|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

Институт кибернетики

*(наименование института, филиала)*

Кафедра проблем управления

*(наименование кафедры)*

**КУРСОВОЙ ПРОЕКТ (РАБОТА)**

по дисциплине Микропроцессорная техника в мехатронике и робототехнике

(*наименование дисциплины)*

**Тема курсового проекта** (работы) Реализация устройства «Обработчик сигналов инкрементного датчика»

**Студент группы** КРБО-02-19 Гаджиев Даниял Стефанович **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*(учебная группа, фамилия, имя, отчество студента*) *(подпись студента)*

**Руководитель курсового проекта (работы)** Трипольский П.А., доцент, к.т.н.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*должность, звание, ученая степень (подпись руководителя)*

**Рецензент** (при наличии) **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*должность, звание, ученая степень (подпись рецензента)*

Работа представлена к защите «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20\_\_г.

Допущен к защите «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20\_\_г.

Оглавление

[**Постановка задачи** 2](#_Toc90946700)

[**Алгоритм Энкодера** 3](#_Toc90946701)

[**Аппаратные средства** 6](#_Toc90946702)

[**Описание регистров для ЮАРТ** 7](#_Toc90946703)

[**Описание регистров для внешних прерываний** 8](#_Toc90946704)

[**Общие сведения о программе** 9](#_Toc90946705)

[**Описание программы:** 9](#_Toc90946706)

[**Использование программой внешних прерываний:** 9](#_Toc90946707)

[**Структура данных Регистры:** 10](#_Toc90946708)

[**Структура подпрограмм и их назначения** 11](#_Toc90946709)

[**Структура программного обеспечения** 12](#_Toc90946710)

[**Методика и результаты тестирования** 13](#_Toc90946711)

[**Вывод:** 18](#_Toc90946712)

[**Список использованной литературы:** 19](#_Toc90946713)

[**Листинг программы.** 20](#_Toc90946714)

# 

# **Постановка задачи**

Требуется реализовать устройство типа «Обработчик сигналов инкрементного датчика» (энкодер). Счетчик должен быть 32-х битным. Для вывода информации счетчика необходимо использовать внешний интерфейс (к примеру ЮАРТ), в который нужно передать число пачками по 8 разрядов.

# **Алгоритм работы Энкодера**

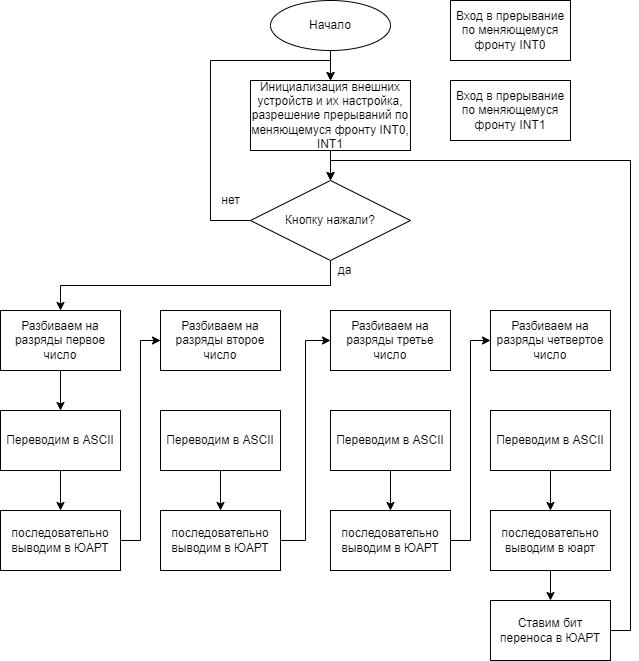


Рис. 1. Блок-схема алгоритма решения задачи

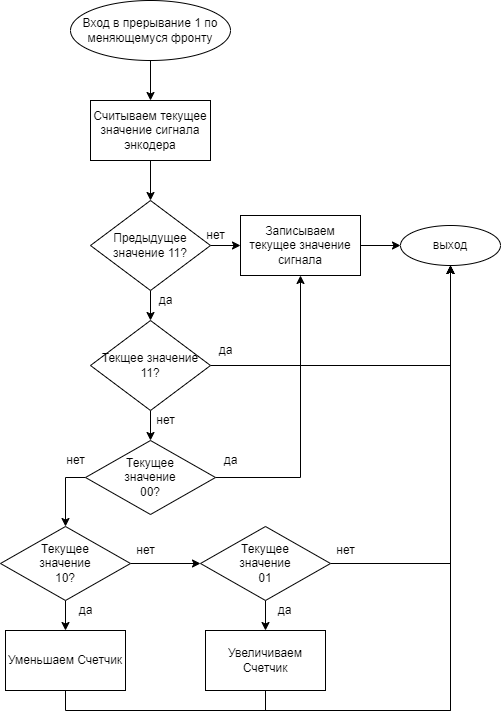
****

Рис. 2. Блок-схема алгоритма прерываний регистров INT0, INT1 по меняющемуся фронту

# **Аппаратные средства**

Для реализации проекта использовался микроконтроллер Atmega8

Устройство состоит из подключенных к микроконтроллеру инкрементного энкодера, с подключенными к нему кнопками для врещения, внешнего интерфейса ЮАРТ, и кнопки для вывод информации в юарт.

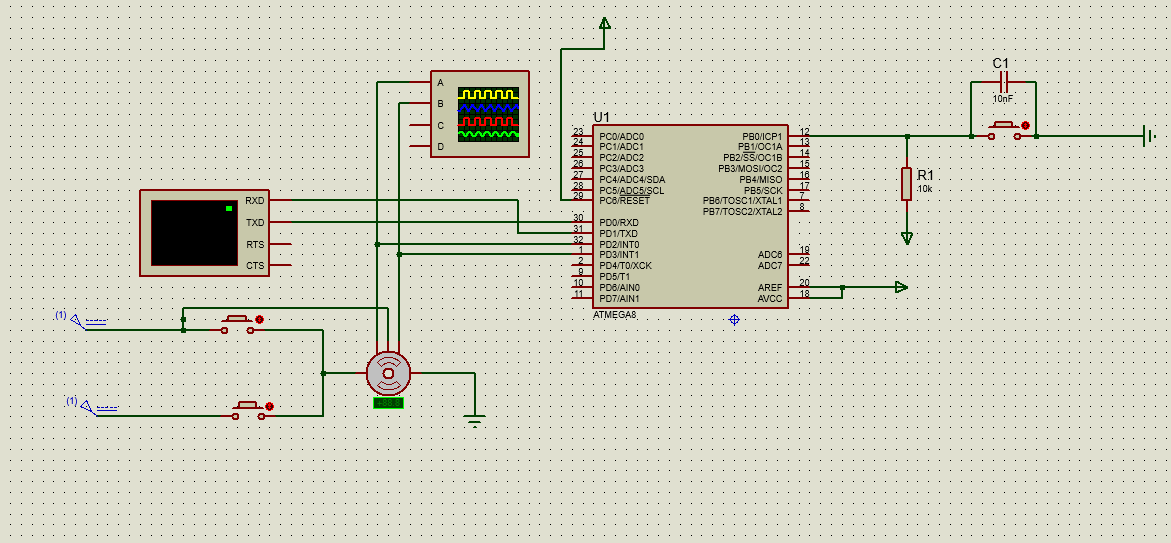
Для подавления дребезга контактов кнопок было использовано аппаратное решение с применением RC-цепочек. Суть данного решения заключается в том, что к портам контроллера подключены кнопки с резисторами номиналом 10кОм, а параллельно с кнопками в схему включены конденсаторы. Расчет емкости конденсаторов основан на формуле:

Где τ - время зарядки конденсатора, R - сопротивление резистора, C - емкость конденсатора. В данном случае время зарядки конденсатора составит 0.01 с.

Электрическая схема устройства показана на Рис. 2. Используемые выходы микроконтроллера показаны в таблице 1

Таблица 1. Список использованных выводов

|  |  |
| --- | --- |
| PD0, PD1 | Шина данных для вывода информации в ЮАРТ |
| PD1, PD2 (INT0, INT1) | Считывание сигнала с энкодера прерывания по меняющимся фронатам |
| PB0 | Пин приема сигнала с кнопки для вывода инфы в ЮАРТ |

****** Рис. 3. Принципиальная схема устройства в Proteus

# **Описание регистров для ЮАРТ**

Для работы с ЮАРТ нужно его правильно настроить, в частности, разрешить прием данных, а также настроить скорость передачи данных.

**Необходимые регистры и их настройка:**

**UCSRA** - Управляющий регистр A. Используемые биты:

* **U2X** - выставляет асинхронный режим работы с удвоенной скоростью

**UCSRB** – Разрешающий обмен информацией. Используем биты:

* **RXEN** - бит разрешающий чтение информации
* **TXEN** - бит разрешающий запись информации

**UCSRC** - Управляющий регистр A. Используемые биты:

* **URSEL** - бит определяет, в какой из регистров модуля производится запись. Если бит установлен, обращение производится к регистру UCSRC; если бит сброшен, обращение производится к регистру UBRRH.
* **UCSZ1/0** - Совместно с разрядом UCSZ2 регистра UCSRB эти разряды определяют количество разрядов данных в посылках

# **Описание регистров для внешних прерываний**

Для работы с постоянно меняющимся ШИМ были задействованы внешние прерывания (INT0/INT1) по меняющемуся фронту.

**Необходимые регистры:**

- **GICR** - Регистр управления маскированием внешних прерываний и размещением таблицы векторов прерываний. Используемые биты:

- **INT0** - Бит, разрешающий запрос внешнего прерывания INT0

- **INT1** - Бит, разрешающий запрос внешнего прерывания INT1

- **MCUCR** - Регистр, содержащий биты управления режимами пониженного энергопотребления MCU и распознаванием сигналов внешних прерываний INT1 и INT0. Используемые биты:

- **ISC01:0** - два бита, определяющие форму сигналов запроса внешнего прерываний INT0. Записываем биты следующим образом: “10" — запрос прерывания происходит по спадающему фронту на входе INT0.

- **ISC11:0** - два бита, определяющие форму сигналов запроса внешнего прерываний INT1. Записываем биты следующим образом: “10" — запрос прерывания происходит по спадающему фронту на входе INT1.

# **Общие сведения о программе**

Название: Encoder

Автор: Гаджиев Даниял Стефанович

Группа: КРБО-02-19.

Дата создания: 19.12.2021.

Программа была написана на языке Assembler с использованием программного комплекса Proteus 8 Professional. Proteus8 – использовался для написания кода и детальной работы с регистрами и для моделирования принципиальной электрической схемы, сборки и отладки конечной версии устройства.

## **Описание программы:**

После инициализации портов и настройки внешних устройств программа переходи в бесконечный цикл, который постоянно проверяет состояние кнопки с выхода B, если логический уровень кнопки меняется происходит переход в подпрограмму, которая поочередно для всех 4-х разрядов счетчика производит разбиение регистра на разряды, после перевод в формат ASCII последовательно выводит их во внешнее устройство

## **Использование программой внешних прерываний:**

Внешние прерывания INT0 и INT1 используются, чтобы считать кол-во оборотов энкодера.

При каждом повороте энкодера на входы INT0, INT1, подаются 2 ШИМ сигнала со скважностью 50% и разность фаз в 90 градусов.

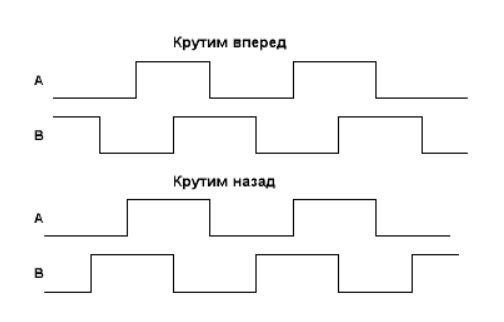


Рис.4. Сигнал, поступающий на выходы INT0, INT1

Каждый раз, когда меняется фронт на одном из выходов происходит прерывание, которое сравнивает предыдущее и нынешнее состояния поступающих ШИМ сигналов, при смене логических уровней с 11 на 10 или 01 алгоритм увеличивает или уменьшает значение 32-х битного счетчика.

# **Структура данных Регистры:**

1. r0 – регистр хранящий результат произведений
2. r15 – регистр для проверки в подпрограмме задержки
3. r16 – регистр для учета переполнений
4. r17 – регистр для инкремента и декремента счетчика
5. flag (r18) – регистр хранящий предыдущее значение лог. уровней INT0, INT1

1. flag2 (r19) – регистр хранящий текущее значение лог. уровней INT0, INT1
2. (r20-r24) – регистры, выполняющие функцию счетчика
3. hundreds (r24) – регистр хранящий разряд сотен
4. dozens (r25) – регистр хранящий разряд сотен
5. ones (r26) – регистр хранящий разряд единиц
6. copy, copy2 (r27, r28) – вспомогательные регистры для разбиения на разряды
7. r29 – регистр для перевода в ASCII
8. r30 – регистр для для учета напряжений на выходах ЮАРТ
9. R31 – регистр счетчик в функции задержки

# **Структура подпрограмм и их назначения**

**Start** – начало программы, в него происходит переход по прерыванию $000, здесь происходит инициализация стека, портов, необходимых регистров, настройка внешних прерываний и прерываний таймеров

**delay** – подпрограмма, инициализирующая цикл подпрограммы задержки

**long\_delay** – цикл задержки, проверяющий не отсчитал ли таймер заданное количество тиков

**refresh** – подпрограмма для инкремента счетчика, требующегося подпрограмме задержки

**begin\_transmitt** – подпрограмма вызывающая другие подпрограммы для разбиения чисел на разряды

**first/second/third/fourth\_section** – подпрограмма разбивающая число на разряды и передающая подпрограмме вывода в ЮАРТ.

**div1 –** подсчет сотен

**div2/3 –** подсчет десятков

**div4 –** подсчет единиц

**usart\_transmit –** вывод данных во внешний интерфейс.

**EXT\_INT0/INT1 –** прерывания, вызывающиеся в нормальном режиме при сменен фронта на выходах INT0, INT1, смотрит предыдущее состояние выходов

**step\_two/three** – проверяют текущее состояние выходов, сравнивает и при определенных условиях совершает переход в подпрограмму инкремента или декременета

**increment** – увеличение счетчика

**decrement** – увеличение счетчика

**increment** – возвращение в основной цикл

# **Структура программного обеспечения**

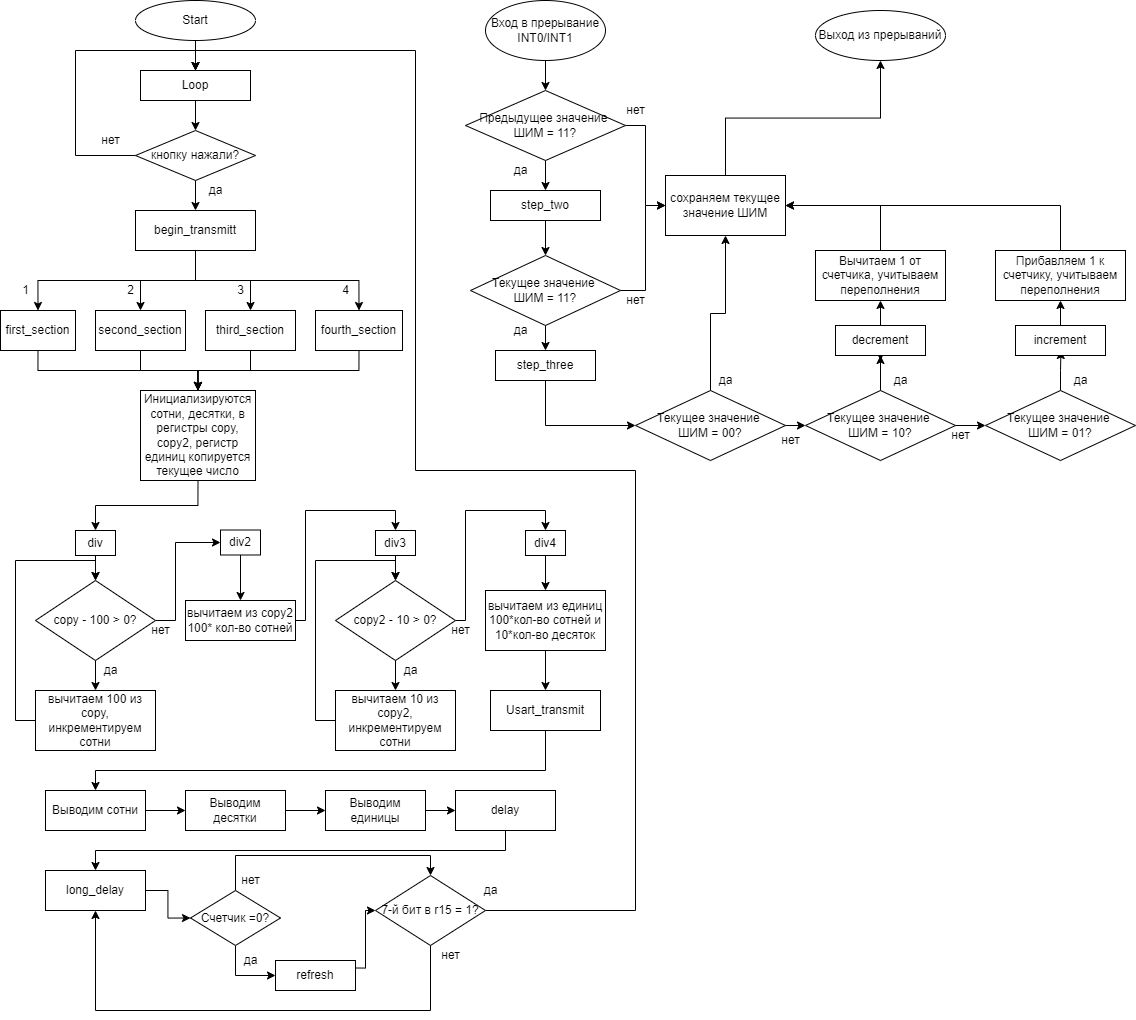


Рис. 5. Структурная схема программы

# **Методика и результаты тестирования**

Тестирование проекта производилось в среде Proteus 8. Для проверки устройства необходимо проверить работоспособность основных функций энкодера:

1. Запускаем программу, не трогаем энкодер и нажимаем кнопку вывода информации

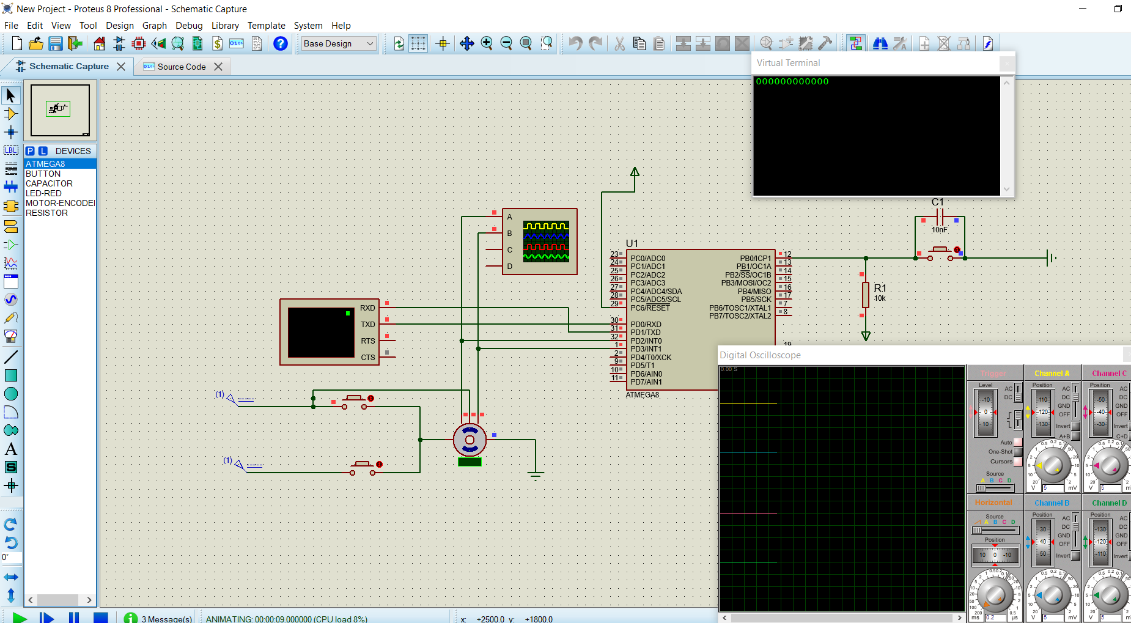


Рис.6. Вывод с энкодера, который никуда не поворачивали

Далее Энкодером подали на мк 3 полных импульса рисунок 7.

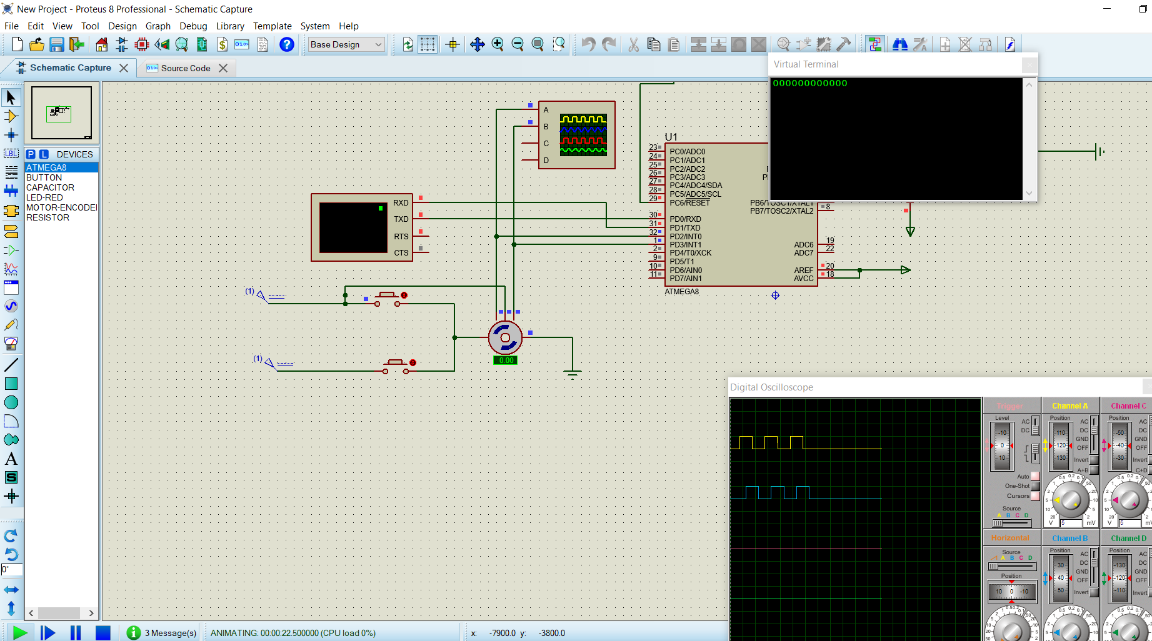


Рис.7. Подали 3 полных импульса

МК корректно их подсчитал рисунок 8.

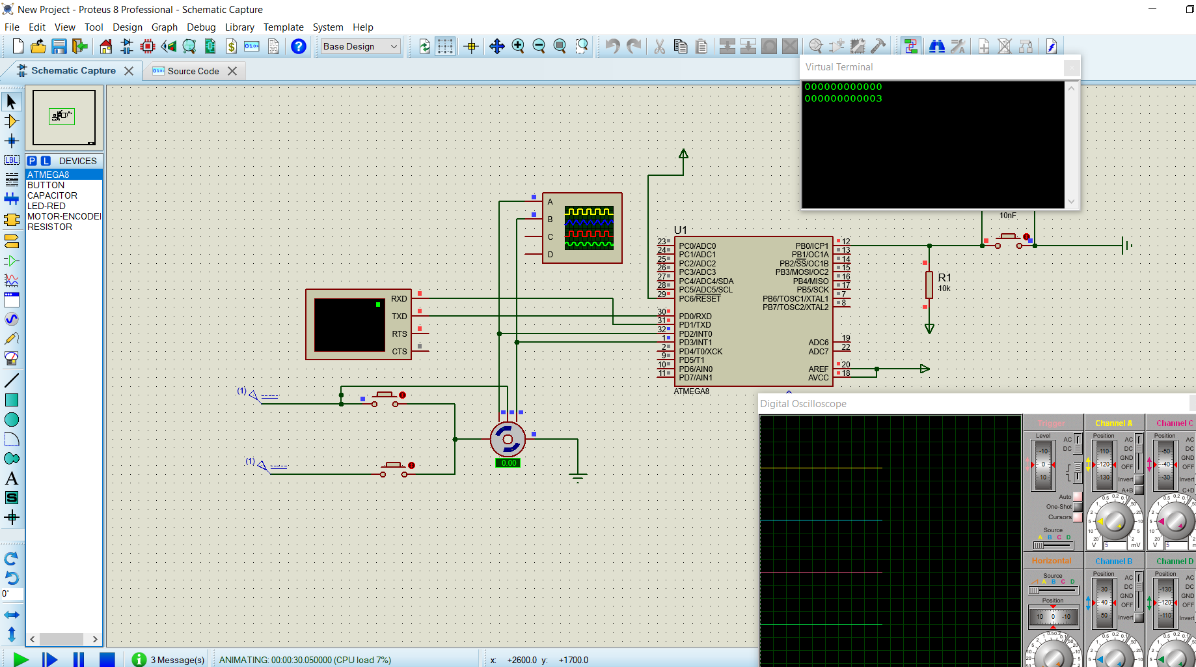


Рис.8. Корректный вывод ЮАРТ

Теперь зажмем зажмем кнопку положительного направления вращения энкодера и посмотрим, что выведет ЮАРТ, рисунки 9, 10.

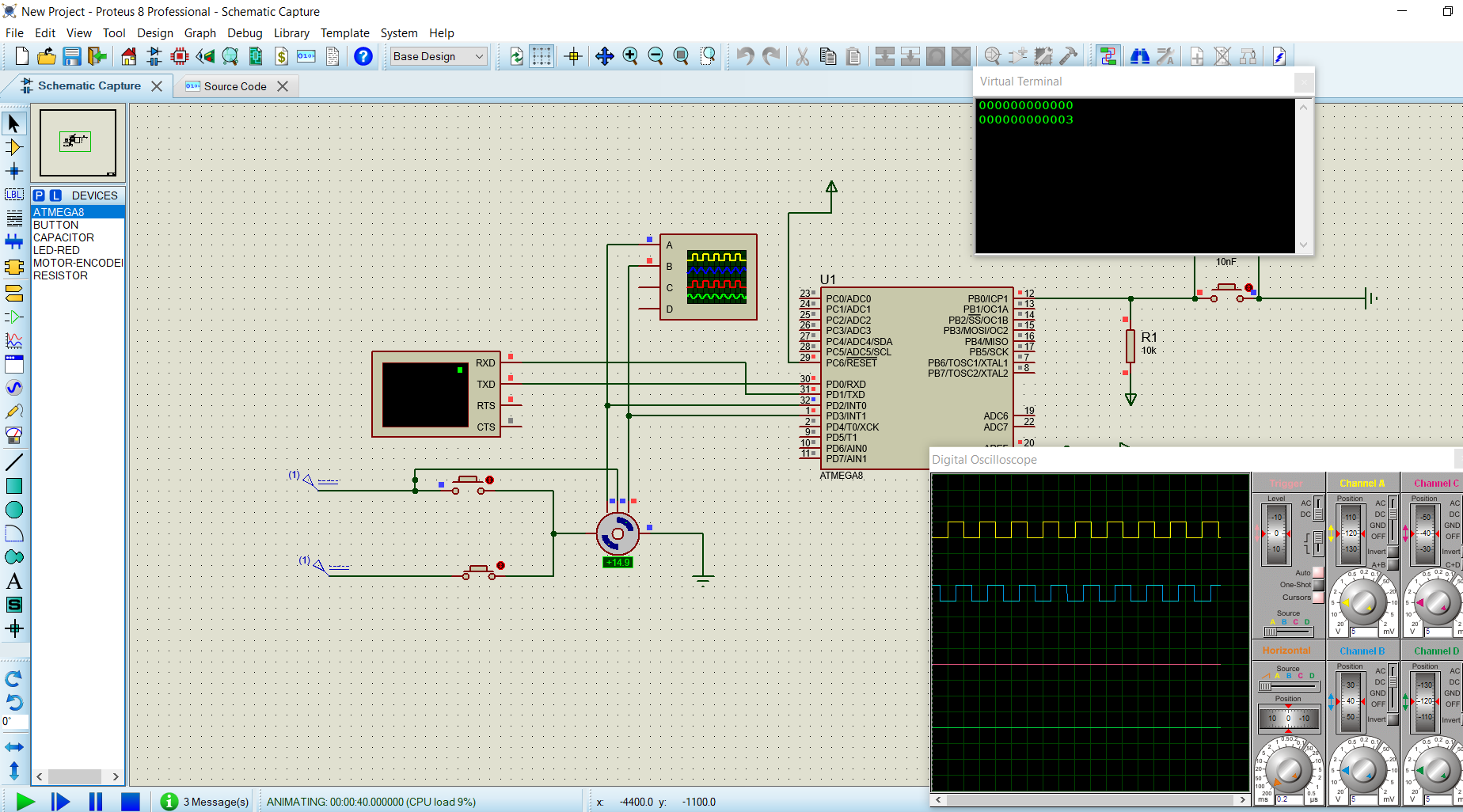


Рис.9. Удерживаем кнопку вращения Энкодера

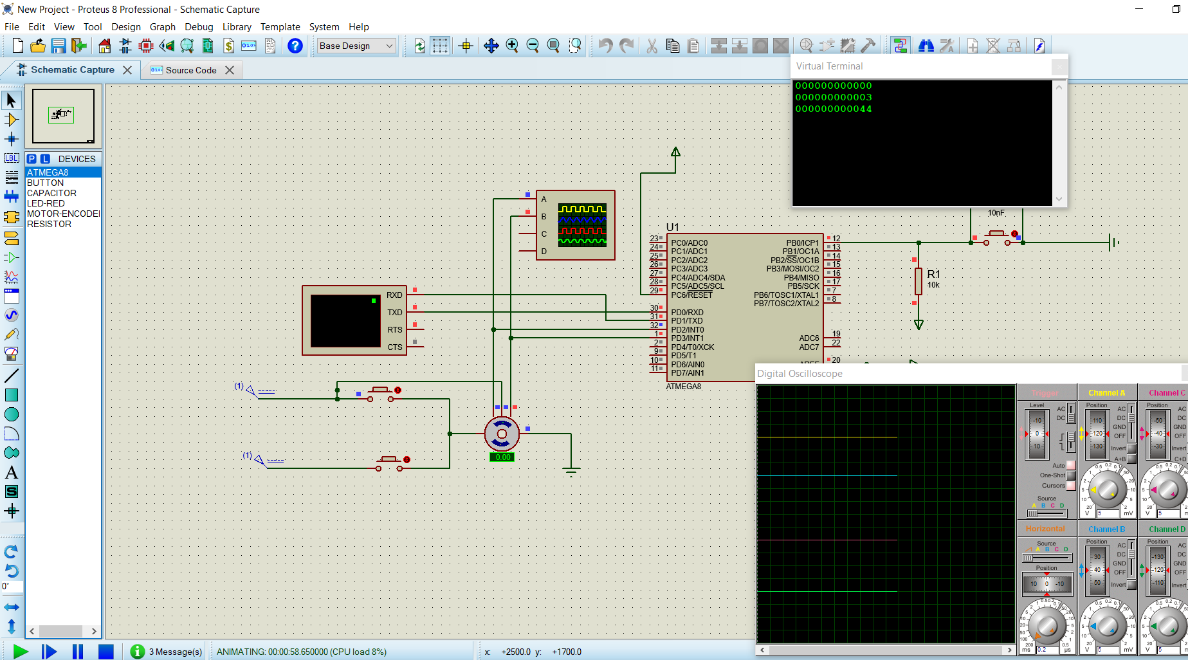


Рис.10. Вывод ЮАРТ

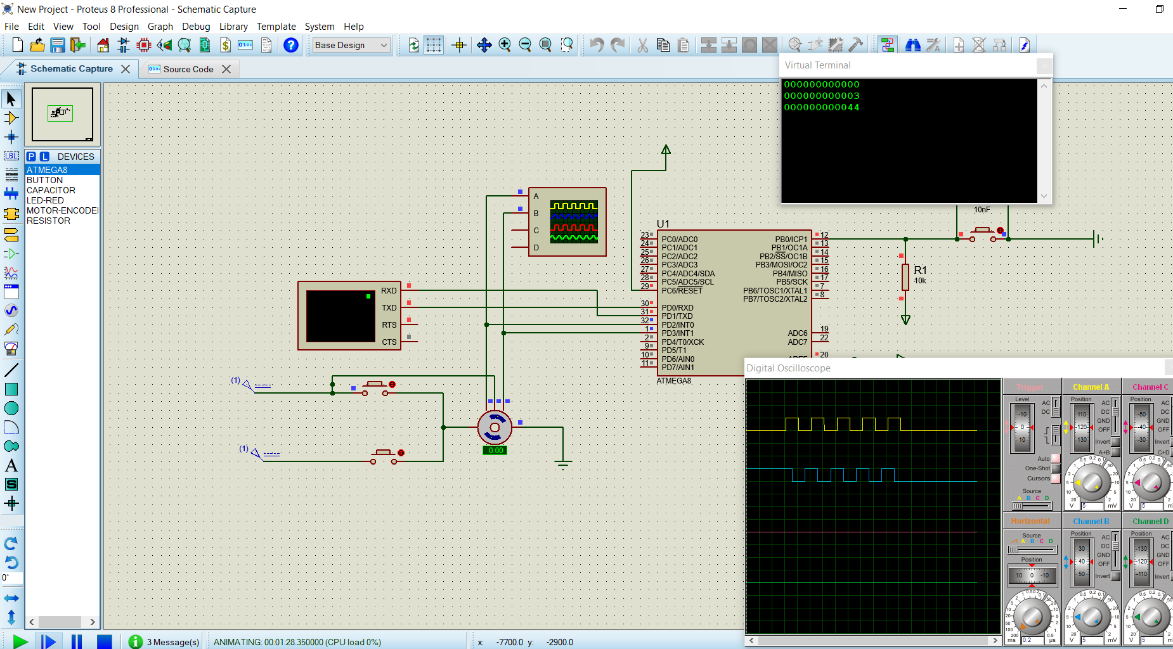
Далее Энкодером подали на мк 5 полных импульсов в отрицательном направлении рисунок11..

Рис.11. Подали 5 обратных импульсов

МК корректно их подсчитал рисунок 12.

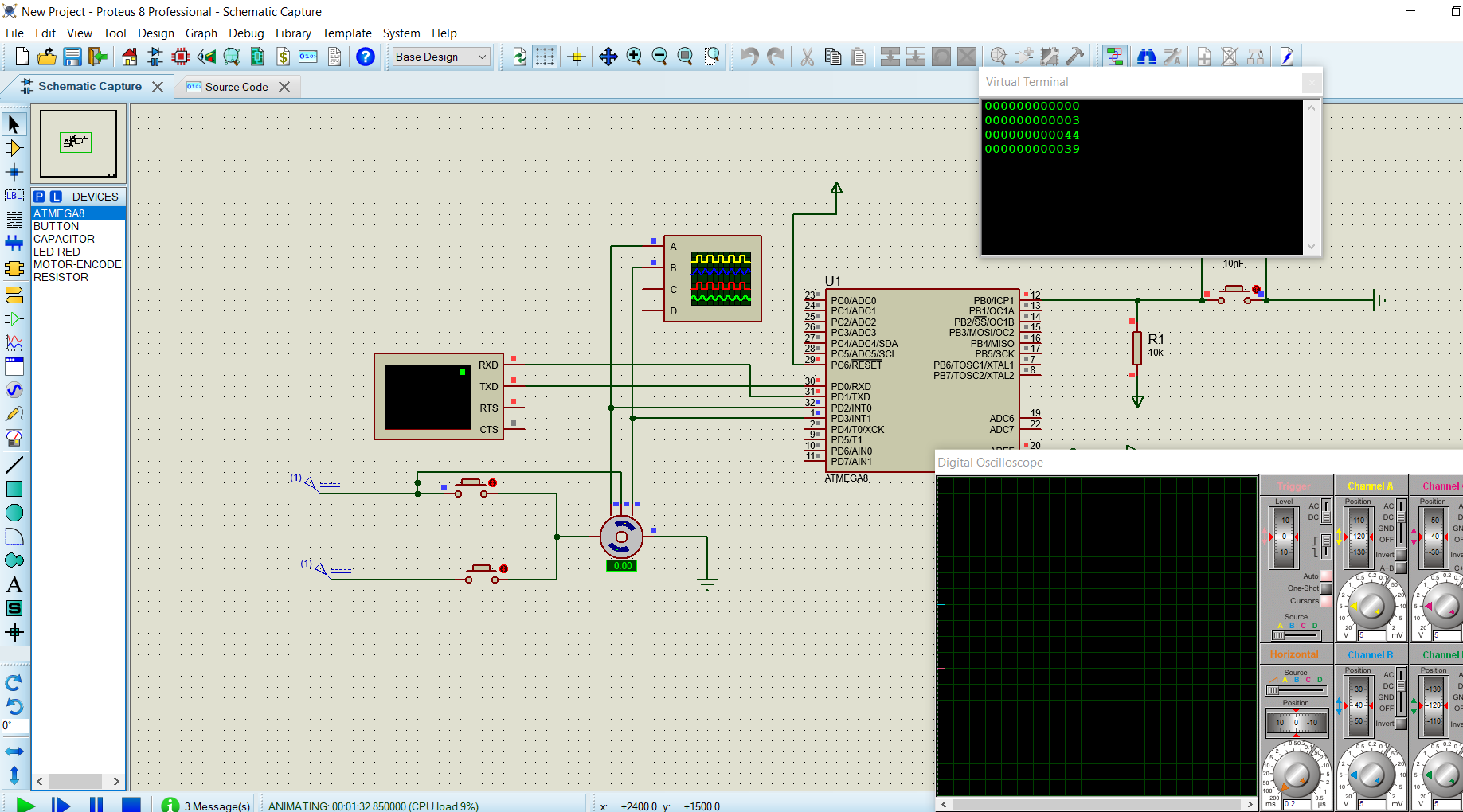
.

Рис.12. Корректный подсчет обратных импульсов

Теперь зажмем кнопку отрицательного направления вращения энкодера и посмотрим, что выведет ЮАРТ, рисунки 13, 14

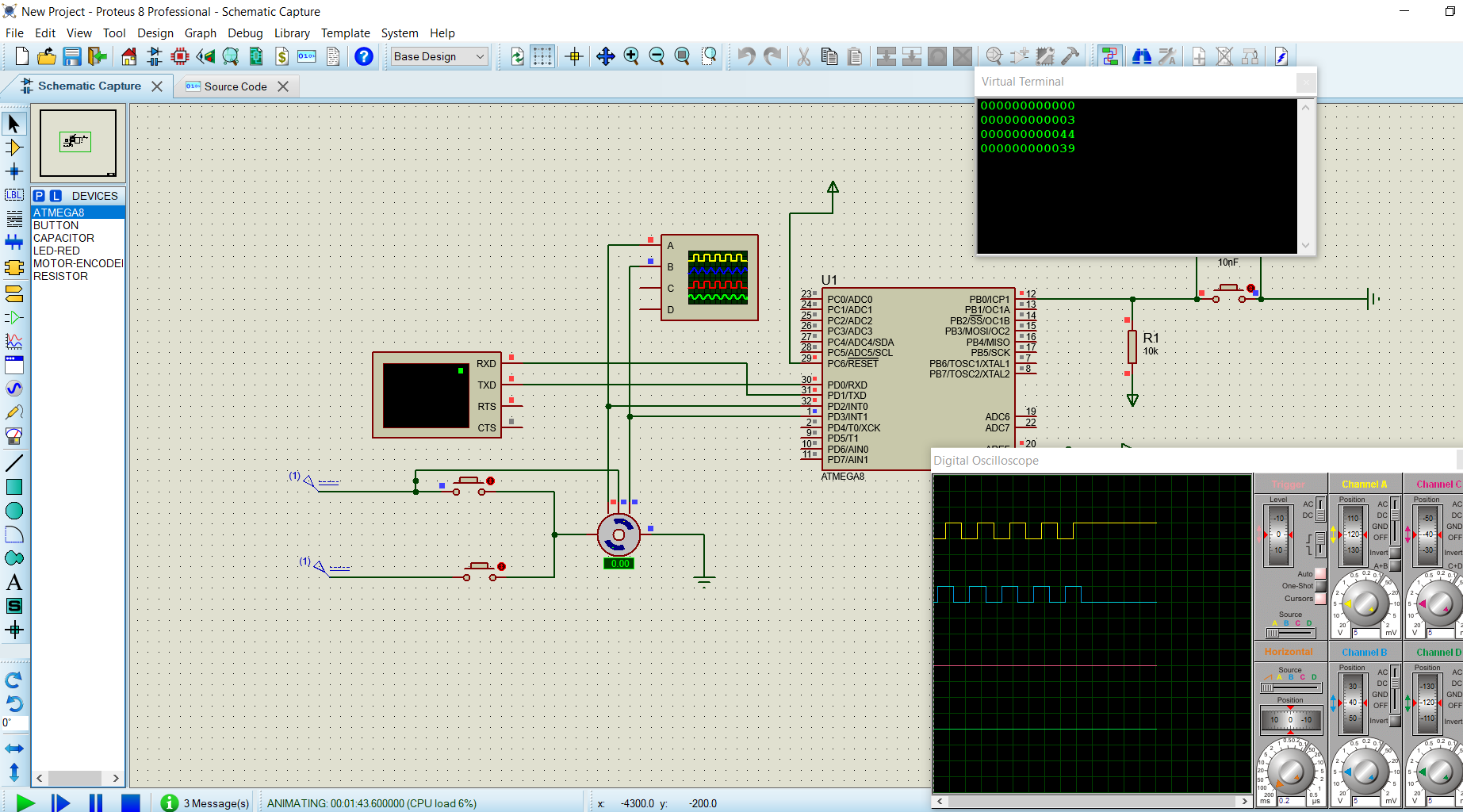
.

Рис.13. Зажали кнопку вращения в обратную сторону

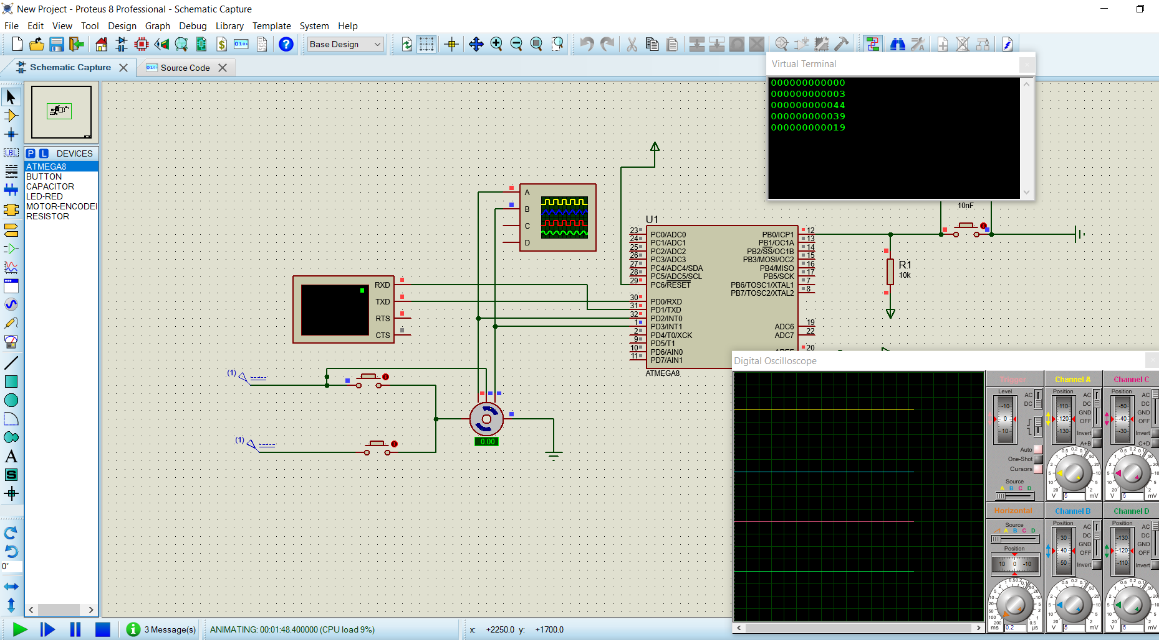


Рис.14. Вывод в ЮАРТ

Также если в нулевой точке начать вращать Энкодер в обратную сторону, можно увидеть что он переполняется и принимает свое максимальное значение (рисунок 15), что связано с отсутствием возможности считать отрицательные такты, предлагается принять что среднее число из всех которые может посчитать мк принять за среднюю точку, и принимать все что выше средней точки за минус, а все что ниже за плюс.

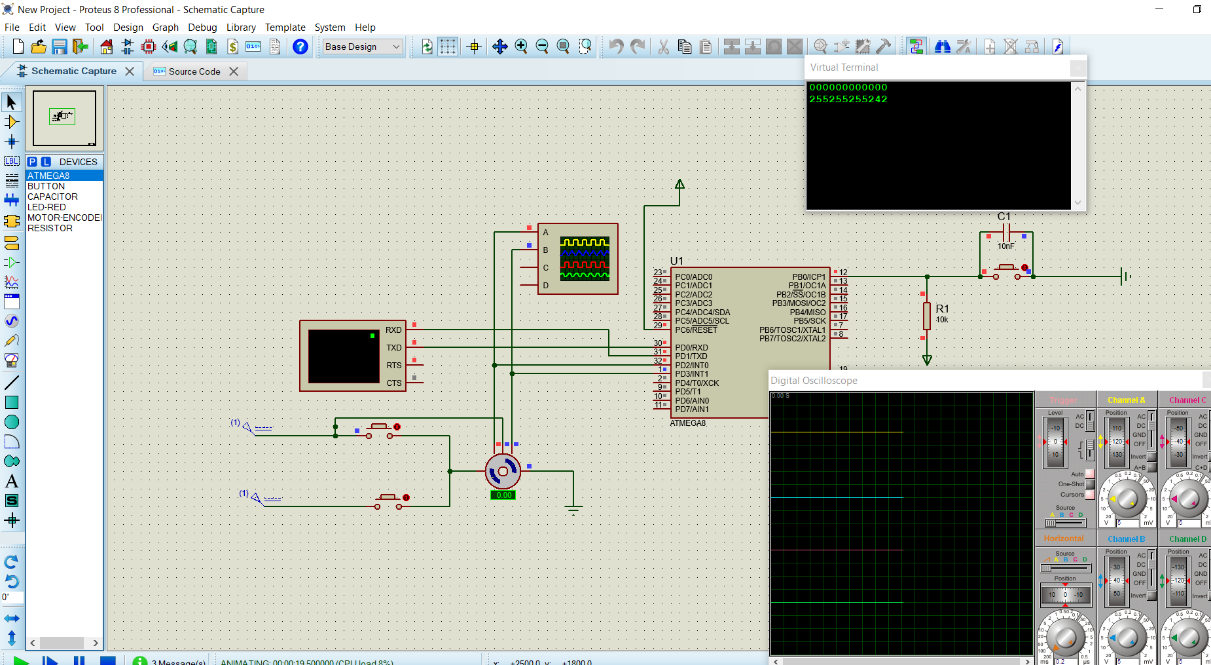


Рис.15. Вращение в обратную сторону с нуля

# **Вывод:**

В результате было разработано аппаратное и программное решение для устройства «Инкрементный счетчик импульсов». Проект включает в себя код написанный на языке программирования Assembler, отвечающий за подсчет оборотов энкодера и выводящий из пользователю на внешний интерфейси собранную в Proteus8 схему, отвечающую за подачу напряжения на соответствующие порты, а также корректную работу внешних устройств. Схема была реализована на основе микроконтроллера ATMEGA16 и прочих внешних устройств.

# **Список использованной литературы:**

1. ATmega8 datasheet [Электронный ресурс]. –URL: <https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2486-8-bit-AVR-microcontroller-ATmega8_L_datasheet.pdf>
2. Studref [Электронный ресурс]. –URL: <https://studref.com/470997/tehnika/registr_upravleniya_statusa_ucsrc_usart_control_status_register>
3. Easy-electronics [Электронный ресурс]. –URL:

http://easyelectronics.ru/avr-uchebnyj-kurs-inkrementalnyj-enkoder.html

1. Cyber-guru [Электронный ресурс]. –URL: <http://www.cyberguru.ru/programming/assembler/assembler-programming-input-output.html?start=1>

# **Листинг программы.**

;====================================================================

; Main.asm file generated by New Project wizard

;

; Created: Пт дек 17 2021

; Processor: ATmega8

; Compiler: AVRASM (Proteus)

;====================================================================

.INCLUDE "m8def.inc"

;====================================================================

; DEFINITIONS

;====================================================================

.DEF first = R20

.DEF second = R21

.DEF third = R22

.DEF fourth = R23

.DEF flag = R18

.DEF flag2 = R19

.DEF hundreds = R24

.DEF dozens = R25

.DEF ones = R26

.DEF copy = R27

.DEF copy2 = R28

;====================================================================

; VARIABLES

;====================================================================

;====================================================================

; RESET and INTERRUPT VECTORS

;====================================================================

.ORG 0x000

rjmp Start

.ORG 0x001

rjmp EXT\_INT0

.ORG 0x002

rjmp EXT\_INT1

.CSEG

;====================================================================

; CODE SEGMENT

;====================================================================

Start:

ldi R16, LOW(RAMEND)

out SPL, R16

ldi R16, HIGH(RAMEND)

out SPH, R16

ldi R28, (1 << U2X)

out UCSRA, R28 ; Удвоенная скорость

ldi R28, 12 ; Скорость = 9600 бит/с при такте 1 МГц

out UBRRL, R28

ldi R28, (1 << RXEN)|(1 << TXEN)

out UCSRB, R28 ; Разрешаем приемник, передатчик и прерывание по приему

ldi R28, (1 << URSEL)|(1 << UCSZ1)|(1 << UCSZ0)

out UCSRC, R28

clr R28

ldi R16, 0b11000000 ;Разрешаем прерывания кнопок 0, 1

out GICR, R16

ldi R16, 0b00000101

out MCUCR, R16

ldi first, 0b00000000

ldi second, 0b00000000

ldi third, 0b00000000

ldi fourth, 0b00000000

ldi R16, 0b00000000

ldi R17, 0b00000001

ldi R30, 0b00001100

in flag, PIND

and flag, R30

ldi flag2, 0b00000000

sei

Loop:

sbis PINB, 0

rcall begin\_transmitt

rjmp Loop

delay:

ldi R31, 0x00

mov r15, R31

ldi R31, 0xff

rjmp long\_delay

long\_delay:

dec R31

cpi R31, 0

breq refresh

sbrc r15, 7

ret

rjmp long\_delay

refresh:

mov R31, r15

inc R31

mov r15, R31

ldi R31, 0xff

rjmp long\_delay

begin\_transmitt:

rcall first\_section

rcall second\_section

rcall third\_section

rcall fourth\_section

rcall delay

rcall delay

ret

;Вывод числа

Usart\_Transmit:

sbis UCSRA, UDRE

rjmp PC - 1

out UDR, hundreds

sbis UCSRA, UDRE

rjmp PC - 1

out UDR, dozens

sbis UCSRA, UDRE

rjmp PC - 1

out UDR, ones

sbis UCSRA, UDRE

rjmp PC - 1

out UDR, R29

ret

div:

subi copy, 100

brcs div2

inc hundreds

rjmp div

div2:

ldi R29, 0x64

mul hundreds, R29

sub copy2, R0

rjmp div3

div3:

subi copy2, 10

brcs div4

inc dozens

rjmp div3

div4:

ldi R29, 0x64

mul hundreds, R29

sub ones, R0

ldi R29, 10

mul dozens, R29

sub ones, R0

ret

first\_section:

mov copy, first

mov copy2, first

mov ones, first

ldi hundreds, 0x00

ldi dozens, 0x00

rcall div

ldi R29, 0b00110000

or hundreds, R29

or dozens, R29

or ones, R29

ldi R29, 0x00

rcall Usart\_Transmit

ret

second\_section:

mov copy, second

mov copy2, second

mov ones, second

ldi hundreds, 0x00

ldi dozens, 0x00

rcall div

ldi R29, 0b00110000

or hundreds, R29

or dozens, R29

or ones, R29

ldi R29, 0x00

rcall Usart\_Transmit

ret

third\_section:

mov copy, third

mov copy2, third

mov ones, third

ldi hundreds, 0x00

ldi dozens, 0x00

rcall div

ldi R29, 0b00110000

or hundreds, R29

or dozens, R29

or ones, R29

ldi R29, 0x0

rcall Usart\_Transmit

ret

fourth\_section:

mov copy, fourth

mov copy2, fourth

mov ones, fourth

ldi hundreds, 0x00

ldi dozens, 0x00

rcall div

ldi R29, 0b00110000

or hundreds, R29

or dozens, R29

or ones, R29

ldi R29, 0x0D

rcall Usart\_Transmit

ret

;====================================================================

EXT\_INT0:

in flag2, PIND

and flag2, R30

cpi flag, 0b00001100

breq step\_two

brne back

step\_two:

cpi flag2, 0b00001100

breq back

brne step\_three

step\_three:

cpi flag2, 0b00000000

breq back

cpi flag2, 0b00000100

breq decrement

cpi flag2, 0b00001000

breq increment

reti

increment:

add fourth, r17

adc third, r16

adc second, r16

adc first, r16

mov flag, flag2

reti

decrement:

sub fourth, r17

sbc third, r16

sbc second, r16

sbc first, r16

mov flag, flag2

reti

back:

mov flag, flag2

reti

EXT\_INT1:

in flag2, PIND

and flag2, R30

cpi flag, 0b00001100

breq step\_two

brne back