о правильности
 * полученного результата.
 * Красный светодиод индикатора логических уровней блока БИЦУ А1 показывает
 * отрицательный знак числа
 */

#include <avr/io.h>
#include "Exp_4_2.h"

#define K_V_R_500 655 //коэффициент приведения результата к В*10 при правом
                      //выравнивании и опорном напряжении 5 В (напряжение питания)

int main(void)
{
    DDRB=0xff; //выводы управления семисегментным индикатором работают на выход
    DDRD|=(1<<PD3)+(1<<PD0); //выход управления точкой семисегментного индикатора и
                             //светодиодом индикатора логических уровней блока БИЦУ
    PORTD|=(1<<PD3); //зажигаем точку семисегментного индикатора
    ADMUX=0x74; //при инициализации АЦП выбираем опорное напряжение равное
               //напряжению питания (~5В), левое выравнивание результата,
               //и канал ADC4:ADC1 с коэффициентом усиления 1
    ADCSRA=0xD2; //первый запуск АЦП без разрешения прерываний и с тактовой
                //частотой 250 кГц
    int k_trans; //коэффициент преобразования показаний АЦП в В;
    k_trans=K_V_R_500; //присваиваем значение коэффициенту преобразования в
                      //соответствии с установками АЦП;
    int ResADC; //введенные данные с ADC
    while(1)
    {
        if (ADCSRA &(1<<ADIF))
        { //закончилось очередное преобразование АЦП
            ResADC=ADC;

```

```

        if(ResADC<0)
        { //отрицательное число
          ResADC=-ResADC; //меняем знак числа
          PORTD|=(1<<PD0); //зажигаем красный светодиод, имитирующий знак минус
        }
        else
        {
          PORTD&=~(1<<PD0); //тушим красный светодиод
        }
        PORTB=PrintBCD99(ResADC/k_trans);
        ADCSRA=0xD2; //запускаем следующее преобразование АЦП
        for(unsigned int i=0; i<5000;i++) ; //цикл имитирующий основную
                                           //часть программы
      }
    }

//программа преобразует число к двоично-десятичному числу, упакованному в один байт
//n – преобразуемое число (в диапазоне от 0...99)
//выход: двоично-десятичное число
unsigned char PrintBCD99(unsigned char n)
{
    unsigned char n1; //младшая тетрада двоично-десятичного числа
    n1=n%10;
    n=n/10;
    n=(n<<4)+n1;
    return n;
}

```

**Рис. 4.3. Текст программы проекта Exp\_4\_2.**

- Сделайте проект стартовым проектом (Переведите курсор на название проекта. Нажмите правую кнопку мыши и выберите из всплывающего меню пункт Set as StartUp Project (Установить как стартовый проект)).
- Откомпилируйте программу.
- Загрузите программу в микроконтроллер с помощью программатора FLIP.
- Снимите данные, получаемые с помощью АЦП, заполнив таблицу 4.3. Измерение входного напряжения  $U_{вх}$  производите с помощью мультиметра.
- Измените полярность входного сигнала, поменяв местами подключения к выводам PF4, PF1.
- По состоянию красного светодиода индикатора логических уровней (горящее состояние имитирует знак минус), убедитесь в изменении полярности сигнала. Установите значения  $U_{вх}$ , аналогичные предыдущему опыту, и заполните последний столбец таблицы 4.3. Сравните модули полученных значений.

$U_{вх}$ , В	Показания семисегментного индикатора, В	
	PF4:PF1	PF1:PF4

- Верните выводы PF4, PF1 в исходное состояние, в соответствии с рис. 4.1.
- Эксп\_4\_3: “Программирование АЦП с использованием предварительного усилителя”. Программа проекта и ее описание представлена на рис. 4.4.

```

/* Эксп_4 "Программирование АЦП микроконтроллера"
 * Эксп_4_3 "Программирование АЦП с использованием предварительного усилителя сигнала"
 * АЦП программируется для ввода данных по каналу ADC1:ADC0 (выводы PF1:PF0)
 * с резистора 10 Ом миниблока А6, выполняющего роль шунта. Регулирование тока шунта
 * производится потенциометр миниблока А5. Микроконтроллер пересчитывает полученный
 * результат в мВ и выводит полученный результат на семисегментные индикаторы. Мультиметр
 * позволяет судить о правильности полученного результата.
 * Тумблер SA2 блока БИЦУ А! переключает величину опорного напряжения:
 * "0" - опорное напряжение равно напряжению питания микроконтроллера (+5 В)
 * "1" - опорное напряжение равно внутреннему опорному напряжению 2.56 В
 * Тумблер SA3 блока БИЦУ А! переключает величину коэффициента усиления:
 * "0" - коэффициент усиления 10;
 * "1" - коэффициент усиления 40.
 */
#include <avr/io.h>
#include "Exp_4_3.h"
#define K_V_10_500 1 //коэффициент приведения результата к мВ при коэффициенте
// усиления 10 и опорном напряжении +5 В
#define K_V_40_500 4 //коэффициент приведения результата к мВ при коэффициенте
// усиления 40 и опорном напряжении +5 В
#define K_V_10_256 2 //коэффициент приведения результата к мВ при коэффициенте
// усиления 10 и опорном напряжении +2.56 В
#define K_V_40_256 8 //коэффициент приведения результата к мВ при коэффициенте
// усиления 40 и опорном напряжении +2.56 В

int main(void)
{
    DDRB=0xff; //выводы управления семисегментным индикатором работают на
//выход
    DDRD|=(1<<PD3); //выход управления точкой семисегментного индикатора
    ADMUX=0xC9; //при инициализации АЦП выбираем опорное напряжение
//2.56 В, и коэффициент усиления 10
    ADCSRA=0xD2; //первый запуск АЦП без разрешения прерываний и с тактовой
//частотой 250 кГц
    unsigned int k_trans; //коэффициент преобразования показаний АЦП в В;
    k_trans=K_V_10_256; //присваиваем значение коэффициенту преобразования в
//соответствии с установками АЦП;
//введенные данные с ADC

    int ResADC;
    while(1)
    {
        if (ADCSRA &(1<<ADIF))
        { //закончилось очередное преобразование АЦП
            ResADC=ADC;
            PORTB=PrintBCD99(ResADC/k_trans);
            if(PIND & (1<<PD2))
            { //тумблер SA2 в состоянии логической "1" (опорное напряжение 2.56 В)
                if(PIND & (1<<PD1))
                { //тумблер SA3 в "1" (коэффициент усиления 40)
                    ADMUX=0xC6; //задаем выбранный режим АЦП
                    ADCSRB=0x20;
                    k_trans=K_V_40_256;
                }
                else
                { //тумблер SA3 в "0" (коэффициент усиления 10)
                    ADMUX=0xC9; //задаем выбранный режим АЦП
                    ADCSRB=0;
                    k_trans=K_V_10_256;
                }
            }
        }
    }
}

```



```

else
{
    //тумблер SA2 в состоянии логического "0" (опорное напряжение 5В)
    if(PIND & (1<<PD1))
    {
        //тумблер SA3 в "1" (коэффициент усиления 40)
        ADMUX=0x46;           //задаем выбранный режим АЦП
        ADCSRB=0x20;
        k_trans=K_V_40_500;

    }
    else
    {
        //тумблер SA3 в "0" (коэффициент усиления 10)
        ADMUX=0x49;           //задаем выбранный режим АЦП
        ADCSRB=0;
        k_trans=K_V_10_500;
    }
}
ADCSRA=0xD2;                //запускаем следующее преобразование АЦП
for(unsigned int i=0;i<50000;i++) ; //цикл, имитирующий основную
//программу
}
}

//программа преобразует число к двоично-десятичному числу, упакованному в один байт
//n – преобразуемое число (в диапазоне от 0...99)
//выход: двоично-десятичное число
unsigned char PrintBCD99(unsigned char n)
{
    unsigned char n1;        //младшая тетрада двоично-десятичного числа
    n1=n%10;
    n=n/10;
    n=(n<<4)+n1;
    return n;
}

```

**Рис. 4.4. Текст программы проекта Exr\_4\_3.**

- Сделайте проект стартовым проектом (Переведите курсор на название проекта. Нажмите правую кнопку мыши и выберите из всплывающего меню пункт Set as StartUp Project (Установить как стартовый проект)).
- Откомпилируйте программу.
- Загрузите программу в микроконтроллер с помощью программатора FLIP.
- Снимите данные, получаемые с помощью АЦП, заполнив таблицу 4.3. Измерение входного напряжения  $U_{вх}$  производите с помощью мультиметра.
- Оцените точность и стабильность получаемых результатов для различных режимов работы.
- Для повышения стабильности получаемых результатов, доработайте программу, выводя на индикатор, не мгновенное значение, а среднее значение из  $n$  результатов (предположим,  $n=16$ ). Оцените уменьшение “дребезга”.

Таблица 4.3

U <sub>вх</sub>	Показания семисегментного индикатора, В			
	U <sub>оп</sub> =5 В, правое (SA3=0, SA2=0)	U <sub>оп</sub> =2.56 В, правое (SA3=1, SA2=0)	U <sub>оп</sub> =2.56 В, левое (SA3=1, SA2=1)	U <sub>оп</sub> =5 В, левое (SA3=0, SA2=1)

- Объедините три программы в одну, так чтобы АЦП последовательно переключал ввод. Тумблеры SA2, S3 задают результат, выводимый на индикатор. При написании программы необходимо учитывать, что для переключения с одного канала на другой АЦП требуется дополнительное время. Поэтому, после переключения на другой канал, необходимо вводить дополнительную выдержку времени или использовать двойной ввод значения с каждого из каналов. При этом правильным будет только второе введенное значение.
- Снимите полученные результаты в таблицу 4.4. Сравните их с результатами таблиц 4.1...4.3.

Таблица 4.4

U <sub>вх</sub>	Показания семисегментного индикатора		
	U <sub>ADC0</sub> , В (напряжение потенциометра)	U <sub>ADC4:ADC1</sub> , В (напряжение нагрузки)	U <sub>ADC1:ADC0</sub> , В (напряжение шунта)

- Выключение стенда производите в любом порядке

---

## 5. ЦАП на базе ШИМ сигналов микроконтроллера

Цель: Научиться использовать ШИМ сигналы микроконтроллера в качестве ЦАП для получения изменяемых аналоговых сигналов.

- Рекомендуемая схема соединений и ее описание
- Перечень аппаратуры и программных средств
- Указания по проведению эксперимента

### Рекомендуемая схема соединений и ее описание

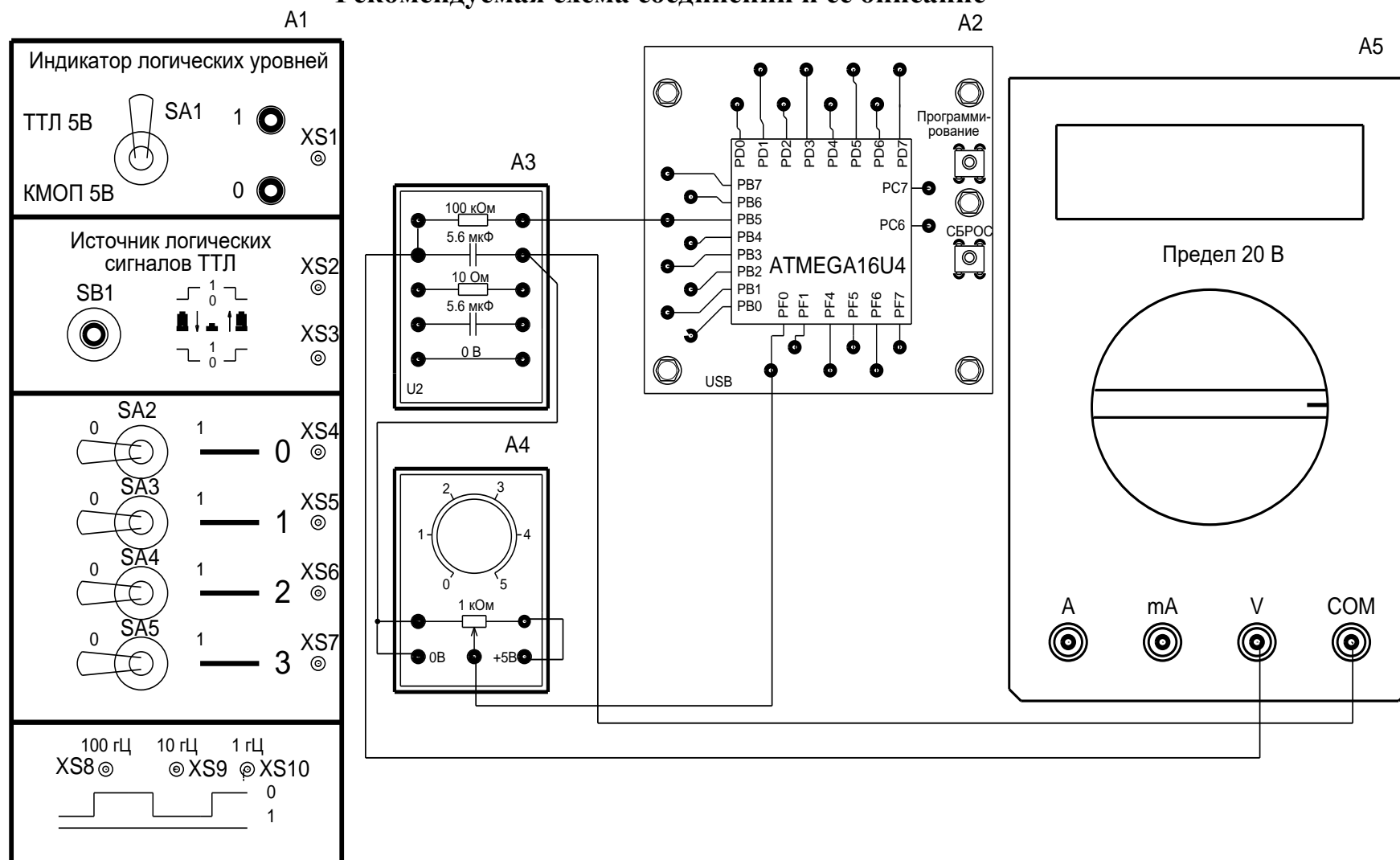


Рис.5.1. Рекомендуемая схема соединений

---

Блок БИЦУ (А1) предназначен для питания остальных блоков. Аналоговый сигнал с движка потенциометра А4 поступает на вход ADC0 (вывод PF0) микроконтроллера А2. Микроконтроллер загружает полученное с АЦП значение в регистр сравнения таймера OCR1A. Таймер запрограммирован на выдачу ШИМ сигнала прямо пропорционального значению регистра OCR1A. Полученный сигнал с вывода OC1A поступает на RC фильтр на элементах миниблока А3. С помощью мультиметра А5 измеряется величина сигнала, поступающего на вход АЦП и выходного сигнала с выхода фильтра.

**Перечень аппаратуры и программных средств***Таблица 5.1.*

Обозначение	Наименование	Тип
A1	Блок БИЦУ	219
A2	Миниблок “Микроконтроллер ATMEGA16U4”	Набор миниблоков 600.25
A3	Миниблок потенциометра	Набор миниблоков 600.25
A4	Миниблок RC элементов	Набор миниблоков 600.25
A5	Мультиметр Mastech UT-53 (или аналогичный)	1403
	Однофазный источник питания ОИП1	218.8
	Компьютер или ноутбук	
	Программа Atmel Studio	Версия 6.1

## Указания по проведению эксперимента

- Подключите компьютер и блок БИЦУ (А1) к однофазному источнику питания ОИП, а сам источник к электрической сети.
- Соберите схему в соответствии с рекомендуемой схемой соединений рис.5.1.
- Подключите блок А1 к компьютеру, используя кабель miniUSB.
- Включите автомат на лицевой панели однофазного источника питания и выключатель СЕТЬ на лицевой панели блока БИЦУ.
- Загрузите в компьютер программу Atmel Studio.
- Загрузите решение Program microcontrollers\Exp\_5\Exp\_5.
- Откройте программу эксперимента Exp\_5.c. Программа эксперимента и ее описание представлена на рис. 5.2.

```

/* Exp_5 "ЦАП на базе ШИМ сигналов микроконтроллера"
 * Exp_5.c АЦП микроконтроллера программируется для ввода данных по каналу ADC0 (вывод PF0)
 * с потенциометра миниблока А4.Полученный сигнал АЦП выдается в регистр сравнения OCR1A
 * таймера 1. При инициализации таймера 1 задан режим работы таймера с разрядностью,
 * совпадающей с разрядностью АЦП и разрешено формирование ШИМ сигнала на выходе.
 * При задании опорного напряжения АЦП равным напряжению питания и правильной работе
 * системы, сигнал на входе АЦП и среднее значение ШИМ сигнала должны совпасть.
 * Для получения среднего значения ШИМ сигнал таймера с вывода PB5 микроконтроллера
 * поступает на RC фильтр из элементов миниблока А3. Напряжение на емкости, является выходом
 * ЦАП. Значения напряжения на входе АЦП и выходе ЦАП могут быть измерены с помощью
 * мультиметра.
 */

#include <avr/io.h>

int main(void)
{
    DDRB|=(1<<PB5);      //выводы управления семисегментным индикатором работают на выход
    ADMUX=0x40;           //при инициализации АЦП выбираем опорное напряжение 5 В
                          //(напряжение питания) и канал ADC0
    TCCR1A=0x83;          //ШИМ сигнал на выводе OCR1A (PB5)? fast PWM 10бит
    TCCR1B=0x9;           //тактовая частота таймера 1 МГц
    ADCSRA=0xD2;          //первый запуск АЦП без разрешения прерываний и с тактовой
                          //частотой 250 кГц

    while(1)
    {
        if (ADCSRA &(1<<ADIF))
        { //закончилось очередное преобразование АЦП
            OCR1A=ADC;
            ADCSRA=0xD2;          //запускаем следующее преобразование АЦП
        }
        for(unsigned int i=0;i<10000;i++) ;    //цикл, имитирующий основную программу
    }
}

```

**Рис. 5.2. Текст программы проекта Exp\_5.**

- Откомпилируйте программу.
- Загрузите программу Exp\_5.hex в микроконтроллер с помощью программатора FLIP.
- Изменяя входное напряжение с помощью потенциометра, заполните таблицу 5.2.

Таблица 5.2

U <sub>ВХ</sub> (вывод PF0), В	U <sub>ВЫХ</sub> ( емкость 5.6 мкФ), В

- Оцените точность преобразования сигнала с помощью ЦАП на базе таймера.
- Измените программу добавив второй канал ЦАП.



---

## **6. Передача данных по последовательным каналам связи**

Цель: Научиться использовать последовательные каналы связи микроконтроллеров, для передачи данных с использованием минимального количества проводов.

## **6.1. Передача данных с использованием канала SPI**

Цель: Научиться использовать последовательный интерфейс SPI микроконтроллера для последовательной передачи данных.

- Рекомендуемая схема соединений и ее описание
- Перечень аппаратуры и программных средств
- Указания по проведению эксперимента

## Рекомендуемая схема соединений и ее описание

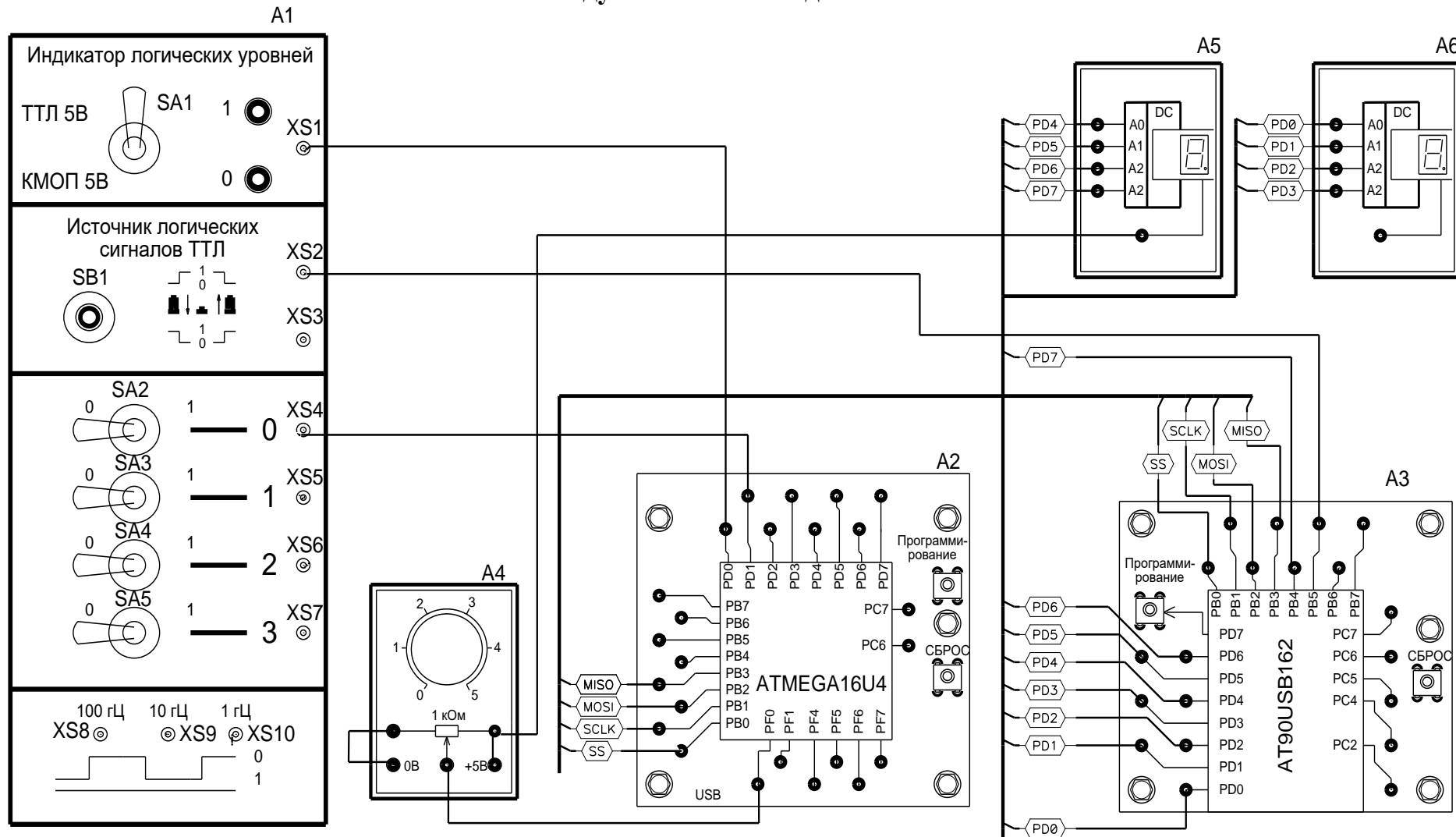


Рис. 6.1.1. Рекомендуемая схема соединений

Микроконтроллер A2 вводит данные с потенциометра миниблока A4 по каналу ADC0 и преобразует полученный результат в Вольты. Далее, микроконтроллер передает полученные данные по каналу SPI микроконтроллеру A3. Микроконтроллер A2 выступает в роли ведущего (master), а микроконтроллер A3 в качестве ведомого (slave). Микроконтроллер A3 принимает данные и выводит полученный результат на семисегментные индикаторы A5, A6. При следующей передаче микроконтроллер A2 передает новые данные, а микроконтроллер A3 возвращает данные, полученные в предыдущем такте, обратно в микроконтроллер A2. По окончании передачи микроконтроллер A2 сравнивает переданные данные, с теми, что он передавал, и если они не совпадают, фиксирует ошибку, зажигая красный светодиод индикатора логических уровней блока БИЦУ A1. Кнопка SB0 позволяет принудительно произвести сброс регистра SPI микроконтроллера A3 в 0, чтобы вызвать ошибку передачи данных.

## Перечень аппаратуры и программных средств

*Таблица 6.1.1*

Обозначение	Наименование	Тип
A1	Блок БИЦУ	219
A2	Миниблок “Микроконтроллер ATMEGA16U4”	Набор миниблоков 600.25
A3	Миниблок “Микроконтроллер AT90USB162”	Набор миниблоков 600.25
A4	Миниблок потенциометра	Набор миниблоков 600.25
A5, A6	Миниблоки семисегментных индикаторов	Набор миниблоков 600.11
	Однофазный источник питания ОИП1	218.8
	Компьютер или ноутбук	
	Программа Atmel Studio	Версия 6.1

## Указания по проведению эксперимента

- Подключите компьютер и блок БИЦУ (А1) к однофазному источнику питания ОИП, а сам источник к электрической сети.
- Соберите схему в соответствии с рекомендуемой схемой соединений рис.6.1.1.
- Подключите миниблок А2 к компьютеру, используя кабель miniUSB.
- Включите автомат на лицевой панели однофазного источника питания и выключатель СЕТЬ на лицевой панели блока БИЦУ.
- Загрузите в компьютер программу Atmel Studio.
- Загрузите решение Program microcontrollers\Exp\_6\_1\Exp\_6\_1.
- Откройте программу эксперимента Exp\_6\_1\_1.c. Сделайте проект Exp\_6\_1\_1 стартовым проектом. Программа эксперимента и ее описание представлена на рис. 6.1.2.

```

/* Exp_6 "Передача данных по последовательным каналам связи"
* Exp_6_1 "Передача данных с использованием канала SPI"
* Exp_6_1_1 "Программирование микроконтроллера в режиме master для передачи данных в режиме
* SPI"
* Exp_6_1_1.c В программе микроконтроллер А2 инициализируется для ввода данных по каналу
* ADC0 и в режим master по передаче данных в режиме SPI. АЦП микроконтроллера постоянно
* вводит значение по каналу ADC0 с движка потенциометра миниблока А4 и преобразует
* полученный результат в * Вольты*10. Полученное значение x1 передается по каналу SPI и
* запоминается. По окончании передачи следующего значения по каналу SPI значение x1
* возвращается в регистр SPI мастера. При правильно работающем канале SPI оно должно
* совпасть с запомненным значением. Если значения не совпадают, зажигается красный светодиод
* индикатора логического уровня.
*/

#include <avr/io.h>
#include "Exp_6_1_1.h"

#define K_V_R_500    20           //коэффициент приведения результата к В*10 при правом
                                   //выравнивании и опорном напряжении 5 В (напряжение
                                   //питания)

int main(void)
{
    DDRB=(1<<PB0)+(1<<PB1)+(1<<PB2);           //выводы SPI, работающие на выход
    DDRD|=(1<<PD0);                             //выход управления светодиодом индикатора
                                                //логических уровней блока БИЦУ А1
    ADMUX=0x40;                                   //при инициализации АЦП выбираем опорное напряжение равное
                                                //напряжению питания (~5В), правое выравнивание результата,
                                                //и канал ADC0
    SPCR=0x53;                                     //бит 7 SPIE "0" - прерывания запрещены
                                                //бит 6 SPIE "1" - работа SPI разрешена
                                                //бит 5 DORD "0" - сдвиг начинается со старшего бита
                                                //бит 4 MSTR\slave "1" - режим ведущего
                                                //бит 3 SPOL "0" - SCK на уровне "0" в ждущем режиме
                                                //бит 2 SPHA "0" - совместно с SPOL определяет, что на
                                                //переднем фронте импульса SCK происходит
                                                //Setup (сдвиг очередного бита), по заднему
                                                //фронту импульса SCK происходит изменение
                                                //состояния шины
                                                //бит 1, 0 - SPR0, SPR1 "11" - частота SCK Fosc\128
    ADCSRA=0xD2;                                   //первый запуск АЦП без разрешения прерываний и с тактовой
                                                //частотой 250 кГц
    int k_trans;                                   //коэффициент преобразования показаний АЦП в В;
    unsigned char n=0;
    unsigned char ADC1, ADC2;                     //заполненные предыдущие результаты АЦП
    unsigned char BackSPI;                       //данные возвращенные обратно по SPI
    k_trans=K_V_R_500;                           //присваиваем значение коэффициенту преобразования в

```

```

int    ResADC;
while(1)
{
    if (ADCSRA &(1<<ADIF))
    { //закончилось очередное преобразование АЦП
        ResADC=PrintBCD99(ADC/k_trans);

        if (n==0) //правильные данные будут возвращены в SPI только со
        { //второго цикла
            n++;
            ADC1=ResADC;
        }
        else if(n==1)
        {
            n++;
            ADC2=ResADC;
            while ((SPSR & (1<<SPIF))==0) ; //если передача данных не
            //окончена, ожидаем ее окончания
            PORTB|=(1<<PB0); //передача окончена,
            //устанавливаем SS в "1"
            BackSPI=SPDR; //после завершения передачи
            //читаем регистр SPDR, чтобы очистить флаг SPIF
        }
        else
        {
            while ((SPSR & (1<<SPIF))==0) ; //если передача данных не
            //окончена, ожидаем ее окончания
            PORTB|=(1<<PB0); //передача окончена,
            //устанавливаем SS в "1"
            BackSPI=SPDR; //после завершения передачи
            //читаем регистр SPDR
            if(ADC1 != BackSPI )
            { //переданные и пришедшие данные не совпадают
                PORTD|=(1<<PD0); //зажигаем светодиод,
                //индицирующий, что в передаче данных произошел сбой
            }
            else
            { //совпадают
                if (PIND & (1<<PD1))
                { //разрешен сброс ранее установленного сигнала ошибки
                    PORTD &=~(1<<PD0);
                }
            }
            ADC1=ADC2;
            ADC2=ResADC;
        }

        PORTB&=~(1<<PB0); //устанавливаем вывод SS в 0
        SPDR=ResADC; //загружаем новые данные в SPI
        ADCSRA=0xD2;
        //запускаем следующее преобразование АЦП
        for(unsigned int i=0; i<100;i++) ; //цикл имитирующий основную
        //часть программы
    }
}

//программа преобразует число к двоично-десятичному числу, упакованному в один байт
//n – преобразуемое число (в диапазоне от 0...99)
//выход: двоично-десятичное число
unsigned char PrintBCD99(unsigned char n)
{

```

```

    unsigned char n1;                //младшая тетрада двоично-десятичного числа
    n1=n%10;
    n=n/10;
    n=(n<<4)+n1;
    return n;
}

```

**Рис. 6.1.2. Текст программы микроконтроллера A2.**

- Откомпилируйте программу.
- Загрузите программу Exr\_6\_1\_1.hex в микроконтроллер с помощью программатора FLIP.
- Переключите кабель miniUSB с микроконтроллера A2 на микроконтроллер A3.
- Откройте программу Exr\_6\_1\_2.c. Сделайте проект Exr\_6\_1\_2 стартовым проектом. Исходный текст программы микроконтроллера A3 представлен на рис. 6.1.3.

```

/* Exr_6 "Передача данных по последовательным каналам связи"
 * Exr_6_1 "Передача данных с использованием канала SPI"
 * Exr_6_1_1 "Программирование микроконтроллера в режиме slave для передачи данных в режиме
 * SPI"
 * Exr_6_1_1.c В программе микроконтроллер A3 инициализируется для ввода данных по каналу
 SPI.
 * Полученные данные выводятся на семисегментные индикаторы.
 * Нажатие кнопки SB1 позволяет загрузить в регистр SPDR нулевую величину, вызывая ошибку при
 * передаче данных по каналу SPI
 * по каналу ADC0 с движка потенциометра миниблока A4 и преобразует полученный результат в
 * Вольты*10. Полученное значение x1 передается по каналу SPI и запоминается. По окончании
 * передачи следующего значения по каналу SPI значение x1 возвращается в регистр SPI мастера.
 * При правильно работающем канале SPI оно должно совпасть с запомненным значением. Если
 * значения не совпадают, зажигается красный светодиод индикатора логического уровня.
 */
#include <avr/io.h>
#define K_V_R_500 655                //коэффициент приведения результата к В*10 при правом
                                     //выравнивании и опорном напряжении 5 В

int main(void)
{
    DDRD=0x7f;                        //выводы управления семисегментным индикатором
    DDRB|=(1<<PB4);
    DDRB|=(1<<PB3);
    SPCR=0x40;                        //выводы SPI, работающие на выход
                                     //бит 7 SPIE "0" - прерывания запрещены
                                     //бит 6 SPIE "1" - работа SPI разрешена
                                     //бит 5 DORD "0" - сдвиг начинается со старшего бита
                                     //бит 4 MSTR\slave "0" - режим ведомого
                                     //бит 3 SPOL "0" - SCK на уровне "0" в ждущем режиме
                                     //бит 2 SPHA "0" - совместно с SPOL определяет, что на
                                     //переднем фронте импульса SCK происходит Setup (сдвиг
                                     //очередного бита), по заднему фронту импульса SCK
                                     //происходит изменение состояния шины
                                     //бит 1,0 - в slave режиме не используются

    while(1)
    {
        while ((SPSR & (1<<SPIF))==0); //ждем приема очередного байта данных по
                                     //каналу SPI
        PORTD=SPDR;                  //зажигаем очередное значение на индикаторе
        if(PORTD & (1<<PD7))
        { //вывод PB7 миниблока не выведен, поэтому копируем его на вывод PD4
            PORTB|=(1<<PB4);
        }
        else
        {
            PORTB&=~(1<<PB4);
        }
    }
}

```



```

        if(PINB & (1<<PB5))
        { //нажата кнопка SB1
            SPDR=0;
            //сбрасываем содержимое SPDR для принудительного вызова ошибки
        }
        for(unsigned int i=0; i<100;i++) ; //цикл имитирую-
щй основную часть программы
    }
}

```

**Рис. 6.1.3. Текст программы микроконтроллера А3.**

- Откомпилируйте программу.
- Загрузите программу в микроконтроллер с помощью программатора FLIP.
- Установите тумблеры SA2 блока БИЦУ А1 установите в положение логического “0”..
- Регулируя потенциометр А4, убедитесь, что микроконтроллер А2 вводит данные по каналу ADC0 и передает их по каналу SPI. Микроконтроллер А3 принимает данные и выводит их на семисегментные индикаторы А5, А6.
- По состоянию светодиодов индикатора логических уровней блока БИЦУ А1 определите, есть ли ошибки при передаче данных.
- Оборвите обратную связь по каналу SPI, нажав на кнопку SB1 (и ли отключив провод между выводами PB3). По состоянию светодиодов индикатора логических уровней блока БИЦУ А1 убедитесь, что микроконтроллер А2 определяет наличие ошибки в связи между каналами.
- Восстановите нормальный режим работы канала SPI.
- Переключите тумблер SA2 в состояние логической ”1”, для сброса ошибки.
- Увеличьте время основного цикла программы микроконтроллера А3 (предположим, увеличив счетчик цикла for до 1000).
- Заново откомпилируйте и загрузите программу.
- Запустите программу микроконтроллера А3, нажав на кнопку СБРОС.
- Запустите программу микроконтроллера А2, нажав на кнопку СБРОС.
- Нажмите кнопку SB1 (обрыв связи). Объясните, почему на индикаторе логических уровней одновременно горят оба светодиода.

## **6.2. Передача данных с использованием канала USART**

Цель: Научиться использовать последовательный интерфейс USART микроконтроллера для последовательной передачи данных.

- Рекомендуемая схема соединений и ее описание
- Перечень аппаратуры и программных средств
- Указания по проведению эксперимента

## Рекомендуемая схема соединений и ее описание

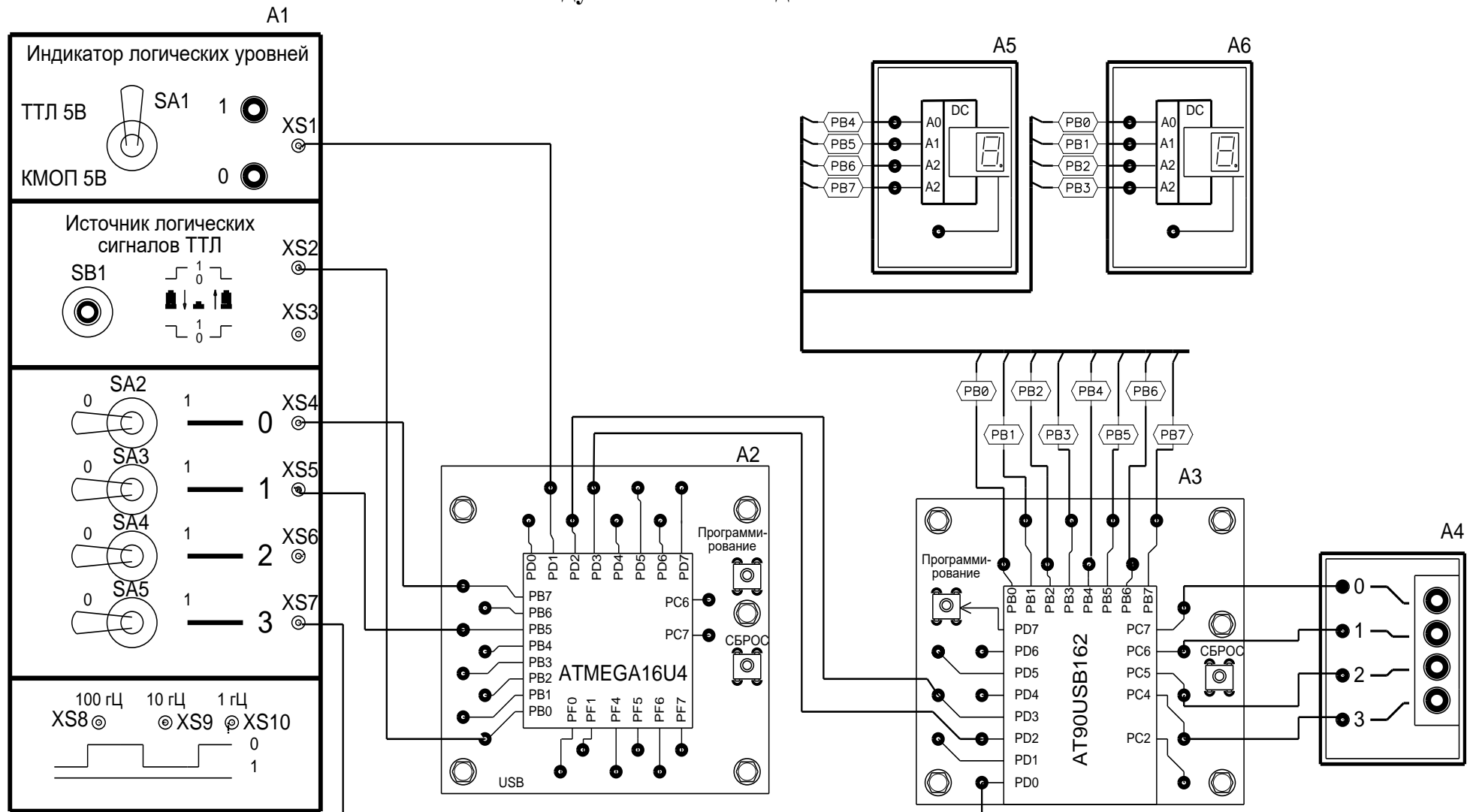


Рис. 6.2.1. Рекомендуемая схема соединений

Микроконтроллер A2 вводит данные со встроенного датчика температуры, преобразует полученную температуру в градусы Цельсия и передает данные по USART с выхода TXD1(вывод PD3) микроконтроллера A2 на вход RXD1 (вывод PD2) микроконтроллера A3. Микроконтроллер A3 принимает полученные данные по каналу USART и выводит значение температуры на семисегментные индикаторы. A5, A6. Миниблок светодиодов показывает состояние флагов приемника USART A3.

Светодиод 0 показывает состояние флага RXC1 USART Receive Complet (Прием данных USART завершен), показывающий, что в USART пришел новый байт данных, который необходимо прочитать.

Светодиод 1 показывает состояние флага DOR1 Data OverRun (Переполнение данных). Флаг устанавливается, если в регистре UDR находится первый непрочитанный байт данных, в регистр сдвига уже пришел второй байт данных и по шине USART пришла команда о начале передачи следующего байта данных. Флаг предупреждает о возможной потере данных. В нормальном режиме работы программы микроконтроллер A3 успевает считывать данные и светодиод SA1 не должен гореть. При включении тумблера SA5 вводится дополнительная задержка, которая приводит к тому, что микроконтроллер перестает успевать считывать информацию.

Светодиод 2 показывает состояние флага FE1 Frame Error (Ошибка структуры). Возникает при несвоевременном приходе очередного стоп бита по шине USART. Ошибка может быть получена при включении тумблера SA2, который меняет число бит передатчика USART с 8 на 9.

Светодиод 3 показывает состояние флага UPEN1 USART parity error ( Ошибка четности USART). Ошибка возникает, если данные пришли с неправильно установленным битом контроля четности. При включении, тумблер SA3 принудительно меняет parity mode (модель четности) с even (четный) на odd (нечетный), вызывая возникновение ошибки USART parity error в приемнике.

При возникновении любой из ошибок UPEN1, FE1, DOR1 микроконтроллер A3 посылает сообщение об ошибке по каналу TXD1 (вывод PD3) микроконтроллера A3 –RXD1 (вывод PD2) микроконтроллера A2. При приеме сообщения микроконтроллер A2 зажигает красный светодиод индикатора логических уровней блока БИЦУ A1. Сброс состояния ошибки может быть произведен кнопкой SB1.

## Перечень аппаратуры и программных средств

*Таблица 6.2.1.*

Обозначение	Наименование	Тип
A1	Блок БИЦУ	219
A2	Миниблок “Микроконтроллер ATMEGA16U4”	Набор миниблоков 600.25
A3	Миниблок “Микроконтроллер AT90USB162”	Набор миниблоков 600.25
A4	Миниблок светодиодов	Набор миниблоков 600.11
A5, A6	Миниблок семисегментных индикаторов	Набор миниблоков 600.11
	Мультиметр Mastech UT-53 (или аналогичный)	
	Однофазный источник питания ОИП1	218.8
	Компьютер или ноутбук	
	Программа Atmel Studio	Версия 6.1

## Указания по проведению эксперимента

- Подключите компьютер и блок БИЦУ (А1) к однофазному источнику питания ОИП, а сам источник к электрической сети.
- Соберите схему в соответствии с рекомендуемой схемой соединений рис.6.2.1.
- Подключите миниблок А2 к компьютеру, используя кабель miniUSB.
- Включите автомат на лицевой панели однофазного источника питания и выключатель СЕТЬ на лицевой панели блока БИЦУ.
- Загрузите в компьютер программу Atmel Studio.
- Загрузите решение Program microcontrollers\Exp\_6\_2\Exp\_6\_2..
- Откройте программу эксперимента Exp\_6\_2\_1.c. Сделайте проект Exp\_6\_2\_1 стартовым проектом. Программа эксперимента и ее описание представлена на рис. 6.2.2.

```

/* Exp_6 "Передача данных по последовательным каналам связи"
* Exp_6_2 "Передача данных с использованием канала USART"
* Exp_6_2_1.c В программе микроконтроллер А2 инициализируется для ввода данных о
* температуре и инициализирует канал USART для передачи данных на скорости
* 19.2 kBit/c в асинхронном режиме.
* Включение тумблера SA2 (вывод PB7) приводит к изменению изменению числа бит с 8
* на 9 вызывая ошибку структуры (Frame Error).
* Включение тумблера SA3 приводит к изменению бита контроля четности, вызывая
* ошибку контроля четности(Parity error)
* При обнаружении ошибки, приемник передает по каналу USART сообщение об ошибке.
* Микроконтроллер А2, приняв это сообщение зажигает красный светодиод индикатора
* логических уровней блока БИЦУ А1
* Кнопка SB1 предназначена для сброса сигнала ошибки
*/

#include <avr/io.h>
#include "Exp_6_2_1.h"

int main(void)
{
    DD RD=(1<<PD1)+(1<<PD3); //вывод управления светодиодом и TXD работают на выход
    //инициализация USART
    UCSR1B= 0x18; //бит 7 RXCIE1 ="0" - прерывания по завершению приема
    // запрещены
    //бит 6 TXCIE1 ="0" - прерывание завершения передачи
    // запрещено
    //бит 5 UDRIE1 ="0" - прерывание, регистр передатчика
    // пустой, запрещено
    //бит 4 RXEN1 ="1" - прием разрешен
    //бит 3 TXEN1 ="1" - передача разрешена
    //бит 2 UCSZ12 ="0" - совместно с UCSZ10, UCSZ11
    // задает режим 8-ми битной передачи
    //бит 1 RXB81 ="0" - значение 9 бита, в 8-ми битном режиме
    // не используется
    //бит 0 TXB81 ="0" - значение 9 бита, в 8-ми битном режиме
    // не используется
    UCSR1C= 0x26; //бит 7, 6 UMSEL1 ="00" - асинхронный режим работы USART
    //бит 5, 4 UPM ="10" - бит четности устанавливается при
    // четном состоянии
    //бит 3 USBS1 ="0" - 1 стоп бит
    //бит 2, 1+бит 3 UCSR1B UCSZ1 ="011" - 8-битный режим
    // передачи
    //бит 0 UCPOL ="0" - в асинхронном режиме работы USART не
    // используется
    UBRR1=2; //частота передачи 19.6 кбит/с при частоте тактового
    // генератора 1 МГц

```

---

```
//Инициализация АЦП для ввода данных по температуре
```

```

ADMUX=0xC7;           //опорное напряжение 2.56 В, правое выравнивание
                        //      результата и канал
ADCSRB=0x20;          //режим измерения температуры
ADCSRA=0xD2;          //первый запуск АЦП без разрешения прерываний и с тактовой
                        //частотой 250 кГц
char  ResADC;          //введенные данные с ADC
while(1)
{
    if (ADCSRA & (1<<ADIF))
    { //закончилось очередное преобразование АЦП
        char tmp;
        ResADC=ADCL; //вводим полученное значение с АЦП
        tmp=ADCH;    //старший байт не используем, но его необходимо прочитать,
                        //чтобы запустить новое преобразование АЦП
        UDR1=PrintBCD99(ResADC-26); //приводим результат к градусам
                                    //Цельсия и передаем полученное значение температуры на
                                    //контроллер A3
        ADCSRA=0xD2; //очередной запуск АЦП
    }
    if (UCSR1A & (1<<RXC1))
    { //пришли новые данные с микроконтроллера A3
        if(UDR1==1)
        {
            PORTD|=(1<<PD1); //зажигаем красный светодиод
                               //индикатора логических уровней, индицирующих ошибку
        }
    }
    if(PINB & (1<<PB0))
    { //сброс индикации ошибки
        PORTD&=~(1<<PD1);
    }
    if(PINB & (1<<PB7))
    { //тумблер SA2 включен (изменяем передачу на 9 битную), имитируя ошибку
      //структуры (Frame Error)
        UCSR1B|= (1<<UCSZ12);
    }
    else
    { //тумблер SA2 выключен (изменяем передачу на 8 битную)
        UCSR1B &= ~(1<<UCSZ12);
    }
    if(PINB & (1<<PB5))
    { //тумблер SA3 включен (изменяем бит четности с четного на нечетный),
      //вызывая ошибку контроля четности (Parity error)
        UCSR1C|=(1<<UPM10);
    }
    else
    { //устанавливаем бит четности в четный
        UCSR1C &= ~(1<<UPM10);
    }
    for(unsigned int i=0; i<500;i++) ; //цикл имитирующий основную часть
                                        //программы
}

//программа преобразует число к двоично-десятичному числу, упакованному в один байт
//n – преобразуемое число (в диапазоне от 0...99)
//выход: двоично-десятичное число
unsigned char PrintBCD99(unsigned char n)
{
    unsigned char n1; //младшая тетрада двоично-десятичного числа
    n1=n%10;

```

```

n=n/10;
n=(n<<4)+n1;
return n;
}

```

**Рис. 6.2.2. Текст программы микроконтроллера A2.**

- Откомпилируйте программу.
- Загрузите программу Eхp\_6\_2\_2.hex в микроконтроллер с помощью программатора FLIP.
- Переключите кабель miniUSB с микроконтроллера A2 на микроконтроллер A3.
- Откройте программу Eхp\_6\_2\_2.c. Сделайте проект Eхp\_6\_2\_2 стартовым проектом. Исходный текст программы микроконтроллера A3 представлен на рис. 6.2.3.

```

/* Eхp_6 "Передача данных по последовательным каналам связи"
* Eхp_6_2 "Передача данных с использованием канала USART"
* Eхp_6_2_2.c В программе микроконтроллер A3 принимает по каналу USART данные о
* температуре и выводит полученный результат на семисегментный индикатор.
* Включение тумблера SA5 приводит к увеличению времени выполнения основной
* программы. Микроконтроллер перестает вовремя считывать, поступающие данные.
* При возникновении ошибки в приеме данных, микроконтроллер выводит данные об
* ошибке на миниблок светодиодов и отправляет микроконтроллеру A2 сообщение об ошибке.
*/

#include <avr/io.h>

int main(void)
{
    DDRB=0xff;                //выходы управления семисегментным индикатором
    DDRC=(1<<PC4)+(1<<PC5)+(1<<PC6)+(1<<PC7);    //выходы управления светодиодами
    DDRD=(1<<PD3);            //выход TXD

    //инициализация USART
    UCSR1B= 0x18;             //бит 7 RXCIE1 ="0" - прерывания по завершению приема
                                // запрещены
                                //бит 6 TXCIE1 ="0" - прерывание завершения передачи
                                // запрещено
                                //бит 5 UDRIE1 ="0" - прерывание. регистр передатчика
                                // пустой, запрещено
                                //бит 4 RXEN1 ="1" - прием разрешен
                                //бит 3 TXEN1 ="1" - передача разрешена
                                //бит 2 UCSZ12 ="0" - совместно с UCSZ10, UCSZ11
                                // задает режим 8-ми битной передачи
                                //бит 1 RXB81 ="0" - значение 9 бита, в 8-ми битном режиме
                                // не используется
                                //бит 0 TXB81 ="0" - значение 9 бита, в 8-ми битном
                                // режиме не используется
    UCSR1C=0x26;              //бит 7, 6 UMSEL1 ="00" - асинхронный режим работы USART
                                //бит 5, 4 UPM ="10" - бит четности устанавливается при
                                // четном состоянии
                                //бит 3 USBS1 ="0" - 1 стоп бит
                                //бит 2, 1+бит 3 UCSR1B UCSZ1 ="011" - 8-битный режим
                                // передачи
                                //бит 0 UCPOL ="0" - в асинхронном режиме работы USART не
                                // используется
    UBRR1=2;                  //частота передачи 19.6 кбит/с при частоте тактового
                                //генератора 1 МГц

    unsigned char mistake=0;  //индикатор ошибки в приеме данных
                                //"0" - ошибка отсутствует
                                //"1" - при передаче данных возникла ошибка
    unsigned int t=200;        //переменная, отвечающая за время выполнения основного

```



```

//цикла программы

while(1)
{
    mistake=0;
    if(UCSR1A &(1<< DOR1))
    { //ошибка переполнения данных (Data Over Run)
        PORTC|=(1<<PC6); //зажигаем светодиод 1
        mistake=1; //запоминаем наличие ошибки
    }
    else
    { //ошибка переполнения данных отсутствует
        PORTC&=~(1<<PC6); //тушим светодиод 1
    }
    if(UCSR1A & (1<<FE1))
    { //ошибка структуры передачи (Frame Error)
        PORTC|=(1<<PC5); //зажигаем светодиод 2
        mistake=1; //запоминаем наличие ошибки
    }
    else
    { //ошибка структуры передачи (Frame Error) отсутствует
        PORTC &= ~(1<<PC5); //тушим светодиод 2
    }
    if(UCSR1A & (1<<UPE1))
    { //ошибка контроля четности (Parity Error)
        PORTC|=(1<<PC4); //зажигаем светодиод 3
        mistake=1;
    }
    else
    { //ошибка контроля четности (Parity Error) отсутствует
        PORTC&=~(1<<PC4); //тушим светодиод 3
    }
    if(UCSR1A & (1<<RXC1))
    { //пришли новые данные с микроконтроллера A2
        PORTB=UDR1; //выводим новые данные на семисегментный индикатор
        PORTC|=(1<<PC7); //зажигаем светодиод 0
    }
    else
    { //новые данные не приходят
        PORTC&=~(1<<PC7); //тушими светодиод 0
    }
    if(mistake==1)
    { //при передаче данных имеется ошибка
        UDR1=mistake; //передаем сообщение об ошибке процессору A2
    }
    if(PIND & (1<<PD0))
    { //включен тумблер SA5, увеличиваем время основного цикла
        t=2000;
    }
    else
    { //возвращаемся к стандартному времени
        t=200;
    }
    for(unsigned int i=0; i<t;i++) ; //цикл, имитирующий основную часть программы
}
}

```

**Рис. 6.2.3. Текст программы микроконтроллера A3.**

- Откомпилируйте программу.
- Загрузите программу в микроконтроллер с помощью программатора FLIP.
- Установите тумблеры SA2, SA3, SA5 в нулевое положение.

- Запустите программу микроконтроллера А3, нажав на кнопку СБРОС.
- Запустите программу микроконтроллера А2, нажав на кнопку СБРОС.
- Если горит красный светодиод индикатора логических уровней блока А1, нажмите кнопку SB1. Убедитесь, что передача происходит в штатном режиме, горит зеленый светодиод индикатора логических уровней и светодиод 0 миниблока светодиодов А4.
- По показаниям семисегментных индикаторов А5, А6 убедитесь, что температура, измеренная микроконтроллером А2, передается по каналу USART на микроконтроллер А3 и ее значение отображается на индикаторах А5, А6.
- Прижмите палец к середине изображения микроконтроллера А2. За счет нагрева микросхемы, расположенной на другой стороне платы происходит изменение температуры кристалла. Убедитесь в изменении температуры по показаниям индикаторов А5, А6.
- По состоянию светодиодов миниблока А4 проконтролируйте, какие флаги устанавливаются в этом режиме передачи. Заполните таблицу 6.2.2 и сделайте вывод о наличии ошибок при передаче данных.

Таблица 6.2.2.

Флаги	Режим работы			
	SA2=0; SA3=0; SA5=0.	SA2=1; SA3=0;SA3=0;	SA3=1; SA2=0;SA5=0;	SA5=1; SA2=0;SA3=0
RXE1				
DOR1				
FE1				
UPEN1				
Наличие ошибки микроконтр. А3				
Светодиод ИЛУ БИЦУ1				

- По состоянию светодиода индикатора логических уровней (ИЛУ) БИЦУ А1, проверьте, передает ли микроконтроллер А3 информацию о своей ошибке на микроконтроллер А2.
- Нажмите кнопку SB1 для сброса состояния ошибки индикатора логических уровней.
- Переключите тумблер SA5 в состояние логической "1" (увеличение времени выполнения основной программы микроконтроллера А3, не связанное с USART). Проконтролируйте состояние флагов и заполните соответствующий столбец таблицы 6.2.2. Сделайте вывод о наличии ошибки микроконтроллера А3 и о том были ли переданы сведения об ошибке на микроконтроллер А2. Предложите варианты исправления этой ошибки.
- Отключите тумблер SA5.
- Нажмите кнопку SB1 для сброса состояния ошибки индикатора логических уровней.
- Включите тумблер SA2 (приводит к несовпадению разрядности данных передатчика и приемника). Проконтролируйте состояние флагов и заполните соответствующий столбец таблицы 6.2.2. Измените программу приемника, чтобы он пытался подобрать требуемую разрядность.
- Отключите тумблер SA2.
- Нажмите кнопку SB1 для сброса состояния ошибки индикатора логических уровней.
- Включите тумблер SA3 (приводит к изменению бита четности передатчика). Проконтролируйте состояние флагов и заполните соответствующий столбец таблицы 6.2.2.

- 
- Сделайте вывод о том, в каких случаях обратная связь об ошибке приемника работает, а в каких нет.

## **7. Использование микроконтроллера в прикладных задачах**

Цель: Показать возможности использования микроконтроллера в типовых задачах.

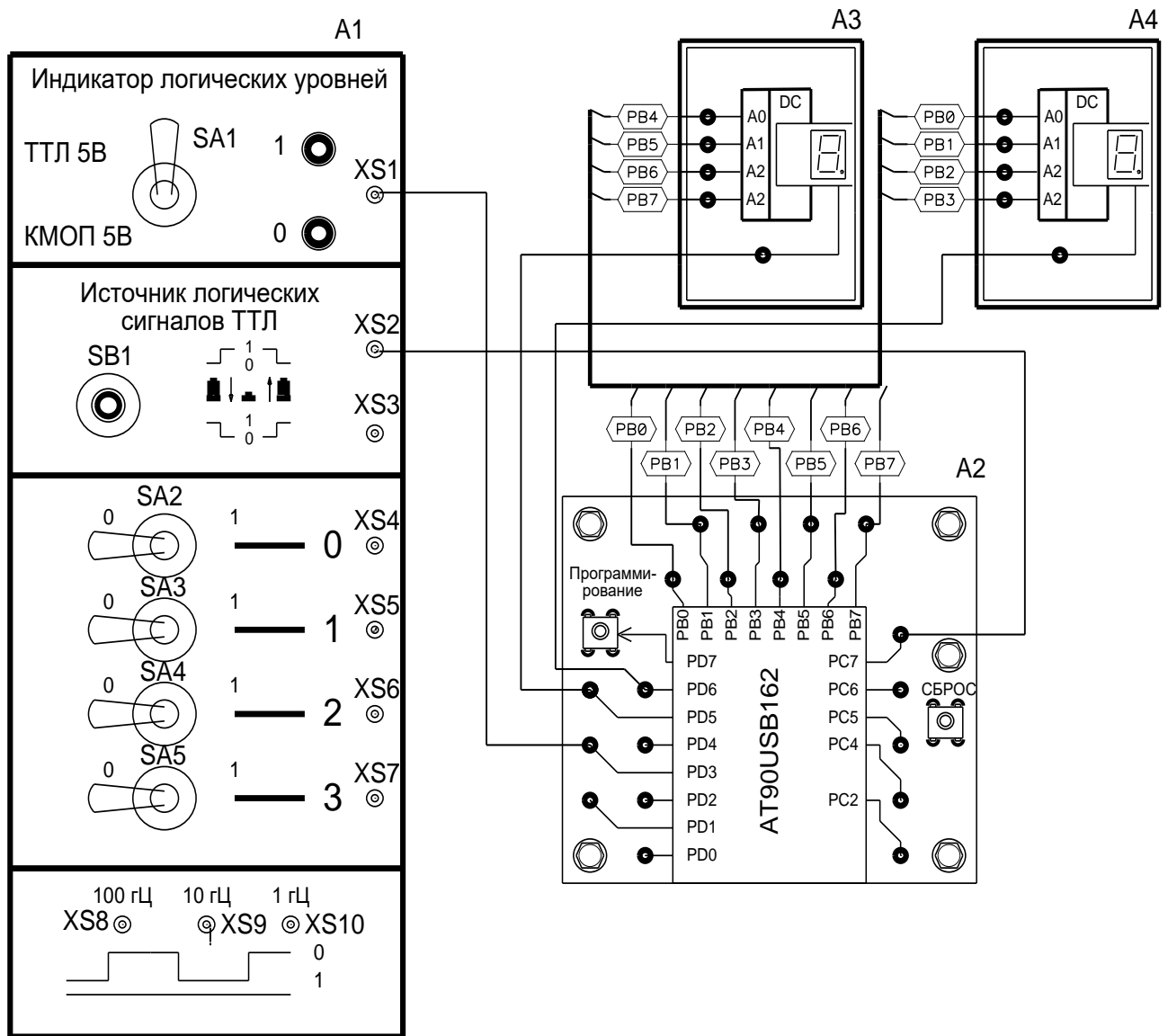
---

## 7.1. Секундомер

Цель: Научиться использовать микроконтроллер для измерения интервалов времени.

- Рекомендуемая схема соединений и ее описание
- Перечень аппаратуры и программных средств
- Указания по проведению эксперимента

### Рекомендуемая схема соединений и ее описание



**Рис. 7.1.1. Рекомендуемая схема соединений**

При поступлении переднего фронта сигнала при нажатии кнопки SB1 таймер 1 осуществляет захват и сбрасывает свое значение в 0. Одновременно таймер переключается на ввод заднего фронта сигнала. При отпускании кнопки SB1 происходит следующий захват и вводится полученное значение, которое пересчитывается в секунды. Если значение превышает величину 99 с (максимально допустимое значение для данного семисегментного индикатора), зажигается красный светодиод, индицирующий перегрузку.

---

**Перечень аппаратуры и программных средств***Таблица 7.1.*

Обозначение	Наименование	Тип
A1	Блок БИЦУ	219
A2	Миниблок “Микроконтроллер AT90USB162”	Набор миниблоков 600.25
A3, A4	Миниблок семисегментных ин- дикаторов	Набор миниблоков 600.11
	Однофазный источник питания ОИП1	218.8
	Компьютер или ноутбук	
	Программа Atmel Studio	Версия 6.1

## Указания по проведению эксперимента

- Подключите компьютер и блок БИЦУ (А1) к однофазному источнику питания ОИП, а сам источник к электрической сети.
- Соберите схему в соответствии с рекомендуемой схемой соединений рис.7.1.1.
- Подключите миниблок А2 к компьютеру, используя кабель miniUSB.
- Включите автомат на лицевой панели однофазного источника питания и выключатель СЕТЬ на лицевой панели блока БИЦУ.
- Загрузите в компьютер программу Atmel Studio.
- Загрузите решение Program microcontrollers\Exp\_7\_1\Exp\_7\_1..
- Откройте программу эксперимента Exp\_7\_1.c. Программа эксперимента и ее описание представлена на рис. 7.1.2.
- Откомпилируйте программу.
- Загрузите программу Exp\_7\_2.hex в микроконтроллер с помощью программатора FLIP.
- Запустите программу микроконтроллера А2, нажав на кнопку СБРОС.
- Нажмите и удерживайте в нажатом состоянии кнопку SB1 блока А1. По показаниям семисегментных индикаторов А3, А4 убедитесь, что секундомер начал отсчет времени.
- Отпустите кнопку SB1. Убедитесь, что отсчет времени закончился, и на индикаторе зафиксировалась длительность импульса.
- Доработайте программу, чтобы она могла измерять время импульсов, начиная с единиц мс. Измерьте время импульса встроенного в блок А1 генератора 100 Гц.

```

/* Exp_7_1 "Секундомер"
 * Exp_7_1.c Микроконтроллер измеряет промежуток времени между передним
 * фронтом и задним фронтом на выводе PC7. Пересчитывает полученную
 * длительность в с и выводит полученный результат на семисегментный индикатор
 */
#include <avr/io.h>
#include <avr/interrupt.h>
#include "Exp_7_1.h"
unsigned char ovf; //счетчик числа переполнений
long Tf; //переменная для хранения времени захвата переднего фронта
unsigned char Flagt; //флаг готовности нового времени импульса
int main(void)
{
    DDRB=0xFF; //выходы управления семисегментными индикаторами
    DDRD|=(1<<PD3)+(1<<PD5)+(1<<PD6); //выходы управления светодиодом и точками
    //семисегментных индикаторов

    //инициализация таймеров
    TCCR1A=0x0; //инициализация таймера 0 с коэффициентом прескалера 1024
    TCCR1B=0xC5; //с захватом сигнала по переднему фронту, подавитель помех
    //от дребезга сигналов включен.
    TIMSK1=(1<<ICIE1)+(1<<TOIE1); //разрешение прерываний таймера по захвату и по
    //переполнению
    TCCR0B=0x1; //коэффициент прескалера таймера 0 равен 1
    TIMSK0=(1<<TOIE0); //разрешение прерывания таймера 0
    float ks=0.00102; //коэффициент преобразования тиков в секунды
    float ks_10=0.0102; //коэффициент преобразования в десятые доли секунды
    sei(); //разрешение прерываний
    while(1)
    {
        if(Flagt==1)
        { //вывод полученного времени импульса на индикатор

            if(Tf>96678)
            { //время больше 99с, переполнение

```



```

        PORTD|=(1<<PD3);           //зажигаем светодиод переполнения
        PORTD&=~(1<<PD5);          //тушим разделительную точку
        Tf-=96678;
        PORTB=PrintBCD99(Tf*ks);
    }
    else if(Tf>9667)
    { //время больше 9,9 с
        PORTD&=~(1<<PD3);          //тушим светодиод переполнения
        PORTD&=~(1<<PD5);          //тушим разделительную точку
        PORTB=PrintBCD99(Tf*ks);
    }
    else
    { //время меньше 9,9 с
        PORTD&=~(1<<PD3);          //тушим светодиод переполнения
        PORTD|=(1<<PD5);           //зажигаем разделительную точку
        PORTB=PrintBCD99(Tf*ks_10);
    }
    Flagt=0;
}
if(Flagt==2)
{ //вывод текущего времени на индикатор
    long t;
    t=TCNT1;
    if(ovf==1)
        t+=0xffff;
    if(t>96678)
    { //время больше 99с, переполнение
        PORTD|=(1<<PD3);           //зажигаем светодиод переполнения
        PORTD&=~(1<<PD5);          //тушим разделительную точку
        Tf-=96678;
        PORTB=PrintBCD99(t*ks);
    }
    else if(t>9667)
    { //время больше 9,9 с
        PORTD&=~(1<<PD3);          //тушим светодиод переполнения
        PORTD&=~(1<<PD5);          //тушим разделительную точку
        PORTB=PrintBCD99(t*ks);
    }
    else
    {
        PORTD&=~(1<<PD3);          //тушим светодиод переполнения
        PORTD|=(1<<PD5);           //зажигаем разделительную точку
        PORTB=PrintBCD99(t*ks_10);
    }
    Flagt=0;
}
//здесь основная программа
}
}

//прерывание таймера1 по захвату
ISR(TIMER1_CAPT_vect)
{
    if(TCCR1B & (1<<ICES1))
    { //захват по переднему фронту, начинаем отсчет импульса
        TCNT1=0;                   //начинаем отсчет новой длительности импульса
        ovf=0;                     //обнуляем счетчик переполнений
        TCCR1B=0x85;               //переключаем таймер на захват по заднему фронту
        PORTB=0;                   //обнуляем индикатор
        TIMSK0=1;                  //разрешение прерывания таймера 0 по переполнению
    }
    else
    { //задний фронт сигнала, можно определить длительность импульса
        Tf=ICR1;                   //время в тиках таймера
    }
}

```

```

        if(ovf==1)
            Tf=Tf+0xffff; //сохраняем текущее значение времени в тиках
        TCCR1B=0xC5;      //переключаем на захват по переднему фронту
        Flag1=1;          //готовы новые данные времени импульса
        TIMSK0=0;         //запрет прерывания таймера 0
    }
}

//прерывание таймера 1 по переполнению
ISR(TIM1_OVF_vect)
{
    if(!(TCCR1B & (1<<ICES1)))
    { //режим отсчета времени импульса
        ovf++;
        if(ovf>1)
        { //переполнение
            Tf=0xffff;      //запоминаем текущее значение времени в тиках
            Flag1=1;
            TCCR1B=0xC5;    //переключаем на захват по переднему фронту
        }
    }
}

//прерывание таймера 0 по переполнению
ISR(TIM0_OVF_vect)
{
    if(!(TCCR1B & (1<<ICES1)))
    { //режим отсчета времени импульса
        Flag2=2;           //разрешение обновления данных на семисегментном
                           //индикаторе
    }
}

//программа преобразует число к двоично-десятичному числу, упакованному в один байт
//n – преобразуемое число (в диапазоне от 0...99)
//выход: двоично-десятичное число
unsigned char PrintBCD99(unsigned char n)
{
    unsigned char n1;      //младшая тетрада двоично-десятичного числа
    n1=n%10;
    n=n/10;
    n=(n<<4)+n1;
    return n;
}

```

**Рис. 7.1.2. Текст программы проекта "Секундомер" Exr\_7\_1.**

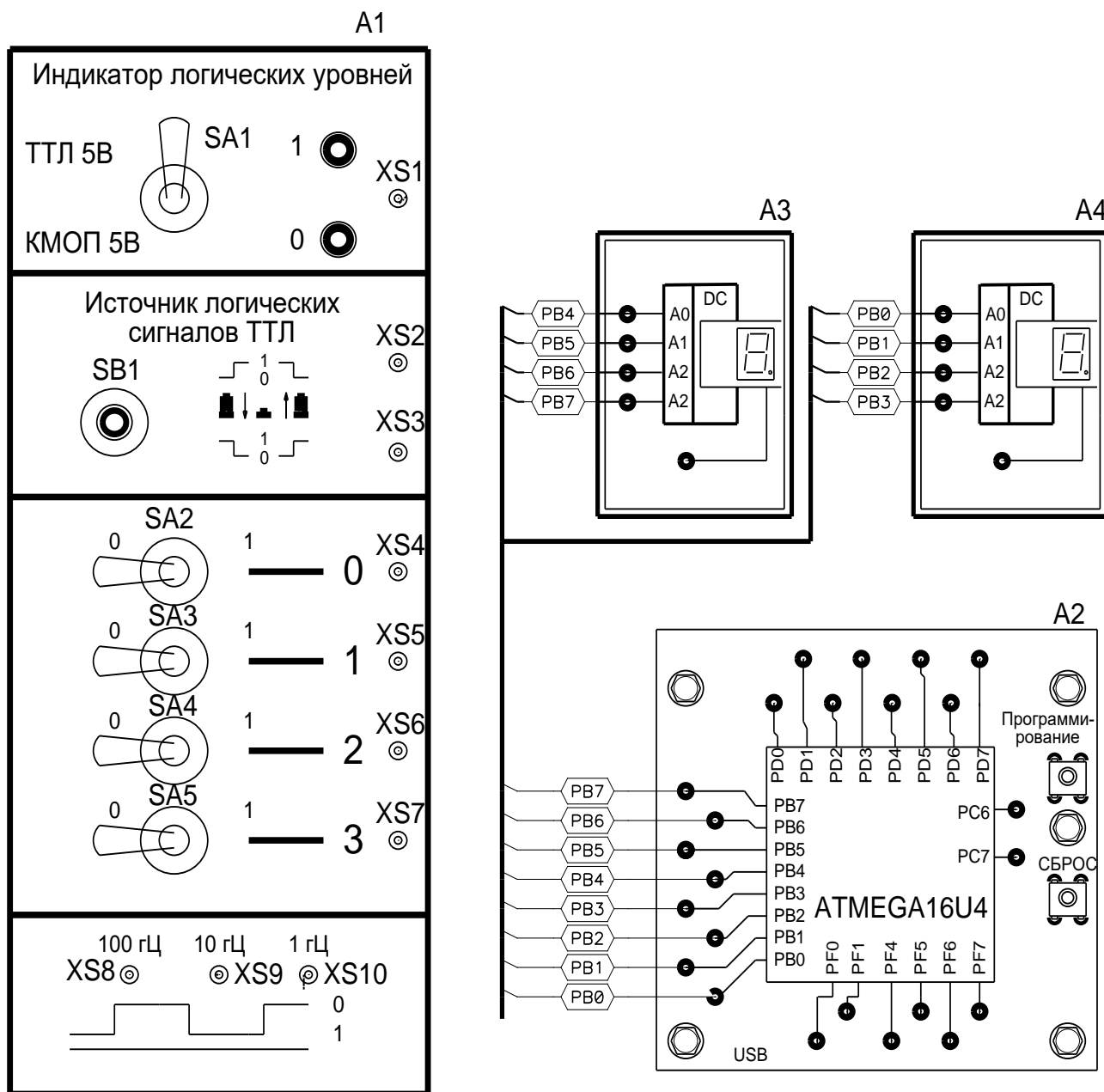
---

## 7.2. Термометр

Цель: Научиться использовать микроконтроллер для ввода данных температуры

- Рекомендуемая схема соединений и ее описание
- Перечень аппаратуры и программных средств
- Указания по проведению эксперимента

### Рекомендуемая схема соединений и ее описание



**Рис. 7.2.1. Рекомендуемая схема электрических соединений**

Блок БИЦУ (A1) служит для питания остальных блоков микроконтроллера.

Микроконтроллер A2 с помощью АЦП измеряет напряжение на встроенном в него датчике температуры. Т.к. потребление микроконтроллера низкое, температура датчика очень близка к температуре окружающей среды. Микроконтроллер пересчитывает показания АЦП в данные температуры в градусах цельсия и выводит полученный результат на семи-сегментные индикаторы A3, A4.

---

**Перечень аппаратуры и программных средств***Таблица 7.2.*

Обозначение	Наименование	Тип
A1	Блок БИЦУ	219
A2	Миниблок “Микроконтроллер ATMEGA16U4”	Набор миниблоков 600.25
A3, A4	Миниблок семисегментных индикаторов	Набор миниблоков 600.11
	Мультиметр Mastech UT-53 (или аналогичный)	
	Однофазный источник питания ОИП1	218.8
	Компьютер или ноутбук	
	Программа Atmel Studio	Версия 6.1

---

### Указания по проведению эксперимента

- Подключите компьютер и блок БИЦУ (А1) к однофазному источнику питания ОИП, а сам источник к электрической сети.
- Соберите схему в соответствии с рекомендуемой схемой соединений рис.7.2.1.
- Подключите миниблок А2 к компьютеру, используя кабель miniUSB.
- Включите автомат на лицевой панели однофазного источника питания и выключатель СЕТЬ на лицевой панели блока БИЦУ.
- Загрузите в компьютер программу Atmel Studio.
- Загрузите решение Program microcontrollers\Exp\_7\_2\Exp\_7\_2..
- Откройте программу эксперимента Exp\_7\_2.c. Программа эксперимента и ее описание представлена на рис. 7.2.2.
- Откомпилируйте программу.
- Загрузите программу Exp\_7\_1.hex в микроконтроллер с помощью программатора FLIP.
- Запустите программу микроконтроллера А2, нажав на кнопку СБРОС.
- По показаниям семисегментных индикаторов А3, А4 убедитесь, что термометр показывает температуру окружающей среды.
- Прижмите палец к середине изображения микроконтроллера А2. За счет нагрева микросхемы, расположенной на другой стороне платы, происходит изменение температуры датчика температуры. Убедитесь в изменении температуры по показаниям индикаторов А3, А4.
- Включите мультиметр в режиме измерения температуры и сравните показания термометра с показаниями мультиметра. При необходимости откорректируйте программу преобразования значений в градусы Цельсия.
- Доработайте программу для уменьшения “дребезга” показаний семисегментных индикаторов.

```

/* Exp_7 "Использование микроконтроллера в прикладных задачах"
* Exp_7_2.c "Термометр"
* В программе микроконтроллер A2 инициализируется для ввода данных
* о температуре со встроенного в микроконтроллер датчика температуры.
* Преобразует полученное значение в градусы Цельсия и выводит
* результат на семисегментные индикаторы A3,A4.
*/
#include <avr/io.h>
#include "Exp_7_2.h"
int main(void)
{
    DDRB=0xff;          //выходы управления семисегментными индикаторами
//Инициализация АЦП для ввода данных по температуре
    ADMUX=0xC7;          //опорное напряжение 2.56 В, правое выравнивание результата и
    ADCSRB=0x20;          //канал измерения температуры
    ADCSRA=0xD2;          //первый запуск АЦП без разрешения прерываний и с тактовой
                        //частотой 250 кГц
    char ResADC;          //данные с АЦП
    while(1)
    {
        if (ADCSRA & (1<<ADIF))
        {//закончилось очередное преобразование АЦП
            char tmp;
            ResADC=ADCL; //вводим полученное значение с АЦП
            tmp=ADCH;    //старший байт не используем, но его необходимо
                        //прочитать, чтобы запустить новое преобразование АЦП
            PORTB=PrintBCD99(ResADC-26); //преобразуем результат к градусам
                        //Цельсия и выводим полученное значение на семисегментные
                        //индикаторы A3, A4
            ADCSRA=0xD2; //очередной запуск АЦП
        }
        for(unsigned int i=0; i<500;i++) ;//цикл имитирующий основную часть программы
    }
}

//программа преобразует число к двоично-десятичному числу, упакованному в один байт
//n – преобразуемое число (в диапазоне от 0...99
//выход: двоично-десятичное число
unsigned char PrintBCD99(unsigned char n)
{
    unsigned char n1;          //младшая тетрада двоично-десятичного числа
    n1=n%10;
    n=n/10;
    n=(n<<4)+n1;
    return n;
}

```

**Рис. 7.2.2. Текст программы проекта "Термометр" Exp\_7\_2.**