СПбГЭТУ «ЛЭТИ»
ПЕРВЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ

Михаил Копычев

МИКРОПРОЦЕССОРНАЯ ТЕХНИКА В МЕХАТРОНИКЕ И РОБОТОТЕХНИКЕ

Встроенный аналого-цифровой преобразователь микроконтроллера

ОПРЕДЕЛЕНИЕ АЦП

АЦП — устройство, преобразующее входной аналоговый сигнал в дискретный код (цифровой сигнал).

АЦП имеет ряд характеристик:

- Разрешение – минимальное изменение величины аналогового сигнала, которое может быть преобразовано данным АЦП. В случае единичного измерения без учёта шумов разрешение напрямую определяется разрядностью АЦП.
- Разрядность АЦП характеризует количество дискретных значений, которые преобразователь может выдать на выходе. В двоичных АЦП измеряется в битах.
- Частота дискретизации характеризует частоту выборки цифровых значений из непрерывного аналогового сигнала.

ATmega128A имеет встроенный 10-и разрядный АЦП последовательного приближения.

РЕГИСТР ВЫБОРА КАНАЛА АЦП – ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	**ADMUX**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Init.	0	0	0	0	0	0	0	0	

Разряды REFS1:0 определяют используемое опорное напряжение АЦП.

REFS1	REFS0	Используемое опорное напряжение
0	0	AREF
0	1	AVCC
1	0	Зарезервировано
1	1	Внутренний ИОН на 2.56 В

ADMUX: ВЫРАВНИВАНИЕ РЕЗУЛЬТАТА АЦП

Разряд ADLAR (ADC Left Adjust Result) определяет выравнивание результата преобразования в регистрах ADCH и ADCL.

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
	-	-	-	-	-	-	ADC9	ADC8	**ADCH**
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	**ADCL**
Bit	7	6	5	4	3	2	1	0	
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Init.	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	**ADCH**
	ADC1	ADC0	-	-	-	-	-	-	**ADCL**
Bit	7	6	5	4	3	2	1	0	
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Init.	0	0	0	0	0	0	0	0	

ADMUX: ВЫБОР КАНАЛА АЦП

Разряды MUX4:0 выбирают канал АЦП. Их комбинации для однополярных каналов представлены в таблице. В случае однополярных каналов комбинация битов MUX соответствует двоичному коду номера канала.

MUX4	MUX3	MUX2	MUX1	MUX0	Описание
0	0	0	0	0	ADC0
0	0	0	0	1	ADC1
0	0	0	1	0	ADC2
0	0	0	1	1	ADC3
0	0	1	0	0	ADC4
0	0	1	0	1	ADC5
0	0	1	1	0	ADC6
0	0	1	1	1	ADC7

РЕГИСТР НАСТРОЙКИ АЦП – ADCSRA

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	**ADCSRA**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Init.	0	0	0	0	0	0	0	0	

Разряд ADEN. В случае записи в него логической единицы происходит включение модуля АЦП.

Разряд ADSC. В случае записи в него логической единицы начинается процесс преобразования аналогового сигнала в цифровой код.

Разряд ADFR. В случае записи в него логической единицы, АЦП переходит в режим автоматического перезапуска.

Разряд ADIF - это флаг прерывания АЦП, устанавливающийся после завершения преобразования АЦП и обновления регистров данных. Для сброса флага необходимо записать в него логическую единицу.

Разряд ADIE. Запись логической единицы в этот бит разрешает прерывания от АЦП.

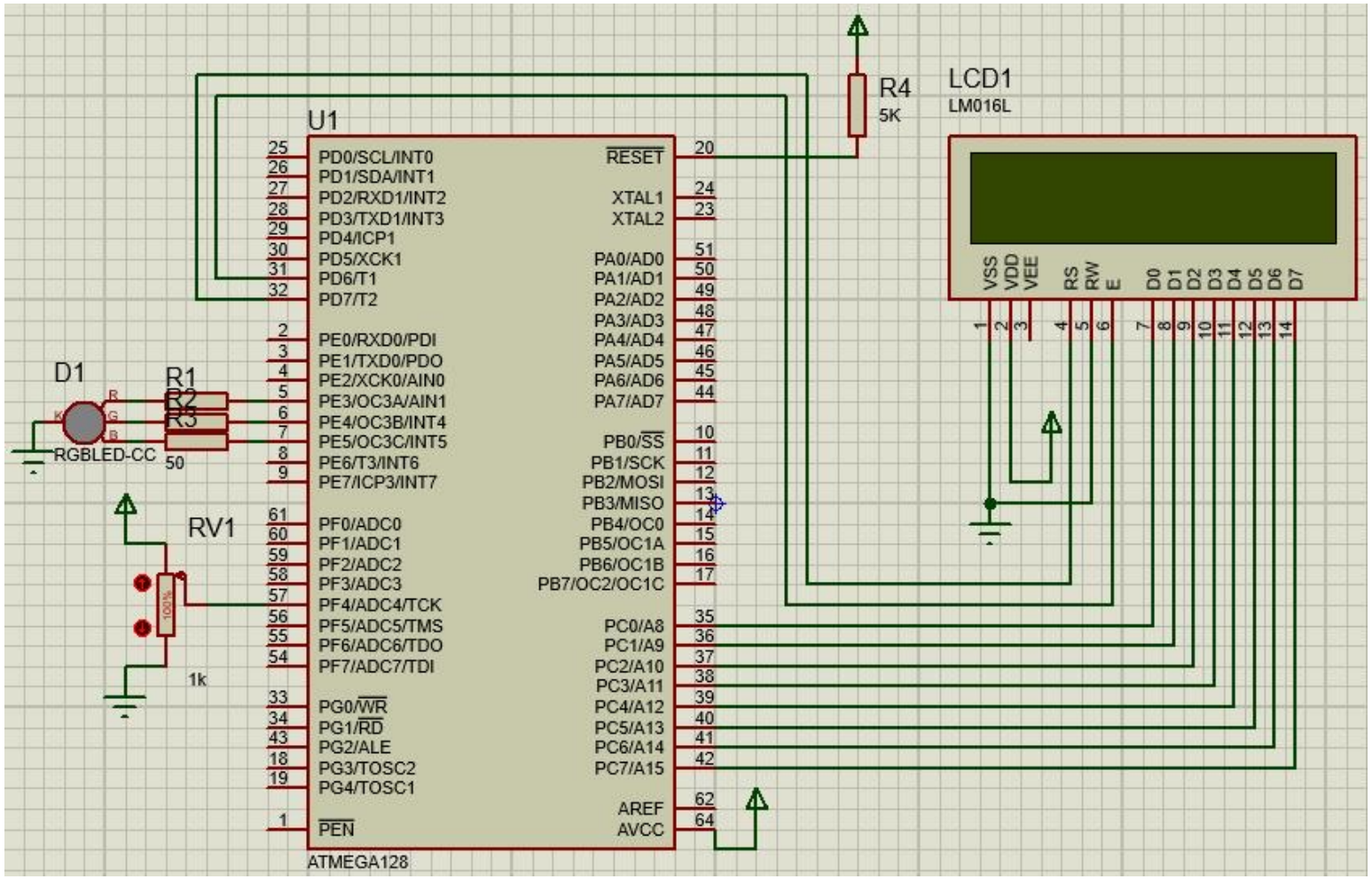
Разряды ADPS2:0 определяют частоту тактирования модуля АЦП, определяющую частоту дискретизации АЦП.

ADCSRA: ВЫБОР ЧАСТОТЫ ТАКТИРОВАНИЯ МОДУЛЯ АЦП

Соотношение между значениями разрядов ADPS2:0 и тактовой частотой АЦП представлено в таблице.

ADPS2	ADPS1	ADPS0	Описание
0	0	0	Без предделителя
0	0	1	С предделителем на 2
0	1	0	С предделителем на 4
0	1	1	С предделителем на 8
1	0	0	С предделителем на 16
1	0	1	С предделителем на 32
1	1	0	С предделителем на 64
1	1	1	С предделителем на 128

РАБОТА В СИМУЛЯТОРЕ (1/2)



Задание:

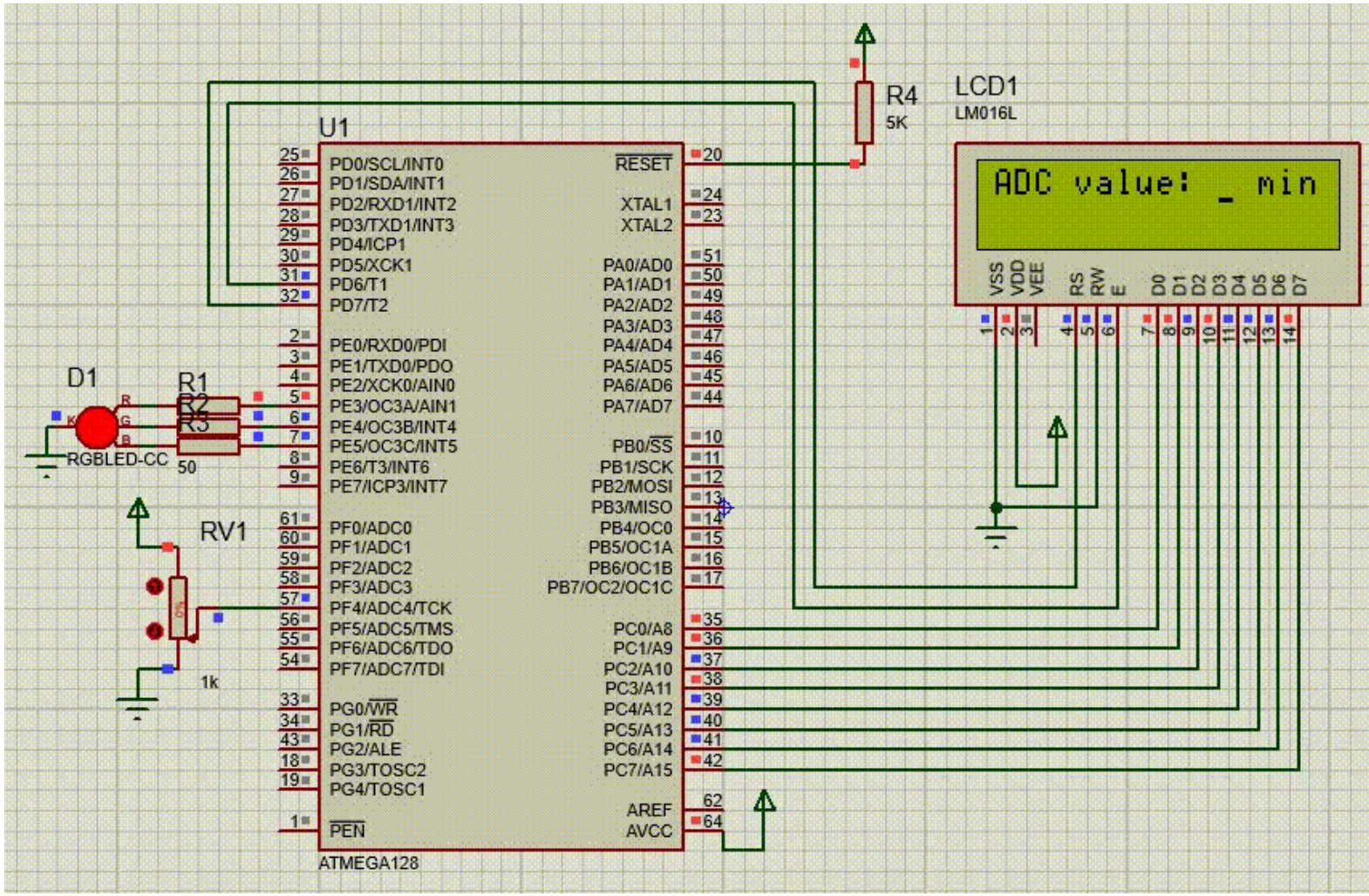
написать программу, которая бы выводила результат аналого-цифрового преобразования на ЖКИ.

Дополнительно программа может обеспечивать изменение цвета RGB светодиода согласно изменению значения аналогового сигнала.

РАБОТА В СИМУЛЯТОРЕ (2/2)

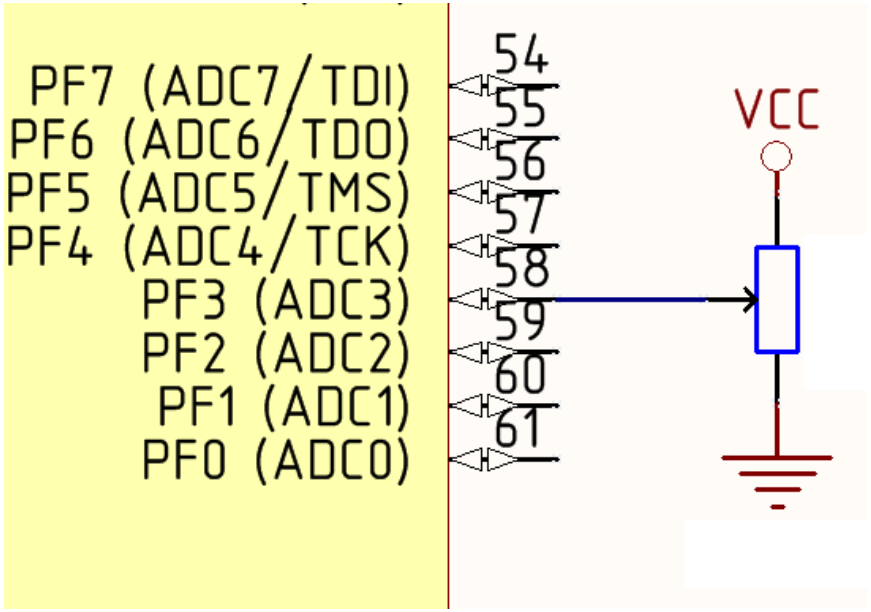
```
/* объявление заголовков прочих функций */
uint16_t adc_res=0;
ISR(ADC_vect)
{
    adc_res=ADCL|(ADCH<<8);
    ADCSRA|=(1<<ADSC);
}

int main(void)
{
    uint8_t words[] = {"ADC value:"}, i, j;
    uint16_t brightness;
    /* настройка регистра DDRE и таймера №3 */
    /* АЦП вкл., начать 1-ое преобразование,
    предделитель на 32 */
    ADCSRA =
(1<<ADEN)|(1<<ADSC)|(1<<ADIE)|(1<<ADPS2)|(1<<ADPS0);
    ADMUX=4|(1<<REFS0);
    LCD_init();
    showMe(words);
    sei();
    while (1)
    {
        brightness = adc_res;
        /* зажигаем диод чрез OCR3xH/L */
        LCD_cmd((1 << 7) | 11);
        /* выводим на экран значение brightness */
    }
}
```



СЧИТЫВАНИЕ ЗНАЧЕНИЯ АЦП

Задание: написать программу управления цветом и яркостью RGB светодиода (каналы OC3A, OC3B и OC3C таймера/счётчика 3) с помощью ползункового потенциометра, подключённого к 3-му каналу АЦП, используя 10-и битный результат преобразования.



ПРИМЕР КОДА НА ЯЗЫКЕ СИ

Задание: управление яркостью RGB светодиода с помощью ползункового потенциометра

```
uin16_t readAdc(uint8_t channel)
{
    uint8_t lowBitValue;
    ADMUX = (1 << REFS0) | channel;      //выбор канала и источника питания АЦП
    ADCSRA |= (1 << ADSC);               //начало преобразования аналогового сигнала
    while((ADCSRA & (1 << ADIF))==0);    //ожидание окончания преобразования
    ADCSRA |= (1 << ADIF);               //сброс флага прерывания по окончании преобразования
    lowBitValue = ADCL;                  //чтение младшей части результата
    return((ADCH<<8) | lowBitValue);     //чтение всего результата
}
int main(void)
{
    DDRE = 0b00111000;
    TCCR3A = (1<<COM3A1) | (1<<COM3B1) | (1<<COM3C1) | (1<<WGM30) | (1<<WGM31);
    TCCR3B = (1 << CS30);                //включение таймера
    ADCSRA = (1 << ADEN);                //включения АЦП
    while (1)
    {
        OCR3BH = readAdc(3) >> 8;
        OCR3BL = readAdc(3);
    }
}
```