Разработка схемы и программы в части сопряжения микропроцессора с элементами ввода информационного сигнала (АЦП)

1. Сведения об АЦП

АЦП – устройство, преобразующее входной аналоговый сигнал в дискретный код. В цифровой фильтрации сигналов АЦП необходимы. Согласно ТЗ нужно использовать ЦАП AD1674

AD1674 – многоцелевой 12 битный аналого-цифровой преобразователь, содержащий встроенные устройство выборки-хранения (УВХ), 10 В источник опорного напряжения (ИОН), буфер тактовых импульсов и выходной буфер с тремя состояниями для связи с микроконтроллером.

Устройство имеет следующие характеристики:

- Монолитный 12 битный 10 мс АЦП со схемой выборки
- Встроенное устройство выборки-хранения
- Цоколевка, соответствующая промышленному стандарту
- 8 и 16 битный микропроцессорный интерфейс
- Определенные и проверенные статические и динамические характеристики
- Униполярный и биполярный входы
- Диапазоны входного сигнала ±5 В, ±10 В, 0 В - 10 В и 0 В - 20 В
- Коммерческий, индустриальный и военный температурные диапазоны
- MIL-STD-883 и SMD корпусные исполнения[1]

Функциональная схема показана на рисунке 1.

Расположение выводов показано на рисунке 2.

Цифровые выводы устройства:

- A_0 – При преобразовании отвечает за его битность. При чтении отвечает за зануление младших битов
- CE – Включение устройства
- CS – Выбор устройства. Активное состояние – 0

- DB* – Параллельные выводы резульата преобразования. Один вывод – один бит
- R/C – Бит выбора режима: чтение данных или преобразование поступающего на вход AD1674 сигнала
- STS – Статус преобразования
- 12/8 – Определяет формат передачи результата преобразования. Либо как одно 12-битное слово, либо как два 8-битных.

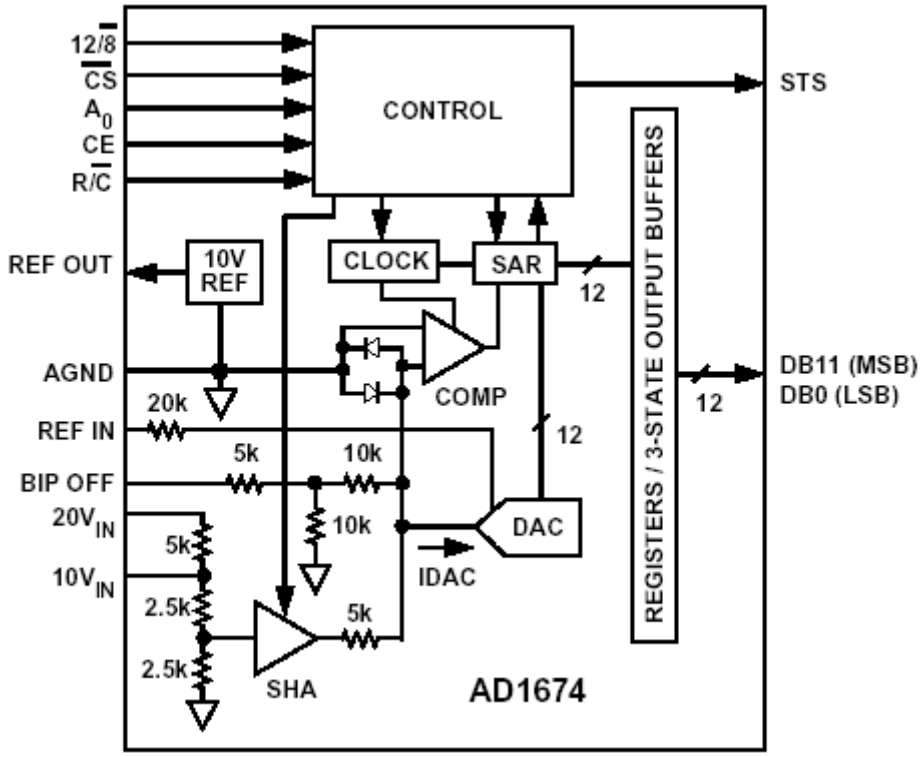

Рисунок 1 – Функциональная схема AD1674

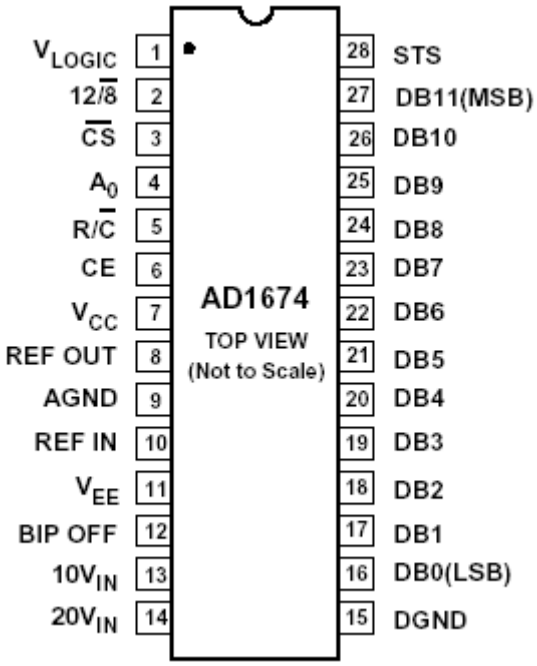

Рисунок 2 – Расположение выводов AD1674

Схема подключения АЦП в униполярном режиме показана на рисунке 3. Она понадобиться нам при моделировании схемы.

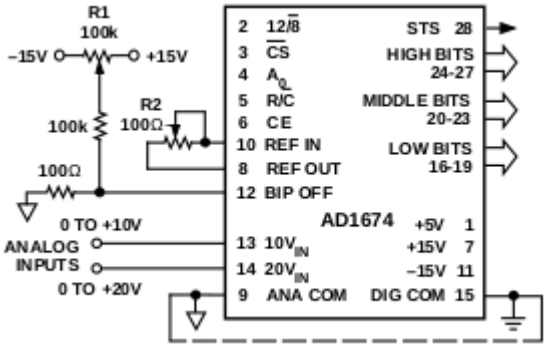

Рисунок 3 – Схема подключения АЦП в униполярном режиме

Таблица истинности для входов АЦП показана в таблице 1. Она нужна, чтобы лучше ориентироваться в режимах работы АЦП

Таблица 1 – Таблица истинности входов АЦП

CE	CS	R/C	12/8	A_0	Операция
0	x	x	x	x	Ничего
x	1	x	x	x	Ничего
1	0	0	x	0	Начало 12-битного преобразования
1	0	0	x	1	Начало 8-битного преобразования
1	0	1	1	x	Считывание в 12-битном режиме
1	0	1	0	0	Считывание 8 старших битов, младшие в нуле
1	0	1	0	1	Считывание 4 битов по бокам, 4 бита по середине = 0

Так же, для того, чтобы понимать как происходит работа с АЦП, нужно иметь ввиду две временные диаграммы. Первая – состояние битов АЦП при преобразовании, показано на рисунке 4. Вторая – состояние битов при процессе считывания, показана на рисунке 5

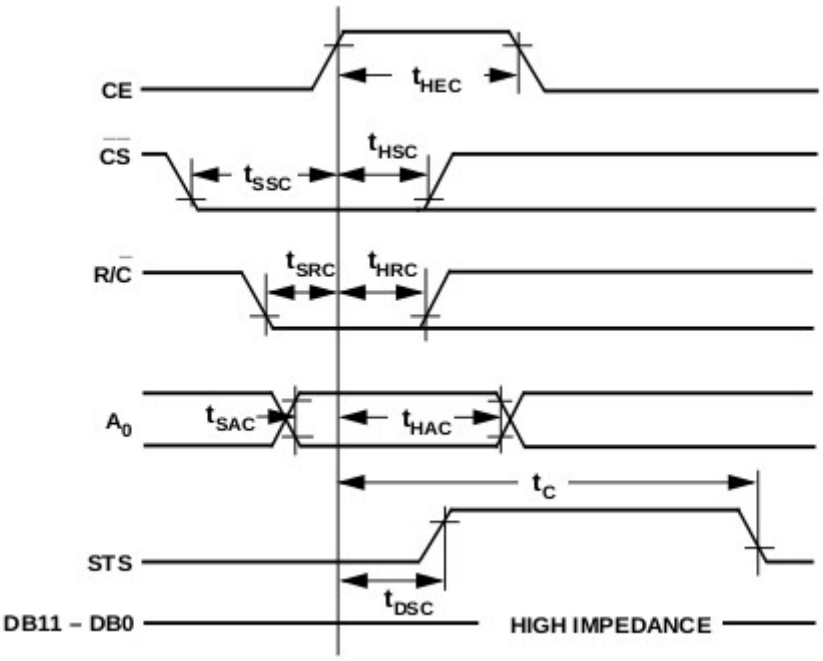

Рисунок 4 – Временная диаграмма преобразования

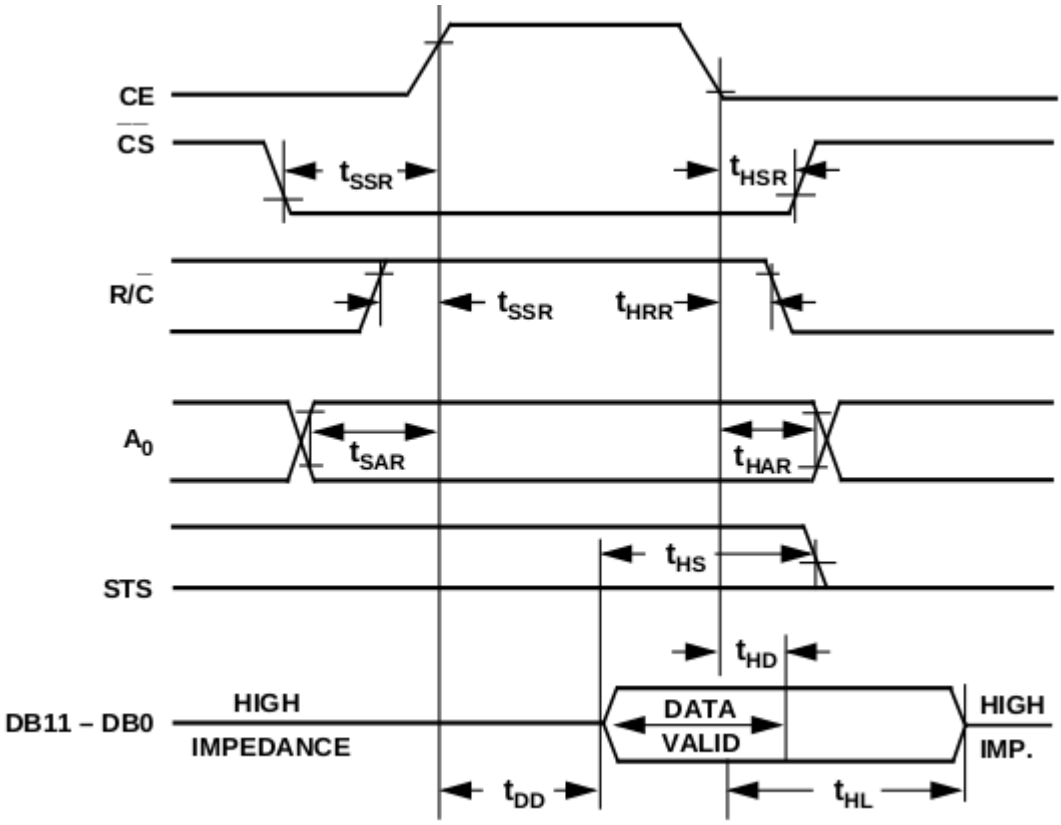

Рисунок 5 – Временная диаграмма считывания результатов преобразования.

2. Написание программы и моделирование работы АЦП

Начать стоит с построения схемы. Имея ввиду рисунок 3 результат сборки схемы в proteus показан на рисунке 6.

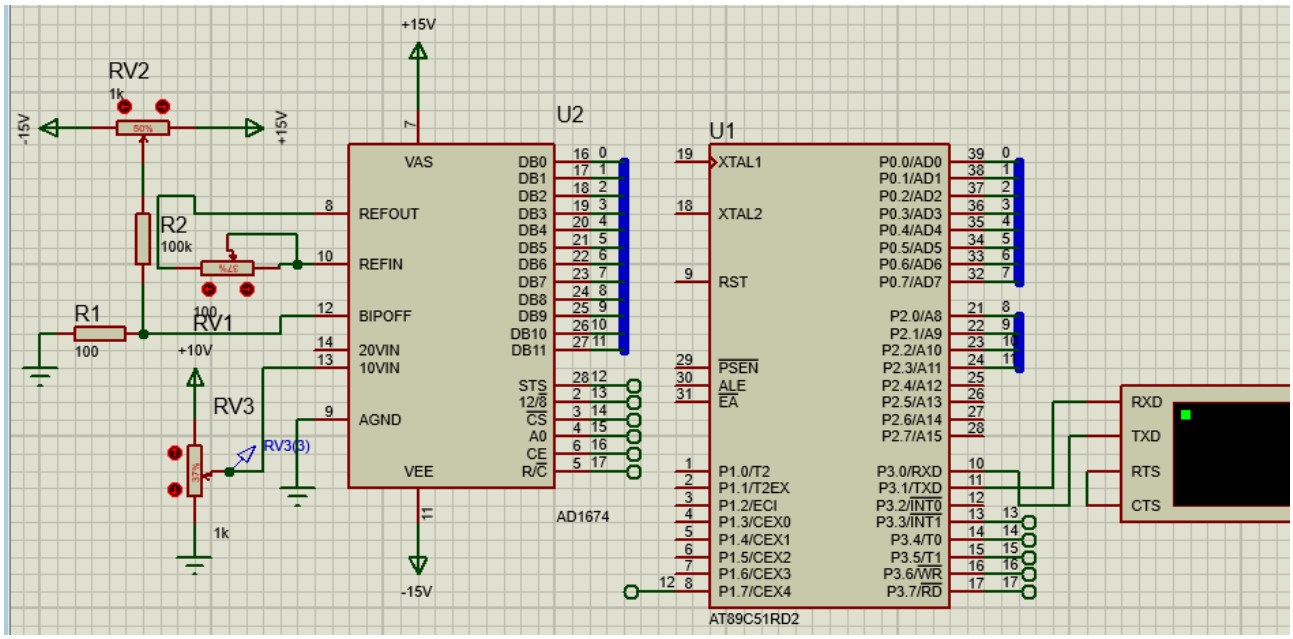

Рисунок 6 – Схема подключения AD1674 к AT89C51ED2

Теперь разберём программу. Будут рассмотрены только части кода связанные непосредственно с AD1674. Полный код программы приведён в приложении 1

Первым делом, определим режим работы АЦП. Для наибольшей точности будет использован 12 битный режим преобразования. При считывании будут передаваться все биты. Это задаётся следующей строчкой:

bit_12_8 = 1; CS = 0; A0 = 0; CE = 1;

Далее, нужно преобразовать сигнал и считать его. Это делается с помощью функции ad1674_read. Она показана ниже. Сначала мы обнуляем R/C бит, чтобы начать преобразование. Затем, ждём окончания преобразования, обнуления бита STS. Ну и наконец устанавливаем бит R/C и ждём один такт, чтобы считать результат преобразования. Наконец возвращаем результат преобразования из функции.

```
float ad1674_read(void){
    R_C = 0;
    while(STS==1);
    R_C = 1; R_C = 1;
```

```
    return P0 | ((P2 & 0xF) << 8);
}
```

Всё, на этом непосредственно работа с АЦП закончена. Далее мы преобразуем полученное значение в вольты и выводим с помощью uart. Результат показан на рисунках 7 и 8.

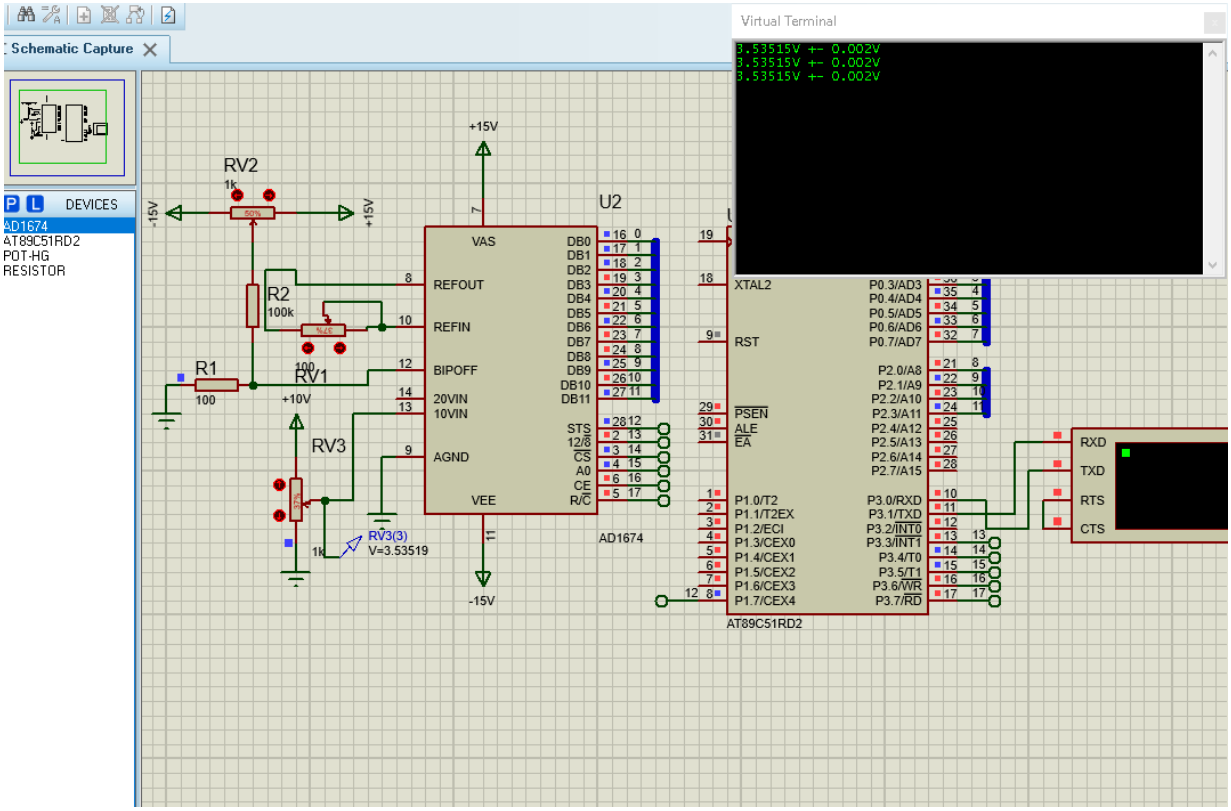

Рисунок 7 – Результат преобразования при напряжении на входе равном 3.53519 В

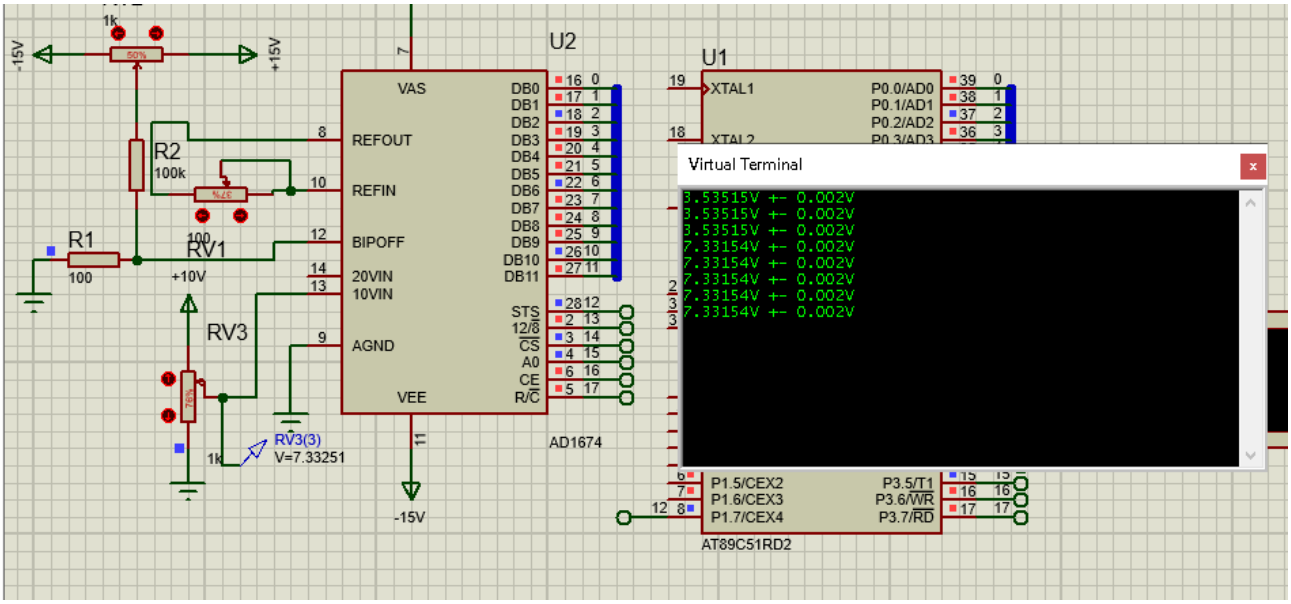

Рисунок 8 – Результат преобразования при 7.33251 на входе

Из полученных результатов видно, что значения поданные на вход АЦП и значение на выходе совпадают, учитывая погрешность в 0.002В

Список источников

1. AD1674 Datasheet. Электронный ресурс URL:https://www.analog.com/media/en/technical-documentation/data-sheets/AD1674.pdf (Дата обращения 13.02.2021)

2. Последовательный порт микроконтроллера 8051. Электронный ресурс URL:https://digteh.ru/MCS51/PoslPort.php (Дата обращения 13.02.2021)

ПРИЛОЖЕНИЕ 1

```
#include <at89c51xd2.h>
#include <stdio.h>

#define bit_12_8        P3_3
#define STS             P1_7
#define CS              P3_4
#define A0              P3_5
#define CE              P3_6
#define R_C             P3_7

void Delay(int nCount){
        while(nCount--);
}

void serial_init(void){
        SCON = 0x50;

        T2CON &= 0xF0;
        T2CON |= 0x30;
        TH2=0xFF;
        TL2=0xFD;
        RCAP2H=0xFF;
        RCAP2L=0xFD;
        TR2 = 1;
}

void serial_IT(char uart_data){
        SBUF = uart_data;
        while(TI==0);
        TI = 0;
}

float ad1674_read(void){
        R_C = 0;
        while(STS==1);
        R_C = 1; R_C = 1;
        return P0 | ((P2 & 0xF) << 8);
}

void String(char *str)
{
        int i;
        for(i = 0; str[i] != 0; i++){
```

```
                serial_IT(str[i]);
        }
}

void main (void)
{
        float adc_data;
        char adc_char_result[7];

        serial_init();
        bit_12_8 = 1; CS = 0; A0 = 0;  CE = 1;

        while(1){
                adc_data = 10*(ad1674_read()/4096);
                sprintf(adc_char_result, "%f", adc_data);
                String(adc_char_result);
                String("V +- 0.002V\n\r");
                Delay(1000000);
        }
}
```