

Министерство образования и науки Российской Федерации

Курский государственный университет

Кафедра *программного обеспечения и администрирования  
информационных систем*

Проектирование систем на микроконтроллерах

Лабораторный практикум

Курск 2019

**Проектирование систем на микроконтроллерах:** методические указания к выполнению цикла лабораторных работ по курсу «Системы реального времени» / сост. А. П. Жмакин; Курск. гос. ун-т. – Курск, 2019, 31 с.

Лабораторный цикл предназначен для изучения архитектуры однокристальных микроЭВМ (микроконтроллеров) и реализации на их основе простых систем реального времени. Работы выполняются в среде САПР **Proteus-8x** и позволяют пользователям познакомиться с принципами построения электронных схем на базе микроконтроллеров и низкоуровневым программированием микроконтроллеров.

Пособие предназначено для студентов направления направления 02.03.03 *Математическое обеспечение и администрирование информационных систем*.

Курский государственный университет, 2019

# Содержание

<b>1. Лабораторная работа №1</b>	
<b>Знакомство со средой проектирования Proteus и архитектурой заданной микроЭВМ.</b>	<b>5</b>
1.1. Содержание работы . . . . .	5
1.2. Литература . . . . .	5
1.3. Контрольные вопросы . . . . .	6
<b>2. Лабораторная работа №2</b>	
<b>Простые решения</b>	<b>7</b>
2.1. Светодиоды и кнопки . . . . .	7
2.2. Звук . . . . .	7
2.3. Семисегментные индикаторы (статика) . . . . .	8
2.4. Содержание отчета . . . . .	8
2.5. Контрольные вопросы . . . . .	8
<b>3. Лабораторная работа №3</b>	
<b>Динамическая семисегментная индикация и матричная клавиатура</b>	<b>10</b>
3.1. Порядок выполнения работы . . . . .	10
3.2. Содержание отчета . . . . .	10
3.3. Контрольные вопросы . . . . .	11
<b>4. Лабораторная работа №4</b>	
<b>Таймерные системы и прерывания</b>	<b>12</b>
4.1. Порядок выполнения работы . . . . .	12
4.2. Содержание отчета . . . . .	12
4.3. Контрольные вопросы . . . . .	13
<b>5. Лабораторная работа №5</b>	
<b>Широтно-импульсная модуляция и АЦП</b>	<b>14</b>
5.1. Порядок выполнения работы . . . . .	15
5.2. Содержание отчета . . . . .	15
5.3. Контрольные вопросы . . . . .	17
<b>6. Приложения</b>	<b>18</b>
6.1. Открыть новый проект в Proteus . . . . .	18
6.2. 16-разрядные таймеры-счётчики с режимами ШИМ в микроЭВМ семейства ATmega . . . . .	21
6.2.1. Программируемый реверсивный 16-разрядный счетчик . . . . .	22
6.2.2. Доступ к 16-разрядным регистрам . . . . .	22
6.2.3. Блок сравнения . . . . .	23
6.2.4. Блок захвата . . . . .	25
6.2.5. Режимы работы . . . . .	25

6.2.6. Регистры управления таймером-счётчиком . . . . .	26
6.3. АЦП микроЭВМ семейства ATmega . . . . .	28

# 1. Лабораторная работа №1

## Знакомство со средой проектирования Proteus и архитектурой заданной микроЭВМ.

### 1.1. Содержание работы

Proteus – это пакет программ класса САПР, объединяющий в себе две основных программы: ISIS – средство разработки и отладки в режиме реального времени электронных схем и ARES – средство разработки печатных плат. В настоящих лабораторных работах мы будем использовать только ISIS.

В рамках лабораторной работы №1 необходимо:

1. Изучит основные элементы архитектуры заданной микроЭВМ:
  - 1.1. Состав основных элементов микроЭВМ, их характеристики и связи.
  - 1.2. Организацию памяти и способы адресации.
  - 1.3. Систему команд.
  - 1.4. Работу портов ввода-вывода.
2. Изучить работу в среде Proteus:
  - 2.1. Способ создания нового проекта в Proteus.
  - 2.2. Интерфейс программы ISIS.
  - 2.3. Библиотеку компонентов.
  - 2.4. Приёмы создания и редактирования электронной схемы.
  - 2.5. Основные средства отладки программы.

### 1.2. Литература

#### Общие вопросы

1. Proteus по-русски Радио-ежегодник 2013, выпуск 24
2. Рюмик С. М. 1000 и одна микроконтроллерная схема. – М.: Додэка-XXI, Вып. 1 – 2010. - 356 с: ил., Вып. 2 – 2011. - 400 с: ил.
3. Бойко В. И. и др. Схемотехника электронных систем. Микропроцессоры и микроконтроллеры. – СПб.: БХВ-Петербург, 2004. – 464 с.: ил.

#### Микроконтроллеры семейства MCS-51

1. Каспер Э. Программирование на языке Ассемблера для микроконтроллеров семейства i8051. - М.: Горячая линия - Телеком, 2004. - 191 с.: ил.
2. Магда Ю. С. Микроконтроллеры серии 8051: практический подход. – М.: ДМК Пресс. 2008.– 228 с.

3. Жмакин А. П., Титов В. С. Однокристальные микроЭВМ в системах управления : Учебное пособие / Курск. гос. тех. ун-т., Курск, 2002. - 244 с.: ил.
4. Веприк В.Н. и др. Микроконтроллеры семейства MCS-51: Учебное пособие. - Новосибирск.

#### Микроконтроллеры семейства PIC-16

1. Предко М. PIC-микроконтроллеры: архитектура и программирование. Пер. с англ. - М.:ДМК Пресс,2010. – 512 с.: ил.
2. PIC16\_Manual.pdf
3. pic16f87x\_руск.pdf
4. Жмакин А. П., Титов В. С. Однокристальные микроЭВМ в системах управления : Учебное пособие / Курск. гос. тех. ун-т., Курск, 2002. - 244 с.: ил.

#### Микроконтроллеры семейства PIC-18

1. Предко М. PIC-микроконтроллеры: архитектура и программирование. Пер. с англ. - М.:ДМК Пресс,2010. – 512 с.: ил.
2. PIC18FXX2\_manual.pdf

#### Микроконтроллеры семейства ATmega

1. Ревич Ю. В. Практическое программирование микроконтроллеров Atmel AVR на языке ассемблера. - 2-е изд., испр. - СПб.: БХВ-Петербург, 2011. - 352 с: ил.
2. Евстифеев А. В. Микроконтроллеры AVR семейства Mega. Руководство пользователя. – М.: Додэка-XXI, 2007. – 592 с.: ил.
3. Белов А. В. Разработка устройств на микроконтроллерах AVR: шагаем от «чайников» до профи. – СПб.: Наука и техника, 2013. – 528 с.: ил.
4. 0054699\_59289\_atmega128\_datasheet\_ruskiy.djvu

### **1.3. Контрольные вопросы**

1. Как открыть новый проект в Proteus с подключением заданного микроконтроллера и встроенного компилятора его Ассемблера?
2. Как отыскать в библиотеке и размещать компоненты схемы в основном окне редактора?
3. Как устанавливать связи между выводами компонентов?
4. Способы разводки проводов и шин.
5. Создание текста программы на Ассемблере, компиляция и отладка программы.
6. Какие устройства входят в состав Вашего микроконтроллера? Каково назначение этих устройств?
7. Как организована память программ и память данных? Каковы способы адресации ячеек памяти?

8. Как программируются параллельные порты на ввод или на вывод?
9. Сколько портов параллельного обмена в Вашем микроконтроллере? Отличаются ли они друг от друга по своим свойствам?
10. Какие типы команд передачи управления присутствуют в системе команд Вашего микроконтроллера?
11. Каким образом осуществляется *косвенная* адресация в Вашем микроконтроллере?
12. Каким образом можно считывать данные из памяти программ?

## 2. Лабораторная работа №2

### Простые решения

В настоящей лабораторной работе мы познакомимся с методами подключения и использования в программах простых устройств ввода-вывода:

- двоичные индикаторы (светодиоды, LED);
- кнопки (BUTTON);
- пьезокерамический генератор звука (SOUNDER);
- одноразрядные 7-сегментные индикаторы.

Необходимо создать Proteus-проект на базе заданного семейства микроЭВМ и реализовать в нём следующие решения<sup>1</sup>.

#### 2.1. Светодиоды и кнопки

Создать новый проект (*Lab\_2.12\_⟨Фамилия⟩*) на заданном микроконтроллере и подключить к портам ОМЭВМ три кнопки (BUTTON) – K1, K2, K3 и три разноцветных светодиода (например LED-BLUE, LED-GREEN, LED-RED). Написать программу, обеспечивающую по каждому нажатию кнопки изменение текущего состояния соответствующего светодиода на противоположное («горит» - «не горит»). Каждой кнопке соответствует свой светодиод.

#### 2.2. Звук

Подключить дополнительно пьезокерамический элемент (SOUNDER) и написать программу, которая запускает звучание по нажатию кнопки K3 и прекращает – по нажатию K2.

Поскольку программы, реализующие задания 2.1 и 2.2 очень небольшие, их можно реализовать в рамках одного проекта, предусмотрев для каждой программы свою *точку старта*. Однако, даже для самых простых программ следует писать в ассемблерном тексте подробные комментарии!

---

<sup>1</sup>Все программы пишутся и отлаживаются только на Ассемблере соответствующей микроЭВМ.

## 2.3. Семисегментные индикаторы (статика)

Создать новый проект (*Lab\_2.3\_⟨Фамилия⟩*) на заданном микроконтроллере<sup>2</sup> и выполнить следующие действия:

- Шаг 1. Подключить один семисегментный индикатор с общим катодом (7SEG-MPX1-CC) и написать программу, которая при каждом нажатии на кнопку K1 выводит на семисегментный индикатор очередную десятичную цифру последовательности 0, 1, 2, ... 8, 9, 0, 1, ... Для этого сформировать таблицу семисегментных кодов десятичных цифр и разместить её в памяти программ.
- Шаг 2. Параллельно подключить к тем же выводам семисегментный индикатор с общим анодом (7SEG-MPX1-CA) и добавить в память программ ещё одну таблицу семисегментных кодов десятичных цифр для индикатора с общим анодом<sup>3</sup>.
- Шаг 3. Написать программу, которая по нажатию кнопки K1 выводит на индикацию очередную цифру, а нажатие кнопки K2 переключает активную кодовую таблицу, причём переключение таблицы должно сопровождаться коротким звуковым сигналом.
- Шаг 4. Дополнительно подключить к четырём линиям свободного порта ещё один семисегментный индикатор со встроенным декодером (7SEG-BCD-GRN) и выводить на него двоичный код индицируемой цифры.

Окончательно схема может принять, например, вид, показанный на рис. 12.

## 2.4. Содержание отчета

1. Описание используемых модулей памяти и способов доступа к ним.
2. Описание особенностей программирования портов ввода-вывода.
3. Схему принципиальную электрическую, разработанную для выполнения заданий 2.1 и 2.2.
4. Текст программы на языке Ассемблер, реализующей задания 2.1 и 2.2 с подробными комментариями.
5. Схему принципиальную электрическую, разработанную для выполнения задания 2.3.
6. Текст программы на языке Ассемблер, реализующей задания 2.3 с подробными комментариями.

## 2.5. Контрольные вопросы

1. Как настроить порт контроллера для работы в качестве выходного порта?
2. Как настроить порт контроллера для работы в качестве входного порта?

---

<sup>2</sup>Можно переименовать проект *Lab\_2.12*, после чего внести в принципиальную схему и программу нужные изменения.

<sup>3</sup>А можно сделать проще – использовать уже существующую таблицу, построенную на Шаге 1. Как?



3. Каким образом и для каких целей к линиям порта подключаются «подтягивающие резисторы»?
4. Как программа может обнаружить факт нажатия кнопки (замыкание контактов)?
5. От чего зависит длительность звучания и высота тона при работе с элементом SOUNDER?
6. В чём отличие в управлении семисегментным индикатором «с общим анодом» от индикатора «с общим катодом»?
7. Как размещать данные в памяти программ и использовать их в программе? Предусмотрена ли такая возможность в используемой Вами микроЭВМ?

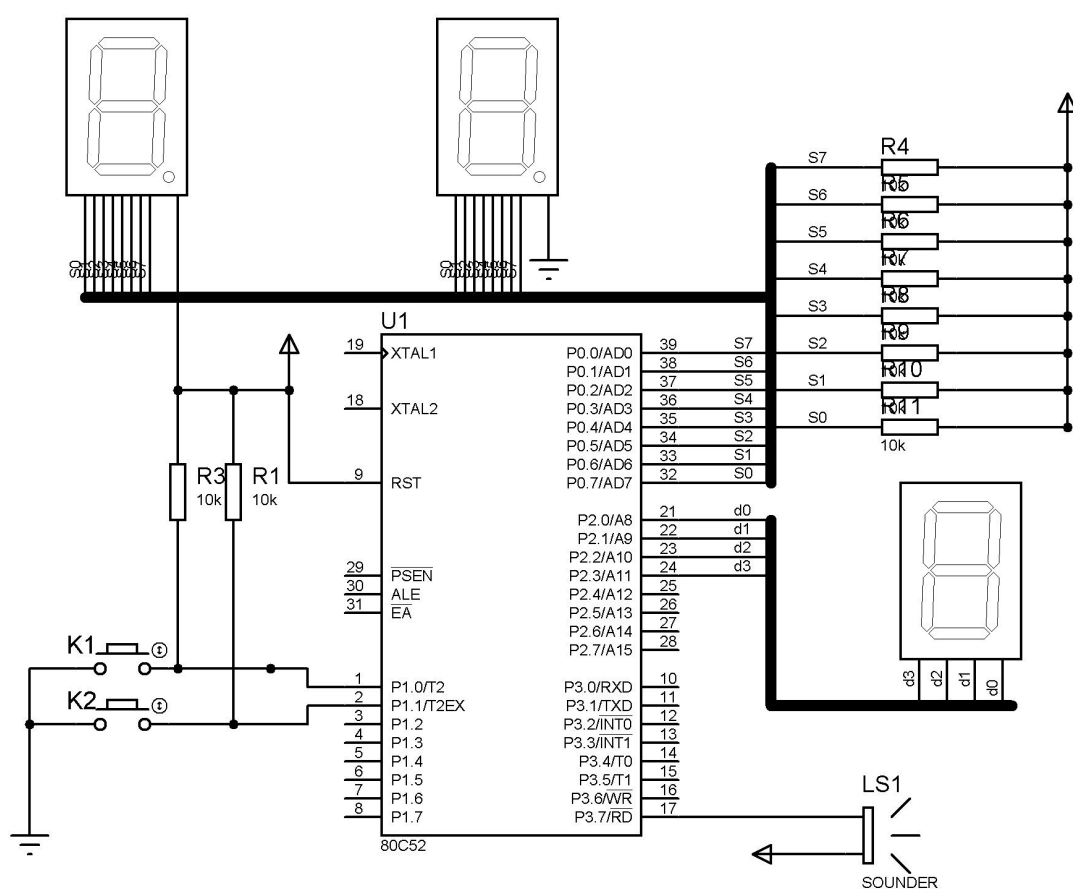


Рисунок 1. Микроконтроллер с подключёнными кнопками и индикаторами. Схема принципиальная электрическая

## 3. Лабораторная работа №3

### Динамическая семисегментная индикация и матричная клавиатура

В этой работе мы познакомимся со способами управления многоразрядной семисегментной индикацией и опросом матричной клавиатуры.



Принцип организации такой структуры на базе абстрактного 8-разрядного микроконтроллера показана на рис. 13. Порты X и Y конфигурированы как выходные, а порт Z – как входной.

Программа реализует бесконечный цикл, в теле которого на индикацию выводится очередной символ (его семисегментный код выводится на линии порта X) и одновременно сканируется один столбец матричной клавиатуры. При обнаружении факта нажатия клавиши определяется её код и выполняется действие, соответствующее нажатой клавише.




#### 3.1. Порядок выполнения работы

Шаг 1. Создать новый проект (*Lab\_3\_⟨Фамилия⟩*) на заданном микроконтроллере.

Шаг 2. Подключить к микроконтроллеру блок из четырёх семисегментных индикаторов (7SEG-MPX4-CC) и блок матричной клавиатуры 4 × 3 (KEYPAD-PHONE).

Шаг 3. Написать и отладить программу, которая вводит набранные на клавиатуре цифры в буфер, преобразует в семисегментный код и размещает в видеопамяти. Вновь введённая цифра попадает в младшую позицию 4-разрядного десятичного числа, остальные разряды сдвигаются влево, старшая цифра теряется. Текущее состояние видеопамяти постоянно индицируется в блоке 7SEG-MPX4-CC. Клавиши  и  пока не используются.

Шаг 4. Расширить функционал отлаженной программы следующим образом:

- 1) по нажатию клавиши  сбросить все набранные цифры;
- 2) по нажатию клавиши  текущее десятичное число преобразуется в шестнадцатеричный код и выводится на индикацию. Повторное нажатие  возвращает десятичный формат вывода.<sup>4</sup>

#### 3.2. Содержание отчета

1. Схема принципиальная электрическая, разработанная для выполнения задания лабораторной работы №3.
2. ГСА отлаженной программы с описанием переменных.
3. Текст программы на языке Ассемблер, полностью реализующей задание лабораторной работы №3 с подробными комментариями.

---

<sup>4</sup>Этот пункт задания выполняется **по желанию**. Его реализация позволит повысить уровень оценки.

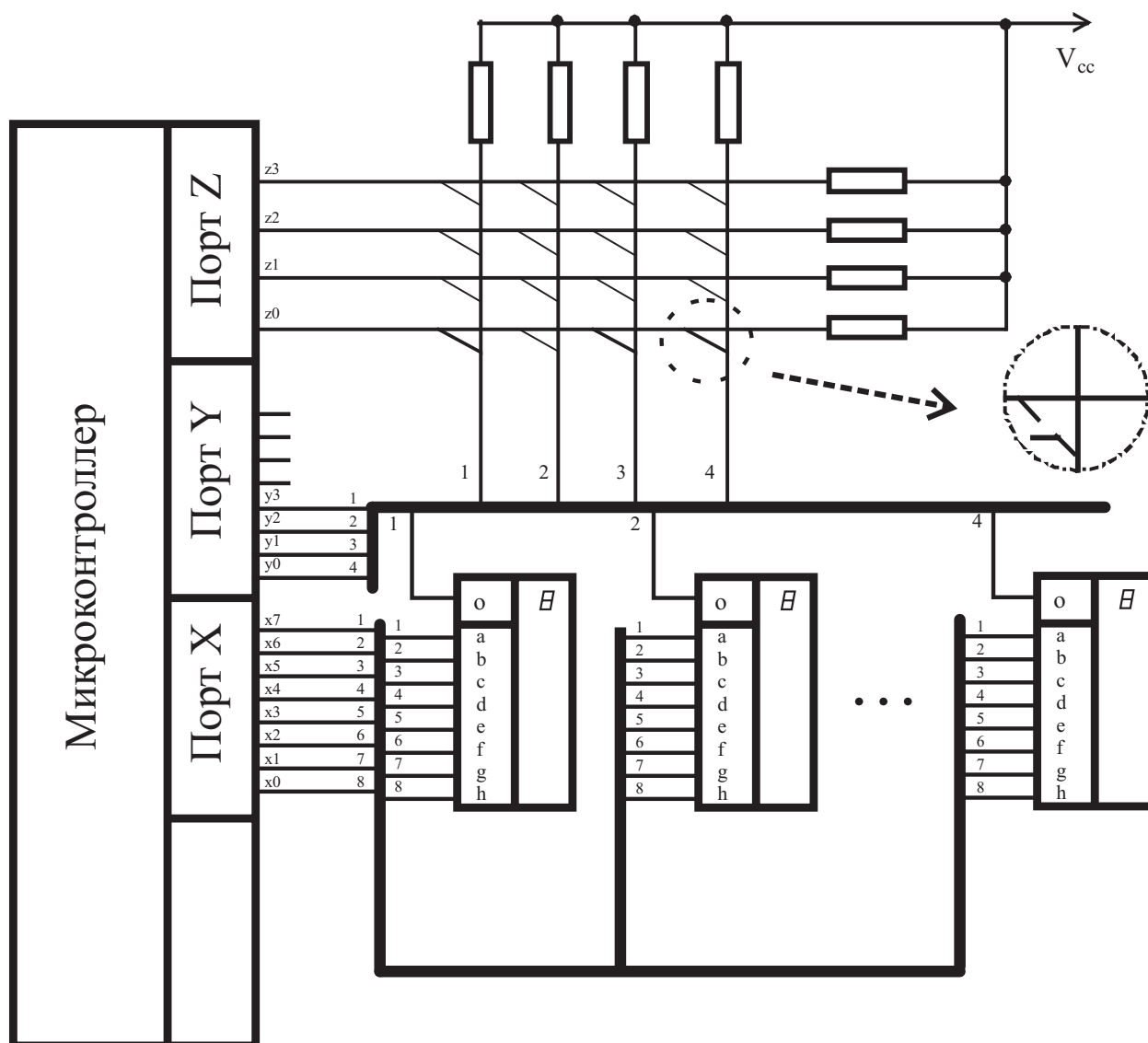


Рисунок 2. Матричная клавиатура и динамическая индикация

### 3.3. Контрольные вопросы

1. Каким образом обнаруживается факт нажатия клавиши?
2. Как определяется код нажатой клавиши?
3. Что произойдёт, если будут одновременно нажаты две клавиши<sup>5</sup> одного столбца? в разных столбцах?
4. Как перевести десятичное число в шестнадцатеричную систему счисления?
5. Какое максимальное шестнадцатеричное число можно отобразить в этой программе?

<sup>5</sup>На модели клавиатуры это не получится.

## 4. Лабораторная работа №4


### Таймерные системы и прерывания


В этой работе мы познакомимся со способами отсчёта заданных отрезков времени с помощью встроенных в микроконтроллер таймерных систем.

#### 4.1. Порядок выполнения работы

Используя опыт управления клавиатурой, 7-сегментной индикацией, генератором звука, полученный при выполнении лабораторных работ №2 и №3, разработать на базе заданного микроконтроллера таймер на задержку до 59 мин 59 сек (max).

С клавиатуры вводятся четыре цифры (минуты, секунды), ввод последующих цифр игнорируется.

Клавиша  – *Пуск* – запускает режим декремента задержки каждую 1 сек. При достижении текущего значения задержки = 0 прекращается декремент и генерируется звуковой сигнал.

Клавиша  – *Сброс* – останавливает счёт, прекращает звучание, очищает буфер задержки и переводит систему в режим ожидания ввода четырёх цифр задержки.

Для формирования счётных импульсов таймера частотой 1 Гц можно использовать разные механизмы управления внутренним таймером/счётчиком:

- 1) программную загрузку начального значения в таймер/счётчик;
- 2) автозагрузку с использованием специального регистра.

Определение факта переполнения внутреннего таймера/счётчика так же может выполняться двумя способами:

- 1) с помощью анализа значения флага переполнения таймера/счётчика;
- 2) с использованием механизма прерывания.

В работе необходимо рассмотреть все варианты использования внутреннего таймера/счётчика и реализовать соответствующие программы.

#### 4.2. Содержание отчета

1. Описание используемого в работе таймера/счётчика, расчёт времени переполнения (совпадения).
2. Схему принципиальную электрическую, разработанную для выполнения задания лабораторной работы №4.
3. ГСА отлаженной программы для одного из вариантов использования таймера/счётчика с описанием переменных.
4. Тексты программ на языке Ассемблер, полностью реализующих задание лабораторной работы №4, с различными вариантами использования внутреннего тайме-

ра/счётчика<sup>6</sup> и подробными комментариями.

### 4.3. Контрольные вопросы

1. Как можно рассчитать коэффициент пересчёта выбранного таймера/счётчика для получения заданной частоты его переполнения?
2. Какие особенности таймерной системы микроконтроллера используются Вами при выполнении задания лабораторной работы №4?
3. Каким образом настраивается подсистема прерываний микроконтроллера для реагирования на переполнение выбранного таймера/счётчика?
4. Какие функции выполняет в Вашей программе обработчик прерывания?
5. Что произойдёт в Вашей программе, если во время работы обработчика прерываний будет нажата клавиша?

---

<sup>6</sup>Если в паре вариантов программ имеются незначительные отличия, то можно привести только одну программу и фрагмент другой, отличающей её от первой.

## 5. Лабораторная работа №5

### Широтно-импульсная модуляция и АЦП

Работа посвящена изучению механизма широтно-импульсной модуляции, способов её реализации на базе однокристальных микроЭВМ, а так же изучению возможностей измерения напряжения с помощью АЦП микроконтроллера. Работу №5 все студенты выполняют на базе микроконтроллера семейства ATmega.

На рисунке 3 (стр. 16) показана примерная принципиальная электрическая схема, которая должна обеспечивать измерение и стабильность постоянного напряжения в точке UX при изменении тока нагрузки.

Стабилизатор выполнен на базе микроЭВМ ATmega64. **Импульсы ШИМ** генерируются на выводе 15 (PB5/OC1A) с помощью 16-разрядного таймера/счётчика TIMER1 (см. раздел 6.2), работающего в режиме 7 (быстрый 10-разрядный ШИМ, см. табл. 1 на стр. 26) и отображаются на канале А четырёхканального осциллографа.

RC-цепочка R4C1 (рис. 3) сглаживает пульсации ШИМ и обеспечивает в точке UX практически постоянное напряжение, величина которого определяется с одной стороны шириной импульса ШИМ, с другой стороны – потребляемым током нагрузки. Переменную нагрузку имитируют подключаемые резисторы R1, R2, R3. При увеличении тока нагрузки скорость разряда конденсатора C1 увеличивается и напряжение на нём падает. В этом случае для поддержания прежнего (заданного) уровня напряжения UX необходимо увеличить ширину импульсов ШИМ. При снижении тока нагрузки уровень UX возрастает и тогда следует уменьшить ширину импульса ШИМ. Форму напряжения UX можно наблюдать на канале В осциллографа.

Для визуального контроля величины напряжения UX используется внешний цифровой вольтметр. Для «ручного» управления шириной импульса ШИМ предусмотрены две кнопки – «+» и «-», подключённые к линиям порта D.

Наличие встроенного АЦП позволяет реализовать на базе ATmega64 **цифровой вольтметр**. Измеряемое напряжение поступает на один из аналоговых входов через операционный усилитель<sup>7</sup> U2.

ATmega64 содержит 10-разрядный многоканальный АЦП (см. раздел 6.3). Результат преобразования располагается в двух 8-разрядных регистрах ADCH.ADCL, причём допускается размещение 10-разрядного результата в 16-разрядном формате с правым или левым выравниванием. При *левом выравнивании* старшие 8 бит результата размещаются в ADCH, а двумя младшими разрядами при непрецизионном измерении можно и пренебречь. Поскольку аналого-цифровое преобразование здесь осуществляется в диапазоне от 0 до 5 вольт, значению 0x00 в ADCH соответствует напряжение 0,00 В, а значению 0xFF – 5,00 В. Программно следует осуществить преобразование байта результата в трёхразрядное десятичное число (два разряда после запятой) и вывести его

---

<sup>7</sup>Можно обойтись без операционного усилителя, подавая измеряемое напряжение UX непосредственно на аналоговый вход, однако использование операционного усилителя значительно увеличивает входное сопротивление вольтметра.

на индикацию.

Можно осуществить **автоматическую стабилизацию напряжения U<sub>X</sub>** на заданном уровне U<sub>E</sub>, для чего достаточно периодически измерять текущее значение U<sub>X</sub>, сравнивать его с U<sub>E</sub> и, в зависимости от знака разности U<sub>E</sub>-U<sub>X</sub>, увеличивать или уменьшать ширину импульсов ШИМ.

## 5.1. Порядок выполнения работы

1. Создать новый проект на базе микросхемы ATmega64.
2. Настроить режим ШИМ, подключить к выходу ШИМ интегрирующую RC-цепочку, а к выходу постоянного напряжения U<sub>X</sub> – нагрузочные резисторы с переключателями и контрольный вольтметр, подключить осциллограф. К одному из входных портов подключить кнопки. Отладить программу, позволяющую по нажатию кнопки «+» или «-» соответственно изменять ширину импульса ШИМ.
3. Подключить (через операционный усилитель) выходное напряжение U<sub>X</sub> к одному из аналоговых входов. Написать процедуру измерения по нажатию кнопки значения U<sub>X</sub> и выдачи байта результата на один из выходных портов, к которому подключена пара индикаторов 7SEG-BCD.
4. Добавить процедуру перевода шестнадцатеричного результата измерения в десятичный в формате «u,uu Вольт». Подключить дополнительно индикатор 7SEG-BCD для вывода результата в указанном формате.
5. Написать процедуру стабилизации напряжения на заданном уровне.

## 5.2. Содержание отчета

1. Описание используемого в работе таймера/счтчика, обоснование выбора режима его работы.
2. Описание АЦП
3. Схему принципиальную электрическую, разработанную для выполнения задания лабораторной работы №.
4. ГСА отлаженной программы с описанием переменных.
5. Текст программы на языке Ассемблер, полностью реализующей задание лабораторной работы №5 с подробными комментариями.





### 5.3. Контрольные вопросы

1. Какими параметрами определяется частота (период) ШИМ-импульсов?
2. Какими параметрами определяется ширина ШИМ-импульса в выбранном режиме работы таймера/счётчика?
3. В чём отличие «быстрой ШИМ» от ШИМ с фазовой и/или частотной коррекцией?
4. Как запускается аналого-цифровое преобразование и как определяется момент завершения преобразования?
5. Как можно перевести результат АЦП (байт – при левом выравнивании) в единицы измерения напряжения (вольты, милливольты)?
6. Попробуйте уменьшить ёмкость конденсатора C1 (например, на порядок). Как это повлияет на форму UX?

## 6. Приложения

### 6.1. Открыть новый проект в Proteus

Шаг 1. Запустить Proteus и в окне *Старт* выбрать команду *Новый проект*.

Шаг 2. Задать имя файла и путь к нему (рис. 4).

Шаг 3. Выбрать переключатель *Создать схему с шаблоном* – будем проектировать принципиальную схему, размер листа – «по умолчанию» (DEFAULT) (рис. 5).

Шаг 4. Выбрать переключатель *Не создавать печатную плату* (рис. 6).

Шаг 5. Выбрать переключатель *Создать проект прошивки*, выбираем заданное семейство и конкретный микроконтроллер. В качестве компилятора с Ассемблера целесообразно выбрать предлагаемый, встроенный в Proteus, продукт (рис. 7).

Шаг 6. Контролируем свой выбор (при необходимости можно вернуться назад) (рис. 8) и приступаем к проектированию.

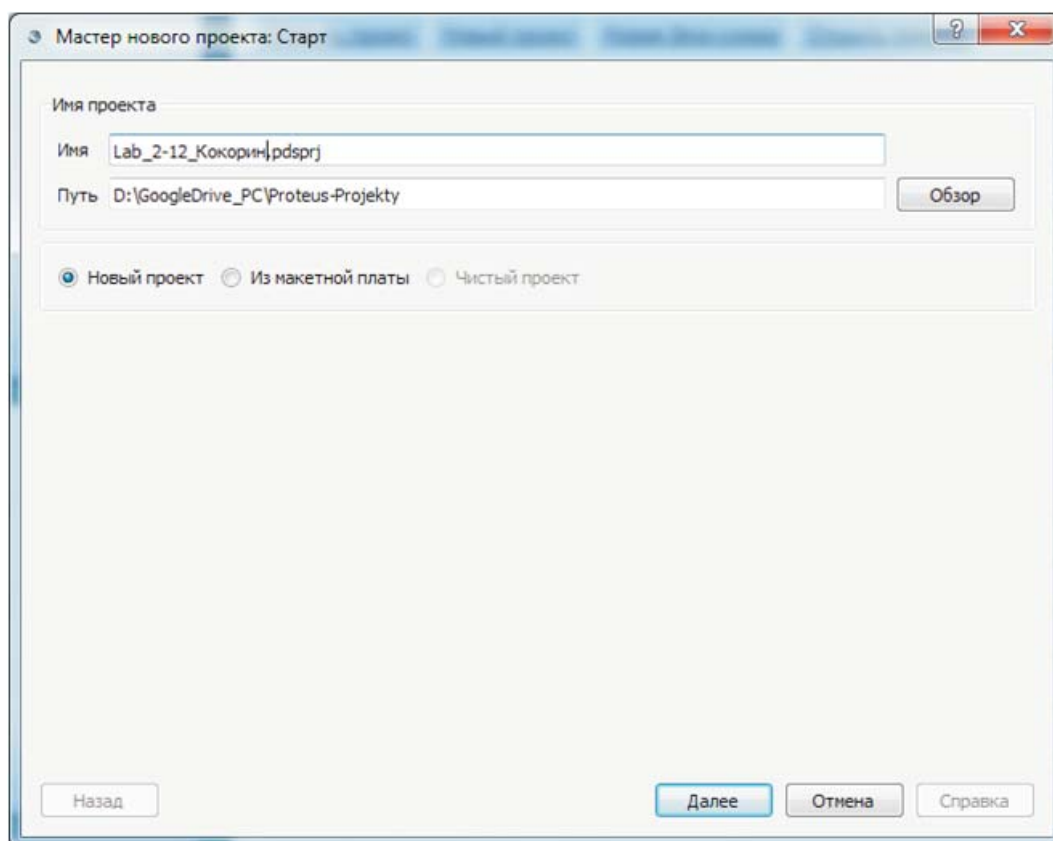


Рисунок 4. Задать имя файла и путь к нему

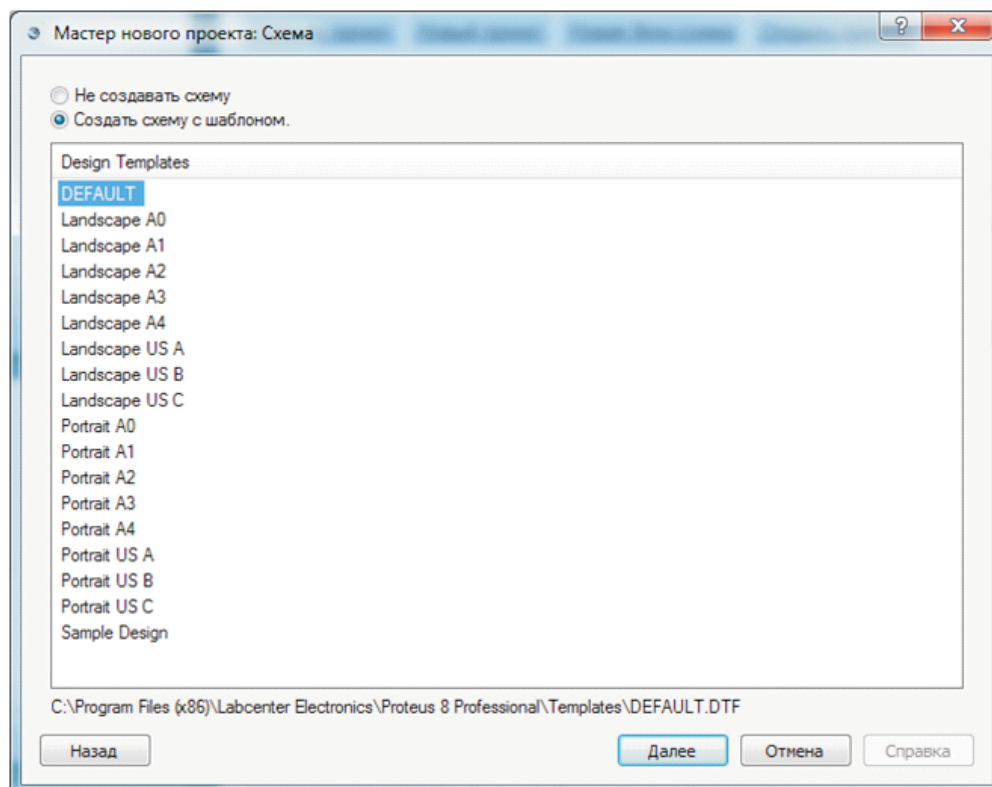


Рисунок 5. Выбираем разработку принципиальной схемы

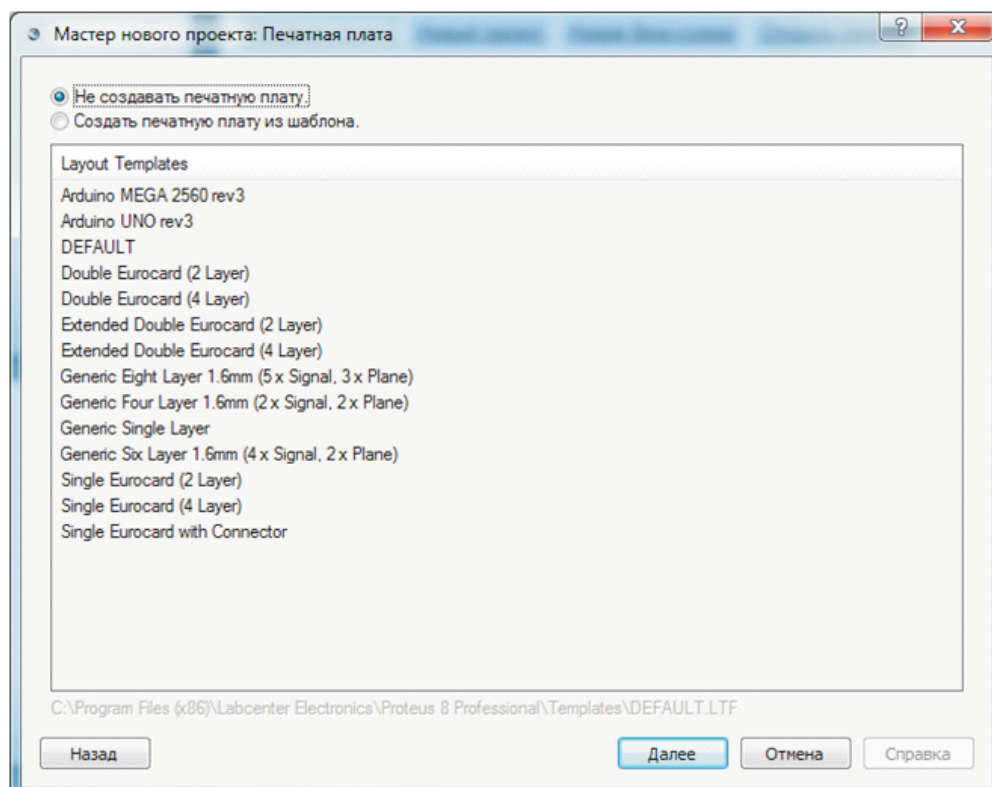


Рисунок 6. Откажемся от разработки печатной платы

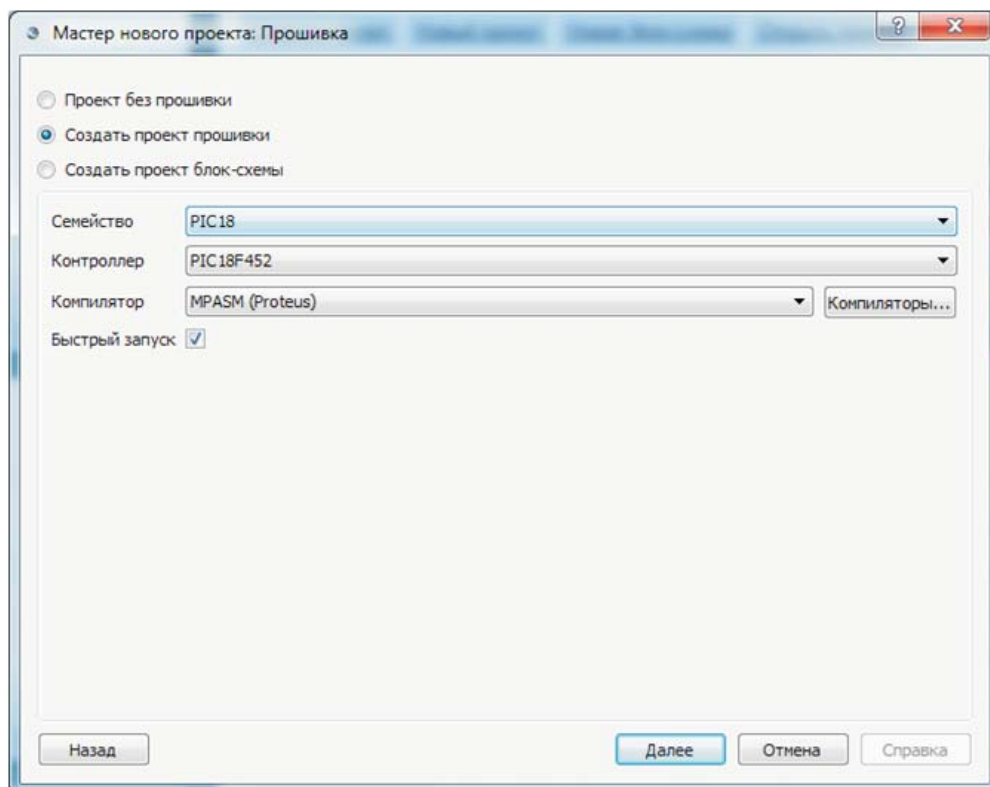


Рисунок 7. Выбираем тип микроконтроллера и компилятор с языка Ассемблер

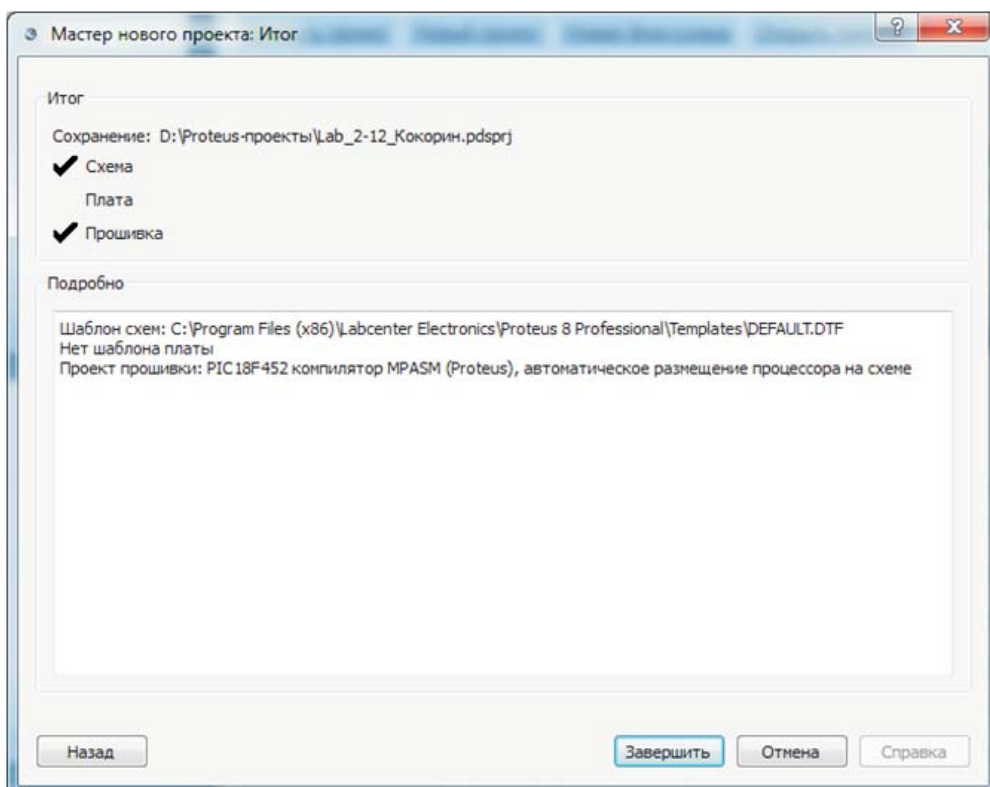


Рисунок 8. Проверяем выбор, корректируем при необходимости

## 6.2. 16-разрядные таймеры-счётчики с режимами ШИМ в микроЭВМ семейства ATmega

В микроконтроллерах ATmega предусмотрены несколько одинаково организованных 16-разрядных таймеров-счётчиков (рис. 9).

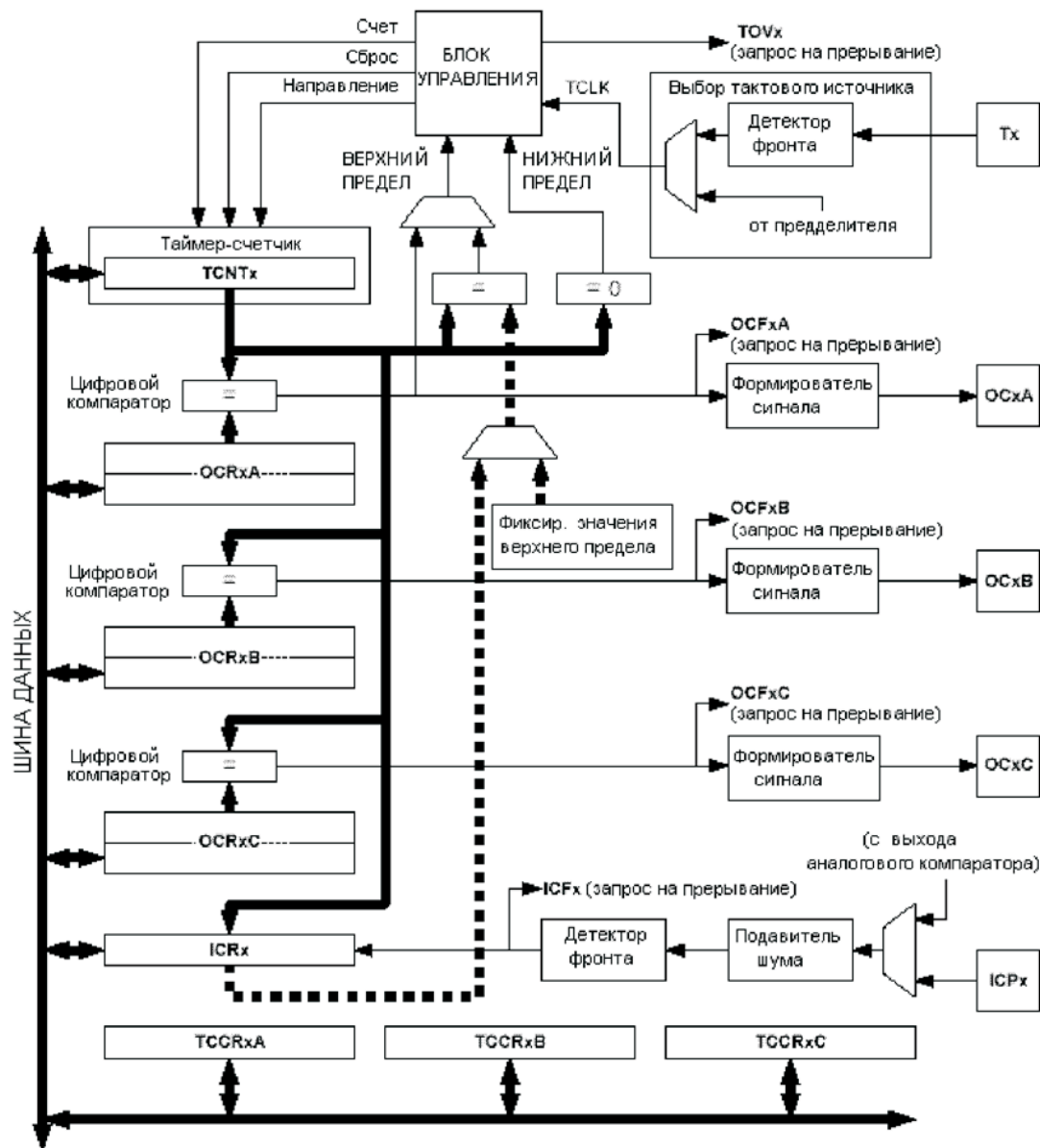


Рисунок 9. 16-разрядный таймер ATmega

В состав каждого таймера/счётчика входят следующие регистры ввода/вывода ( $n \in \{1, 3, [4, 5]\}$  – номер счётчика):

- 16-битный счетный регистр TCNT<sub>n</sub>;
- 16-битный регистр захвата ICR<sub>n</sub>;
- два или три 16-битных регистра сравнения OCR<sub>nA</sub>, OCR<sub>nB</sub>, OCR<sub>nC</sub>;

- два или три 8-битных регистра управления TCCRnA, TCCRnB, [TCCRnC].

### 6.2.1. Программируемый реверсивный 16-разрядный счетчик

Основным элементом 16-разрядного таймера-счетчика является программируемый реверсивный 16-разрядный счетчик. На рисунке 10 представлена функциональная схема счетчика и окружающих его элементов.

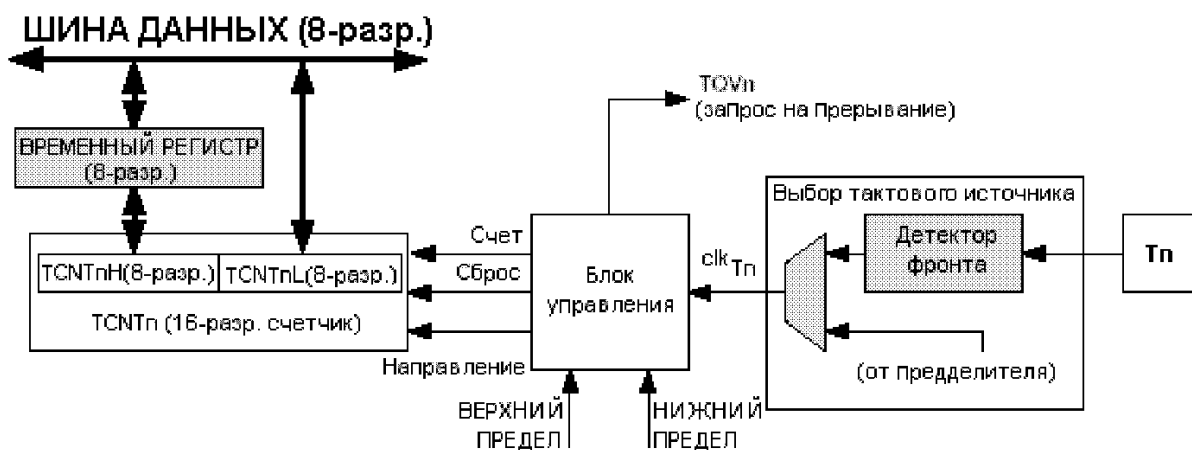


Рисунок 10. Функциональная схема счетчика

Внутренние сигналы:

*clkTn* - синхронизация таймера-счетчика.

*Счёт* - инкрементирует или декрементирует состояние TCNTn.

*Направление* - задает прямой счёт (инкрементирование) или обратный счёт (декрементирование).

*Сброс* - установка всех разрядов TCNTn в лог. «0».

*Верхний предел* - сигнализирует о достижении TCNTn заданного в выбранном режиме предельного значения.

*Нижний предел* - сигнализирует о достижении TCNTn минимального значения.

Если *Нижний предел* всегда равен 0, а *Максимальное значение* – 0xFFFF, то *Верхний предел* может принимать фиксированные значения 0x00FF, 0x01FF, 0x03FF или определяться содержимым регистров OCRnA или ICRn.

### 6.2.2. Доступ к 16-разрядным регистрам

Чтобы обеспечить одновременный доступ ко всем разрядам 16-разрядного регистра в ЭВМ с 8-разрядной шиной, предусмотрен **механизм буферирования**. Содержимое 16-разрядного счетчика разбито на две 8-разрядные ячейки, расположенных в памяти ввода-вывода: TCNTnH – старший байт счётчика, TCNTnL – младший байт счетчи-

ка, причём ЦПУ вместо непосредственного доступа к регистру TCNTnH обращается к специальному 8-разрядному *временному регистру*.

Очевидно, обращение к счётчику TCNTn должно осуществляться двумя последовательными командами. При записи первым загружается значение старшего байта – TCNTnH, которое автоматически помещается во временный регистр. Затем, когда поступает команда на запись младшего байта TCNTnL, оба значения объединяются и запись производится одновременно в обе «половинки» 16-разрядного регистра.

Наоборот, при чтении первым должен быть прочитан младший байт, при этом значение старшего автоматически фиксируется в тот же временный регистр, и при следующей операции чтения старшего байта его значение извлекается оттуда. Таким образом, и при чтении значения двух байтов TCNTn соответствуют одному и тому же моменту времени.

Аналогично осуществляется доступ и к регистрам сравнения OCRnA, OCRnB, OCRnC.

Повторим: для того чтобы манипуляции с 16-разрядными регистрами были успешными, необходимо:

**при чтении** – сначала прочесть младший байт \*RL, потом старший \*RH.

**при записи** – сначала записать старший байт \*RH, потом младший \*RL.

В состав каждого 16-разрядного таймера-счётчика входит единственный временный регистр, который используется для обращения ко всем 16-разрядным регистрам таймера-счётчика, поэтому если выполняется запись в несколько 16-разрядных регистров и при этом значение старшего байта одинаково для всех регистров, то достаточно однократно выполнить запись старшего байта. Однако при этом также следует учитывать возможность некорректного завершения такой операции, если используются прерывания.

### 6.2.3. Блок сравнения

16-разрядный цифровой компаратор непрерывно сравнивает значение TCNTn со значением регистра порога сравнения (OCRnx). Если значение TCNT равно OCRnx, то компаратор формирует сигнал совпадения (равенства). Следующий за совпадением такт ЦПУ устанавливает флаг сравнения (OCFnx).

Если бит OCIEnx = 1, то установка флага сравнения приведет к генерации запроса на прерывание по результату сравнения. Флаг OCFnx автоматически сбрасывается после перехода на вектор обработки прерывания. Этот же флаг сбрасывается программно, если записать в него логическую «1».

Сигнал совпадения используется формирователем выходного сигнала, результирующая форма которого зависит от выбранного режима работы таймера (см. табл. 1) и режима формирования выходных импульсов (см. табл. 2).

Если задан любой из 12 режимов широтно-импульсной модуляции, то доступ в регистры OCRnx осуществляется с **двойной буферизацией**, иначе двойная буферизация отключается.

Двойная буферизация синхронизирует обновление регистра порога сравнения OCR<sub>n</sub>x по достижении верхнего или нижнего предела счета в зависимости от выбранного режима работы. Такая синхронизация предотвращает возможность возникновения несимметричных ШИМ-импульсов нечетной длины, тем самым гарантируя отсутствие сбоев при генерации прямоугольных импульсов.

Рисунок 11. Блок сравнения

При двойной буферизации ЦПУ фактически осуществляет доступ к буферному регистру  $OCR_{lx}$ . Если же двойная буферизация отключена, то ЦПУ обращается к регистру  $OCR_{lx}$  непосредственно. Содержимое регистра  $OCR_{lx}$  (в т.ч. и буферного) может измениться только путем непосредственной записи в него (таймер-счетчик не обновляет содержимое данного регистра автоматически аналогично регистрам  $TCNT_n$  и  $ICR_n$ ). Таким образом,  $OCR_{lx}$  считывается напрямую, а не через временный регистр старшего байта.



#### 6.2.4. Блок захвата

Таймер-счетчик содержит блок захвата (рис. 12), который запоминает состояние счетчика при возникновении внешнего события, тем самым определяя время его возникновения.

В качестве события выступает внешний сигнал, подключенный к выводу ICPn. Для таймера-счетчика 1 может использоваться и аналоговый компаратор в качестве источника внешнего события. Результат захвата состояния таймера может использоваться для вычисления частоты, скважности импульсов и других параметров импульсных сигналов, для создания журнала событий.

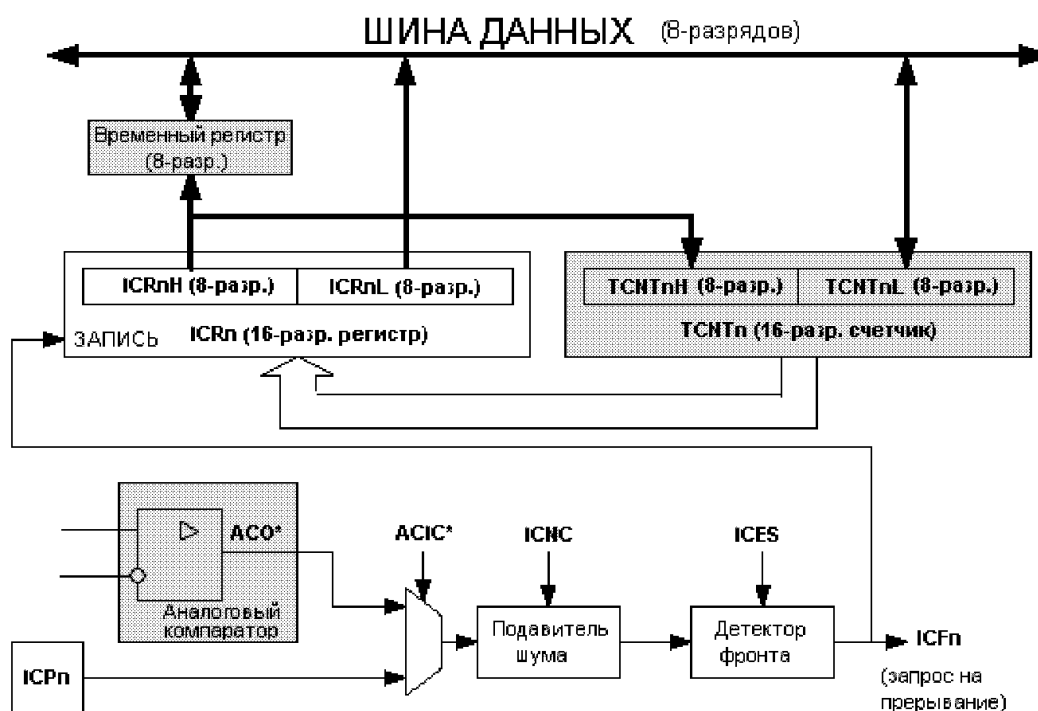


Рисунок 12. Блок захвата

Чтение и запись в ICRn производится по описанным выше правилам доступа в 16-разрядный регистр, причём запись в регистр ICRn возможна только в том случае, если выбран режим работы таймера, в котором значение регистра ICRn задает верхний предел счета.

#### 6.2.5. Режимы работы

СТС – сброс по совпадению;

БШИМ – быстрая широтно-импульсная модуляция;

ШИМ ФК – широтно-импульсная модуляция с фазовой коррекцией;

ШИМ ФЧК – широтно-импульсная модуляция с фазовой и частотной коррекцией.

Таблица 1. Режимы работы 16-разрядного таймера-счётчика

№ реж.	WGM [3:0]	Режим работы	Верхний предел	Обновление OCR	Установка флага TOV
0	0000	Нормальный	0xFFFF	сразу после записи	МАКС
1	0001	ШИМ ФК 8-разр.	0x00FF	на вершине счёта	на нижнем пределе
2	0010	ШИМ ФК 9-разр.	0x01FF	на вершине счёта	на нижнем пределе
3	0011	ШИМ ФК 10-разр.	0x03FF	на вершине счёта	на нижнем пределе
4	0100	СТС	OCRnA	сразу после записи	МАКС
5	0101	БШИМ 8-разр.	0x00FF	на вершине счёта	на вершине счёта
6	0110	БШИМ 9-разр.	0x01FF	на вершине счёта	на вершине счёта
7	0111	БШИМ 10-разр.	0x03FF	на вершине счёта	на вершине счёта
8	1000	ШИМ ФЧК	ICRn	на нижнем пределе	на нижнем пределе
9	1001	ШИМ ФЧК	OCRnA	на нижнем пределе	на нижнем пределе
10	1010	ШИМ ФК	ICRn	на вершине счёта	на нижнем пределе
11	1011	ШИМ ФК	OCRnA	на вершине счёта	на нижнем пределе
12	1100	СТС	ICRn	сразу после записи	МАКС
13	1101	(резерв)	–	–	–
14	1110	БШИМ	ICRn	на вершине счёта	на вершине счёта
15	1111	БШИМ	OCRnA	на вершине счёта	на вершине счёта

#### 6.2.6. Регистры управления таймером-счётчиком

	7	6	5	4	3	2	1	0
<b>TCCR1A</b>	COM1A[1:0]		COM1B[1:0]		COM1C[1:0]		WGM1[1:0]	

	7	6	5	4	3	2	0
<b>TCCR1B</b>	ICNC1	ICNS1	–	WGM1[3:2]		CS1[2:0]	

	7	6	5	4	0
<b>TCCR1C</b>	FOC1A	FOC1B	FOC1C	—	

COM[1:0] – определяет режим формирования выходного сигнала (табл. 2);

WGM[3:0] – задаёт режим работы (табл. 1);

ICNC – включает («1») подавитель шума на входе захвата;

ICNS – задаёт фронт на входе захвата («1» – передний);

CS[2:0] – определяет источник тактового сигнала (табл. 3);

FOC – бит принудительной установки результата сравнения.

Таблица 2. Режимы формирования выходного сигнала

СОМ[1:0]	без ШИМ	быстрый ШИМ	ШИМ ФК и ФЧК
0 0	Сигналы ОСнА/В/С отключены	Сигналы ОСнА/В/С отключены	Сигналы ОСнА/В/С отключены
0 1	Инверсия при совпадении	Инверсия при совпадении – только для ОСнА в режиме 15; в остальных режимах ОСнА и во всех режимах ОСнВ/С отключены	Инверсия при совпадении – только для ОСнА в режимах 9 или 14; в остальных режимах ОСнА и во всех режимах ОСнВ/С отключены
1 0	Сброс («0») при совпадении	Сброс при совпадении, установка на вершине счёта	Сброс при совпадении во время прямого счёта, установка при совпадении во время обратного счёта
1 1	Установка («1») при совпадении	Установка при совпадении, сброс на вершине счёта	Установка при совпадении во время прямого счёта, сброс при совпадении во время обратного счёта

Таблица 3. Источник тактового сигнала

CS[2:0]	Источник счётных импульсов
000	Нет синхронизации, счётчик остановлен.
001	$clk_{IO}/1$
010	$clk_{IO}/8$
011	$clk_{IO}/64$
100	$clk_{IO}/256$
101	$clk_{IO}/1024$
110	По заднему фронту внешнего источника с вывода Тп
111	По переднему фронту внешнего источника с вывода Тп

### 6.3. АЦП микроЭВМ семейства ATmega

ATmega64 содержит 10-разрядный АЦП последовательного приближения. АЦП связан с 8-канальным аналоговым мультиплексором, 8 однополярных входов которого связаны с линиями порта F.

В качестве внутреннего опорного напряжения может выступать напряжение от внутреннего ИОН<sup>8</sup> на 2,56 В или напряжение AVCC.

АЦП преобразовывает входное аналоговое напряжение в 10-разр. код методом последовательных приближений. Минимальное значение соответствует уровню GND, а максимальное уровню AREF минус 1 мл. разр. К выводу AREF может быть подключено напряжение AVCC или внутренний ИОН на 1.22В путем записи соответствующих значений в биты REFSn в регистр ADMUX. Несмотря на то, что ИОН на 2.56В находится внутри микроконтроллера, к его выходу может быть подключен блокировочный конденсатор для снижения чувствительности к шумам, т.к. он связан с выводом AREF.

Работа АЦП разрешается путем установки бита ADEN в ADCSRA. Выбор опорного источника и канала преобразования не возможно выполнить до установки ADEN. Если ADEN = 0, то АЦП не потребляет ток, поэтому, при переводе в экономичные режимы сна рекомендуется предварительно отключить АЦП. АЦП генерирует 10-разрядный результат, который помещается в пару регистров данных АЦП ADCH и ADCL. По умолчанию результат преобразования размещается в младших 10-ти разрядах 16-разр. слова (выравнивание справа), но может быть размещен в старших 10-ти разрядах (выравнивание слева) путем установки бита ADLAR в регистре ADMUX. В этом случае можно, пренебрегая точностью, использовать только регистр ADCH, содержащий 8 старших разрядов результата измерения.

Одиночное преобразование запускается путем записи логической «1» в бит запуска преобразования АЦП ADSC. Данный бит остается в высоком состоянии в процессе преобразования и сбрасывается по завершении преобразования. Если в процессе преобразования переключается канал аналогового ввода, то АЦП автоматически завершит текущее преобразование прежде, чем переключит канал.

---

<sup>8</sup>Источник Опорного Напряжения

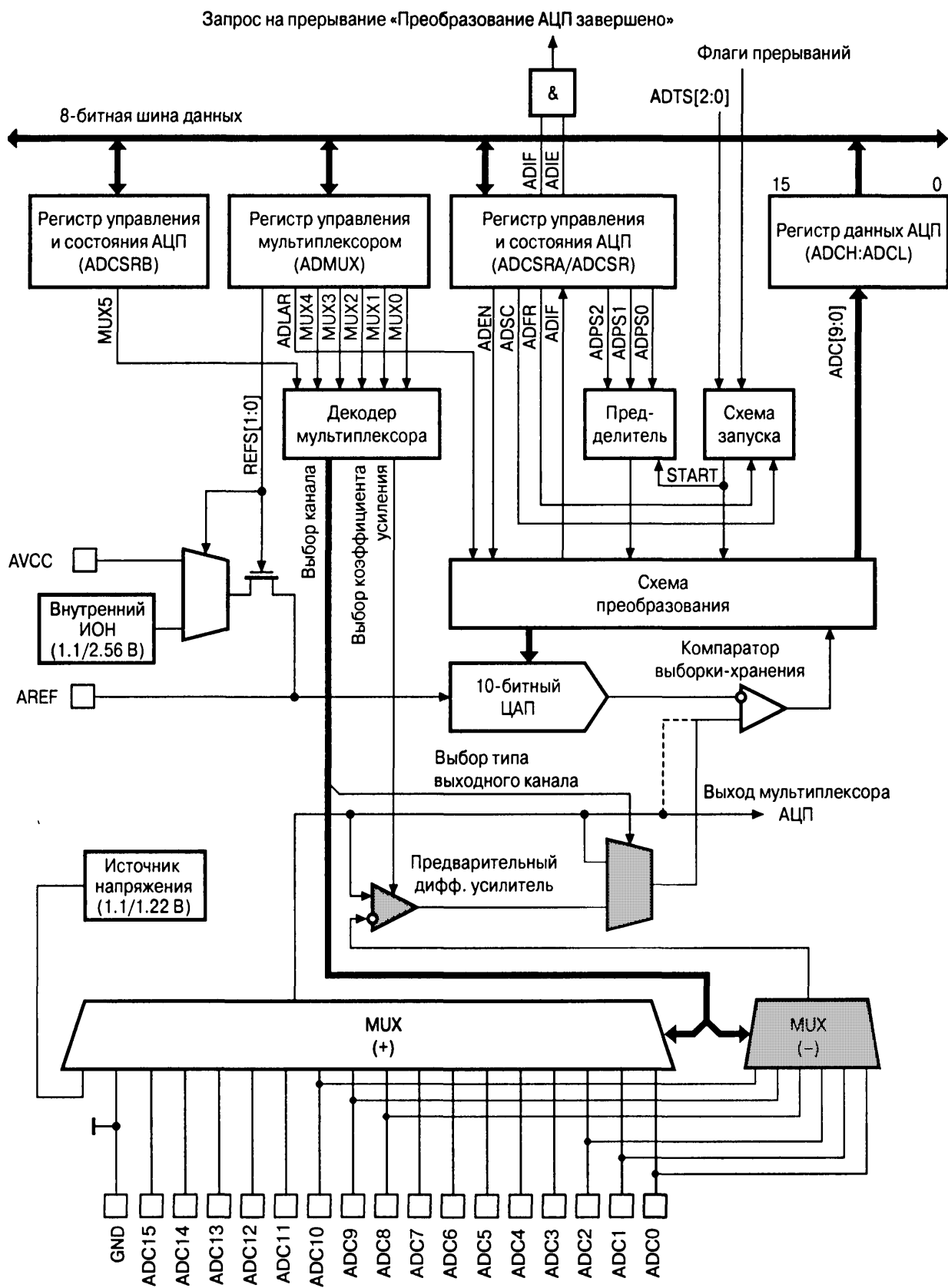


Рисунок 13. Структурная схема модуля АЦП

## Регистры управления и состояния АЦП (АТmega64)

	7	6	5	4	3	2	0
<b>ADCSRA</b>	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS[2:0]	

ADEN – Разрешение АЦП («1» – включено, «0» – выключено);

ADSC – Запуск преобразования («1» – начать преобразование);

ADATE – Выбор режима работы АЦП;

ADIF – Флаг прерывания от АЦП;

ADIE – Разрешение прерывания от АЦП;

ADPS[2:0] – Выбор частоты преобразования. Коэффициент деления предделителя определяется выражением  $2^{ADPS}$ .

Во всех моделях, кроме АТmega8х и АТmega128х, запуск АЦП возможен не только по команде пользователя, но и по прерыванию от некоторых периферийных устройств, имеющих в составе микроконтроллера. Для выбора режима работы в этих моделях используется бит ADATE регистра ADCSRA и биты ADTDS[2:0] регистра ADCSRB.

Если бит ADATE сброшен в 0, АЦП работает в режиме одиночного преобразования. Если же бит ADTAE установлен в 1, функционирование АЦП определяется содержанием битов ADTS[2:0] регистра ADCSRB (табл. 4).

	7	3	2	0
<b>ADCSRB</b>	—			ADTS[2:0]

Таблица 4. Источник сигнала для запуска преобразования

ADTS[2:0]	Источник стартового сигнала
000	Режим непрерывного преобразования
001	Прерывание от аналогового компаратора
010	Внешнее прерывание INT0
011	Прерывание по событию «Совпадение» («Совпадение А») таймера/счетчика T0
100	Прерывание по переполнению таймера/счетчика T0
101	Прерывание по событию «Совпадение В» таймера/счетчика T1
110	Прерывание по переполнению таймера/счетчика T1
111	Прерывание по событию «Захват» таймера/счетчика T1

Поля регистра ADMUX позволяют назначить входной канал для измерения, выбрать источник опорного напряжения и определить способ выравнивания 10-разрядного результата в 16-разрядном регистре ADC.

	7	6	5	4	0
<b>ADMUX</b>	REFS[1:0]		ADLAR	MUX[4:0]	

REFS[1:0] – Выбор источника опорного напряжения: 00 – AREF, внутренний ИОН отключён; 01 – AVCC; 10 – зарезервировано; 11 – внутренний ИОН 2,56 В.

ADLAR – Выравнивание результата преобразования: «0» – правостороннее выравнивание, «1» – левостороннее.

MUX[4:0] – Выбор входного канала. Для однополярных измерений без внутреннего усиления MUX[4:3] = 00, а MUX[2:0] соответствует номеру входного канала ADC.