AVR Урок 22. Изучаем АЦП. часть 1 |

Posted on

Урок 22

Часть 1

Сегодня мы начнем изучать очень интересную технологию, а для микроконтроллера – периферию – это **аналго-цифровой преобразователь** или как его называют **АЦП**. В английской аббревиатуре, гораздо чаще встречающейся в технической документации – **ADC** (**Analog-to-Digital Converter**). Это такая штука, которая преобразует величину электрического сигнала в цифровой код. Затем данный код мы уже используем для обработки или для отображения тем или иным образом данной электрической величины. Это очень распространённая периферия или технология. АЦП активно используется в звукозаписи, измерительной технике, видеозаписи и во многих других случаях. Поэтому нас обойти данную вещь стороной никак не получится, тем более АЦП поддерживается аппаратно в контроллерах **AVR**.

В контроллере Atmega8 АЦП имеет следующие характеристики

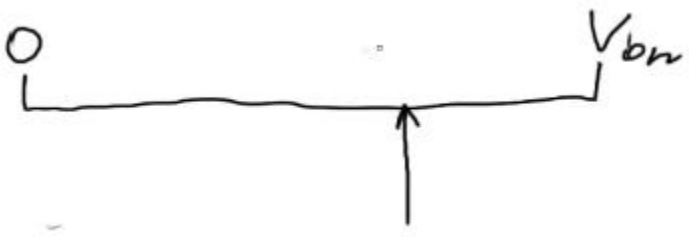
- Разрешение 10 бит,
- Время преобразования одного показания от 13 до 250 микросекунд в зависимости от битности измерения, а также от тактовой частоты генератора, тактирующего контроллер,
- Поддержка запуска по прерываниям,

Есть ещё масса различных характеристик, с которыми мы, возможно, познакомимся в дальнейшем.

Как вообще работает цифровое преобразование?

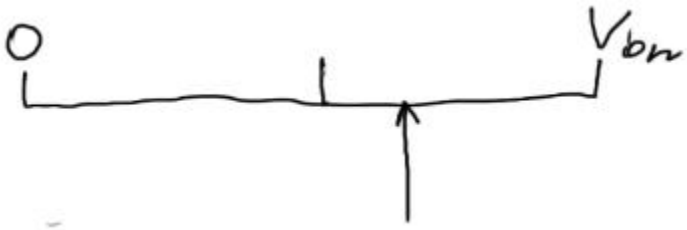
Берётся опорное напряжение и сравнивается с измеряемым. Соответственно, опорное напряжение всегда должно быть больше измеряемого. Если это не так, то нужно будет применять делители напряжения.

Изобразим схематично процесс измерения

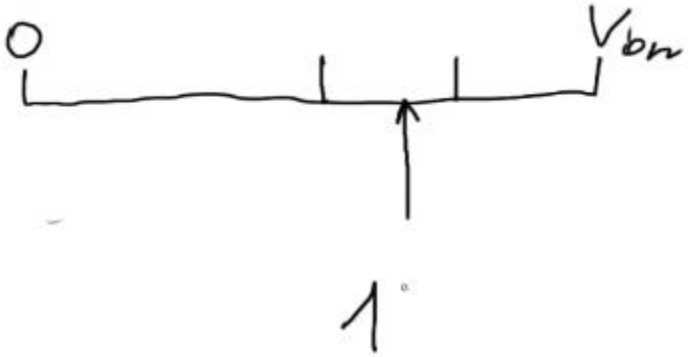


Отрезок – это диапазон измерений. Он находится между нулём и опорным напряжением. А стрелка – это измеряемое напряжение.

Данный отрезок делится пополам, и АЦП оценивает, в какой половине находится приложенное напряжение

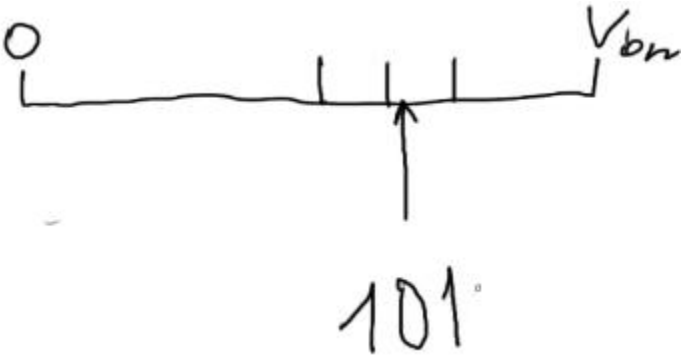


Если оно находится в стороне нуля, то в самый старший бит результата записывается 0, а если в стороне максимального напряжения, то единица. У нас будет единица. Затем та половина отрезка, на которой находится измеряемое напряжение делится ещё пополам, и АЦП опять измеряет, в какой половинке уже данного отрезка у нас находится измеряемое напряжение



Оценка идёт по тому же принципу – в какой стороне отрезок. В нашем случае будет 0. И этот ноль записывается в следующий более младший бит

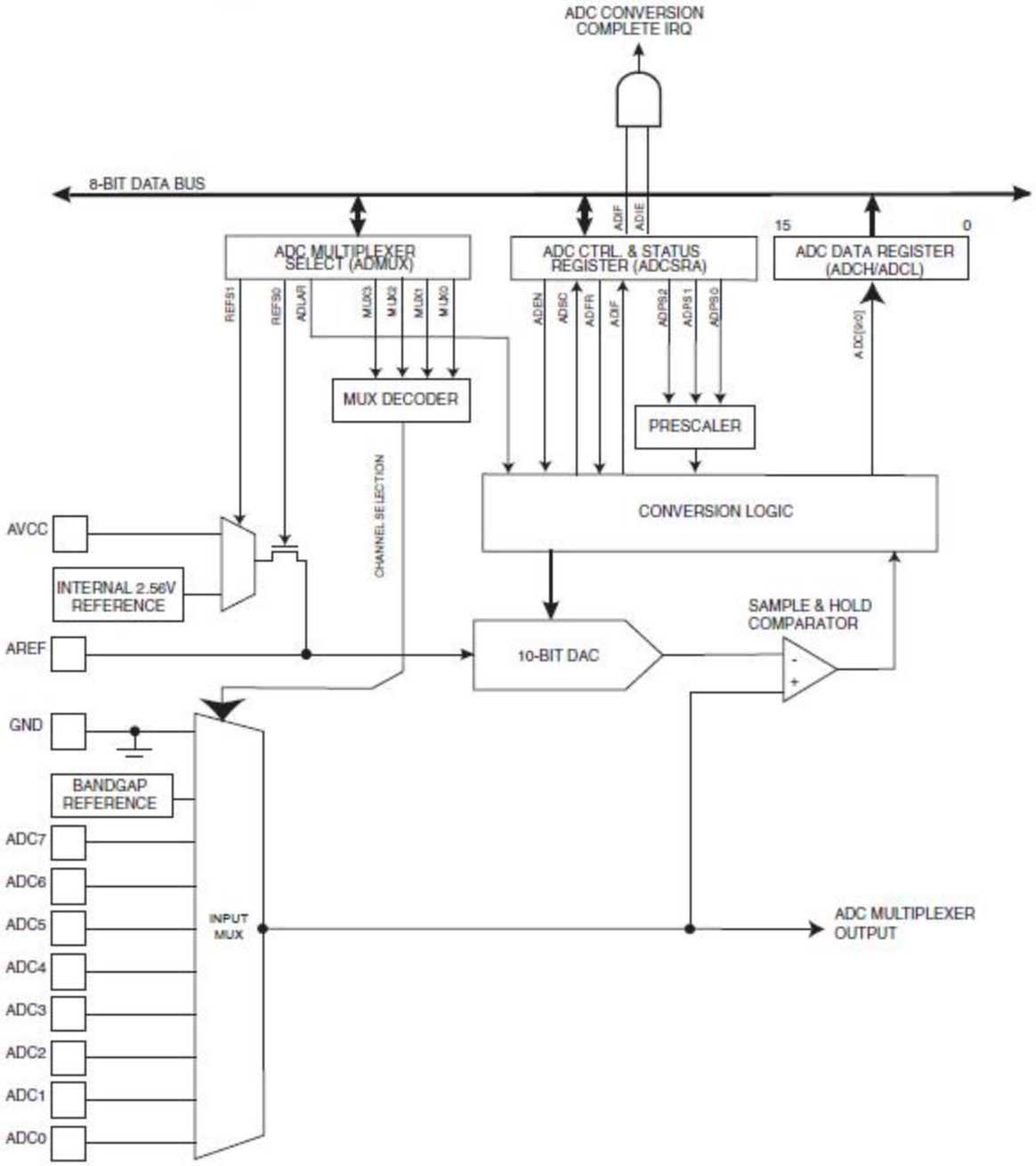
Затем та четвертинка опять делится пополам и АЦП опять оценивает,где находится отрезок



И АЦП так и продолжает такой процесс до тех пор, пока не кончатся ячейки для битов. То есть если мы используем 10-битный режим, то. соответственно, и будет 10 подобных измерений и заполнятся 10 бит величины. Чем больше бит, тем точнее результат, но уже потребуется на это больше времени и более серьёзный и точный АЦП. Имея данный результат, нам несложно будет посчитать величину измеренного напряжения. Мы знаем. что если у нас АЦП 10-битный, то данный результат у нас лежит в промежутке от 0 до 1024, получаемтся всего у нас 1023 отрезка. И мы затем наш результат делим на 1023 и умножаем на величину опорного напряжения.

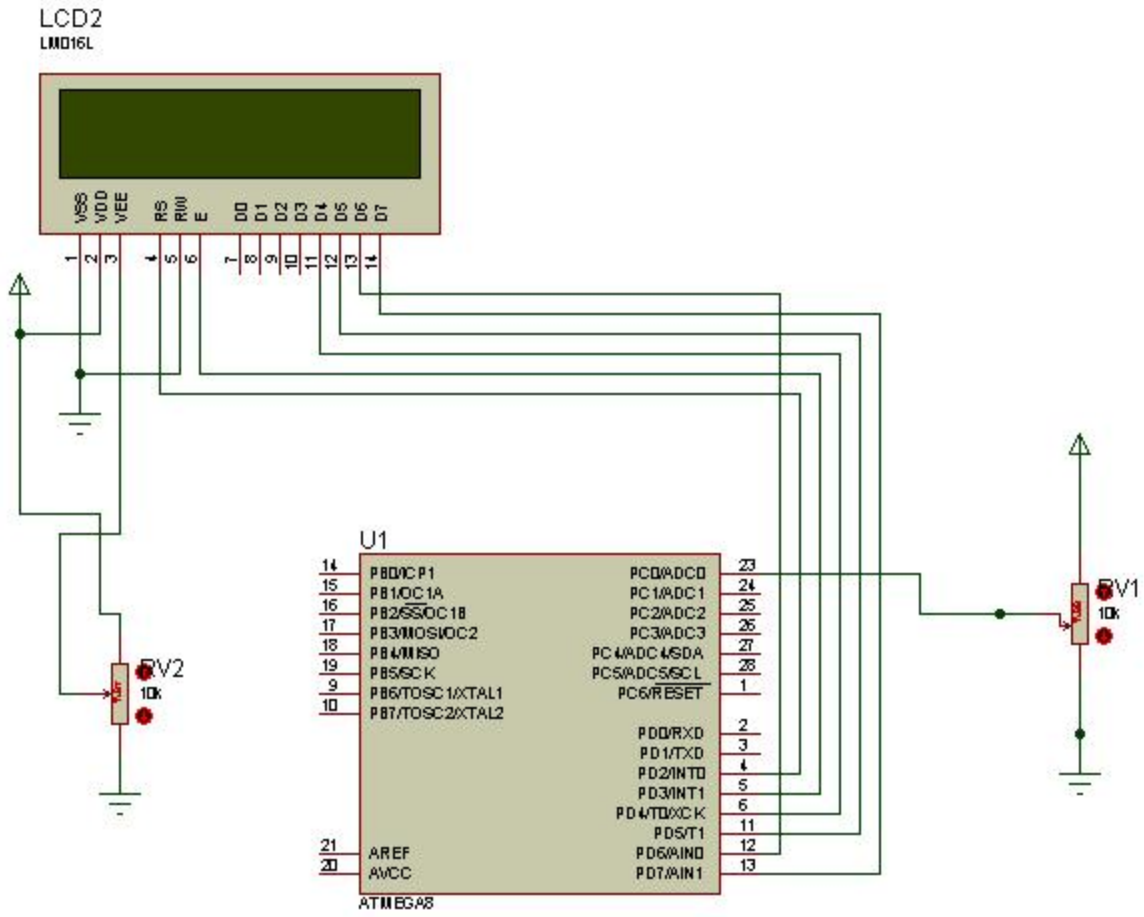
Посмотрим блок-схему АЦП в контроллере Atmega8

Figure 90. Analog to Digital Converter Block Schematic Operation



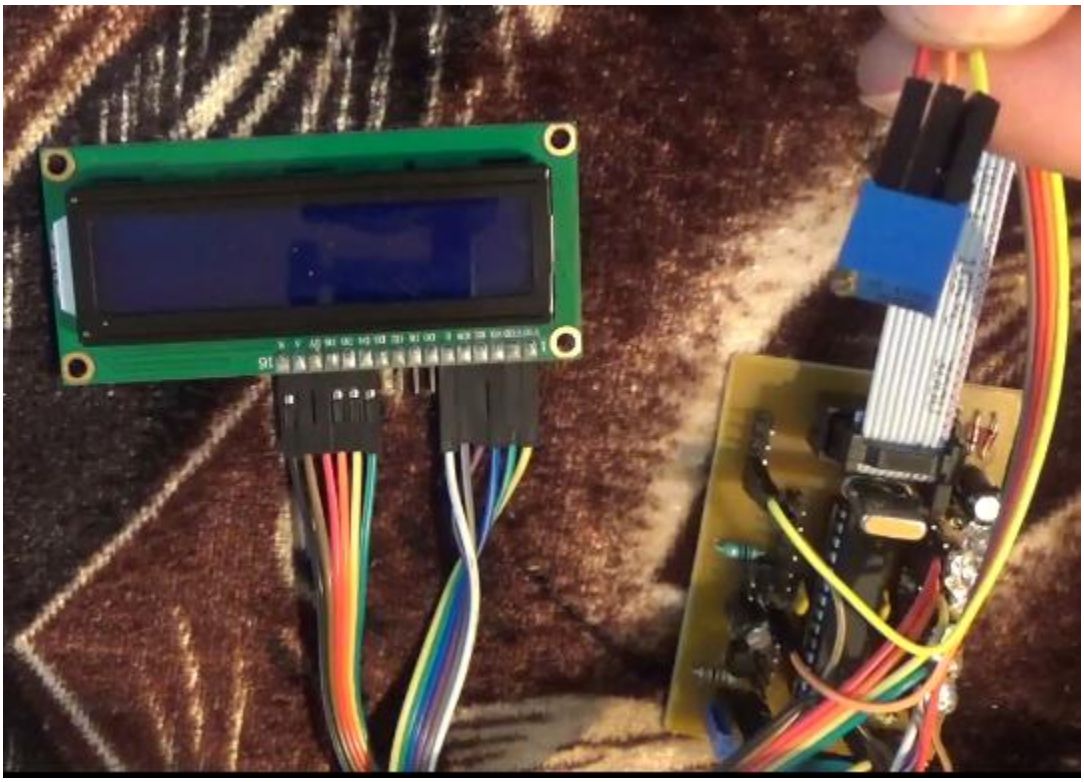
Мы видим, что у нас есть мультиплексор, имеющий 8 каналов, вход для опрного напряжения AREF. Также существует 8-разрядная шина данных, значения с которых записываются в определённый регистр. Есть регистр данных, регистр управления и состояния, а также регистр управления мультиплексором.

Мы будем использовать самый первый вход ADC0 и в качестве источника измеряемого напряжения будем использовать центральную ножку переменного резистора, подлключенного к контактам питания



Работать будем сначала в протеусе. Также мы видим, что у нас подключен дисплей самым обычным образом.

Также мы всё подключим и на живом контроллере



Существует несколько вариантов опорных напряжений, которые мы можем использовать.в нашем АЦП. Мы будем использовать внутреннее опорное напряжение на 2,56 вольт, оно проще, не нужно ничего подключать. Возможно, при таком варианте не очень сильная точность, но перед нами не стоит задача создать точный измерительный прибор. У нас есть задача – изучить возможность использования АЦП в контроллере AVR.

А вот и таблица вариантов опорных напряжений для АЦП

Table 74. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

Перечислим данные варианты сверху вних по таблице. 1 вариант – это внутреннее опорное напряжение, равное напряжению питания, 2 вариант – опорное напряжение подаётся на вход AREF извне, 3 вариант – внутреннее 2,56 вольт с использованием внешнего конденсатора, которы у нас уже припаян к отладочной плате к определённым ножкам контроллера.

Также в АЦП есть делитель частоты на величину от 2 до 128. Делитель этот для того, чтобы мы добивались частоты работы АЦП не больше 200 кГц, иначе точность измерений будет очень малой и мы просто растеряем самые младшие биты. Если у нас возникнет требование имено к скорости измерений и нам уже точность будет на так важна, то мы сможем использовать и более высокие частоты измерений.

Теперь немного поближе познакомимся с регистрами АЦП.

Регистр **ADCSRA** – управляющий и статусный регистр

ADC Control and Status Register A – ADCSRA

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	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	

Теперь побитно.

ADEN – данный бит включает АЦП.

ADSC – при установке в 1 заставляет АЦП начинать преобразование.

ADFR – используется в режиме с использованием прерываний. При установке в 1 включает круговой режим, при котором измерения автоматически следуют одно за другим.

ADIF – бит, также используемый только в режиме прерываний. Это флаг прерываний, который устанавливается в определённых условиях.

ADIE – бит, включающий режим прерываний.

ADPS2-ADPS0 – биты, от комбинации которых зависит величина делителя

Table 76. ADC Prescaler Selections

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

Регистр **ADMUX** – это регистр для управления каналами мультиплексора АЦП

ADC Multiplexer Selection Register – ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
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	

Но, помимо непосредственно битов управления каналами у данного регистра есть ещё некоторые управляющие биты

REFS1-REFS0 – биты, включающие определённый режим использования опорного напряжения. Таблица была размещена на данной странице выше.

ADLAR – это бит организации расположения измеренных 10 битов в двух байтах регистровой пары данных. Поближе мы с этим расположением познакомимся чуть позже.

MUX3-MUX0 – биты, включающие определённый канал мультиплексора

Table 75. Input Channel Selections

MUX3..0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	
1001	
1010	
1011	
1100	
1101	
1110	1.30V (V_{BG})
1111	0V (GND)

Отсюда видно, что мы можем пользоваться несколькими каналами сразу только по очереди, попеременно включая различные комбинации данных битов. Также внизу есть две комбинации для калибровки нашего АЦП.

Ну и, наконец, регистровая пара **ADCH** и **ADCL**, состоящая из старшего и младшего байта в которую и заносится измеряемый результат. А как именно он туда укладывается, этот результат, зависит от состояния бита ADLAR, рассмотренного выше в регистре **ADMUX**

ADLAR = 0

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	

ADLAR = 1

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	

То есть, если бит ADLAR не выставлен, то младшие 8 бит результата находятся в младшем байте регистровой пары, а 2 старших бита – в младших битах старшего байта. Если же бит ADLAR у нас выставлен, то 8 самых старших бит результата находятся в сатршем байте, а 2 младших в 2 старших битах младшего байта регистровой пары. Второй вариант нам интересен при исользовании 8-битного режима. В данном случае мы читаем только старший байт.

Проект был создан полностью из проекта **урока по изучению 4-битного режима подключения LCD**

Test09 и был назван **MyADCLCD**.

Также для выноса кода для реализации периферии АЦП были созданы стандартным образом два файла **adc.h** и **adc.c**. Соответственно файл adc.h был подключен и в файле main.h и в adc.c.

В файл MyADCLCD.c код был также полностью скопирован из главного файла проекта Test09, всё лишнее было удалено. Код в данном файле после данной операции принял следующий вид

```
#include "main.h"

//---------------------------------

void port_ini(void)

{

 PORTD=0x00;

 DDRD=0xFF;

}

//---------------------------------

int main(void)

{

 port_ini(); //Инициализируем порты

 LCD_ini(); //Инициализируем дисплей

 clearlcd(); //Очистим дисплей

 while(1)

 {

  setpos(0,0);

  _delay_ms(500);

 }

}
```

А уже весь полезный код по программированию аналого-цифрового преобразователя мы начнём писать в **следующей части** занятия.

Предыдущий урок Программирование МК AVR Следующая часть

Программатор и дисплей можно приобрести здесь:

Программатор (продавец надёжный) **USBASP USBISP 2.0**

Дисплей LCD 16×2

Смотреть ВИДЕОУРОК (нажмите на картинку)

Post Views: 19 487