Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет
им. Г.И. Носова»
Многопрофильный колледж

УТВЕРЖДАЮ
Директор
_____ / С.А. Махновский
«27» февраля 2019 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ И ЛАБОРАТОРНЫХ РАБОТ
по учебной дисциплине
ОПЦ.07 «ОСНОВЫ МИКРОПРОЦЕССОРНЫХ СИСТЕМ УПРАВЛЕНИЯ В ЭНЕРГЕТИКЕ»
для студентов специальности 08.02.09 Монтаж, наладка и эксплуатация электрооборудования промышленных и гражданских зданий

Магнитогорск, 2019

ОДОБРЕНО
Предметно-цикловой комиссией
Монтаж и эксплуатация электрооборудования
Председатель С.Б. Меняшева
Протокол №6 от 20.02.2019 г.

Методической комиссией МпК
Протокол №5 от 21.02.2019 г

Составитель:
преподаватель ФГБОУ ВО «МГТУ им. Г.И. Носова» МпК Ремез.Т.Б

Методические указания по выполнению практических и лабораторных работ разработаны на основе рабочей программы учебной дисциплины «Основы микропроцессорных систем в энергетике».

Содержание практических и лабораторных работ ориентировано на формирование универсальных учебных действий, подготовку обучающихся к освоению программы подготовки специалистов среднего звена.

СОДЕРЖАНИЕ

1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Состав и содержание практических и лабораторных занятий направлены на реализацию Федерального государственного образовательного стандарта среднего профессионального образования.

Ведущей дидактической целью практических занятий является формирование профессиональных практических умений (умений выполнять определенные действия, операции, необходимые в последующем в профессиональной деятельности) или учебных практических умений, необходимых в последующей учебной деятельности.

Ведущей дидактической целью лабораторных занятий является экспериментальное подтверждение и проверка существенных теоретических положений (законов, зависимостей).

В соответствии с рабочей программой учебной дисциплины «Основы микропроцессорных систем в энергетике» предусмотрено проведение практических и лабораторных занятий. В рамках практического/лабораторного занятия обучающиеся могут выполнять одну или несколько практических/лабораторных работ.

В результате их выполнения, обучающийся должен:

уметь:

- составлять функциональные и структурные схемы управления различными электроэнергетическими объектами;
- выбирать средства технической реализации микропроцессорных систем управления;
- программировать микропроцессорные системы управления на основе ПЛК широкого применения.

Содержание практических и лабораторных занятий ориентировано на подготовку обучающихся к освоению профессионального модуля программы подготовки специалистов среднего звена по специальности и овладению ***профессиональными компетенциями:***

ПК 2.3. Организовывать и производить наладку и испытания устройств электрооборудования промышленных и гражданских зданий.

ПК 2.4. Участвовать в проектировании силового и осветительного электрооборудования.

А также формированию ***общих компетенций***:

ОК01: Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам

ОК02: Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности

ОК03: Планировать и реализовывать собственное профессиональное и личностное развитие.

ОК04: Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.

ОК05: Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста.

ОК07: Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях

ОК09: Использовать информационные технологии в профессиональной деятельности

ОК10: Пользоваться профессиональной документацией на государственном и иностранных языках

Выполнение обучающихся практических и лабораторных работ по учебной дисциплине «Основы микропроцессорных систем в энергетике» направлено на:

- обобщение, систематизацию, углубление, закрепление, развитие и детализацию полученных теоретических знаний по конкретным темам учебной дисциплины;
- формирование умений применять полученные знания на практике, реализацию единства интеллектуальной и практической деятельности;
- формирование и развитие умений: наблюдать, сравнивать, сопоставлять,

анализировать, делать выводы и обобщения, самостоятельно вести исследования, пользоваться различными приемами измерений, оформлять результаты в виде таблиц, схем, графиков;

- приобретение навыков работы с различными приборами, аппаратурой, установками и другими техническими средствами для проведения опытов;

- развитие интеллектуальных умений у будущих специалистов: аналитических, проектировочных, конструктивных и др.;

- выработку при решении поставленных задач профессионально значимых качеств, таких как самостоятельность, ответственность, точность.

Практические и лабораторные занятия проводятся после соответствующей темы, которая обеспечивает наличие знаний, необходимых для ее выполнения.

2 ПЕРЕЧЕНЬ ПРАКТИЧЕСКИХ И ЛАБОРАТОРНЫХ ЗАНЯТИЙ

Разделы/темы	Темы практических и лабораторных занятий	Количество часов	Требования ФГОС СПО (уметь)
Раздел I.Микропроцессорные системы (МПС)		**30**	
Тема 1.1 Общие сведения об МПС	Практическая работа 1. Представление информации в различных системах счисления.	2	У2 У01.1, У01.2, У01.5, У01.6, У01.9, У01.11, У02.1, У02.2, У02.4, У02.5, У026, У02.7, У03.2, У04.2, У05.2, У09.1, У09.2, У10.7
	Практическая работа 2. Логические основы цифровых устройств	4	У2 У01.1, У01.2, У01.5, У01.6, У01.9, У01.11, У02.1, У02.2, У02.4, У02.5, У026, У02.7, У03.2, У04.2, У05.2, У09.1, У09.2, У10.7
	Практическая работа 3. Изучение принципов построения и работы цифровых последовательных устройств	4	У2 У01.1, У01.2, У01.5, У01.6, У01.9, У01.11, У02.1, У02.2, У02.4, У02.5, У026, У02.7, У03.2, У04.2, У05.2, У09.1, У09.2, У10.7
	Практическая работа 4. Изучение принципов построения и работы цифровых комбинационных устройств	4	У2 У01.1, У01.2, У01.5, У01.6, У01.9, У01.11, У02.1, У02.2, У02.4, У02.5, У026, У02.7, У03.2, У04.2, У05.2, У09.1, У09.2, У10.7
	Практическая работа 5. Изучение принципов организации запоминающих устройств	2	У2 У01.1, У01.2, У01.5, У01.6, У01.9, У01.11, У02.1, У02.2, У02.4, У02.5, У026, У02.7, У03.2, У04.2, У05.2, У09.1, У09.2, У10.7
	Практическая работа 6. Изучение схемы типовой МПС	4	У1 У01.1, У01.2, У01.5, У01.6, У01.9, У01.11, У02.1, У02.2, У02.4, У02.5, У026, У02.7, У03.2, У04.2, У05.2, У09.1, У09.2, У10.7
Тема 1.3 МПС на основе программируемых	Лабораторная работа 1. Программирование ПЛК на языке LD	2	У1, У3 У01.1, У01.2, У01.5, У01.6, У01.9, У01.11, У02.1, У02.2, У02.4, У02.5, У026, У02.7, У03.2, У04.2, У05.2, У07.2, У07.3, У09.1, У09.2, У10.7

логических контроллеров	Лабораторная работа 2. Программирование ПЛК на языке ST.	2	У1, У3 У01.1, У01.2, У01.5, У01.6, У01.9, У01.11, У02.1, У02.2, У02.4, У02.5, У026, У02.7, У03.2, У04.2, У05.2, У07.2, У07.3, У09.1, У09.2, У10.7
	Лабораторная работа 3. Программирование ПЛК на языке IL.	2	У1, У3 У01.1, У01.2, У01.5, У01.6, У01.9, У01.11, У02.1, У02.2, У02.4, У02.5, У026, У02.7, У03.2, У04.2, У05.2, У07.2, У07.3, У09.1, У09.2, У10.7
	Лабораторная работа 4. Программирование ПЛК на языке FBD.	2	У1, У3 У01.1, У01.2, У01.5, У01.6, У01.9, У01.11, У02.1, У02.2, У02.4, У02.5, У026, У02.7, У03.2, У04.2, У05.2, У07.2, У07.3, У09.1, У09.2, У10.7
	Лабораторная работа 5. Программирование ПЛК на языке SFC.	2	У1, У3 У01.1, У01.2, У01.5, У01.6, У01.9, У01.11, У02.1, У02.2, У02.4, У02.5, У026, У02.7, У03.2, У04.2, У05.2, У07.2, У07.3, У09.1, У09.2, У10.7
ИТОГО		**30**	

3 МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Тема 1.1 Общие сведения об МПС
Практическая работа 1.
Представление информации в различных системах счисления.

Цель: закрепить знания об основных системах счисления, используемых в цифровой технике.

Выполнив работу, Вы будете:

уметь:

- выбирать средства технической реализации микропроцессорных систем управления;

Материальное обеспечение:

не требуется

Задание 1. Данные для выполнения задания приведены в таблице 1 (по вариантам).

1) определите количество разрядов числа;
2) запишите алфавит данной системы счисления;
3) запишите число в виде полинома. В общем виде

$$x = a_n \times p^n + a_{n-1} \times p^{n-1} + a_{n-2} \times p^{n-2} + \ldots a_1 \times p^1 + a_0 \times p^0$$

Таблица 1 – Исходные данные

Вариант	Число	Вариант	Число	Вариант	Число
1	2564852_{10}	11	2612264_8	21	54896213_{10}
2	362580_8	12	$B14B4_{16}$	22	321323125_8
3	1101011_2	13	4589723_{10}	23	$345A655_{16}$
4	$3B9C781E_{16}$	14	21404233_8	24	1100101_2
5	25978631_{10}	15	$46089B_{16}$	25	789562_{10}
6	1110011_2	16	1001101_2	26	232150_8
7	1456987_{10}	17	3658921_{10}	27	13468_{16}
8	5435533_8	18	15752251_8	28	110110_2
9	$163B5B_{16}$	19	$37D4A9_{16}$	29	4523698_{10}
10	726196_{10}	20	110011_2	30	$4506B2_{16}$

Задание 2. Переведите число из двоичной системы счисления в десятичную систему счисления. Полученное десятичное число переведите в восьмеричную и шестнадцатеричную систему счисления. Данные для выполнения задания приведены в таблице 2 (по вариантам).

Таблица 2 - Исходные данные

Вариант	Число	Вариант	Число	Вариант	Число
1	1101100	11	1011010000	21	111001010
2	111100000	12	1100100001	22	110000010
3	110001011	13	1110010000	23	11100110
4	1000001001	14	110000001	24	110100100
5	1001011000	15	110010100	25	1011110010
6	1110011	16	1000001101	26	100100010
7	100100111	17	11011010	27	1000101011
8	101000001	18	101101110	28	11000011
9	1010001110	19	10001010	29	110000110
10	111000000	20	11001000	30	101000011

Форма представления результата:

Отчет по работе должен содержать:

1) наименование работы и цель работы;
2) результаты работы;
3) выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному ответу.

Оценка «удовлетворительно» ставится, если в работе не получен ответ и приведено неполное выполнение задания, но ход выполнения задания верный

Оценка «неудовлетворительно» ставится, если задание не выполнено или если приведен правильный ответ, но решение отсутствует.

Практическая работа 2.
Логические основы цифровых устройств

Цель: закрепить знания об основных логических функциях, научиться строить логические схемы по логическим выражениям, выполнять анализ и синтез логических схем.

Выполнив работу, Вы будете:

уметь:

- выбирать средства технической реализации микропроцессорных систем управления;

Материальное обеспечение:

не требуется

Краткие теоретические сведения

Анализ логической схемы – это составление полного её описания, которое содержит таблицу истинности, логические функции, временные диаграммы.

Пример 1. Выполнить анализ логической схемы (рис. 1): составить логическую функцию, таблицу истинности, временные диаграммы.

Решение:

Заданная схема имеет три входных сигнала X0, X1, X2 и три выходных сигнала У0, У1, У2. Запишем для каждого выходного сигнала логическую функцию:

$$Y0 = X0 \cdot X1 \cdot X2$$

$$Y1 = X0 \vee X1 \vee X2$$

$$Y2 = X0 \oplus X1 \oplus X2$$

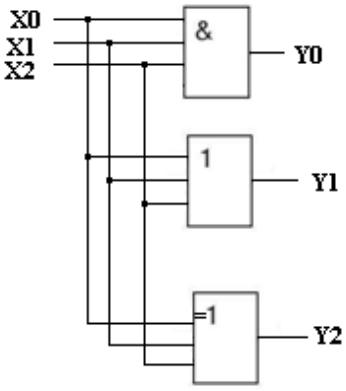


Рисунок 1 - Логическая схема

Таблица истинности (табл. 1) содержит значения выходных сигналов (аргументов), соответствующие различным комбинациям входных сигналов. Схема имеет три входа (X0, X1, X2), для которых существует 2^3=8 вариантов набора. Таблица составлена в соответствии с логикой работы элементов.

Таблица 1 - Таблица истинности

№	Аргументы			Функции		
	X0	X1	X2	У0	У1	У2
0	0	0	0	0	1	0
1	0	0	1	0	1	1
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	0	1	1
5	1	0	1	0	1	0
6	1	1	0	0	1	0
7	1	1	1	1	0	1

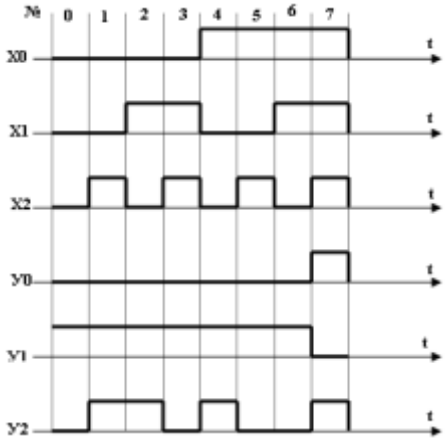


Рисунок 2 - Временные диаграммы, поясняющие работу схемы

Пример 2. Выполнить анализ логической схемы (рис.3): составить логическую функцию, таблицу истинности, временные диаграммы.

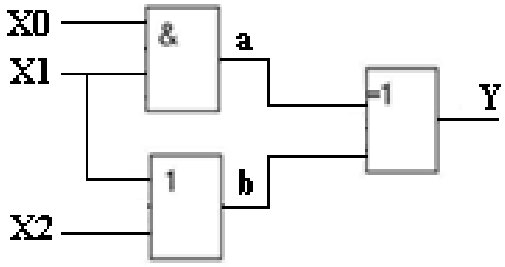


Рисунок 3 – Логическая схема

Решение:

Дополнительно обозначим внутренние узлы схемы ***a*** и ***b*** и добавим для этих узлов столбцы в таблице истинности (табл. 2). Сигнал ***a*** формируется логическим элементом И, на входы которого поданы сигналы X0 и X1. Сигнал ***b*** формируется элементом ИЛИ, на входы которого поступают сигналы X1 и X2. Сигналы ***a*** и ***b*** позволяют определить выходной сигнал ***Y*** по правилам логического элемента «Искл. ИЛИ». Временные диаграммы приведены на рисунке 4.

Таблица 2 - Таблица истинности

№	Аргументы			Функции		
	X0	X1	X2	a	b	Y

0	0	0	0	0	0	0
1	0	0	1	0	1	1
2	0	1	0	0	1	1
3	0	1	1	0	1	1
4	1	0	0	0	0	0
5	1	0	1	0	1	1
6	1	1	0	1	1	0
7	1	1	1	1	1	0

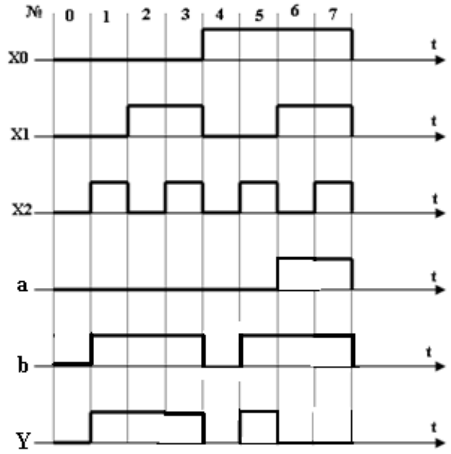


Рисунок 4 - Временные диаграммы

Составление логической функции по заданной схеме. Сначала записываем уравнения для вспомогательных сигналов ***a*** и ***b,*** а затем для выходного сигнала ***Y.***

$a = X0 \cdot X1$

$b = X1 \vee X2$

$Y = a \oplus b = (X0 \cdot X1) \oplus (X1 \vee X2)$

Синтез логических схем – это проектирование и разработка электрической принципиальной схемы устройства по определенным правилам на основе логических элементов.

Пример 3. Выполнить синтез комбинационной схемы, заданной логической функцией; составить таблицу истинности, временную диаграмму. Логическая функция

$Y = (a \cdot \overline{b}) \vee (b \cdot \overline{a})$

Решение:

Для построения логической схемы, реализующей логическую функцию, необходимо логические элементы, предназначенные для выполнения логических операций располагать, начиная от входа в порядке, указанном в выражении (рис.5).

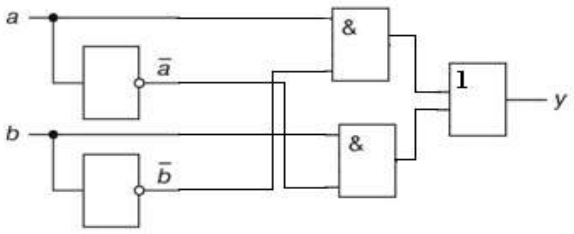


Рисунок 5 – Логическая схема

Логическое уравнение содержит два аргумента ***a*** и ***b*** и их инверсии $\overline{a}$, $\overline{b}$. На входы логических элементов И подаются a, $\overline{b}$ и $\overline{a}$, b соответственно. Результаты логического умножения подаются на вход логического сложения (функция ИЛИ).

Таблица истинности содержит четыре комбинации входных аргументов (табл. 3). Временные диаграммы приведены на рисунке 6.

Таблица 3 - Таблица истинности

a	b	$\overline{a}$	$\overline{b}$	$a \cdot \overline{b}$	$b \cdot \overline{a}$	Y
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	1	0	0	0	0	0

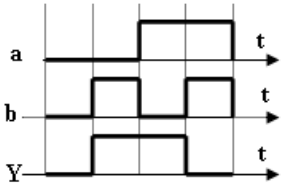


Рисунок 6 - Временные диаграммы

Пример 4. Построить структуру логического устройства, реализующего логическуфункцию $y = (a + b + c)(a + b + c)(\overline{a} + b + c)(\overline{a} + \overline{b} + c)$

Структура логического устройства, реализующего логическую функцию, приведена на рисунке 7

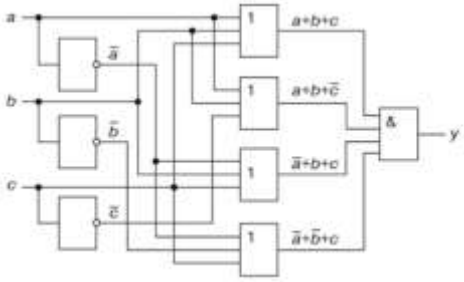
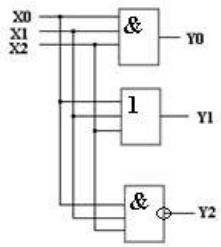
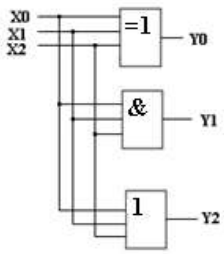
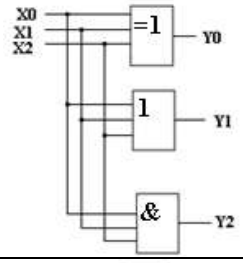
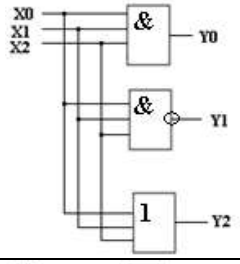
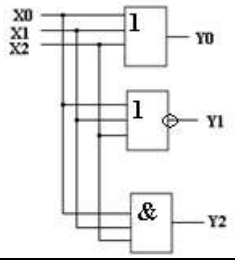
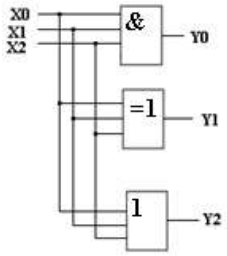


Рисунок 7 – Схема логического устройства

Задание 1. Выполнить анализ комбинационной схемы: составить логическую функцию, таблицу истинности, временные диаграммы. Данные для выполнения задания приведены в таблице 4 (по вариантам).

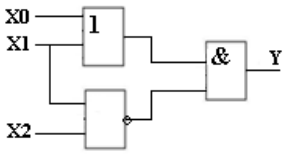
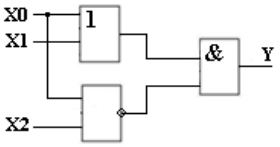
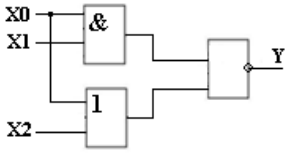
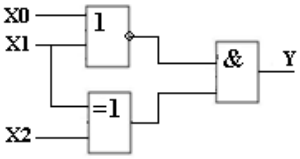
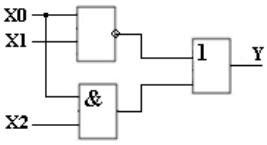
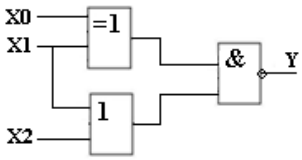
Таблица 4 – Исходные данные для выполнения задания 1

Вариант		Вариант	
1, 7, 13, 19		4,10, 16, 22	

2,8, 14,20		5,11,17, 23	
3,9, 15, 21		6,12, 18, 24	

Задание 2. Выполнить анализ комбинационной схемы: составить логическую функцию, таблицу истинности, временные диаграммы. Данные для выполнения задания приведены в таблице 5 (по вариантам).

Таблица 5 – Исходные данные для выполнения задания 2

Вариант		Вариант	
1,7, 13, 19		2,10,16,22	
3,8,14, 20		4,11, 17, 23	
5,9,15, 21		6, 12, 18, 24	

Задание 3. Выполнить синтез комбинационной схемы, заданной логической функцией. Данные для выполнения задания приведены в таблице 6 (по вариантам).

Таблица 6 – Исходные данные для выполнения задания 3

Вариант	Логическая функция
1, 6, 11, 16, 21, 26	$y = (\overline{a}b + \overline{c})(\overline{a} + \overline{b} + c)(a + b + c)$
2, 7, 12, 17, 22, 27	$y = (a + b + \overline{c})(\overline{a} + \overline{b}c)(a + \overline{b} + \overline{c})$
3, 8, 13, 18, 23, 28	$y = (b + a\overline{c})(\overline{a} + bc)(a + \overline{b} + c)$
4, 9, 14, 19, 24, 29	$y = (\overline{a}\,\overline{b} + \overline{c})(a + \overline{b} + c)(ab + \overline{c})$
5, 10, 15, 20, 25, 30	$y = (a + \overline{b}c)(\overline{a} + b + \overline{c})(ab + c)$

Форма представления результата:

Отчет по работе должен содержать:

1) наименование работы и цель работы;
2) результаты работы;

3) выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Практическая работа 3.
Изучение принципов построения и работы цифровых последовательных устройств

Цель: закрепить знания об основных видах цифровых последовательных устройств: триггерах, счетчиках, регистрах, их характеристиках и построении.

Выполнив работу, Вы будете:

уметь:

- выбирать средства технической реализации микропроцессорных систем управления;

Материальное обеспечение:

не требуется

Теоретические сведения

Триггеры - это большой класс электронных устройств, обладающих способностью длительно находиться в одном из двух устойчивых состояний и чередовать их под воздействием внешних сигналов. Триггеры — это логические устройства с памятью. Их выходные сигналы в общем случае зависят не только от сигналов, приложенных к входам в данный момент времени, но и от сигналов, воздействовавших на них ранее.

Триггер Т (рис. 1) в общем случае можно представить как устройство, состоящее из ячейки памяти (ЯП) и логического устройства (ЛУ) управления, преобразующего входную информацию в комбинацию сигналов, под воздействием которых ЯП принимает одно из двух устойчивых состояний.

Информационные сигналы поступают на входы А и В ЛУ и преобразуются в сигналы, поступающие на внутренние входы S' и R' ЯП. Процесс преобразования информационных сигналов осуществляется при воздействии сигналов, подаваемых на вход Т предустановки и вход С синхронизации. Вход Т обычно используется для разрешения приема информации, а исполнительный вход С обеспечивает тактируемый прием информации. В простейшем триггере ЛУ может отсутствовать, а информационные сигналы подаются непосредственно на входы S и R ЯП.

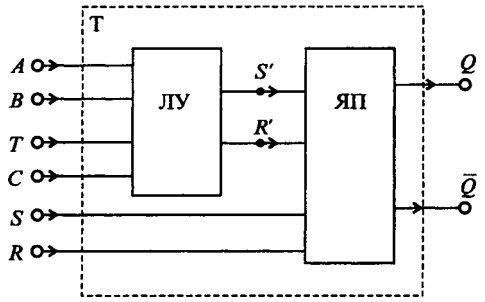


Рис.5

Триггер называют *синхронным* при наличии входа С, а при его отсутствии — *асинхронным*. Изменение состояния асинхронного триггера происходит сразу же после соответствующего изменения потенциалов на его информационных входах А и В. В синхронном триггере изменение состояния может произойти только в момент присутствия соответствующего сигнала на входе С. Синхронизация может осуществляться импульсом (потенциалом) или фронтом (перепадом потенциала). В первом случае сигналы на информационных входах оказывают влияние на состояние триггера только при разрешающем потенциале на входе С. Во втором случае воздействие информационных сигналов проявляется только в момент изменения потенциала на входе С, т. е. при переходе его от 1 к 0 или от 0 к 1. Универсальные триггеры могут работать как в синхронном, так и в асинхронном режимах.

В зависимости от свойств, числа и назначения входов триггеры можно разделить на несколько видов:

1. RS-триггер имеет два информационных входа *S* и *R*. Подача на вход *S* сигнала 1, а на вход *R* сигнала 0 устанавливает на выходе Q триггера сигнал 1. Наоборот, при сигналах S=0 и R=1 сигнал на выходе триггера Q=0. Функционирование RS-триггера определяется уравнениями: $\mathbf{Q_n=S+\underline{R}Q_{n-1}}$.

Для RS-триггера комбинация S=1 и R=1 является запрещенной. После такой комбинации информационных сигналов состояние триггера будет неопределенным: на его выходе Q может быть 0 или 1. RS-триггеры могут быть асинхронными или синхронными (в этом случае у них имеется вход С).

RS-триггер можно построить на логических элементах И-НЕ или ИЛИ-НЕ. На рис. 6 приведены УГО и схема асинхронного RS-триггера. На рис.2 приведены УГО и схема синхронного RS-триггера (RST-триггера).

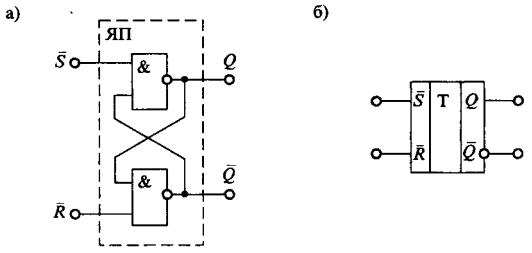


Рис.2

2. RST-триггер (синхронный RS-триггер) – отличается от RS-триггера тем, что имеет тактовый вход С (Clock — времязадающий) и его состояние может изменяться только при поступлении на этот вход тактового импульса (ТИ). В промежутках между ТИ изменения сигналов на информационных входах не вызывают переключения триггера, а лишь предопределяют то состояние, в которое он переключается при поступлении очередного ТИ. На рис. 3 показаны схема RST-триггера и его УГО. При отсутствии ТИ (С = 0) состояния на выходах логических элементов 1 и 2 (т. е. на входах $\underline{S}$, $\underline{R}$) измениться не могут, поэтому состояния на выходах Q и $\underline{Q}$ остаются постоянными. При появлении ТИ (С = 1) элементы 1 и 2 по входам S и $\underline{R}$ функционируют как инверторы, т. е. точно так же, как в схеме RS-триггера. Характеристическое уравнение RST-триггера: $\mathbf{Q_n=C*(S+\underline{R}Q)_{n-1}}$. Для RST-триггера. Как и для RS, недопустимо сочетание S = R = 1.

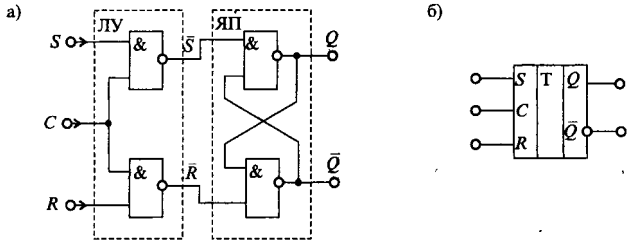


Рис.3

3. JK-триггер имеет также два информационных входа J и К. Подобно RS-триггеру, в JK-триггере J и К – это входы установки выхода Q триггера в состояние 1 или 0. Однако, в отличие от RS-триггера, в JK-триггере наличие J=K=l приводит к переходу выхода Q триггера в противоположное состояние. JK-триггеры синхронизируются только перепадом потенциала на входе С. Условие функционирования JK-триггера имеет вид: $\mathbf{Q_n=(J\underline{Q}+K\underline{Q})_{n-1}}$.

JK- триггер можно построить базе логических элементов и синхронного RS-триггера. На рис.4. приведены УГО и схема JK-триггера.

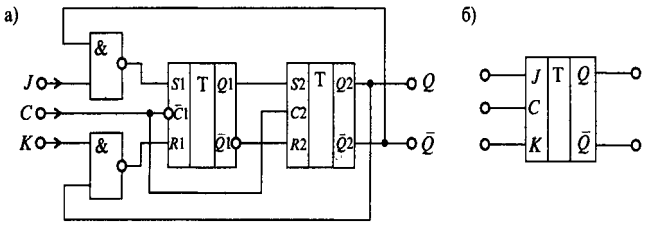


Рис.4

4. D-триггер, или триггер задержки, при поступлении синхросигнала на вход С устанавливается в состояние, соответствующее потенциалу на входе D. Уравнение функционирования D-триггера имеет вид: $\mathbf{Q_n=D_{n-1}}$. Это уравнение показывает, что выходной сигнал Q_n изменяется не сразу после изменения входного сигнала D, а только с приходом синхросигнала, т. е. с задержкой на один период импульсов синхронизации (Delay — задержка).

Синхронизация D-триггера может осуществляться импульсом или фронтом. D-триггер можно построить на основе логических элементах И-НЕ. На рис.5 приведены УГО и схема D-триггера.

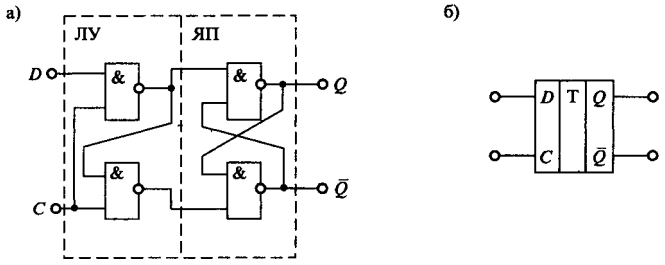


Рис.5

5. T-триггер, или счетный триггер, изменяет состояние выхода по фронту импульса на входе С. Кроме синхровхода С Т-триггер может иметь подготовительный вход Т. Сигнал на этом входе разрешает (при Т=1) или запрещает (при Т=0) срабатывание триггера от фронтов импульсов на входе С. Функционирование Т-триггера определяется уравнением: $\mathbf{Q_n=(\underline{Q}T+Q\underline{T})_{n-1}}$. Из этого уравнения следует, что при Т=1 соответствующий фронт сигнала на входе С переводит триггер в противоположное состояние. Частота изменения потенциала на выходе Т-триггера в два раза меньше частоты импульсов на входе С. Это свойство Т-триггера позволяет строить на их основе двоичные счетчики. Поэтому эти триггеры и называют счетными. На рис. 6 приведены схема (на базе RST-триггера) и УГО Т-триггера.

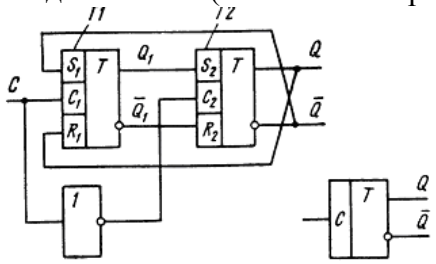


Рис.6

Основные параметры ИМС триггеров можно разделить на две группы:

1. статические:

- входное напряжение высокого $U_{1вх}$ и низкого $U_{0вх}$ уровней;
- ток потребления от источника питания $I_{потр}$;
- напряжение $U_{пит}$;
- нагрузочная способность ($К_{раз}$);
- мощность, потребляемая от источника питания $P_{потр}$.

2. динамические:

- время t_{01} переключения из низкого уровня в высокий, время t_{10} переключения из высокого уровня в низкий;

- максимальная частота f_{max} переключения.

Основные параметры ИМС триггеров различных серий приведены в табл. 5.

Счетчик – это цифровое устройство, предназначенное для подсчета числа импульсов. В процессе работы счетчик последовательно изменяет свое состояние в определенном порядке. Длина списка разрешенных состояний счетчика называется *модулем пересчета, основанием пересчета* или *емкостью счетчика* ($К_С$).

Одно из возможных состояний счетчика принимается за начальное. Если счетчик начал счет от начального состояния, то каждый импульс, кратный модулю счета $К_С$ снова устанавливает счетчик в начальное состояние, а на выходе счетчика появляется сигнал переноса P (или займа Z).

Последовательность внутренних состояний счетчика можно кодировать различными способами. Чаще всего используют:

• двоичное (двоичные счетчики) или двоично-десятичное (декадные счетчики) кодирование;

• одинарное кодирование, когда состояние счетчика представлено местом расположения одной-единственной единицы или одного-единственного нуля (кольцевые счетчики).

Если коды расположены в возрастающем порядке, то счетчик называют *суммирующим* (Up-counter). Счетчики, у которых коды расположены в убывающем порядке, называют *вычитающими* (Down-counter), а счетчики, у которых направление перебора кода может изменяться, называют *реверсивными* (Up/Down counter).

Если для работы счетчика требуется наличие синхросигнала, то такой счетчик называют *синхронным*. Счетчики, которые работают без синхросигналов, называют *асинхронными*.

Обобщенная схема счетчика приведена на рис. 7. Счетчик СТ можно представить в общем случае как устройство, которое содержит входную логику, управляющую работой счетчика, и выходную логику, которая используется для указания окончания счета или формирования сигнала переноса P. Для приведения счетчика в начальное состояние используется сигнал сброса, поступающий на вход R.

Параллельный код для предварительной установки счетчика поступает на входы $S_0 \ldots S_n$. Сигнал разрешения параллельной загрузки M останавливает счет и позволяет подготовленным на входах $S_0 \ldots S_n$ данным загрузиться в счетчик в момент прихода очередного тактового импульса C. Счетчик считает тактовые импульсы, поступающие на вход C, если присутствует сигнал разрешения счета на входе V.

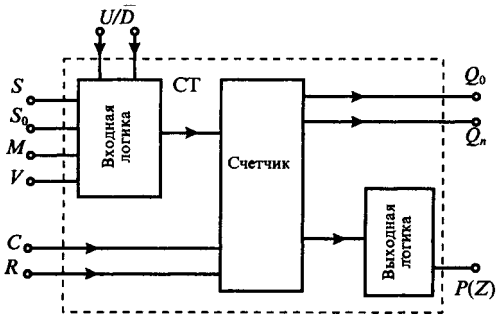


Рис. 7

Выходными сигналами счетчика обычно являются сигналы, снимаемые с выходов отдельных разрядов $Q_0 \ldots Q_n$, сигнал окончания счета или сигнал переноса P.

Двоичные счетчики строят чаще всего на основе Т-триггеров, поскольку эти триггеры могут и хранить свое состояние, и суммировать с ним по модулю 2 входной сигнал. Двоичный n-разрядный счетчик содержит n Т-триггеров, и его емкость $К=2^n$. Связи между триггерами быть различных типов.

От вида связи существенно зависят время переключения счетчика в новое состояние, его аппаратные затраты и ряд других свойств. Чаще всего используют три типа связей: непосредственную, тракт последовательного переноса, тракт параллельного переноса.

1. Счетчики с непосредственным переносом.

В данном счетчике переключение триггеров, вызванное срезом входного сигнала, происходит триггер за триггером, *последовательно,* и задержка распространения n-разрядного счетчика в n раз больше задержки распространения одного Т-триггера. Если разрядов много, то большая задержка может оказаться серьезным недостатком такого

счетчика. Для наращивания разрядности счетчика вход еще одного Т-триггера или такого же счетчика подключается непосредственно к выходу старшего разряда.

Достоинствами счетчика с непосредственной связью являются предельная простота схемы и легкость ее наращивания для увеличения разрядности.

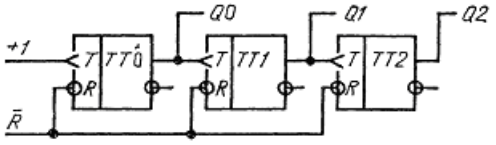


Рис.8.

2.Счетчики с последовательным переносом.

Принцип последовательного переноса заключается в следующем: входной импульс проходит сквозь все каскады счетчика, в которых содержатся единицы, попутно сбрасывая все их в нуль, и переводит в 1 первый встреченный на пути погашенный триггер, причем сквозь этот каскад импульс уже не проходит. На выходе последнего элемента И вырабатывается сигнал переноса CR, который используется при наращивании разрядности счетчика. Этот счетчик по значению задержки распространения не имеет существенных преимуществ перед счетчиком с непосредственной связью. Однако он оказывается значительно удобнее, если в процессе работы счетчика потребуется с помощью логических элементов выполнить какие-либо переключения в связях между триггерами, например, переключить счетчик на вычитание или изменить содержимое отдельных триггеров.

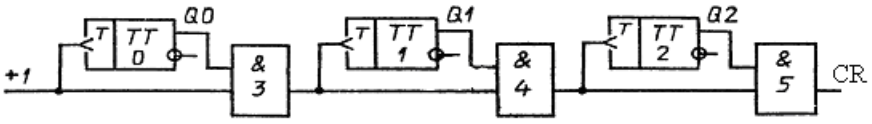
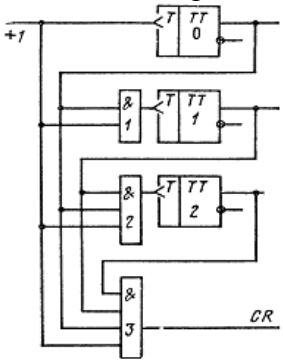


Рис.9

3. Счетчики с параллельным переносом.

Для уменьшения времени задержки используют счетчик с параллельным переносом. Принцип параллельного переноса заключается в следующем. На входе каждого триггера (кроме самого первого) включен конъюнктор. Входной сигнал +1 поступает параллельно на все конъюнкторы и там, где они открыты, вызывает одновременное переключение соответствующих триггеров. На вход каждого конъюнктора кроме входного сигнала поданы выходы всех триггеров младше данного разряда. Поэтому при подаче сигнала +1 изменят свое состояние все те триггеры, перед которыми все более младшие триггеры находились в состоянии 1.



В счетчике с параллельным переносом все триггеры начинают переключаться одновременно, в результате время задержки у счетчиков с параллельным переносом заметно меньше, чем у счетчиков с последовательным, и притом не зависит от числа разрядов.

Рис.10

Различные области применения требуют счетчиков с модулями пересчета, не только кратными целой степени двойки, но и другими, например для работы в десятичной системе — 10, для схем часов и календарей— 60, 24, 7… В общем случае требуется строить счетчики по любому заданному основанию К. На базе готовых счетчиков счетчик по произвольному основанию можно построить тремя основными способами.

а) Двоичный счетчик разрядности n, такой, чтобы 2^n было больше К, дополняется элементом И, который по состояниям выходов Q_i обнаруживает код конца счета (обычно К—1), после чего по цепи R сбрасывает счетчик в 0

Достоинства способа: естественная двоичная последовательность кодов от 0 до К—1; использование обычно имеющегося в счетчиках входа R. В случае суммирующего счетчика достаточно собрать на элементе И лишь прямые выходы тех триггеров, которые при коде конца счета равны 1. Число входов элемента И, таким образом, зависит от кода конца счета.

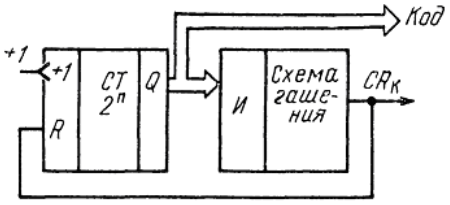


Рис.11

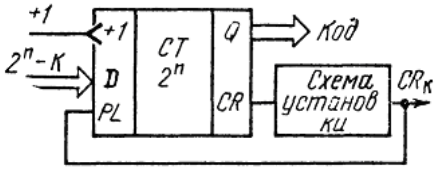


Рис.12

б) Двоичный счетчик перед началом счета по тракту параллельной загрузки загружается кодом дополнения числа К до 2^n. Кодом конца счета в этом случае является естественное переполнение счетчика, т. е. код “Все единицы”, обнаруживаемый штатным трактом переноса, в результате чего вырабатывается сигнал CR. Сигнал CR, воздействуя на вход PL, управляющий параллельной загрузкой, снова устанавливает в счетчике дополнение К до 2^n . Достоинство способа — использование штатного тракта переноса и имеющихся во многих счетчиках входов параллельной загрузки, а также легкая смена основания пересчета. Для этого входы Di нужно подключить не к константам 1 и 0 (питание и общий провод), а к выходу специального регистра начальных состояний. Недостатком способа является неестественная последовательность получаемых кодов, требующая в случае их использования перекодировки. Поэтому данный способ применяется, когда показания счетчика не важны, а используется лишь сигнал его выходного переноса. Это типично для задачи деления частоты входных сигналов на некоторое число К. Счетчики, выполняющие эту функцию, называют делителями.

в) В качестве кода начала и конца счета выбирают некоторую произвольную пару кодов, разность между которыми равна К—1. Этому способу присущи сложности как первого, так и второго, и используется он лишь в специальных случаях.

Основные параметры ИМС счетчиков можно разделить на две группы:

1. статические:

- входное напряжение высокого $U_{1вх}$ и низкого $U_{0вх}$ уровней;
- ток потребления от источника питания $I_{потр}$;
- напряжение $U_{пит}$;
- нагрузочная способность ($К_{раз}$);
- мощность, потребляемая от источника питания $Р_{потр}$
- модуль счета $К_с$.

2. динамические:

- время t_{01} переключения из низкого уровня в высокий, время t_{10} переключения из высокого уровня в низкий;
- максимальная частота f_{max} переключения.

Регистр – это устройство для запоминания многоразрядных слов. Для построения регистров необходимое число триггеров объединяют вместе и рассматривают как единый функциональный узел. Типовыми внешними связями регистра являются информационные входы D_i, вход сигнала записи (или загрузки) С, вход гашения R, выходы триггеров Q_i. В упрощенном варианте регистр может не иметь входа гашения инверсных выходов. Выпускаемые промышленностью регистры иногда объединяют на кристалле микросхемы с другими узлами, в паре с которыми регистры часто используются в схемах цифровой

аппаратуры. Существуют микросхемы, в которых регистры могут принимать входные данные с двух и более источников, выбираемых сигналами на адресных входах микросхемы, и передавать содержимое регистра на различные приемники.

Сразу несколько регистров содержат микросхемы регистровой памяти. Микросхемы регистровой памяти легко наращиваются по разрядности и допускают наращивание по числу регистров. Они разработаны для построения блоков регистров общего назначения (РОН) и других специализированных блоков памяти небольшого объема, предназначенных для временного хранения исходных данных и промежуточных результатов в цифровом устройстве.

По количеству линий передачи переменных (по типу используемых в составе регистра триггеров) регистры делят на:

- однофазные – построенные на одновходовых триггерах (D-,T-триггерах);
- парафазные - построенные на двухвходовых триггерах (RS-, JK-триггерах).

По системе синхронизации регистры разделяют на:

- однотактные;
- двухтактные;
- многотактные.

По функциональному назначению регистры бывают:

- накопительные;
- сдвигающие.

По способам приема и выдачи информации регистры делят на:

- параллельные – принимают информацию в параллельном коде т.е. одновременно несколько разрядов числа;
- последовательные – принимают информацию в последовательном коде т.е. поразрядно;
- параллельно-последовательные – способны принимать информацию как поразрядно, так и сразу несколько разрядов.

Сдвигающий, или сдвиговый, регистр *(shift register)* - это регистр, содержимое которого при подаче управляющего сигнала СДВИГ может сдвигаться в сторону старших или младших разрядов. Данные регистры помечаются специальным значком: ←, →, ↔. Направление стрелки показывает направление сдвига, а двунаправленная стрелка обозначает двунаправленный регистр.

УГО однонаправленного сдвигающего регистра показано на рис. 13, схема этого регистра на базе D-триггера — на рис. 14. Регистр состоит из цепочки триггеров. Пусть на рисунке триггер ТТ0—младший, ТТ3—старший; D-вход каждого триггера (кроме ТТ0) подключен к выходу соседнего младшего триггера. Когда на все объединенные С-входы триггеров поступает активный отрицательный фронт сигнала СДВИГ, выход каждого триггера принимает состояние своего младшего соседа и, таким образом, информация, содержащаяся в регистре, сдвигается на один разряд в сторону старших разрядов, влево. Триггер ТТ0 принимает при этом состояние последовательного входа DS (data serial). Регистр загружается данными, последовательно поступающими по этому входу. Считывать данные, хранимые в регистре, можно как в последовательном коде, с выхода последнего разряда, так и в параллельном, сразу со всех разрядов.

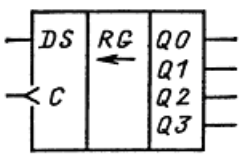


Рис.13.

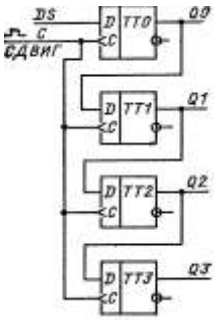


Рис.14.

Двунаправленный (bidirectional) сдвигающий регистр (рис.15) способен сдвигать содержимое и влево, и вправо. При двунаправленном сдвиге появляются два последовательных входа: вход, принимающий данные, вдвигаемые в регистр со стороны младшего разряда справа,—DR *(data right)* и вход со стороны старшего разряда слева—DL *(data left).* Эти входы используются и при наращивании регистра: DR подключается к выходу старшего разряда соседней младшей секции общего регистра, DL подключается к выходу Q_0 младшего разряда соседней старшей секции общего регистра. Кроме последовательных входов сдвигающие регистры часто имеют параллельные входы (D0—D3), с помощью которых регистр можно загрузить параллельным кодом сразу за один такт.

Кроме перечисленных входов у регистра имеются несколько управляющих входов: SL *(shift left)* – сдвиг влево, SR *(shift right)* – сдвиг вправо и PL*(parallel load)* – параллельная загрузка.

Часто для более экономного использования выводов микросхемы управляющие входы SL, SR, PL не выводят из корпуса непосредственно, а управляют ими через небольшой дешифратор режимов, т.е имеются входы S1, SO. В разных микросхемах кодировка режимов различна, например, код S1S0=00 находится в режиме хранения, при S1S0=01 выполняется сдвиг вправо, при S1S0=10—влево, и при S1S0=1—прием параллельного кода. В однонаправленных сдвиговых регистрах код режима может быть одноразрядным: 0—сдвиг, 1—параллельная загрузка

Применения сдвигающих регистров очень разнообразны. В арифметике сдвиг числа на один разряд влево соответствует умножению его на 2, сдвиг вправо—делению пополам. Сдвигающий регистр, содержащий всего одну единицу, может выполнять роль счетчика, отображающего число поступивших на вход сигналов положением единицы на линейной шкале (например, горящая лампочка номера этажа в лифте). Кроме того, сдвигающие регистры преобразуют параллельный код в последовательный и обратно, как это показано на рис. 16. Выходной регистр RG1 некоторого блока передает данные в линию. Входной регистр RG2 другого блока принимает их. При соответствующем значении сигнала управления режимом данные параллельным кодом загружаются через D-входы в регистр RG1. Затем и RG1, и RG2 переводятся в режим сдвига и на их C-входы подается серия из четырех (в данном случае) C-импульсов. При этом содержимое регистра-передатчика разряд за разрядом появляется на выходе Q3, последовательным кодом передается по линии и через вход DS вдвигается в регистр-приемник. После этого переданные данные могут быть считаны параллельным кодом с выходов Q0—Q3 регистра 2. Следует обратить внимание на то, что в приемник нужно передавать не только последовательный код данных, но и синхроимпульсы, необходимые для управления сдвигом на приемной стороне.

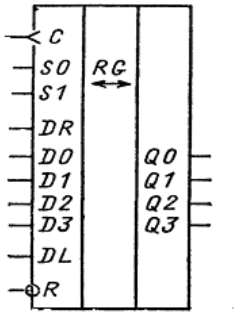


Рис.15

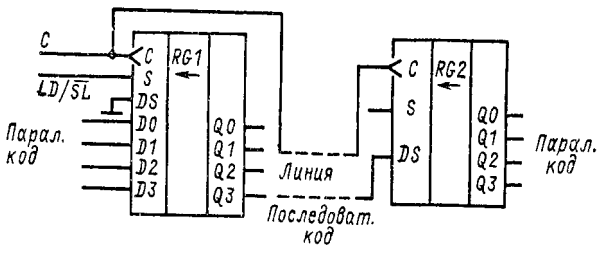


Рис.16

Практические задания

1. Заполните таблицу.

Обозначение вывода триггера	Назначение вывода триггера.
A, B	
T	

C	
S', R'	
Q	
$\underline{Q}$	

2. Составьте временную диаграмму Т-триггера, используя уравнение функционирования.
3. Подпишите наименования триггеров, которым соответствуют приведенные уравнения:
   1. $Q_n=C*(S+\underline{R}Q)_{n-1}$
   2. $Q_n=D_{n-1}$
   3. $Q_n=(\underline{Q}T+Q\underline{T})_{n-1}$
   4. $Q_n=S+\underline{R}Q_{n-1}$
   5. $Q_n=(J\underline{Q}+K\underline{Q})_{n-1}$

4. Заполните таблицу, используя справочники и интернет-ресурсы

Наименование	Тип логики	Назначение	$U_{пит}$, В	$P_{пот}$, мВт	$t_{зд}$, нс	f_{max}, МГц
555ТР2						
155ТВ1						
555ТМ2						
561ТР2						
561ТВ1						
561ТМ3						

5. Двоичный счетчик, емкостью $К_С$ = 8, начал считать с комбинации (кода) 000. Какая комбинация будет на выходе счетчика через: а) 5 импульсов, б) 8 импульсов, в) 11 импульсов?

6. Имеется двоичный двухразрядный счетчик. Укажите последовательность появления кодов на выходе счетчика, если он: а) суммирующий, б) вычитающий.

7. Заполните таблицу.

Обозначение вывода счетчика	Назначение вывода счетчика.
$S_0 \ldots S_n$	
$Q_0 \ldots Q_n$	
M	
V	
P	
C	
R	
Up	
Dn	

8. Определите емкость счетчика, если он состоит из 5 триггеров.

9. Постройте 4-х разрядный счетчик с непосредственным переносом и определите его временную задержку, если известно, что $t_{зд}$ одного триггера составляет 25 нс.

10. Какой схемой нужно дополнить двоичный счетчик разрядностью n=4, если нужно сделать счетчик с К=10?

11. Как будут изменяться состояния двоичного счетчика с n=3, если на его базе построен счетчик с К=5 после: а) 5 импульсов, б) 8 импульсов, в) 10 импульсов?

12. Заполните таблицу, используя справочники и интернет-ресурсы

Наименование	Тип логики	Назначение	Модуль счета $К_с$	f_{max}, МГц
155ИЕ5				
555ИЕ18				

155ИЕ7				
561ИЕ8				
561ИЕ11				

13. Составьте обобщенную структурную схему регистра.

14. Постройте схему и УГО пятиразрядного сдвигового регистра и поясните назначение его выводов.

15. Заполните таблицу, используя справочники и интернет-ресурсы

Наименование	Тип логики	Назначение	f_{max}, МГц
155ИР1			
155ИР13			
531ИР11			
561ИР9			

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. конспект теоретических сведений
3. практические задания;
4. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Практическая работа 4.

Изучение принципов построения и работы цифровых комбинационных устройств

Цель: закрепить знания об основных видах цифровых комбинационных устройств: мультиплексорах и демультиплексорах, преобразователях кодов, арифметических устройствах их характеристиках и построении.

Выполнив работу, Вы будете:

уметь:

- выбирать средства технической реализации микропроцессорных систем управления;

Материальное обеспечение:

не требуется

Теоретические сведения

Мультиплексор (MUX) – это функциональный узел, который обеспечивает передачу цифровой информации, поступающей по нескольким входным линиям связи, на одну выходную линию. Выбор входной линии, информация с которой поступает на выход, осуществляется при помощи сигналов поступающих на адресные входы. Обобщенная схема мультиплексора приведена на рис. 1. Мультиплексор (Multiplexer) в общем случае можно представить в виде коммутатора управляемого входной логической схемой. Входные логические сигналы X_i поступают на входы коммутатора и через коммутатор передаются на выход Y. Управление коммутатором осуществляется входной логической схемой. На вход логической схемы подаются адресные сигналы $A_к$ (Adress). Мультиплексоры могут иметь дополнительный управляющий вход E (Enable), который может выполнять стробирование выхода Y. Кроме этого некоторые мультиплексоры могут иметь выход с тремя состояниями: два состояния 0 и 1 и третье состояние — отключенный выход (выходное сопротивление равно бесконечности). Перевод мультиплексора в третье состояние производится сигналом OE (Output Enable).

Для обозначения коммутационных возможностей мультиплексора можно пользоваться условно записью (n → 1), где n — число входов. Так, например, мультиплексор с функцией (1 → 1) является одиночным ключом, а мультиплексор (4 → 1) имеет четыре входа и один выход.

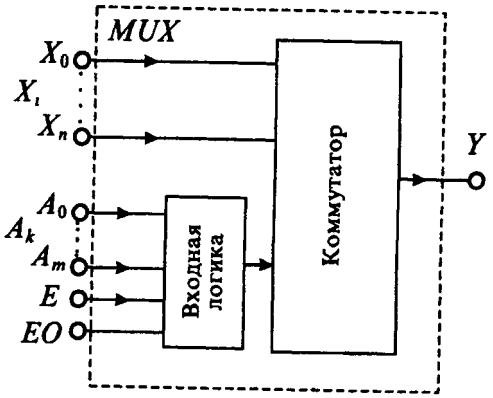
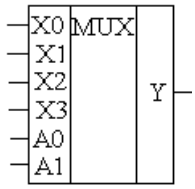


Рис.1

В зависимости от соотношения числа информационных входов n и числа адресных входов m мультиплексоры делятся на полные и неполные. Если выполняется условие $n = 2^m$, то мультиплексор будет полным. Если это условие не выполняется, т. е. $n<2^m$, то мультиплексор будет неполным. Наибольшее распространение получили мультиплексоры (2→1) с n=2 и m=1, (4→1) с n=4 и m=2, (8→1) с n=8 и m=3 и (16→1) с n=16 и m=4. Для неполных мультиплексоров число входных линий может быть любым, но, разумеется, не больше 2^m.

На рис. 2 показано УГО мультиплексора 4→1 и его таблица истинности.



A_1	A_0	Y
0	0	X_0
0	1	X_1
1	0	X_2
1	1	X_3

Рис.2

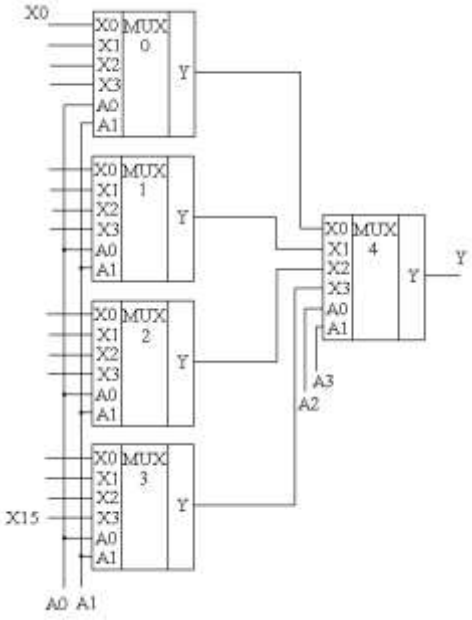


Рис.3

Для расширения числа входных линий можно использовать пирамидальное каскадирование мультиплексоров. На рисунке 3 приведен двухкаскадный мультиплексор типа (16→1) с управлением по четырем адресным линиям $A_0…A_3$. Первая группа мультиплексоров MUX0…MUX3 управляется младшими разрядами адресных сигналов A_0 и A_1. Выходной мультиплексор MUX4 управляется старшими разрядами адресных сигналов A_2 и A_3. Такое каскадирование мультиплексоров почти вдвое увеличивает задержку выходных сигналов.

Демультиплексор (DMX) – это функциональный узел, который обеспечивает передачу цифровой информации, поступающей по одной линии, на несколько выходных линий. Выбор выходной линии осуществляется при помощи сигналов, поступающих на адресные входы. Таким образом, демультиплексор выполняет преобразование, обратное действию мультиплексора.

Обобщенная схема демультиплексора, приведенная на рис. 4, сходна со схемой мультиплексора. Входной сигнал X поступает на вход коммутатора и через него передается на выходы $Y_0 \ldots Y_n$. Адресные сигналы $A_0 \ldots A_к$ имеют то же назначение, что и у мультиплексора. Сигнал стробирования E разрешает передачу входного сигнала через коммутатор.

Для обозначения коммутационных возможностей демультиплексоров можно пользоваться записью, аналогичной мультиплексорам (1→n), где n — число выходов демультиплексора. Так, например, демультиплексор (1→2) имеет два выхода, а демультиплексор (1→ 4) — четыре выхода.

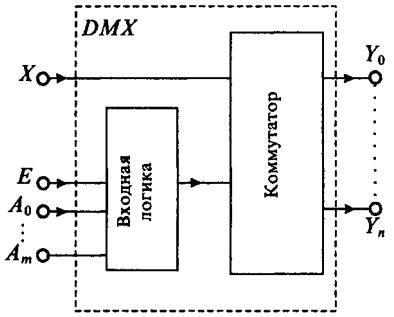
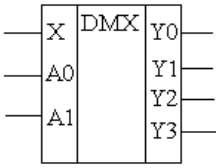


Рис.4

Демультиплексоры, как и мультиплексоры, могут быть полными и неполными. Деление мультиплексоров на эти две категории производится так же, как и у мультиплексоров, с той лишь разницей, что под n понимается число выходов, а не входов, как в мультиплексоре.

В зависимости от соотношения числа информационных выходов n и числа адресных входов m демультиплексоры делятся на полные и неполные. Если выполняется условие $n = 2^m$, то демультиплексор будет полным. Если это условие не выполняется, т. е. $n<2^m$, то демультиплексор будет неполным. Для неполных мультиплексоров число входных линий может быть любым, но не больше 2^m.

На рис. 5 показано УГО демультиплексора 1→4 и его таблица истинности.



A_1	A_0	Y_0	Y_1	Y_2	Y_3
0	0	X	0	0	0
0	1	0	X	0	0
1	0	0	0	X	0
1	1	0	0	0	X

Рис.5

Интегральные микросхемы мультиплексоров и демультиплексоров можно разделить на группы по следующим признакам:

- по числу выходов;
- по числу мультиплексоров и демультиплексоров в одном корпусе;
- по наличию стробирующего импульса E;
- по способности передавать сигналы в двух направлениях.

Операция изменения кода числа называется его **перекодированием**. ИМС, выполняющие эти операции, называются ***преобразователями кодов*.**

Преобразователи кодов бывают простые (выполняют стандартные операции изменения кода чисел, например, преобразований двоичного кода в одинарный или обратную операцию) и сложные (выполняют нестандартные преобразования кодов и их схемы разрабатывают каждый раз с помощью алгебры логики).

Примем, что преобразователи кодов имеют n входов и k выходов. Соотношения между n и k могут быть любыми: n=k, n<k и n>k. При преобразовании кода чисел с ними могут выполняться различные дополнительные операции, например, умножение на весовые коэффициенты. Примером невесового преобразования является преобразование двоично-десятичного кода в двоичный. Весовые преобразователи кодов используются при преобразовании числовой информации.

Интегральные микросхемы преобразователей кодов выпускаются только для наиболее распространенных операций:

- преобразователи двоично-десятичного кода в двоичный код;
- преобразователи двоичного кода в двоично-десятичный код;
- преобразователи двоичного кода в код управления сегментными индикаторами;
- преобразователи двоичного или двоично-десятичного кода в код управления шкальными или матричными индикаторами.

Примерами простейших преобразователей кодов, которые широко применяются в цифровых устройствах, являются шифраторы и дешифраторы.

Шифратор – это кодовый преобразователь, который имеет n входов и k выходов, и при подаче сигнала на один из входов (обязательно только на один) на выходах появляется двоичный код возбужденного входа. Число выходов и входов в полном шифраторе связано соотношением $n=2^k$.

Рассмотрим принцип построения шифратора на примере преобразования 8-разрядного единичного кода в двоичный код. Схема такого шифратора и его условное схематичное обозначение приведена на рис. 6. Если все входные сигналы имеют нулевое значение, то на выходе шифратора будем иметь нулевой код $Y_0=Y_1=Y_2==0$.

Младший выход, т. е. выход с весовым коэффициентом, равным 1, должен возбуждаться при входном сигнале на любом из нечетных входов, так как все нечетные номера в двоичном представлении содержат единицу в младшем разряде. Следовательно, младший выход — это выход схемы ИЛИ, к входам которой подключены все входы с нечетными номерами.

Следующий выход имеет вес два. Он должен возбуждаться при подаче сигналов на входы с номерами 2, 3, 6, 7, т. е. с номерами, имеющими в двоичном представлении единицу во втором разряде. Таким образом, входы элемента ИЛИ должны быть подключены к входным сигналам, имеющим указанные номера.

Старший разряд двоичного кода формируется из входных сигналов с номерами 4, 5, 6 и 7, т. е. из четырех старших разрядов единичного кода. Все рассмотренные состояния шифратора можно увидеть в таблице, приведенной на рис.7.

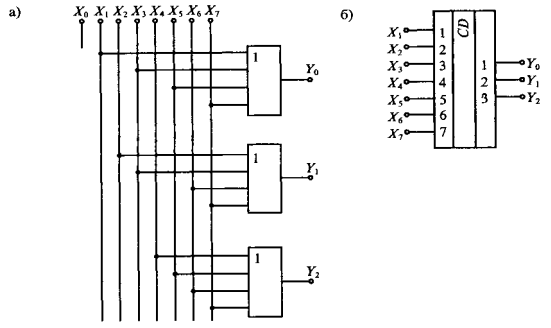


Рис.6

Состояния выходов шифратора 8×3

X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Рис.7

Как следует из выполненного построения, при помощи шифратора можно сократить (сжать) информацию для передачи ее по меньшему числу линий связи, так как k<n. Обратное

преобразование, т. е. восстановление информации в первоначальном виде можно выполнить с помощью дешифратора. Максимальное число входов шифратора не может превышать количество возможных комбинаций выходных сигналов, т. е. необходимо выполнение условия $n \le 2^k$.

В цифровых системах с помощью шифраторов обеспечивается связь между различными устройствами посредством ограниченного числа линий связи.

Дешифратор – это преобразователь двоичного n-разрядного кода в унитарный 2^n-разрядный код, все разряды которого, за исключением одного, равны нулю. Дешифраторы бывают полные и неполные. Для полного дешифратора выполняется условие: $k = 2^n$, где n — число входов, а k — число выходов.

В неполных дешифраторах имеется n входов, но реализуется $k<2^n$ выходов. Так, например, дешифратор, имеющий 4 входа и 10 выходов, будет неполным, а дешифратор, имеющий 2 входа и 4 выхода, будет полным.

Рассмотрим принцип построения дешифратора на примере преобразования трехразрядного двоичного кода в унитарный код. Если считать, что входы и выходы упорядочены по возрастающим номерам, т. е. считать, что коду 000 соответствует выход Y_0, коду 001 — выход Y_1 и т. д., то для полного дешифратора можно записать восемь упорядоченных уравнений:

$$\begin{cases} Y_0 = \overline{X_4}\,\overline{X_2}\,\overline{X_1} \\ Y_1 = \overline{X_4}\,\overline{X_2}\,X_1 \\ Y_2 = \overline{X_4}\,X_2\,\overline{X_1} \\ Y_3 = \overline{X_4}\,X_2\,X_1 \\ Y_4 = X_4\,\overline{X_2}\,\overline{X_1} \\ Y_5 = X_4\,\overline{X_2}\,X_1 \\ Y_6 = X_4\,X_2\,\overline{X_1} \\ Y_0 = X_4\,X_2\,X_1. \end{cases}$$

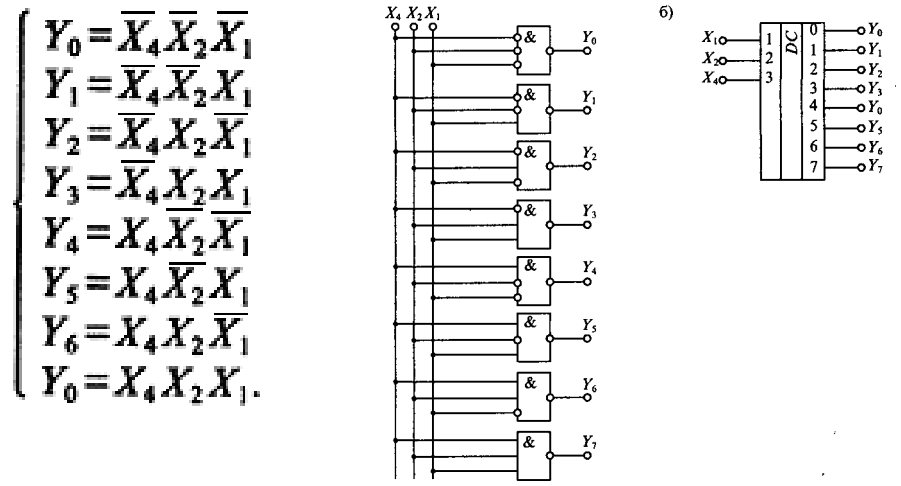


Рис.8

Реализовать восемь уравнений можно с помощью восьми трехвходовых элементов И. Полученная схема дешифратора и его условное схематичное изображение приведены на рис. 8.

Сумматор – это функциональный узел, выполняющий арифметическое сложение чисел.

Сумматор имеет n входов разрядов слагаемого A, n входов разрядов слагаемого B и вход переноса cr [p] (от *carry*—перенос). Выходами сумматора являются n выходов разрядов суммы S и выход переноса CR [P].

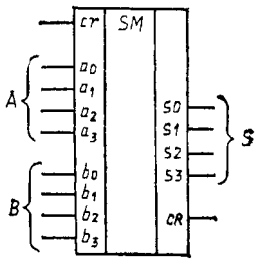
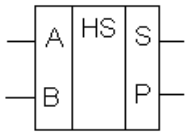


Рис. 9

Простейшим суммирующим элементом является ***полусумматор*** (ПС). ПС предназначен для сложения двух одноразрядных двоичных чисел *(HS (half sum — полусумма)).* Он имеет два входа A и B для двух слагаемых и два выхода: S (сумма) и P (перенос). На рис. 10 приведены УГО и таблица истинности ПС.

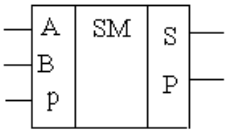


Входы		выходы	
A	B	S	P
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Рис.10

Полусумматор имеет два входа и пригоден поэтому для использования только в младшем разряде. Устройство для суммирования двух многоразрядных чисел должно иметь, начиная со второго разряда, три входа: два для слагаемых A_i и B_i и один для сигнала переноса p_{i-1} с предыдущего разряда.

Одноразрядный полный сумматор (ОПС) предназначен для сложения двух одноразрядных двоичных чисел. ОПС отличается от ПС тем, что на его базе можно построить многоразрядные сумматоры т.к. ОПС имеет вход переноса. На рис.11 приведены УГО и таблица истинности ОПС.



Входы			выходы	
A	B	p	S	P
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Рис.11

Многоразрядные сумматоры можно построить на базе ОПС несколькими способами:

• Сумматор с последовательным переносом *(ripple carry adder)* (Рис.12).

Выход переноса P каждого разряда подключен ко входу переноса p соседнего старшего разряда. Входной перенос всего n-разрядного сумматора подается на вход p самого младшего разряда. Выходной перенос P самого старшего разряда является выходом переноса всего n-разрядного сумматора. В данном сумматоре тракты переносов всех одноразрядных сумматоров включены последовательно. Поэтому, даже при минимальной задержке тракта переноса одноразрядного сумматора в 1τ задержка n-разрядного сумматора не может быть менее $n\tau$.

• Сумматор с параллельным переносом.

Для уменьшения задержки используется принцип *параллельного переноса,* когда входной перенос каждого разряда вырабатывается независимо от переноса соседнего младшего разряда. Он формируется как функция только слагаемых и входного переноса $p_{груп}$ всего n-разрядного сумматора. Для всех разрядов сигналы переноса p формируются параллельно.

Задержка T получения суммы сумматора с параллельным переносом слагается из одинаковых для всех (кроме первого) разрядов задержки блока переноса — (2-3) τ в зависимости от логического базиса и задержки трехвходовой схемы сложения по модулю 2—4 τ. От числа разрядов ни задержка получения суммы, ни задержка получения выходного

переноса $P_{груп}$ не зависят. Аппаратурные затраты W сумматора с параллельным переносом заметно превышают W сумматора с последовательным переносом и быстро растут с ростом разрядности.

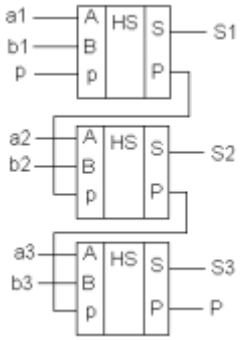


Рис.12

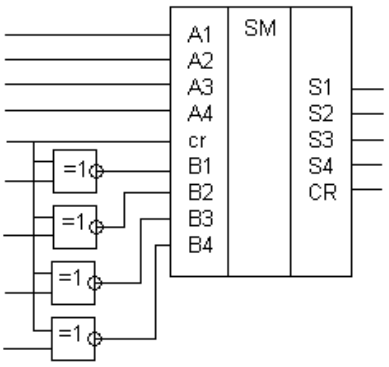


Рис.13

• Сумматоры с групповым переносом.

Для ускорения переноса в сумматорах с большим числом разрядов применяют принцип *группового переноса.* Сумматор разбивают на группы *(I ярус)*, представляющие собой небольшие сумматоры с разрядностью обычно от 2 до 8. Каждый такой мини-сумматор имеет свой штатный вход переноса p. Суть группового переноса заключается в том, что в дополнение к тракту переноса внутри группы, который в общем может быть как параллельным, так и последовательным, строят *тракт переноса II яруса* между группами, который вырабатывает сигналы *групповых переносов,* подаваемые на входы p всех мини-сумматоров. Тракт группового переноса удается построить так, что время распространения переноса в нем между группами оказывается меньше, чем если бы этот перенос распространялся по цепям внутригрупповых трактов *(трактов I яруса).*

С помощью сумматоров можно выполнять и действие вычитания. Для этого второе слагаемое (вычитаемое) нужно перевести в дополнительный код. Существуют универсальные устройства, выполняющие как вычитание, так и сложение. На рис. 13 приведена схема такого устройства. Оно имеет управляющий сигнал, который переводит **сумматор-вычитатель** в режим сложения или вычитания.

Операция вычитания не единственная, которую можно выполнять с помощью сумматора. Рассмотрим построение умножителя на базе сумматора на примере ИМС К155ИМ3. На рис. 14 показана схема для перемножения двух двоичных чисел: четырехразрядного $A=A_4A_3A_2A_1$ и трехразрядного $B=B_3B_2B_1$. Семиразрядное произведение на выходе формируется за счет параллельного умножения множимого на каждый разряд множителя логическими элементами 2И и сложения промежуточных произведений со сдвигом на один разряд—сумматором. M_7—бит (1 или 0) переноса из предыдущего разряда. Применение логических элементов И для выполнения арифметической операции умножения в данном случае закономерно, поскольку в рамках одного разряда и арифметическое умножение, и логическое (конъюнкция) подчиняются общим правилам.

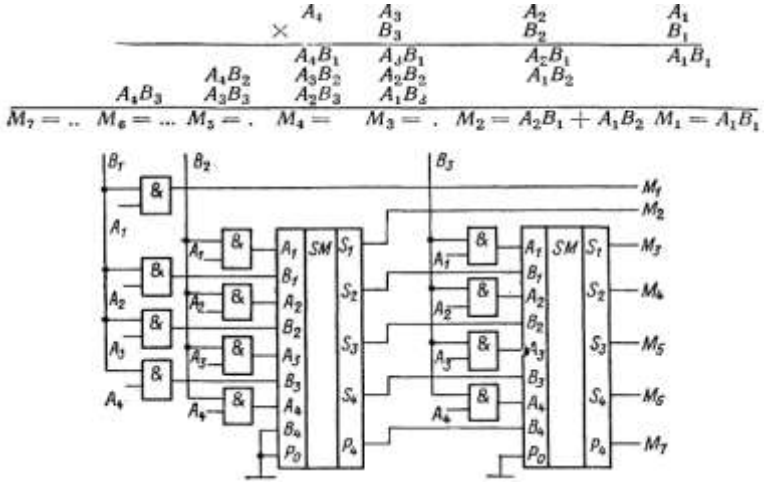
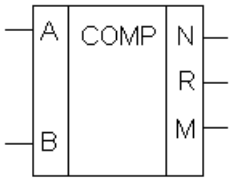


Рис.14.

Цифровые компараторы *(magnitude comparator)* выполняют сравнение двух чисел, заданных в двоичном (двоично-десятичном) коде. В зависимости от схемного исполнения компараторы могут определять равенство A = B (A и B—независимые числа с равным количеством разрядов) либо вид неравенства: A<B или A>B. Результат сравнения отображается соответствующим логическим уровнем на выходе. Компараторы выполняют, как правило, все эти операции и имеют три выхода. Цифровые компараторы широко применяются для выявления нужного числа (слова) в потоке цифровой информации, для отметки времени в часовых приборах, для выполнения условных переходов в вычислительных устройствах; в устройствах автоматики компараторы используются для сигнализации о выходе величин за пределы допуска, в приводах следящих систем для определения направления воздействия, ликвидирующего рассогласование, при построении счетчиков и сумматоров по произвольному основанию.

Логическая схема, выполняющая операцию «исключающее ИЛИ—НЕ», может быть использована как одноразрядный компаратор.

УГО интегрального одноразрядного компаратора и его таблица истинности приведены на рис.15. Компаратор имеет 2 входа (для числа A и B) и 3 выхода (N, R, M).



Входы		Выходы		
B	A	N A>B	R A=B	M A<B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Рис.15

Практические задания

1. Заполните таблицу.

Обозначение вывода	Назначение вывода
	Выход
	Адресные входы
	Вход управления третьим состоянием
	Информационные сигналы
	Стробирующий управляющий вход

2. Определите количество адресных линий мультиплексора, если он обозначен 24→1

3. Составьте УГО и таблицу истинности для мультиплексора 8→1.

4. Заполните приведенные строки таблицы истинности мультиплексора, изображенного на рис. 3

A3	A2	A1	A0	Y
0	0	0	0	
0	1	1	0	
1	0	0	0	
1	1	0	1	
0	1	1	1	

5. Определите количество выходных линий демультиплексора, если он имеет 6 адресных линий.

6. Составьте УГО и таблицу истинности для демультиплексора 1→8.

7. Заполните таблицу, используя справочники и интернет-ресурсы

Наименование	Функциональное назначение	Число входов / выходов	Число разрядов

К155КП1			
К155КП5			
К55КП17			
К155ИД3			
К531ИД7			
К564КП1			
К564КП2			

8. Постройте УГО, логическую схему и таблицу состояний шифратора 4х2.
9. Выберите примеры полных и неполных дешифраторов: 3х8, 3х6, 4х15, 5х32, 2х3.
10. Постройте УГО, логическую схему и таблицу состояний дешифратора 2х4.
11. Заполните таблицу, используя справочники и интернет-ресурсы

Наименование	Функциональное назначение	Кол-во входов	Кол-во выходов
К155ПР6			
К155ПР7			
К533ИВ2			
К555ИД3			
К555ИД10			

12. Составьте УГО двухразрядного сумматора
13. Постройте схему пятиразрядного последовательного сумматора.
14. Постройте схему умножителя 3х разрядного числа на 2х разрядное
15. Заполните таблицу, используя справочники и интернет-ресурсы

Наименование	Функциональное назначение
К155ИМ1	
К155ИМ2	
К155ИМ3	
К564ИП2	
К555СП1	

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. конспект теоретических сведений
3. практические задания;
4. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Практическая работа 5.
Изучение принципов организации запоминающих устройств

Цель: закрепить знания об основных видах запоминающих устройств, их характеристиках и построении.

Выполнив работу, Вы будете:

уметь:

- выбирать средства технической реализации микропроцессорных систем управления;

Материальное обеспечение:

не требуется

Теоретические сведения

Цифровые запоминающие устройства (ЗУ) предназначены для записи, хранения и выдачи информации, представленной в виде цифрового кода.

По функциональному назначению все виды ЗУ можно разделить на следующие группы:

• оперативные запоминающие устройства (ОЗУ, или RAM) — устройства памяти цифровой информации, объединенные со схемами управления, обеспечивающими режимы записи, хранения и считывания цифровой (двоичной) информации в процессе ее обработки;

• постоянные запоминающие устройства (ПЗУ, или ROM) — матрицы пассивных элементов памяти со схемами управления, предназначенные для воспроизведения неизменной информации, заносимой в матрицу при изготовлении (в режиме хранения информации энергия не потребляется);

• программируемые постоянные запоминающие устройства (ППЗУ, или PROM) — постоянные запоминающие устройства с возможностью однократного электрического программирования; они отличаются от ПЗУ тем, что позволяют в процессе применения микросхемы однократно изменить состояние запоминающей матрицы электрическим путем по заданной программе;

• репрограммируемые постоянные запоминающие устройства (РПЗУ, или EEPROM) — постоянные запоминающие устройства с возможностью многократного электрического перепрограммирования; они отличаются от ППЗУ тем, что допускают многократную электрическую запись информации, но число циклов записи и стирания ограничено (до 10^4 циклов);

• репрограммируемые постоянные запоминающие устройства с ультрафиолетовым стиранием и электрической записью информации (РПЗУ УФ, или EPROM), они отличаются от РПЗУ только способом стирания информации с помощью ультрафиолетового освещения, для чего в корпусе микросхемы имеется специальное окно;

• ассоциативные запоминающие устройства (АЗУ, или CAM) — «безадресные» ЗУ, в которых поиск и выборка информации осуществляется по содержанию произвольного количества разрядов хранящихся в АЗУ чисел, независимо от физических координат ячеек памяти.

По способу хранения информации ЗУ делятся на две группы:

1. статические – элементы памяти в них представляют собой бистабильные ячейки (триггеры). Бывают синхронными и асинхронными. Синхронные статические ЗУ имеют статический накопитель (матрицу элементов памяти) и динамические цепи управления, требующие синхронизации, аналогично динамическим ЗУ.

2. динамические – для хранения информации используются инерционные свойства реактивных элементов (например, конденсаторов), что требует периодического восстановления (регенерации) состояния элементов памяти в процессе хранения информации. В большинстве динамических ЗУ регенерация совмещается с обращением к элементам памяти.

По технологии выполнения ЗУ можно разделить на следующие виды:

• полупроводниковые ЗУ на основе биполярных структур (ТТЛ);

• полупроводниковые ЗУ на основе полевых транзисторов с изолированным затвором: p-МОП, n-МОП и КМОП;

• полупроводниковые ЗУ на основе приборов с зарядовой связью (ПЗС);

• магнитные ЗУ на основе цилиндрических магнитных доменов (ЦМД).

Для использования в РПЗУ разработаны специальные структуры:

• с лавинной инжекцией заряда и плавающим затвором (ЛИПЗ МОП), которые применяются в РПЗУ УФ;

• со структурой металл — нитрид кремния — окисел кремния — полупроводник (МНОП), которые используются в РПЗУ с электрическим стиранием, в том числе и с избирательным стиранием.

По способу обращения к массиву памяти все ЗУ делятся на:

1. адресные (большинство видов ЗУ) – обращение к элементам памяти производится по их физическим координатам, задаваемым внешним двоичным кодом-адресом. Адресные ЗУ бывают следующих типов:

• с произвольным обращением – допускают любой порядок следования адресов;

• с последовательным обращением, в которых выборка элементов памяти возможна только в порядке возрастания или убывания адресов (обычно такие ЗУ выполняются на регистрах сдвига).

2. безадресные (ассоциативные) – не имеют входов адресных сигналов: поиск и выборка информации в таких ЗУ осуществляется по ее содержанию и не зависит от физических координат элементов памяти.

Основными характеристиками запоминающих устройств являются: их информационная емкость, быстродействие и время хранения информации.

Все параметры ЗУ можно разделить на две группы:

1. Статические параметры ЗУ характеризуют его работу в установившемся режиме. Система статических параметров ЗУ представляет собой совокупность некоторых контрольных точек его вольт-амперных характеристик. Статические параметры ЗУ можно разделить на общие, входные и выходные. В табл. 1 приведены некоторые статические параметры ЗУ.

Таблица 1. Статические параметры ЗУ

Параметр	Обозначение
Напряжение питания	U_{CC}
Ток потребления	I_{CC}
Напряжение питания в режиме хранения	U_{CCS}
Ток потребления в режиме хранения	I_{CCS}
Напряжение логической «1»	U_H
Напряжение логического «0»	U_L

2. Динамические параметры ЗУ определяются происходящими в нем временными процессами. Систему динамических параметров ЗУ составляет совокупность временных переходов входных и выходных сигналов, соответствующих границам правильного функционирования ЗУ. К динамическим параметрам относятся основные временные характеристики ЗУ, такие как время выбора микросхемы tcs, время выбора адреса t_A, время выборки сигнала t_{RD} и некоторые другие.

Кроме этого используются также специальные классификационные параметры ЗУ, по которым выполняют их разделение по группам в соответствующих сериях ИМС ЗУ. В качестве классификационных параметров могут использоваться также некоторые статические и динамические параметры. В табл. 2 приведены основные классификационные параметры ЗУ.

Таблица 2. Основные классификационные параметры ЗУ

Параметр	Обозначение
Информационная емкость	N
Число слов в ЗУ	n
Разрядность	т
Коэффициент разветвления по выходу	К
Число циклов перепрограммирования	N_{CY}
Потребляемая мощность	P_{CC}
Потребляемая мощность в режиме хранения	P_{CCS}

Структурная схема и УГО ***статического ОЗУ*** приведены на рис.1. Основой статического ОЗУ является накопитель или матрица памяти, состоящая из отдельных запоминающих (бистабильных) ячеек (обычно в качестве этих ячеек используются различного рода триггеры). Двоичная информация, записанная в такую ячейку, может сохраняться в этой ячейке до тех пор, пока не будет заменена другой или не будет снято напряжение питания.

Накопитель или матрица памяти состоит из n строк. В состав каждой строки входят m запоминающих ячеек, образующих m-разрядное слово. Информационная емкость накопителя равна N=nm, где n — число строк (или слов), m — число столбцов (или разрядов). Соответствующие шины в накопителе управляются от дешифраторов строк (X) и столбцов (Y), на входы которых поступают адресные сигналы $A_0 \ldots A_N$. При записи и считывании осуществляется обращение (выборка) к одной или нескольким запоминающим ячейкам одновременно. Дешифраторы строк и столбцов выполняют выбор требуемых ячеек памяти с помощью адресных сигналов $X_0 \ldots X_n$ и $Y_0 \ldots Y_m$.

Устройство управления определяет режим работы схемы ОЗУ. По сигналу CS разрешаются или запрещаются операции записи и считывания. Сигнал CS позволяет выбрать требуемую микросхему памяти в ЗУ, состоящем из ряда микросхем. Подача сигнала на вход WR/RD при наличии сигнала CS=0 выбора микросхемы позволяет выбрать режим записи, если WR/RD=0, или считывания, если WR/RD=l.

Данные, подлежащие записи, поступают на вход DI, а данные, подлежащие чтению, снимаются с выхода DO. Устройства записи и считывания обеспечивают прием и выдачу сигналов информации с уровнями, согласующимися с серийными цифровыми микросхемами.

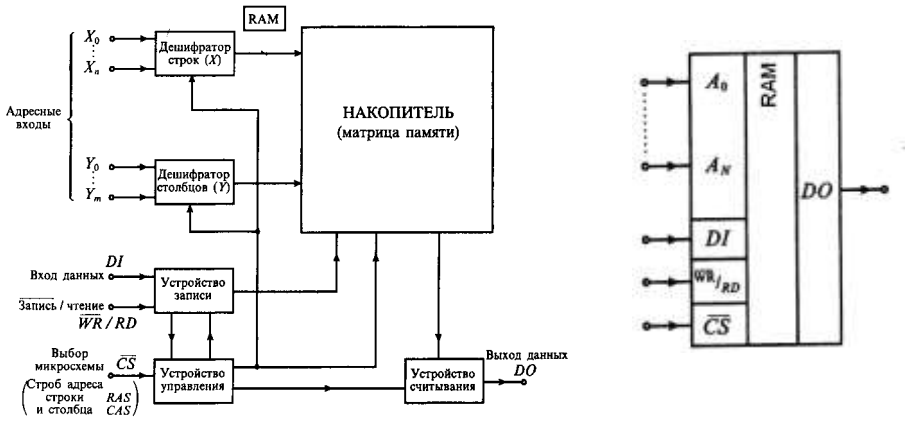


Рис. 1

По режиму питания статические ОЗУ можно разделить на 2 группы:

а) с активным режимом питания – накопитель и схема управления потребляют практически одинаковую мощность при всех операциях: записи, считывания и хранения информации.

б) с активно-пассивным режимом питания - некоторые узлы переходят в режим малого потребления или полностью отключаются, если микросхема находится в режиме хранения информации. В результате при хранении информации потребляемая микросхемой мощность уменьшается. При переходе в режим записи или считывания напряжения и токи питания восстанавливаются до номинальных значений. Использование активно-пассивного режима питания в несколько раз уменьшает среднюю мощность, потребляемую микросхемой. По этой причине большинство микросхем ОЗУ используют такой режим.

Для увеличения информационной емкости широко используются ***динамические ОЗУ***, в которых информация хранится в виде заряда соответствующих емкостей. При токе утечки обратно смещенного p-n-перехода около 10^{-10} А и емкости хранения 0,1пФ время хранения не превышает 1мс. В связи с этим необходимо восстановление (регенерация) хранимой информации с периодом не более 1 мс. Емкостные ячейки памяти выполняются или на биполярных, или на МОП – транзисторах.

Для динамических ОЗУ характерны некоторые особенности, которые существенно отличают их от статических:

- для выполнения регенерации заряда необходимы соответствующие блоки;
- малая потребляемая мощность;

- последовательная адресация - вначале на адресный вход подается строб адреса строки RAS, а затем строб адреса столбца CAS. Для этих стробов имеются специальные выводы микросхемы, которые показаны на структурной схеме (см. рис.46).
- для управления динамическим ОЗУ необходимы последовательности импульсов, которые обычно формируются специальными генераторами.

По способу регенерации микросхемы динамических ОЗУ делятся на:

- адресные – производится перебор регенерируемых ячеек так, чтобы за период регенерации восстановить заряды во всех ячейках; адресные сигналы поступают в регистры, а затем на дешифраторы адресов.

- безадресные - заряды восстанавливаются во всех ячейках при помощи специальных тактовых импульсов.

Устройство типовой ячейки памяти и УГО динамического ОЗУ приведено на рис. 2. Хранение информации происходит в емкости C_{GS} (затвор — исток) полевого транзистора, а транзистор VT1 выполняет роль ключа выборки. Сохранность информации при выборке и хранении обеспечивается при помощи усилителя-регенератора. Режим хранения обеспечивается периодической регенерацией заряда емкости C_{GS} с частотой около сотни герц. В процессе регенерации уменьшение заряда на емкости C_{GS} компенсируется усилителем регенератором.

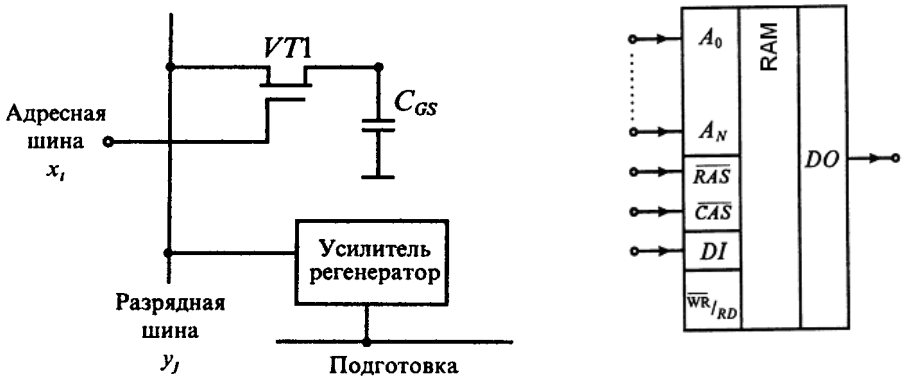


Рис.2

Микросхемы ***ПЗУ*** можно разделить на две группы:

- однократно программируемые – информация после записи меняться не может, и микросхема работает только в режиме считывания.

- перепрограммируемые.

Структурная схема ПЗУ приведена на рис. 3. От ОЗУ ПЗУ отличается отсутствием устройства записи и линий, которые его обслуживают. Кроме того, изменяется выполнение накопителя (матрицы памяти). В настоящее время находят применение два типа накопителей ПЗУ:

- масочные ПЗУ (МПЗУ) – накопитель программируется на стадии изготовления, когда информация, записываемая в него, определяется построением одного из слоев схемы при помощи специального фотошаблона.
- программируемые ПЗУ (ППЗУ) – накопитель выполняют на базе ЗЯ с плавкими перемычками; их упрощенная схема приведена на рис. 50. При программировании эти плавкие перемычки пережигают с помощью специального программирующего устройства т.о. процесс записи информации в схему представляет собой избирательное разрушение плавких перемычек током, обеспечиваемым устройством программирования.

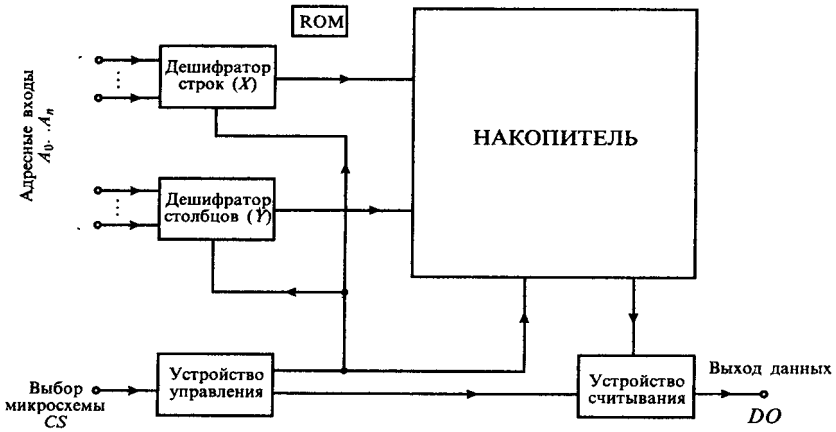


Рис.3

На рис. 4 плавкие перемычки ПП показаны в виде предохранителей, включенных в эмиттеры многоэмиттерных транзисторов $VT_0 \ldots VT_n$. Программируемые элементы включены между эмиттерами транзисторов матриц и разрядными шинами. Наличие перемычки соответствует логическому 0 на выходе усилителя считывания, а отсутствие перемычки — логической единице. Пережигание перемычек в режиме программирования выполняется серией импульсов по специальной программе.

Для повышения надежности работы ПЗУ методика программирования предусматривает подачу серии 40… 100 импульсов после фиксации момента пережигания перемычки, а также обязательную термотренировку запрограммированного ПЗУ при определенной температуре (около 100°С) в заданном электрическом режиме. На рис.4 приведено УГО ПЗУ.

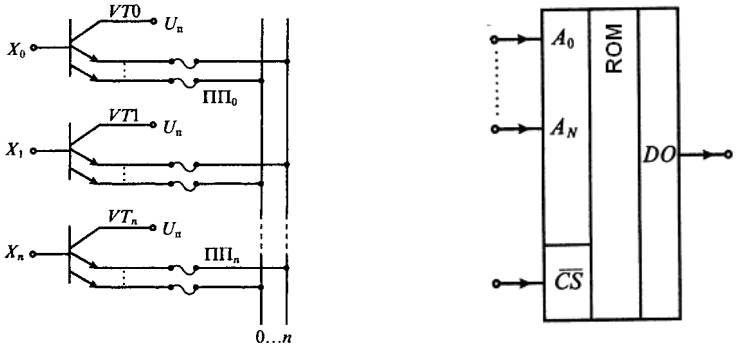


Рис.4

Запоминающие ячейки ***РПЗУ*** строятся на n-МОП или КМОП транзисторах. В них используются физические явления хранения заряда на границе между двумя различными диэлектрическими средами или проводящей и диэлектрической средой. В такой структуре при высоком напряжении на затворе (около 30 В) внутри МОП структуры образуется некоторый заряженный слой, который приводит к изменению порогового напряжения МОП транзистора (электрическое программирование). При постоянном напряжении на затворе в режиме считывания информации это приводит к изменению тока считывания.

Поскольку затвор транзистора со всех сторон окружен изолирующим слоем, ток утечки очень мал и хранение информации достаточно длительное (десятки лет).

По способу стирания информации РПЗУ делятся на две группы:

1) с ультрафиолетовым стиранием (EPROM) – в таких приборах пользуются облучением кристалла через специальное прозрачное окно в корпусе микросхемы ультрафиолетовым светом, что приводит к резкому увеличению тока утечки и способствует рассасыванию носителей заряда.

2) с электрическим стиранием EEPROM (в том числе РПЗУ с избирательным стиранием EAROM) - над плавающим затвором размещают второй— управляющий — затвор. Подача напряжения на него приводит к рассасыванию заряда за счет туннельного эффекта. Эти РПЗУ имеют преимущества перед РПЗУ УФ, так как не требуют при

перепрограммировании специальных источников ультрафиолетового света. Структурная схема такого РПЗУ с шинным управлением приведена на рис. 5.

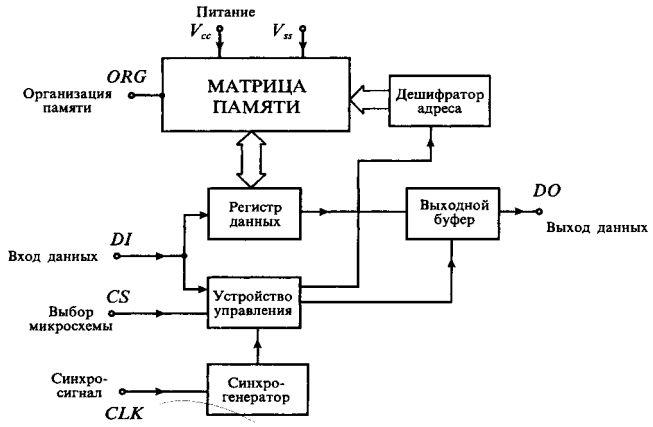


Рис.5

Практические задания

1. Схематически изобразите классификацию ЗУ.
2. Заполните таблицу.

Обозначение вывода	Назначение вывода
$X_0 - X_n$	
$Y_0 - Y_m$	
DI	
WR	
RD	
CS	
DO	

3. Составьте сравнительную таблицу характеристик статических и динамических ОЗУ, выбрав в качестве сравниваемых параметров тип ячейки памяти, принцип хранения информации, вид управления, входные и выходные сигналы, способ обращения к ячейкам памяти, потребляемую мощность.
4. Заполните таблицу, используя справочники и интернет-ресурсы

Наименование	Функциональное назначение	Объем, Кб
КР541РУ1А		
К573РФ1		
КР558РР1		
КР568РЕ1		

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. конспект теоретических сведений
3. практические задания;
4. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Практическая работа 6.
Изучение схемы типовой МПС

Цель работы: изучить принципы построения и функционирования МПС на примере МПС серии КР580

Выполнив работу, Вы будете:

уметь:

- выбирать средства технической реализации микропроцессорных систем управления;

Материальное обеспечение:

не требуется

Теоретические сведения

Комплект микросхем серии КР580, выполненных по n-МДП- и ТТЛШ - технологии, характеризуется архитектурным единством, которое обеспечивается автономностью и функциональной законченностью отдельных микросхем, унификацией их интерфейса, программируемостью микросхем, их логической и электрической совместимостью. Восьмиразрядная организация, фиксированный набор команд, большой выбор периферийных микросхем различного назначения, относительно высокое быстродействие, умеренное потребление мощности обеспечивают МПК широкое применение при создании средств вычислительной техники, устройств локальной автоматики, контроллеров измерительных приборов и периферийных устройств. микро-ЭВМ для управления технологическими процессами и измерительными системами и др. Состав МПК серии КР580 приведен в табл. 1.

Таблица 1 - Состав серии КР580

Тип микросхем	Функциональное назначение
КР580ВМ80А	Однокристальный 8-разрядный микропроцессор
КР580ВВ51А	Программируемый последовательный интерфейс
КР580ВИ53	Программируемый таймер
КР580ВВ55А	Программируемый параллельный интерфейс
КР580ВТ57	Контроллер прямого доступа к памяти
КР580ВН59	Контроллер прерываний
КР580ВВ79	Интерфейс клавиатуры
КР580ВГ75	Контроллер ЭЛТ
КР580ВК91А	Интерфейс МП-канал общего пользования
КР580ВА93	Приемопередатчик МП-канал общего пользования
КР580ГФ24	Генератор тактовых сигналов
КР580ВК28, КР580ВК38	Системный контроллер и шинный формирователь
КР580ИР82, КР580ИР83	Буферный регистр/регистр с инверсией
КР580ВА86, КР580ВА87	Шинный формирователь/формирователь с инверсией

Типовая электрическая принципиальная схема микропроцессорной системы на базе микросхем серии КР580 приведена на рис. 1. Число и состав микросхем в системе определяются требованиями, предъявляемыми потребителем. Необходимыми компонентами в любой системе являются:

- микропроцессор КР580ВМ80А,
- генератор КР580ГФ24,
- системный контроллер КР580ВК28 (КР580ВК38),
- буферная схема адреса (построена на двух микросхемах КР580ВА86 (КР580ВА87) для обеспечения нагрузочной способности по шине адреса).

Объем памяти ЗУ и использование одной или нескольких периферийных микросхем КР580ВВ51А, КР580ВИ53, КР580ВВ55А, КР580ВТ57, КР580ВН59, КР580ВВ79 или КР580ВГ75 определяет пользователь.

Микропроцессорная система имеет системную шину, образуемую из трех шин:

- адреса А15—А0,
- данных D7—D0,
- управления.

Системная шина позволяет строить микропроцессорную систему по модульному принципу: модуль центрального процессора, модуль ЗУ, модуль УВВ и т. д. Каждый модуль может содержать собственные буферные схемы адреса и данных. Двунаправленные выводы данных периферийных микросхем рекомендуется подключать к системной шине через шинные формирователи (КР580ВА86, КР580ВА87 или КР589АП16, К589АП26). Магистральная структура микропроцессорной системы позволяет подключать микросхемы ЗУ общей емкостью до 64К байт и микросхемы УВВ до 256 каналов ввода и до 256 каналов вывода.

Для помехоустойчивости системы низкочастотные помехи по цепи питания необходимо блокировать конденсатором суммарной емкостью из расчета 0,1 мкФ на каждую микросхему, включенным между шинами +5 В и GND непосредственно в начале шины -5 В.

Высокочастотные помехи необходимо блокировать конденсатором емкостью 0,015—0,022 мкФ, включенным между каждым выводом +5 В микросхемы и шиной GND в непосредственной близости от микросхем (не далее 5 мм).

Для увеличения быстродействия системы трехстабильные линии шины адреса и данных рекомендуется подключать к шинам +5 В через резисторы сопротивлением 2,2 кОм.

Микросхемы серии КР580 по входам и выходам совместимы с микросхемами ТТЛ серий К133 и К155.

Основные стыковочные параметры ИМС серии КР580 даны в табл. 2.

Таблица 2 – Параметры ИМС

Параметр	Обозначение	Значения параметров [макс (мин.)]
Напряжение питания, В	U_{CC}	5,25(4,75)
Входное напряжение низкого уровня, В	U_{IL}	0,8
Входное напряжение высокого уровня, В	U_{IH}	(2,0)
Выходное напряжение низкого уровня, В	U_{OL}	0,45
Выходное напряжение высокого уровни, В	U_{OH}	(2,4)
Выходной ток низкого уровня, мА	I_{OL}	2,2
Выходной ток высокого уровня, мА	I_{OH}	-0,4
Ток утечки на входах, мкА	I_{LI}	±10
Ток утечки на входах/выходах, мкА	I_{OZ}	±10
Емкость нагрузки, пФ	C_L	100
Емкость на входах, пФ	C_I	10
Емкость на входах/выходах, пФ	C_O	20

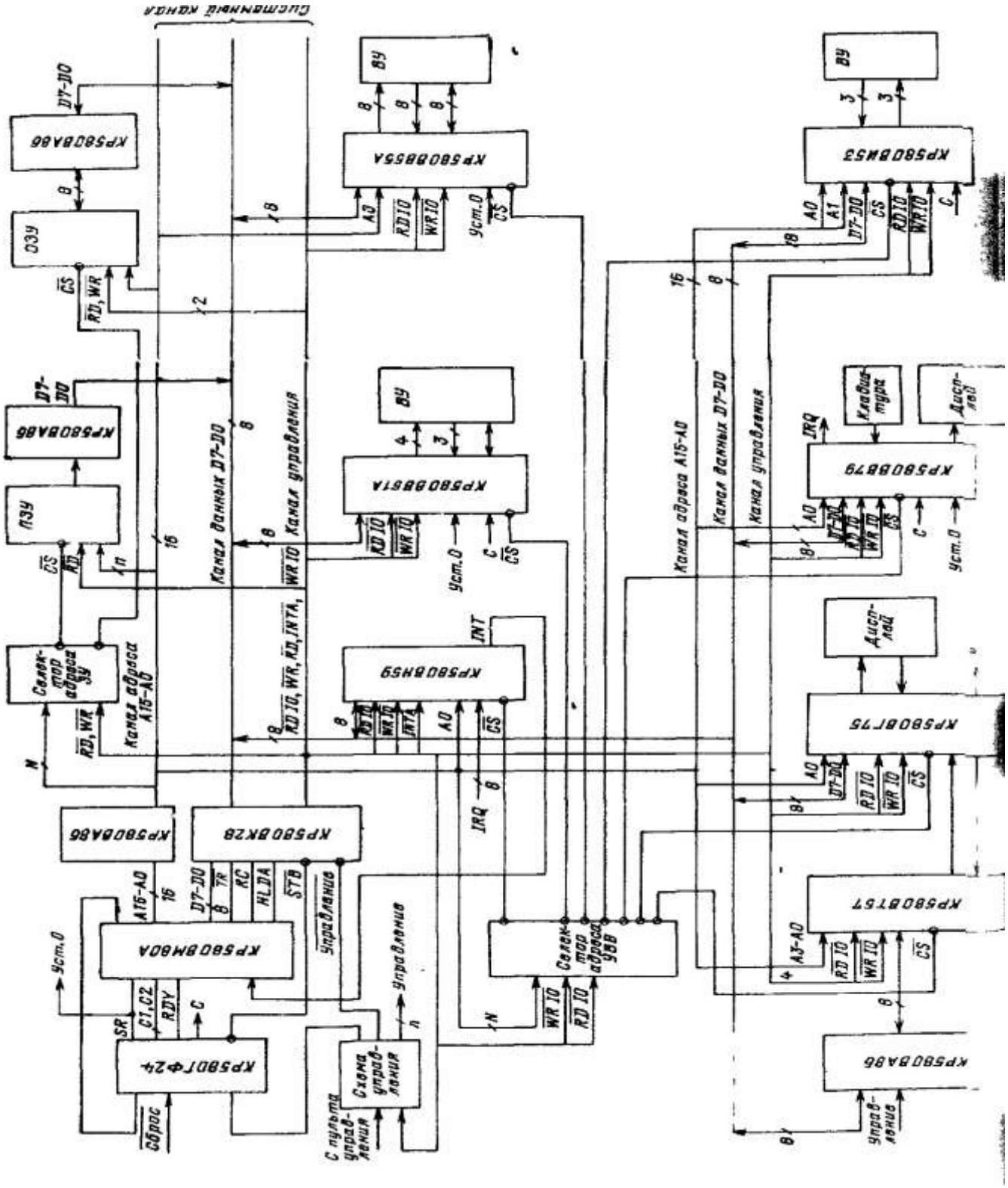


Рис.1 – Типовая электрическая принципиальная схема МПС на основе МП К580ВМ80А

Микросхема КР580ГФ24 — генератор тактовых сигналов фаз Cl, C2, предназначен для синхронизации работы микропроцессора КР580ВМ80А.

Генератор формирует:

- две фазы Cl,C2 с положительными импульсами, сдвинутыми во времени, амплитудой 12 В и частотой 0,5—3,0 МГц;
- тактовые сигналы опорной частоты амплитудой напряжения уровня ТТЛ;
- стробирующий сигнал состояния STB длительностью не менее ($T_{ОП}$/9—15 нс), где $T_{ОП}$— период тактовых сигналов опорной частоты;

- тактовые сигналы С, синхронные с фазой С2, амплитудой напряжения уровня ТТЛ.

Генератор синхронизирует сигналы RDYIN и RESIN с фазой С2. Стробирующий сигнал состояния STB формируется при наличии на входе SYN напряжения высокого уровня, поступающего с выхода микропроцессора КР580ВМ80А в начале каждого машинного цикла. Сигнал STB используют для занесения информации состояния микропроцессора в микросхему КР580ВК28 или КР580ВК38 для формирования управляющих сигналов.

Для согласования работы микропроцессора КР580ВМ80А с другими устройствами сигнал RDYIN синхронизируется по фазе С2 на выходе RDY генератора.

Выходной сигнал SR пользуют для установки в исходное состояние микропроцессора и других микросхем в системе.

Для автоматической установки микропроцессора КР580ВМ80А в исходное состояние при подаче напряжений питания ко входу RESIN микросхемы КР580ГФ24 подключают цепь, состоящую из элементов R,VD, С2.

Задание

Изучить принципиальную схему МПС (назначение устройств и принцип их соединения) и нарисовать обобщенную структурную схему МПС с указанием типовых устройств, входящих в состав МПС

Контрольные вопросы

1. Каково назначение ИМС КР580ВМ80А? Покажите на схеме.
2. Каково назначение ИМС КР580ВВ51А? Покажите на схеме.
3. Каково назначение ИМС КР580ВВ55? Покажите на схеме.
4. Каково назначение ИМС КР580ВИ53А? Покажите на схеме.
5. Каково назначение ИМС КР580ВА86? Покажите на схеме.
6. Каково назначение ИМС КР580ГФ24? Покажите на схеме.
7. Какие шины входят в состав системной шины в данной МПС?
8. Каким схемным способом выполняется защита от помех в данной МПС?
9. Как осуществляется установка в исходное состояние МП и других ИМС в системе?
10. Какую функцию выполняет ИМС КР580ВК28? Покажите на схеме.

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Перечень обязательных компонентов МПС и их назначение.
3. Структурная схема МПС
4. Ответы на контрольные вопросы

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 МПС на основе программируемых логических контроллеров (ПЛК)

Лабораторная работа 1.
Программирование ПЛК на языке LD

Цель работы: Изучить принципы составления прикладных программ для промышленных логических контроллеров на языке LD пакета CoDeSys.

Выполнив работу, Вы будете:

уметь:

- составлять функциональные и структурные схемы управления различными электроэнергетическими объектами;
- выбирать средства технической реализации микропроцессорных систем управления;
- программировать микропроцессорные системы управления на основе ПЛК широкого применения.

Материальное обеспечение:

Стенд ПЛК «ОВЕН», персональный компьютер со специализированным ПО CoDeSys

1. Общие сведения.

Язык LD (Ladder Diagram) или РКС (Релейно-Контактные Схемы) представляет собой графическую форму записи логических выражений в виде контактов и обмоток реле. LD предназначен для программирования промышленных контроллеров (ПЛК).

Синтаксис языка удобен для замены логических схем, выполненных на релейной технике. Слева и справа схема ограничена вертикальными линиями – шинами питания. Между ними расположены цепи, образованные контактами и обмотками реле, по аналогии с обычными электронными цепями. Слева любая цепь начинается набором контактов, которые посылают слева направо состояние «ON» или «OFF», соответствующие логическим значениям ИСТИНА или ЛОЖЬ.

Каждому контакту соответствует логическая переменная. Если переменная имеет значение ИСТИНА, то состояние передается через контакт. Иначе правое соединение получает значение выключено ("OFF").

Контакты могут быть соединены параллельно, тогда соединение передает состояние логическое ИЛИ. Если контакты соединены последовательно, то соединение передаёт логическое И.

Контакт может быть инвертируемым. Такой контакт обозначается с помощью символа |/| и передает состояние "ON", если значение переменной ЛОЖЬ.

LD позволяет:

-выполнять последовательное соединение контактов;

-выполнять параллельное соединение контактов;

- применять нормально разомкнутые или замкнутые контакты;
- использовать переключаемые контакты;
- записывать комментарии;
- включать Set / Reset-выходы;
- переходы;
- включать в диаграмму функциональные блоки;
- управлять работой блоков по входам EN.

2. Порядок выполнения работы

Из ПРИЛОЖЕНИЯ 1 по заданию преподавателя выбирается вариант технологической схемы.

На основании заданной технологической схемы и описания технологического процесса разработать:

1. технологические требования к схеме управления;
2. принципиальную электрическую схему автоматического управления технологической установкой;

3. прикладную программу для ПЛК.
4. дать описание работы принципиальной схемы.

Схема и программа должна предусматривать:

1. запуск всех машин и механизмов в последовательности, направленной против движения продукта;
2. остановку всех машин и механизмов в последовательности, совпадающей с направлением движения продукта;
3. остановку поточных линий по команде «рабочий стоп» с целью очистки тракта;
4. режим пуско-наладочных работ;
5. звуковой или световой сигнал при пуске сложных технологических установок;
6. аварийное отключение (при аварийном отключении одной из машин, должны остановиться без выдержки времени все машины, работающие на ее загрузку, а с выдержкой времени все машины работающие на отгрузку).

3. Составление прикладной программы.

1. При создании программы используется среда программирования **CoDeSys V2.3** (далее – **CoDeSys**).

Перед созданием проекта пользователь, используя утилиту **InstallTarget** в составе **CoDeSys**, устанавливает для применяемого контроллера файл целевых задачи **(Target,файл)**, который обеспечивает программный доступ к ресурсам ПЛК.

2. Создание проекта программы

При создании проекта используется язык релейных диаграмм **LD (Ladder Diagram)**, реализующий структуры, подобные электрическим цепям в коммутационной автоматике.

Пользователь запускает **CoDeSys** последовательным выбором приложений:

Пуск →Все программы→ 3S Software →CoDeSys V2.3 →CoDeSys V2.3.

Новый проект открывается из главного меню: **File New**. В открывшемся окне (рис .1) выбирается тип контроллера, **PLC150.U-L**, выбор подтверждается нажатием клавиши **ОК**.

Рисунок 1.1- Окно конфигурации «Target Settings» программы

После выбора проекта выводится экранная форма, задающая тип, имя и язык программирования первичного компонента **New POU**, главной программы контроллера**.** Необходимо выбрать язык программирования LD, установив флаги в позициях, указанных на рис.1.2.

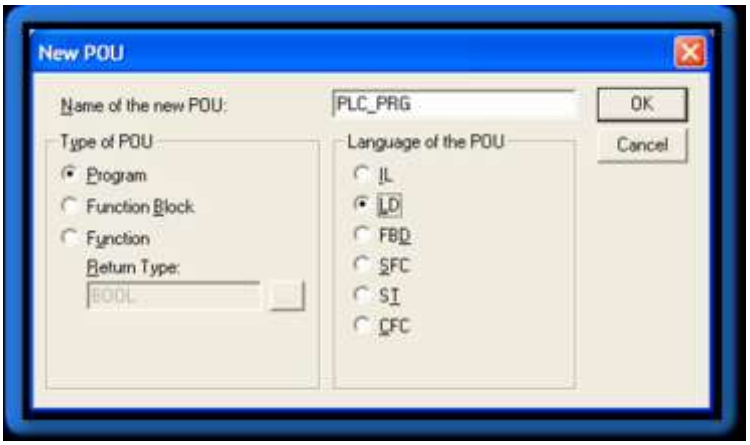


Рисунок 1.2- Вид окна «New POU» с отмеченными параметрами

Примечание. *Имя главной программы PLC_PRG и ее тип менять нельзя.*

После подтверждения выбора нажатием клавиши **ОК** откроется окно нового проекта с именем по умолчанию **Untitled**. В нем присутствует одна вкладка **POUs.**

Весь проект хранится в одном файле, имя которого отображается в заголовке окна. Для ввода имени файла во второй строке меню быстрого запуска активизируется клавиша записи и в появившейся форме указывается имя файла: например, **LD, проект 1.pro**.

4. Параметры входов и выходов контроллера

Цепям контроллера, используемым в разрабатываемой электрической схеме, присваиваются имена переменных. В дальнейшем эти имена используются в программе для работы с конкретным входом или выходом контроллера.

Для присвоения имени какому-либо ресурсу ввода/вывода контроллера необходимо на вкладке ресурсов **(Resources)** Организатора объектов **CoDeSys** запустить утилиту **PLC Configuration (Конфигуратор ПЛК)** .

В появляющейся иерархической структуре – дереве **Конфигурации ПЛК** – пользователь открывает папки (модули) входов **(Discrete input)** и выходов **(Discrete output)** ПЛК, и именует необходимые каналы. Перед адресом указывается имя (идентификатор переменной) для цепей входов и выходов схемы созданного проекта.

Именование канала (входа или выхода) производится следующим образом: двойным щелчком манипулятора «мышь» при курсоре, установленном в начале строки названия канала, осуществляется переход в режим редактирования и вводится имя переменной канала.

Экранная форма, представленная иллюстрируют выполненныепользователем именования каналов при использовании четырех входов **(IX0.0.4, IX0.0.5, IX0.0.6, IX0.0.7)** и одного выхода **(QX1.0).**

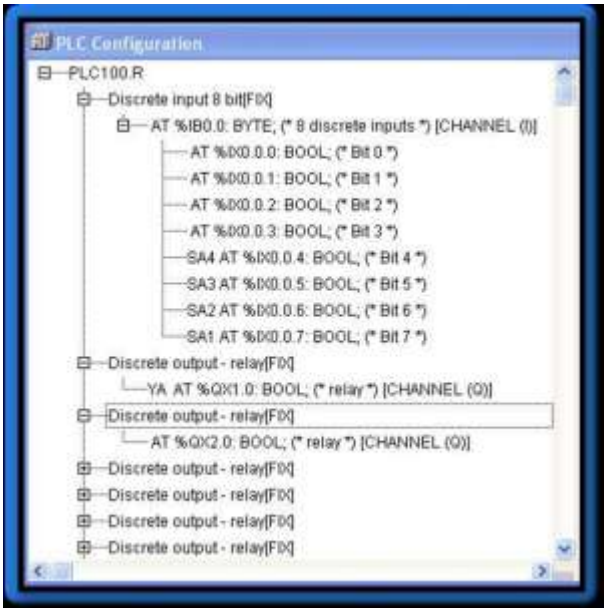


Рисунок 1.3- Экранная форма для именования входов и выходов при работе программы с цепями ПЛК

5. Создание программы на языке LD

При написании программы в рабочей зоне вкладки **POUs** последовательно вводятся типы компонентов и их обозначения, как это представлено на рис.1. 4.

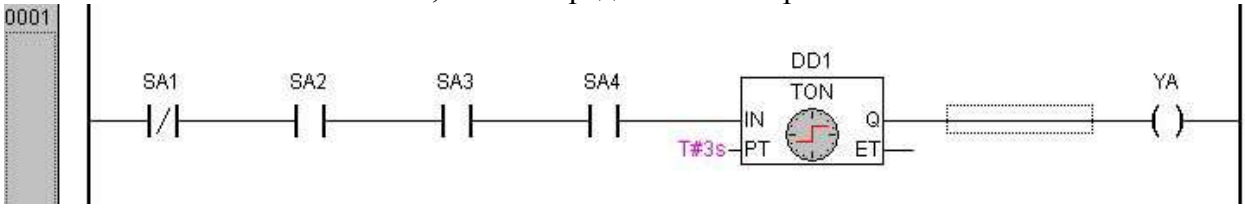


Рисунок 1.4. Пример программы на языке LD

Пользователь при составлении виртуальной схемы может следовать приведенной ниже инструкции.

Инструкция по созданию программы:

1) **создание нормально замкнутого контакта:** в контекстном меню выбрать команду **Contact (negated)** или нажать кнопку на панели инструментов. Символы вопросов (рис. 1.5 (а)) необходимо заменить именем, например **SA1**.

Описывать переменную в данном случае не требуется, так как она уже была указана в окне PLC(Configuration и связана с конкретным дискретным входом;

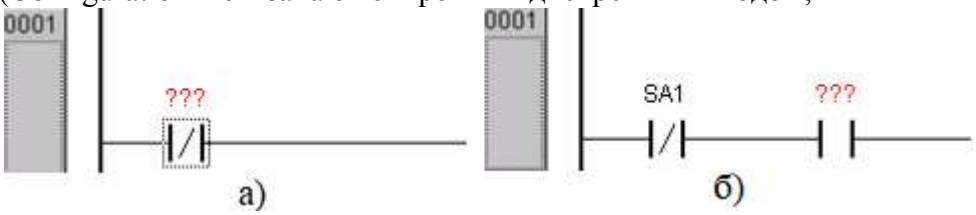


Рисунок 1.5 - Создание нормально замкнутого (а)
и разомкнутого (б) контактов

2) **создание нормально разомкнутого контакта** делается аналогичным образом, только используется команда контекстного меню **Contact** или кнопка на панели инструментов (рис.1.5(б));

3) **функциональный блок**: из контекстного меню выбирается команда **Function Block…**, – в появившемся окне **«Input Assistant»** (рис.1.6) из раздела **Standard Function Blocks** в библиотеке с именем **STANDARD.LIB** в папке **Timer** выбирается вид таймера – **TON (FB)**. На схеме перед входом **PT** указывается время задержки в формате **T#3s**. Над блоком вводится имя, например **DD1** и на клавиатуре нажимается клавиша «стрелка вправо» подтверждаются свойства функционального блока;

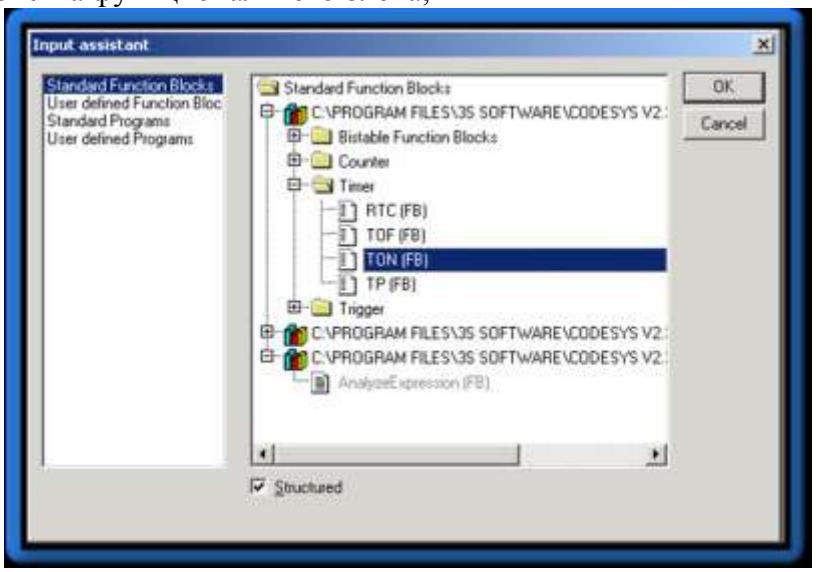


Рисунок 1.6 - Выбор таймера

5) **указание выхода цепи:** в контекстном меню выбирается команда **Coil** или нажимается кнопка  на панели инструментов. На схеме появляется условное обозначение обмотки реле. Символы вопросов замещаются именем.

6. Запись программы в контроллер

Настройка соединения ПК с ОВЕН ПЛК для загрузки и проверки работы программы в автономном режиме производится следующим образом.

Для информационного обмена ПК с ОВЕН ПЛК используется кабель программирования, входящий в комплект поставки. Им соединяются **COM**(порт компьютера и порт **Debug RS,232** контроллера (на лицевой панели).

Для настройки канала соединения из основного меню **CoDeSys** выбирается команда **Online-Communication parameters**. В диалоговом меню командой **New…** открывается диалоговое окно, в котором соединению присваивается имя (например, **COM**) и выбирается (из перечня) вид соединения **Serial (RS232)**. Выбор подтверждается нажатием клавиши **ОК**.

После указанных действий в окне коммуникационных параметров появляется канал соединения с именем **COM.** В зоне настроек **(Value)** для параметра **Baudrate** устанавливается значение **115200** (бит/сек – скорость соединения с компьютером). Значение может быть изменено двойным щелчком левой кнопки манипулятора «мышь» на значении. Для сохранения нового значения в окне курсором мыши выбор подтверждается нажатием клавиши **ОК**.

Программное соединение с ОВЕН ПЛК включается из главного меню **CoDeSys** командой **Online** - **Login**. При этом флаг перед строкой меню **Simulation Mode** должен быть снят.Как только система устанавливает связь с ОВЕН ПЛК, появляется запрос на подтверждение загрузки новой программы, пользователь подтверждаетзагрузку: **Да.**

После завершения записи проекта в оперативную память ОВЕН ПЛК, запуск работы программы осуществляется выбором команды **Online** - **Run** (или нажатием на лицевой панели ОВЕН ПЛК кнопки **<Старт>**).

Управлять работой ОВЕН ПЛК можно при помощи переключателей.

Форма представления результата:

1. технологические требования к схеме управления;
2. принципиальная электрическую схему автоматического управления технологической установкой;
3. прикладная программа для ПЛК.
4. описание работы принципиальной схемы
5. выводы по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторная работа 2
Программирование ПЛК на языке ST.

Цель работы: Изучить принципы составления прикладных программ для промышленных логических контроллеров на языке ST пакета CoDeSys.

Выполнив работу, Вы будете:

уметь:

- составлять функциональные и структурные схемы управления различными электроэнергетическими объектами;
- выбирать средства технической реализации микропроцессорных систем управления;
- программировать микропроцессорные системы управления на основе ПЛК широкого применения.

Материальное обеспечение:

Стенд ПЛК «ОВЕН», персональный компьютер со специализированным ПО CoDeSys

1. Общие сведения.

Язык ST (Structured Text) — это язык высокого уровня. Синтаксически ST представляет собой несколько адаптированный язык Паскаль. Вместо процедур Паскаля в ST используются компоненты программ стандарта МЭК. На основе ST можно создавать гибкие процедуры обработки данных.

Основой ST-программы служат выражения. Результат вычисления выражения присваивается переменной при помощи оператора «:=», как и в Паскале. Каждое выражение обязательно заканчивается точкой с запятой «;». Выражение состоит из переменных констант и функций, разделенных операторами:

iVarl := 1 + iVar2 / ABS(iVar2);

Стандартные операторы в выражениях ST имеют символьное представление, например математические действия: +, -, *, / операции сравнения и т.д.

Помимо операторов, элементы выражения можно отделять пробелами и табуляциями для лучшего восприятия.

Таблица 1 - Операторы в ST

Операция	Обозначение	Приоритет
Выражение в скобках	**(Выражение)**	**Самый высокий**
	Имя функции (список параметров)	
	EXPT	
	-	
	NOT	
	*****	
	/	
	MOD	
	+	
	<, >, <=, >=	
	<, >	
	=	
	AND	
	XOR	
	OR	**Самый низкий**

Основные типовые конструкции в ST

Оператор присваивания

Перед оператором присваивания находится операнд (переменная или адрес), которому присваивается значение выражения, стоящего после оператора присваивания.

Пример:

Var1: = Var2 * 10;

После выполнения этой операции Var1 принимает значение в десять раз большее, чем Var2.

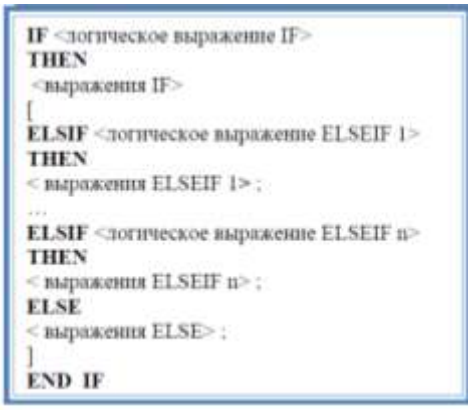
Вызов функционального блока в ST

Функциональный блок вызывается с помощью имени экземпляра функционального блока и списка входных параметров с присваиванием данных в круглых скобках. В следующем примере вызывается таймер с параметрами IN и PT. Значение выходной переменной Q присваивается переменной A. К выходной переменной можно обратиться с помощью имени экземпляра функционального блока, точки, следующей за ним и имени выходной переменной:

```
CMD_TMR (IN: = %IX5, PT: = 300);
A: =CMD_TMR.Q
```

Оператор выбора IF

Оператор выбора позволяет выполнить различные группы выражений в зависимости от условий, выраженных логическими выражениями. Полный синтаксис оператора IF (если) выглядит так:

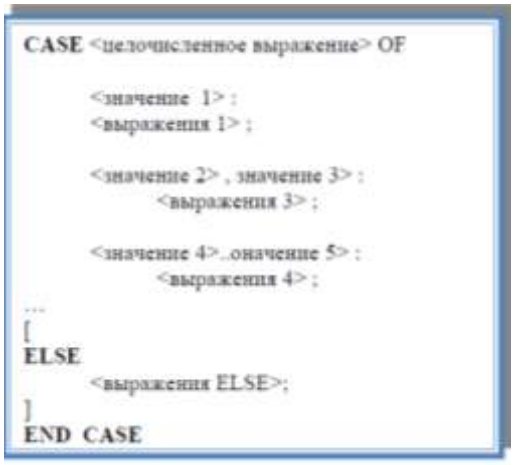


Если <логическое выражение IF> ИСТИНА, то выполняются выражения первой группы - <выражения IF>. Прочие выражения пропускаются, альтернативные условия не проверяются. Часть конструкции в квадратных скобках является необязательной и может отсутствовать. Если <логическое выражение IF> ЛОЖЬ, то одно за другим проверяются условия ELSIF. Первое истинное условие приведет к выполнению соответствующей группы выражений. Прочие условия **ELSIF** анализироваться не будут. Групп **ELSIF** может быть несколько или не быть совсем. Если все логические выражения дали ложный результат, то выполняются выражения группы **ELSE,** если, она есть. Если группы **ELSE** нет, то не выполняется ничего. В простейшем случае оператор **IF** содержит только одно условие:

IF bReset THEN
iVarl := 1;
iVar2 := 0;
END_IF

Оператор множественного выбора CASE

Оператор множественного выбора CASE позволяет выполнить различные группы выражений в зависимости от значения одной целочисленной переменной или выражения. Синтаксис:



Если значение выражения совпадает с заданной константой, то выполняется соответствующая группа выражений. Прочие условия не анализируются (<значение 1>: <выражения 1> ;). Если несколько значений констант должны соответствовать одной группе выражений, их можно перечислить через запятую (<значение2> , <значение3> : <выражения 3> ;). Диапазон значений можно определить через две точки (<значение 4>..<значение 5> : <выражения 4> ;).

Группа выражений ELSE является необязательной. Она выполняется при несовпадении ни одного из условий (<выраженияELSE> ;).

Пример:

CASE byLeft/2 **OF**
0,127:
bReset := TRUE;
Varl :=0;
16..24:
Varl :- 1;
ELSE

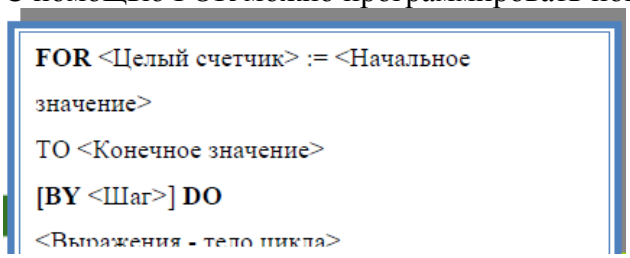
Varl := 2;

END_CASE

Значениями выбора **CASE** могут быть только целые константы, переменные использовать нельзя. Одинаковые значения в альтернативах выбора задавать нельзя, даже в диапазонах.

Цикл FOR

С помощью FOR можно программировать повторяющиеся процессы. Синтаксис:



Перед выполнением цикла счетчик получает начальное значение. Далее тело цикла повторяется, пока значение счетчика не превысит конечного значения. Счетчик увеличивается в каждом цикле. Начальное и конечное значения и шаг могут быть как константами, так и выражениями.

Счетчик изменяется после выполнения тела цикла. Поэтому если задать конечное значение меньшее начального, то при положительном приращении цикл не будет выполнен ни разу. При одинаковых начальном и конечном значениях тело цикла будет выполнено один раз. Часть конструкции BY в скобках необязательна, она определяет шаг приращения счетчика. По умолчанию счетчик увеличивается на единицу в каждой итерации. В качестве счетчика можно использовать переменную любого целого типа.

Пример:

Varl := 0;

FOR cw := 1 TO 10 **DO**

Varl := Varl + 1;

END_FOR

Данный цикл будет выполнен 10 раз и соответственно Varl будет иметь значение 10. Шаг изменения счетчика итераций может быть и отрицательным. Начальное условие в этом случае должно быть больше конечного. Цикл будет закончен, когда значение счетчика станет меньше конечного значения.

Цикл FOR исключительно удобен для итераций с заранее известным числом повторов.

Для построения правильного цикла достаточно соблюдать два простых формальных требования:

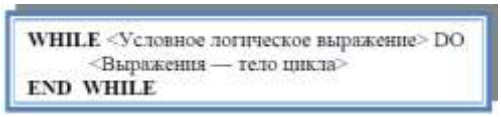
• не изменяйте счетчик цикла и условие окончания в теле цикла. Счетчик и переменные образующие конечное условие в цикле, можно использовать только для чтения;

• не задавайте в качестве конечного условия максимальное для типа переменной счетчика значение. Так, если для однобайтного целого без знака задать константу 255, то условие окончания не будет выполнено никогда. Цикл станет бесконечным.

Циклы WHILE и REPEAT

Циклы **WHILE** и **REPEAT** обеспечивают повторение группы выражений, пока верно условное логическое выражение. Если условное выражение всегда истинно, то цикл становится бесконечным.

Синтаксис **WHILE:**



Условие в цикле WHILE проверяется до начала цикла. Если логическое выражение изначально имеет значение ЛОЖЬ, тело цикла не будет выполнено ни разу.

Пример:

ci := 64;

WHILE ci > **1 DO**
Varl := Varl + **1;**
ci := ci/2;
END_WHILE

Синтаксис REPEAT:

```
REPEAT
<Выражения - тело цикла >
UNTIL <Условное логическое выражение>
END_REPEAT
```

Правильно построенный цикл WHILE или REPEAT обязательно должен изменять переменные, составляющие условие окончания в теле цикла, постепенно приближаясь к условию завершения. Если этого не сделать, цикл не закончится никогда.

2. Порядок выполнения работы

Из ПРИЛОЖЕНИЯ 2 по заданию преподавателя выбирается вариант технологической схемы.

На основании заданной технологической схемы и описания технологического процесса разработать:

1. технологические требования к схеме управления;
2. таблицу сигналов;
3. прикладную программу для ПЛК.
4. дать описание работы принципиальной схемы.

В таблицу сигналов вносятся:

1. порядковый номер переменной;
2. имя переменной (не должно содержать пробелов и кириллицы);
3. тип переменной (дискретный, аналоговый);
4. класс переменной (локальная, глобальная);
5. адрес (для внутренних переменных не заполняется).

Пример заполнения таблицы параметров.

№	Наименование параметра	Имя	Тип	Класс	Адрес
1	Температура воздуха	Temp_1	аналоговый	глобальный	%IB0
2	Кнопка «Пуск»	SB1	дискретный	локальный	-

3. Создание проекта программы.

При создании проекта используется язык структурированного текста **ST**, реализующий структуры, подобные структурам языка Pascal.

Пользователь запускает **CoDeSys** последовательным выбором приложений:

Пуск→ Все программы→ 3S Software CoDeSys V2.3 →CoDeSys V2.3.

Новый проект открывается из главного меню: **File New**. В открывшемся окне (рис. 2.1) выбирается тип контроллера, **PLC150.U-L**, выбор подтверждается нажатием клавиши **ОК**.

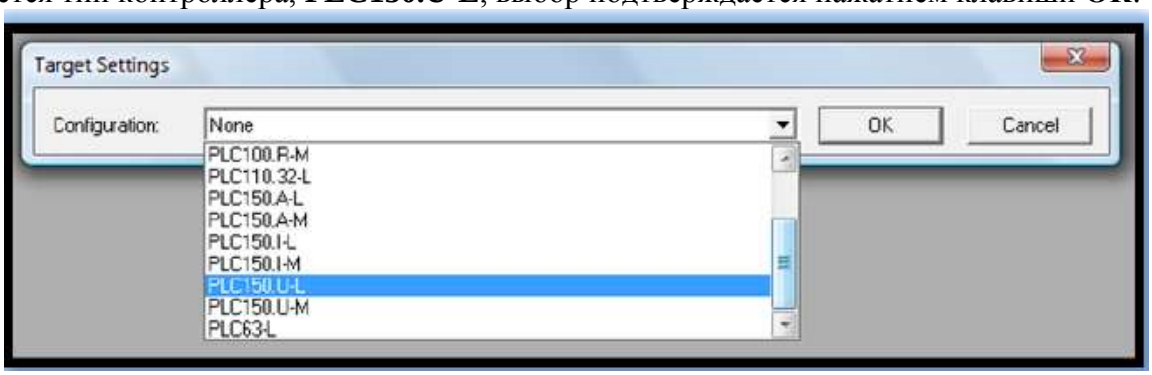


Рисунок 2.1-Окно конфигурации «Target Settings» программы

После выбора проекта выводится экранная форма, задающая тип, имя и язык программирования первичного компонента **New POU**, главной программы контроллера**.**

Необходимо выбрать язык программирования ST, установив флаги в позициях, указанных на рис.2.

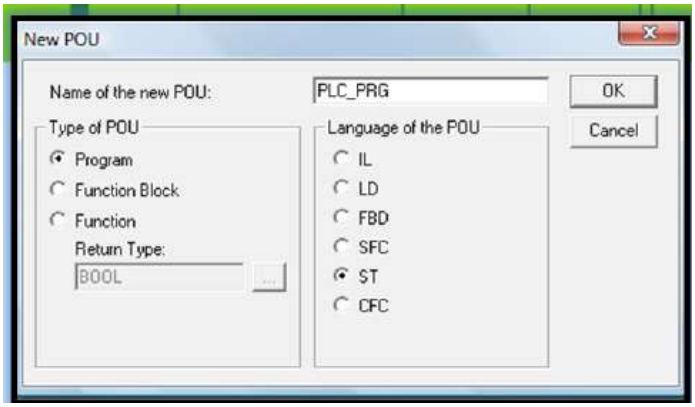


Рисунок 2.2- Вид окна «New POU» с отмеченными параметрами

Примечание. *Имя главной программы PLC_PRG и ее тип менять нельзя.*

После подтверждения выбора нажатием клавиши **ОК** откроется окно нового проекта с именем по умолчанию **Untitled**.

4. Параметры входов и выходов.

Параметры входов и выходов контроллера задаются на вкладке ресурсов **(Resources)** Организатора объектов **CoDeSys** с помощью утилиты **PLC Configuration (Конфигуратор ПЛК) (ЛР№1 п.3)**.

5. Создание программы на языке ST

При написании программы в рабочей зоне вкладки **POUs** вводится текст, состоящий из имен переменных и операторов (Таблица 1).

При необходимости используются приведенные выше конструкции IF,FOR, CASE, WHILE и REPEAT.

В тексте программы необходимо учесть технологические требования, предъявляемые к системе управления.

Форма представления результата:

6. технологические требования к схеме управления;
7. принципиальная электрическую схему автоматического управления технологической установкой;
8. прикладная программа для ПЛК.
9. описание работы принципиальной схемы
10. выводы по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторная работа 3
Программирование ПЛК на языке IL.

Цель работы: Изучить принципы составления прикладных программ для промышленных логических контроллеров на языке IL пакета CoDeSys.

Выполнив работу, Вы будете:

уметь:

- составлять функциональные и структурные схемы управления различными электроэнергетическими объектами;
- выбирать средства технической реализации микропроцессорных систем управления;

- программировать микропроцессорные системы управления на основе ПЛК широкого применения.

Материальное обеспечение:

Стенд ПЛК «ОВЕН», персональный компьютер со специализированным ПО CoDeSys

1. Общие сведения

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 4 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:

Метка:	**Оператор**	**Операнд**	**Комментарий**
МЕТКА1:	**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 «налету» выполняет синтаксический контроль и выделение цветом. Так, корректно введенные операторы выделяются голубым цветом.

Аккумулятор

Абсолютное большинство инструкции 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 диверсию 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:

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.

2. Порядок выполнения работы

Из ПРИЛОЖЕНИЯ 2 по заданию преподавателя выбирается вариант технологической схемы.

На основании заданной технологической схемы и описания технологического процесса разработать:

1. технологические требования к схеме управления;
2. таблицу сигналов;
3. прикладную программу для ПЛК.
4. дать описание работы прикладной программы.

В таблицу сигналов вносятся:

- порядковый номер переменной;
- имя переменной (не должно содержать пробелов и кириллицы);
- тип переменной (дискретный, аналоговый);
- класс переменной (локальная, глобальная);
- адрес (для внутренних переменных не заполняется).

Пример заполнения таблицы параметров.

№	Наименование параметра	Имя	Тип	Класс	Адрес
1	Температура воздуха	Temp_1	аналоговый	глобальный	%IB0
2	Кнопка «Пуск»	SB1	дискретный	локальный	-

3. Создание проекта программы.

При создании проекта используется язык структурированного текста **ST**, реализующий структуры, подобные структурам языка Pascal. Пользователь запускает **CoDeSys**

последовательным выбором приложений: **Пуск →Все программы →3S Software→ CoDeSys V2.3→ CoDeSys V2.3.**

Новый проект открывается из главного меню: **File New**. В открывшемся окне (рис. 3.1) выбирается тип контроллера, **PLC150.U-L**, выбор подтверждается нажатием клавиши **ОК**.

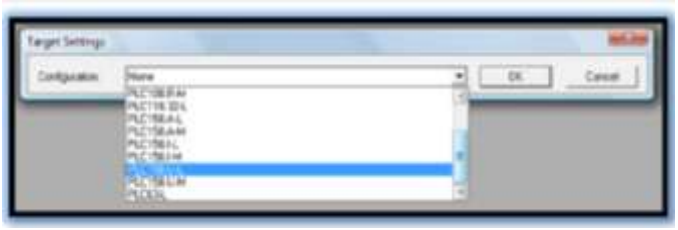


Рисунок 3.1-Окно конфигурации «Target Settings» программы

После выбора проекта выводится экранная форма, задающая тип, имя и язык программирования первичного компонента **New POU**, главной программы контроллера**.** Необходимо выбрать язык программирования IL, установив флаги в позициях, указанных на рис.3.2.

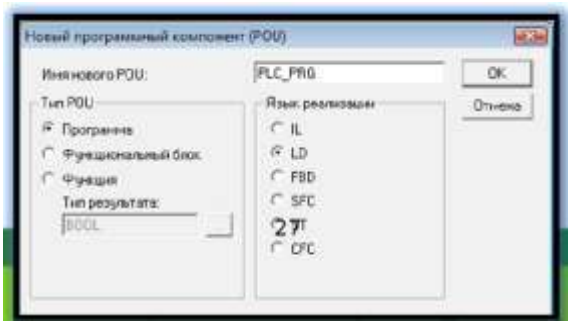


Рисунок 3.2- Вид окна «New POU» с отмеченными параметрами

Примечание. *Имя главной программы PLC_PRG и ее тип менять нельзя.*

После подтверждения выбора нажатием клавиши **ОК** откроется окно нового проекта с именем по умолчанию **Untitled**.

4. Параметры входов и выходов.

Параметры входов и выходов контроллера задаются на вкладке ресурсов **(Resources)** Организатора объектов **CoDeSys** с помощью утилиты **PLC Configuration (Конфигуратор ПЛК) (ЛР№1 п.3)**.

5. Создание программы на языке IL

При написании программы в рабочей зоне вкладки **POUs** вводится текст, состоящий из имен переменных меток, операторов, операндов, комментариев.

В тексте программы необходимо учесть технологические требования, предъявляемые к системе управления.

Форма представления результата:

1. технологические требования к схеме управления;
2. принципиальная электрическую схему автоматического управления технологической установкой;
3. прикладная программа для ПЛК.
4. описание работы принципиальной схемы
5. выводы по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторная работа 4
Программирование ПЛК на языке FBD.

Цель работы: Изучить принципы составления прикладных программ для промышленных логических контроллеров на языке FBD пакета CoDeSys.

Выполнив работу, Вы будете:

уметь:

- составлять функциональные и структурные схемы управления различными электроэнергетическими объектами;
- выбирать средства технической реализации микропроцессорных систем управления;
- программировать микропроцессорные системы управления на основе ПЛК широкого применения.

Материальное обеспечение:

Стенд ПЛК «ОВЕН», персональный компьютер со специализированным ПО CoDeSys

1. Общие сведения

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 зависят от числа входов и выходов и устанавливается графическим редактором автоматически.

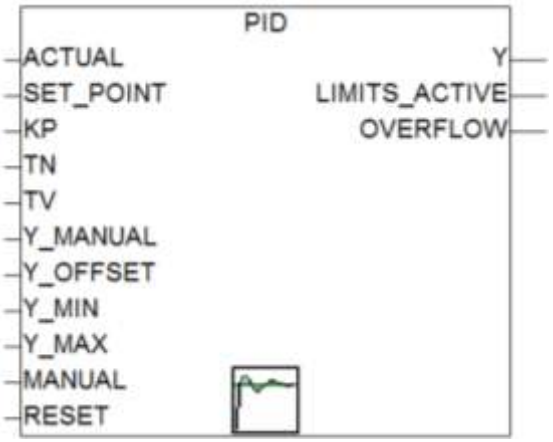


Рисунок 4.1- Пример графического представления
экземпляра функционального блока PID

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 (рис.4.2). Значения переменных, вычисленные в одной цепи, доступны в последующих цепях сразу в том же рабочем цикле.

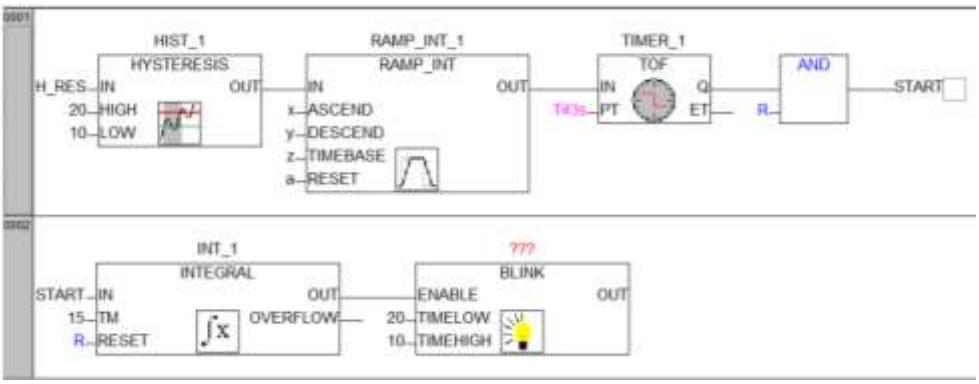


Рисунок 4.2 - Диаграмма FBD из двух цепей

Соединительные линии

Прямоугольники POU в FBD соединены линиями связи. Соединения имеют направленность слева направо. Вход блока может быть соединен с выходом блока, расположенного слева от него. Помимо этого, вход может быть соединен с переменной или константой. Соединение должно связывать переменные или входы и выходы одного типа. В отличие от компонента переменная изображается на диаграмме без прямоугольной рамки. Ширина соединительной линии в FBD роли не играет. Стандарт допускает использование соединительных линий разной ширины и стиля для соединений разного типа.

Порядок выполнения FBD

Выполнение FBD-цепей идет слева направо, сверху вниз. Блоки, расположенные левее, выполняются раньше. Блок начинает вычисляться только после вычисления значений всех его входов. Дальнейшие вычисления не будут продолжены до вычисления значений на всех выходах.

Другими словами, значения на всех выходах графического блока появляются одновременно. Вычисление цепи считается законченным только после вычисления значений на выходах всех входящих в нее элементов.

В некоторых системах программирования пользователь имеет возможность свободно передвигать блоки с сохранением связей. В этом случае ориентироваться нужно исходя из порядка соединений. Редактор FBD CoDeSys автоматически расставляет блоки в порядке выполнения.

Инверсия логических сигналов

Инверсия логического сигнала в FBD изображается в виде окружности на соединении, перед входом или переменной. Инверсия не является свойством самого блока и может быть легко добавлена или отменена непосредственно в диаграмме.

Метки, переходы и **возврат**

Порядок выполнения FBD-цепей диаграммы можно принудительно изменять, используя метки и переходы, точно так же, как и в релейных схемах. Метка ставится в начале любой цепи, являясь, по сути, названием данной цепи. Цепь может содержать только одну метку. Имена меток подчинены общим правилам наименования идентификаторов МЭК. Графический редактор автоматически нумерует цепи диаграммы. Эта нумерация применяется исключительно для документирования и не может заменять метки. Переход обязательно связан с логической переменной и выполняется, если переменная имеет значение ИСТИНА. Для создания безусловного перехода используется константа ИСТИНА, связанная с переходом.

Оператор возврата **RETURN** можно использовать в FBD так же, как и переход на метку, т. е. в связке с логической переменной. Возврат приводит к немедленному окончанию работы программного компонента и возврату на верхний уровень вложений. Для основной программы это начало рабочего цикла ПЛК.

2. Порядок выполнения работы

Из ПРИЛОЖЕНИЯ 2 по заданию преподавателя выбирается вариант технологической схемы.

На основании заданной технологической схемы и описания технологического процесса разработать:

1. технологические требования к схеме управления;
2. таблицу сигналов;
3. прикладную программу для ПЛК.
4. дать описание работы прикладной программы.

В таблицу сигналов вносятся:

- порядковый номер переменной;
- имя переменной (не должно содержать пробелов и кириллицы);
- тип переменной (дискретный, аналоговый);
- класс переменной (локальная, глобальная);
- адрес (для внутренних переменных не заполняется).

Пример заполнения таблицы параметров.

№	Наименование параметра	Имя	Тип	Класс	Адрес
1	Температура воздуха	Temp_1	аналоговый	глобальный	%IB0
2	Кнопка «Пуск»	SB1	дискретный	локальный	-

3. Создание проекта программы.

При создании проекта используется язык структурированного текста **ST**, реализующий структуры, подобные структурам языка Pascal. Пользователь запускает **CoDeSys** последовательным выбором приложений: **Пуск →Все программы →3S Software→ CoDeSys V2.3→ CoDeSys V2.3.**

Новый проект открывается из главного меню: **File New**. В открывшемся окне (рис. 4.3) выбирается тип контроллера, **PLC150.U-L**, выбор подтверждается нажатием клавиши **ОК.**

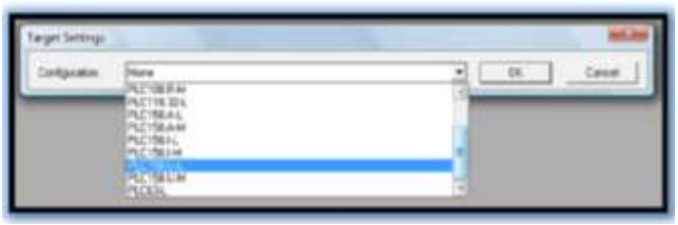


Рисунок 4.3-Окно конфигурации «Target Settings» программы

После выбора проекта выводится экранная форма, задающая тип, имя и язык программирования первичного компонента **New POU**, главной программы контроллера**.** Необходимо выбрать язык программирования FBD, установив флаги в позициях, указанных на рис.4.4.

Рисунок 4.4- Вид окна «New POU» с отмеченными параметрами

Примечание. *Имя главной программы PLC_PRG и ее тип менять нельзя.*

После подтверждения выбора нажатием клавиши **ОК** откроется окно нового проекта с именем по умолчанию **Untitled**.

4. Параметры входов и выходов.

Параметры входов и выходов контроллера задаются на вкладке ресурсов **(Resources)** Организатора объектов **CoDeSys** с помощью утилиты **PLC Configuration (Конфигуратор ПЛК) (ЛР№1 п.3)**.

5. Создание программы на языке FBD

При написании программы в рабочей зоне вкладки **POUs** вводится текст, состоящий из имен переменных меток, операторов, операндов, комментариев.

В тексте программы необходимо учесть технологические требования, предъявляемые к системе управления.

Форма представления результата:

1. технологические требования к схеме управления;
2. принципиальная электрическую схему автоматического управления технологической установкой;
3. прикладная программа для ПЛК.
4. описание работы принципиальной схемы
5. выводы по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторная работа 5
Программирование ПЛК на языке SFC.

Цель работы: Изучить принципы составления прикладных программ для промышленных логических контроллеров на языке SFC пакета CoDeSys.

Выполнив работу, Вы будете:

уметь:

- составлять функциональные и структурные схемы управления различными электроэнергетическими объектами;
- выбирать средства технической реализации микропроцессорных систем управления;
- программировать микропроцессорные системы управления на основе ПЛК широкого применения.

Материальное обеспечение:

Стенд ПЛК «ОВЕН», персональный компьютер со специализированным ПО CoDeSys

1. Общие сведения

SFC – это графический язык, который позволяет описать хронологическую последовательность различных действий в программе. Любая SFC-схема составляется из элементов, представляющих шаги и условия переходов (рис.5.1).

Для этого действия связываются с шагами (этапами), а последовательность работы определяется условиями переходов между шагами. Шаги показаны на схеме прямоугольниками. Реальная работа шага (действия) описывается в отдельном окне системы программирования и не отражается на диаграмме. О назначении шага SFC говорит только его название или, если этого не достаточно, краткое текстовое описание (комментарий).

Шаги на схеме могут быть пустыми, что не вызывает ошибки при компиляции проекта. Пустые шаги являются нормой при применении программирования сверху вниз, характерного для SFC. Определить действия, соответствующие шагу, можно в любое время. Нет ничего удивительного, если пустые шаги останутся и в законченном проекте. Задачей пустого шага является ожидание перехода.

Переходы

Ниже шага на соединительной линии присутствует горизонтальная черта, обозначающая переход. Условием перехода может служить логическая переменная, логическое выражение, константа или прямой адрес.

Переход выполняется при соблюдении двух условий:

1) переход разрешен (соответствующий ему шаг активен);

2) условие перехода имеет значение TRUE.

Простые условия отображаются непосредственно на диаграмме справа от черты, обозначающий переход. В CoDeSys на диаграмме можно записывать только выражения на языке ST. Для громоздких условий применяется другой подход. Вместо условия на диаграмме записывается только идентификатор перехода. Само же условие описывается в отдельном окне с применением языка IL, ST, LD или FBD. Переменные или прямые адреса используются в условии перехода только для чтения. В условном выражении перехода нельзя вызывать экземпляры функциональных блоков и использовать операцию присваивания. Признаком того, что идентификатор перехода на диаграмме является отдельно реализованным условием, а не простой логической переменной, служит закрашенный угол перехода. В качестве условия перехода может быть задана логическая константа. Если задано TRUE, то шаг будет выполнен однократно, за один рабочий цикл, далее управление перейдет к следующему шагу. Если задано условие FALSE, то шаг будет выполняться бесконечно.

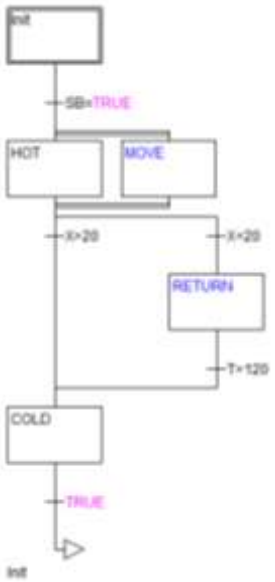


Рисунок 5.1 - Пример SFC диаграммы

Начальный шаг

Каждая SFC-схема начинается с шага, выделенного графически двойными вертикальными линиями или по всему периметру. Это - *начальный шаг*. Наименование начального шага может быть произвольным (по умолчанию Init). Начальный шаг присутствует обязательно, хотя и может быть пустым.

Параллельные ветви

Несколько ветвей SFC могут быть параллельными. Признаком параллельных ветвей на схеме является двойная горизонтальная линия. Каждая параллельная ветвь начинается и заканчивается шагом. То есть условие входа в параллельность всегда одно, условие выхода тоже одно на всех.

Параллельные ветви выполняются теоретически одновременно. В жизни это означает - в одном рабочем цикле, слева направо. Условие перехода, завершающее параллельность, проверяется только в случае, если в каждой параллельной ветви активны последние шаги.

Альтернативные ветви

Несколько ветвей SFC могут быть альтернативными ветвями. Признаком альтернативных ветвей на схеме является одинарная горизонтальная линия. Каждая альтернативная ветвь начинается и заканчивается собственным условием перехода. Проверка альтернативных условий выполняется слева направо. Если верное условие найдено, то прочие альтернативы не рассматриваются. В альтернативных ветвях всегда работает только одна из ветвей, поэтому ее окончание и будет означать переход к следующему за

альтернативной группой шагу. При создании альтернативных ветвей желательно задавать взаимоисключающие условия. В этом случае вероятность допустить ошибку при анализе или в процессе доработки диаграммы значительно ниже.

Переход на произвольный шаг

В общем случае SFC-схема выполняется сверху вниз. Стандартом допускается создание переходов на произвольный шаг. Для этого применяются соединительные линии с промежуточными стрелками или поименованные переходы. То есть переход выполняется на шаг, имя которого указано под стрелкой. В англоязычных источниках переход на произвольный шаг называется «прыжок» (jump).

Прыжок из одной ветви параллельного блока наружу вызывает эффект размножения маркера. Прыжок внутрь параллельного блока нарушает параллельность ветвей. Подобных трюков необходимо избегать.

2. Порядок выполнения работы

Из ПРИЛОЖЕНИЯ 2 по заданию преподавателя выбирается вариант технологической схемы.

На основании заданной технологической схемы и описания технологического процесса разработать:

1. технологические требования к схеме управления;
2. таблицу сигналов;
3. прикладную программу для ПЛК.
4. дать описание работы прикладной программы.

В таблицу сигналов вносятся:

- порядковый номер переменной;
- имя переменной (не должно содержать пробелов и кириллицы);
- тип переменной (дискретный, аналоговый);
- класс переменной (локальная, глобальная);
- адрес (для внутренних переменных не заполняется).

Пример заполнения таблицы параметров.

№	Наименование параметра	Имя	Тип	Класс	Адрес
1	Температура воздуха	Temp_1	аналоговый	глобальный	%IB0
2	Кнопка «Пуск»	SB1	дискретный	локальный	-

3. Создание проекта программы.

При создании проекта используется язык структурированного текста **ST**, реализующий структуры, подобные структурам языка Pascal. Пользователь запускает **CoDeSys** последовательным выбором приложений: **Пуск →Все программы →3S Software→ CoDeSys V2.3→ CoDeSys V2.3.**

Новый проект открывается из главного меню: **File New**. В открывшемся окне (рис. 5.2) выбирается тип контроллера, **PLC150.U-L**, выбор подтверждается нажатием клавиши **ОК.**

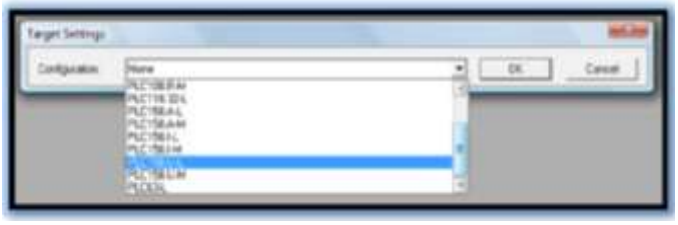


Рисунок 5.2-Окно конфигурации «Target Settings» программы

После выбора проекта выводится экранная форма, задающая тип, имя и язык программирования первичного компонента **New POU**, главной программы контроллера**.** Необходимо выбрать язык программирования FBD, установив флаги в позициях, указанных на рис.5.3.

Рисунок 5.3 Вид окна «New POU» с отмеченными параметрами

Примечание. *Имя главной программы PLC_PRG и ее тип менять нельзя.*

После подтверждения выбора нажатием клавиши **ОК** откроется окно нового проекта с именем по умолчанию **Untitled**.

4. Параметры входов и выходов.

Параметры входов и выходов контроллера задаются на вкладке ресурсов **(Resources)** Организатора объектов **CoDeSys** с помощью утилиты **PLC Configuration (Конфигуратор ПЛК) (ЛР№1 п.3)**.

5. Создание программы на языке FBD

При написании программы в рабочей зоне вкладки **POUs** вводится текст, состоящий из имен переменных меток, операторов, операндов, комментариев.

В тексте программы необходимо учесть технологические требования, предъявляемые к системе управления.

Форма представления результата:

1. технологические требования к схеме управления;
2. принципиальная электрическую схему автоматического управления технологической установкой;
3. прикладная программа для ПЛК.
4. описание работы принципиальной схемы
5. выводы по работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

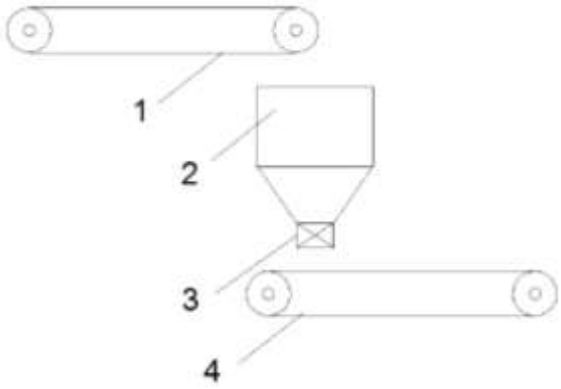
Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Приложение 1

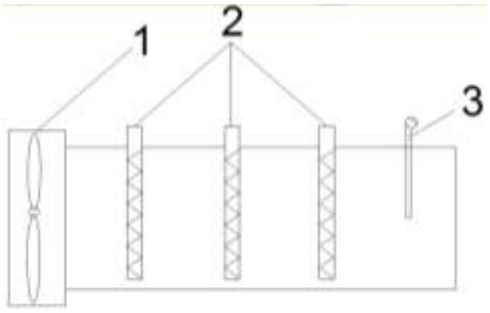
Задание 1 Линия дозирования продукта

Продукт с помощью загрузочного транспортера 1 попадает в бункер 2. Транспортер работает до тех пор, пока вес продукта в бункере не станет больше заданного. Затем транспортер 1 останавливается, срабатывает задвижка 3 и включается транспортер 4. После разгрузки бункера, задвижка закрывается, транспортер 4 останавливается и загрузка начинается вновь.



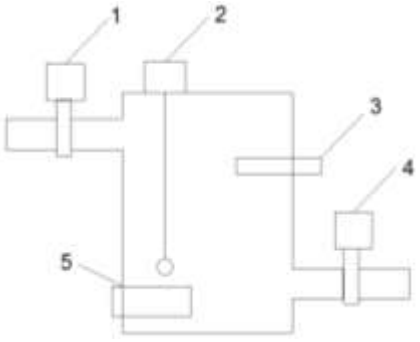
Задание 2 Тепловая пушка

Воздух вентилятором 1 прогоняется через тепловую пушку. В зависимости от уставки температуры включается определенное количество нагревательных элементов 2. Следует учесть, что нагревательные элементы не должны работать при выключенном вентиляторе. 3-измеритель температуры.



Задание 3 Водонагревательная установка

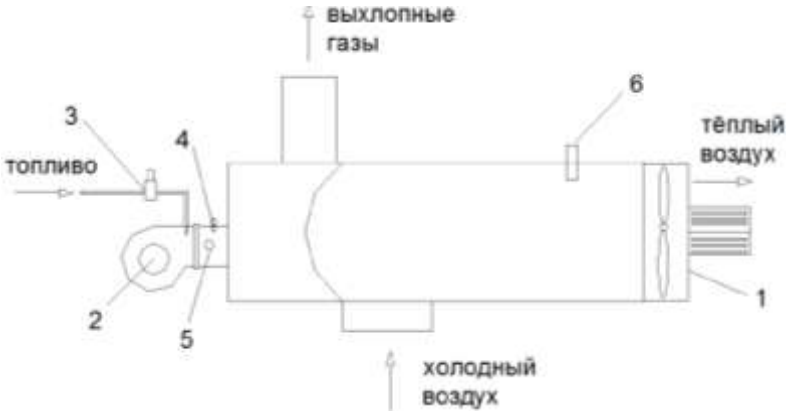
Вода через заливной клапан 1 заполняет ёмкость до определенного уровня, измеряемого датчиком уровня 2. Вода ТЭНом 5 нагревается до заданной температуры, измеряемой датчиком температуры 3, и сливается через сливной клапан 4.



Задание 4 Теплогенератор

При нажатии на кнопку пуск, звучит предупредительная сигнализация и запускается основной вентилятор теплого воздуха 1. После запуска основного вентилятора, включается топливный вентилятор 2 для продувки (10 с). Затем включается топливный соленоидный

клапан 3 и топливная смесь закачивается в камеру сгорания (5 с). Срабатывает запальная свеча 4 (4 с). Реле пламени 5 контроллирует наличие пламени. Если пламя не появилось в течение 5 с., процесс розжига выполняется еще раз (с продувки воздухом 15с.). При повторном незапуске агрегата включается продувка 1мин. и аварийная сигнализация. При нормальном запуске агрегата, система должна контролировать температуру воздуха на выходе термопреобразователем 6 и изменять скорость вращения топливного вентилятора 2. При остановке агрегата, продувка должна осуществляться до тех пор, пока температура не упадет ниже Tmin.



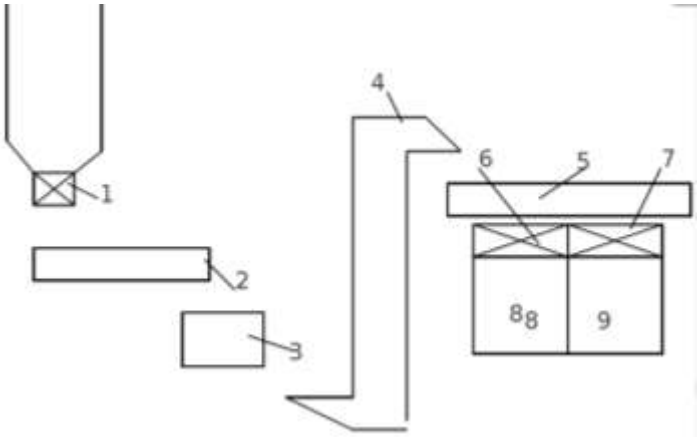
Задание 5 Зерно - овощехранилище

Зерно через задвижку 1 поступает на дробилку 2 и далее на транспортер–смеситель 5. Сюда же поступают переработанные в мойке- корнерезке 4 корнеплоды (3 транспортер нарезанных корнеплодов).

Транспортером смесителем 5 смесь загружается в смеситель 6. Предусмотреть совместную и раздельную работу линий зерна и корнеплодов.

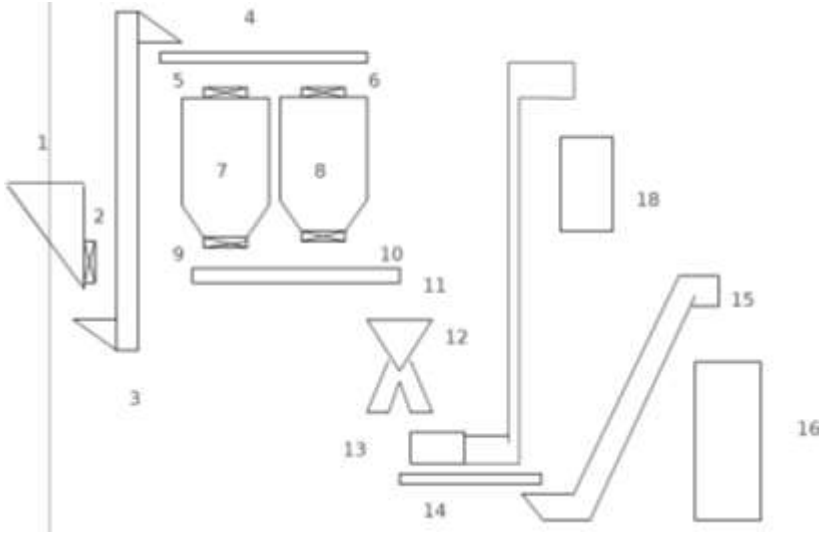
Задание 6 Зернохранилище

Зерно из бункера через задвижку 1 поступает на транспортер 2 и далее в дробилку 3. Измельченное зерно норией 4 подается на шнековый транспортер 5 и далее либо в бункер 8 либо в бункер 9. Линия должна отключиться при заполнении одного из бункеров. Режим работы электродвигателей поточной линии кратковременный.



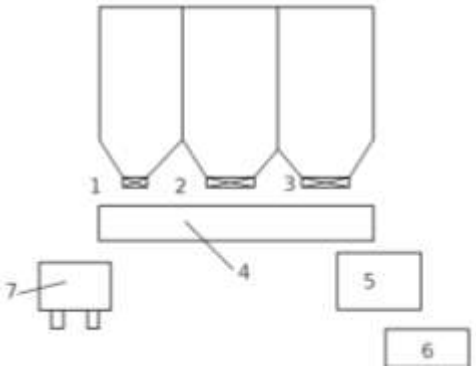
Задание 7 Отжим масла

Из завальной ямы 1 семечки через задвижку 2 норией подаются на шнековый транспортер и затем через задвижку 5 и 6 заполняют бункера 7 и 8. Из бункеров 7 и 8 через задвижки 9 и 10 семечки поступают на наклонный транспортер 11, который заполняет жим 12. После жима масло из накопительной емкости насосом 13 подается в емкость 18. Жмых после отжима поступает на транспортер 14 и далее норией 15 загружается в накопительный бункер 16. Режим работы двигателей кратковременный.



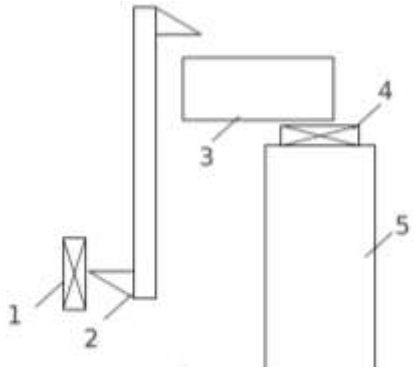
Задание 8 Элеватор

Зерно поступает на транспортер 4 через одну из задвижек 1,2 или 3 или все вместе (выбор задвижки производится оператором) и далее либо в тележку 7 либо на дробилку 5 и далее в бункер 6. Схема должна отключаться при срабатывании датчика уровня в бункере 6 или при срабатывании датчика давления под тележкой.



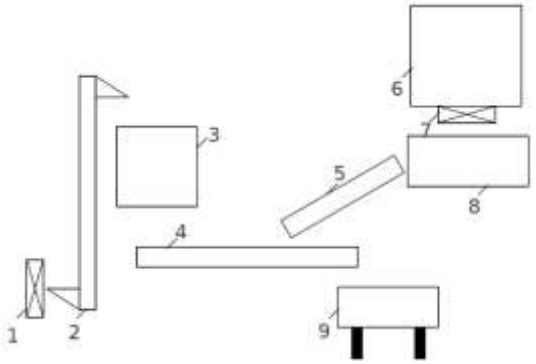
Задание 9 Мельница

Зерно из завальной ямы через заслонку 1 норией 2 подается на дробилку 3, где оно измельчается. Измельченное зерно через заслонку 4 загружается в бункер 5. Предусмотреть отключение схемы в рабочем порядке и при срабатывании датчиков уровня. Двигатели технологической схемы работают в кратковременном режиме.



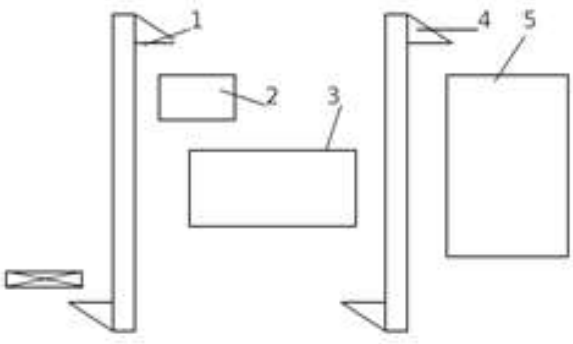
Задание 10 Переработка зерна и корнеплодов

Технологическая линия состоит из линии переработки зерна и линии переработки корнеплодов. В состав линии переработки зерна входят задвижка 1 в завальной яме, нория 2, дробилка 3. Линия переработки корнеплодов содержит бункер нерезаных корнеплодов 6, задвижку бункера7, мойку корнерезку 8, транспортер измельченных корнеплодов 5. Продукты с обеих линий поступают на транспортер смеситель 4 и далее загружаются в тележку 9. Предусмотреть раздельную и совместную работу линий переработки зерна и корнеплодов.



Задание 11 Перевалка зерна

Зерно из завальной ямы норией 1 подается на триерный блок 3. Очищенное зерно норией 4 загружается в бункер 5. Предусмотреть работу линии с очисткой зерна и без очистки.

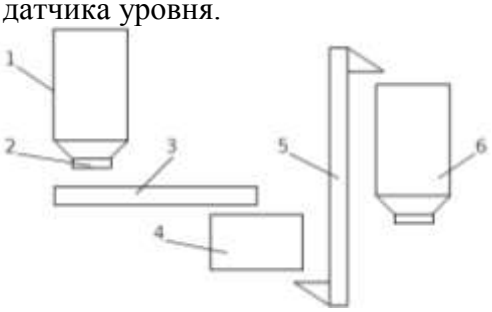


Задание 12 Подача зерна

Зерно через заслонку 1 норией 2 подается на метательный транспортер3.

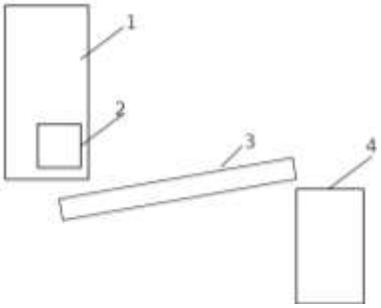
Задание 13 Подача зерна на мельницу

Зерно из бункера 1 через заслонку 2 шнековым транспортером 3 подается на мельницу 4. Продукт помола норией 5 подается в бункер 6. Предусмотреть отключение линии при заполнении бункера по сигналу датчика уровня.



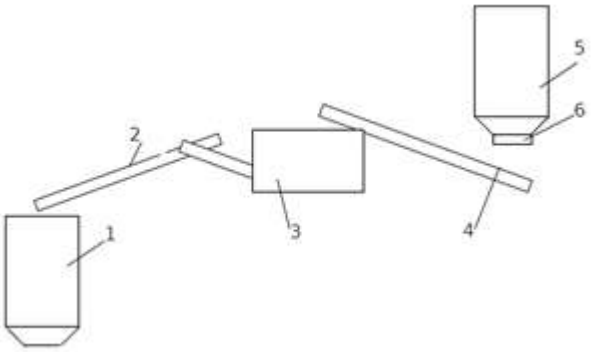
Задание 14 Кормоподача

Продукт из бункера 1 шнековым дозатором корма 3 подается в бункер дозатор кормораздатчика 4. Предусмотреть отключение линии при срабатывании датчика уровня в бункере дозаторе 4. Для исключения образования сводов при хранении корма предусматривается вибратор 2.



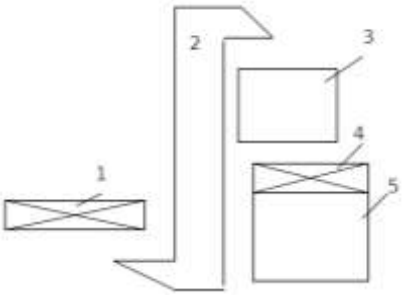
Задание 15 Овощехранилище

Корнеплоды из бункера 5 через электромагнитную заслонку 6 поступают на скребковый транспортер 4, который производит загрузку корнеклубнемойки 3. Измельченные корнеплоды шнековым транспортером 2 загружаются в смеситель 1.



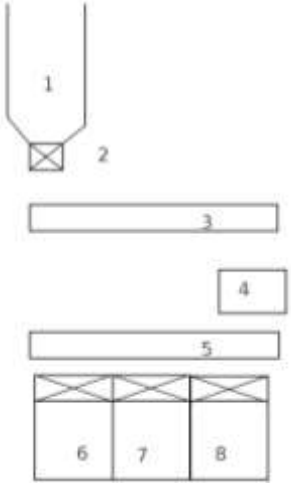
Задание 16 Дробилка

При открытии заслонки 1 продукт норией 2 подается в дробилку 3. Измельченный продукт из дробилки через заслонку 4 заполняет бункер 5. Предусмотреть отключение линии при заполнении бункера по сигналу датчика уровня



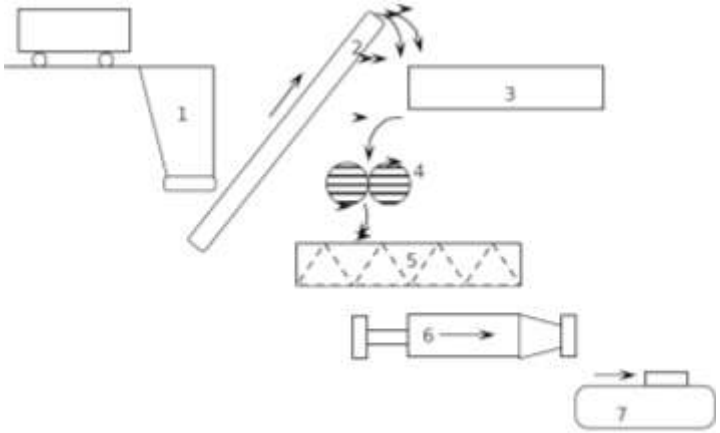
Задание 17 Зернодробилка

Зерно из бункера 1 через заслонку 2 шнековым транспортером 3 подается на дробилку 4. Измельченный продукт транспортером 5 через электромагнитные заслонки 6, 7, 8 загружается в один из бункеров. Выбор бункера осуществляется оператором.



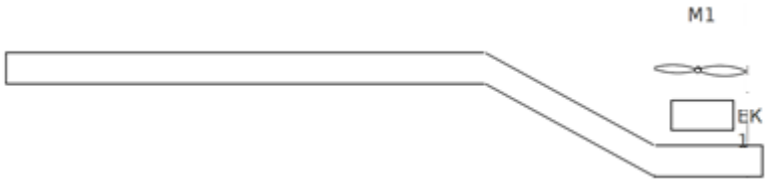
Задание 18 Подготовка глины

Глина из завальной ямы 1 транспортером подается на камневыделительные вальцы 3. Далее глина будет проходить через гладкие вальцы 4 и поступать в глиномешалку 5. Глина прессом 6 выдавливается и поступает на резательный механизм 7.



Задание 19 Отопительно-вентиляционная установка

Подача воздуха в отопительно-вентиляционную систему осуществляется вентилятором. В холодное время года воздух подогревается калорифером. Теплый воздух в помещение попадает через систему воздуховодов. В том случае, если температура воздуха в помещении понижается двигатель вентилятора переходит на пониженную частоту вращения.



Задание 20 Раздача корма

Продукт на платформенный раздатчик корма 3 подается загрузочным транспортером 2 и шнековым дозатором корма из бункера 1. Платформенный раздатчик начинает движение после того, как на него падает первая порция корма. При этом транспортер 3 движется вправо. При наезде на конечный выключатель SQ1 корм сбрасывается в кормушки и транспортер останавливается. Обратное движение платформенного раздатчика начинается через одну-две секунды, при этом происходит заполнение второй половины платформенного раздатчика. Через выдержку времени должно произойти отключение шнекового дозатора корма, а остатков корма на загрузочном транспортере 2 должно хватить для заполнения оставшейся части фронта кормления. При наезде на конечный выключатель SQ2 происходит сбрасывание корма во вторую половину кормушек и отключение всей схемы. Сброс корма в кормушки производится плужковыми сбрасывателями.