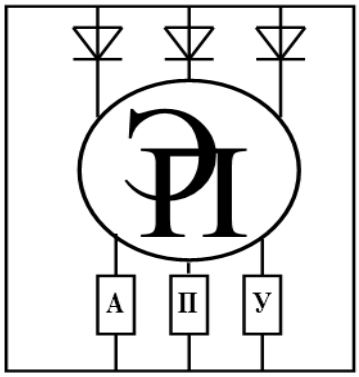
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Электропривод и АПУ»

СИСТЕМЫ ПРОГРАММНОГО УПРАВЛЕНИЯ

Методические рекомендации к лабораторным работам для студентов специальностей 1-53 01 05 «Автоматизированные электроприводы» и 6-05-0713-04 «Автоматизация технологических процессов и производств» очной и заочной форм обучения

Часть 2



Могилев 2025

УДК 000.4
ББК 32.295
С40

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Электропривод и АПУ» «29» августа 2025 г., протокол № 1

Составитель канд. техн. наук, доц. Л. Г. Черная

Рецензент канд. техн. наук, доц. С. В. Болотов

В методических рекомендациях к лабораторным работам для студентов специальностей 1-53-01 05 «Автоматизированные электроприводы» и 6-05-0713-04 «Автоматизация технологических процессов и производств» дневной и заочной форм обучения изложены методы разработки программ для систем управления на базе программируемых логических контроллеров.

Учебное издание

СИСТЕМЫ ПРОГРАММНОГО УПРАВЛЕНИЯ

Часть 2

Ответственный за выпуск	А. С. Коваль
Корректор	А. А. Подошевко
Компьютерная верстка	М. М. Дударева

Подписано в печать 11.12.2025. Формат 60×84/16. Бумага офсетная. Гарнитура Таймс. Печать трафаретная. Усл. печ. л. 2,79. Уч.-изд. л. 2,94. Тираж 26 экз. Заказ № 869.

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

Введение

Лабораторные занятия по дисциплине «Системы программного управления» прививают студентам навык программирования логических контроллеров, облегчают восприятие и понимание основных теоретических положений, способствуя их более глубокому усвоению.

Методические рекомендации соответствуют программе курса «Системы программного управления». Они служат основой для самостоятельной подготовки и проведения лабораторных работ с последующим оформлением и анализом результатов и предусматривают изучение теоретического материала по учебной и справочной литературе.

К выполнению лабораторных работ студенты допускаются после ознакомления с правилами и инструкцией по технике безопасности и проведения инструктажа по безопасным методам работы с программируемыми логическими контроллерами, компьютерами с оформлением соответствующей записи в журнале.

Для получения допуска к очередным занятиям студенты предварительно изучают содержание лабораторной работы, рекомендации к выполнению задания и представляют законченный отчет по предыдущей работе.

1 Лабораторная работа № 7. Программирование логических контроллеров на языке IL в системе CoDeSys

Цель работы: изучить принципы составления прикладных программ для промышленных логических контроллеров (ПЛК) на языке IL пакета CoDeSys; приобрести навыки программирования на языке IL в системе CoDeSys.

Задание

1 Изучить основные правила составления управляющих программ на языке IL в системе CoDeSys.

2 Изучить порядок ввода редактирования и отладки управляющих программ на языке IL в системе CoDeSys.

3 Составить управляющую программу, реализующую управление двигателем на языке IL.

4 Проверить работу управляющей программы в режиме эмуляции.

5 Записать программу в память контроллера и проверить ее выполнение.

Рекомендации к выполнению задания

При выполнении работы необходимо дополнительно использовать Руководство пользователя по программированию ПЛК в CoDeSys 2.3 и Справочную систему комплекса программирования CoDeSys.

Язык IL (Instruction list) дословно – список инструкций. Это типичный ассемблер с аккумулятором и переходами по меткам. Набор инструкций стандартизован и не зависит от конкретной целевой платформы.

Поскольку IL самый простой в реализации язык, он получил очень широкое распространение.

Наибольшее влияние на формирование современного IL оказал язык программирования STEP контроллеров фирмы Siemens. Язык IL позволяет работать с любыми типами данных, вызывать функции и функциональные блоки, реализованные на любом языке.

Таким образом, на IL можно реализовать алгоритм любой сложности, хотя текст будет достаточно громоздким.

В составе МЭК-языков IL применяется при создании компактных компонентов, требующих тщательной проработки, на которую не жалко времени. При работе с IL гораздо адекватнее, чем с другими языками, можно представить, как будет выглядеть оттранслированный код. Благодаря чему IL выигрывает там, где нужно достичь наивысшей эффективности. К компиляторам это относится в полной мере. В системах исполнения с интерпретатором промежуточного кода выигрыш не столь значителен.

Текст на IL – это текстовый список последовательных инструкций. Каждая инструкция записывается на отдельной строке. Инструкция может включать четыре поля, разделенные пробелами или знаками табуляции:

Метка:	Оператор	Операнд	Комментарий

Метка инструкции не является обязательной, она ставится только там, где нужно. Оператор присутствует обязательно. Операнд необходим почти всегда. Комментарий – необязательное поле, записывается в конце строки.

Ставить комментарии между полями инструкции нельзя.

Пример IL-программы:

```
METKA1:  LD  Sync  (*пример IL*)
         AND  Start
S  Q
```

Для лучшего восприятия строки IL выравнивают обычно в колонки по полям. Редактор CoDeSys выравнивает текст автоматически. Помимо этого, редактор выполняет синтаксический контроль и выделение цветом операторов и переменных. Так, корректно введенные операторы выделяются голубым цветом.

Аккумулятор.

Абсолютное большинство инструкций IL выполняют некоторую операцию с содержимым аккумулятора. Операнд, конечно, тоже принимает участие в инструкции, но результат опять помещается в аккумулятор.

Например, инструкция SUB 10 отнимает число 10 от значения аккумулятора и помещает результат в аккумулятор. Команды сравнения сравнивают значение операнда и аккумулятора, результат сравнения ИСТИНА или ЛОЖЬ вновь помещается в аккумулятор. Команды перехода на метку способны анализировать аккумулятор и принимать решение – выполнять переход или нет.

Аккумулятор IL является универсальным контейнером, способным сохранять значения переменных любого типа.

В аккумулятор можно поместить значение типа BOOL, затем INT или REAL, транслятор не будет считать это ошибкой. Такая гибкость не означает, что аккумулятор способен одновременно содержать несколько значений разных типов. Только одно, причем тип значения также фиксируется в аккумуляторе. Если операция требует значение другого типа, транслятор выдаст ошибку.

В стандарте МЭК вместо термина «аккумулятор» используется термин «результат» (result). Так, инструкция берет «текущий результат» и формирует «новый результат». Тем не менее почти все руководства по программированию различных фирм широко используют термин «аккумулятор».

Переход на метку.

Программа на IL выполняется подряд сверху вниз. Для изменения порядка выполнения и организации циклов применяется переход на метку. Переход на метку может быть безусловным JMP − выполняется всегда, независимо от чего-либо. Условный переход JMPC выполняется только при значении аккумулятора ИСТИНА.

Переход можно выполнять как вверх, так и вниз. Метки являются локальными, другими словами, переход на метку в другом POU не допускается. Переходы нужно организовывать достаточно аккуратно, чтобы не получить бесконечный цикл.

Скобки.

Последовательный порядок выполнения команд IL можно изменять при помощи скобок. Открывающая скобка ставится в инструкции после операции. Закрывающая скобка ставится в отдельной строке.

Инструкции, заключенные в скобки, выполняются в первую очередь. Результат вычисления инструкций в скобках помещается в дополнительный аккумулятор, после чего выполняется команда, содержащая открывающую скобку. Например:

```
LD 1
ST  Counter
LD  5
MUL   (2
SUB  1
)
ST  y    (*y = 5* (2 − 1) = 5*)
LD 5
MUL 2
SUB 1
ST y    (*y = 5 * 2 − 1 = 9*)
```

Скобки могут быть вложенными. Каждое вложение требует организации некоего временного аккумулятора. Это вызывает неоднозначность при выходе из блока скобок командами JMP, RET, CAL и LD. Применять эти команды в скобках нельзя.

Модификаторы.

Добавление к мнемонике некоторых операторов символов-модификаторов «C» и «N» модифицирует смысл инструкции.

Символ «N» (negation) вызывает инверсию значения операнда до выполнения инструкции. Операнд должен быть типов BOOL, BYTE, WORD или DWORD.

Символ «C» (condition) добавляет проверку условий к командам перехода, вызова и возврата. Команды JMPC, CALC, RETC будут выполняться только при значении аккумулятора ИСТИНА. Добавление символа «N» приводит к сравнению условия с инверсным значением аккумулятора. Команды JMPCN, CALCN, RETCN будут выполняться только при значении аккумулятора ЛОЖЬ. Модификатор «N» без «C» не имеет смысла в данных операциях и не применяется.

Операторы.

Стандартные операторы IL с допустимыми модификаторами представлены в таблице 1.1.

Таблица 1.1 − Список операторов

Оператор	Модификатор	Описание
LD	N	Загрузить значение операнда в аккумулятор
ST	N	Присвоить значение аккумулятора операнду
S		Если аккумулятор ИСТИНА, установить логический операнд (ИСТИНА)
R		Если аккумулятор ИСТИНА, сбросить логический операнд (ЛОЖЬ)
AND	N, (	Поразрядное И
OR	N, (	Поразрядное ИЛИ
XOR	N, (	Поразрядное ИЛИ
NOT		Поразрядная инверсия аккумулятора
ADD	(	Сложение
SUB	(	Вычитание
MUL	(	Умножение
DIV	(	Деление
MOD	(	Деление по модулю
GT	(	>
GE	(	=>
QE	(	=
NE	(	< >
LE	(	<=
LT	(	<
JMP	CN	Переход к метке
CAL	CN	Вызов функционального блока
RET	CN	Выход из POU и возврат в вызывающую программу

Операторы S и R применяются только с операндами типа BOOL. Прочие операторы работают с любыми переменными базовых типов.

Приведенный список содержит операторы, поддерживаемые в обязательном порядке. Трансляторы кода CoDeSys для различных аппаратных платформ реализуют различные подмножества дополнительных операторов.

Вызов функций.

В программах на языке IL функция вызывается по имени функции как оператор. Входные переменные функции задаются в виде списка параметров, следующих за именем функции, при этом первый параметр не указывается. В качестве первой переменной функции используется значение аккумулятора.

После выполнения функции результат функции сохраняется в аккумуляторе.

Пример обращения к функции LIMIT_REAL с тремя входными переменными, которым присваиваются значения переменных A, B, C:

```
LD A
LIMIT_REAL B, C
ST result
```

Если значение первой переменной, которое будет обработано, находится уже в аккумуляторе, команда загрузки LD может быть опущена.

Если обработка результата должна продолжиться, команда сохранения ST может быть также опущена.

Вызов функциональных блоков.

Перед вызовом функциональных блоков в программах на языке IL они должны быть объявлены. В этом объявлении каждому применяемому экземпляру функционального блока присваивается имя и тип.

Вызов функциональный блока происходит с помощью команды CALL и следующим за командой именем экземпляра функционального блока. Каждый экземпляр может вызываться только один раз.

Значения входных переменных функционального блока задаются двумя способами:

1) списком входных параметров, следующим за оператором CALL и именем;

2) присваиванием перед оператором CALL входным переменным значений входных параметров.

Пример вызова функционального блока CTU с именем экземпляра COUNT и входными переменными CU, RESET, PV:

```
CALL COUNT (CU :=Sb1, RESET := FALSH, PV :=125)
```

Присваивание входным переменным значений входных параметров производится последовательность команд, состоящим из команд загрузки фактических параметров и сохранения во входных переменных. Порядок загрузки и сохранения параметров не имеет значения.

Для обращения к входным переменным функционального блока необходимо указать имя экземпляра функционального блока, следующей за ней точкой и именем переменной:

<Имя экземпляра>.<Имя входной переменной >

Пример вызова функционального блока CTU с именем экземпляра COUNT и входными переменными CU, RESET, PV с предварительной загрузкой и сохранением входных параметров:

```
LD Sb1
ST COUNT.CU
LD FALSH
```

```
ST COUNT. RESET
LD 125
ST COUNT. PV
CALL COUNT
```

Выходы функционального блока могут использоваться в программе как переменные, доступные только для чтения. Для обращения к выходным переменным функционального блока необходимо указать имя экземпляра функционального блока, следующей за ней точкой и имя переменной:

<Имя экземпляра>.<Имя выходной переменной >

Пример использования выхода функционального блока CTU с именем экземпляра COUNT и выходными переменными Q, CV:

```
LD   COUNT.Q
ST   KM1
LD   COUNT.CV
ST   CCC
```

В языке IL задать актуальные параметры и считать значения выходов можно непосредственно при вызове экземпляра функционального блока. Для входных переменных применяется присваивание «:=», выходы считываются при помощи «=>». Этот процесс упрощается, если использовать «Ассистент ввода» (Input Assistant)(<F2>) с включенной опцией «Вставка с аргументами» (With arguments).

Содержание отчета

1 Титульный лист установленного образца.
2 Цель работы.
3 Задание.
4 Список инструкций языка программирования IL.
5 Листинг программы.
6 Ответы на контрольные вопросы.
7 Вывод по лабораторной работе.

Контрольные вопросы

1 Какие типы переменных могут обрабатываться операторами языка IL системы CoDeSys?
2 Какие операторы языка IL могут использоваться для вывода управляющих сигналов?
3 Каким образом организуются переходы в управляющих программах, написанных на языке IL?

4 Как изменяется значение переменной типа REAL, если к оператору добавлен модификатор N?

5 Как вставить комментарии в управляющую программу на языке IL?

6 Как вставить функциональный блок в управляющую программу на языке IL?

7 Как организовать выдержку времени в управляющей программе на языке IL?

8 Какие строки программы на языке IL могут устанавливаться как точки останова?

9 Как установить точки останова при отладке программы на языке IL?

10 Как определить значение переменной при отладке программы на языке IL?

2 Лабораторная работа № 8. Программирование логических контроллеров на языке FBD в системе CoDeSys

Цель работы: изучить принципы составления прикладных программ для промышленных логических контроллеров (ПЛК) на языке FBD пакета CoDeSys; приобрести навыки программирования на языке FBD в системе CoDeSys.

Задание

1 Изучить основные правила составления управляющих программ на языке FBD в системе CoDeSys.

2 Изучить порядок ввода редактирования и отладки управляющих программ на языке FBD в системе CoDeSys.

3 Изучить состав стандартных операторов и стандартных функций системы программирования CoDeSys.

4 Изучить состав функциональных блоков библиотек Util.lib.

5 Изучить пример и задание к лабораторной работе.

6 Разработать схему подключения кнопок, датчиков и реле к контроллеру.

7 Разработать алгоритм и программу управления тепловой пушкой.

8 Проверить работу управляющей программы в режиме эмуляции.

9 Записать программу в память контроллера и проверить ее выполнение.

Объект управления.

Схема тепловой пушки представлена на рисунке 2.1.

Воздух вентилятором *1* прогоняется через тепловую пушку. В зависимости от режима работы включается определенное количество нагревательных элементов *2*, датчик температуры *3* контролирует температуру воздуха на выходе пушки.

Необходимо разработать схему, позволяющую реализовать автоматическое управление тепловой пушкой по заданному алгоритму.

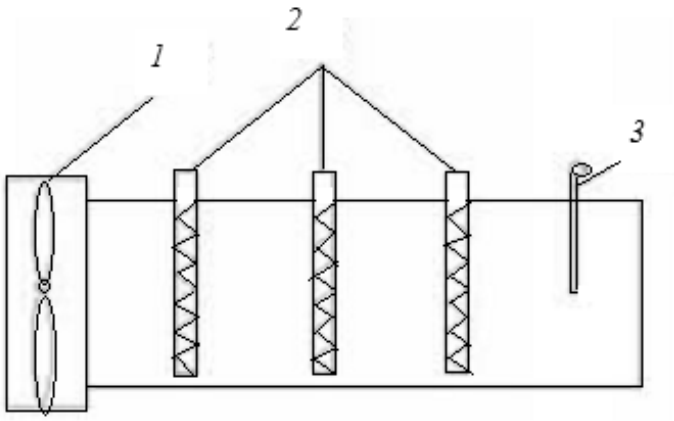


Рисунок 2.1 − Схема тепловой пушки

Варианты заданий

1 Кратковременное нажатие кнопки запускает вентилятор, через 3 с включается первый нагреватель. Если температура воздуха через 5 с ниже 50 °С, включается второй нагреватель, затем − третий. Если температура превышает 80 °С, нагреватели отключаются. Нажатие второй кнопки отключает тепловую пушку.

2 Кратковременное нажатие кнопки запускает вентилятор, через 4 с включаются три нагревателя. Если температура воздуха через 5 с выше 50 °С, один нагреватель отключается, если температура выше 60 °С, отключаются два нагревателя. Если температура превышает 90 °С, нагреватели отключаются. Нажатие второй кнопки отключает тепловую пушку.

3 Кратковременное нажатие кнопки запускает вентилятор, через 3 с включается первый нагреватель. Если температура воздуха через 5 с ниже 50 °С, включается второй нагреватель, если ниже 40 °С – второй и третий нагреватели. Если температура превышает 80 °С, последний включенный нагреватель отключается. Нажатие второй кнопки отключает нагреватели и через 4 с вентилятор.

При разработке схемы управления необходимо учесть, что нагреватели не должны включаться при отключенном вентиляторе.

Рекомендации к выполнению задания

Для выполнения задания необходимо обратиться к руководству фирмы ОВЕН (на компьютере: раздел 6 Документация, 04 CoDeSys) «Руководство пользователя по программированию ПЛК в CoDeSys 2.3». (UserManual_V23_RU.pdf) и оперативной справочной системе CoDeSys.

FBD – это графический язык программирования. Он работает с последовательностью цепей, каждая из которых содержит логическое или арифметическое выражение, вызов функционального блока, переход или инструкцию возврата.

Диаграмма FBD строится из компонентов, отображаемых на схеме прямоугольниками. Входы POU изображаются слева от прямоугольника, выходы – справа. Внутри прямоугольника указывается тип POU и наименования входов и выходов. Для экземпляра функционального блока его наименование указывается сверху над прямоугольником. В графических системах программирования прямоугольник компонента может содержать картинку, отражающую его тип. Размер прямоугольника зависит от числа входов и выходов и устанавливается графическим редактором автоматически. На рисунке 2.2 представлен пример представления экземпляра функционального блока PID.

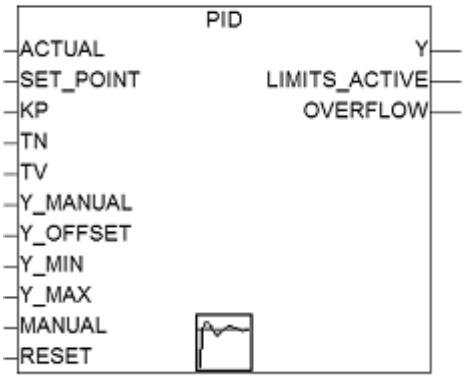


Рисунок 2.2 – Пример графического представления экземпляра функционального блока PID

Программа в FBD не обязательно должна представлять большую единую схему. Как и в LD, диаграмма образуется из множества цепей, которые выполняются одна за другой.

В CoDeSys все цепи одного POU отображаются в едином графическом окне, пронумерованные и разделенные горизонтальными линиями (рисунок 2.3).

Значения переменных, вычисленные в одной цепи, доступны в последующих цепях сразу в том же рабочем цикле.

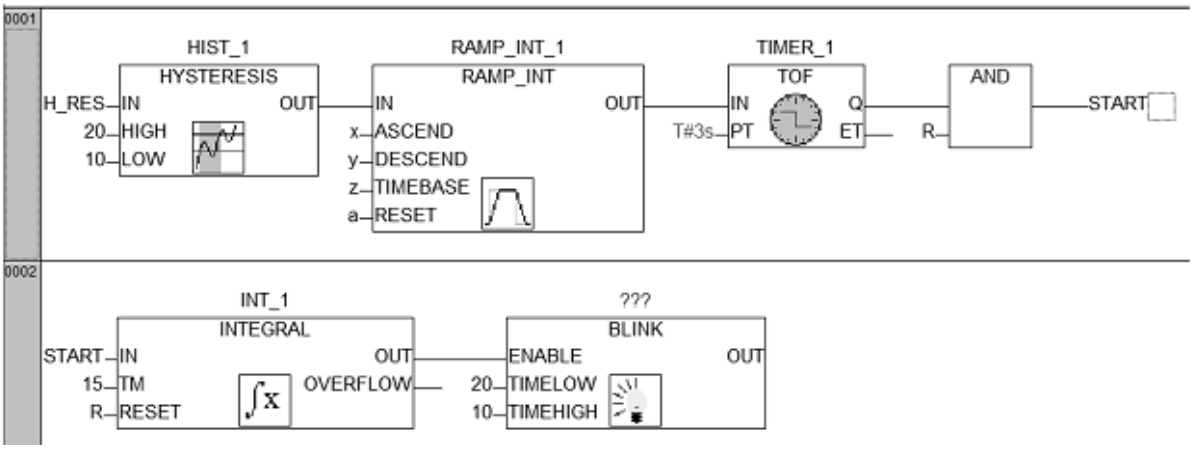


Рисунок 2.3 – Диаграмма FBD из двух цепей

Состав графических блоков языка FBD.

Основными графическими блоками языка FBD являются стандартные операторы, функции, функциональные блоки и программы. Стандартные операторы и стандартные функции являются частью системы программирова-ния CoDeSys, логика их работы (программа) написана на языке СИ и не может быть изменена (изменять можно только их параметры). Часто используемые функциональные блоки включены в библиотеки Standart.lib и Util.lib, которые также входят в состав системы программирования CoDeSys. Кроме них в программах можно использовать функции, функциональные блоки и программы пользователя, которые могут входить в подключаемые библиотеки либо разрабаты-ваться пользователем на любом из языков программирования среды CoDeSys.

Пример программы на языке FBD в среде CODESYS представлен на рисунке 2.4.

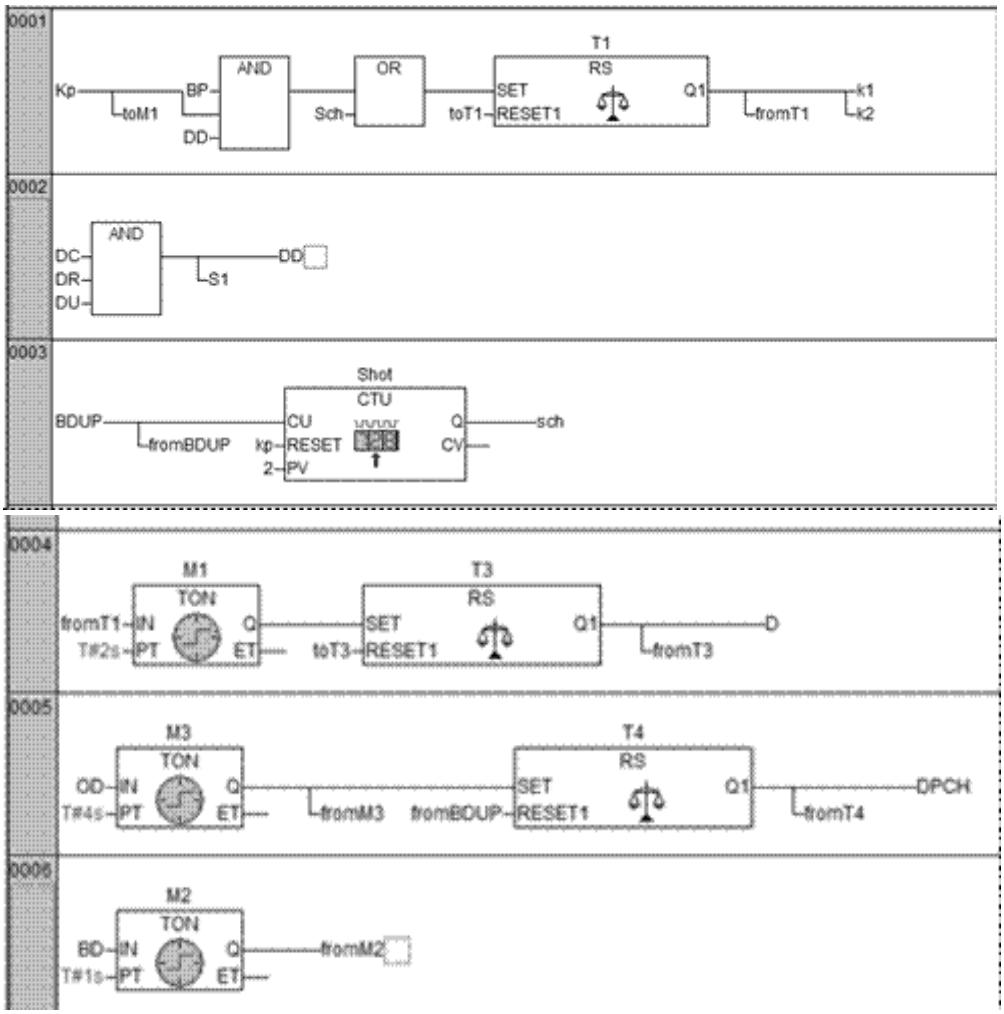
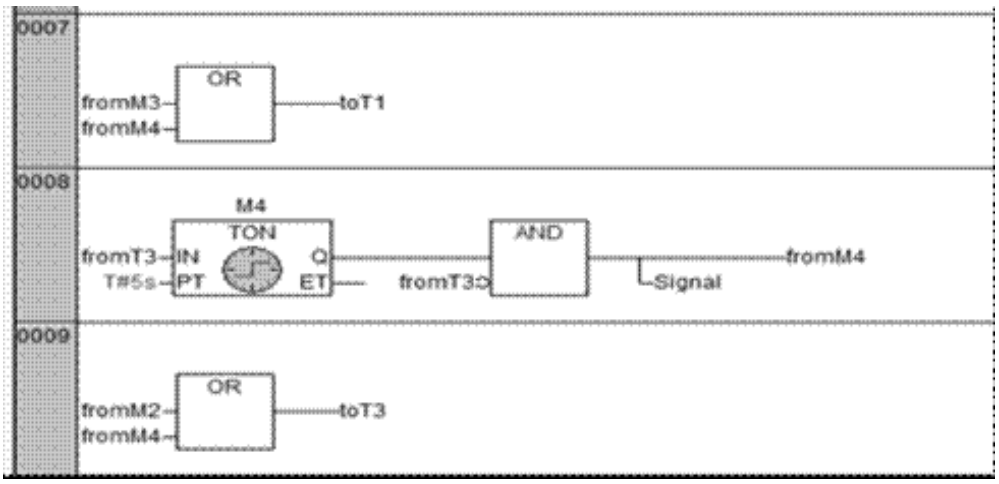


Рисунок 2.4 – Листинг программы на языке FBD



Окончание рисунка 2.4

Описание переменных представлено на рисунке 2.5.

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     T1: RS;
0004     Kp: BOOL;
0005     k1: BOOL;
0006     k2: BOOL;
0007     DC: BOOL;
0008     DR: BOOL;
0009     DU: BOOL;
0010     BP: BOOL;
0011     DD: BOOL;
0012     toM1: BOOL;
0013     Sch: BOOL;
0014     T3: RS;
0015     D: BOOL;
0016     OD: BOOL;
0017     M1: TON;
0018     M3: TON;
0019     T4: RS;
0020     DPCH: BOOL;
0021     M2: TON;
0022     BD: BOOL;
0023     fromM2: BOOL;
0024     BDUP: BOOL;
0025     Shot: CTU;
0026     fromM3: BOOL;
0027     toT1: BOOL;
0028     fromT3: BOOL;
0029     fromM4: BOOL;
0030     M4: TON;
0031     toT3: BOOL;
0032     fromBDUP: BOOL;
0033     toS2: BOOL;
0034     Signal: BOOL;
0035     S2: BOOL;
0036     fromT1: BOOL;
0037     fromT4: BOOL;
0038 END_VAR
```

Рисунок 2.5 – Описание переменных

Приведенные выше переменные описываются в области объявления переменных программы PLC_PRG и используются в качестве вспомогательных переменных. Все переменные принадлежат к типу BOOL.

Содержание отчета

1 Титульный лист установленного образца.
2 Цель работы.
3 Задание.
4 Список стандартных операторов и основных стандартных функций и функциональных блоков.

5 Описание функциональных блоков библиотек Util.lib, предназначенных для обработки аналоговых сигналов (PD, PID, GEN, RAMP_INT, RAMP_REAL, LIMITALARM).

6 Разработанная схема и листинг программы.

7 Ответы на контрольные вопросы.

8 Вывод по лабораторной работе.

Контрольные вопросы

1 Какие POU могут вставляться в программы языка FBD системы CoDeSys?

2 Какие блоки языка FBD допускают добавление входов и как это производится в системе CoDeSys?

3 Каким образом организуются переходы в управляющих программах, написанных на языке FBD?

4 Как изменяется последовательность выполнения программы при выполнении оператора RETURN?

5 Как вставить комментарии в управляющей программе на языке FBD?

6 Какие позиции может занимать курсор при составлении программы на языке программирования FBD?

7 Как изменить последовательность блоков в цепи программы на языке FBD?

8 Как изменить стиль и цвет соединительных линий в программе на языке FBD?

9 Как определить состояние выхода блока при отладке программы на языке FBD?

3 Лабораторная работа № 9. Создание визуализаций проекта в среде CoDeSys

Цель работы: изучить принципы создания визуализаций проекта в системе CoDeSys; приобрести навыки создания визуализаций в системе CoDeSys.

Задание

1 Изучить основные правила создания визуализаций в системе CoDeSys.

2 Изучить состав элементов визуализации в системе программирования CoDeSys.

3 Изучить порядок вставки, позиционирования и конфигурирования элементов визуализации в системе CoDeSys.

4 Изучить порядок конфигурирования объектов визуализации в системе CoDeSys.

5 Изучить пример к лабораторной работе.

6 Создать визуализацию для программы управления технологической установкой – насосным агрегатом (включение электродвигателя насоса при срабатывании датчика нижнего уровня, отключение насоса при срабатывании датчика верхнего уровня).

При создании визуализации предусмотреть индикацию состояния датчиков технологической установки, а также возможность управления установкой из визуализации.

7 Проверить работу визуализации при выполнении программы.

Рекомендации к выполнению задания

Визуализация представляет собой графическое изображение проектируемой системы, которое может служить пользовательским интерфейсом для контроля и управления работой системы.

Визуализация может исполняться в системе программирования, в отдельном приложении CoDeSys HMI, как Web-приложение на сервере или как целевая программа в ПЛК.

Редактор визуализации CoDeSys предоставляет набор готовых графических элементов, которые могут быть связаны соответствующим образом с переменными управляющей программы. Форма и цвет графических элементов могут изменяться при работе программы в зависимости от значений переменных.

Свойства отдельных элементов визуализации, а также визуализации в целом устанавливаются в соответствующих диалогах конфигурации и диалоге свойств объекта. Здесь определяется начальный вид элементов и выполняется привязка динамических свойств к значениям переменных проекта.

Создание файла визуализации.

Для создания визуализации нужно перейти на вкладку «Визуализации» организатора объектов, после чего правой кнопкой мыши вызвать контекстное меню в пустом поле и выбрать строку «Добавить объект», после чего появится окно визуализации (рисунок 3.1).

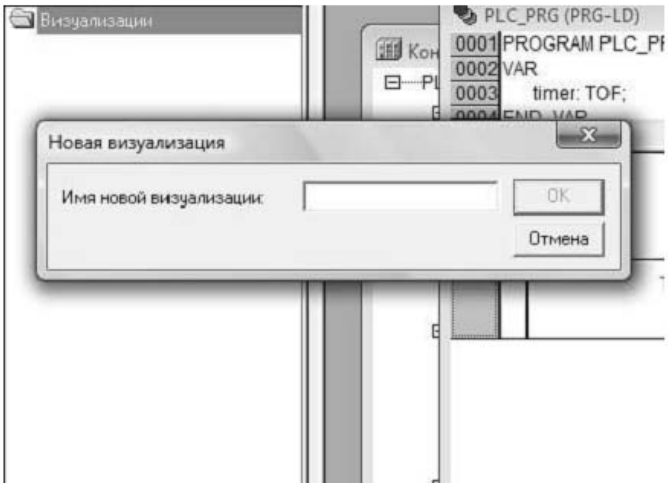


Рисунок 3.1 – Вызов окна визуализации

Элемент визуализации – это графический элемент, который используется при построении объекта визуализации. Возможные элементы представлены в виде иконок на панели инструментов CoDeSys (рисунок 3.2). Каждый элемент имеет собственную конфигурацию (набор свойств). Имеется возможность вставлять в визуализацию различные геометрические формы, а также точечные рисунки, метафайлы, кнопки и существующие визуализации.

Рисунок 3.2 – Панель инструментов редактора визуализации

У каждого элемента визуализации есть свои свойства. Вызвать свойства объекта визуализации можно двойным щелчком мыши по объекту, либо активизировать правой кнопкой мыши объект в контекстном меню и выбрать строку «Конфигурировать». В открывшемся окне можно задавать необходимые свойства объекта визуализации (рисунок 3.3).

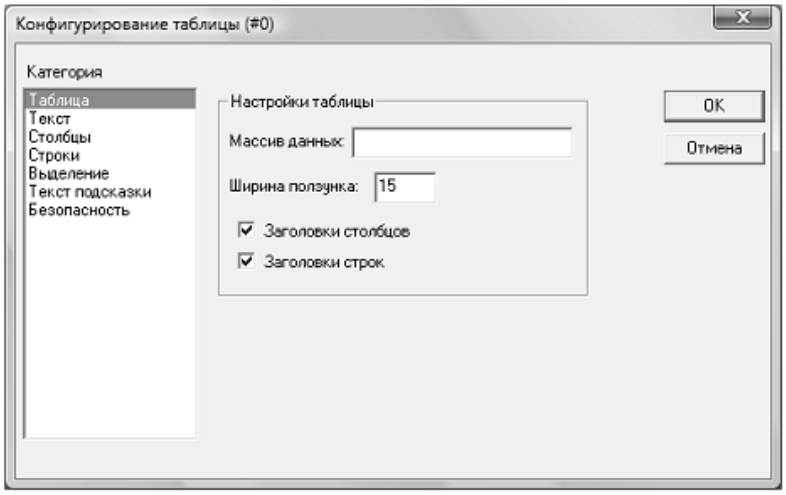


Рисунок 3.3 – Окно конфигурирования

Пример – Создание простейшей визуализации.

Создадим программу на языке LD в виде цепочки, состоящей из «Контакта» и «Катушки».

Перейдем на вкладку «Визуализация» и вызовем контекстное меню для добавления новой визуализации (рисунок 3.4).

В появившемся окне введем имя для визуализации (рисунок 3.5).

После ввода имени появляется новое окно для создания визуализации.

Вставляем с помощью панели инструментов элементы «Эллипс» и «Кнопку», как на рисунке 3.6.

Рисунок 3.4 – Добавление объекта визуализации

Рисунок 3.5 – Ввод имени визуализации

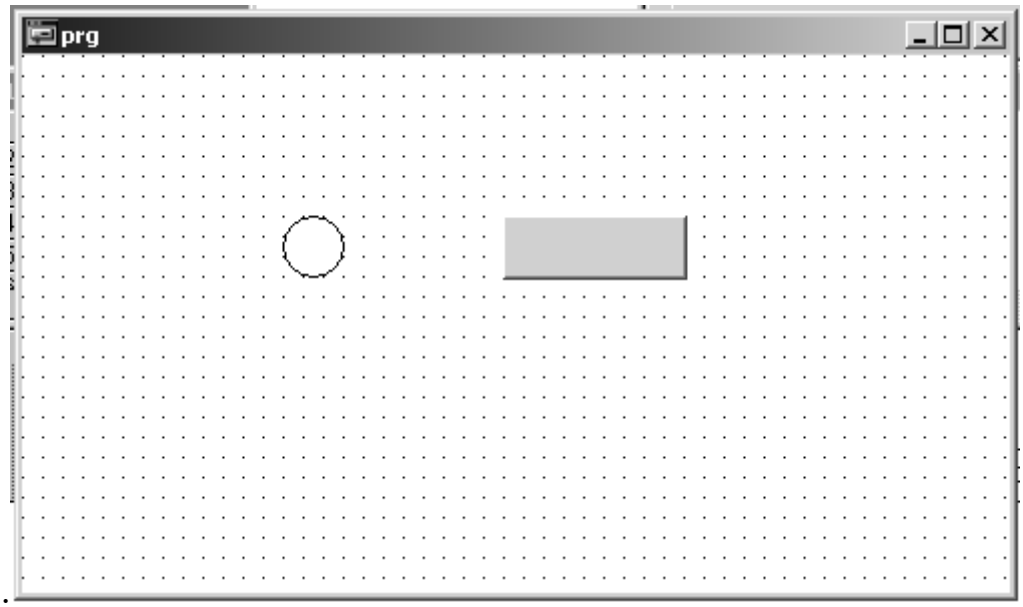


Рисунок 3.6 – Ввод элементов визуализации

Теперь необходимо сконфигурировать эти элементы. Пусть необходимо сделать следующую визуализацию: эллипс должен менять цвет, когда срабатывает катушка реле, а при нажатии на кнопку должна изменять значение переменная типа BOOL.

Сначала добавляем переменную C типа BOOL в проект (рисунок 3.7).

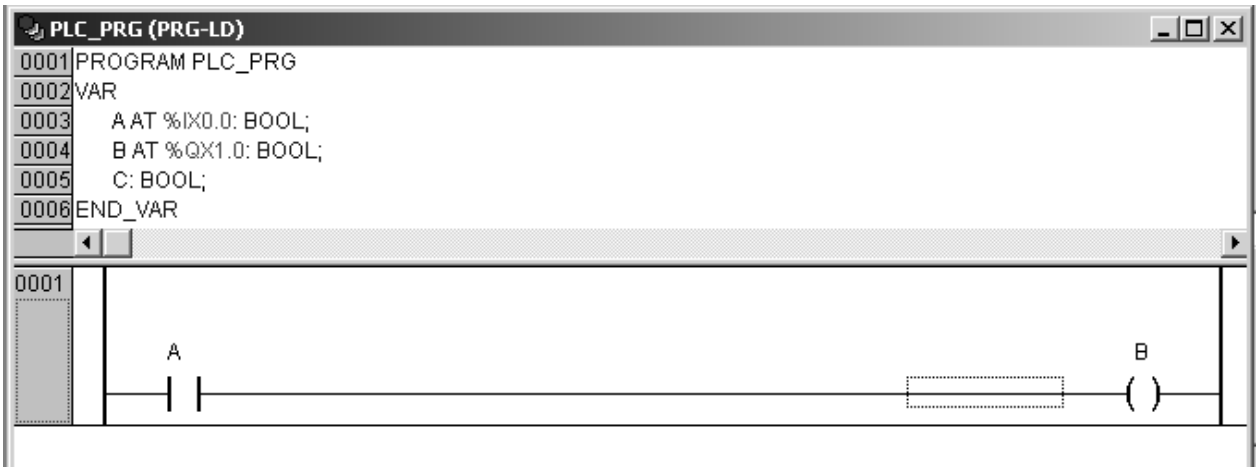


Рисунок 3.7 – Переменные проекта

Затем перейдем в окно редактора визуализации и щелкнем по эллипсу для вызова окна с настройками (рисунок 3.8).

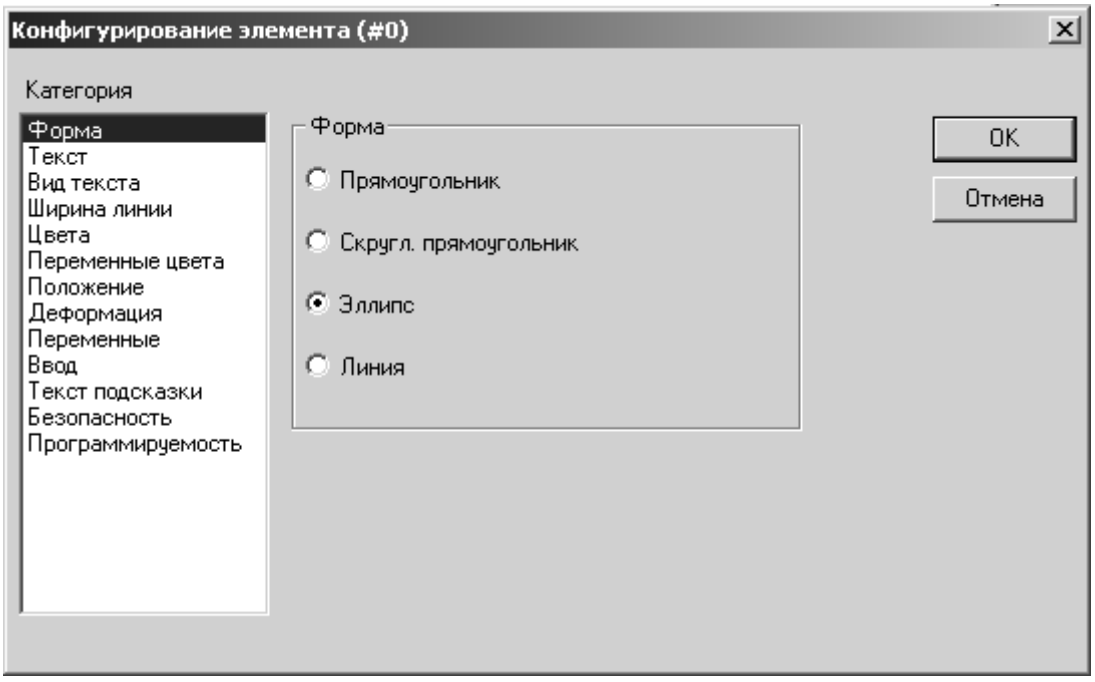


Рисунок 3.8 – Окно конфигурирования эллипса

Настроим категории «Цвета» и «Переменные». В категории «Цвета» выберем цвет эллипса в двух состояниях (рисунок 3.9).

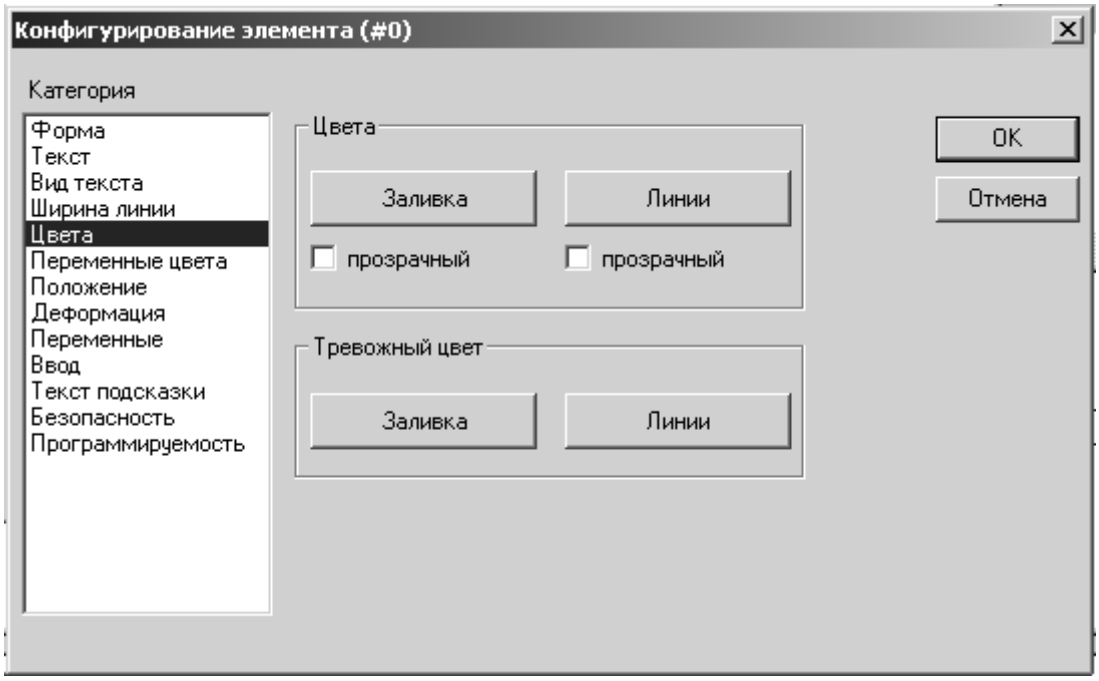


Рисунок 3.9 – Окно конфигурации цвета эллипса

Затем необходимо указать переменную, которая будет изменять цвет. Для этого перейдем в категорию «Переменные» и установим курсор в поле «Изм. цвета:» (рисунок 3.10).

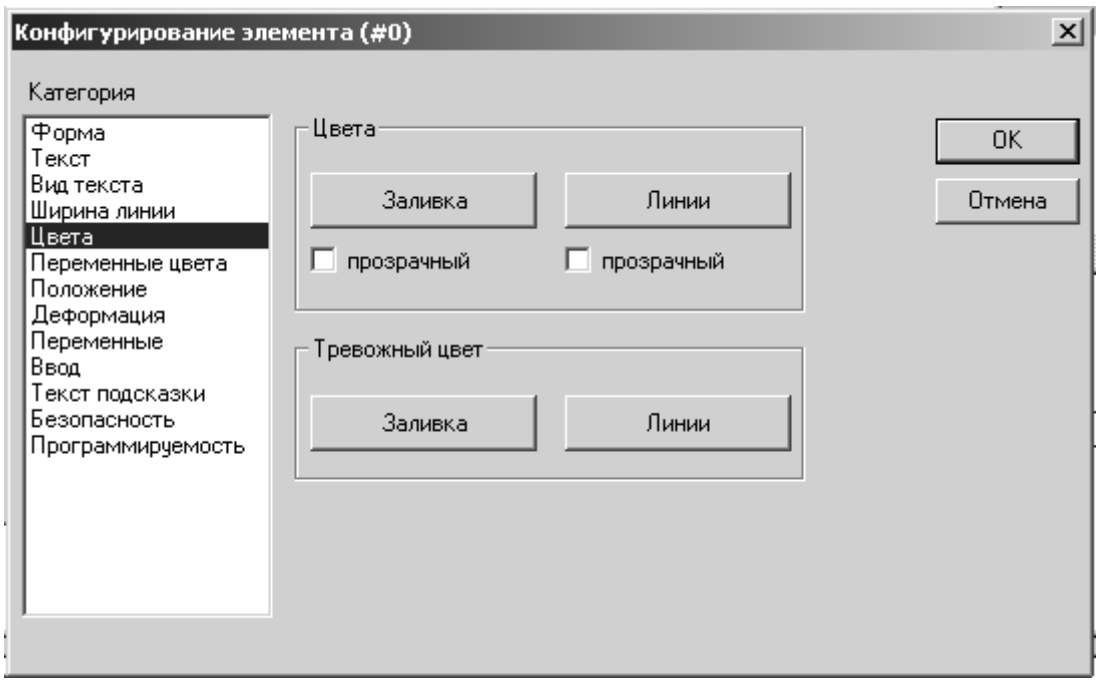


Рисунок 3.10 – Окно конфигурации переменных эллипса

Для связывания с переменными используем ассистент ввода (клавиша F2), в появившемся окне выберем нужную переменную – переменная B (рисунок 3.11).

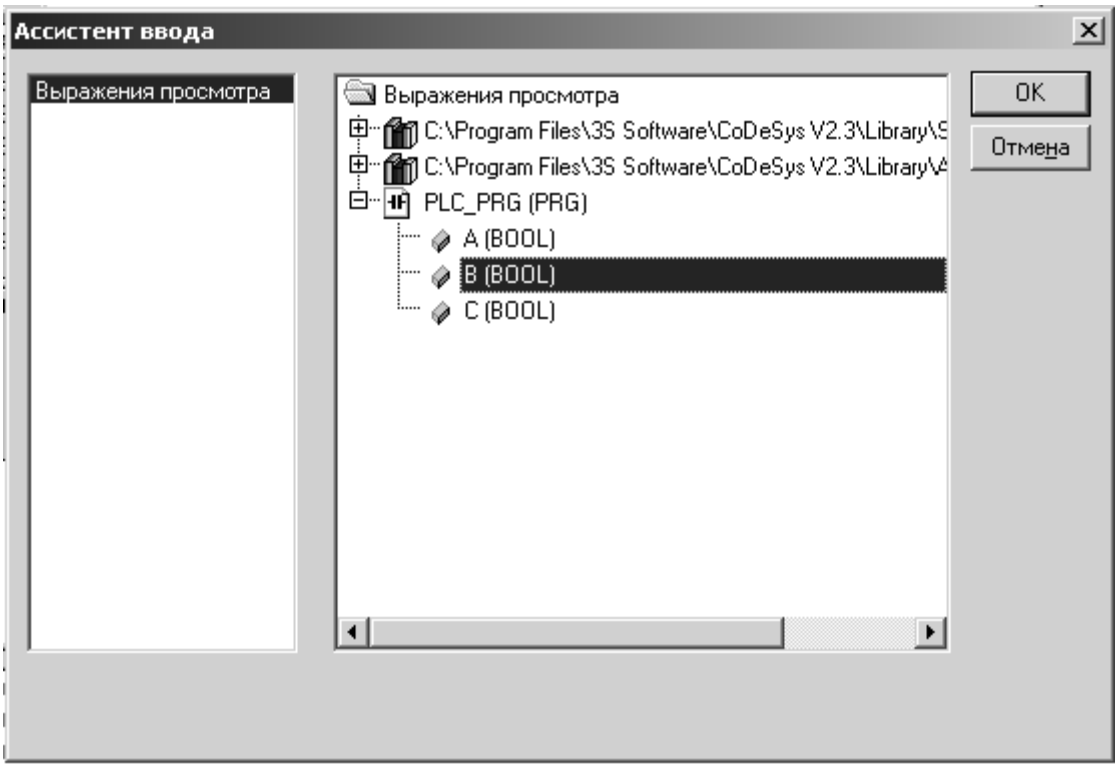


Рисунок 3.11 – Окно ассистента ввода

После нажатия кнопки OK в окне конфигурации переменных появится переменная для изменения цвета (рисунок 3.12).

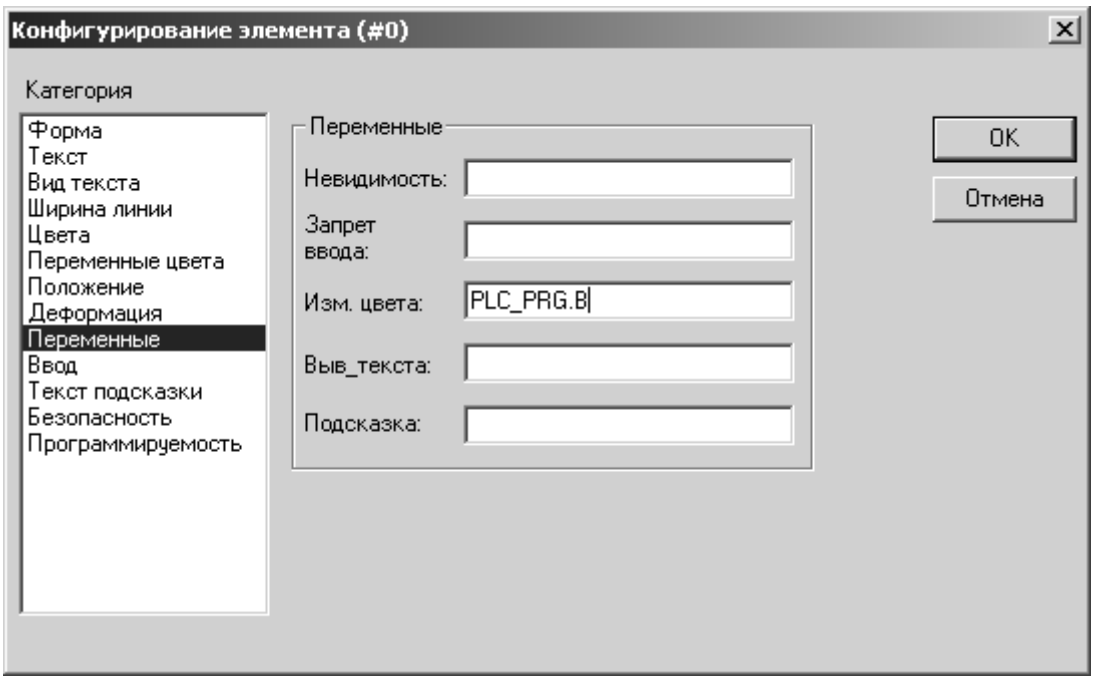


Рисунок 3.12 – Сконфигурированное окно переменных эллипса

Далее вызовем окно конфигурирования кнопки (рисунок 3.13) и выберем категорию «Ввод».

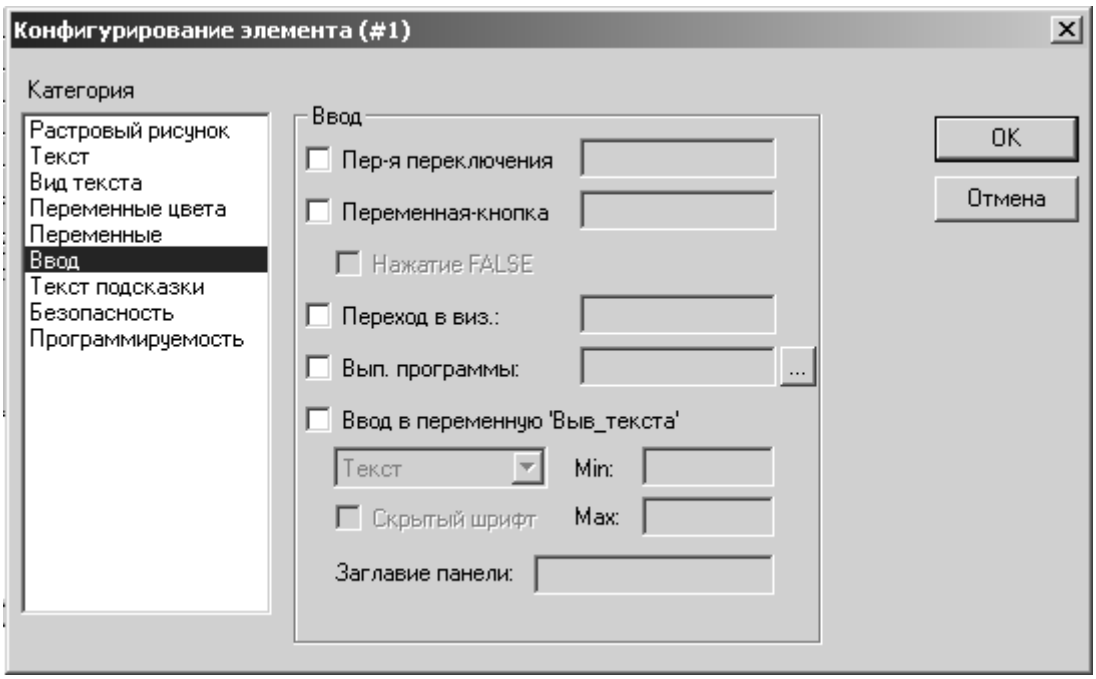


Рисунок 3.13 – Окно конфигурирования переменных кнопки

Поставим галочку «Переменная кнопка» (чтобы получить кнопку с фиксацией, необходимо выбирать «Переменная переключения») и, установив курсор в появившееся поле ввода, вызовем ассистент ввода, чтобы привязать к кнопке переменную C (рисунок 3.14).

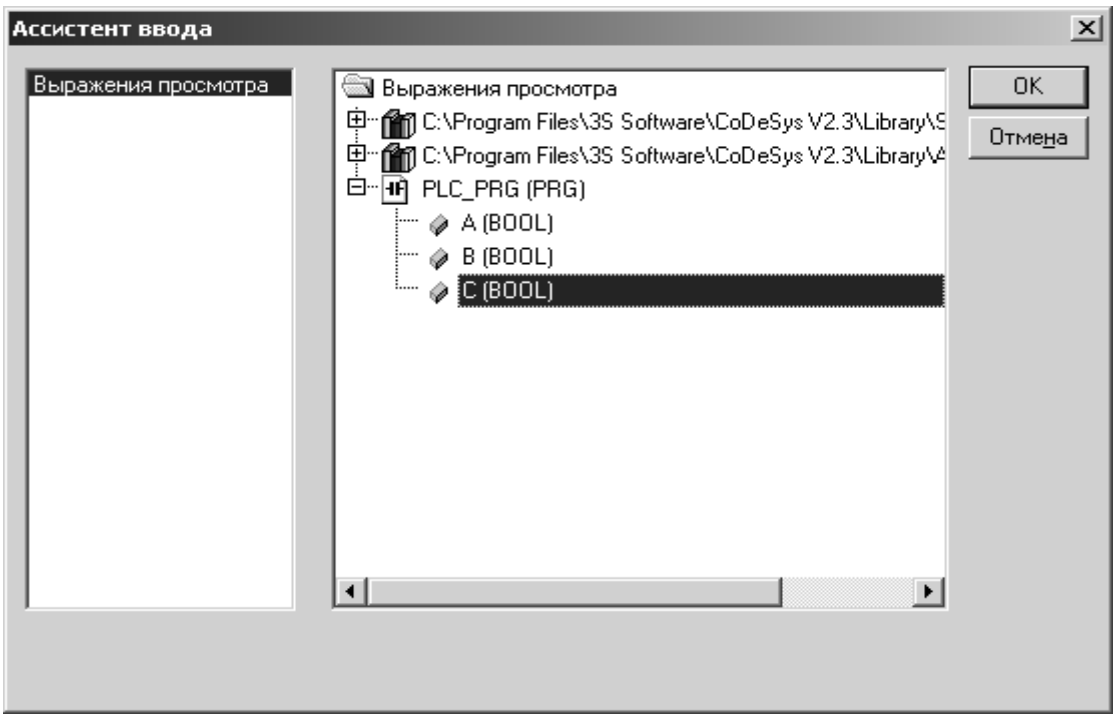


Рисунок 3.14 – Присваивание переменной

Сконфигурированное окно переменных показано на рисунке 3.15.

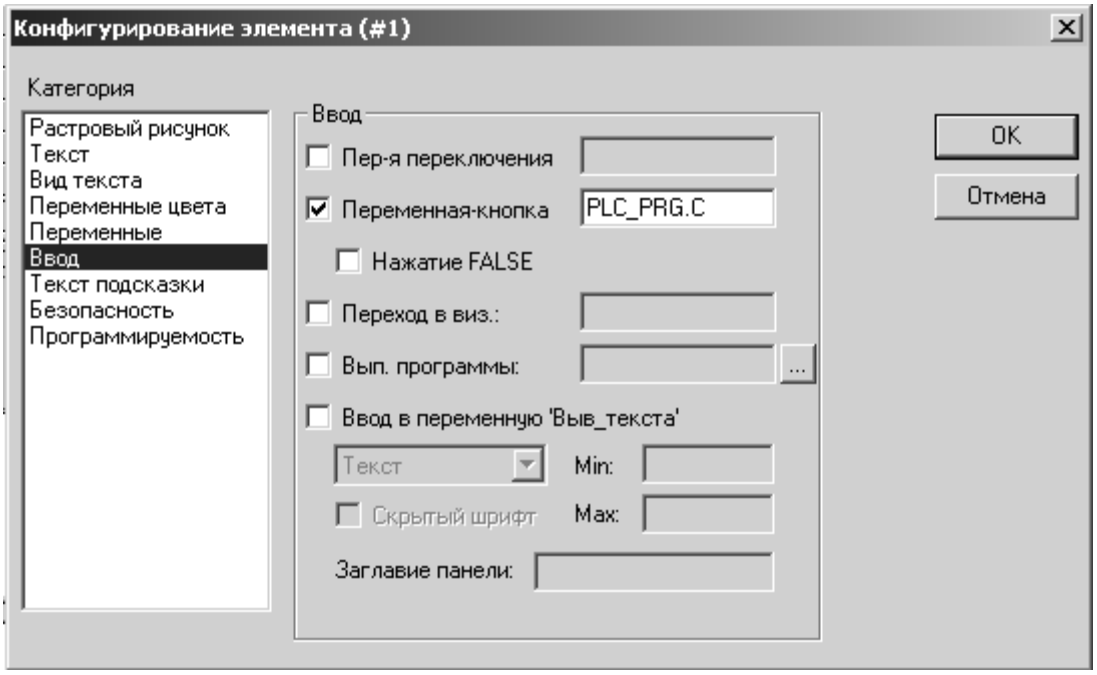


Рисунок 3.15 – Сконфигурированное окно переменных кнопки

Создание визуализации завершено. Можно загрузить программу в ПЛК и проверить работу визуализации.

Содержание отчета

1 Титульный лист установленного образца.
2 Цель работы.

3 Задание.
4 Описание последовательности создания визуализации.
5 Список элементов визуализации в системе CoDeSys.
6 Листинг программы.
7 Print Screen экрана визуализации.
8 Ответы на контрольные вопросы.
9 Вывод по лабораторной работе.

Контрольные вопросы

1 Какие формы визуализации реализованы в системе CoDeSys?
2 Что такое «объект визуализации» и «элемент визуализации» в системе CoDeSys ?
3 Какие элементы визуализации можно использовать при создании визуализаций в системе CoDeSys?
4 Каким образом осуществляется выбор и вставка элемента визуализации в системе CoDeSys?
5 Какие действия можно производить над элементом визуализации в системе CoDeSys?
6 Как одновременно переместить несколько элементом визуализации в системе CoDeSys?
7 Какие свойства элемента визуализации можно изменять при конфигурировании его в системе CoDeSys?
8 Как добавить текст к элементу визуализации в системе CoDeSys?
9 Какие свойства объекта визуализации можно изменять при конфигурировании его в системе CoDeSys?
10 Как установить фоновый рисунок при создании визуализации в системе CoDeSys ?

4 Лабораторная работа № 10. Разработка программ управления технологическими установками на языке SFC

Цель работы: изучить методики разработки программ управления технологическими установками для промышленных логических контроллеров (ПЛК) на языке SFC пакета CoDeSys; приобрести навыки разработки управляющих программ на языке SFC в системе CoDeSys.

Задание

1 Изучить принцип работы заданной технологической установки.
2 Разработать функциональную схему управления технологической установкой.
3 Выбрать элементы (кнопки, датчики, исполнительные устройства, индикаторы), необходимые для реализации схемы управления установкой.
4 Составить схему алгоритма управления установкой.

5 Разработать схему подключения выбранных элементов к программируемому контроллеру.

6 Составить таблицу входных и выходных сигналов контроллера с указанием подключаемого элемента, типа сигнала, символьного обозначения, адреса.

7 Составить управляющую программу, реализующую управление установкой в виде SFC-схемы.

8 Проверить работу управляющей программы в режиме эмуляции.

9 Записать программу в память контроллера и проверить ее выполнение.

Рекомендации к выполнению задания

Для выполнения задания необходимо обратиться к руководству фирмы ОВЕН (на компьютере: раздел 6 Документация, 04 CoDeSys) «Руководство пользователя по программированию ПЛК в CoDeSys 2.3». (UserManual_V23_RU.pdf) и оперативной справочной системе CoDeSys.

Устройство управления последовательным технологическим процессом может быть реализовано на базе программируемого логического контроллера (ПЛК).

Задача ПЛК в этом случае – обеспечить не только логику управления установкой, но и обеспечить заданную последовательность выполнения тактов.

Управляющую программу для таких систем рекомендуется составлять на стандартном языке программирования ПЛК – языке SFC.

Типовая последовательность разработки управляющей программы на языке SFC содержит следующие этапы:

– изучение технологической установки;

– выбор элементов управления пульта оператора;

– составления алгоритма управления установкой;

– составление таблиц входных и выходных сигналов контроллера;

– написание программы.

Разработку алгоритма необходимо начинать с анализа технологического процесса. При анализе работы технологического объекта следует выделить состояния, в которых может находиться оборудование в процессе работы, определить действия, выполняемые в каждом состоянии, и условия перехода между состояниями. Если технологический объект представляет собой комплексную установку, состоящую из нескольких функциональных устройств, управление которыми может осуществляться отдельно, то для каждого устройства определяется своя последовательность состояний. Результатом анализа должно быть описание работы оборудования в словесной или формализованной форме. Основные способы описания работы оборудования в формализованном виде:

– циклограмма работы оборудования;

– сетевой граф изменения состояний;

– схема алгоритма последовательности действий.

Схема алгоритма управления составляется в виде последовательности условных графических изображений действий и условий перехода. Действия изображаются в виде прямоугольников, условия переходов – в виде ромбов.

Алгоритм управления технологической установкой описывает процесс взаимодействия ПЛК с объектом и оператором. В отличие от алгоритма работы оборудования в нем указываются действия, которые необходимо выполнять контроллеру, чтобы обеспечить требуемое поведение технологической установки. Действия контроллера заключаются в обработке информационных сигналов датчиков, выдаче управляющих сигналов на исполнительные устройства, выполнении временных функций и функций счета, обслуживании пульта оператора.

Для разработки программы желательно составление таблиц входных и выходных сигналов программируемого контроллера. Таблицы должны содержать наименование и условное обозначение сигналов, их источник или приёмник, адресацию сигналов и их привязку к контактам разъёмов контроллера.

Завершающим этапом проектирования программно-логической подсистемы управления дискретным процессом является написание программы управления для программируемого контроллера, которая составляется на основании информации, подготовленной на предыдущих этапах, и ее отладка.

Пример разработки управляющей программы.

Объект управления представляет собой роботизированный технологический комплекс (РТК). В состав РТК входят (рисунок 4.1): промышленный робот, конвейер подающий, контейнер для складирования деталей.

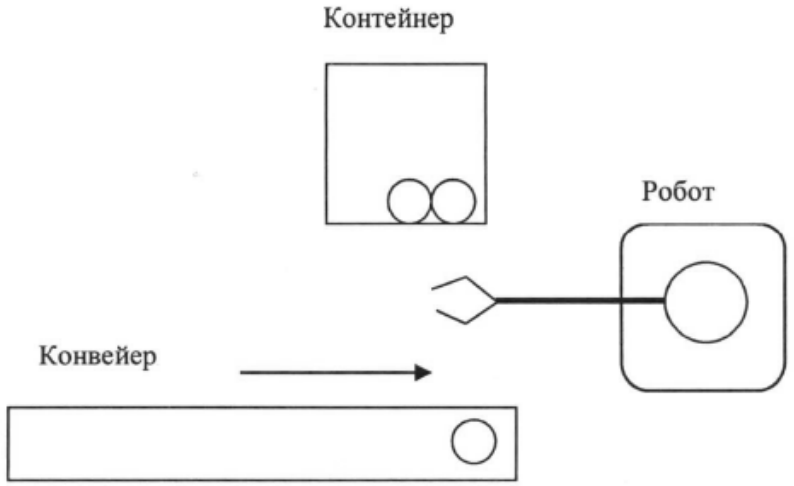


Рисунок 4.1 – Структурная схема РТК

В исходном состоянии РТК отключен и не выполняет никаких действий. Переключение в рабочее состояние происходит по сигналу оператора. В рабочем состоянии РТК выполняет следующие действия. Если деталь поступает на начало конвейера, конвейер включается и перемещает деталь к

конечной точке, затем конвейер отключается. Робот поворачивается, берет деталь и укладывает ее в контейнер. После поступления очередной детали цикл повторяется. Работа установки прекращается по сигналу оператора или при заполнении контейнера (емкость контейнера 50 деталей). Возобновляется работа после смены контейнера по сигналу оператора.

Привод конвейера осуществляется от асинхронного двигателя, в начале и конце конвейера установлены датчики контроля наличия детали. Робот приводится в движение с помощью пневмоцилиндров, которые управляются электроклапанами, установленными на распределительной пневмопанели. Крайние положения руки робота контролируются датчиками конечного положения. Еще один датчик конечного положения используется для контроля наличия контейнера в позиции складирования. Для включения-отключения РТК на панели оператора установлены две кнопки. Состояние оборудования можно определить по индикатору, который также находится на панели оператора. Светящийся индикатор указывает, что РТК находится в рабочем режиме.

Управление РТК осуществляется контроллером ПЛК-110-60, который расположен в шкафу электроавтоматики. Там же находится автоматический выключатель и пускатель двигателя конвейера.

Структурная схема системы управления РТК представлена на рисунке 4.2.

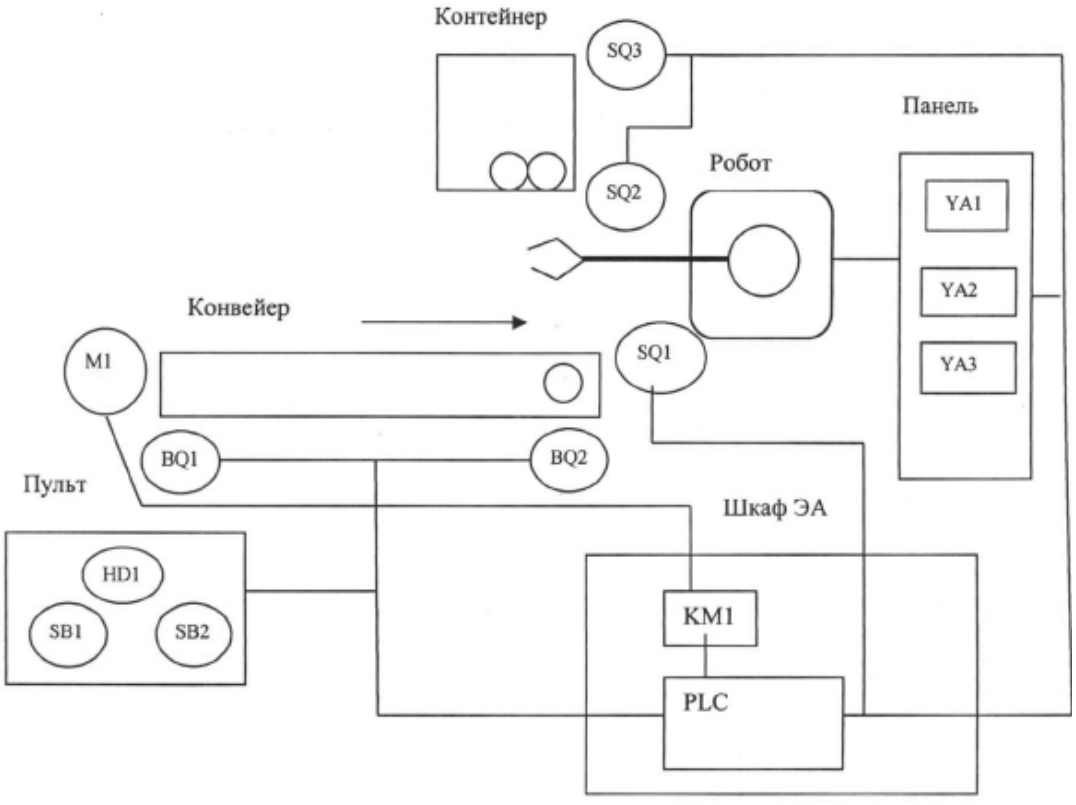


Рисунок 4.2 – Структурная схема системы управления РТК

В таблице 4.1 представлено назначение элементов РТК.

Таблица 4.1 – Назначение элементов РТК

Назначение датчиков РТК	Назначение исполнительных устройств	Назначение элементов пульта оператора
BQ 1 – наличие детали в начале конвейера BQ 2 – наличие детали в конце конвейера SQ 1 – робот в позиции захвата детали SQ 2 – робот в позиции складирования детали SQ 3 – наличие контейнера в позиции складирования	M1 – асинхронный двигатель привода конвейера YA1 – клапан пневмоцилиндра схвата YA2 – клапан пневмоцилиндра поворота робота в позицию захвата YA3 – клапан пневмоцилиндра поворота робота в позицию складирования	SB1 – кнопка ПУСК – включение РТК SB2 – кнопка СТОП – отключение РТК HD1 – индикатор РАБОТА

В составе оборудования РТК имеется два функциональных устройства, работающих практически независимо друг от друга: конвейер и робот. Это позволяет рассматривать работу каждого устройства в отдельности.

В цикле работы конвейера можно выделить четыре состояния:

1) конвейер отключен;

2) конвейер ожидает поступления детали в начальную позицию;

3) конвейер перемещает деталь в конечную позицию;

4) конвейер ожидает снятия детали.

Изменение состояния конвейера определяется сигналами кнопок пульта оператора и датчиков наличия детали на конвейере.

В исходном состоянии конвейер отключен. Переход в состояние ожидания детали происходит при наличии сигнала кнопки ПУСК (SB1 = 1) при условии, что контейнер не заполнен (CT < 50) и находится в позиции складирования (SQ3 = 1). При поступлении детали в начальную позицию (BQ1 = 1) включается двигатель (KM1 = 1) и деталь перемещается в конечную позицию (BQ2 = 1). Затем двигатель отключается (KM1 = 0) и конвейер ожидает снятия детали роботом. После снятия детали (BQ2 = 0) конвейер переходит в состояние ожидания поступления очередной детали. Отключение конвейера происходит при поступлении сигнала от кнопки СТОП (SB2 = 0) при условии, что он находится в режиме ожидания детали, или после заполнения контейнера (CT = 50).

В цикле работы робота можно выделить следующие состояния:

– робот отключен;

– робот ожидает поступления детали в позицию захвата;

– робот переносит деталь в контейнер.

Изменение состояния робота определяется сигналами кнопок пульта оператора и датчиком наличия детали в позиции захвата.

В исходном состоянии робот отключен. Переход в состояние ожидания детали происходит при наличии сигнала кнопки ПУСК (SB1 = 1) при условии, что контейнер не заполнен (CT < 50) и находится в позиции складирования (SQ3 = 1). При поступлении детали в позицию захвата (BQ2 = 1) робот переносит деталь в контейнер и переходит в состояние ожидания поступления

очередной детали. Отключение робота происходит при поступлении сигнала от кнопки СТОП (SB2 = 0) при условии, что он находится в режиме ожидания детали, или после заполнения контейнера (CT = 50).

Цикл переноса детали включает в себя следующие действия:

– поворот руки робота в позицию захвата (YA1 = 1);

– зажим схвата (YA3 = 1);

– поворот руки робота в позицию складирования (YA1 = 0, YA2 = 1);

– разжим схвата (YA3 = 0);

– поворот руки робота в исходное состояние (YA2 = 0).

Выполнение операций поворота руки робота подтверждается сигналами датчиков SQ1 и SQ2. Для повышения надежности при переносе детали необходимо предусмотреть выдержку времени между операциями разжима-зажима схвата и началом движения руки робота.

Схема алгоритма управления конвейером представлена на рисунке 4.3.

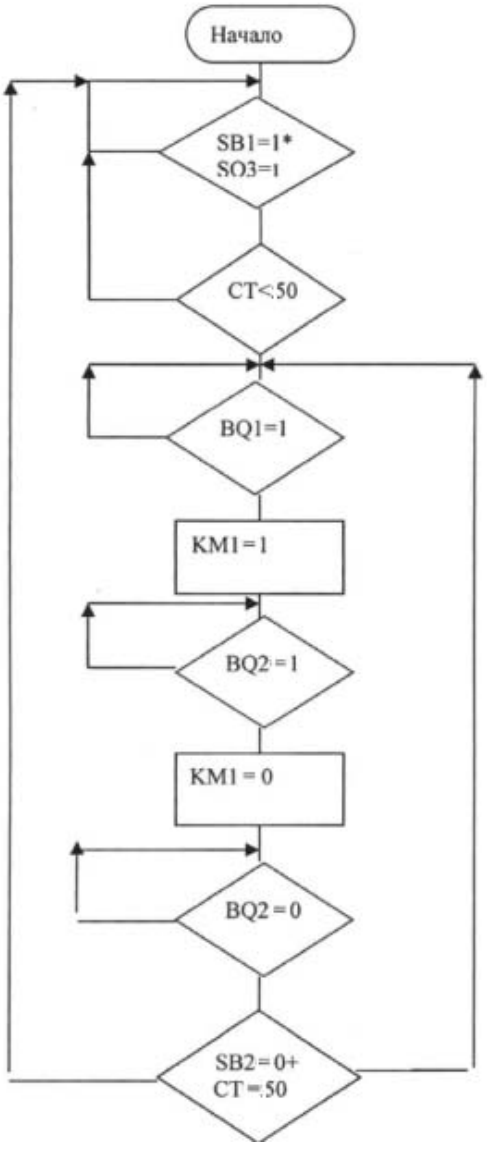


Рисунок 4.3 – Схема алгоритма управления конвейером

Схема алгоритма управления роботом представлена на рисунке 4.4.

Схема подключения пульта оператора, датчиков и исполнительных устройств к ПЛК проектируемой системы управления приведена на рисунке 4.5.

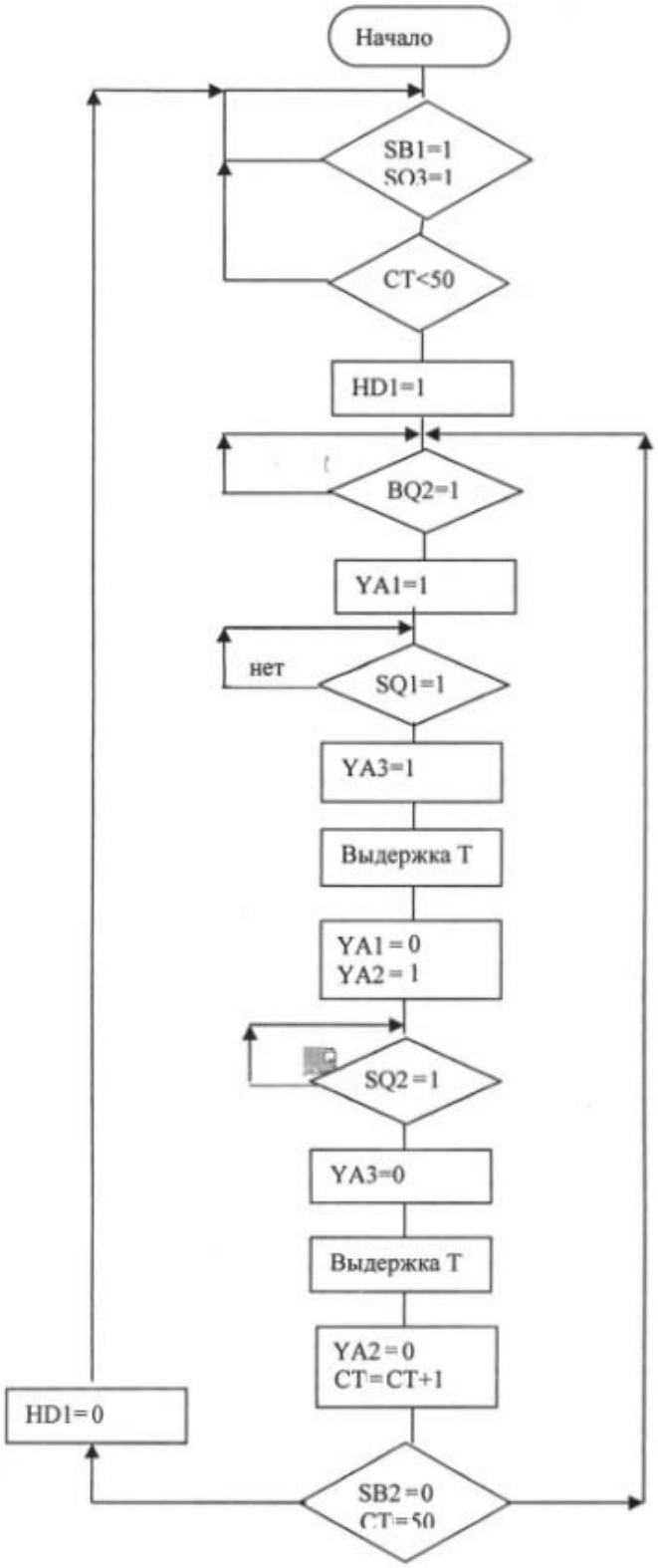


Рисунок 4.4 – Схема алгоритма управления роботом

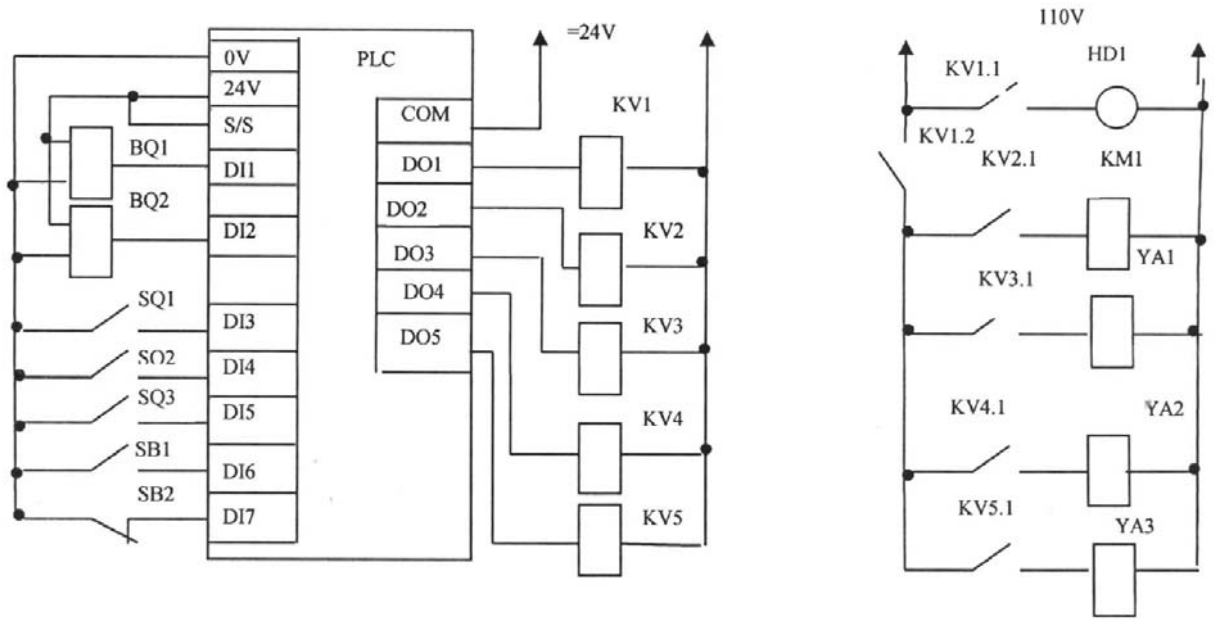


Рисунок 4.5 – Схема подключения ПЛК 110-60К

Входные сигналы ПЛК и соответствующие им символьные переменные, используемые в управляющей программе, приведены в таблице 4.2.

Таблица 4.2 – Таблица входных сигналов ПЛК

Номер входа	Подключаемый элемент	Тип сигнала	Символьное обозначение	Адрес
DI1	BQ1	BOOL	DETN	%IX0.0
DI2	BQ2	BOOL	DETK	%IX0.1
DI3	SQ1	BOOL	ROBZH	%IX0.2
DI4	SQ2	BOOL	ROBSKL	%IX0.3
DI5	SQ3	BOOL	KONT	%IX1.0.0
DI6	SB1	BOOL	PUSK	%IX1.0.1
DI7	SB2	BOOL	STOP	%IX1.0.2

Выходные сигналы ПЛК и соответствующие им символьные переменные, используемые в управляющей программе, приведены в таблице 4.3.

Таблица 4.3 – Таблица выходных сигналов ПЛК

Номер входа	Подключаемый элемент	Тип сигнала	Символьное обозначение	Адрес
DO1	KV1-HD1	BOOL	RAB	%QX2.0
DO2	KV2-KM1	BOOL	KONV	%QX2.1
DO3	KV3-YA1	BOOL	POVZH	%QX2.2
DO4	KV4-YA2	BOOL	POVSKL	%QX2.3
DO5	KV5-YA3	BOOL	SHVAT	%QX3.0.0

В таблице 4.4 приведены символьные переменные, объявляемые в управляющей программе.

Таблица 4.4 – Таблица символьных переменных

Символьное обозначение	Тип переменной	Адрес	Атрибут	Начальное значение	Комментарий
COUNT	BYTE	%MB10.0	RETAIN	0	Счетчик деталей

Управляющая программа, разработанная в виде схемы на языке SFC, представлена на рисунке 4.6.

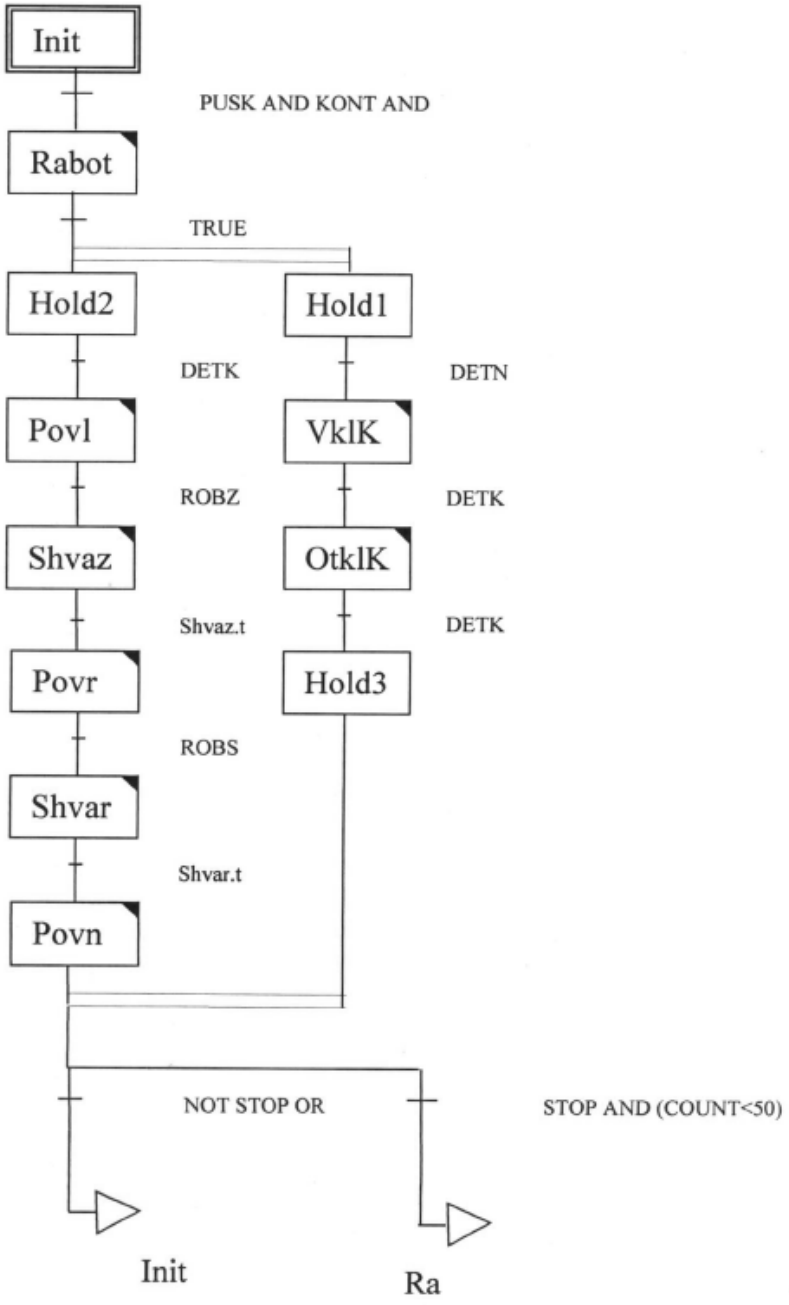


Рисунок 4.6 – Управляющая программа на языке SFC

Каждому шагу схемы SFC соответствует определенное состояние оборудования РТК. Управление конвейером и роботом в программе осуществляется в параллельных ветвях, что отражает их определенную функциональную независимость.

Действия, выполняемые в шаге, заключаются во включении-отключении исполнительных устройств РТК, т. е. в присваивании переменной, соответствующей выходному сигналу, значения 0 либо 1. Исполнительные устройства, на которые выдается управляющий сигнал, соответствуют устройствам, указанным в соответствующих блоках алгоритма управления РТК. Шаги Hold1, Hold2 и Hold3 являются пустыми, никаких действий они не выполняют и служат для перевода робота и конвейера в состояние ожидания.

Условия переходов между шагами определяются значением переменных входных сигналов датчиков и кнопок пульта оператора, а также переменной COUNT, значение которой равно количеству деталей в контейнере. Выдержка времени после зажима-разжима схвата реализована с помощью неявных переменных контроля времени активности шага Shvaz.t и Shvar.t.

Содержание отчета

1 Титульный лист установленного образца.
2 Цель работы.
3 Задание.
4 Функциональная схема управления технологической установкой.
5 Блок-схема алгоритма работы установки.
6 Схема подключения элементов управления к программируемому логическому контроллеру.
7 Таблица входных и выходных сигналов контроллера.
8 Таблица внутренних переменных с указанием символьного обозначения, типа сигнала и адреса.
9 Листинг программы.
10 Ответы на контрольные вопросы.
11 Вывод по лабораторной работе.

Контрольные вопросы

1 В каких случаях технологический процесс можно разделить на отдельные такты?
2 Какие элементы системы управления технологическим процессом рекомендуется располагать на пульте оператора?
3 Каким образом можно описать алгоритм работы технологической установки?
4 Чем характеризуется состояние технологического оборудования при описании его работы в виде сетевого графа?
5 Какую информацию рекомендуется помещать в таблицу входных-входных сигналов контроллера?

6 Чем отличается алгоритм работы технологического оборудования от алгоритма управления оборудованием?

7 Каким образом шаги схемы на языке SFC связаны с алгоритмом работы технологического оборудования?

5 Лабораторная работа № 11. Разработка программ управления электроприводами типовых механизмов

Цель работы: изучить особенности управления электроприводами станков с ЧПУ, принципы составления прикладных программ для станков с ЧПУ.

Задание

1 Изучить основные правила создания управляющей программы.

2 Изучить особенности линейной, круговой интерполяции.

3 Составить программу управления электроприводами для создания изображения буквы «А».

4 Составить программу управления электроприводами для создания изображения «Δ-треугольник»).

5 Ознакомиться с ГОСТ 20999−83 *Устройства числового программного управления для металлообрабатывающего оборудования. Кодирование информации управляющих программ*. Программирование в G- и M-кодах (язык программирования ISO 7bit).

5.1 Рекомендации к выполнению задания

5.1.1 Критерии выбора электроприводов

Тип приводного двигателя для станков выбирают по следующим характеристикам.

1 Производительность.

По этому параметру сервоприводы значительно превосходят шаговые электромоторы. На станок с ЧПУ для обработки крупных деталей или заготовок из твердых материалов лучше установить сервомотор. Такой электропривод обеспечит более высокую скорость обработки твердых материалов. Для малогабаритного промышленного оборудования среднего класса точности, предназначенного для обработки мягких материалов, лучше выбрать шаговый двигатель.

2 Эксплуатационные расходы.

Программирование исполнителя. Такой привод намного дороже в обслуживании, соответственно, расходы на его эксплуатацию будут выше.

3 Точность.

Сервоприводы для станков с ЧПУ необходимы для высокоточной автоматизированной обработки. Такой привод позволяет позиционировать положение рабочего органа с точностью до 0,02 мкм, в то время как максимальная точность шаговой электрической машины − 0,01 мм.

4 Цена.

Стоимость шагового двигателя значительно ниже цены сервопривода. Для работы с серводвигателем необходимо наличие специальных контроллеров и устройств обратной связи, что, несомненно, приводит к увеличению стоимости станка. При невысоком бюджете лучше предпочесть первый вариант.

5 Уровень шума.

По этому показателю сервомоторы предпочтительней. Работа шаговых электродвигателей сопровождается звуком, соответствующим частоте шагов на различных оборотах.

На рисунке 5.1 изображены конструкции шаговых электродвигателей и серводвигателей.

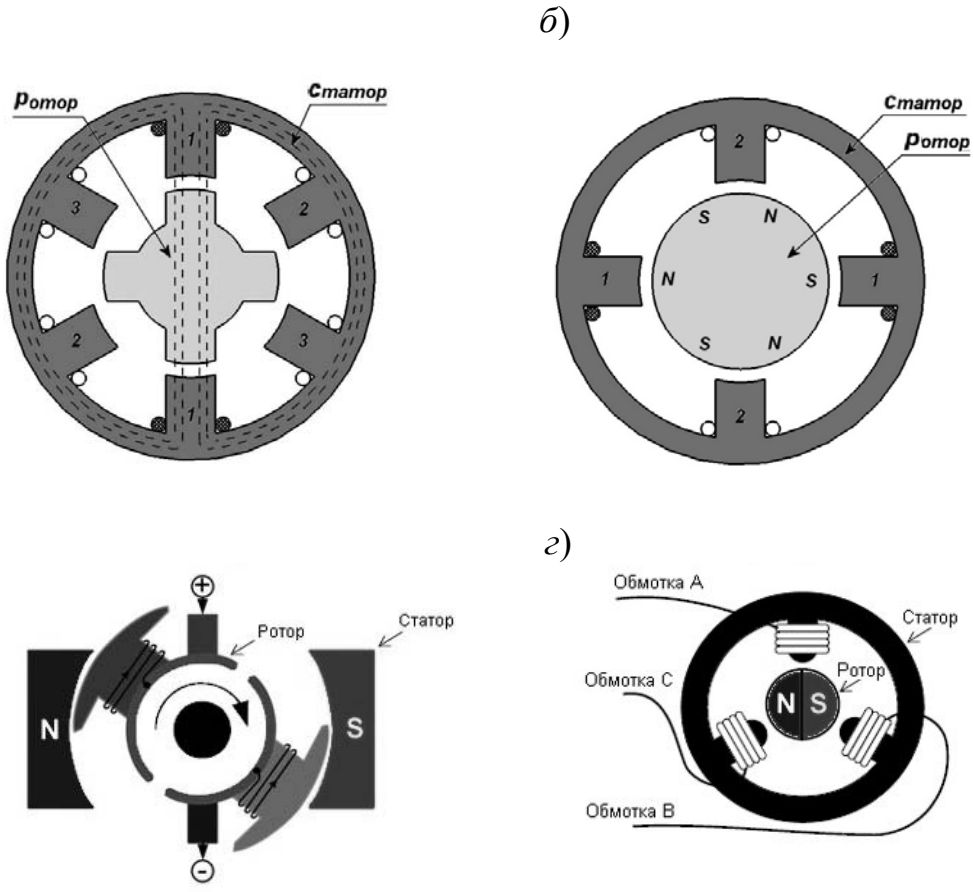


Рисунок 5.1 – Конструкции шаговых электродвигателей с переменными (*а*) и постоянными (*б*) магнитами и серводвигателей с коллекторным (*в*) и бесколлекторным (*г*) двигателями

5.1.2 Языки для программирования станков с ЧПУ.

С момента появления первых станков с ЧПУ до внедрения новейших обрабатывающих центров появились различные языки для программирования обработки. Сегодня программирование в G- и M-кодах является наиболее популярным.

Языки программирования были созданы ассоциацией EIA (Electronic Industries Alliance) и в дальнейшем доработаны до готового к использованию

формата (RS274X), которые были утверждены в качестве международного стандарта ISO 6983-1:1982 (ISO – International Organization for Standardization). Со временем стандарт был дополнен и расширен, превратился в NIST RS-274NGC (code in the dialect defined by the Next Generation Controller (NGC)). Для регламентации положений ввели ГОСТ 20999–83 *Устройства числового программного управления для металлообрабатывающего оборудования. Кодирование информации управляющих программ*, а обозначать их в технической литературе стали как язык ИСО-7 бит (ISO 7bit). В Республике Беларусь этот ТНПА действует до настоящего времени.

С тех пор широко используются как самостоятельно, так и в роли базового подмножества для создания сходных языков, постоянно совершенствуются и расширяются.

Официально этот язык считается стандартом для американских и европейских производителей оборудования с ЧПУ. Однако производители систем ЧПУ хоть и придерживаются этих стандартов для описания основных функций, но допускают вольности и отступления от правил, когда речь заходит о каких-либо специальных возможностях своих систем.

Системы ЧПУ Fanuc (Япония) были одними из первых, адаптированными под работу с G- и M-кодами ISO и использующими этот стандарт наиболее полно. В настоящее время стойки Fanuc являются очень популярными и наиболее распространенными как за рубежом, так в России и в Беларуси.

Стойки ЧПУ других известных производителей (например, Sinumerik (Siemens)) также имеют возможности по работе с G- и M-кодами, однако некоторые коды все же могут отличаться. Но знать все коды всех систем ЧПУ необходимости нет. Достаточно знать набор основных G- и M-кодов, а о возникшей разнице в программировании специфических функций можно узнать из документации к конкретной системе.

Структура управляющей программы в общем случае подчиняется следующим правилам в соответствии с международными стандартами и ГОСТ 20999–83 *Устройства числового программного управления для металлообрабатывающего оборудования. Кодирование информации управляющих программ*.

1 Каждая управляющая программа начинается символом «начало программы», подающим системе управления сигнал о начале выполнения программы. Вид символа «начало программы» зависит от особенностей применяемой системы ЧПУ. Наиболее часто в отечественных и зарубежных системах ЧПУ используется символ «%». При этом кадр с символом «начало программы» не нумеруется. Нумерация кадров начинается с последующего кадра.

2 Если управляющей программе необходимо присвоить обозначение, то его располагают в кадре с символом «начало программы» непосредственно за символом.

3 Если текст управляющей программы необходимо сопроводить комментарием, например сведениями об особенностях наладки станка, то его размещают перед символом «начало программы».

4 Управляющая программа должна заканчиваться символом «конец программы», подающим системе управления сигнал на прекращение

выполнения управляющей программы, останов шпинделя, приводов подач и выключение охлаждения. Наиболее часто в отечественных и зарубежных системах ЧПУ используется символ «%». Информация, помещенная в тексте управляющей программы после этого символа не должна восприниматься системой ЧПУ.

5 Информация, расположенная в тексте управляющей программы между символами «начало программы» и «конец программы» и заключенная в круглые скобки, не должна приниматься системой ЧПУ к исполнению. При этом в тексте внутри скобок не должны применяться символы «начало программы» и «главный кадр».

Команды даются последовательно и логично, поэтому программа состоит, как правило, из этапов (рисунок 5.2):

1) пуск;
2) загрузка инструмента;
3) включение шпинделя;
4) подача охлаждения;
5) перемещение инструмента в исходное положение;
6) запуск процесса обработки;
7) отключение охлаждения;
8) останов шпинделя;
9) возвращение шпинделя на исходную позицию;
10) завершение программы.

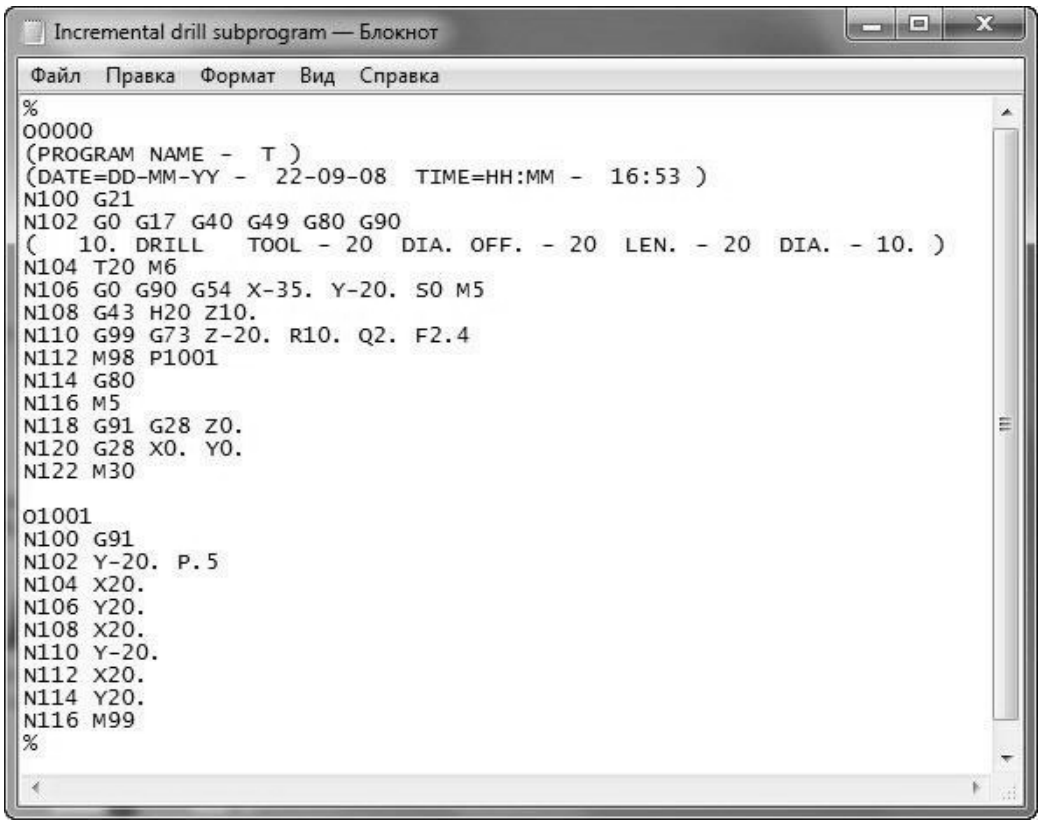


Рисунок 5.2 – Пример программы

5.1.3 G- и M-коды УЧПУ: *описание*.

G-команды являются основными и подготовительными, M-команды – вспомогательными (технологическими). Записываются вместе в строчку (первые – в начале, вторые – в конце) или, другими словами, покадрово – для наглядности листинга. В результате алгоритм представляет собой совокупность символьных блоков – с адресами и числовыми значениями.

В задачи G-группы входит определение линейной или круговой скорости, а также направления движения рабочих инструментов оборудования. Кроме того, они обязаны регламентировать расточку отверстий и нарезание резьбы, управлять координированием и другими особенностями дополнительной аппаратуры.

M-коды программирования ЧПУ призваны дополнять основные, упрощая выполнение алгоритма. На практике их роль сводится к смене лезвий, сверл (или других органов), к вызову и завершению подпрограмм.

Помимо этих двух распространенных семейств, также есть S-команды, определяющие специфику основного движения; F-команды, ответственные за характер подачи; D-, H-, T-команды, выражающие ключевые параметры навесных элементов.

Описание G-команд представлено в таблице 5.1.

Таблица 5.1 – Основные G-команды для станков с ЧПУ

Символ	Описание
1	2
G00	Перемещение инструмента на холостом ходу с ускорением
Задание интерполяции	
G01	Линейной
G02	Круговой по часовой стрелке
G03	В направлении, обратном предыдущему (против)
G04	Включение задержки (в миллисекундах)
G10	Задание новых начальных точек отсчета
G11	Отмена
G16	Работа в полярной системе координат
Режим измерений	
G20	В дюймах
G21	В метрах
G22	Активация стоп-рамок станка – пределов перемещения
G23	Отмена
G28	Возврат к референтной точке
G30	Перемещение по Z-оси вверх
Компенсация габаритов рабочего органа	
G40	Отмена (для размеров)
G41	Радиуса слева
G42	Радиуса справа
G43	Высоты положительно
G44	Высоты отрицательно

Окончание таблицы 5.1

1	2
G53	Переход на координаты оборудования
G54-59	Переключение на заданные оператором значения
G68	Поворот под нужным углом
G69	Отмена
Цикл сверления	
G80	Отказ
G81	Включение
G82	С задержкой
G83	С отходом
G84	Резьбование
Активация системы координат	
G90	Абсолютной
G91	Относительной
Формат подачи F	
G94	мм/мин
G95	мм/об
G98	Отмена
G99	Отказ от возвращения на точку «подхода» после выполнения цикла

Описание команд М-кода представлено в таблице 5.2.

Таблица 5.2 – Вспомогательные (технологические) команды М-кода для станков ЧПУ

Символ	Описание
M00	Остановка до нажатия на «старт»
M01	Аналогично предыдущей, но при условии действия режима подтверждения
M02	Завершение алгоритма
Начало вращения шпинделя	
M03	По часовой стрелке
M04	Против
M05	Остановка
M06	Смена рабочего органа
M07	Активация дополнительного охлаждения
Основное охлаждение	
M08	Включение
M09	Выключение
M30	Конец вывода данных
M98	Начало подпрограммы
M99	Ее завершение, возврат к главному алгоритму

Параметры команд, заданные латинскими буквами, представлены в таблице 5.3.

Таблица 5.3 – Параметры команд, заданные латинскими буквами

Символ	Описание
Координаты точек по соответствующим осям	
X n.n	По оси X
Y n.n	По оси Y
Z n.n	По оси Z
Скорость	
F	Рабочей подачи
S	Вращения шпинделя
R	Радиус (либо, реже, показатель стандартного цикла)
I, J, K	Габариты дуги, наблюдаемой в случае круговой интерполяции
D	Коррекция действующего инструмента
P	Задержка (или количество вызовов подцикла)
L	Подпрограмма по метке

Строкой безопасности называется кадр, содержащий G-коды, которые переводят станок ЧПУ в определенный стандартный режим, отменяют ненужные функции и обеспечивают безопасную работу с управляющей программой. В начале программы для обработки строкой безопасности является кадр N1.

Пример

N1 G21 G40 G49 G54 G80 G90

5.1.4 Отработка перемещений в ЧПУ.

Существует два принципиально различных режима перемещения инструмента относительно обрабатываемого материала (заготовки): режимы позиционирования и интерполяции (контурные).

Перемещения в режиме позиционирования, называемые холостыми, идут без обработки и отрабатываются с максимальной скоростью, не указываемой в управляющей программе. Примером могут служить перемещения сверла от одной позиции сверления к другой. Траектории холостых перемещений не регламентированы, система ЧПУ выдерживает лишь координаты точек запрограммированных остановок.

Перемещения в режиме интерполяции называют контурными. Современные системы ЧПУ поддерживают режимы линейной, круговой, а иногда и других видов интерполяции.

Контурные режимы:

1) перемещение вдоль одной оси;

2) перемещение рабочего стола прямолинейно, но не параллельно ни одной из осей станка (рисунок 5.3);

3) перемещение рабочего стола криволинейно, по дуге (рисунок 5.4).

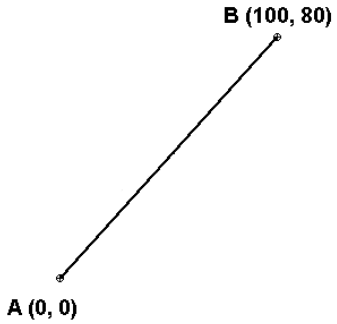


Рисунок 5.3 – Прямолинейное перемещение

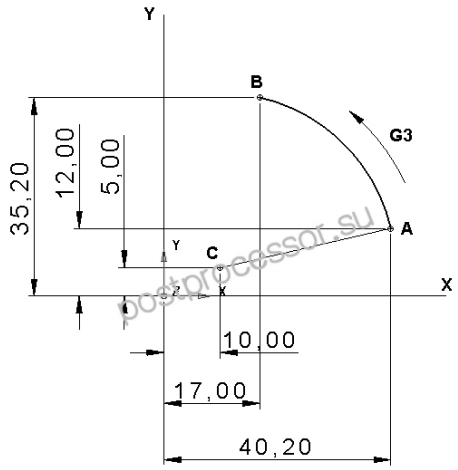


Рисунок 5.4 – Круговые перемещения

5.1.5 Программирование линейных перемещений.

Чтобы использовать линейный интерполятор (осуществлять программирование линейных перемещений), используется подготовительная функция G01 и указываются координаты конечной точки перемещения с заданной скоростью:

G01 X n.n Yn.n Z n.n Fn.n,

где X, Y, Z – адреса линейных осей;

F – скорость перемещения.

Например, для программирования прямолинейного перемещения из точки А в точку В (см. рисунок 5.3) со скоростью 1000 мм/мин необходимо в устройстве управления сформировать следующий кадр:

G01 X100 Y80 F1000.

5.1.6 Программирование круговых перемещений.

Для программирования движения по дуге (см. рисунок 5.4) используют подготовительные функции G2 (обход по часовой стрелке) и G3 (обход против часовой стрелки). Так как дуга − плоский элемент, то необходимо указывать

в какой плоскости производится движение XY, XZ или YZ. Данным плоскостям соответствуют подготовительные функции G17, G18, G19.

Дуга на плоскости может быть задана следующими параметрами:

− центром C (координаты Xc, Yc);

− начальной точкой A (X1, Y1);

− конечной точкой B (X2, Y2);

− радиусом R.

Программирование кругового движения в разных устройствах ЧПУ (УЧПУ) задается по-разному в зависимости от настройки интерполятора. В общем, достаточно знать начальные координаты дуги (они известны по предыдущему кадру движения фрезы), конечные координаты и координаты центра. Программирующий кадр имеет вид

G17 G02 Xn.n Yn.n In.n Jn.n Fn.n,

где G17 – плоскость XY;

G02 – режим круговой интерполяции с обходом по часовой стрелке;

Xn.n, Yn.n – координаты конечной точки дуги;

In.n, Jn.n – координаты центра дуги;

Fn.n – скорость перемещения инструмента.

Для дуги, изображенной на рисунке 5.4, кадр программы будет иметь следующий вид. Предполагаем, что инструмент уже находится в точке A (40.20; 12.00):

G17 G02 X17.0 Y35.20 I10.0 J5.0 F100.

5.1.7 Интерполятор системы ЧПУ.

Интерполятор системы ЧПУ – вычислительный блок, задающий последовательность управляющих воздействий для перемещения рабочего органа в соответствии с функциональной связью между координатами опорных точек.

Интерполятор производит интерполяцию – расчёт координат промежуточных точек траектории движения рабочего органа по координатам конечных точек и определенному закону интерполяции в плоскости или в пространстве.

При задании информации (вводе очередного кадра программы в устройство управления) в виде координат конечной для данного участка точки интерполятор автоматически определяет положения всех точек, по которым должен переместиться рабочий орган, выдает поочередно соответствующие импульсы на привода по координатам.

УЧПУ автоматически определяет радиус и выдает импульсы на круговое движение.

На рисунке 5.5 представлена линейная и круговая интерполяция.

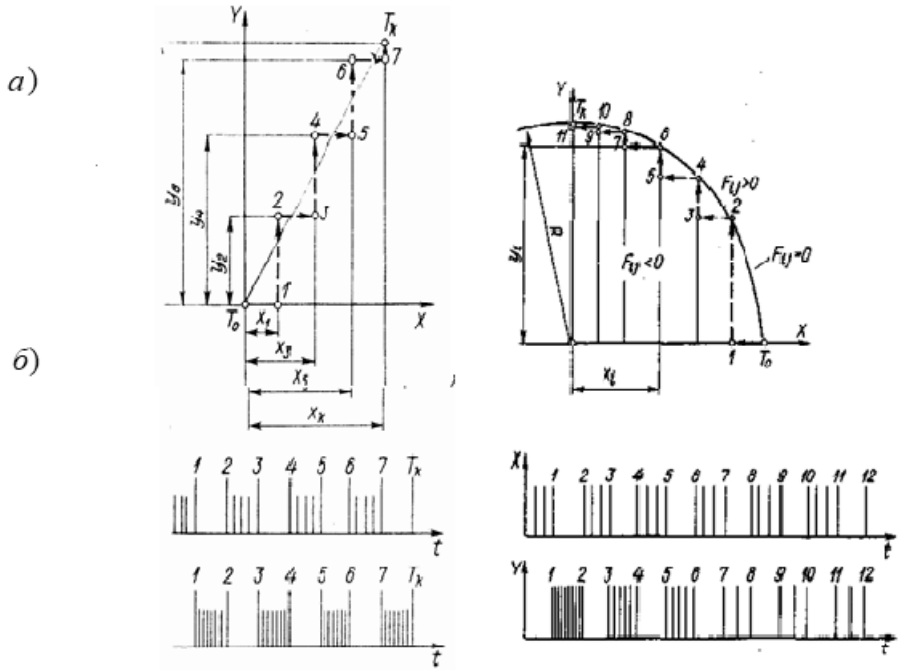


Рисунок 5.5 – Линейная и круговая интерполяция

5.1.8 Пример написания программы.

Для примера написания программы возьмем первую букву русского алфавита А (рисунок 5.6).

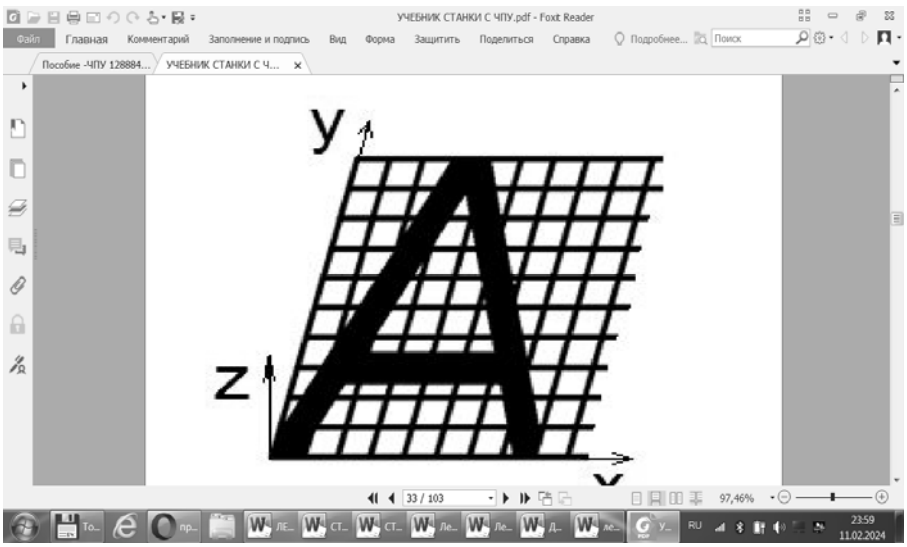


Рисунок 5.6 – Обрабатываемая буква А (черным цветом выделен результат обработки)

Зададим поле обработки, вписав букву А в прямоугольник 10×8 мм. Обработку буквы А будем производить на условном вертикально-фрезерном станке с ЧПУ фрезой диаметром 1 мм в заготовке из органического стекла. Составляемую управляющую программу запишем в таблицу 5.4.

Таблица 5.4 – Управляющая программа обработки буквы А

Кадр	Содержание	Комментарий
%	Начало программы	–
N1	G90 G40 G17	Система координат абсолютная, компенсация на радиус инструмента выключена, плоскость интерполяции XY
N2	S500 M3	Задать скорость вращения шпинделя и включить шпиндель
N3	G0 X0.5 Y0.5	Переход в точку начала обработки на холостом ходу
N4	Z1.0	Подход к заготовке по Z, не доходя 1 мм, на холостом ходу
N4	G1 Z-1.0 F100	Врезание в заготовку на подаче 100 мм/мин
N5	X3.75 Y 9.5	Первый штрих буквы А
N6	X4.25	Продолжение движения
N7	X8.5 Y0.5	Второй штрих буквы А
N8	G0 Z1	Подъем режущего инструмента на безопасную высоту Z = 1 мм, на холостом ходу
N9	X2.0 Y3.0	Перевод инструмента в точку обработки штриха X2.0 Y3.0, на холостом ходу
N10	G1 Z-1.0 F100	Врезание в заготовку на подаче 100 мм/мин
N11	X6.5	Обработка штриха буквы А
N12	G0 Z12	Отвод инструмента от заготовки на холостом ходу
N13	M5	Выключить шпиндель
N14	M30	Конец программы

Программа обработки фрезой контура «Δ-треугольник» представлена на рисунке 5.7.

```
M3 S1000
G21 (mm)
G0 Z5
G0 X0 Y0
G1 Z-1 F80.0
G1 X25 Y50 Z-1 F100
G1 X50 Y0 Z-1
G1 X0 Y0 Z-1
G0 Z5
M5
M2
```

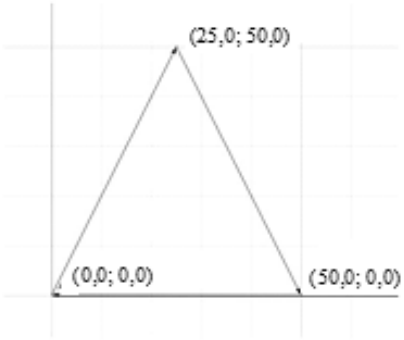


Рисунок 5.7 – Программа обработки фрезой контура «Δ-треугольник»

5.1.9 Пример создания базовой программы ЧПУ.

Создадим минимальную программу G-кода VMC-HMC (вертикально-горизонтальный обрабатывающий центр) для сверления двух отверстий:

```
G21 G90 G54;
G1 M6;
S1000 M3;
M08;
G00;
X50. Y0.
G43 H1 Z10;
G81 Z-20. F120;
X-50;
G80;
G00;
G91;
G28 Z0;
G28 X0. Y0;
M05;
M30.
```

Разъяснения кадров.

G21 G90 G54; -> Выбор (G21) миллиметра в абсолютном режиме (G90) с использованием первой (G54) системы координат.

T1 M6 ; -> (M6) Крепление (T1) Инструмент-1.

S1000 M3 ; -> (M3) Поверните инструмент по часовой стрелке на (S1000) 1000 об/мин.

M08 ; -> Открыть охлаждающую жидкость.

G00 ; -> Будьте готовы к быстрому позиционированию.

X50. Y0. -> Перейти к X50. Y0. координаты.

G43 H1 Z10. ; -> (G43) Возьмите (H1) смещение высоты инструмента 1 и переместитесь на Z10.

G81 Z-20. F120. ; -> В режиме прямого сверления (цикл сверления) просверлите отверстие глубиной 20 мм, двигаясь со скоростью 120 мм/мин (в X50. Y0.).

X-45 ; -> Просверлите отверстие в X-45. Y0.

G80; -> Закрыть цикл сверления, чтобы позиции не повторялись для цикла прямого сверления.

G00; -> Будьте готовы к быстрому позиционированию.

G91; -> Измените режим на инкрементную систему, чтобы она двигалась с последней точки.

G28 Z0. ; -> Перейти к верхнему пределу оси Z (исходное положение Z).

G28 X0. Y0. ; -> Перейти к пределам осей X и Y (исходное положение X и Y).

M05 -> Остановить шпиндель (инструмент вращается со скоростью 1000 об/мин).

M30; -> Конец программы (подайте звуковой и световой сигнал, подтверждающий завершение программы).

Содержание отчета

1 Титульный лист установленного образца.
2 Цель работы.
3 Задание.
4 Основные правила создания управляющей программы.
5 Линейная, круговая интерполяция, примеры программ.
6 Листинг программы с пояснениями (изображение буквы А).
7 Листинг программы с пояснениями (изображение «Δ-треугольник»), разъяснения шагов программы.
8 Ответы на контрольные вопросы.
9 Вывод по лабораторной работе.

Контрольные вопросы

1 Как осуществляется программирование линейных перемещений? Пример.
2 Как осуществляется программирование круговой интерполяции? Пример.
3 Язык числового программного управления, нормативный документ.
4 Назначение G-кодов (основные типы команд), М-кодов (основные типы команд).
5 Структура управляющей программы.
6 Режимы перемещения инструмента.
7 Итерполятор систем с ЧПУ, назначение, принцип работы (рисунки линейной и круговой интерполяции – отработка перемещений).
8 Виды приводов систем ЧПУ, конструкция, достоинства и недостатки.

Список литературы

1 **Шишов, О. В.** Программируемые контроллеры в системах промышленной автоматизации: учебник / О. В. Шишов. – М. : ИНФРА-М, 2023. – 365 с.
2 **Волков, М. А.** Управление техническими и технологическими системами : учеб. пособие / М. А. Волков, А. Ю. Постыляков, Д. В. Исаков. – М. ; Вологда: Инфра-Инженерия, 2022. – 252 с.
3 Информационные системы и цифровые технологии : учеб. пособие: в 2 ч. / В. В. Трофимов, М. И. Барабанова, В. И. Кияев, Е. В. Трофимова ; под общ. ред. проф. В. В. Трофимова и В. И. Кияева. – М. : ИНФРА-М, 2021. – Ч. 1. – 253 с.
4 Руководство пользователя по программированию ПЛК в CoDeSys 2.3. – Смоленск: Пролог, 2008. – 452 с.
5 Общие сведения о CoDeSys. – URL: http://www.3s-software.ru / publications (дата обращения : 05.04.2025).