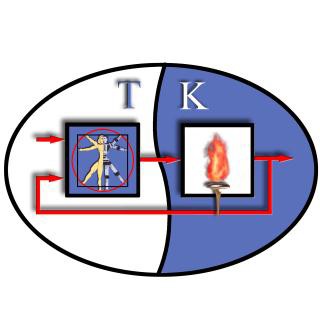
***МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ***



***ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ***

***ВЫСШЕГО ОБРАЗОВАНИЯ***

***«Белгородский Государственный Технологический Университет им. В.Г. Шухова» (БГТУ им. В.Г. Шухова)***

***ИЭИТУС***

***Кафедра Технической Кибернетики***

***Лабораторная работа № 3.2***

***дисциплина: «Микроконтроллеры в РТС»***

***тема: «Реализация АЦП последовательного приближения»***

***Выполнил:***

*студент группы МР-41 Рощук Р.Д.*

***Принял:***

*ст. преп. кафедры ТК*

*Гольцов Ю.А.*

*Белгород 2020*

**Цель работы**

Изучить принцип работы АЦП последовательного приближения, и реализовать заданный алгоритм на микроконтроллере 8051.

**Задание к работе**

Разработать программу, выполняющую преобразование уровень напряжений 0-5 вольт в числовые значения. Числовые значения вывести на UART. Программа должна использовать ЦАП и компаратор, входящие в симулятор EdSim51DI.

**Выполнение**

Аналого-цифровой преобразователь последовательного приближения (SAR, Successive Approximation Register) измеряет величину входного сигнала, осуществляя ряд последовательных «взвешиваний», то есть сравнений величины входного напряжения с рядом величин, генерируемых следующим образом:

1. на первом шаге на выходе встроенного цифро-аналогового преобразователя устанавливается величина, равная 1/2Uref (здесь и далее мы предполагаем, что сигнал находится в интервале (0 – Uref).

2. если сигнал больше этой величины, то он сравнивается с напряжением, лежащим посередине оставшегося интервала, т.е., в данном случае, 3/4Uref. Если сигнал меньше установленного уровня, то следующее сравнение будет производиться с меньшей половиной оставшегося интервала (т.е. с уровнем 1/4Uref).

3. Шаг 2 повторяется N раз. Таким образом, N сравнений («взвешиваний») порождает N бит результата.

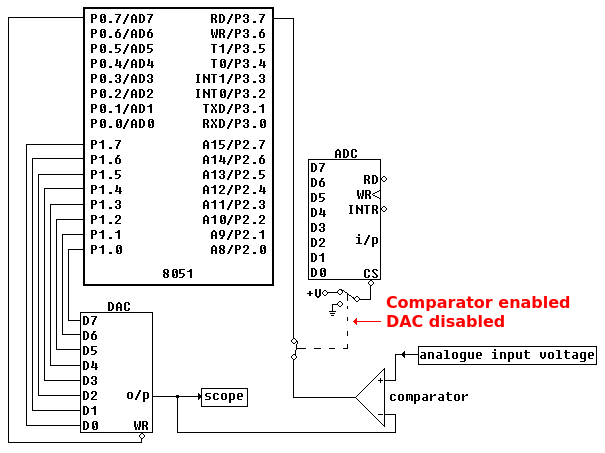


Рис 1. Схема

В начальный момент выполняется настройка таймера 1 (8бит с автозагрузкой) и разрешается прерывание от таймера 1. Срабатывание прерывания таймера 0 будет происходить раз в 10мс:

2^16-10000=55536=0D8F0h

TH0=0D8h; TL0=0F0h

Разрешаем все прерывания, запускаем таймеры и уходим в бесконечный цикл для работы только лишь на прерываниях.

В прерывании от таймера 0 загружаем ноль в вывод P0.7, тем самым включая микросхему ЦАП. Выключаем таймер 1 чтобы на индикаторах не было мусора. После этого возвращаем указанные выше значения TH и TL для поддержания 10мс периода опроса. Выводить данные будем в формате «1.234».

В регистрах 50h и 51h (старшая и младшая части) мы храним значение ЦАП.

В регистрах 56h и 57h мы храним дельты следующего изменения.

В регистре 58h мы храним перенос для дельты.

В регистре R0 мы храним значение ЦАП.

В регистре 59h мы храним дельту для ЦАПа.

Загружаем в регистры начальные значения от которых будем приближать и первые дельты изменения.

Далее в цикле высчитываем дельты для старшего и младшего регистра значений индикатора, а также для ЦАП. Выводим новое значение ЦАП в порт и смотрим на значение, подаваемое компаратором. Если оно больше 0, то необходимо отнять дельты (с учетом переносов). Если значение равно нулю, то необходимо прибавить дельты. После нескольких проходом данного цикла мы получим приближенное значение входного сигнала.

Т.к. мы выводим только по одному символу, то нам при каждом вызове прерывания надо будет выделять из этих двух регистров единицы, десятые, сотые и тысячные напряжения. Так для единиц и сотых будем определять целое от деления на 10, а десятых и тысячных как остаток от данного деления.

52h – единицы;

53h – десятые;

54h – сотые;

55h – тысячные;

Запускаем 1 таймер и SETB P0.7 включаем дешифратор для индикатора.

В конце происходит выход из прерывания.



Рис 2. Блок схема

**Вывод на индикаторы**

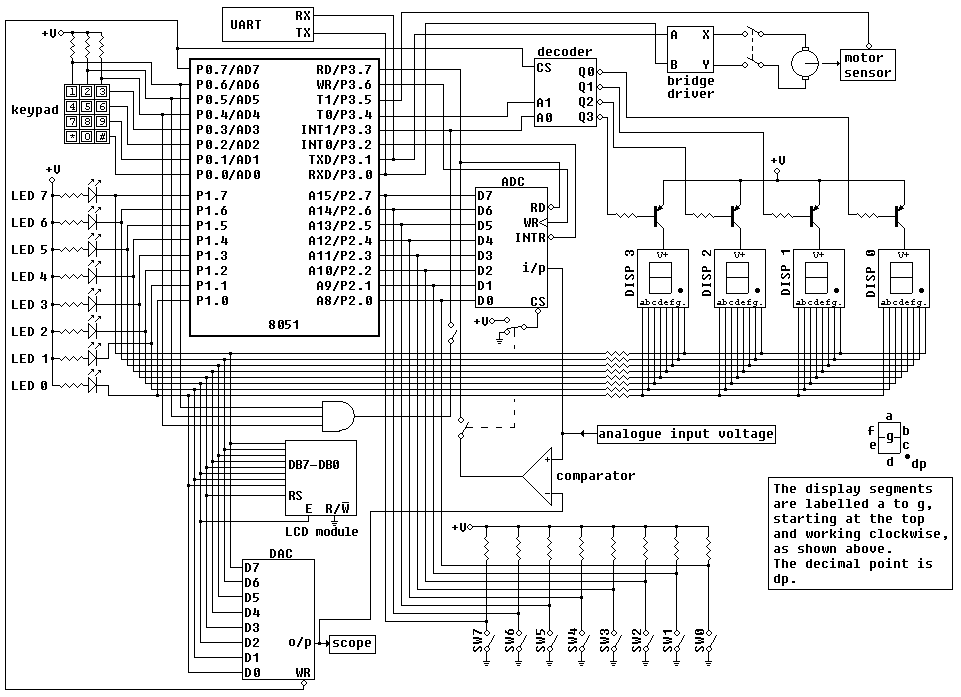
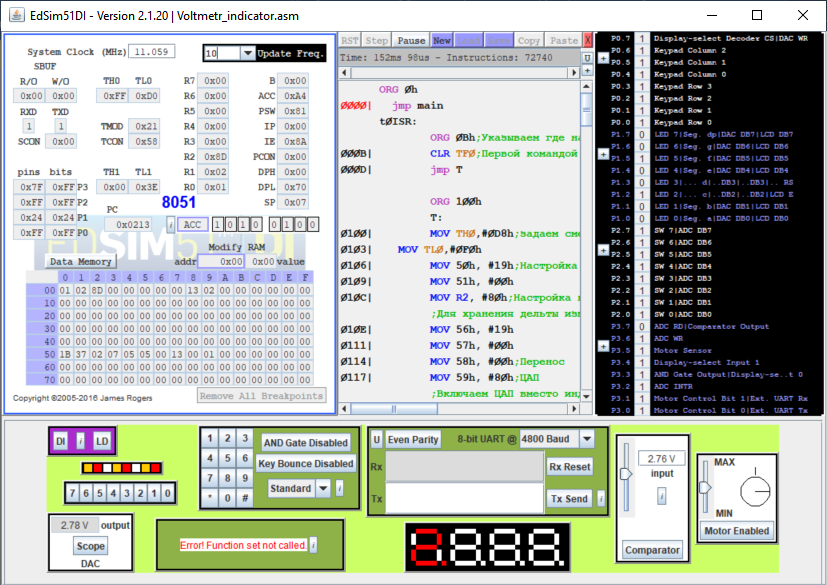
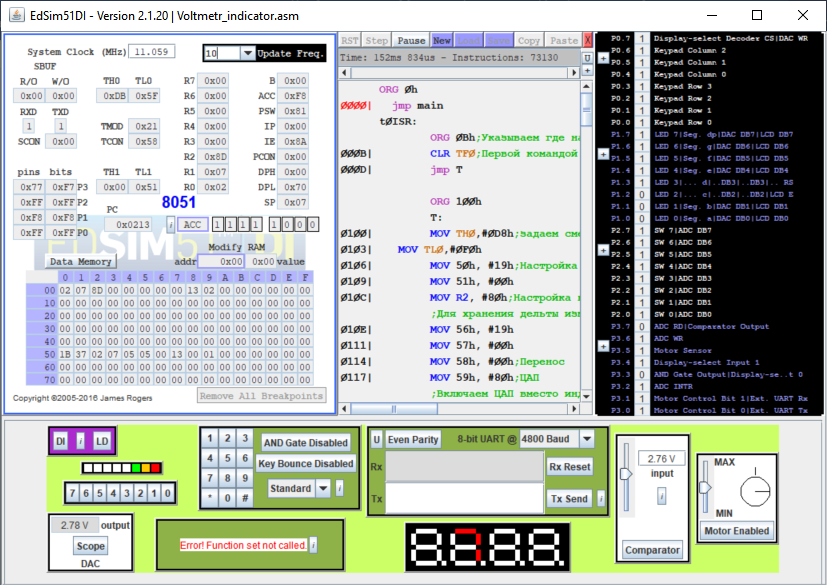


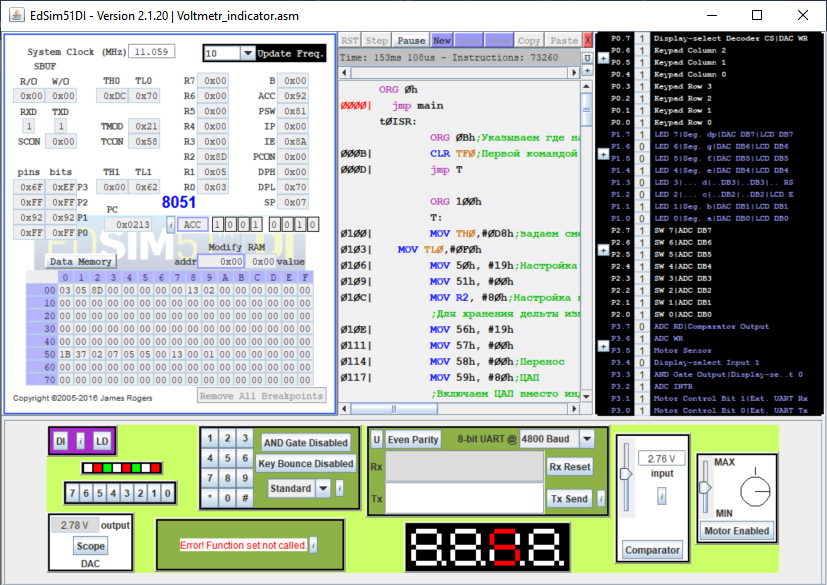
Рис 3. Схема симулятора

Вывод на индикаторы осуществляем с помощью таймера 1. При прерывании от его переполнения отображаем один из символов. Какой именно символ будет выводиться отвечает регистр R0. Для хранения кодов символов мы будем использовать память микроконтроллера с 70h. Загрузка в регистр А осуществляется через указатель данных DPTR (MOVC A, @A + DPTR). В нем хранится значение 70h. Для того чтобы иметь доступ к разным элементам таблица надо задавать смещение в А до пересылки в него значения. Таблица состоит из 10 значений кодов символов. Смещение хранится в регистре R1 и представляет собой цифру от 0 до 9 которую надо выводить.

**Примеры работы программ**

a) 

б) 

в) 

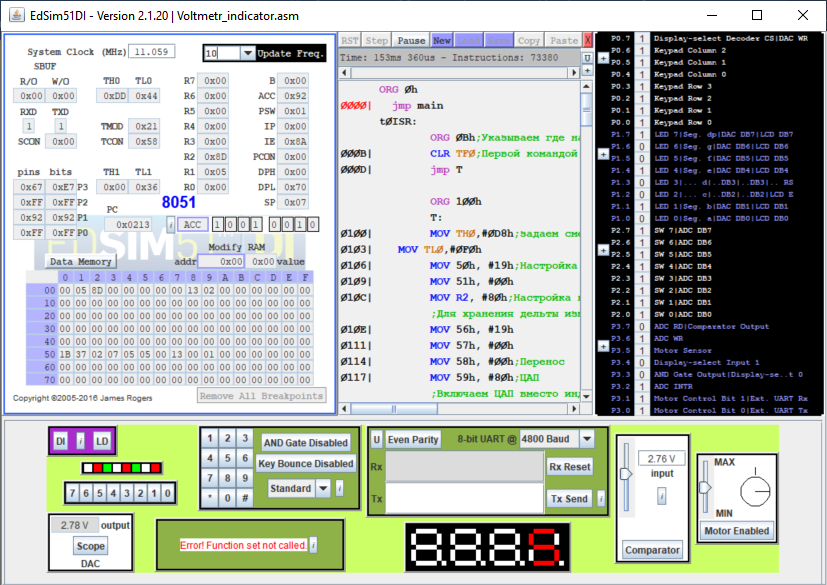
г) 

Рис 4. 2.76В, динамический вывод.

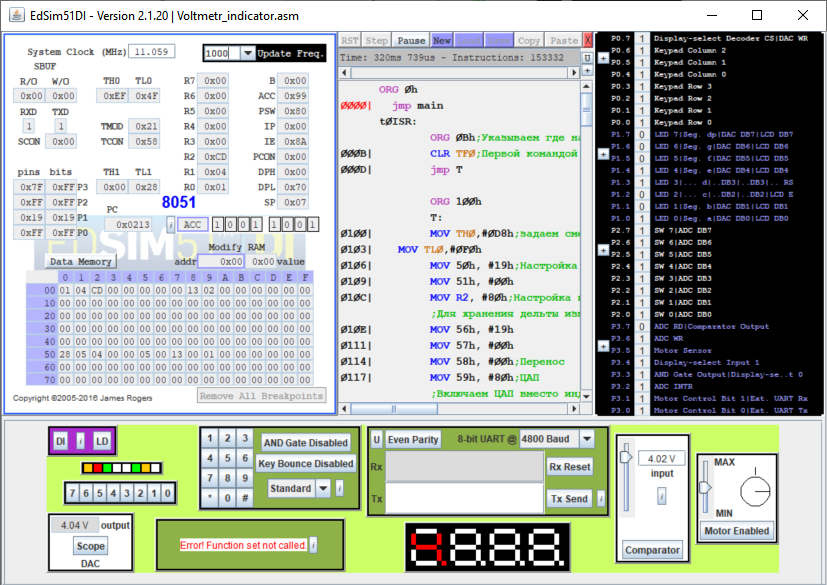


Рис 5. 4.02В

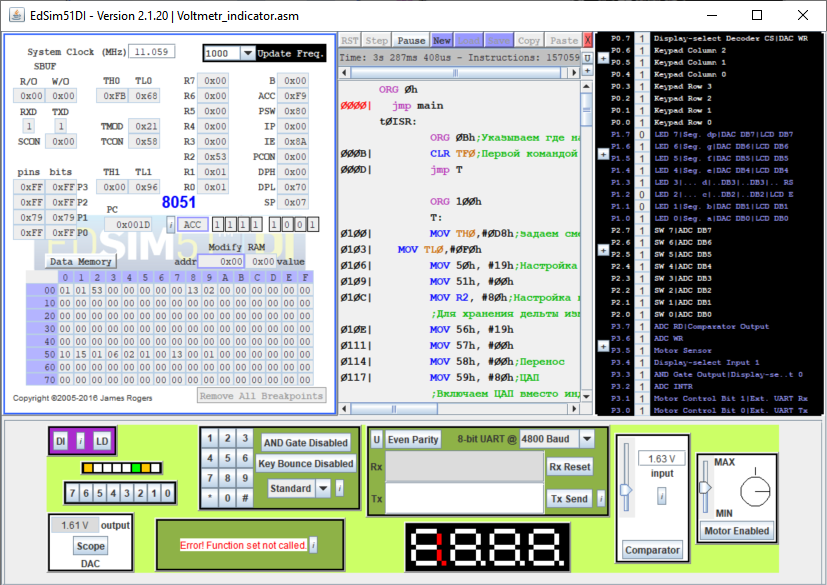


Рис 6. 1.63В

**Текст программы синус:**

ORG 0h

jmp main

t0ISR:

ORG 0Bh;Указываем где находится обработчик прерывания таймера 0

CLR TF0;Первой командой обработчика является команда очистки флага прерывания

jmp T

ORG 100h

T:

MOV TH0,#0D8h;задаем смещение срабатывания таймера

MOV TL0,#0F0h

MOV 50h, #19h;Настройка начальных значений АЦП (2500мВ)

MOV 51h, #00h

MOV R2, #80h;Настройка начальных значений ЦАП (128)

;Для хранения дельты изменения

MOV 56h, #19h

MOV 57h, #00h

MOV 58h, #00h;Перенос

MOV 59h, #80h;ЦАП

;Включаем ЦАП вместо индикаторов

CLR TR1;Останавливаем 1 таймер

CLR P0.7

Priblishenie:

;Расщитываем следующую дельту

MOV 58h, #0h

Mov B, #2h;Для вывода

Mov A,56h

Div AB

MOV 56h, A

Mov A,B

JZ Net\_perenoca;Переход, если аккумулятор равен нулю

MOV 58h, #32h;50

Net\_perenoca:

Mov B, #2h

Mov A,57h

Div AB

ADD A,58h

MOV 57h, A

Mov B, #2h;Для ЦАП

Mov A,59h

Div AB

MOV 59h, A

;Делаем приближение

mov P1,R2

JNB P3.7,umenhit;Переход, если бит равен 0

Mov A,51h

ADD A,57h

Mov B,#64h;Проверяем на перенос

Div AB

Mov 51h,B

ADD A,50h

ADD A,56h

Mov 50h,A

Mov A,R2

ADD A,59h

Mov R2,A

sjmp dalee

umenhit:

Mov A,50h

SUBB A,56h

Mov 50h,A

Mov A,51h

;Проверяем на перенос

SUBB A,57h

JNC BEZPER;Переход, если перенос не равен единице

Mov A,51h;Если будем занимать разряд

ADD A,#64h

SUBB A,57h

Mov 51h,A

Mov A,R2

SUBB A,59h

Mov R2,A

Mov A,50h;Отнимаем 1 из старшего регистра

SUBB A,#1h

Mov 50h,A

jmp dalee

BEZPER: Mov A,51h

SUBB A,57h

Mov 51h,A

Mov A,R2

SUBB A,59h

Mov R2,A

dalee:

inc R7

CJNE R7,#7h,Priblishenie ;Сравнение с константой и переход, если не равно

mov R7,#0h

;Разделяем на разряды

mov A,50h

mov B, #0Ah

div AB

mov 52h,A

mov 53h,B

mov A,51h

mov B, #0Ah

div AB

mov 54h,A

mov 55h,B

MOV A,#00h;Чтобы не ругался эмулятор

MOV B,#00h

;Включаем индикаторы вместо ЦАП

SETB TR1;Запускаем 1 таймер

SETB P0.7

RETI;Выход из обработчика прерываний

t1ISR:

ORG 1Bh;Указываем где находится обработчик прерывания таймера 1

CLR TF1;Первой командой обработчика является команда очистки флага прерывания

;Выводим на циферблат

MOV P1,#0FFh;Очищаем экран

CJNE R0,#0h,D1;единицы

mov R1,52h;Загружаем выводимую цифру

SETB p3.3;Настраиваем один из 4 индикаторов на вывод

SETB p3.4

D1:

CJNE R0,#1h,D2;десятые

mov R1,53h

CLR p3.3

SETB p3.4

D2:

CJNE R0,#2h,D3;сотые

mov R1,54h

SETB p3.3

CLR p3.4

D3:

CJNE R0,#3h,D4;тысячные

mov R1,55h

CLR p3.3

CLR p3.4

D4:

;Выводим нужную цифру

mov DPL,#70h ;Загрузка указателя данных

mov A, R1;Смещение

movc A,@A+DPTR ;Пересылка в аккумулятор байта из ПП

MOV P1,A;Загружаем новое значение

CJNE R0,#0h,T1;Точка после единиц

CLR P1.7

T1:

inc R0

CJNE R0,#00000100b,NET;Сравнение регистра с константой и переход, если не равно

MOV R0,#0b

NET:

RETI;Выход из обработчика прерываний

ORG 70h

db 11000000b

db 11111001b

db 10100100b

db 10110000b

db 10011001b

db 10010010b

db 10000010b

db 11111000b

db 10000000b

db 10010000b

ORG 200h

main:

;T1 M1=1 M0=0 для переключения сегментов 8 бит

;T0 M1=0 M0=1 для АЦП/16бит

MOV TMOD,#00100001b

;обновление данных 100гц(10мс)

MOV TH0,#0FFh;2^16-10000=55536=0D8F0h

MOV TL0,#099h

;разрешаем прерывание от таймера 0

SETB ET0

;разрешаем прерывание от таймера 1

SETB ET1

;разрешаем выполнение прерываний

SETB EA

SETB TR0;запускаем таймер 0

SETB TR1;запускаем таймер 1

jmp $;бесконечный цикл ожидания

END