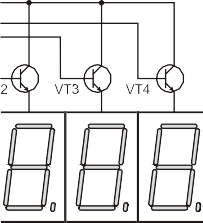
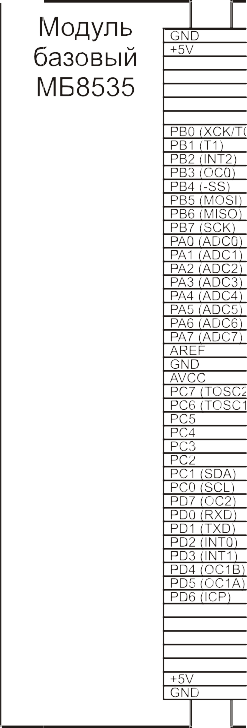
Лабораторная работа № 6



# Использование внутреннего АЦП

Цель работы:

1. Изучение принципа работы внутреннего аналого-цифрового преобра- зователя (АЦП) микроконтроллера ATmega8535.
2. Программирование режимов работы АЦП.

Оборудование и программное обеспечение:

1. Лабораторный макет MK8535 в составе: модуль базовый МБ8535 +

модуль сменный МС02.

1. Кабель соединительный для программирования МК.
2. Персональный компьютер.
3. Интегрированная среда программирования CodeVisionAVR.

# Подготовка к работе

Рассмотрим и выполним пример, в котором на основе АЦП реа- лизован простейший цифровой вольтметр. Показания вольтметра выво- дятся на 4-разрядный десятичный индикатор модуля МС02.

Структурная схема лабораторного макета показана на рис. 45. В качестве опорного напряжения для АЦП используем напряжение на входе аналогового питания AVCC (+5B), отсюда диапазон измеряемых напряжений от 0 В до +5 В. Показания выводятся на индикатор с тремя знаками после запятой. Стабилитроны VD1…VD3 номиналом 5,6 В совместно с резисторами R13…R15 защищают входы МК от напряже- ний выше 5 В и ниже нуля.

Для организации индикатора воспользуемся процедурами из ла- бораторной работы № 5 «Использование динамической индикации», добавив лишь индикацию точки после старшего разряда.

Создадим проект в CodeVisionAVR и зададим нужный режим ра- боты МК. Линии 4…7 порта A настроить на вывод. Их начальное значе- ние – 0. Линии 0…7 порта C также настроить на вывод. Их начальное значение – 1. Эти настройки нужны для организации индикатора.

На вкладке **ADC** окна **CodeWizardAVR** разрешим работу АЦП, выберем тактовую частоту и режим одиночных преобразований (рис. 46).

Рис. 45. Структурная схема лабораторного макета

Рис. 46. Настройка АЦП на вкладке ADC



В сформированном **CodeWizardAVR** тексте программы обратите внимание предлагаемую системой функцию **read\_adc().**

# // Read the AD conversion result

**unsigned int read\_adc(unsigned char adc\_input)**

**{**

**ADMUX = adc\_input | ADC\_VREF\_TYPE;**

**ADCSRA |= 0x40; // Start the AD conversion while ((ADCSRA & 0x10)==0); // Wait for the AD**

**// conversion to complete ADCSRA |= 0x10; // Сбросить флаг ADIF return ADCW;**

**}**

**#include <mega8535.h> #include <delay.h>**

**#define ADC\_VREF\_TYPE 0x40**

**// Вып. одно преобразование и возвр. результат**

**unsigned int read\_adc(unsigned char adc\_input)**

**{**

**ADMUX = adc\_input | ADC\_VREF\_TYPE;**

**ADCSRA |= 0x40; // Запуск АЦП**

**// Ждать конец преобразования**

**while ((ADCSRA & 0x10)==0);**

**ADCSRA |= 0x10; // Сбросить флаг ADIF return ADCW;**

Эта функция запускает АЦП, дожидается конца преобразования и возвращает результат в формате 16 бит. Номер входного канала задан параметром **adc\_input**.

Работа вольтметра обеспечивается путем циклических программ- ных запусков АЦП и непрерывной индикации результата.

Измеренное напряжение вычисляется по формуле

**}**

# struct

**{**

# unsigned char d0; // значение 0 разряда unsigned char d1; // значение 1 разряда unsigned char d2; // значение 2 разряда

*VIN*

  *ADC* *VREF* .

1024

# unsigned char d3; // значение 3 разряда

**} dig;**

Чтобы использовать все разряды индикатора, значение *VIN* допол- нительно умножается на 1000. Получился индикатор «с тремя знаками после запятой». Однако, следует понимать, что наш вольтметр не разли- чает изменений напряжения, меньших 0,005 В (*VREF* /1024), и потому третий знак после запятой является недействительным. Здесь он служит лишь для иллюстрации действия шумов. Также напомним, что в нашем макете *VREF* = *AVCC* = 5 В. С учетом сказанного формула для измеренно- го напряжения примет вид

# void IntToDigits(int v)

**{**

**dig.d0 = v%10; // выделить 0 разряд**

**v /= 10;**

**dig.d1 = v%10; // выделить 1 разряд**

**v /= 10;**

**dig.d2 = v%10; // выделить 2 разряд**

**v /= 10;**

**dig.d3 = v%10; // выделить 3 разряд**

*VIN*

**}**

 1000  *ADC*  5, 00 .

1024

Ниже приведен текст программы для МК ATmega8535. В нем опущены те строки, автоматически сформированные **CodeWizardAVR**, которые не относятся непосредственно к данному примеру.

# void ShowDigits()

**{**

**// таблица дешифратора цифр 0...9**

**static unsigned char BitSet[10] = {0x41,0x77, 0x29,0x23,0x17,0x83,0x81,0x67,0x01,0x03};**

**PORTC = BitSet[dig.d0]; // цифра 0 разряда PORTA = 0x10; // зажечь 0 разряд delay\_ms(1);**

**PORTA = 0x00; // погасить 0 разряд PORTC = BitSet[dig.d1]; // цифра 1 разряда PORTA = 0x20; // зажечь 1 разряд delay\_ms(1);**

**PORTA = 0x00; // погасить 1 разряд PORTC = BitSet[dig.d2]; // цифра 2 разряда PORTA = 0x40; // зажечь 2 разряд delay\_ms(1);**

**PORTA = 0x00; // погасить 2 разряд**

**// цифра 3 разряда + точка**

**PORTC = BitSet[dig.d3] & 0xFE;**

**PORTA = 0x80; // зажечь 3 разряд**

**delay\_ms(1);**

**PORTA = 0x00; // погасить 3 разряд**

**}**

# while (1)

**{**

# Vin = 1000 \* (read\_adc(0) \* 5.00 / 1024); IntToDigits(Vin);

**ShowDigits();**

**}**

**}**

# Задание

1. Создать проект и реализовать рассмотренный выше пример программы.
2. Проследить, изменяется ли качество измерений на различных такто- вых частотах АЦП ?
3. Организовать работу вольтметра по методу усреднения многократных выборок.
4. Организовать работу вольтметра с дифференциальным входами

ADC0 и ADC1.

# void main(void)

**{**

**unsigned int Vin;**

**PORTA=0x00; DDRA=0xF0; // Port A initialization PORTC=0xFF; DDRC=0xFF; // Port C initialization**

**// ADC initialization**

**// ADC Clock frequency: 125,000 kHz**

**// ADC Voltage Reference: AVCC pin**

**// ADC High Speed Mode: Off**

**// ADC Auto Trigger Source: None ADMUX = ADC\_VREF\_TYPE;**

**ADCSRA = 0x86; SFIOR& = 0xEF;**

# ТАЙМЕРЫ-СЧЕТЧИКИ В МИКРОКОНТРОЛЛЕРАХ AVR

Микроконтроллеры семейства Mega, в зависимости от модели, имеют в своем составе от двух до четырех таймеров-счетчиков общего назначения – Т0, Т1, Т2 и Т3. Таймер – независимый аппаратный узел, который может использоваться для отсчета и измерения временных ин- тервалов или как счетчик внешних событий, для генерации сигналов с широтно-импульсной модуляцией (ШИМ).

Здесь будут рассмотрены основные функции таймеров на примере таймера-счетчика Т0.

# Принцип работы таймера-счетчика

Структурная схема блока таймера-счетчика приведена на рис. 47. Основу таймера-счетчика составляет реверсивный двоичный счетчик **TCNT**, на вход которого поступают тактовые сигналы. Режим работы счетчика задается в регистре **TCCR**. По достижении счетчиком некото- рого значения (нулевого 0x00, предельного 0xFF или заданного в реги- стре **OCR**) вырабатывается сигнал на выходе **OC** (и сигнал прерыва- ния). На выходе **TOV** возникает сигнал переполнения счетчика (и сиг- нал прерывания). В сам счетчик **TCNT** может быть загружено начальное значение с шины данных МК.

Счетчик может перезапускаться периодически, тогда на выходе **OC**

получается сигнал с заданным периодом или заданной длительности.

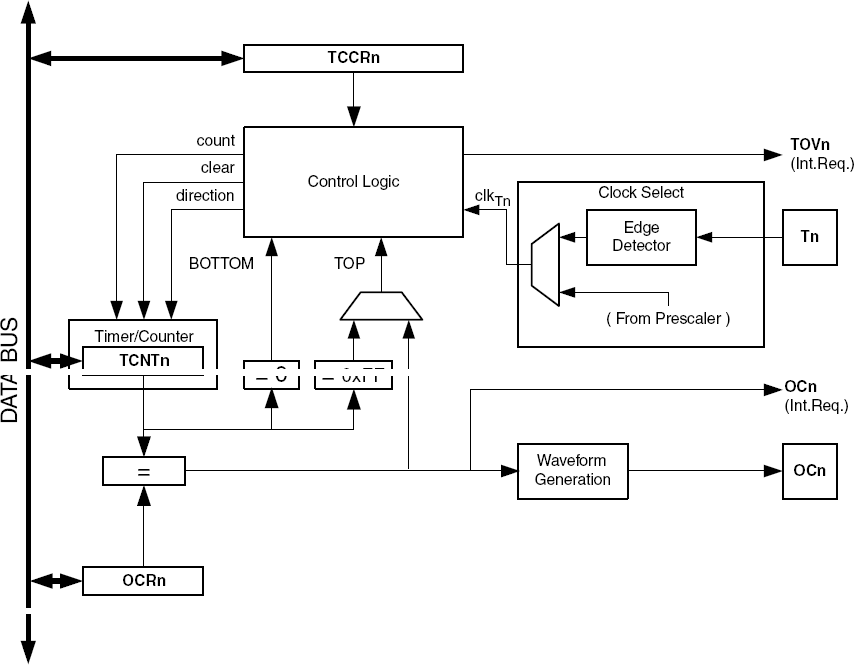
Если тактовые сигналы поступают на вход **T**, то в **TCNT** может быть подсчитано их количество.

Таким образом, таймер-счетчик, в зависимости от режима работы, может применяться как для формирования временных интервалов заданной длительности, так и для подсчета длительности или числа входных сигналов.

Возможность изменять длительность импульсов на выходе **OC**

позволяет осуществить широтно-импульсную модуляцию (ШИМ).

Рис. 47. Структурная схема 8-разрядного таймера-счетчика Т0

Источником тактовых импульсов могут служить либо сигналы с выходов предделителя, либо внешние сигналы, поступающие на вывод T0 микроконтроллера. Предделители последовательно делят тактовую частоту МК с коэффициентами 1, 8, 64, 256 или 1024.

# Режимы работы таймеров-счетчиков

Режим работы счетчика Т0 задается в регистре TCCR0. Формат регистра TCCR0 микроконтроллера ATmega8535 показан на рис. 48.

7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **FOC0** | **WGM00** | **COM01** | **COM00** | **WGM01** | **CS02** | **CS01** | **CS00** |
| W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Рис. 48. Формат регистра TCCR0 микроконтроллера ATmega8535

Биты CS02:CS00 определяют источник и частоту тактовых импуль- сов таймера-счетчика (см. табл. 14).

# РЕЖИМЫ NORMAL И CTC

*Таблица 14*

Выбор источника тактового сигнала для таймера-счетчика T0

|  |  |  |  |
| --- | --- | --- | --- |
| CS02 | CS01 | CS00 | Источник тактового сигнала |
| 0 | 0 | 0 | Таймер остановлен |
| 0 | 0 | 1 | clkI/O |
| 0 | 1 | 0 | clkI/O /8 (с предделителя) |
| 0 | 1 | 1 | clkI/O /64 (с предделителя) |
| 1 | 0 | 0 | clkI/O /256 (с предделителя) |
| 1 | 0 | 1 | clkI/O /1024 (с предделителя) |
| 1 | 1 | 0 | Внешний источник на входе T0. Тактирование по спаду |
| 1 | 1 | 1 | Внешний источник на входе T0. Тактирование по фронту |

Биты WGM01:WGM00 задают режим работы таймера-счетчика Т0 (см. табл. 15). Биты COM01:COM00 определяют вид сигнала на вы- ходе OC0, в зависимости от режима работы.

# Режим Normal

Это наиболее простой режим работы таймеров/счетчиков. Счет- ный регистр функционирует как обычный суммирующий счетчик. По каждому импульсу тактового сигнала clkT0 осуществляется инкре- мент счетного регистра. При переходе через значение 0xFF возникает переполнение, и счет продолжается со значения 0х00. В том же такте сигнала clkT0, в котором обнуляется регистр TCNT0, устанавливается в 1 флаг переполнения ТОV0.

При равенстве счетного регистра и регистра сравнения устанав- ливается флаг прерывания OCF0 и, если разряд OCIE0 регистра TIMSK установлен в 1, генерируется прерывание. Наряду с установкой флага при равенстве счетного регистра и регистра сравнения может изменяться состояние вывода ОС0 микроконтроллера. Каким образом оно будет изменяться, определяется разрядами СОМ01:СОМ00 регистра TCCR0 в соответствии с табл. 16.

*Таблица 16*

Управление выводом OC0 в режиме Normal

*Таблица 15*

|  |  |  |
| --- | --- | --- |
| СОМ01 | СОМ00 | Описание |
| 0 | 0 | Таймер/счетчик отключен от вывода ОС |
| 0 | 1 | Состояние вывода меняется на противоположное |
| 1 | 0 | Вывод сбрасывается в 0 |
| I | 1 | Вывод устанавливается в 1 |

Режимы работы таймера-счетчика Т0

При необходимости состояние вывода ОС0 может быть изменено принудительно, записью лог. 1 в разряд FOC0 регистра TCCR0. Преры- вание при этом не генерируется.

|  |  |  |
| --- | --- | --- |
| WGM0l | WGM00 | Режим работы таймера-счетчика |
| 0 | 0 | Normal |
| 0 | 1 | Phase correct PWM (ШИМ с точной фазой) |
| 1 | 0 | СТС (Clear Timer on Compare – Сброс при совпадении) |
| 1 | 1 | Fast PWM (Fast Pulse Width Modulation – Быстродей- ствующий ШИМ) |

# Режим СТС (Сброс при совпадении)

В этом режиме счетный регистр тоже функционирует как обыч- ный суммирующий счетчик, инкремент которого осуществляется по каждому импульсу тактового сигнала clkT0. Однако максимально воз- можное значение счетного регистра и, следовательно, разрешающая способность определяется регистром сравнения OCR0. После достиже- ния значения, записанного в регистре сравнения, счет продолжается

со значения 0x00. В том же такте сигнала clkT0, в котором обнуляется счетный регистр, устанавливается флаг прерывания TOV0 регистра TIFR. Временные диаграммы для этого режима работы тайме- ра/счетчика приведены на рис. 49.

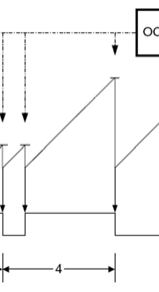
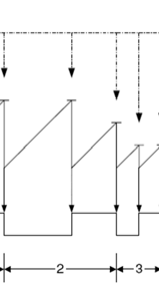
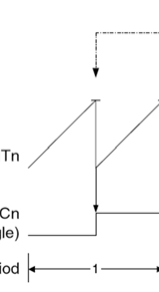


Рис. 49. Временные диаграммы для режима СТС

При достижении счетчиком максимального значения устанавли- вается флаг OCF0 регистра TIFR. Одновременно с установкой флага может изменяться состояние вывода ОС0 микроконтроллера. Поведение вывода определяется разрядами СОМ01:СОМ00, также как в режиме Normal (см. табл. 15).

Для генерации сигнала заданной частоты необходимо записать в разряды СОМ01:СОМ00 значение 01 (переключение состояния вывода). Максимальная частота, которую можно будет получить на выходе OC0, равна *f*clk\_OC0 = *f*clk\_I/O/2. Частота генерируемого сигнала будет опреде- ляться выражением.