АЦП МК ATMega16

Основные регистры
Понятие преобразования
Понятие прерывания
Реализация прерываний для АЦП на МК ATMega16

Что такое АЦП?

Иллюстрация работы 3-х разрядного АЦП, f=1 кГц

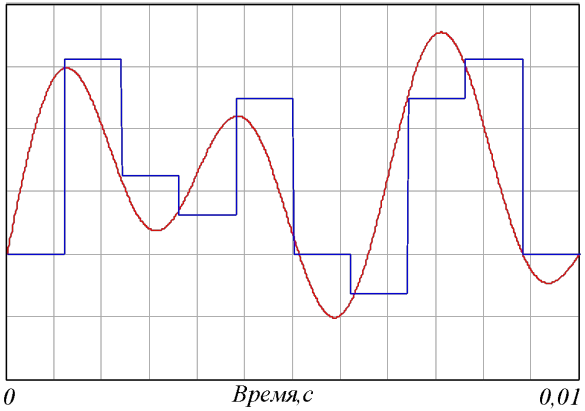


Иллюстрация работы 3-х разрядного АЦП, f=5 кГц

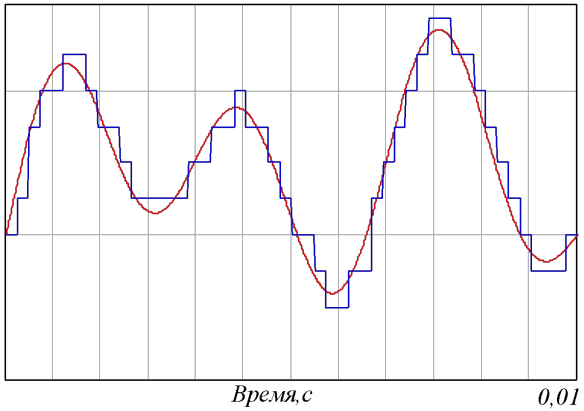


Иллюстрация работы 3-х разрядного АЦП, f=8кГц

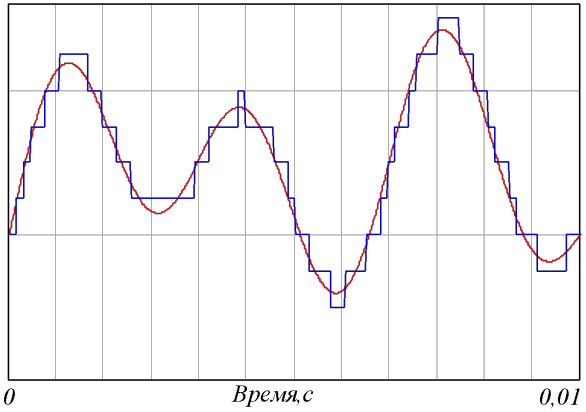
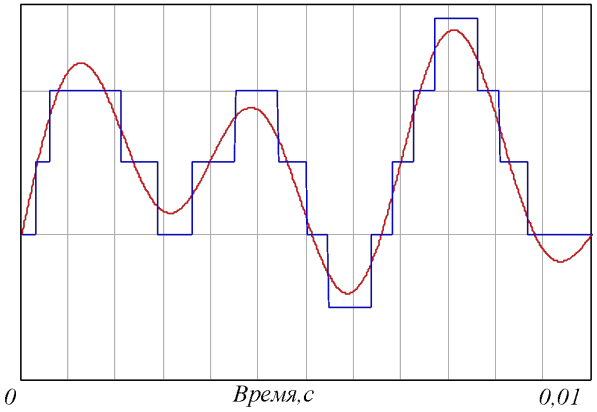


Иллюстрация работы 2-х разрядного АЦП, f=8кГц



Основные характеристики АЦП

- разрешающая способность;
- абсолютная точность;
- предельная частота дискретизации;
- диапазон входных напряжений

На вход АЦП подается непрерывный аналоговый сигнал, а на выходе получается последовательность цифровых значений.

Разрешающая способность

Разрешающая способность (разрешение) – это способность АЦП различать два значения входного сигнала. Определяется как величина обратная максимальному числу кодовых комбинаций на выходе АЦП.

У AVRa АЦП 10-ти разрядный. Максимальное число кодовых комбинаций равно 210 = 1024. Разрешающая способность равна 1/1024 от всей шкалы допустимых входных напряжений.

Для работы АЦП необходим источник опорного напряжения (ИОН). Микроконтроллеры AVR позволяют в качестве ИОНа использовать напряжение питания, внутренний опорный источник на 2,56 В и напряжение на выводе AREF (внешний ИОН).

Абсолютная точность

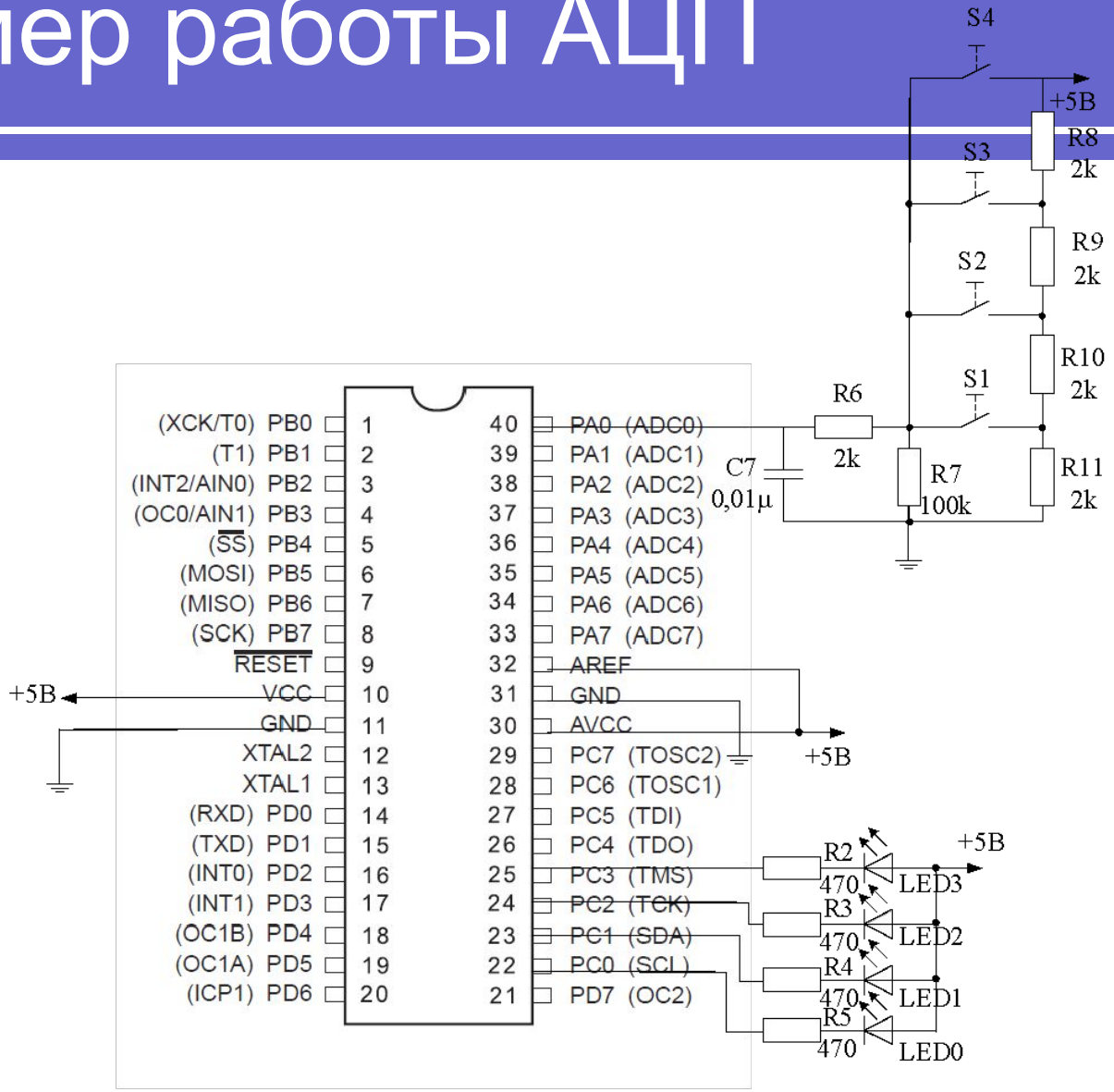
Абсолютная точность – отклонение реального преобразования от идеального. Это составной результат нескольких погрешностей АЦП. Выражается в количестве младших значащих разрядов (LSB - least significant bit) АЦП. Для AVR абсолютная погрешность АЦП = ±2LSB.

Предельная частота дискретизации

Предельная частота дискретизации определяет быстродействие АЦП и измеряется в герцах или количестве выборок в секунду (SPS – samples per second). Для микроконтроллеров AVR эта величина равна 15 kSPS.

Диапазон входных напряжений – это минимальное и максимальное значение напряжения, которое можно подавать на вход АЦП. Для микроконтроллера AVR он равен 0 – Vcc (напряжение питания)

Пример работы АЦП



Алгоритм работы АЦП МК ATMega16

- Основная программа
  Инициализация портов
  Инициализация АЦП
  Разрешение прерываний
  Бесконечный цикл
  {
  Если кнопка нажата, зажечь нужный светодиод
  Если нет, погасить все светодиоды
  }

  Обработчик прерывания АЦП
  Считать напряжение на входе АЦП
  Определить в какой диапазон оно попадает
  Записать номер кнопки в буфер

Инициализация АЦП

Чтобы запустить модуль АЦП, его нужно предварительно настроить. За это отвечают три регистра:
Регистр управления мультиплексором - **ADMUX**
Регистр управления и состояния - **ADCSRA**
Регистр специальных функций - **SFIOR**

Регистр ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

За выбор ИОНа отвечают биты REFS1, REFS0.

Результат преобразования хранится в двух регистрах (ADCH, ADCL). Бит ADLAR задает направление выравнивания результата преобразования. 0 – выравнивание вправо (в ADCH заняты 2 младших бита, ADCL занят весь), 1 – выравнивание влево (ADCH занят весь, в ADCL только 2 старших бита).

Регистр ADMUX

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Регистр ADMUX

Номер выбранного в данный момент канала мультиплексора задается битами MUX4, MUX3, MUX2, MUX1, MUX0

ADMUX =(0<<REFS1)|(1<<REFS0)|(1<<ADLAR)|(0<<MUX3)|(0<<MUX2)|(0<<MUX1)|(0<<MUX0)

Регистр ADCSRA

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Чтобы АЦП заработал его надо включить, установить бит ADEN – 1

Запуск преобразования осуществляется установкой бита ADSC -1.

АЦП может работать в двух режимах: одиночное преобразование и непрерывное. Если преобразование одиночное, бит ADATE – 0.

Когда АЦП закончил преобразование, он подает запрос на прерывание. Разрешить прерывание: ADIE установить 1.

Тактовый сигнал АЦП формируется из тактового сигнала микроконтроллера путем деления на фиксированные коэффициенты.

Регистр ADCSRA

Для установки коэффициентов предделителя тактового сигнала предназначены биты ADSP2, ADSP1, ADSP0.

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

ADIF - это флаг прерывания. Он устанавливается аппаратно, когда преобразование завершено.

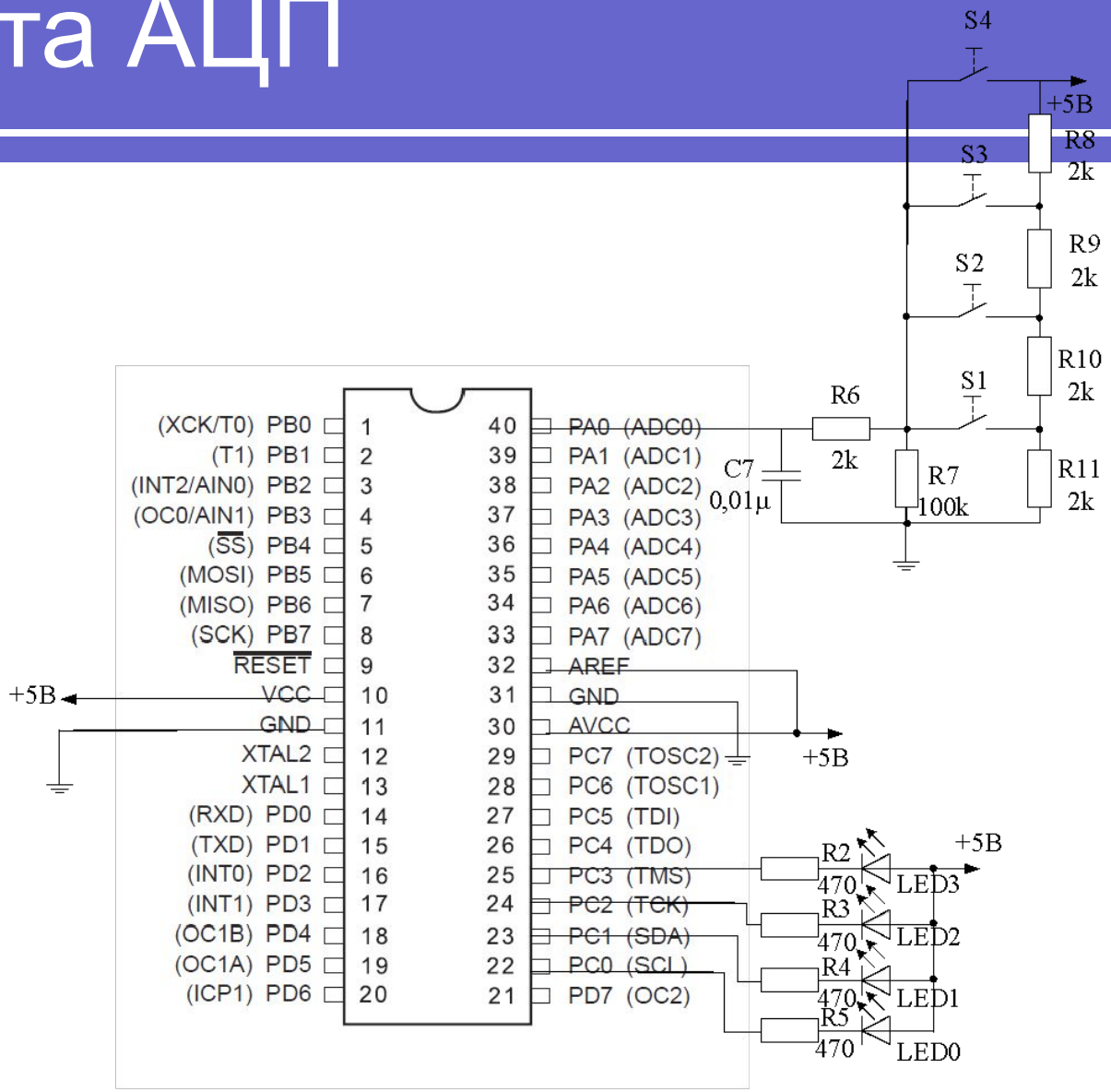
ADCSRA =
(1<<ADEN)|(0<<ADSC)|(0<<ADATE)|(1<<ADIE)|
(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)

Регистр SFIOR

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

Работа АЦП


S4
+5B
R8
2k
S3
R9
2k
S2
R10
2k
S1
R6
2k
R7
100k
R11
2k
C7
0,01µ
(XCK/T0) PB0 1
(T1) PB1 2
(INT2/AIN0) PB2 3
(OC0/AIN1) PB3 4
(SS) PB4 5
(MOSI) PB5 6
(MISO) PB6 7
(SCK) PB7 8
RESET 9
VCC 10
GND 11
XTAL2 12
XTAL1 13
(RXD) PD0 14
(TXD) PD1 15
(INT0) PD2 16
(INT1) PD3 17
(OC1B) PD4 18
(OC1A) PD5 19
(ICP1) PD6 20
40 PA0 (ADC0)
39 PA1 (ADC1)
38 PA2 (ADC2)
37 PA3 (ADC3)
36 PA4 (ADC4)
35 PA5 (ADC5)
34 PA6 (ADC6)
33 PA7 (ADC7)
32 AREF
31 GND
30 AVCC
29 PC7 (TOSC2)
28 PC6 (TOSC1)
27 PC5 (TDI)
26 PC4 (TDO)
25 PC3 (TMS)
24 PC2 (TCK)
23 PC1 (SDA)
22 PC0 (SCL)
21 PD7 (OC2)
+5B
+5B
+5B
R2
470
LED3
R3
470
LED2
R4
470
LED1
R5
470
LED0

Код программы

- **Код программы**
- //программирование микроконтроллеров AVR на Си - осваиваем АЦП

```
#include<ioavr.h>
#include<intrinsics.h>

#defineStartConvAdc() ADCSRA |= (1<<ADSC)

#define KEY_NULL  0
#define KEY_S1    1
#define KEY_S2    2
#define KEY_S3    3
#define KEY_S4    4

//кнопочный буфер
volatileunsignedcharKeyBuf = 0;

intmain(void)
{
  unsignedchartmp;

  //настраиваем порты
  DDRC = 0xff;
  PORTC = 0xff;
```

Продолжение

```
//инициализируем АЦП
//ион - напряжение питания, выравнивание влево, нулевой канал
ADMUX = (0<<REFS1)|(1<<REFS0)|(1<<ADLAR)|(0<<MUX3)|(0<<MUX2)|(0<<MUX1)|(0<<MUX0);
//вкл. ацп, режим одиночного преобр., разрешение прер., частота преобр. = FCPU/128
ADCSRA = (1<<ADEN)|(1<<ADSC)|(0<<ADATE)|(1<<ADIE)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);

//разрешаем прерывания и запускаем преобразование
__enable_interrupt();
StartConvAdc();
```

```
//основной цикл программы - опрос кнопочного буфера
while(1)
{
  tmp = KeyBuf;
  if(tmp)
  {
    tmp--;
    PORTC = ~(1<<tmp);
  }
  else
    PORTC = 0xff;
}
return 0;
}
```

Продолжение

```
#pragmavector=ADC_vect
__interruptvoidadc_my(void)
{
  //считываем старший регистр АЦП
  unsignedcharAdcBuf = ADCH;

  //опеределяем в какой диапазон попадает его значение
  if(AdcBuf> 240)
    KeyBuf = KEY_S4;
  elseif (AdcBuf> 180)
    KeyBuf = KEY_S3;
  elseif(AdcBuf> 120)
    KeyBuf = KEY_S2;
  elseif(AdcBuf> 35)
    KeyBuf = KEY_S1;
  else
    KeyBuf = KEY_NULL;

  //запускаем преобразование и выходим
  StartConvAdc();
}
```