СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Пневматический привод – это механическое устройство, в гильзе которого сила сжатого воздуха, через поршень, трансформируется в возвратно-поступательное движение и заставляет работать какой-либо механизм. В самом простом представлении пневмоцилиндр - это отрезок трубы различного диаметра и длины, закрытой на торцах крышками. В крышки вворачиваются пневматические соединения - фитинги, для подачи сжатого воздуха внутрь цилиндра.

Внутри цилиндра вмонтирован поршень, который жестко соединяется со штоком пневматического привода. Через фитинг происходит подача сжатого воздуха, двигая поршень, который поступательно перемещается и приводит в действие рабочий механизм. До тех пор, пока сжатый воздух находится в полости цилиндра, поршень под давлением, пневматический цилиндр выполняет свою функцию. Обратный процесс происходит тогда, когда сжатый воздух сбрасывается из полости цилиндра, а затем подаётся в противоположную сторону. Поршень совершает движение, но в обратном направлении, и работа исполнительного механизма прекращается

Целью научной работы является построение автоматной модели управления пневмоцилиндрами, моделирование пневмоцилиндра в среде SimInTech, описание взаимодействия автоматной модели, реализованной на языке Си и среды динамического моделирования SimInTech.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Описание пневмоцилиндра

Пневматические цилиндры – это приспособления для линейного перемещения рабочего органа станков и других механизмов. В отличие от исполнительных устройств поворотного типа, имеющих довольно сложную конструкцию, пневмоцилиндры состоят из полой гильзы, внутри которой при помощи сжатого воздуха движется шток, создавая втягивающее и толкающее воздействие на механизмы.

Сжатый подается в поршневую полость пневмоцилиндра, штоковая полость в этот момент с помощью распределителя соединяется с атмосферой, давление сжатого воздуха воздействует на поршень, заставляя его перемещаться, до тех пор, пока он не упрется в переднюю крышку. Пневмоцилиндр совершает прямой ход, его шток выдвигается.

Для осуществления обратного хода необходимо подать сжатый воздух в штоковую полость, а поршневую - соединить с атмосферой. Под действием давления сжатого воздуха поршень станет перемещаться, шток будет задвигаться. Направление потоков сжатого воздуха в поршневую и штоковую полости, а также соединение их с атмосферой или линией сброса осуществляется с пневматических распределителей.

Пневматические цилиндры используются во всех отраслях промышленности, где необходимо совершать поступательные перемещения. Массовое применение пневмоцилиндры получили в прессовом производстве, линиях разлива и упаковки продуктов, в составе транспортных средств, при погрузочно-разгрузочных работах, в подъёмниках и конвейерных системах.

1.2. Описание модели пневмоцилиндра в среде SimInTech

Общий вид модели пневмоцилиндра в среде SimInTech показан на Рисунке 1.2.1.

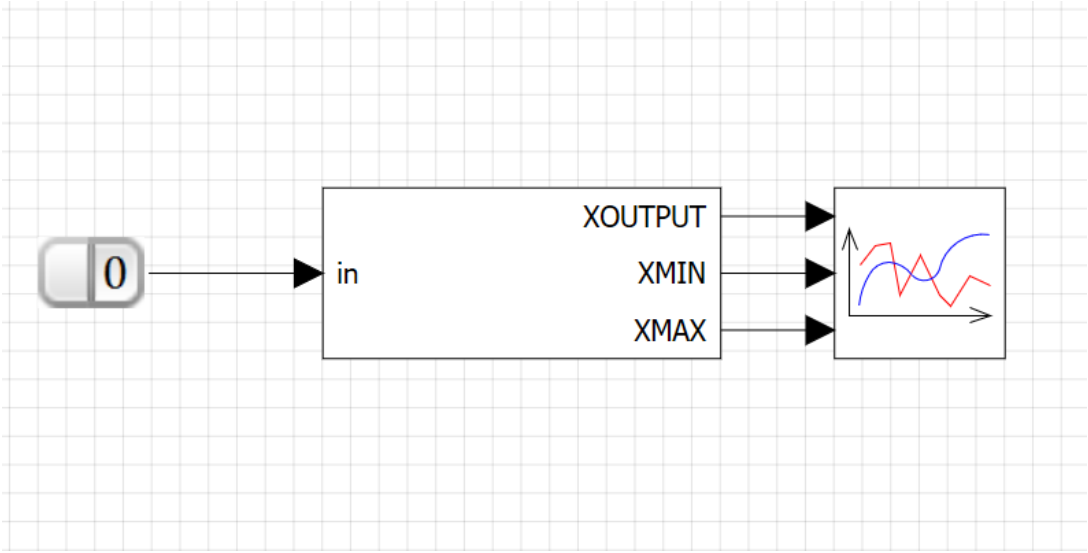

Рисунок 1.2.1 — Модель пневмоцилиндра.

Внутри модели находится модель, показанная на Рисунке 1.2.2. В ней используются механическ2ий элемент поступательного движения, пневмомеханический преобразователь поступательного типа, пневматическая полость переменного объема, порт входа сигнала, источник и дренаж, а также два пневматических турбулентных дросселя с регулированием по произвольному параметру. Параметры перечисленных элементов показаны на Рисунках 1.2.2-1.2.10.

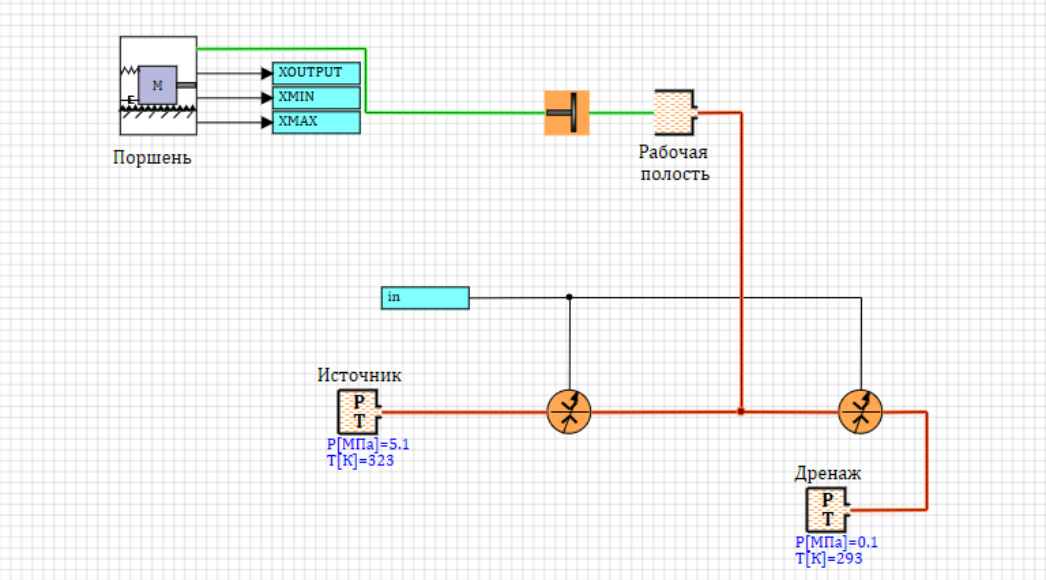

Рисунок 1.2.2 — Описание модели пневмоцилиндра.

Для пневмоцилиндра, опишем дополнительно три выходных параметра: датчик положения поршня, концевой датчик положения поршня в крайнем верхнем положении и концевой датчик положения поршня в крайнем нижнем положении. Механический элемент поступательного движения не имеет подобных выходных датчиков, поэтому придётся их сделать самостоятельно. Войдём в субмодель и добавим необходимые выходные сигналы (Рисунок 1.2.3.)

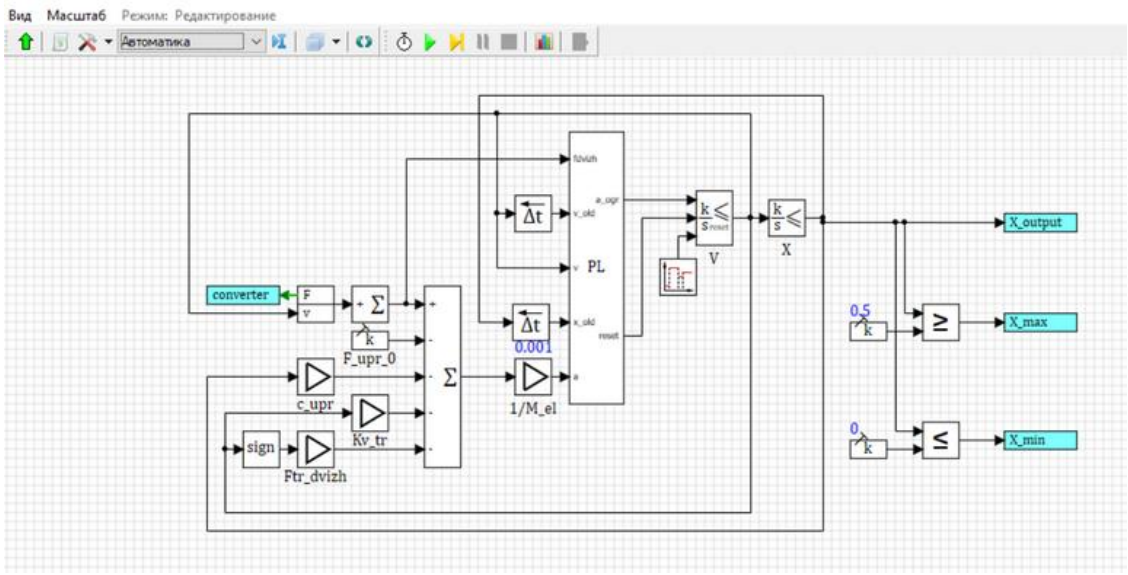

Рисунок 1.2.3 — Внутреннее устройство механического элемента поступательного движения.

2. ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1. Описание задания

Моделируемый процесс задается пятеркой (Y, T, D, E, P). Где Y — множество пневмоцилиндров; T — множество отрезков времени за которые набор пневмоцилиндров на заданном шаге должны изменить положение; D множество отрезков времени в течении которых на заданном шаге процесса должно удерживаться положение пневмоцилиндров после их стабилизации;

E— множество пар $(p_i p_j)$ переходов с шага процесса p_i, в котором за время t_i не удалось сменить положение пневмоцилиндров, на шаг p_j;

P — последовательность наборов пневмоцилиндров изменяемых на заданном шаге процесса.

Последовательность наборов цилиндров представляет из себя строку, в которой задана последовательность включения и отключения пневмоцилиндров, при этому выключение (цилиндр находится в крайнем нижнем положении) обозначается отрицанием y_1, а включение (цилиндр находится в крайнем верхнем положении) обозначено без отрицания y_1. В одну единицу шага процесса в скобках указывается в какое состояние должны перейти пневмоцилиндры. Например, $P = (y_1 y_2)$ говорит о том, что по завершении этого шага пневмоцилиндры y_1 и y_2 должны перейти в крайнее нижнее положение, при этом предыдущее положение этих цилиндров должно быть крайним верхним $(y_1 y_2)$.

Отдельно задается время, за которое пневмоцилиндры должны переместиться за одну единицу шага процесса $(T = \{t_1, t_2, t_3, \ldots, t_n\})$ и время нахождения на данном шаге процесса $(D = \{d_1, d_2, d_3, \ldots, d_n\})$

2.2. Исходные данные

Моделируемый процесс варианта №4 задаётся пятёркой ниже:

4.4 Задание №4

$$Y = y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8$$

$$\begin{aligned} P = {} & (\overline{y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8}), (y_2), \\ & (y_8 \overline{y_2} y_4), (y_2 y_3), \\ & (y_1 \overline{y_2} y_3 \overline{y_4} y_5 y_6 y_7 \overline{y_8}), (y_3 y_4 \overline{y_5}), \\ & (y_2 \overline{y_3}), (y_8 \overline{y_2}), \\ & (\overline{y_8 y_1 y_6}), (y_1 y_2 y_3 \overline{y_4} y_5 y_6 \overline{y_7}), \\ & (\overline{y_1 y_2 y_3} y_4 \overline{y_5 y_6} y_8), (y_2), \\ & (\overline{y_8} y_1 \overline{y_4 y_5}), (\overline{y_1 y_2} y_3 \overline{y_5} y_6 y_7 y_8), \\ & (y_4 y_5 \overline{y_6 y_7}), (\overline{y_8} y_1 y_2 \overline{y_5 y_7}), \\ & (\overline{y_1 y_3 y_4} y_5 y_6 \overline{y_7} y_8), (\overline{y_2} y_3 y_4 \overline{y_5 y_6}) \end{aligned}$$

$$T = \begin{cases} t_4 = t_6 = t_{11} = t_{13} = 120 \\ t_1 = t_2 = t_5 = t_8 = t_{10} = t_{12} = t_{14} = 56 \\ t_9 = 60 \\ t_0 = t_3 = t_{16} = 45 \\ t_7 = t_{15} = 30 \end{cases}$$

$$D = \begin{cases} d_6 = 33 \\ d_0 = d_1 = d_{12} = 70 \\ d_2 = d_7 = d_{15} = 60 \\ d_4 = d_5 = d_8 = d_9 = d_{11} = d_{14} = 45 \\ d_3 = d_{10} = d_{13} = d_{16} = 78 \end{cases}$$

$$E = (p_7, p_4), (p_8, p_{12}).$$

2.3. Построение диаграммы Мура автоматной модели

Построим диаграмму Мура (Рисунок 2.3.1) для исходных данных с помощью ориентированного графа, вершины которого соответствуют состояниям автомата, а дуги – входным условиям. Автомат включает в себя 18 состояний, состояние ошибки, переход в которое выполняется при невыполнении перехода в следующее состояние в заданный промежуток времени, и начальное состояние.

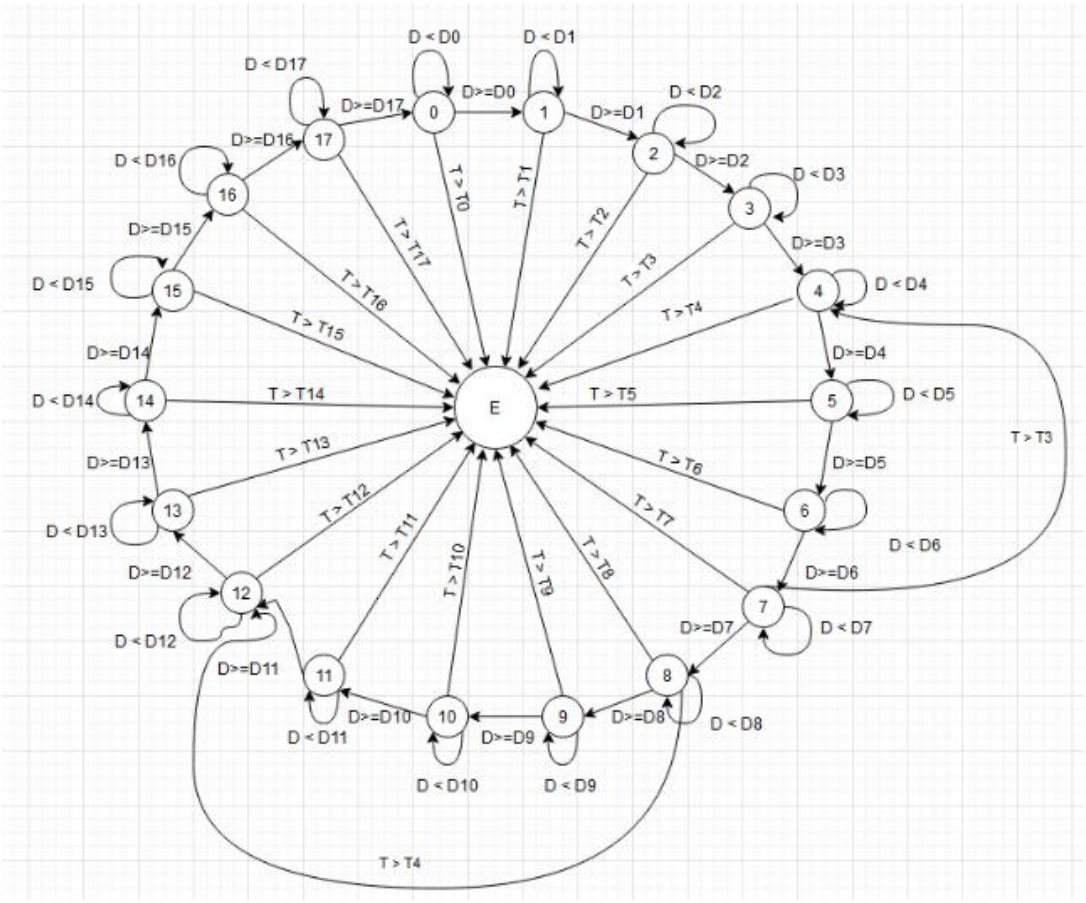

Рисунок 2.3.1 – Диаграмма последовательности Мура.

2.4. Кодирование автоматной модели на языке Си

На листинге 2.4.1 представлены шаги работы автоматной модели, на которых изменяется заданная последовательность пневмоцилиндров. На листинге 2.4.2 представлена структура для описания параметров пневмоцилиндра. Она содержит два входных сигнала показывающие, что пневмоцилиндр находится в одном из двух крайних положений, а также выходной управляющий пневмоцилиндром сигнал.

Листинг 2.4.1 — Перечисление шагов работы автоматной модели

```
enum PneumoState {
    PneumoState_Initialization = 0,
    PneumoState_0, PneumoState_1, PneumoState_2,
    PneumoState_3, PneumoState_4, PneumoState_5,
    PneumoState_6, PneumoState_7, PneumoState_8,
    PneumoState_9, PneumoState_10, PneumoState_11,
    PneumoState_12, PneumoState_13, PneumoState_14,
    PneumoState_15, PneumoState_16, PneumoState_17,
    PneumoState_FatalException
};
```

Листинг 2.4.2 — Структура пневмоцилиндра

```
struct PneumoCylinder {
    int input_signal[2];
    int output_signal;
};
```

На листинге показан 2.4.3. показан список пневмоцилиндров, учавствующих в создании автоматной модели. На листинге 2.4.4. показана структура управляющего пневмоцилиндра автомата. Она содержит текущее состояние автоматной модели, время, прошедшее с начала изменения состояния пневмоцилиндра, время, в течении которого автомат удерживает пневмоцилиндры в одном положении, массив времён, за которые автомат должен изменить положение пневмоцилиндров и массив времён, на которых автомат должен удерживать состояния пневмоцилиндров, а также массив пневмоцилиндров, которыми автомат управляет.

Листинг 2.4.3 — Список пневмоцилиндров, структура автомата.

```
#define PNEUMO_CYLINDER_Y1 0
#define PNEUMO_CYLINDER_Y2 1
#define PNEUMO_CYLINDER_Y3 2
#define PNEUMO_CYLINDER_Y4 3
#define PNEUMO_CYLINDER_Y5 4
#define PNEUMO_CYLINDER_Y6 5
#define PNEUMO_CYLINDER_Y7 6
#define PNEUMO_CYLINDER_Y8 7

struct PneumoEngine {
    enum PneumoState state;
    int timeout;
    int delay;
    int timeouts[PneumoState_FatalException];
    int delays[PneumoState_FatalException];
    struct PneumoCylinder cylinders[8];
};

void pneumo_engine_init(struct PneumoEngine *engine);

bool pneumo_engine_tick(struct PneumoEngine *engine);

void pneumo_error_handler(struct PneumoEngine *engine, enum PneumoState state);

void delay_timeout_init(struct PneumoEngine *engine, int delta_t, int delta_d, enum
PneumoState state);

void pneumo_state_body(struct PneumoEngine *engine, int cylinder_sygnals[], enum
PneumoState next_state, enum PneumoState error_state);

void pneumo_state_changer(struct PneumoEngine* engine, int cylinder_sygnals[]);

bool pneumo_state_checker(struct PneumoEngine* engine, int cylinder_sygnals[]);
```

На листинге 2.4.4 показана часть исполнения функции инициализации, которая выполняется при включении автоматной модели. В ней настраиваются временные задержки по заданию и начальные выходные сигналы. В качестве аргумента этой функции подается структурная переменная (engine), рассмотренная в листинге 2.4.3.

Листинг 2.4.4 — Часть кода функции инициализации

```
void delay_timeout_init(struct PneumoEngine* engine, int delta_t, int delta_d, enum
PneumoState state) {
        engine->timeouts[state] = TIMEOUT_DELTA(delta_t);
        engine->delays[state] = DELAY_DELTA(delta_d);
}

void pneumo_engine_init(struct PneumoEngine* engine) {
        if (engine != 0) {
                for (int i = 0; i < 8; i++)
                {
                        engine->cylinders[i].input_signal[PNEUMO_CYLINDER_SIGNAL_UP]
= 0;
                        engine-
>cylinders[i].input_signal[PNEUMO_CYLINDER_SIGNAL_DOWN] = 0;
                        engine->cylinders[i].output_signal = 0;
```

```
                }

                engine->state = PneumoState_0;
                engine->delay = 0;
                engine->timeout = 0;

                delay_timeout_init(engine, 45, 70, PneumoState_0);
                delay_timeout_init(engine, 56, 70, PneumoState_1);
                delay_timeout_init(engine, 56, 60, PneumoState_2);
                delay_timeout_init(engine, 45, 78, PneumoState_3);
                delay_timeout_init(engine, 120, 45, PneumoState_4);
                delay_timeout_init(engine, 56, 45, PneumoState_5);
                delay_timeout_init(engine, 120, 33, PneumoState_6);
                delay_timeout_init(engine, 30, 60, PneumoState_7);
                delay_timeout_init(engine, 56, 45, PneumoState_8);
                delay_timeout_init(engine, 60, 45, PneumoState_9);
                delay_timeout_init(engine, 56, 78, PneumoState_10);
                delay_timeout_init(engine, 120, 45, PneumoState_11);
                delay_timeout_init(engine, 56, 70, PneumoState_12);
                delay_timeout_init(engine, 120, 78, PneumoState_13);
                delay_timeout_init(engine, 56, 45, PneumoState_14);
                delay_timeout_init(engine, 30, 60, PneumoState_15);
                delay_timeout_init(engine, 45, 78, PneumoState_16);
                delay_timeout_init(engine, 45, 78, PneumoState_17);
        }
}
```

На листинге 2.4.6 показана часть исполнения функции обработки переходов, которая в зависимости от текущего состояния автоматной модели формирует сигналы для изменения состояний пневмоцилиндров. В качестве аргумента этой функции подается структурная переменная (engine), рассмотренная в листинге 2.4.4. Внутри функции задаётся логическая переменная (ret), которая меняет своё значение с истины на ложь в случае перехода процесса симуляции в состояние исключения. Переход автомата из одного состояния в другое осуществляется с помощью логического оператора. В случае, если пневмоцилиндры успели сменить своё состояние на требуемое по заданию, автомат переходит в следующее состояние. В противном случае, автомат переходит в ошибочное состояние, а в состояниях 1 и 5 автомат переходит в состояния 6 и 2 соответственно, с переведением всех пневмоцилиндров в крайнее нижнее положение.

Листинг 2.4.5 — Вспомогательные функции

```
void pneumo_error_handler(struct PneumoEngine* engine, enum PneumoState state) {
        engine->state = state;
        int cylinder_sygnals[] = {0, 0, 0, 0, 0, 0, 0, 0};
        pneumo_state_changer(engine, cylinder_sygnals);
        engine->delay = 0;
        engine->timeout = 0;
}

void pneumo_state_changer(struct PneumoEngine* engine, int cylinder_sygnals[]) {
        for (int i = 0; i < 8; i++)
        {
                switch (cylinder_sygnals[i])
                {
                        case 0:
                                engine->cylinders[i].output_signal = 0;
                                break;
                        case 1:
                                engine->cylinders[i].output_signal = 1;
                                break;
                }
        }
}

bool pneumo_state_checker(struct PneumoEngine* engine, int cylinder_sygnals[]) {
        for (int i = 0; i < 8; i++)
        {
                switch (cylinder_sygnals[i])
                {
                        case 0:
                                if (!engine-
>cylinders[i].input_signal[PNEUMO_CYLINDER_SIGNAL_DOWN])
                                        return false;
                                break;
                        case 1:
                                if (!engine-
>cylinders[i].input_signal[PNEUMO_CYLINDER_SIGNAL_UP])
                                        return false;
                                break;
                }
        }
    return true;
}

void pneumo_state_body(struct PneumoEngine* engine, int cylinder_sygnals[], enum
PneumoState next_state, enum PneumoState error_state) {
        pneumo_state_changer(engine, cylinder_sygnals);
        if (pneumo_state_checker(engine, cylinder_sygnals)) {
                engine->timeout = 0;
                engine->delay++;
                if (DELAY_GE(engine)) {
                        engine->state = next_state;
                        engine->delay = 0;
                        engine->timeout = 0;
                }
        }
        else if (TIMEOUT_GE(engine)) {
                pneumo_error_handler(engine, error_state);
        }
        else if (engine->delay > 0 && !DELAY_GE(engine)) {
                pneumo_error_handler(engine, error_state);
        } }
```

```
case PneumoState_1: {
        int cylinder_sygnals[] = { 0, 1, 0, 0, 0, 0, 0, 0 };
        pneumo_state_body(engine, cylinder_sygnals, PneumoState_2,
PneumoState_FatalException);
        break;
}
```

Листинг 2.6 — Первое состояние автомата

```
#define TIMEOUT_GE(engine) ( (engine)->timeout > (engine)->timeouts[(engine)->state]
)
#define DELAY_GE(engine) ( (engine)->delay > (engine)->delays[(engine)->state] )
```

Рисунок 2.7 — Макроопредления TIMEOUT_GE и DELAY_GE

По условию, есть множество пар (p_7, p_4) переходов с состояния PneumoState_7, в котором за время t_4 не удалось сменить положение пневмоцилиндра, на состояние PneumoState_4. По условию задания, меняем положения цилиндра на положения в 11 состоянии (Листинг 2.6). Если на входном сигнале единица, то сбрасывается timeout, и, если время задержки больше, чем эталонное время задержки для второго состояния, автомат переходит в состояние PneumoState_5. Иначе, если счетчик timeout превышает эталонное время для данного состояния, то автомат переходим в состояние PneumoState_4, а затем задвигаем все пневмоцилиндры и сбрасываем delay и timeout.

```
case PneumoState_7: {
   int cylinder_sygnals[] = { 1, 0, 0, 1, 0, 1, 1, 1 };
   pneumo_state_body(engine, cylinder_sygnals, PneumoState_8, PneumoState_4);
   break;
}
```

Листинг 2.8 — Второе состояние автомата

В случае перехода автомата в состояние PneumoState_FatalException (Листинг 2.9) переменной логического типа ret присваивается значение false.

```
bool pneumo_engine_tick(struct PneumoEngine* engine) {
        bool ret = true;
        if (0 == engine)
                return false;
        switch (engine->state) {
                case PneumoState_0: {
                        int cylinder_sygnals[] = { 0, 0, 0, 0, 0, 0, 0, 0 };
                        pneumo_state_body(engine, cylinder_sygnals, PneumoState_1,
PneumoState_FatalException);
                        break;
                }
                ...
                case PneumoState_FatalException: {
                        ret = false;
                        break;
                }
        }
        engine->timeout++;
        return ret;
}
```

Листинг 2.9 — Функция логики переходов автомата, ошибочное состояние

2.5. Модель автомата в среде SimInTech

Для использования написанного на языке Си кода в качестве управляющего пневмоцилиндрами модуля, необходимо провести генерацию кода. С помощью блока «Язык программирования», входных и выходных контактов задаётся структура генерируемого управляющего модуля. Схема для генерации кода показана на Рисунке 2.5.1.

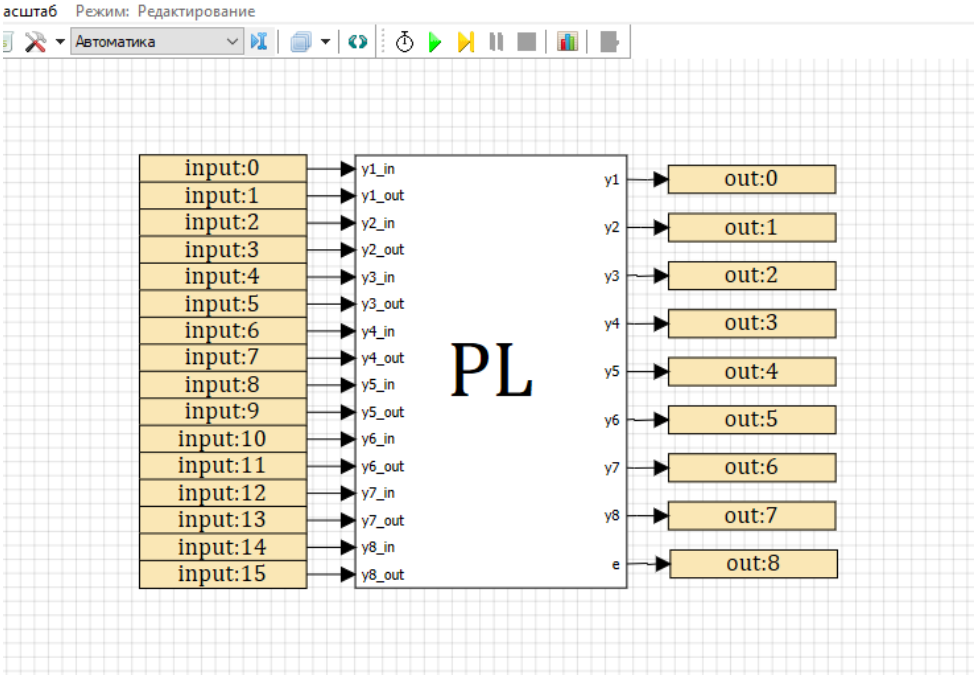

Рисунок 2.5.1 — Схема для генерации кода.

Далее необходимо подключить файлы, описывающие работу автоматной модели на языке Си, а также объявить поле структуры автомата.

После настройки блока «Язык программирования» необходимо сгенерировать программу с помощью встроенной функции «Сгенерировать программу» в программе SimInTech. При успешной генерации в папке появится dll библиотека, которую можно подключить как внешний модулю в схему симуляции автоматной модели. Для этого на схему моделирования автоматной модели необходимо добавить восемь моделей пневмоцилиндров, блоки.

«Внешняя DLL» и «Временной график». Схема для моделирования автоматной модели показана на Рисунке 2.5.2.

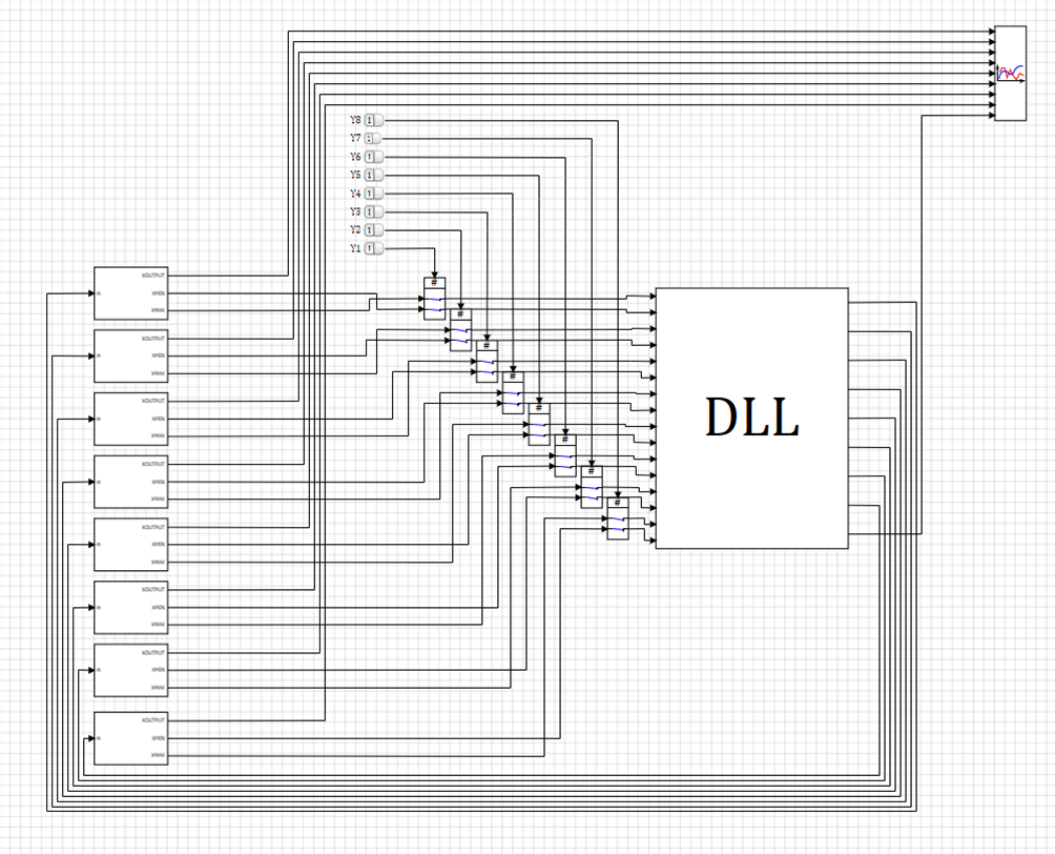

Рисунок 2.5.2 — Схема автоматной модели

В свойствах блока «Внешняя DLL» указывается количество входных и выходных портов, а также путь к сгенерированной dll библиотеке (Рисунок 2.5.6).

Свойства | Общие | Порты | Визуальные слои

Название	Имя	Формула	Значение
Тип сортировки	SortType		Функциональный
Количество портов	nport		16
Массив размерностей выходов	outdims		[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]
Имена загружаемых DLL	dllnames		D:\Simintech\KPMod\Code\mydiagram.dll
Имена оборудования	component...		a1
Имена файлов проектов для отладки	prjnames		
Создавать по умолчанию не существующ...	useemptyv...		☐Нет
Количество потоков	nthread		1

Рисунок 2.5.3 — Свойства блока «Внешняя DLL».

В результате симуляции процесса управления автоматной модели генерируется график движения и положения всех пневмоцилиндров, а также состояния, обозначающее корректность работы системы управления (Рисунок 2.5.7). При переходе в последнее состояние пневмоцилиндр будет сохранять его.

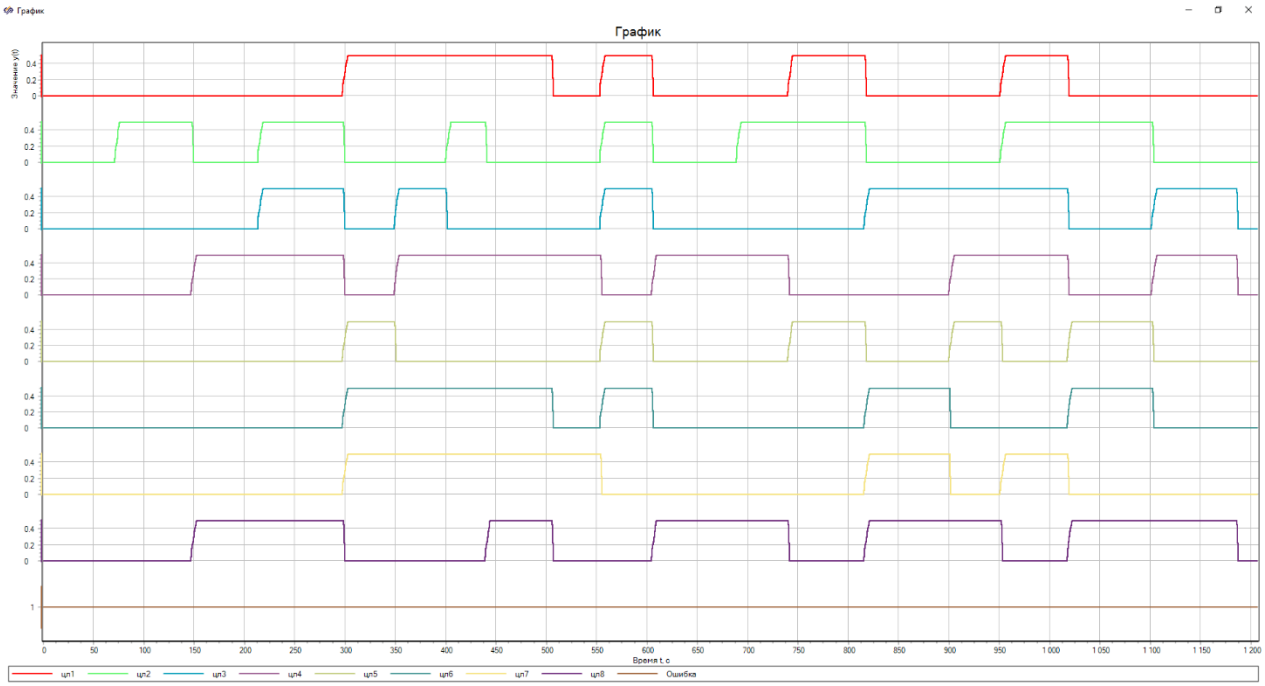

Рисунок 2.5.4 — График симуляции управления автоматной моделью.

2.6. Моделирование отказа пневмоцилиндра

Смоделируем вызов исключения пневмоцилиндра.

В результате симуляции получается, что автоматная модель не успевает изменить положения поршней за этот промежуток времени и переходит в новое состояние.

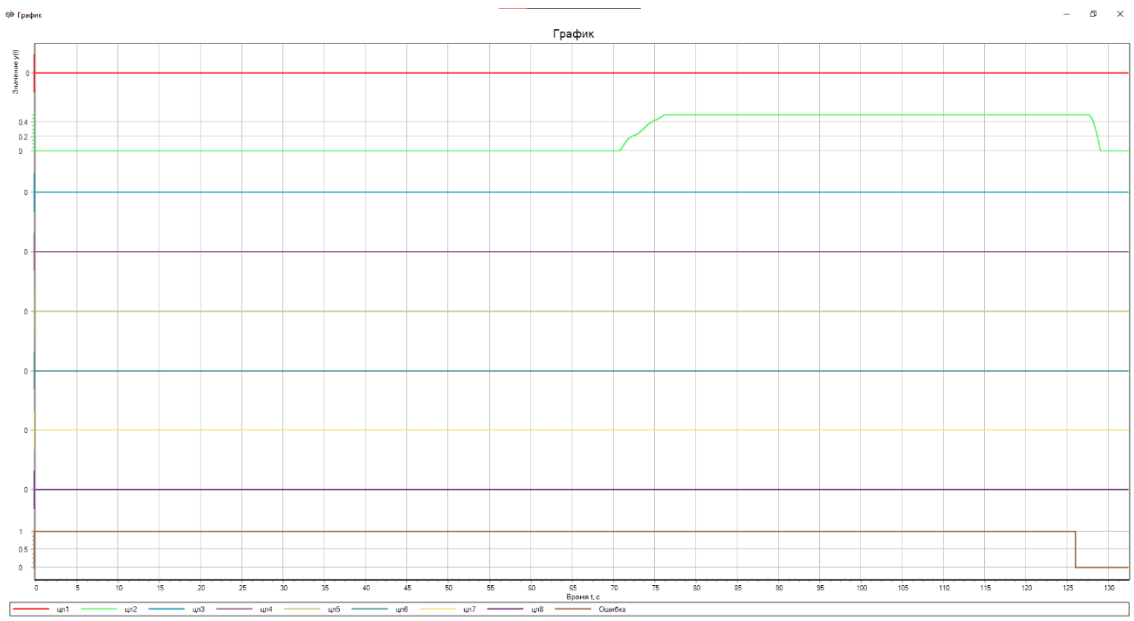

Рисунок 2.6.1 — График симуляции управления автоматной моделью (переход в ошибочное состояние).

В результате симуляции получается, что автоматная модель не успевает изменить положения поршней за этот промежуток времени и переходит, согласно условию, во второе состояние. График симуляции показан на Рисунке 2.6.2.

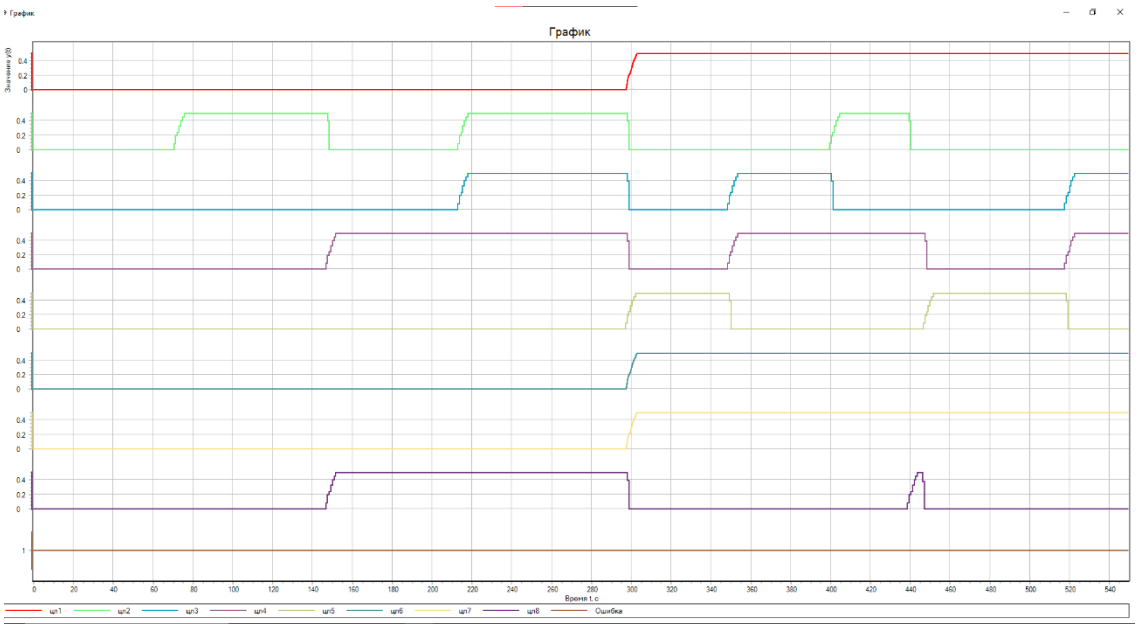

Рисунок 2.6.2 — График симуляции управления автоматной моделью (переход в 7-ое состояние из 4-го)

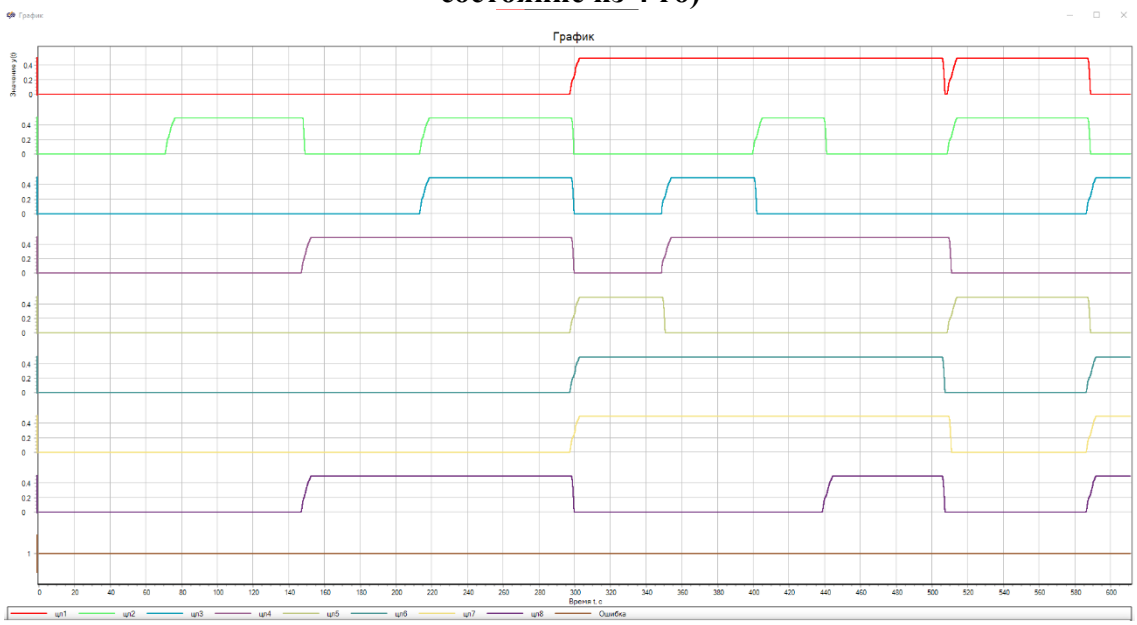

Рисунок 2.6.3 — График симуляции управления автоматной моделью (переход из 8го состояния в 12ое).

ВЫВОДЫ

В данной научной работе была спроектирована автоматная модель, одновременно управляющая восемью пневмоцилиндрами заданной по условию последовательностью состояний, была написана программа на языке программирования Си и успешно подключена в виде модуля, управляющая автоматной моделью.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Поликарпова Н. И., Шалыто А. А. Автоматное программирование. 2008. — 167 с.: ил.

2. Статья на сайте Хабр [Электронный ресурс]. - Режим доступа: https://habr.com/ru/post/331556/, свободный - (19.05.2022).

3. Введение в автоматное программирование [Электронный ресурс]. - Режим доступа: https://www.osp.ru/pcworld/2003/10/166609, свободный - (19.05.2022).

4. Введение в Си [Электронный ресурс]. - Режим доступа: https://habr.com/ru/post/464075/, свободный - (19.05.2022).