**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Кубанский Государственный Университет**

Кафедра информационных технологий

ОТЧЕТ

о выполнении лабораторной работы №5

по дисциплине «Системы реального времени»

Выполнил:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ст. гр. 44 Иванов А.А.\_\_\_\_\_\_\_

Проверил:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

доц. каф. ИТ Полетайкин А.Н.

Краснодар

2024

**Организация обмена между датчиками, УВМ и исполнительными устройствами**

Цель: изучение принципов организации инфообмена между ядром СРВ и периферийными устройствами; приобретение практических навыков разработки функциональных схем, алгоритмов и управляющих программ для управления технологическими процессами с использованием измерительных преобразователей и исполнительных устройств.

**Задание**

Задан технологический процесс или функционирование объекта управления, контроль за состоянием которого осуществляется с помощью аналоговых и дискретных датчиков. Для аналоговых датчиков определены допустимый диапазон измерений, диапазон выходного сигнала, инерционность и интервал или режим опроса. Известен АЦП, преобразующий входную аналоговую информацию в двоичный код. В соответствии с измеряемыми значениями необходимо выдать аналоговое управляющее воздействие на исполнительные устройства, регулирующие протекание технологического процесса или состояние объекта управления. Для вывода аналоговой информации используется ЦАП. Разрядности АЦП и ЦАП, диапазоны аналоговых сигналов и максимальное время преобразования ЦАП заданы в таблице. Варианты технологических процессов приведены в приложении. По согласованию с преподавателем технологический процесс может быть скорректирован.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № вар. | Разрядность АЦП | Разрядность ЦАП | Диапазон напряжения, В | | Макс. время преобразования ЦАП, мс |
| Входной сигнал | Выходное воздействие |
| 31 |  | 10 |  | 0…300 мА |  |

Таблица 1 – варианты выбора характеристик преобразователей

Порядок выполнения:

1. Сформировать список устройств для устройства связи с объектом (УСО).

2. Построить градуировочную характеристику для каждого аналогового датчика (при наличии).

3. Рассчитать двоичные эквиваленты контрольных и управляющих непрерывных величин.

4. Определить количество, разрядность и назначение портов ввода-вывода.

5. Определить назначение отдельных разрядов портов.

6. Разработать структурную схему интерфейса связи.

7. Разработать алгоритм функционирования объекта.

8. Рассчитать параметры процедуры временной задержки (при необходимости).

9. Составить программу управления (драйвер) на языке ассемблера.

10. На языке высокого уровня разработать приложение, в графическом режиме имитирующее функционирование СРВ. Обеспечить индикацию контрольных и управляющих величин. Предусмотреть возможность оперативного изменения значений контрольных параметров объекта. Масштаб времени может быть любым.

**Заданный технологический процесс**

31. Управление газорезкой МНЛЗ. Объект управления – газорежущее устройство, предназначенное для отрезания заготовки определённой длины при ручьевой разливке стали на МНЛЗ (машина непрерывного литья заготовок).

Литьё стальных заготовок осуществляется непрерывным способом посредством разливки стали из ковша параллельно по нескольким ручьям. На входе каждого ручья расположен кристаллизатор, который интенсивно охлаждается водой и обеспечивает первичное затвердевание слитка. Далее слиток охлаждается открытой струёй, после чего режется на заготовки одинаковой длины.

При выходе из системы охлаждения непрерывный слиток необходимо разрезать на заготовки заданной длины. Учёт длин отрезаемых заготовок осуществляет внешняя подсистема расчёта длины заготовки (ПРДЗ).

На резак посредством дискретно управляемых клапанов подаётся газ и кислород. Резак перемещается поперёк ручья под управлением дискретного электроприводного устройства (предусмотрено два дискретных управляющих сигнала для движения резака вперёд и назад).

Резак монтируется на тележке, которая может перемещаться вдоль ручья в направлении исходной позиции за счёт работы дискретно управляемого электропривода. В исходной позиции тележки и в позиции максимального удаления расположены концевые выключатели (КВ) КВ0 и КВ1. Перемещение тележки в исходную позицию завершается при срабатывании КВ0.

Также на тележке смонтировано механическое устройство захвата, которое обеспечивает фиксацию резака на слитке. Таким образом реализуется перемещение тележки вперёд для резки слитка. Захват приводится в действие соленоидным клапаном, который управляется постоянным током согласно графикам, показанным на рисунке 1:

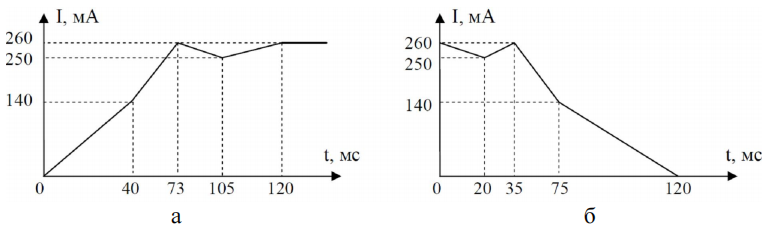


Рисунок 1 – управляющее воздействие соленоидного клапана захвата заготовки (а – захват, б – отпускание)

На пульте оператора имеется главная аварийная кнопка, по нажатию на которую работа объекта полностью блокируется: резак выключается, захват отпускается.

Исходное состояние объекта: захват отпущен, клапаны закрыты, резак и тележка в исходных позициях.

Цикл резки начинается, когда от ПРДЗ поступает разрешающий сигнал. Разрешением для резки также является выполнение следующих условий:

– главная аварийная кнопка не нажата;

– давление газа и кислорода выше минимального.

После включения захвата включается резак и начинает движение поперёк ручья. Время резки T определяется по формуле:

T = (L+30)/V\_c

где L – заданная длина стороны квадратного профиля или диаметр круглого профиля заготовки, V\_c = 25 мм/с – скорость движения резака.

За 1.5 с до конца резки выдаётся разрешение на запуск рольганга для транспортировки заготовки на перекладыватель. По окончанию резки объект возвращается в исходное состояние.

Если в процессе резки тележка доезжает до позиции максимального удаления, что фиксируется срабатыванием КВ1, то процесс прерывается и объект возвращается в исходное состояние. Загорается лампочка «Заготовка не отрезана».

Если в процессе резки дискретные датчики давления фиксируют падение давления газа или кислорода, то процесс прерывается и объект возвращается в исходное состояние. Загорается лампочка «Заготовка не отрезана».

**Ход работы**

**1 Сформированный список устройств для устройства связи с объектом**

Список устройств для УСО:

1. Датчики:

*1. Датчики давления газа и кислорода (дискретные):* они определяют, что давление находится выше минимального уровня; дают дискретный выходной сигнал (вкл/выкл).

*2. Концевые выключатели КВ0 и КВ1 (дискретные):* КВ0 определяет, что тележка находится в исходной позиции; КВ1 фиксирует достижение максимального удаления тележки; дают дискретный выходной сигнал (вкл/выкл).

2. Исполнительные устройства:

*1. Соленоидный клапан захвата (аналоговый):* управляет фиксацией и отпусканием резака на заготовке; управляющий сигнал – аналоговый (по графику управляющего тока).

*2. Клапаны газа и кислорода (дискретные):* управляют подачей газов в резак; управляющий сигнал – дискретный (вкл/выкл).

*3. Электроприводы резака и тележки (дискретные):* привод резака управляет движением поперёк ручья; управляющий сигнал – дискретный (вперёд/назад). Привод тележки управляет перемещением вдоль ручья; управляющий сигнал – дискретный (вперёд/назад).

3. Устройства контроля:

*1. Лампа индикации «Заготовка не отрезана» (дискретная):* загорается при аварийной ситуации (срабатывание КВ1 или падение давления); управляющий сигнал – дискретный (вкл/выкл).

4. Сигналы и интерфейсы управления:

*1. ПРДЗ (подсистема расчёта длины заготовки, дискретная):* передаёт разрешающий сигнал для начала резки; входной сигнал – дискретный (вкл/выкл).

*2. Главная аварийная кнопка (дискретная):* полностью блокирует работу системы; входной сигнал – дискретный (вкл/выкл).

5. Устройства преобразования и управления:

*1. ЦАП (цифро-аналоговый преобразователь):* обрабатывает входные двоичные сигналы и выдаёт аналоговое выходное представление входных данных (генерирует силу тока (Амперы)).

*2. Микроконтроллер:* центральное устройство управления, которое принимает данные от датчиков, формирует управляющие сигналы и обеспечивает логику работы.

**2 Градировочная характеристика соленоидного захвата**

Имеются следующие графики управления захвата (Рисунок 2.1):

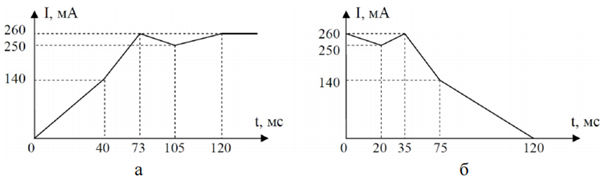


Рисунок 2.1 – управляющее воздействие соленоидного клапана захвата заготовки (а – захват, б – отпускание)

Выделим ключевые точки для построения линейных участков:

1. График захвата (а):

1. Начальная точка: t = 0 мс, I = 0 мА

2. Вершина 1: t = 40 мс, I = 140 мА

3. Вершина 2: t = 73 мс, I = 260 мА

4. Вершина 3: t = 105 мс, I = 250 мА

5. Конечная точка: t = 120 мс, I = 260

2. График отпускания (б):

1. Начальная точка: t = 0 мс, I = 260 мА

2. Вершина 1: t = 20 мс, I = 250 мА

3. Вершина 2: t = 35 мс, I = 260 мА

4. Вершина 3: t = 75 мс, I = 140 мА

5. Конечная точка: t = 120 мс, I = 0 мА

По этим точкам можно аппроксимировать каждый график для получения формул линейной зависимости, по которым получится примерно одинаково считать выходные данные от входных значений.

Для этого воспользуемся библиотекой scipy для Python (код аппроксимаций представлен в файле approc.py). Используем имеющиеся точки линейных участков, в результате чего получим такой график линейных зависимостей (Рисунок 2.2):

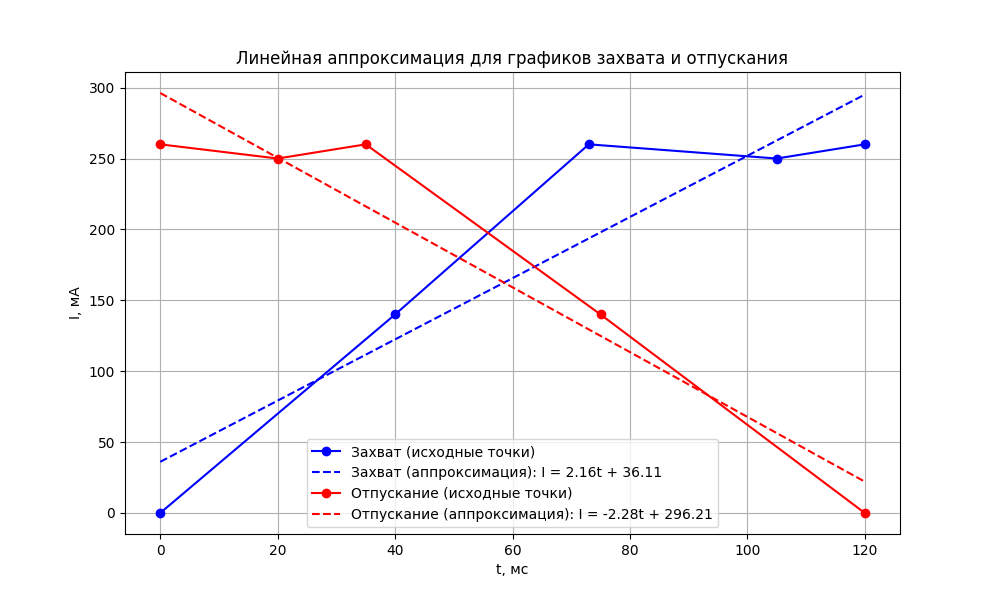


Рисунок 2.2 – аппроксимация графиков захвата и отпускания

Таким образом мы получили формулы для расчёта:

1. Захват: I = 2.16 \* t + 36.11

2. Отпускание: I = -2.28 \* t + 296.21

**3 Двоичные эквиваленты контрольных и управляющих непрерывных величин**

Сначала определим, что конкретно нужно рассчитать: *контрольные величины* – параметры, которые изменяются в процессе работы системы и используются для управления, то есть это входные данные; *управляющие величины* – это параметры, которые генерирует система для воздействия на исполнительные механизмы, то есть это выходные данные.

Произведём расчёт двоичных эквивалентов амперных величин тока для ЦАП. Формула для расчёты используется следующая: D = ROUND[((u-u\_1)/(u\_2-u\_1))\*2^n], где u – входной сигнал, u1 и u2 – предельные значения преобразования (0 и 300 мА), n – разрядность устройства (10).

Код расчёта величин представлен в файле calc\_tsap.py.

Результаты расчётов представлен ниже (Таблица 3.1):

|  |  |  |
| --- | --- | --- |
| I, мА | D, десятичная система | D, шестнадцатеричная система |
| 0 | 0 | 0h |
| 140 | 478 | 1DEh |
| 250 | 853 | 355h |
| 260 | 887 | 377h |
| 300 | 1024 | 400h |

Таблица 3.1 – кодировка амперной величины для ЦАП

Значения, представленные выше (Таблица 3.1) показывают, какие двоичные данные будет получать ЦАП для преобразования в ток соответствующей силы.

**4-5 Количество, разрядность и назначение портов ввода-вывода (а также отдельных разрядов)**

По имеющемуся списку датчиков и устройств можно использовать такую схему портов ввода-вывода:

1. 16-битный порт ввода (необходимые разряды):

1. Разряд 0: ПРДЗ (сигнал начала цикла резки);

2. Разряд 1: Главная аварийная кнопка (проверка);

3. Разряд 2: Концевой выключатель КВ0;

4. Разряд 3: Концевой выключатель КВ1;

5. Разряд 4: Датчик давления газа;

6. Разряд 5: Датчик давления кислорода;

7. Разряды 6-15: Зарезервированы (не нужны).

2. 16-битный порт вывода (необходимые разряды):

1. Разряд 0: Лампа индикации;

2. Разряд 1: Электропривод тележки (движение вперёд);

3. Разряд 2: Электропривод тележки (движение назад);

4. Разряд 3: Электропривод резака (движение вперёд);

5. Разряд 4: Электропривод резака (движение назад);

6. Разряд 5: Клапан газа;

7. Разряд 6: Клапан кислорода;

8. Разряд 7: Разрешение для рольганга;

8. Разряды 8-15: Зарезервированы (не нужны).

Итог:

Всего нужно портов: 1 порт ввода (16-разрядный), у которого используются 0-5 разряды; 1 порта вывода (16-разрядный), у которого используются 0-7 разряды.

**6 Структурная схема интерфейса связи**

Текстовое описание связи всех компонентов:

1. Данные от ПРДЗ (0), аварийной кнопки (1), КВ0 (2), КВ1 (3), датчика давления газа (4), датчика давления кислорода (5) напрямую передаются в порт ввода (в соответствующие разряды), т.к. все эти сигналы – дискретные;

2. Микроконтроллер через шину получает данные из порта ввода и производит все вычисления;

3. Микроконтроллер через шину передаёт данные в порт вывода, к которому подключены прочие дискретные устройства, которые считывают данные так-же с определённых разрядов: лампа индикации (0), электропривод тележки (вперёд, 1), электропривод тележки (назад, 2), электропривод резака (вперёд, 3), электропривод резака (назад, 4), клапан газа (5), клапан кислорода (6), рольганг, с которым мы напрямую не работаем (7);

4. Микроконтроллер передаёт двоичное представление нужной силы тока в ЦАП;

5. ЦАП преобразует двоичные данные в аналоговый сигнал, который передаёт напрямую в соленоидный клапан захвата через шину.

Графически вышеописанный интерфейс связи представлен на рисунке (Рисунок 6.1):

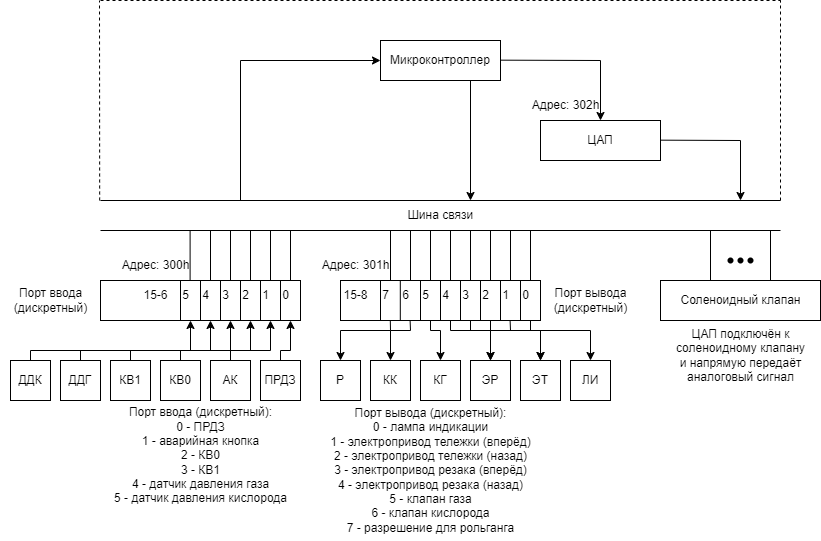


Рисунок 6.1 – интерфейс связи

**7 Алгоритм функционирования объекта**

Описание общей структуры алгоритма функционирования объекта:

1. Инициализация: указать адреса портов ввода-вывода; рассчитать время резки Т исходя из константных параметров L (в задаче не сказано, что этот параметр может варьироваться) и V\_c; установить статус объекта как «ожидание».

2. Ожидание разрешающего сигнала от ПРДЗ: при получении разрешающего сигнала (1) от ПРДЗ проверить: аварийная кнопка не нажата; давление газа и кислорода в норме (1 и 1); КВ0 активен (1). После выполнения всех проверок приступить к выполнению захвата заготовки.

3. Захват заготовки: подавать определённую силу тока на протяжении определённого кол-ва временных шагов; проверять, что аварийная кнопка не нажата (0), давление газа и кислорода в норме (1 и 1), КВ0 активен (1).

4. Выполнение резки: проверка аварийной кнопки (0), давления газа и кислорода (1 и 1), КВ0 не активен (0), КВ1 не активен (0); запуск движения резака вперёд, включить подачу газа и кислорода; за 1.5 секунды до конца резки подать разрешающий сигнал рольгангу; если тележка достигла КВ1 или давление газа и кислорода упали – включить лампу «Заготовка не отрезана», вернуть систему в исходное положение.

5. Окончание резки: отключить подачу газа и кислорода, выполнить отпускание заготовки (подавать определённую силу тока на протяжении определённого кол-ва временных шагов); выполнить возвращение резака (т.к. датчики положения резака отсутствуют – выполнить перемещение «вслепую» (по времени, аналогичному резке, перемещать резак назад)) и тележки (до срабатывания КВ0) в исходное положение.

6. Постоянная проверка на аварийную ситуацию: если нажата аварийная кнопка, в процессе резки сработал КВ1 или упало давления газа или кислорода – остановить все процессы (прекратить подачу газа и кислорода, выполнить отпускание заготовки) и вернуть объект в исходное состояние (тележка и резак в исходных положениях).

Более наглядно весь вышеописанный алгоритм представлен на рисунке (Рисунок 7.1):

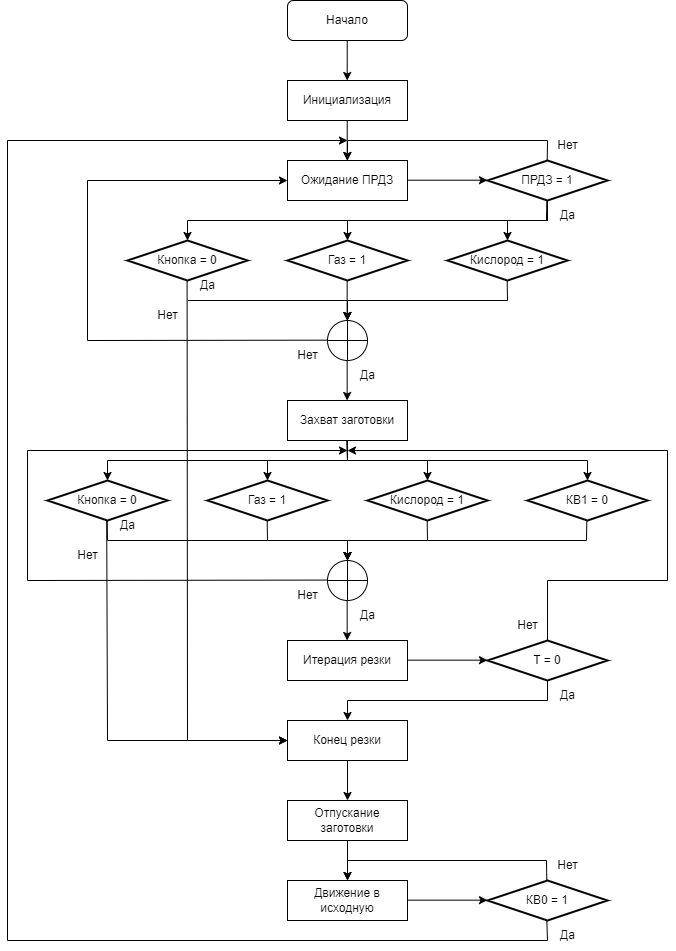


Рисунок 7.1 – блок-схема алгоритма

В вышепоказанной блок-схеме упрощено отображение задержки в 1 мс между итерациями захвата, резки, отпускания, однако сам алгоритм будет реализовывать такие задержки.

Задержки сделаны по одной причине: при циклическом опросе датчиков и высчитывании всех управляющих величин система будет тратить значительно меньше времени, чем 1 мс, что очень важно при произведении захвата, резки и отпускания заготовки (каждый из этих процессов длится строго определённое время и должен получать строго определённые управляющие данные для корректной работы). Так что решено игнорировать ту временную величину, которую система будет тратить на высчитывание и передачу данных на порты, после чего будет вызываться миллисекундная задержка для корректной работы в цикле.

**8 Параметры процедуры временной задержки**

Как уже упоминалось в предыдущем пункте, для корректной работы с процессами, занимающими определённое время (захват, резка, отпускание), решено использовать подпрограмму задержки, которая будет вызывать задержку ровно в 1 миллисекунду. Код такой подпрограммы представлен ниже:

\_delay:

push ebx ;15 тактов

push ecx ;15 тактов

pushf ;14 тактов

mov cx, 100 ;4 такта

first\_cycle:

mov bx, 1189 ;4 такта

second\_cycle:

nop ;3 такта

dec bx ;2 такта

jnz second\_cycle ;16/4 такта

loop first\_cycle ;17/5 тактов

mov bx, 117 ;4 такта

third\_cycle:

dec bx ;2 такта

jnz third\_cycle ;16/4 такта

nop ;3 такта

nop ;3 такта

dec bx ;2 такта

dec bx ;2 такта

popf ;12 тактов

pop ecx ;12 тактов

pop ebx ;12 тактов

ret ;20 тактов

Формула расчёта количества тактов представлена в файле delay.py.

Сам перевод тактов в миллисекунды исходит из тактовой частоты используемого вычислительного процессора: 2.50 ГГц -> 2 500 000 000 тактов в секунду = 2 500 000 тактов в миллисекунду.

Таким образом, вышеописанная задержка будет вызываться каждый раз после вычисления управляющего воздействия для итерации, например, захвата. Это выполняется потому, что процесс захвата длится 120 миллисекунд, и в каждую конкретную миллисекунду захват должен получать определённую аналоговую величину силы тока. Однако расчёт нужной величины силы тока по формуле и передача управляющего сигнала происходит значительно быстрее, чем за 1 миллисекунду. Таким образом, для «синхронизации» микроконтроллера с процессами захвата, резки, отпускания после каждой итерации будет вызываться задержка.

При этом, как уже упоминалось, из-за быстрой скорости расчёта микроконтроллера можно пренебречь тем количеством тактов, которое тратит микроконтроллер на вычисление управляющих величин.

**9 Программа управления (драйвер) на языке ассемблера**

Код программы управления (драйвера) представлен ниже:

.486

.model flat

.data

port\_IN DW 300h

port\_OUT DW 301h

address\_TSAP DW 302h

catch\_release\_T DB 120

cutting\_T DW 5000

cutting\_rolgang\_T DW 3500

system\_status DW 0

cur\_iter Dw 0

param\_1 DD 2.16

param\_2 DD 36.11

param\_3 DW ?

param\_4 DD -2.28

param\_5 DD 296.21

res\_calc DW ?

.code

\_delay:

push ebx

push ecx

pushf

mov cx, 100

first\_cycle:

mov bx, 1189

second\_cycle:

nop

dec bx

jnz second\_cycle

loop first\_cycle

mov bx, 117

third\_cycle:

dec bx

jnz third\_cycle

nop

nop

dec bx

dec bx

popf

pop ecx

pop ebx

ret

\_check\_PRDZ:

mov bx, ax

and bx, 0000000000000001b

ret

\_check\_emergency\_button:

mov bx, ax

and bx, 0000000000000010b

ret

\_check\_KV0:

mov bx, ax

and bx, 0000000000000100b

ret

\_check\_KV1:

mov bx, ax

and bx, 0000000000001000b

ret

\_check\_gas\_pressure:

mov bx, ax

and bx, 0000000000010000b

ret

\_check\_oxygen\_pressure:

mov bx, ax

and bx, 0000000000100000b

ret

\_start:

MAIN\_CYCLE:

xor ecx, ecx

mov dx, port\_IN

;in ax, dx

mov ax, 0000000000110101b

mov bx, system\_status

cmp bx, 0

je WAITING

cmp bx, 1

je CATCH

cmp bx, 2

je CUTTING

cmp bx, 3

je RELEASE

cmp bx, 4

je SOURCE

cmp bx, 5

je RELEASE\_LAMP

cmp bx, 6

je SOURCE\_LAMP

WAITING:

call \_check\_emergency\_button

jnz BLOCKING

call \_check\_PRDZ

jnz PRDZ\_true

mov dx, port\_OUT

mov ax, 0000000000000000b

;out dx, ax

jmp AGAIN

PRDZ\_true:

mov bx, 1

mov system\_status, bx

mov bx, 0

mov cur\_iter, bx

jmp CATCH

CATCH:

call \_check\_emergency\_button

jnz BLOCKING

mov bx, cur\_iter

cmp bx, 120

ja catch\_true

mov param\_3, bx

fld dword ptr [param\_1]

fimul word ptr [param\_3]

fadd dword ptr [param\_2]

fistp word ptr [res\_calc]

inc bx

mov cur\_iter, bx

mov dx, address\_TSAP

mov ax, res\_calc

;out dx, ax

mov dx, port\_OUT

mov ax, 0000000000000000b

;out dx, ax

call \_delay

jmp AGAIN

catch\_true:

mov bx, 0

mov cur\_iter, bx

mov bx, 2

mov system\_status, bx

jmp CUTTING

CUTTING:

call \_check\_emergency\_button

jnz BLOCKING

call \_check\_KV1

jnz cutting\_false

call \_check\_gas\_pressure

jz cutting\_false

call \_check\_oxygen\_pressure

jz cutting\_false

mov bx, cur\_iter

cmp bx, 3500

ja cutting\_rolgang

inc bx

mov cur\_iter, bx

mov dx, port\_OUT

mov ax, 0000000001101000b

;out dx, ax

call \_delay

jmp AGAIN

cutting\_rolgang:

mov bx, cur\_iter

cmp bx, 5000

ja cutting\_true

inc bx

mov cur\_iter, bx

mov dx, port\_OUT

mov ax, 0000000011101000b

;out dx, ax

call \_delay

jmp AGAIN

cutting\_true:

mov bx, 0

mov cur\_iter, bx

mov bx, 3

mov system\_status, bx

jmp RELEASE

cutting\_false:

mov bx, 0

mov cur\_iter, bx

mov bx, 5

mov system\_status, bx

jmp RELEASE\_LAMP

RELEASE:

call \_check\_emergency\_button

jnz BLOCKING

mov bx, cur\_iter

cmp bx, 120

ja release\_true

mov param\_3, bx

fld dword ptr [param\_4]

fimul word ptr [param\_3]

fadd dword ptr [param\_5]

fistp word ptr [res\_calc]

inc bx

mov cur\_iter, bx

mov dx, address\_TSAP

mov ax, res\_calc

;out dx, ax

mov dx, port\_OUT

mov ax, 0000000000000000b

;out dx, ax

call \_delay

jmp AGAIN

release\_true:

mov bx, 0

mov cur\_iter, bx

mov bx, 4

mov system\_status, bx

jmp SOURCE

SOURCE:

call \_check\_emergency\_button

jnz BLOCKING

call \_check\_KV0

jnz KV0\_true

mov bx, cur\_iter

cmp bx, 5000

ja cutting\_source

inc bx

mov cur\_iter, bx

mov dx, port\_OUT

mov ax, 0000000000010100b

;out dx, ax

call \_delay

jmp AGAIN

KV0\_true:

mov bx, cur\_iter

cmp bx, 5000

ja cutting\_source

inc bx

mov cur\_iter, bx

mov dx, port\_OUT

mov ax, 0000000000010000b

;out dx, ax

call \_delay

jmp AGAIN

cutting\_source:

mov bx, 0

mov cur\_iter, bx

mov system\_status, bx

jmp WAITING

RELEASE\_LAMP:

call \_check\_emergency\_button

jnz BLOCKING

mov bx, cur\_iter

cmp bx, 120

ja release\_lamp\_true

mov param\_3, bx

fld dword ptr [param\_4]

fimul word ptr [param\_3]

fadd dword ptr [param\_5]

fistp word ptr [res\_calc]

inc bx

mov cur\_iter, bx

mov dx, address\_TSAP

mov ax, res\_calc

;out dx, ax

mov dx, port\_OUT

mov ax, 0000000000000001b

;out dx, ax

call \_delay

jmp AGAIN

release\_lamp\_true:

mov bx, 0

mov cur\_iter, bx

mov bx, 6

mov system\_status, bx

jmp SOURCE

SOURCE\_LAMP:

call \_check\_emergency\_button

jnz BLOCKING

call \_check\_KV0

jnz KV0\_lamp\_true

mov bx, cur\_iter

cmp bx, 5000

ja cutting\_lamp\_source

inc bx

mov cur\_iter, bx

mov dx, port\_OUT

mov ax, 0000000000010101b

;out dx, ax

call \_delay

jmp AGAIN

KV0\_lamp\_true:

mov bx, cur\_iter

cmp bx, 5000

ja cutting\_lamp\_source

inc bx

mov cur\_iter, bx

mov dx, port\_OUT

mov ax, 0000000000010001b

;out dx, ax

call \_delay

jmp AGAIN

cutting\_lamp\_source:

mov bx, 0

mov cur\_iter, bx

mov system\_status, bx

jmp WAITING

BLOCKING:

mov dx, port\_OUT

mov ax, 0000000000000000b

;out dx, ax

jmp SHUTDOWN

AGAIN:

jmp MAIN\_CYCLE

SHUTDOWN:

ret

end \_start

Теперь перейдём к подробному описанию работы вышепоказанного алгоритма.

Краткое описание структуры алгоритма: задержка, а также проверки отдельных разрядов входных значений вынесены в подпрограммы, которые вызываются из различных участков алгоритма; алгоритм поделён на логические блоки, переход в каждый из которых выполняется благодаря статусу системы (особая переменная, которая указывает, какую часть кода сейчас нужно выполнять); из логического блока вызываются только нужные на данном этапе выполнения проверки входных данных, после чего выполняется переход в другие части алгоритма (при выполнении задачи текущего логического блока).

**Полное описание**

В основном цикле в первую очередь считываются входные данные с порта. После этого выполняется определение текущего режима работы системы по статусы системы, в зависимости от которого программа переходит в один из нескольких блоков: ОЖИДАНИЕ, ЗАХВАТ, РЕЗКА, ОТПУСКАНИЕ, ИСХОДНАЯ, ОТПУСКАНИЕ С ЛАМПОЙ, ИСХОДНАЯ С ЛАМПОЙ. Также есть 2 особых блока алгоритма: БЛОКИРОВКА и ВЫКЛЮЧЕНИЕ.

БЛОКИРОВКА вызывается, если на любом этапе выполнения алгоритма выяснилось, что была нажата главная аварийная кнопка. В таком случае в порт вывода подаются все 0 (остановка любых действий объекта), после чего выполняется переход в ВЫКЛЮЧЕНИЕ.

ВЫКЛЮЧЕНИЕ выполняет только одну инструкцию – прерывает бесконечный цикл работы программы, т.е. при активации этого участка кода весь алгоритм завершается, и оператору нужно заново запускать работу объекта.

ОЖИДАНИЕ – стартовое состояние системы, в котором системы ожидает получение разрешающего сигнала от ПРДЗ. Если такой сигнал не приходит – система передаёт в порт вывода все 0 и начинает работу основного цикла заново. Если сигнал от ПРДЗ пришёл – изменяет статус системы на ЗАХВАТ, устанавливает текущую итерацию как 0 и переходит к захвату.

ЗАХВАТ выполняет активацию соленоидного клапана. По текущей итерации высчитывается значение силы тока через сопроцессор (ассемблер не может работать с числами с плавающей запятой, а также округлять значения) и передаётся в ЦАП. Также увеличивается текущая итерация на 1 и начинает заново выполняться основной цикл. Если в блоке ЗАХВАТ текущая итерация стала больше нужной (больше 120), тогда захват считается выполненным – текущая итерация устанавливается на 0, устанавливается новый статус системы и управление переходит к блоку РЕЗКА.

В РЕЗКЕ происходит проверка датчика КВ1, а также датчиков давления газа и кислорода. Если датчики передают разрешающие значения – выполняется итерация резки (логика аналогична логике захвата, т.е. вычисления происходят итерационно). После выполнения итерации резки передаются соответствующие данные на порт вывода (движение резака вперёд, подача газа и кислорода). За полторы секунды до конца резки (сравнивается значение текущей итерации с константой) системы начинает выдавать дополнительно разрешение на рольганг. После завершения цикла резки система устанавливает текущую итерацию на 0, устанавливает новое состояние и переходит к блоку ОТПУСКАНИЕ. Если на какой-то итерации резки случается так, что был достигнут КВ1 или упало давление газа или кислорода – резка сразу прекращается и управление переходит в блок ОТПУСКАНИЕ С ЛАМПОЙ.

ОТПУСКАНИЕ и ОТПУСКАНИЕ С ЛАМПОЙ идентичны по своей логике: выполняется код, полностью аналогичный блоку ЗАХВАТ с той лишь разницей, что величина силы тока вычисляется по другой формуле. Также ОТПУСКАНИЕ отличается от ОТПУСКАНИЯ С ЛАМПОЙ тем, что не выводит сигнал на активацию лампы индикации «Заготовка не отрезана». Также переход после ОТПУСКАНИЯ осуществляется в блок ИСХОДНАЯ, а после ОТПУСКАНИЕ С ЛАМПОЙ – в ИСХОДНАЯ С ЛАМПОЙ.

ИСХОДНАЯ и ИСХОДНАЯ С ЛАМПОЙ также отличаются только тем, что ИСХОДНАЯ не выводит сигнал на лампу индикации. Сама логики работы данных блоков следующая: системе нужно вернуть тележку и резак в исходные состояния. Выполняется проверка КВ0, и пока он не активен – подаётся сигнал приводу тележки на движение назад. Кроме проверки КВ0 выполняется перемещение резака «вслепую»: т.к. у нас отсутствует датчик, обозначающий что резак находится в исходном положении, то движение резака осуществляется итерационно циклом с таким же количеством итераций, что и цикл резки. Как только резак приехал в исходную позицию, а также достигнут КВ0 – система устанавливает статус на ОЖИДАНИЕ и начинает работу основного цикла заново.

Для синхронизации вышеописанного алгоритма с реальным временем, что очень важно при работе с процессами захвата, резки и отпускания, а также движение резака в исходную позицию, выполняется вызов подпрограммы задержки на 1 мс. Как это работает? Пример при работе цикла ЗАХВАТА: при состоянии системы в «ЗАХВАТЕ» выполнение всех инструкций осуществляется примерно за 432 такта, что равняется  миллисекундам (при тактовой частоте исполнительного процессора в 2.50 ГГц, который используется на момент вычислений). Эта величина составляет  от одной миллисекунды, так что без использования подпрограммы задержки весь цикл захвата, который должен выполняться 120 миллисекунд, выполнился бы примерно за 1 миллисекунду, что является некорректным. Для корректировки процесса после каждой итерации вызывается задержка в 1 миллисекунду, а используемое время выполнения всех инструкций итерации игнорируется (т.к. это слишком малая величина).

**10 Высокоуровневое имитационное приложение**

Имитационное приложение представляет из себя панель управления, которая позволяет в реальном времени (имитирует работу в реальном времени) посмотреть, как СРВ ведёт себя при различных входных данных. Запрограммированный алгоритм полностью идентичен программе-драйверу, описанной в 9 пункте.

Для написания приложения использовался язык Python и графическая библиотека TKinter.

Код программы описан в файле gui.py, а ниже показано само приложение на различных этапах работы (рисунки 10.1 – 10.5):

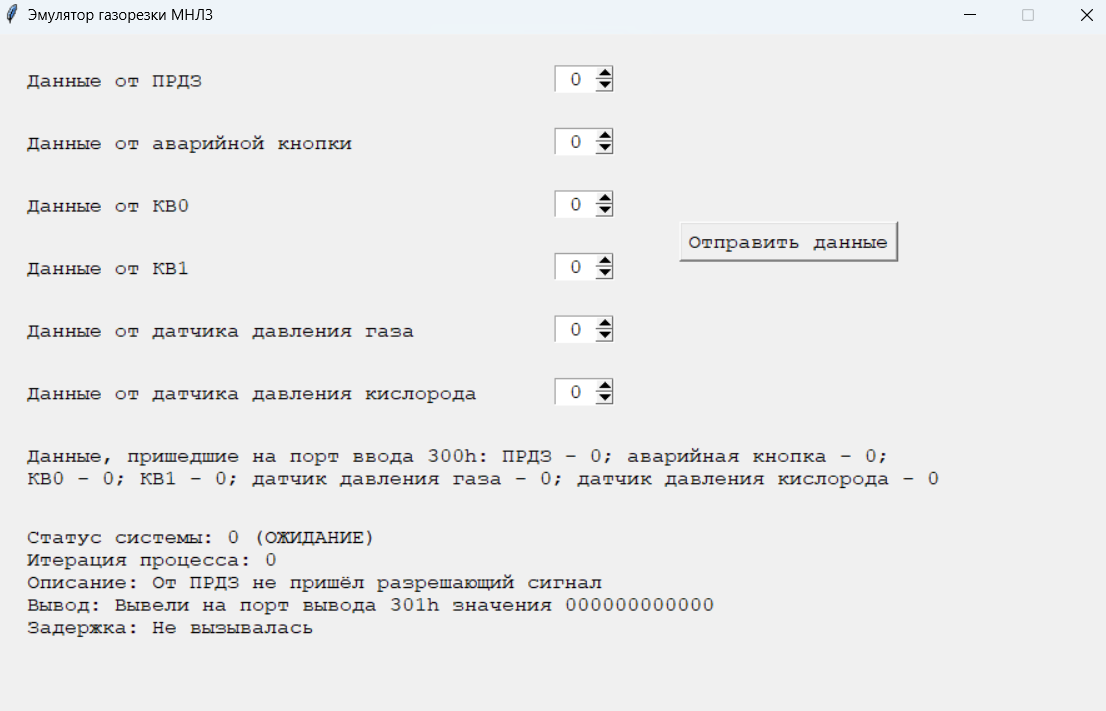


Рисунок 10.1 – этап ожидания

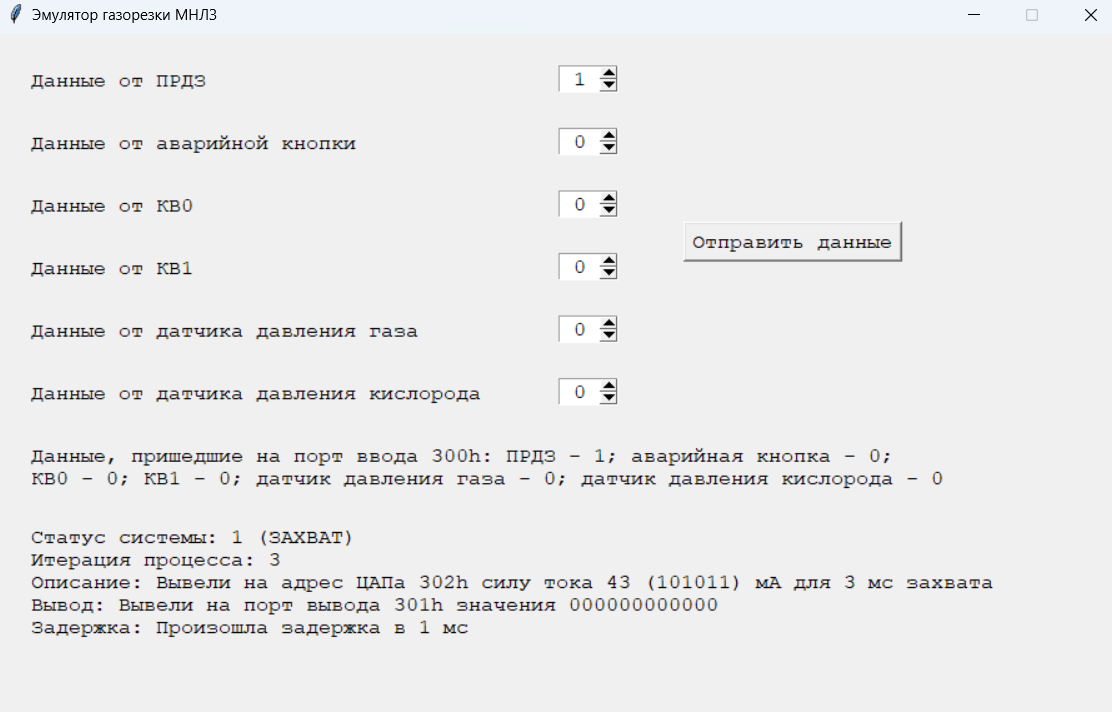


Рисунок 10.2 – этап захвата

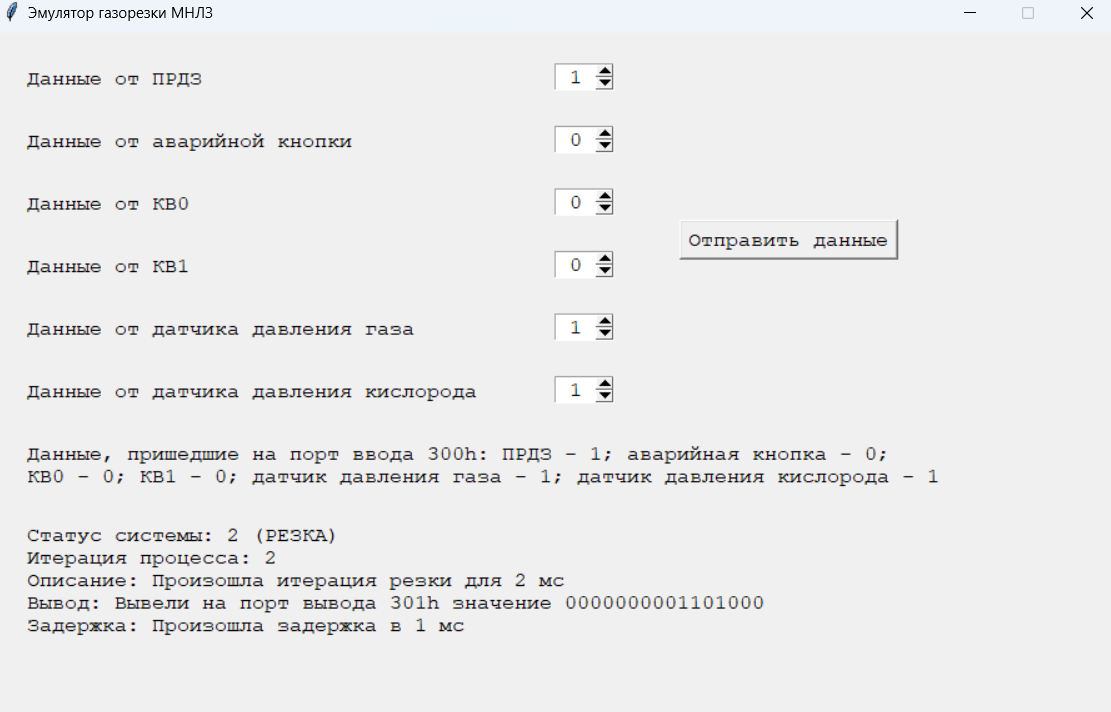


Рисунок 10.3 – этап резки

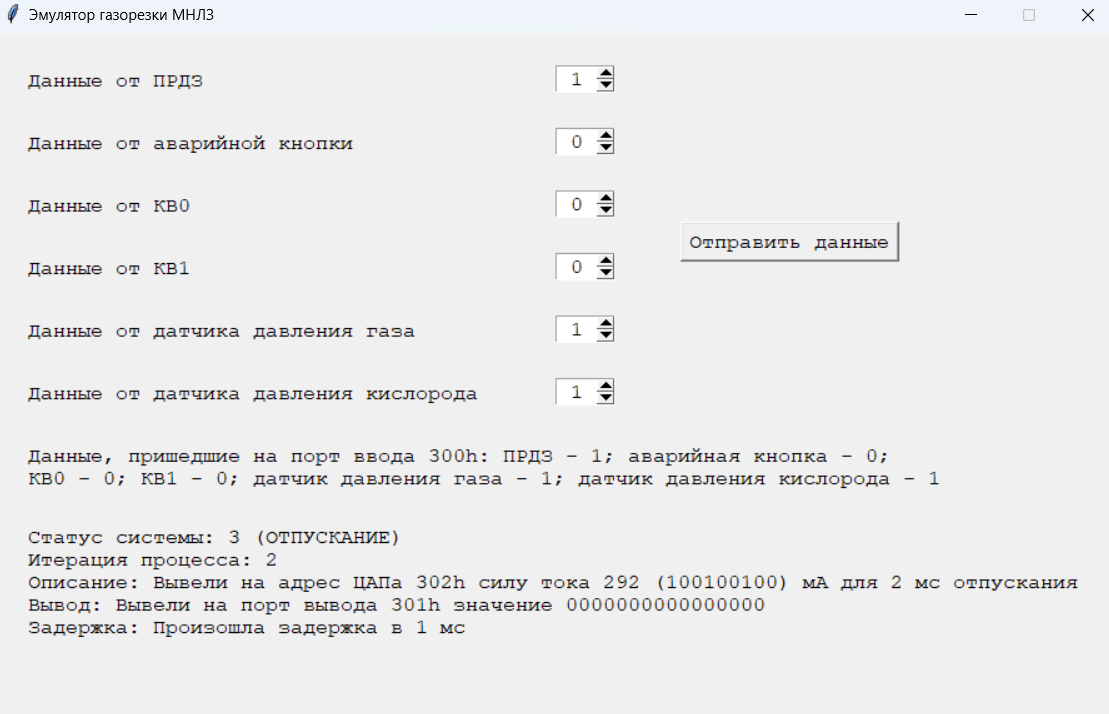


Рисунок 10.4 – этап отпускания

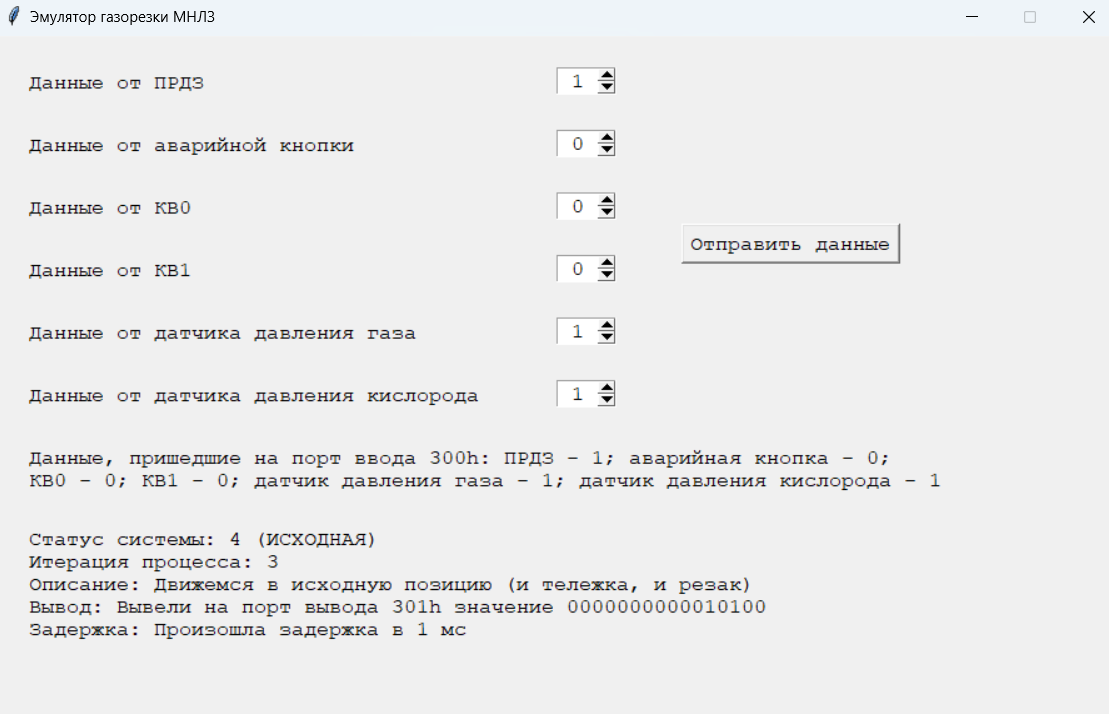


Рисунок 10.5 – этап исходной