**Министерство науки и высшего образования Российской̆ Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования  
«Московский̆ государственный̆ технический̆ университет   
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)**

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ» (ИУ) |
| КАФЕДРА | «ПРОЕКТИРОВАНИЕ И ТЕХНОЛОГИЯ ПРОИЗВОДСТВА ЭА» (ИУ4) |

**Домашнее задание**

по дисциплине:

**ПРОЕКТИРОВАНИЕ МИКРОПРОЦЕССОРНЫХ СИСТЕМ**

**Вариант 1**

**Группа:** ИУ4-72Б

**Студент:** Дычаковская А.А.

**Преподаватель:** Леонидов В.В.

ИУ4-72Б

Дычаковская А.А.

Леонидов В.В.

Москва, 2023 г.

**Преподаватель:**

ИУ4-72Б

Дычаковская А.А.

Леонидов В.В.

Москва, 2023 г.

Москва, 2023 г.

1. Задание

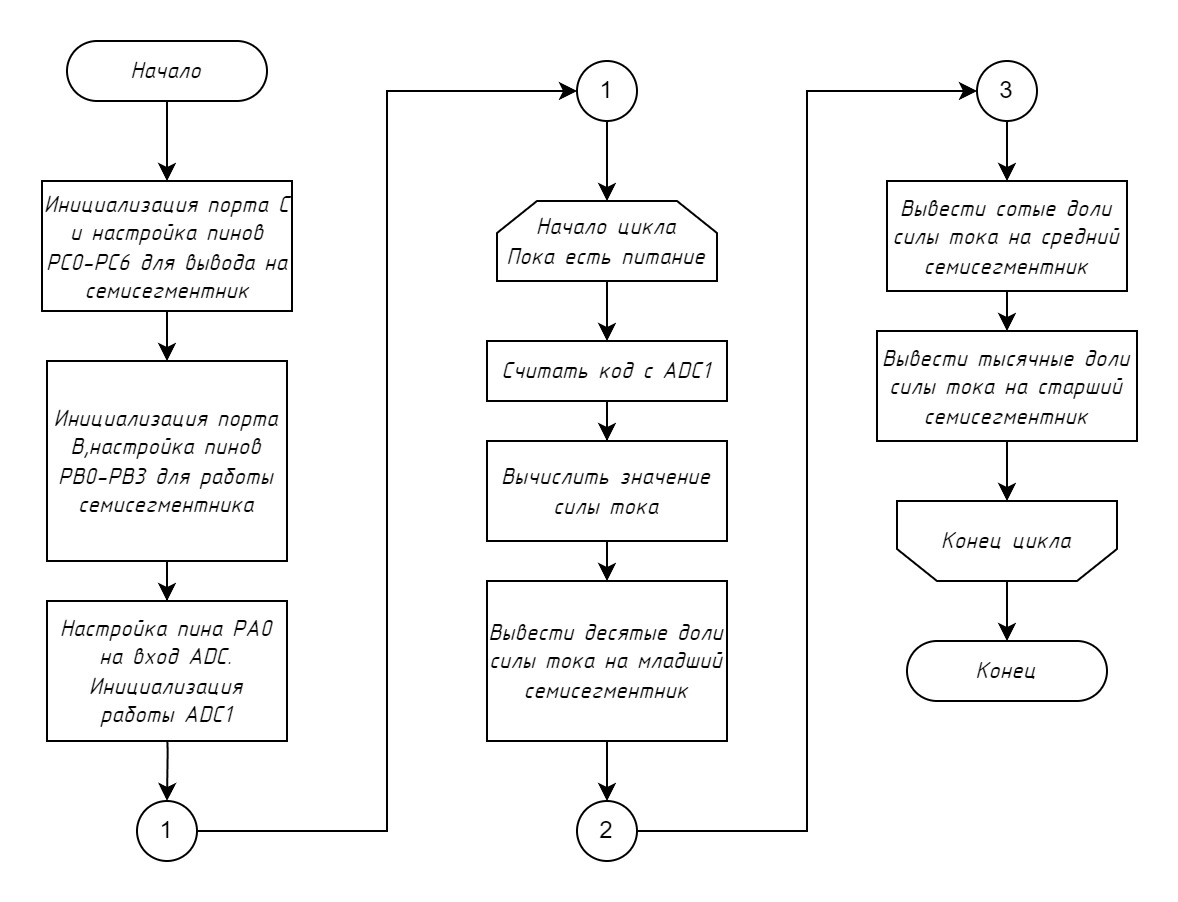
Задача №2

Разработать схему электрическую принципиальную (если задание требует разработку устройства) и программу на языке C с использованием библиотеки CMSIS согласно Вашему варианту. Использование сторонних библиотек (HAL, LL, StdPeriph и т.п.) для работы с периферией МК или внешними устройствами не допускается. В качестве МК использовать STM32F103RBT6, частоту системной шины принять равной 64 МГц.

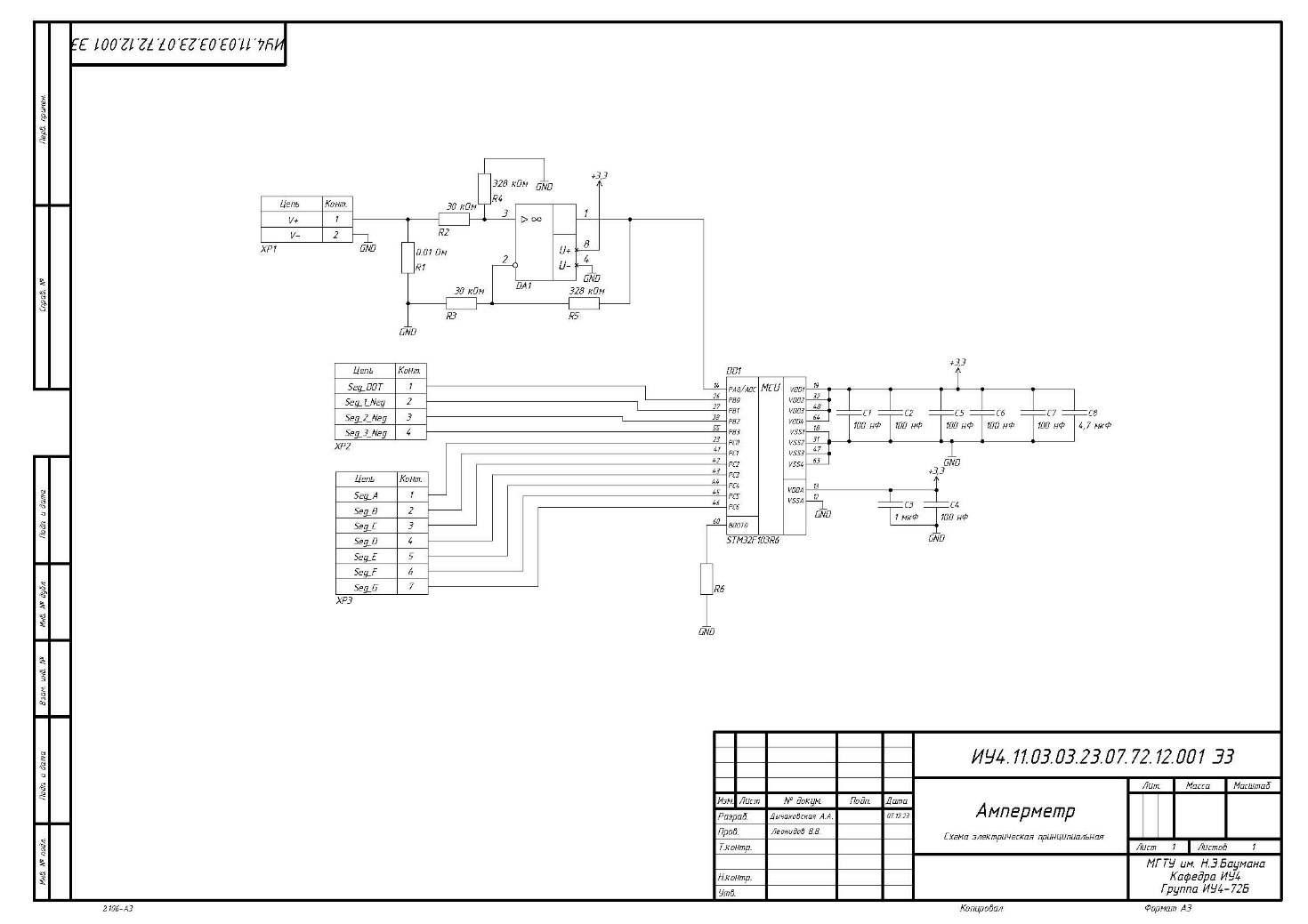
Разработать цифровой амперметр с диапазоном измерений от 0 А до 30 А с точностью измерения не хуже ± 0,1 А. Значение тока в Амперах отображать на трехразрядном семисегментном индикаторе в виде числа с фиксированной точкой.

2. Алгоритмы работы

Алгоритм работы цифрового амперметра представлен на блок-схеме ниже.



3. Схема электрическая принципиальная (Э3)

****

4. Листинг кода

Листинг кода для задачи 2. Файл main.h:

**#ifndef** \_\_MAIN\_H

**#define** \_\_MAIN\_H

**#include** "stm32f1xx.h"

**#define** disp\_0() GPIOC->ODR = 0x3F; //0011 1111

**#define** disp\_1() GPIOC->ODR = 0x6; //0000 0110

**#define** disp\_2() GPIOC->ODR = 0x5b; //0101 1011

**#define** disp\_3() GPIOC->ODR = 0x4f; //0100 1111

**#define** disp\_4() GPIOC->ODR = 0x66; //0110 0110

**#define** disp\_5() GPIOC->ODR = 0x6d; //0110 1101

**#define** disp\_6() GPIOC->ODR = 0x7d; //0111 1101

**#define** disp\_7() GPIOC->ODR = 0x7; //0000 0111

**#define** disp\_8() GPIOC->ODR = 0x7f; //0111 1111

**#define** disp\_9() GPIOC->ODR = 0x6f; //0110 1111

**#define** indicator\_0() GPIOB->ODR = 0xc; //1100

**#define** indicator\_1() GPIOB->ODR = 0xb; //1011

**#define** indicator\_2() GPIOB->ODR = 0x6; //0110

**void** **init\_portC**(**void**);

**void** **init\_portB**(**void**);

**void** **init\_ADC**(**void**);

**void** **disp**(**int** result);

**void** **indicator**(**int** i);

uint16\_t **ADC\_proc**(uint8\_t n);

**void** **delay**(uint32\_t value);

**#endif**

Файл main.c:

**#include** "main.h"

**void** **init\_portC**(**void**) {

//пины PC0-PC7 отвечают за сегменты индикатора

RCC->APB2ENR |= RCC\_APB2ENR\_IOPCEN; /\*вкл тактирование порта с\*/

GPIOC->CRL &= ~(GPIO\_CRL\_CNF0|GPIO\_CRL\_MODE0); /\*зануляем биты CRL (0000)\*/

GPIOC->CRL |= GPIO\_CRL\_MODE0; /\*ставим маску 0011 - 50MHz output mode\*/

GPIOC->CRL &= ~(GPIO\_CRL\_CNF1|GPIO\_CRL\_MODE1);

GPIOC->CRL |= GPIO\_CRL\_MODE1;

GPIOC->CRL &= ~(GPIO\_CRL\_CNF2|GPIO\_CRL\_MODE2);

GPIOC->CRL |= GPIO\_CRL\_MODE2;

GPIOC->CRL &= ~(GPIO\_CRL\_CNF3|GPIO\_CRL\_MODE3);

GPIOC->CRL |= GPIO\_CRL\_MODE3;

GPIOC->CRL &= ~(GPIO\_CRL\_CNF4|GPIO\_CRL\_MODE4);

GPIOC->CRL |= GPIO\_CRL\_MODE4;

GPIOC->CRL &= ~(GPIO\_CRL\_CNF5|GPIO\_CRL\_MODE5);

GPIOC->CRL |= GPIO\_CRL\_MODE5;

GPIOC->CRL &= ~(GPIO\_CRL\_CNF6|GPIO\_CRL\_MODE6);

GPIOC->CRL |= GPIO\_CRL\_MODE6;

}

**void** **init\_portB**(**void**) {

/\*вкл тактирование порта В\*/

RCC->APB2ENR |= RCC\_APB2ENR\_IOPBEN;

//пин PB0 отвечает за точку 2-го индикатора

GPIOB->CRL &= ~(GPIO\_CRL\_CNF0|GPIO\_CRL\_MODE0);

GPIOB->CRL |= GPIO\_CRL\_MODE0;

//пины PB1-PB3 отвечают за землю индикаторов

GPIOB->CRL &= ~(GPIO\_CRL\_CNF1|GPIO\_CRL\_MODE1);

GPIOB->CRL |= GPIO\_CRL\_MODE1;

GPIOB->CRL &= ~(GPIO\_CRL\_CNF2|GPIO\_CRL\_MODE2);

GPIOB->CRL |= GPIO\_CRL\_MODE2;

GPIOB->CRL &= ~(GPIO\_CRL\_CNF3|GPIO\_CRL\_MODE3);

GPIOB->CRL |= GPIO\_CRL\_MODE3;

//пин PA0 отвечает за вход АЦП

RCC->APB2ENR |= RCC\_APB2ENR\_IOPAEN;

GPIOA->CRL &= ~(GPIO\_CRL\_CNF0|GPIO\_CRL\_MODE0);

}

**void** **init\_ADC**(**void**){

RCC->APB2ENR |= RCC\_APB2ENR\_ADC1EN; //Включить тактирование АЦП

//Настройка времени преобразования каналов

ADC1->SMPR2 |= ADC\_SMPR2\_SMP0; //Канал 0 - 239.5 тактов

ADC1->CR2 |= ADC\_CR2\_EXTSEL; //Выбрать в качестве источника запуска SWSTART

ADC1->CR2 |= ADC\_CR2\_ADON; //Включить АЦП

delay(10); //Задержка перед калибровкой

ADC1->CR2 |= ADC\_CR2\_CAL; //Запуск калибровки

**while** ((ADC1->CR2 & ADC\_CR2\_CAL) != 0);//ожидание окончания калибровки

}

**void** **disp**(**int** result) { //отображаем число на одном индикаторе

**switch** (result) {

**case** 0: disp\_0(); **break**;

**case** 1: disp\_1(); **break**;

**case** 2: disp\_2(); **break**;

**case** 3: disp\_3(); **break**;

**case** 4: disp\_4(); **break**;

**case** 5: disp\_5(); **break**;

**case** 6: disp\_6(); **break**;

**case** 7: disp\_7(); **break**;

**case** 8: disp\_8(); **break**;

**case** 9: disp\_9(); **break**;

}

}

**void** **indicator**(**int** i) { //включаем определенный семисегментник

**switch** (i) {

**case** 0: indicator\_0(); **break**;

**case** 1: indicator\_1(); **break**;

**case** 2: indicator\_2(); **break**;

}

}

uint16\_t **ADC\_proc**(uint8\_t n) {

ADC1->SQR3 = n; //Записываем номер канала в регистр SQR3

ADC1->CR2 |= ADC\_CR2\_SWSTART; //Запускаем преобразование в регулярном канале

**while**((ADC1->SR & ADC\_SR\_EOC) != 0);//Ждем окончания преобразования

**return** ADC1->DR; //Читаем результат

}

**int** **main**(**void**) {

init\_portC();

init\_portB();

init\_ADC();

**while**(1) {

uint16\_t result = ADC\_proc(0);//в result считываем показания с АЦП

result = result\*4560/4096;

indicator(0);

disp(result / 10 % 10);

delay(1000);

indicator(1);

disp(result / 100 % 10);

delay(1000);

indicator(2);

disp(result / 1000);

delay(1000);

}

}

**void** **delay**(uint32\_t value) {

**for** (uint32\_t i=0;i<value;i++);

}

**5. Проверка работоспособности**

Проверка работоспособности задачи 2. Для проверки работоспособности схема была собрана и смоделирована в программе Proteus 8.11. Ниже приведены результаты моделирования для различных значений тока.

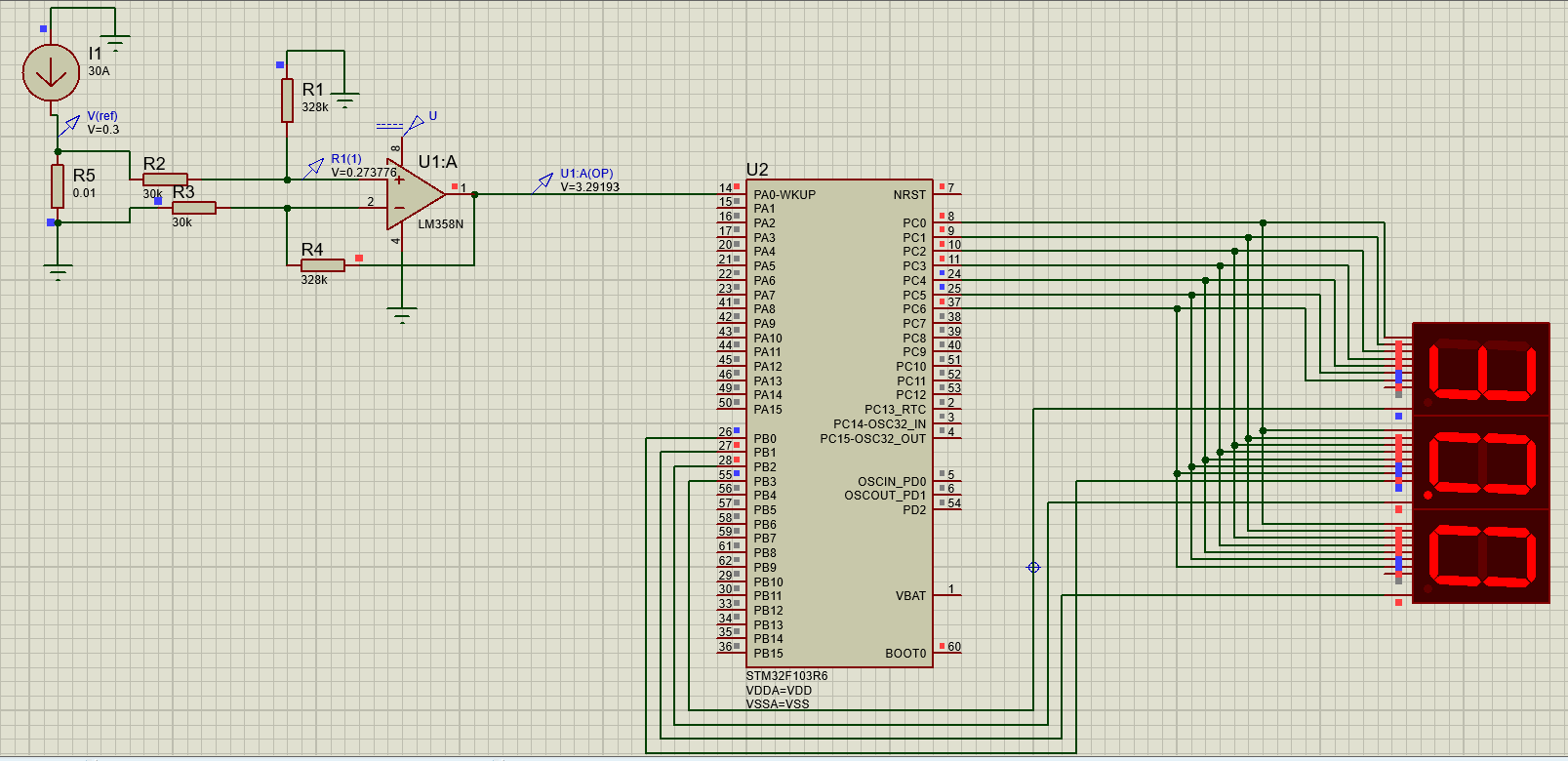


Рисунок 1 – Результат моделирования при

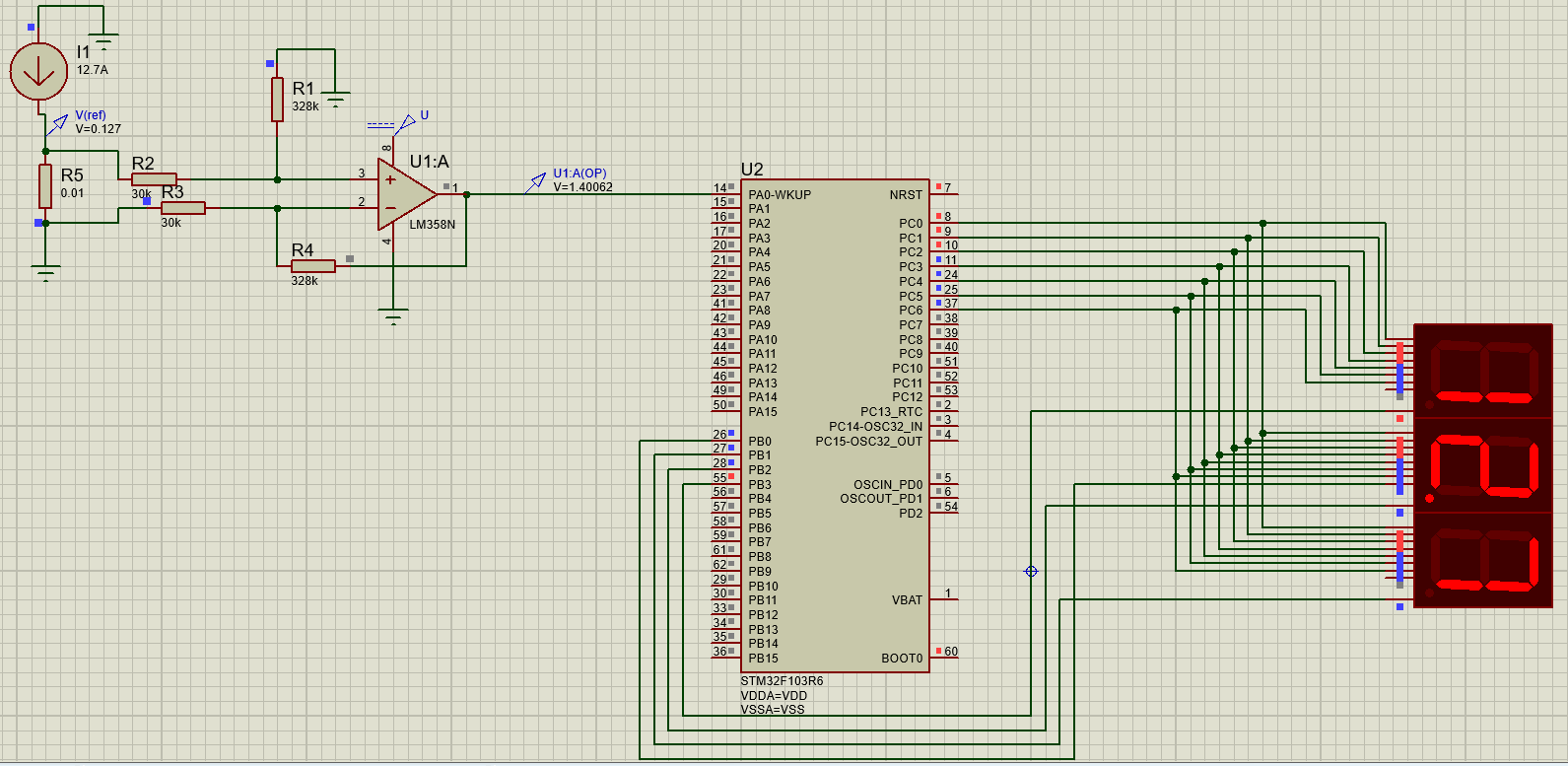


Рисунок 2 – Результат моделирования при

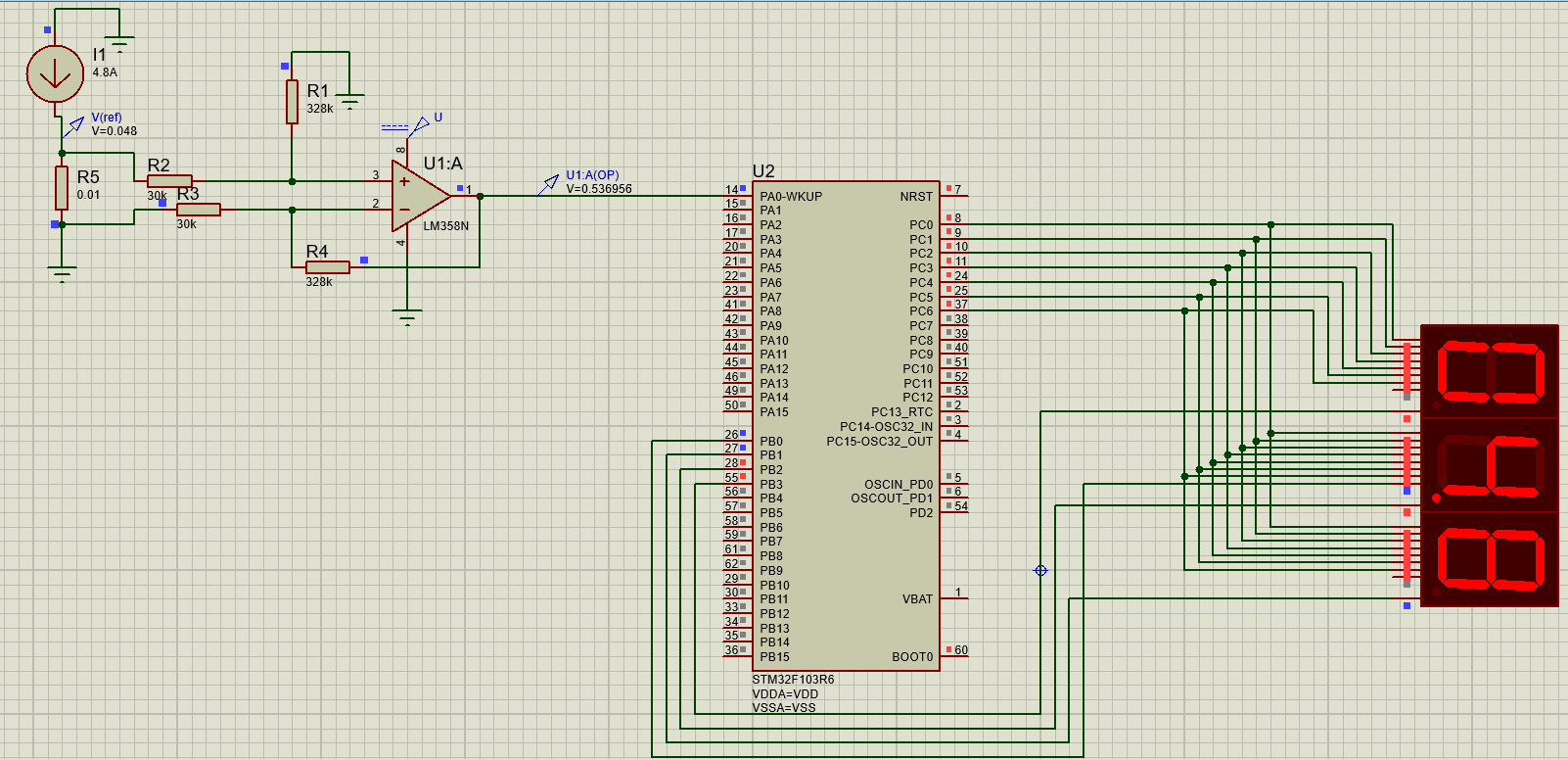


Рисунок 3 – Результат моделирования при

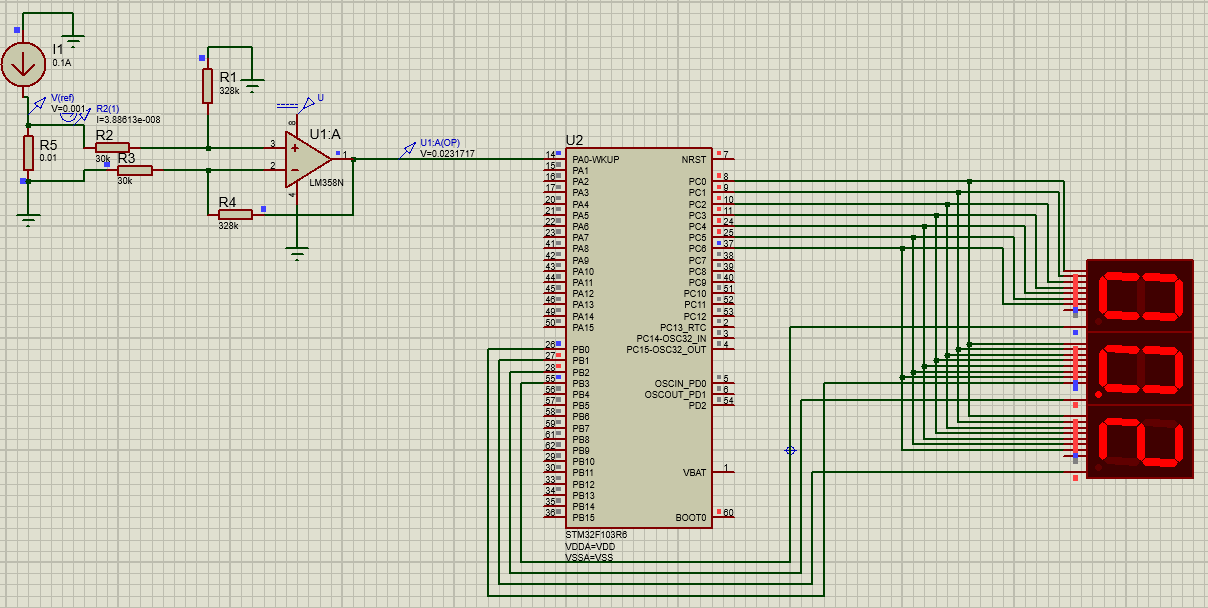


Рисунок 4 – Результат моделирования при

**Заключение**

В ходе данной работы был получен опыт написания кода на языке Си с использованием библиотеки CMSIS, а также разработан цифровой амперметр с диапазоном измерения от 0 до 30 А и точностью ± 0,1 А.