СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

В настоящем сборнике представлен ряд лабораторных работ, связанных с изучением реальных систем и моделированием элементов гибких производственных систем, а именно, роботов и станков с системами числового программного управления. Рассматривается устройство, моделирование, принципы ручного управления, работа в автоматическом режиме и технологическое программирование роботов и ГПС на примере учебных стендов, в состав которых входят роботы и станки, и трехмерных виртуальных моделей. Изучение динамики сложных программных систем управления производится с использованием эмулятора сетей Петри.

Знания, полученные при работе с моделями роботов, дадут студентам представления о подходах к моделированию робототехнических систем и проектированию трехмерных моделей технологических объектов в конструкторском пакете SolidWorks.

В лабораторной работе № 1 изучается устройство и технологическое программирование сборочного робота ЦПР-1П.

Работы № 2–4 связаны с изучением конструктивного устройства макета учебного ГПМ, состоящего из робота и двух токарных станков. На макете студенты могут разрабатывать, отлаживать технологические программы и изготавливать реальные детали.

Работы № 5 и 6 проводятся на роботе, снабженном системой технического зрения. Робот может собирать заданные конструкции из набора деталей допустимого класса.

Работа № 7 позволяет исследовать на компьютерной модели некоторые алгоритмы формирования оптимальной (по заданному критерию) траектории движения робота (раздел подготовлен магистром ФТФ ТГУ К.А. Мусатенко).

В работах № 8–10 студенты знакомятся с процессом моделирования роботов в конструкторском пакете SolidWorks, позволяющем строить трехмерные модели, программировать их анимацию и рассчитывать некоторые технические и механические параметры конструкции.

В работе № 11 рассмотрен один из подходов исследования динамических систем управления на примере построения модели простой (используемой для управления фрезерным станком) операционной системы реального времени (раздел подготовлен магистром ФТФ ТГУ В.И. Третьяковой). В современных системах управления важную роль играет интерфейсная часть, связывающая вычислительную машину и

объект управления. Эти вопросы, частично, рассматриваются в лабораторных работах № 12 и 13.

Для выполнения работы № 1 используется специально разработанная программная модель технологического робота ЦПР-1П. (пакет «**CPR3D»,** загрузочный модуль **cpr.exe**).

Работы № 2–4 выполняются на учебном стенде **Гибкий производственный модуль.**

Работы № 5–6 выполняются на учебном стенде **Технологический робот с техническим зрением.**

Для выполнения работы № 7 используется специально разработанный пакет «**Ксюша»**, загрузочный модуль **course.exe**.

Работы № 8–10 выполняются в конструкторском пакете **SolidWorks**.

Работа № 11 выполняется в **эмуляторе сетей Петри**.

Для выполнения работ № 12–13 используется учебный стенд, на базе микроконтроллера **ATMega8.**

1. ИЗУЧЕНИЕ УСТРОЙСТВА И СИСТЕМЫ ПРОГРАММИРОВАНИЯ СБОРОЧНОГО РОБОТА ЦПР-1П

Лабораторная работа № 1 (4 часа)

1.1. Цели работы

Целями лабораторной работы являются изучение устройства и принципа работы промышленного робота (ПР) ЦПР-1П и микроконтроллера МКП-1, освоение методики программирования движений ПР.

1.2. Задания к лабораторной работе

1. Изучить основные характеристики и принципы работы робота ЦПР-1П.
2. Исследовать возможности компьютерной модели робота при работе в ручном режиме.
3. Создать управляющую программу в соответствии с заданием преподавателя.
4. Ответить на контрольные вопросы.
5. Написать отчет по проделанной работе.

1.3. Краткая характеристика промышленного робота ЦПР-1П

Робот промышленный ЦПР-1П предназначен для автоматизации процессов подачи или удаления деталей в сборочном, штамповочном производстве и автоматизации операций загрузки-разгрузки технологического оборудования различного назначения. Робот имеет микропроцессорную систему управления с объемом памяти, допускающим расширение, что позволяет его использовать для автоматизации технологических операций различной сложности. Исполнение робота позволяет достаточно просто объединять его с технологическим оборудованием в робототехнические комплексы [1].

1.3.1. Состав промышленного робота и его технические характеристики

Промышленный робот ЦПР-1П состоит из манипулятора и микропроцессорного устройства управления МКП-1 (рис. 1.1). В манипуляторе для создания движения по степеням подвижности и в схвате используются пневмопривода. Основные технические характеристики робота приведены в табл. 1.1.

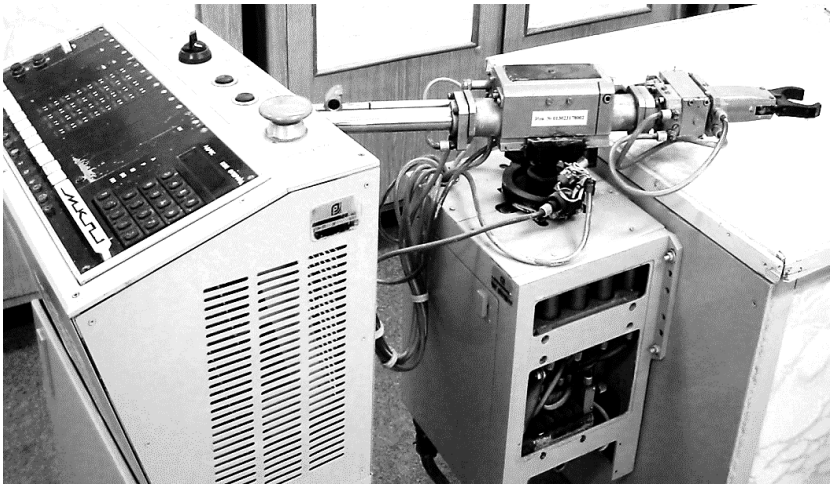


Рис. 1.1. Робот-манипулятор

Таблица 1.1

Технические характеристики робота

Число степеней подвижности	4
Номинальная грузоподъемность, кг	1
Максимальная абсолютная погрешность позиционирования ПР, мм	±0,1
Давление воздуха в сети питания, Мпа	0,4
Максимальный расход воздуха, $м^3$/ч, не более	6
Система управления	Цикловая
Управление подъемом, поворотом и выдвижением руки манипулятора осуществляется по позиционному принципу	
Управление сгибом кисти, захватом – по временному принципу	
Число одновременно управляемых движений по степеням подвижности	1
Число каналов связи с внешним оборудованием	

Окончание табл. 1.1

Входов	8
Выходов	7
Ввод и отладка программы	С клавиатуры МКП стойки управления
Объем памяти для хранения рабочих программ	512 байт
Формат одной команды	2 байта
Параметры горизонтального перемещения	
Максимальное линейное перемещение, мм	200
Время перемещения, с, не менее	0,4
Максимальная скорость, м/с, не более	0,9
Максимальное ускорение, м/с^2, не более	35
Максимальная абсолютная погрешность, мм	±0,05
Параметры вертикального перемещения	
Максимальное линейное перемещение, мм	100
Время перемещения, с, не менее	0,5
Максимальная скорость, м/с, не более	0,3
Максимальное ускорение, м/с^2, не более	20
Максимальная абсолютная погрешность, мм	±0,05
Поворот относительно вертикальной оси	
Максимальное угловое перемещение, град	240
Время перемещения, с, не менее	1,2
Максимальная скорость, град/с, не более	300
Максимальное ускорение, град/с^2, не более	$2{,}8 \cdot 10^3$
Параметры механизма сгиба	
Максимальное угловое перемещение, град	90±2
Время перемещения, с, не менее	0,4
Максимальная скорость, град/с, не более	300
Максимальное ускорение, град/с^2, не более	$4 \cdot 10^3$
Максимальная абсолютная погрешность позиционирования, мм	±0,08
Показатели захватного устройства	
Усилие захватывания, Н, не менее	90
Время захватывания, с, не более	0,3
Время отпускания, с, не более	0,3

1.3.2. Устройство и принцип работы манипулятора

Общий вид манипулятора представлен на рис. 1.1. На рис. 1.2 представлен упрощенный вид сборочного чертежа. Описание конструкторского исполнения манипулятора дано в первой части лабораторного практикума [1, с. 61].

Схема пневматическая принципиальная манипулятора представлена на рис. 1.3. Сжатый воздух из пневмосети поступает в блок подготовки воздуха в стойке управления. Из блока подготовки отфильтрованный,

ненасыщенный маслом воздух поступает в блок электропневматических распределителей РЭ. Воздух в распределители подается под давлением (0,5±0,02) МПа, которое поддерживается регулятором давления К и контролируется реле давления РД. При пониженном давлении в сети концевой выключатель в РД отключает питание стойки управления, и работа робота прекращается. Выходы распределителей соединены с соответствующими полостями цилиндров исполнительных механизмов.

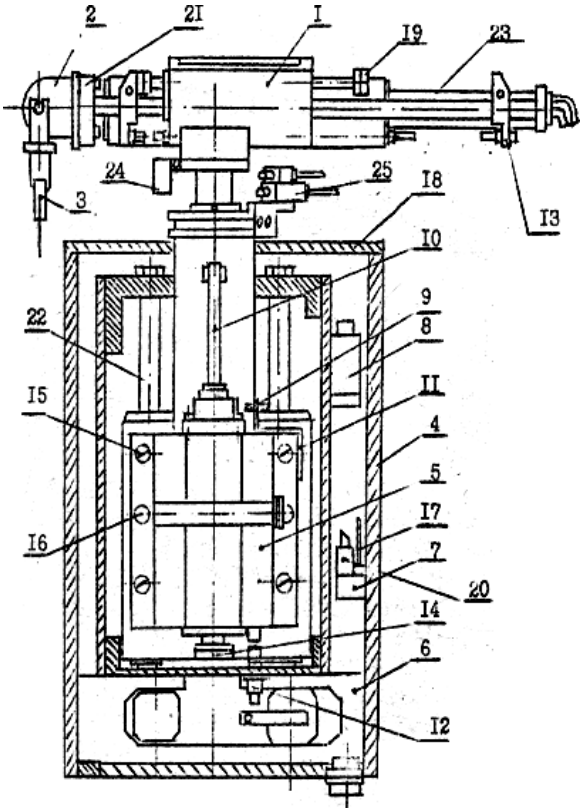


Рис. 1.2. Манипулятор: 1 – механизм горизонтальных перемещений; 2 – механизм сгиба; 3 – захват; 4 – корпус; 5 – механизм подъёма; 6 – механизм поворота; 7 – блок дросселей с эжектором; 8 – блок электропневмораспределителей; 9 – датчик; 10 – упор; 11 – кронштейн; 12 – датчик; 13 – упор; 4 – гайка; 12 – болт; 16 – штифт; 17 – эл. плата; 18 – крышка; 19 – демпфер; 20 – пневматическая плата; 21 – фланец; 22 – колонна; 23 – направляющая; 24 – упор; 25 – датчик

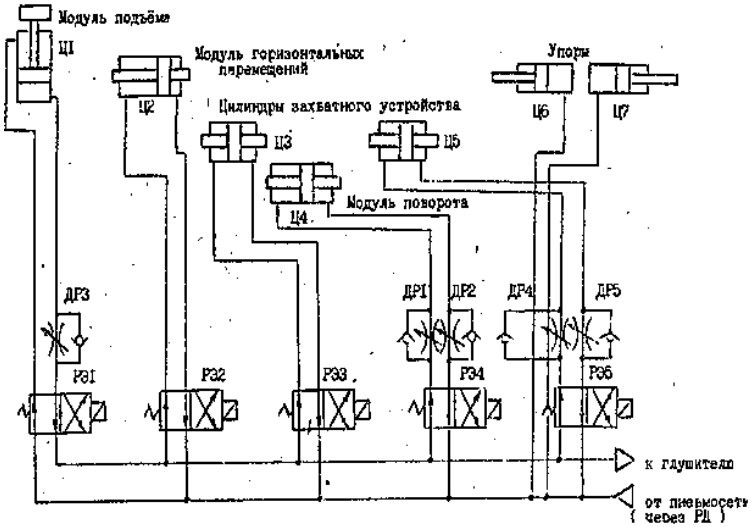


Рис. 1.3. Принципиальная пневматическая схема робота ЦПР-1П

1.4. Система программирования робота

Устройство управления ПР выполнено в виде самостоятельной стойки напольного типа, в которой размещены панель управления, электронная плата, блок подготовки воздуха, микроконтроллер (МК), лицевая панель которого совмещена с панелью стойки управления.

1.4.1. Назначение микроконтроллера и режимы работы

Микроконтроллер МКП-1 предназначен для циклового двухпозиционного управления манипуляторами и технологическим оборудованием. Областью наиболее эффективного применения МК является управление робототехническими комплексами и автоматическими линиями при автоматизации технологических процессов в условиях серийного и мелкосерийного производства.

Пульт управления (ПУ) представляет собой средство общения оператора с МК (рис. 1.1) и включает клавиатуру для ввода кодов команд и управления режимами работы МК, однострочный дисплей, предназначенный для отображения контролируемой оператором информации, и индикаторы режимов работы (рис. 1.4).

В каждый текущий момент времени МК может находиться в одном из пяти режимов работы: автоматическом (А), ручном (Р), пошаговом (Ш), ввода программы (ВП), просмотра программы (ПП). Для отображения режима используются светодиодные индикаторы А, Р, Ш, ВП, ПП пульта управления.

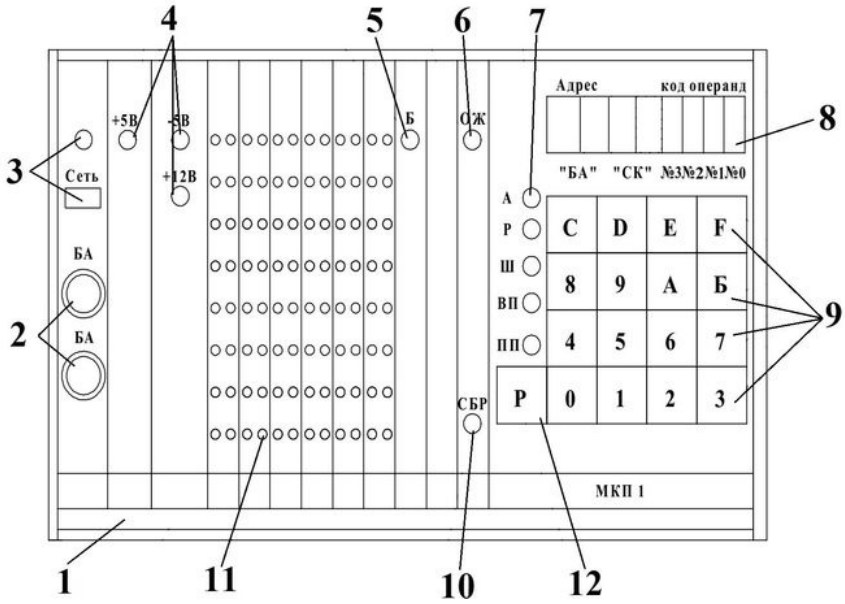


Рис 1.4. Схема панели пульта управления микроконтроллера МКП:
1 – корпус; 2 – предохранители, включенные в цепь первичного направления;
3 – выключатель «Сеть» и индикатор напряжения питающей сети;
4 – индикаторы наличия направлений вторичных стабилизированных источников питания +5В, +12В, – 5В; 5 – индикатор энергонезависимого напряжения Б для модуля энергонезависимого запоминающего устройства; 6 – индикатор ожидания ОЖ;
7 – индикаторы (А, Р, Ш, ВП, ПП) режимов работы микроконтроллера; 8 – однострочный дисплей; 9 – информационные клавиши; 10 – кнопка «СБР» переключения микроконтроллера на ручной режим работы; 11 – индикаторы состояния входов и выходов микроконтроллера; 12 – переключатель режима работы микроконтроллера; БА – служебный регистр; СК – счётчик команд микроконтроллера; № 3, № 2 – числовые значения разрядов кода операции; № 1, № 0 – числовые значения разрядов кода операнда

Автоматический режим работы является основным и предназначен для управления технологическим оборудованием в соответствии с алгоритмом, реализованным в виде управляющей программы, хранимой в запоминающем устройстве.

Режим управления *Ручной* обеспечивает возможность выполнения команды сразу после ввода ее с клавиатуры ПУ без запоминания кода в ОЗУ МК, что позволяет реализовать оперативную отладку и настройку управляемого оборудования.

В *Пошаговом* режиме оператор имеет возможность осуществить выполнение управляющей программы по отдельным шагам. В паузах между выполнением команд процессор выводит на индикацию адрес (содержимое счетчика команд) и содержимое области памяти запоминающего устройства, хранящей команду, которая будет выполняться на следующем шаге.

Режим *Ввод программы* используется для записи кодов команд управляющей программы в запоминающее устройство.

В режиме *Просмотра программы* на дисплее в зоне адреса отображаются значения счетчика команд, в зонах кода операции и операнда – код команды. При нажатии на любую информационную клавишу значение адреса увеличится (уменьшится) на единицу, и на дисплей выведется адрес и код следующей программы.

1.4.2. Система команд

МК оснащен системой команд, предназначенной для решения задач *циклового и программно-логического управления* дискретными производственными процессами. Исходящая информация для составления программ может быть представлена циклограммой работы оборудования, блок-схемой алгоритма управления или в виде булевых функций.

Команды МК можно разделить по функциональному назначению на следующие группы: команды ввода-вывода; команды управления программой; команды управления счетчиками; команды контроля и редактирования программ; команды тестового контроля функциональных блоков.

Совокупность команд МК, образующая управляющую программу, записывается и хранится в модуле памяти. Необходимо помнить, что адресация команд начинается не с единицы, а с нуля.

1.5. Описание компьютерной модели робота

Виртуальная модель робота визуально представляет достаточно адекватную модель с точки зрения процесса обучения. Работа в ручном режиме, система программирования и реализация выполнения технологической программы, в основном, соответствует реальному роботу, что позволяет использовать модель для обучения или для набора и отладки программ для реального робота.

В программном плане модель представляет собой скомпилированный exe-файл и папку с данными «Data». Таким образом, для открытия компьютерной модели необходимо выбрать загрузочный модуль «cpr.exe» и двумя кликами мыши запустить его на выполнение. При запуске программы появляется диалоговое окно, в котором можно выбрать параметры отображения модели (рис. 1.5).

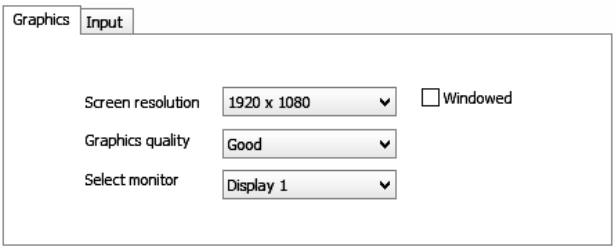


Рис. 1.5. Окно предварительной настройки компьютерной модели робота

В этом окне выпадающая строка Screen resolution позволяет выбрать разрешение экрана монитора, на котором будет отображаться симулятор, и под которые был адаптирован интерфейс управления виртуальной моделью. Заметим, что под разрешением экрана понимается величина, определяющая количество точек (элементов растрового изображения) на единицу площади (или единицу длины). Возможность выбора между оконным и полноэкранным режимами регулируется меткой Windowed. Строка *Graphics quality* позволяет выбрать детализацию прорисовки трехмерных объектов, которая определяет степень нагрузки на графический процессор компьютера. Следовательно, на слабопроизводительных компьютерах рекомендуется выставлять минимальный режим прорисовки.

После выбора необходимых настроек запустится меню работы с программой (рис. 1.6). На рис. 1.7 показан вид трехмерной модели манипулятора ЦПР-1П.

Рис. 1.6. Главное меню

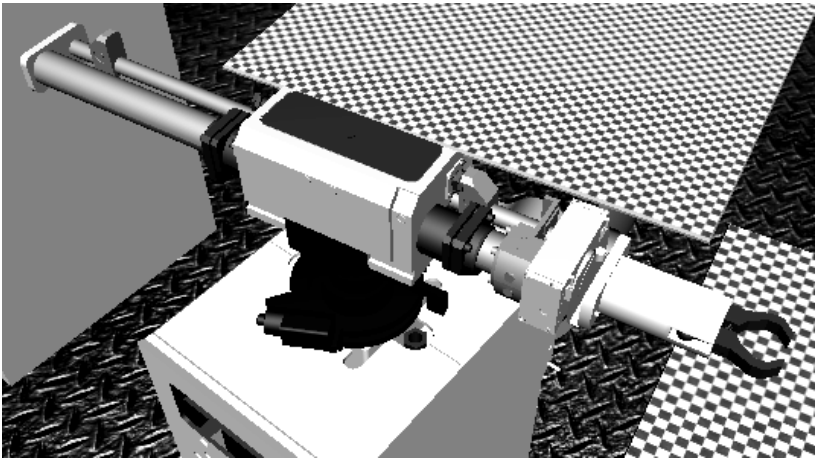


Рис. 1.7. Робот-манипулятор на сцене рабочей области

Для начала обучения необходимо выбрать вкладку меню *Обучение*. После этого осуществится переход в рабочую область, которая изображена на рис. 1.8.

Рабочее пространство модели робота – фиксировано концевыми выключателями и не может изменяться. Модель робота манипулятора поддерживает ручной и автоматический режимы.

Система команд, поддерживаемых моделью робота, изображена в табл. 1.2.

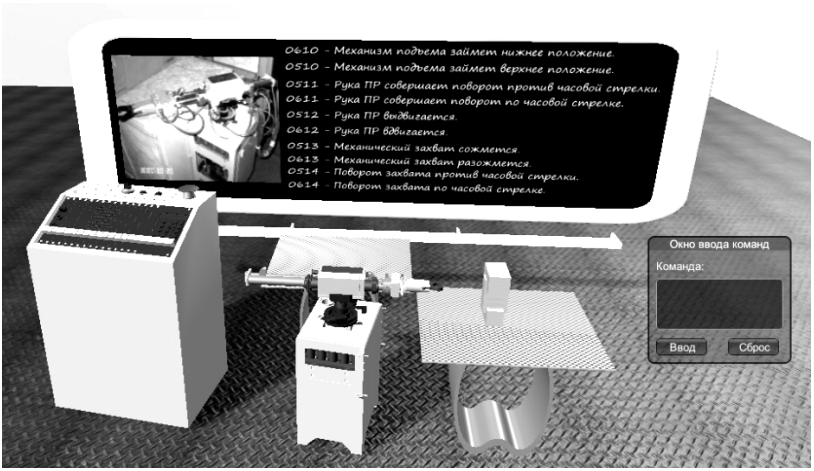


Рис. 1.8. Рабочая область

Таблица 1.2

Список команд, поддерживаемых роботом

0610	Механизм подъема займет нижнее положение
0510	Механизм подъема займет верхнее положение
0511	Рука ПР совершает поворот против часовой стрелки
0611	Рука ПР совершает поворот по часовой стрелке
0512	Рука ПР выдвигается
0612	Рука ПР вдвигается
0513	Механический захват сожмется
0613	Механический захват разожмется
0514	Поворот захвата против часовой стрелки
0614	Поворот захвата по часовой стрелке

Команды следует вводить в *Окно ввода команд*. Команды вводятся построчно или через пробел. После нажатия кнопки «Ввод» начнётся выполнение введённых команд манипулятором. Кнопка *Сброс* перезапустит сцену с начальными параметрами. Само окно ввода команд, для удобства, перетаскиваемое и изображено на рис. 1.9.

Так же на сцене существует деталь (рис. 1.10), которую манипулятор может перемещать, к примеру, с одного стола на другой.

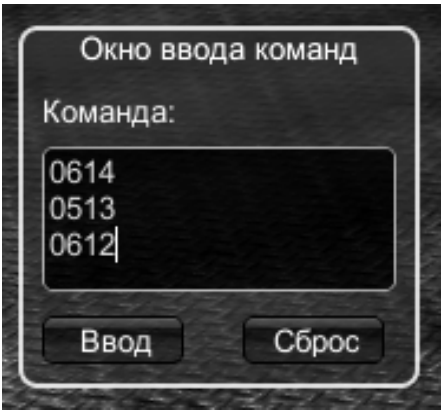


Рис. 1.9. Окно для ввода команд

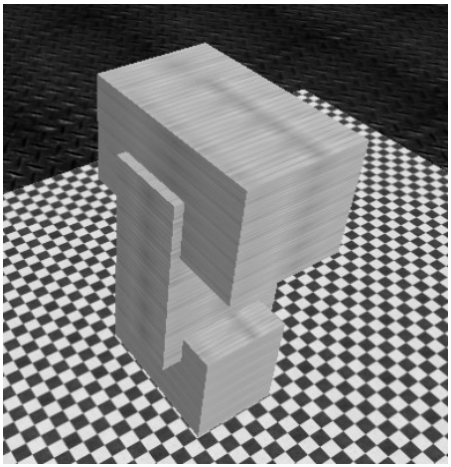


Рис. 1.10. Деталь на сцене рабочей области

Режим управления *Ручной* обеспечивает возможность выполнения команды сразу после ввода ее с клавиатуры пульта управления, что позволяет реализовать оперативную отладку программы и настройку управляемого оборудования. Для включения ручного режима не нужно предпринимать специальных действий. Достаточно после ввода кода одной команды в *Окно ввода команд* запустить программу на выполнение.

Автоматический режим работы является основным и предназначен для управления технологическим роботом в соответствии с алгоритмом, реализованным в виде управляющей программы. Связь пульта управления с процессором в этом режиме ограничена возможностью остановки программы. Клавиатура для ввода кодов команд заблокирована. Для выполнения программы в *Автоматическом* режиме необходимо ввести коды всех команд и после этого запустить программу на исполнение.

На сцене присутствует доска с командами и их описанием (рис. 1.11). На доске команд присутствует рамка с фотографией манипулятора, при клике на которую осуществляется переход на сцену *Фото*.

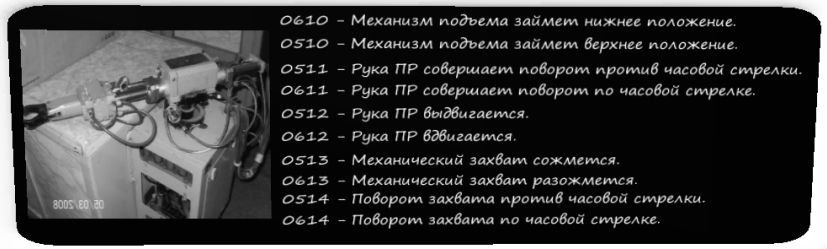


Рис. 1.11. Доска с описанием команд

При клике на *Доску команд* появится перетаскиваемое окно, дублирующее описание команд. Кнопка *Сброс* закроет это окно (рис. 1.12).

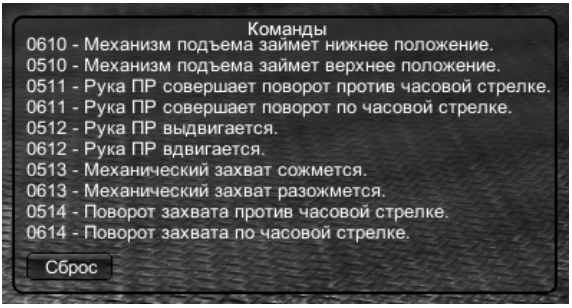


Рис. 1.12. Дополнительное окно с описанием команд

Управление камерой сцены, позволяющей рассмотреть манипулятор и выполняемые им действия с различных ракурсов, осуществляется клавишами w, a, s, d клавиатуры (рис. 1.13).

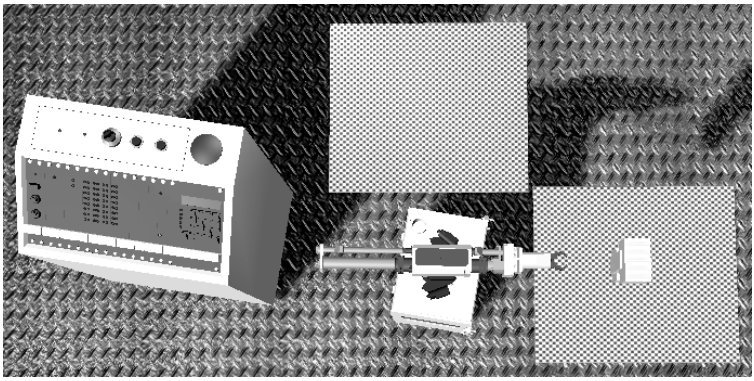


Рис. 1.13. Точка обзора изменена

Возврат в основное меню осуществляется с помощью клавиши Esc клавиатуры.

При выборе вкладки *Манипулятор* (рис. 1.6) осуществится переход на сцену с краткой информацией о роботе ЦПР-1П (рис. 1.14).

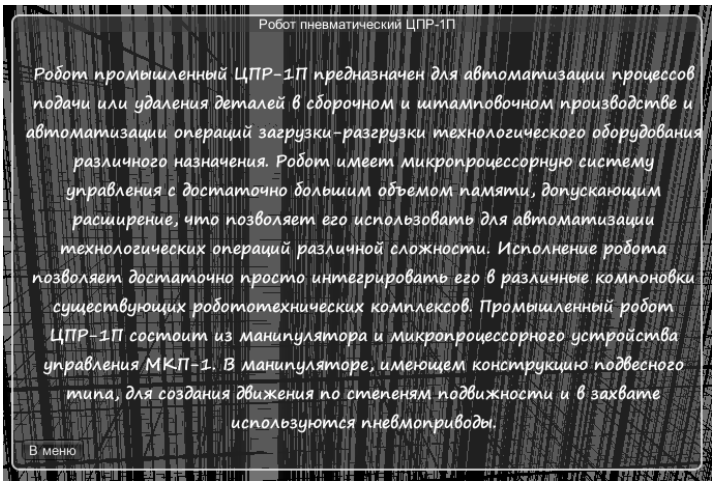


Рис. 1.14. Сцена с кратким описанием робота-манипулятора

Вернуться в основное меню поможет кнопка *Назад* или клавиша Esc клавиатуры.

При выборе вкладки «Фото» откроется сцена, позволяющая ознакомиться с реальным внешним видом манипулятора ЦПР-1П и просмотреть его фотографии в различных ракурсах (рис. 1.15).

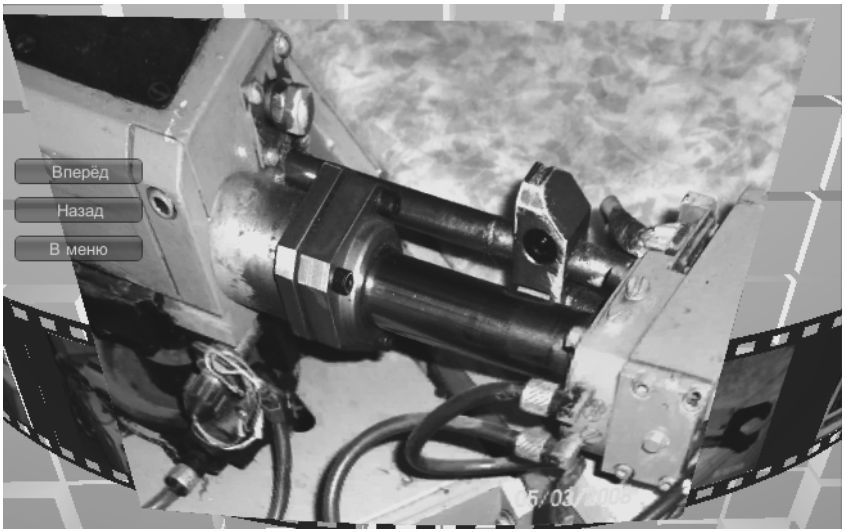


Рис. 1.15. Сцена с фотографиями манипулятора

Вкладки *О программе* и *Выход* предназначены для ознакомления с информацией о программе и ее авторами, а также для завершения работы с программой соответственно.

1.6. Контрольные вопросы

1. Что такое манипулятор, автооператор и промышленный робот?
2. Назовите, какие движения осуществляются манипулятором.
3. Что понимается под цикловым, позиционным и контурным управлением промышленного робота?
4. Чем характерна система ***программного управления***?
5. Чем характерна система ***адаптивного управления***?
6. Чем характерна система ***оптимального управления***?
7. К какой системе управления относится система данного робота?
8. Какие виды управления совмещает в себе промышленный робот ЦПР-1П?

9. Назовите основные достоинства и недостатки пневматических роботов.

10. В какой системе координат работает манипулятор робота ЦПР-1П?

11. Как задается рабочая зона данного робота?

12. Опишите формат кадра технологической программы?

13. Опишите процесс набора технологической программы.

14. Опишите процесс работы робота в ручном режиме.

15. Опишите процесс работы робота в программном режиме.

16. Критически проанализируйте процесс взаимодействия пользователя и виртуальной модели робота.

17. Назовите основные достоинства и недостатки компьютерной модели робота.

2. ИЗУЧЕНИЕ ЛАБОРАТОРНОГО СТЕНДА ГИБКОГО ПРОИЗВОДСТВЕННОГО МОДУЛЯ

Лабораторная работа № 2 (4 часа)

2.1. Цели работы

1. Изучить структуру и конструктивное исполнение гибкого производственного модуля (**ГПМ**) [2, 3].
2. Изучить принципы построения интерфейса системы числового программного управления.
3. Научиться работать с ГПМ в виртуальном режиме.
4. Изучить особенности работы с токарным станком.
5. Научиться запускать стенд ГПМ в работу.

2.2. Задания к лабораторной работе

1. Изучить требования по технике безопасности и расписаться в соответственном журнале.
2. Изучить структуру оборудования, оценить число степеней свободы всех технологических элементов ГПМ.
3. Проанализировать особенности исполнительных механизмов (тип датчиков и двигателей, принципы управления ими, точностные характеристики).
4. Научиться включать лабораторный стенд.
5. Набрать и загрузить учебную управляющую программу для робота.
6. В режиме имитатора (виртуальной модели робота) запустить ее на выполнение. При этом убедиться, что робот выполняет все запрограммированные действия и не попадает в запрещенные области.
7. Выполнить пп. 5 и 6 для обоих станков.
8. Подготовить ответы на контрольные вопросы.

Написать ***рукописный*** отчет по пп. 2, 3, 5–8 с обязательным рассмотрением допущенных ошибок и анализа работы технологических программ.

Для получения допуска к выполнению работы необходимо представить преподавателю первую часть отчета, содержащую титульный лист, и описания по пп. 2, 4–6.

2.3. Описание ГПМ

2.3.1. Структура Гибкого производственного модуля

Гибкий производственный модуль (рис. 2.1) состоит из робота 3, стола 4 для перемещения робота вдоль станков, двух токарных станков 1 и 2, накопителя-стеллажа для заготовок и деталей 5, кнопки аварийного отключения питания 6, а также модулей управления (позиция 6 на рис. 2.2).

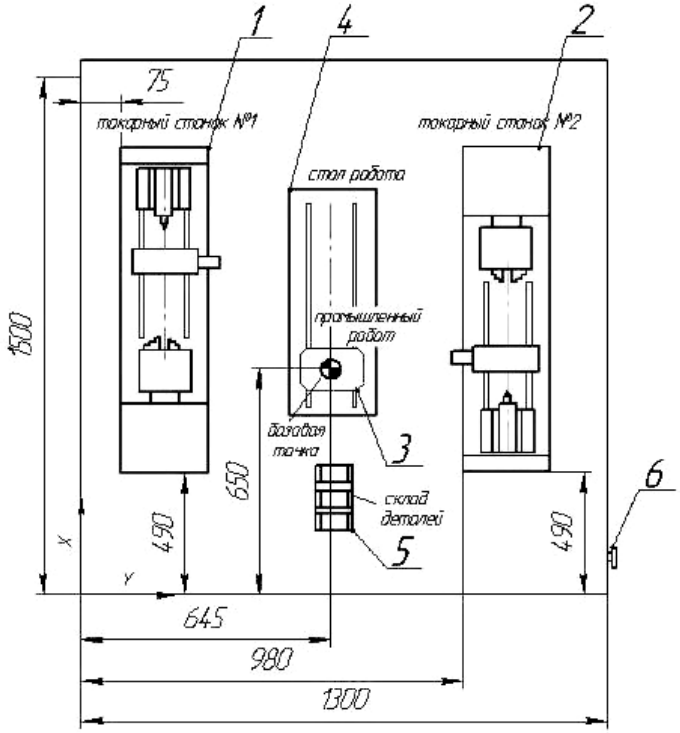


Рис. 2.1. Расположение элементов ГПМ

Внешний вид ГПМ представлен на рис. 2.2. Позиции 1 и 2 – токарные станки, 3 – стол для передвижения робота 4, 5 – стеллаж для заготовок и деталей.

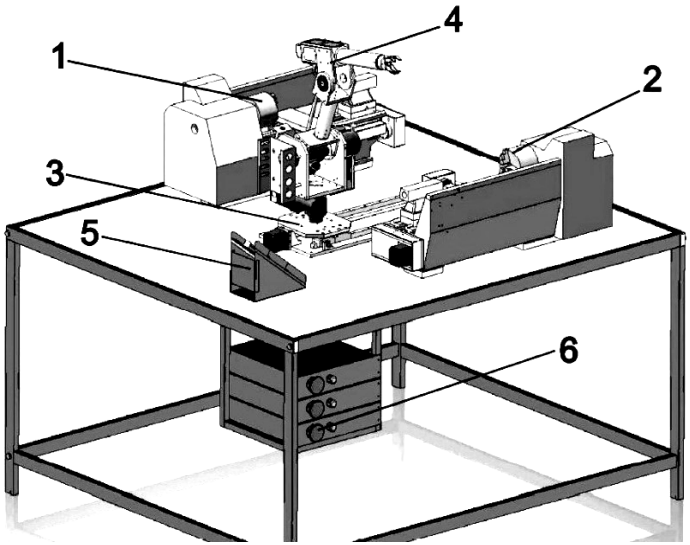


Рис. 2.2. Внешний вид стенда ГПМ

2.3.2. Порядок включения ГПМ

1. Внимательно изучить данное учебное пособие.
2. Проверить подключение всех кабелей к роботу, станку и компьютеру.
3. Подать питание на робот и станки путем подключения стенда в сеть.
4. Включить кнопки питания на блоках управления 6 (рис. 2.2) станками и роботом.
5. Дождаться пока робот и станки автоматически выполнят режим ***восстановление*** (выйдут *в ноль* и *откалибруются*).
6. Включить компьютер и запустить систему управления (имитатор ГПМ), выполнить настройку интерфейса робота и станков.

2.3.3. Интерфейс системы программного управления

Программа имеет модульную структуру. Каждый модуль это отдельная программа, позволяющая работать с рабочими единицами ГПМ. Каждый из модулей может работать автономно, независимо от других. Главная страница интерфейса программы представлена на рис. 2.3.

Модуль управления роботом является главным и позволяет оперировать модулями робота и станков.

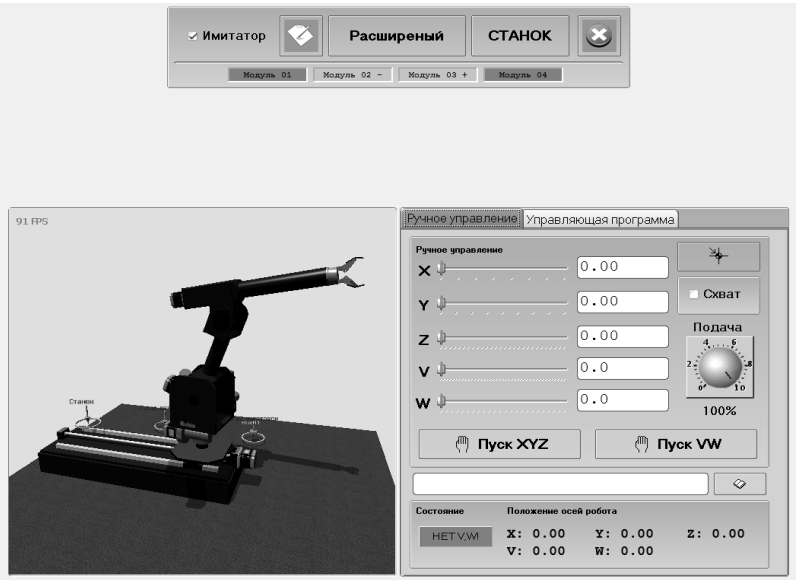


Рис. 2.3. Интерфейс главной страницы программы (сокращенный вариант)

Переключение между модулями при наладке осуществляется нажатием на соответствующую кнопку, расположенную в верхней части экрана. Появляется выпадающий список, где предоставляется возможность выбрать интересующий Вас модуль (станок). ***Зеленым цветом обозначаются модули, которые подключены***. Нажатием на необходимый модуль вызывается окно управления станком.

Основное окно является модулем управления роботом. Функциональные особенности модуля рассмотрим подробнее. Существует два формата модуля – *сокращенный* и *расширенный*. Функциональность этих форматов отличается лишь только возможностью настройки, а также наличием визуализации робота.

В сокращенном формате (рис. 2.3) есть окно визуализации положения робота в пространстве и расположения станков. В расширенном формате (рис. 2.4) есть возможность дополнительной настройки системы, просмотра информации и просмотра управляющей программы (УП).

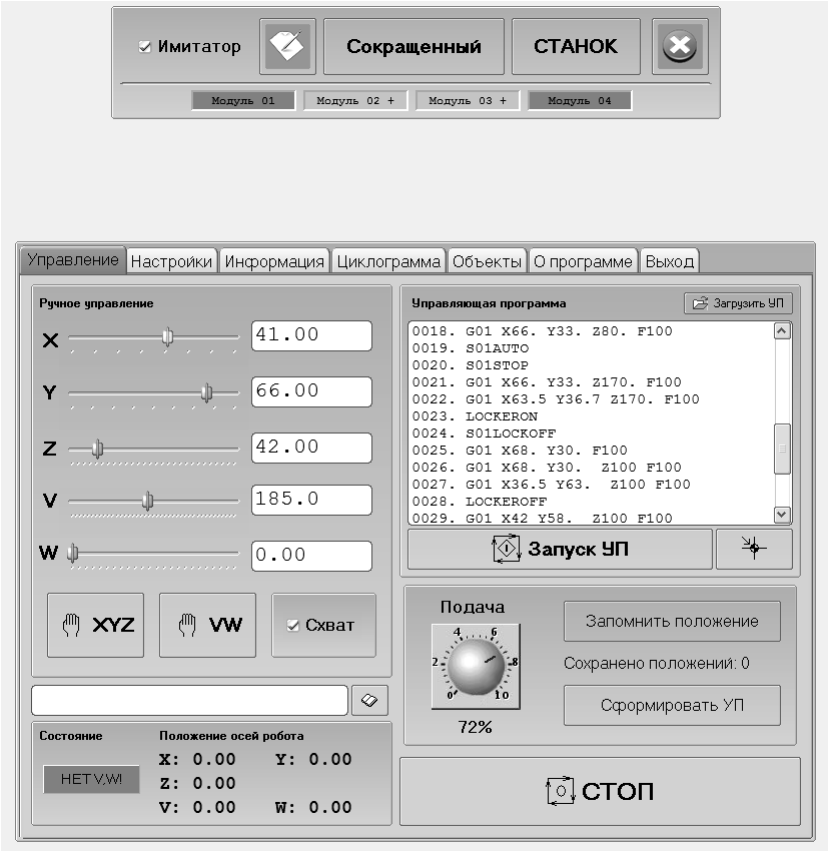


Рис. 2.4. Расширенный вариант интерфейса программы

Переключение между вышеперечисленными форматами осуществляется нажатием на соответствующую кнопку.

Управляющая стендом программа «Робот2010 v1.1» может работать в двух режимах: в режиме **имитатора** и в режиме **станка**. В режиме *Имитатор* можно создавать УП и проводить их отладку виртуально, т.е. не подключая робот и станки. В режиме *Станок* появляется возможность управления и станком и роботом. Переключение между режимами осуществляется постановкой или снятием галочки в поле *Имитатор*, расположенном верхней части окна.

Модули 2, 3 предназначены для вызова управляющих программ для токарного станка – *Stepper токарный*. С помощью этой программы выполняется управление, как в ручном, так и в автоматическом режимах.

Полное руководство по программированию токарного станка представлено в учебных пособиях [2, 3].

Кнопка «Выход» завершает работу программы «Робот2010 v1.1».

Преимуществом модульной структуры программного обеспечения является то, что работу объектов управления можно организовывать *параллельно*.

В сокращенном формате в левом окне появляются два режима – *Ручное управление* и *Управляющая программа*. В ручном режиме ***управлять роботом можно только вручную***.

В правом верхнем углу окна расположена кнопка режима ***восстановления***.

Перемещение рабочих органов робота по осям X,Y,Z и V можно осуществлять перетаскиванием бегунков. В поле расположенном справа от шкалы будет отображаться соответствующее численное значение перемещения.

Перемещения (повороты) по осям X,Y,Z измеряются в градусах, а по V – в миллиметрах. После задания определенных значений, можно привести в движение звенья робота в режиме ручного управления с помощью соответствующих кнопок.

Координаты X,Y,Z могут выполняться как последовательно (отдельно), так и одновременно (с интерполяцией по трем координатам) (но отдельно от линейных координат). Координата V работает по такому же принципу.

Подача для всех координат задается в процентах от максимально возможной.

Слишком большую подачу при обучении задавать не рекомендуется!

Кнопка «Схват» управляет сжатием и разжиманием схвата.

Кнопка вывода робота в нулевое положение выводит робот в аппаратный ноль и обнуляет все датчики координат.

Все вышеперечисленные кнопки дублируются собственными командами. Командная строка расположена под панелью ручного управления.

Например, чтобы передвинуть робот в точку с координатами X10 Y10 Z10, необходимо в командной строке прописать: G01 X10. Y10. Z10. и нажать Enter на клавиатуре или кнопку *Пуск XYZ*.

Полный список команд робота описан в главе 3.3.1.

В нижней части окна отображаются фактические координаты робота.

Кнопка *История команд*, открывает окно, где отображаются все команды посылаемые роботу. Можно сохранить необходимую последовательность команд и преобразовать ее в УП. При необходимости можно произвести очистку окна.

Кнопка *Загрузка УП* позволяет загрузить с диска заранее сформированную управляющую программу. Управляющей программой должен быть текстовый файл с расширением **txt** или **prg**. Формирование УП можно производить в стандартной программе «Блокнот».

Формирование программы можно провести также в автоматизированном режиме путем записи истории команд в отдельный файл, а затем, переходя во вкладку «управляющая программа», нажать «Сформировать УП» и сохранить файл с расширением prg.

Кнопка *Стоп* останавливает выполнение УП. ***После команды Стоп выполнение УП можно возобновить только с начала***.

В окне **расширенного формата** (рис. 2.5) возможно просмотреть и изменить исходные настройки, построить циклограммы работы объектов и т.п.

Все данные, находящиеся во вкладке **настройки** влияют на правильность работы робота и вспомогательных программ (управляющие программы станков). Пользователям рекомендуется не изменять эти настройки:

– дискретность приводов влияет на соответствие виртуального и действительного перемещения,

– сжатие схвата подразумевает задание количества импульсов на сжатие или разжатие в УП,

– максимальные и минимальные скорости влияют на скорости перемещения приводов.

При изменении каких-либо параметров необходимо сохранить их нажатием на соответствующую кнопку.

Кнопка **Внешние модули** дает возможность настроить порты подключения внешних модулей, т.е. станков или иных устройств.

Вкладка **Информация** содержит отчетную информацию о выполненных командах.

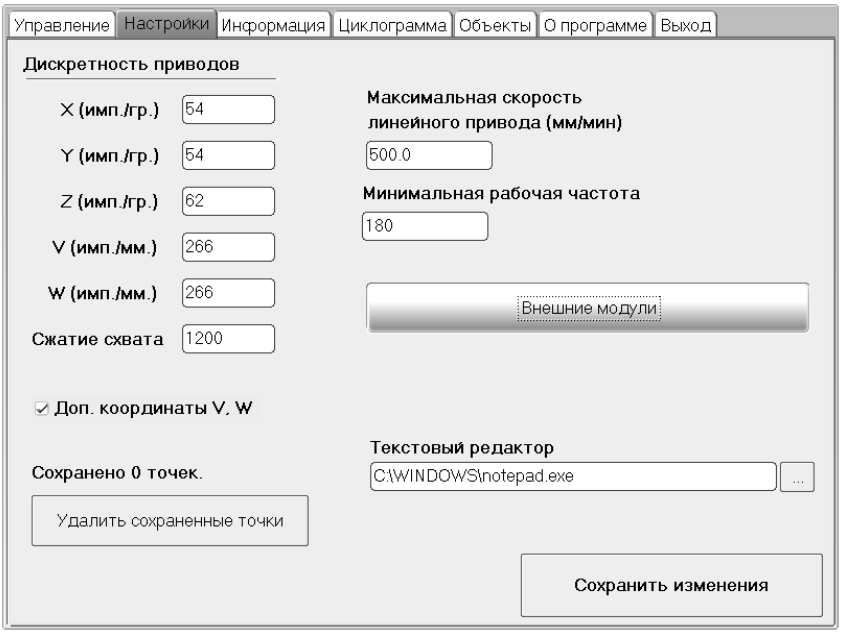


Рис. 2.5. Вкладка «Настройки»

Вкладка **Объекты** позволяет настроить объекты для виртуальной работы, т.е. работы в имитаторе. Окно (рис. 2.6) слева содержит 20 объектов. Объекты могут быть телами или маркерами, в зависимости от настройки. Тело – это физический объект, который робот может зажать и переместить. Маркер – это виртуальная метка, расположенная в пространстве, в которую необходимо переместить тело.

Вкладка **Выход** содержит команду выхода из программы, которая завершает сеанс и сохраняет измененные настройки.

Перед окончанием работы ГПМ необходимо использовать режим *восстановление*. В противном случае в следующий сеанс работы может случиться авария.

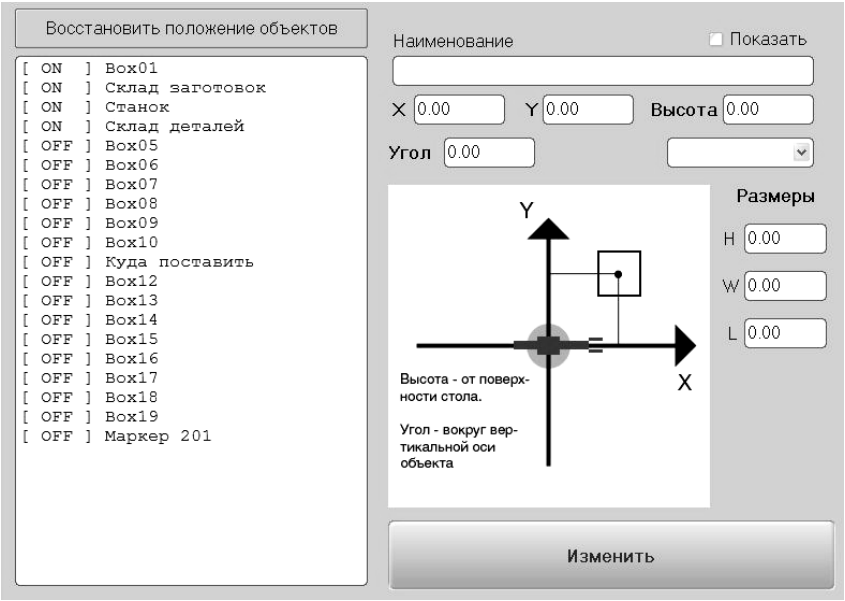


Рис. 2.6. Окно настройки объектов для работы в имитаторе

2.4. Контрольные вопросы

1. Основные положения безопасной работы со стендом?
2. Правила техники безопасности при эксплуатации ГПС.
3. Что такое *Гибкое автоматизированное производство*?
4. Назначение стенда ГПМ?
5. Состав и структура стенда?
6. Основные требования к станкам для ГПС.
7. Основные требования к роботу для ГПС.
8. Порядок включения стенда?
9. Что такое выход в *ноль*? Как он реализуется? В каких случаях он используется?
10. Что будет, если в технологической программе не использовать выход в *ноль*?
11. Зачем используется режим ручного управления?
12. Технические характеристики ГПМ.
13. Особенности устройства робота ГПМ?
14. Назначение и характеристики токарного станка?

15. Особенности устройства токарного станка?

16. Какие привода используются в конструкции робота и их особенности?

17. Какие привода используются в конструкции токарного станка, особенности при работе с ними?

18. Какие датчики используются в конструкции робота?

19. Какие датчики используются в конструкции токарного станка?

20. Как работает система выхода в ноль робота ГПМ?

21. Как работает система выхода в ноль токарного станка ГПМ?

22. Какими элементами стенда ГПМ можно управлять вручную, без использования программного обеспечения? Как это происходит?

23. Как организован интерфейс стенда ГПМ?

24. Какими элементами стенда ГПМ можно управлять виртуально? Каким образом?

25. Какими элементами стенда ГПМ можно управлять программно? Каким образом?

3. ПРОГРАММИРОВАНИЕ ГПМ

Лабораторная работа № 3 (4 часа)

3.1. Цель работы

Целью данной лабораторной работы является изучение допустимых команд для управления технологическими объектами, принципов построения технологических (УП) программ для управления гибким производственным модулем [2, 3].

3.2. Задания к лабораторной работе

1. Изучить структуру кадра технологической программы.
2. Изучить учебные УП (см. тестовые программы 2 и 3 из Приложения 3.2), записать их в отчет с комментариями к основным действиям работы робота и станков.
3. Используя тестовые программы 2 и 3 из приложения 3.2, составить, набрать и загрузить учебную управляющую программу для робота без использования команд управления станками.
4. Составить, набрать и загрузить управляющую программу для управления станками (смотреть приложение 3.2 и главу 3.4).
5. В режиме имитатора убедиться, что робот и станки выполняют все запрограммированные действия и не возникает аварийной ситуации.
6. Запустить ГПМ в работу по набранным программам.
7. Реализовать параллельную работу робота и станков путем интегрирования УП станками в программу управления роботом. Отладить программу и повторно запустить ГПМ в работу.
8. Проанализировать результат работы ГПМ.
9. Подготовить ответы на контрольные вопросы.
10. Написать ***рукописный*** отчет по пп. 1–9 с обязательным рассмотрением допущенных ошибок и анализа работы технологических программ.

Для получения допуска к выполнению работы необходимо представить преподавателю первую часть отчета, содержащую титульный лист, и описания по пп. 1–2.

3.3. Управление роботом

3.3.1. Управление перемещением робота

Характеристики робота даны в табл. 3.1 прил. 3.1.

Управляющая программа состоит из последовательного набора операторов – *функций*. Каждый оператор определяет, какие элементарные операции и с какими параметрами должен выполнить технологический объект. В стенде используется несколько измененный код ISO для составления технологических программ. В частности ***не используется номер кадра.***

Перемещение по осям робота задается функцией **G00** или **G01**:

G01 X… Y… Z… V… F… , причем значение X,Y, Z – в градусах, значение V – в миллиметрах.

Если указаны какие-либо из параметров X, Y, Z – выполняется одновременное перемещение по указанным координатам в заданную точку со скоростью F.

Если параметр F в кадре не указан – перемещение будет выполнено с заданной ранее скоростью.

Например: G01 X10.5 Z15.6

Если в команде G01 указан параметр V, то параметры X, Y, Z игнорируются и выполняется движение только по координате V.

Например: G01 V45.5

Выход в исходное положение (*восстановление*) требует выполнение двух команд:

RHOME – координаты X, Y, Z будут возвращены в исходное положение, схват будет разжат.

SHOME – координата V будет возвращена в исходное положение.

3.3.2. Управление схватом

Управление схватом осуществляется шаговым двигателем.

LOCKERON P – сжатие схвата на заданное количество шагов.

Пример: LOCKERON P500

Если параметр P не задан, будет использовано значение, указанное на вкладке «Настройки».

Пример: LOCKEROFF P500 – разжатие схвата на заданное количество шагов.

3.4. Управление станками

Характеристики станка даны в прил. 3.1, табл. 3.2.

Лабораторный стенд ГПМ содержит два станка (программно команды управления первым станком распознаются приставкой **S02**, а вторым станком – приставкой **S03**), которые программируются аналогично. Для первого станка используются следующие основные команды.

S02AUTO – запуск на станке загруженной Управляющей программы. Параметр **NOWAIT** позволяет запустить УП на станке независимо от работы других элементов ГПМ.

Пример:

S02AUTO NOWAIT.

S02LOCKON P – включение зажима детали на станке. Параметр P задает величину сдвига зажима в мм. Если P не задан, то по умолчанию принимается значение 10мм.

Пример: S01LOCKON P5000

S02LOCKOFF P – зажим на станке – в исходное положение (т.е. разжим заготовки). Если P не задан, то величина сдвига зажима по умолчанию принимает значение 10мм.

S02STOP – прерывание выполнения текущей команды на станке и остановка ***главного*** движения.

S02WAITFORREADY – ожидание готовности Модуля 02.

В приложении 3.2. представлены учебные программы управления роботом и станками.

3.5. Контрольные вопросы

1. Что такое *коды ISO*?
2. Структура технологического программирования в кодах ISO?
3. Формат кадра в кодах ISO?
4. Особенности системы программирования робота ГПМ?
5. Как реализована система программирования токарного станка?
6. Что такое *ускоренное движение*?
7. Какими командами задается выход робота в заданную точку?
8. Какими командами задается выход робота в точку с относительными координатами?
9. Какими командами можно задать траекторию движения робота?
10. Какими командами можно задать скорость движения робота?
11. Что такое *линейная* интерполяция?

12. Что такое *нелинейная* интерполяция?
13. Какие команды условного управления Вы знаете?
14. Как программируется взаимодействие между роботом и станками ГПМ?
15. Какая из программ (станка или робота) является главной?
16. В чем проблемы наладки ГПМ?

Приложение 3.1

Таблица 3.1

Технические характеристики робота со сферической системой координат

Количество степеней свободы	3+схват
Тип системы управления	PCNC
Максимальный вылет кисти, мм	500
Углы поворота звеньев, град – основание – плечо – предплечье	 340 0–80 0–95
Грузоподъемность при макс. вылете, г	1000
Величина раскрытия схвата, мм	90
Максимальное усилие сжатия схвата, Н	30
Минимальный шаг поворота по осям, град – основание – плечо – предплечье	 0,5 0,5 0,5
Максимальная скорость разворота по осям, град/с	30
Погрешность повторяемости, не более, мм	1
Максимальное перемещение стола, мм	450
Тип интерфейса	USB
Питание	220В, 50 Гц
Максимальная потребляемая мощность, Вт	70

Таблица 3.2

Основные параметры станка модели WM180Ф3

Обрабатываемые материалы	Металлы, пластмассы, дерево
Номинальное напряжение питания	220
Тип двигателя главного движения	Коллекторный, постоянного тока
Максимальная частота вращения шпинделя, мин$^{-1}$	2500
Диаметр сквозного отверстия шпинделя, мм	21
Максимальный диаметр обрабатываемой в центрах заготовки, мм	100
Максимальная длина обрабатываемой заготовки, мм	270
Наибольший диаметр изделия, зажимаемого в патроне (обратные кулачки), мм	30 (80)
Максимальная скорость подачи, мм/мин	600
Число одновременно управляемых координат	2
Тип приводов подач	Шаговые
Класс системы ЧПУ	PCNC
Виды интерполяции	Линейная, круговая, сплайновая
Мощность, потребляемая станком, кВт, не более	1

Приложение 3.2

Примеры учебных программ

Тестовая программа 1 для токарного станка ГПМ

```
N1 M03 S1000
N2 G00 Z-80. F100
N3 X-10.
N4 Z-20.
N5 X0.
N6 X-10. Z-80.
N7 X0. Z-20.
N8 X-10.  Z-80.
N9 X0. Z-20.
N10 M06 T2
N11 X-10. Z-80.
N12 X0. Z-20.
N13 X-10. Z-80.
N14 X0. Z-20.
N15 X-10. Z-80.
N16 X0. Z-20.
N17 X-10. Z-80.
N18 X-10. Z-80.
N19 X0. Z-20.
N20 X-10. Z-80.
N21 X0. Z0.
N22 M05
N23 M0
```

Тестовая программа 2 для робота ГПМ

Робот берет деталь из верхней ячейки и устанавливает в станок № 3, запускается программа обработки, в это время робот берет заготовку из ячейки 2, устанавливает в станок № 2. Далее ожидание готовности станка № 3 и снятие детали в ячейку 1 склада готовых изделий. Далее ожидание готовности станка № 2 и снятие детали в ячейку 2 склада готовых изделий.

```
N1 G01 Z12. F200
N2 G01 X38. Y35.
N3 G01 X29. Y43. F100
N4 LOCKERON
N5 G01 X10. Y10. F200
N6 G01 Z80. F200
N7 G01 V230. F300
N8 G01 X56. Y18. F200
N9 G01 V207.
N10 S03LOCKON P5500
N11 LOCKEROFF
N12 G01 X65. Y8. F200
N13 G01 X0. Y0. F200
N14 S03AUTONOWAIT
N15 G01 V0. F200
N16 G01 Z12. F200
N17 G01 X52. Y30. F200
N18 G01 X44. Y43. F200
N19 LOCKERON
N20 G01 X50. Y30. F200
N21 G01 X0. Y10. F200
N22 G01 Z260. F200
N23 G01 V280. F200
N24 G01 X56. Y18.2 F200
N25 G01 V287. F200
N26 S02LOCKON P8500
N27 LOCKEROFF
N28 G01 X70. Y3. F200
N29 G01 X0. Y0. F200
N30 S02AUTONOWAIT
N31 RHOME
N32 SHOME
N33 S03WAITFORREADY
N34 G01 Z80. F200
N35 G01 V210. F200
N36 G01 X65. Y8. F100
N37 G01 X56. Y18. F100
N38 LOCKERON
N39 S03LOCKOFF P5500
N40 G01 V217. F200
N41 G01 X60. Y8. F200
N42 G01 X0. Y0. F200
N43 G01 Z12. F200
N44 G01 V0. F200
N45 G01 X38. Y35. F100
N46 G01 X29. Y43. F100
N47 LOCKEROFF
N48 G01 X40. Y35. F100
N49 G01 X0. Y0. F200
N50 S02WAITFORREADY
N51 G01 Z260. F200
N52 G01 V287. F200
N53 G01 X60. Y8. F100
N24 G01 X56. Y18.2 F100
N55 LOCKERON
N56 S02LOCKOFF P8500
N57 G01 V270. F200
N58 G01 X60. Y8. F200
N59 G01 X0. Y0. F200
N60 G01 Z12. F200
N61 G01 V0. F300
N62 G01 X52. Y30. F200
N63 G01 X44. Y43. F100
N64 LOCKEROFF
N65 G01 X52. Y30. F100
N66 G01 X0. Y0. F200
N67 RHOME
N68 SHOME
N69 END
```

Тестовая программа 3 для робота и токарного станка ГПМ

Технологический цикл будет состоять из следующих операций:

1. Робот, перемещаясь по столу согласно управляющей программе, берет заготовку из склада, перемещает её в рабочую зону токарного станка № 1 (ТС № 1).
2. Станок № 1 зажимает заготовку между упорными центрами и начинает производить её обработку по технологической программе № 1.
3. Далее робот, не дожидаясь окончания работы ТС № 1, возвращается к складу, берет новую заготовку и переносит ее на токарный станок № 2 (ТС № 2).
4. Станок № 2 также зажимает заготовку между упорными центрами и обрабатывает ее по ТП № 2.
5. В это время робот возвращается в исходное положение ожидать готовности модулей.
6. После завершения работы станков робот забирает готовые детали и переносит ее на склад готовых изделий.

Чертеж детали и предварительные настройки для станков приведены в лабораторной работе № 4.

Управляющая программа для робота:

```
G01 Z13 F150
G01 X37 Y27
G01 X30 Y41 //Робот перемещается в положение для захвата заготовки
LOCKERON P1200 //робот зажимает заготовку в схвате
G01 X0 Y0
G01 V225
G01 Z80
G01 X57 Y16
G01 V215 //Робот перемещает заготовку в рабочую зону ТС №1
S02LOCKON P4750 //ТС №1 зажимает заготовку в упорных центрах
LOCKEROFF P1200 //Робот разжимает заготовку
G01 X58 Y0
G01 X0
RHOME
SHOME // Робот возвращается в исходное положение
```

S02AUTO NOWAIT //запуск технологической программы №1 на станке с пометкой без ожидания, т.е. управление сразу возвращается роботу без ожидания окончания ТП

G01 X51 Y33 Z13 F150

G01 X48 Y39 // Робот перемещается для захвата второй заготовки

LOCKERON P1200 //схват робота зажимает заготовку

G01 X0 Y0

G01 V279

G01 Z260

G01 X58 Y15

G01 V297 // Робот перемещает заготовку в рабочую зону ТС №2

S03LOCKON P7350 //станок зажимает заготовку в центрах

LOCKEROFF P1200 //робот разжимает заготовку

G01 Y0

RHOME

SHOME //Робот возвращается в исходное положение

S03AUTO //запуск на ТС №2 технологической программы №2

G01 V230 F150

G01 Z80

G01 X58 Y0

G01 X57 Y16 // Робот перемещается в рабочую зону ТС №1

LOCKERON P1450 //Робот зажимает готовую деталь

S02LOCKOFF P4600 //Станок №1 разжимает упорные центра

G01 V237

G01 X0 Y0

G01 Z165

G01 V281

G01 X54 Y32 // Робот переносит деталь на склад готовых изделий

LOCKEROFF P1450 //Робот разжимает схват

G01 X0 Y0

G01 Z260

G01 X61 Y5

G01 X57 Y16 // Робот перемещается ко второму станку

LOCKERON P1450 //Робот зажимает схват

S03LOCKOFF P7300 //Станок разжимает деталь

G01 V270

G01 X0 Y0

G01 Z165

G01 V281

G01 X54 Y32 // Робот перемещает деталь на склад готовых изделий

LOCKEROFF P1450 //Робот разжимает схват

RHOME //Возвращение с исходное положение

SHOME

Технологическая программа для токарного станка № 1

G92 X11.14. Z-0.5. //установка вылетов для режущего инструмента

M04 S1000 //включение движения шпинделя против часовой стрелки со скоростью 1000 об/мин

G01 X19. Z0. F100

G01 X19. Z-111.

G01 X18.5 Z-111.

G01 X18.5 Z0.

G01 X18. Z0.

G01 X18. Z-111. F50

G01 X19. Z-111. F100

G01 X19. Z-11. // Черновая обработка

G01 X18. Z-11. F100 //Начало чистовой обработки

G01 X15. Z-15.

G01 X16. Z-15.

G01 X16. Z-30.

G01 X18. Z-30.

G01 X18. Z-15.

G01 X15. Z-15.

G01 X15. Z-30. F50

G02 X18. Z-55. R150 F50 //использование круговой интерполяции для формирования радиусной поверхности

G02 X15. Z-80. R150 F50

G01 X16. Z-80. F100

G01 X16. Z-95.

G01 X18. Z-95.

G01 X18. Z-80.

G01 X15.5. Z-80.

G01 X15.5. Z-95.

G01 X16. Z-95.

G01 X16. Z-80.

G01 X15. Z-80.

G01 X15. Z-95. F50

G01 X18. Z-100.

G01 X18. Z-111.

G01 X20. Z-111. F100

G01 X40. Z0. //завершение чистовой обработки

M05 //выключение вращения шпинделя

M02 //конец программы

Технологическая программа для токарного станка № 2

G92 X-0.3. Z1. //установка вылетов для режущего инструмента

M04 S1000 //включение движения шпинделя против часовой стрелки со скоростью 1000 об/мин

G01 X19. Z2. F100 // Черновая обработка

G01 X19. Z-111.

G01 X18.5 Z-111.

G01 X18.5 Z2.

G01 X18. Z2.

G01 X18. Z-111. F50

G01 X19. Z-111. F100

G01 X19. Z-11.

G01 X18. Z-11. F100

G01 X15. Z-15.

G01 X17. Z-15.

G01 X17. Z-40.

G01 X18. Z-30.

G01 X18. Z-15.

G01 X16. Z-15.

G01 X16. Z-30.

G01 X18. Z-30.

G01 X18. Z-15.

G01 X15. Z-15.

G01 X15. Z-30. F50 //начало чистовой обработки

G01 X17. Z-30. F100

G02 X18. Z-55. R150 F50 //использование круговой интерполяции для формирования радиусной поверхности

G01 X18. Z-30. F100

G01 X16. Z-30.

G02 X18. Z-55. R150 F50

G01 X18. Z-30. F100

G01 X15. Z-30.

G02 X18. Z-55. R150 F50

G01 X18. Z-70. F100

G01 X17. Z-70.

G01 X17. Z-97.

G01 X18. Z-97.

G01 X18. Z-76.

G01 X16. Z-76.

G01 X16. Z-96.

G01 X18. Z-96.

G01 X18. Z-55.

G02 X15. Z-80. R150 F50

G01 X15. Z-95. F50

G01 X18. Z-100.

G01 X18. Z-111.

G01 X20. Z-111. F100

G01 X35. Z0. //завершение чистовой обработки

M05 //выключение вращения шпинделя

M02 //конец программы

Приложение 3.3

Система координат станка определяется местоположением начала координат станка – нулевой точки станка. В данном случае применена система плавающего нуля станка, т.е. любое положение крестового стола относительно инструмента может быть принято за нулевое. УЧПУ ориентирована на правую систему координат (ГОСТ 20999-83), при которой взгляд наблюдателя (оператора станка, технолога-программиста УП) со стороны положительного направления оси Y на квадранты плоскости XZ видит отработку функции круговой интерполяции G02 «по часовой стрелке».

Может быть, при необходимости, осуществлена программная установка новой системы координат (детали), установка новой системы координат (детали) параметрическим образом; установка данных инструмента.

Формат кадра

Каждый отдельный кадр Управляющей программы должен соответствовать формату:

[Номер кадра] [Команда] [Параметры команды]

В одном кадре не должно быть одновременно M и G функций с одинаковыми параметрами. То есть если M функция имеет параметр X Y Z, то в этой же строке не должно быть G функции с параметрами X Y Z. В таблице 3.3 приведен список адресных букв, обозначающих команды и их параметры, а та же расшифровки.

Таблица 3.3

Список команд и параметров

Адресная буква	Назначение
N	Порядковый номер кадра
G	Команда задания режима операции (линейная, круговая интерполяция и т.д.)
X, Y, Z	Значения координат
I, J, K	Координаты центра дуги окружности
F	Скорость суппорта
S	Скорость вращения шпинделя
T	Номер корректора инструмента
M	Вспомогательная команда
R	Радиус дуги окружности
P	Длительность паузы, номер подпрограммы, номер фиксированной точки, параметр команды
Q	Параметр команды

Составляющие кадра отделяются друг от друга ОДНИМ пробелом.

Строка, начинающаяся с символа «;», считается комментарием.

Например:

; Включение шпинделя

N100 S1000 M03

Файл управляющей программы представляет собой обычный текстовой файл с расширением .PRG. Каждая отдельная строка файла должна содержать кадр программы или строку комментария, первым символом которой должен быть символ «;». Пустые строки игнорируются при загрузке файла.

Задание значений координат и параметров

Если значение координаты или радиуса задано с десятичной точкой, то значение принимается заданным в миллиметрах. Если значение задано без десятичной точки, то значение принимается заданным в дискретах.

Пример: X100 – 100 дискрет; X100. – 100 миллиметров.

Параметрическое задание значений

Задание значения из параметра: X#200n; Y#210n; Z#220n. Задание значение из переменной: XEn; YEn; ZEn.

Задать значение параметра можно следующим образом:

Пример: N10 #2005 = 35.5 – Значение 35.5; N10 #2005 = E10 – Из переменной E10

Математические операции с параметрами

Сложение: #2001 = #2002 + #2201 – Сумма значений параметров #2002 и #2201 записывается в параметр #2001

Вычитание: #2001 = #2002 – #2201 – Разность значений параметров #2002 и #2201 записывается в параметр #2001

Умножение: #2001 = #2002 * #2201 – Произведение значений параметров #2002 и #2201 записывается в параметр #2001

Деление: #2001 = #2002 / #2201 – Остаток от деления параметра #2002 на #2201 записывается в параметр #2001

Скорость суппорта и шпинделя

Скорость суппорта – Fn (мм/мин)

Скорость шпинделя – Sn (об/мин)

Скорость суппорта в миллиметрах на 1 оборот шпинделя – En

Параметры инструмента

Установка значений вылетов инструмента: N01 Tn. Значения вылетов суммируются с текущей координатой.

Список функций токарного станка:

M00 – Программируемый останов выполнения УП. Выполнение будет продолжено после того, как оператор нажмет кнопку ОК в диалоге, вызванном данной функцией.

M02 – Остановка выполнения программы.

M03 – Включение шпинделя по часовой стрелке

Пример: S1000 M03 – включение шпинделя на частоту 1000 об/мин.

M04 – Включение шпинделя против часовой стрелки

M05 – Останов шпинделя

M06 – Смена инструмента на одну позицию Револьверной головки. В режиме ИМИТАТОР, совместно с параметром T смена инструмента производится в позицию Револьверной головки, указанную параметром T.

Пример: N10 T2 M06 – выбор 2-ой позиции РГ. При этом так же устанавливаются вылеты инструмента для 2-ой позиции револьверной головки.

M13 Pn - Зажим на токарном станке (движение пиноли вперед). Где n – время зажима в секундах.

Пример: M13 P10

M14 Pn - Разжим на токарном станке (движение пиноли назад). Где n – время зажима в секундах.

Пример: M14 P10

M100 – Установка координат фиксированной точки

Пример: M100 Xn Yn Zn P10 – установка фиксированной точки номер 10 с координатами Xn Yn Zn. Допустимо задание значение параметров X, Y и Z из переменных E.

Пример: M100 XE11 P4 – установка значения фиксированной точки по координате X равному значению переменной E11. Допустимо задание значение параметров X, Y и Z из параметров 200n, 210n, 220n.

Пример: M100 X#2005 P4 – Установка значения фиксированной точки по координате X равному значению параметра 2005.

M101 – Установка значения переменной E

Пример: M101 @_число_ En – запись значения _число_ в переменную En.

M102 – Установка значений вылетов инструмента

Пример: M102 Xn Yn Zn P_инструмент_ – установка значений вылетов Xn Yn Zn для корректора инструмента с заданным номером. Допустимо задание значение параметров X, Y и Z из переменных E.

Пример: M102 XE11 P4 – установка вылета по координате X равному значению переменной E11.

M103 – Установка точности аппроксимирования дуги окружности

Пример: M103 Cn – разбиение дуги окружности на n отрезков. Не рекомендуется ставить слишком большую точность – уменьшается быстродействие привода подач.

M200 – Сравнение параметра E с числом: Если E < @ тогда _

Пример: M200 En @_число_ G71 P100 – сравнение переменной En с числом: Если E < @ тогда выполнение функции G71 P100.

M201 – Сравнение параметра E с числом: Если E > @ тогда _Действие_

Пример: N01 M201 En @_число_ G71 P100 – Сравнение переменной En с числом: Если E > @ тогда выполнение функции G71 P100.

M202 – Сравнение параметра E с числом: Если E = @ тогда _Действие_

Пример: N01 M202 En @_число_ G71 P100 – Сравнение переменной En с числом: Если E = @ тогда выполнение функции G71 P100.

M203 – Сравнение параметра E с числом: Если E != @ тогда _Действие_ Пример: M203 En @_число_ G71 P100 – Сравнение переменной En с числом: Если E != @ тогда выполнение функции G71 P100.

M300 – Сложение E = E + Число

Пример: M300 En @_число_ – Сложение переменной En с Числом и запись результата в переменную En.

M301 – Вычитание E = E – Число

M302 – Умножение E = E * Число

Пример: M302 En @_число_ – Умножение переменной En на Число и запись результата в переменную En.

M303 – Деление E = E / Число

M305 – Сохранить переменные E в файл evariables.dat

M306 – Загрузить переменные E из файла evariables.dat

G00 – Перемещение в точку на максимальной скорости

Пример: G00 Xn Yn Zn

G01 – Линейная интерполяция.

Пример: G01 Xn Zn

G02 – Круговая интерполяция (по часовой стрелке)

Пример: N15 G02 U-10. V-10. I-10. K0. F150 – Дуга окружности, конечная точка которой находится со смещением U-10. V-10. от начальной точки, центр окружности находится со смещением I-10. K0. от начальной точки.

I – Относительное смещение центра окружности относительно начальной точки по координате X.

K – Относительное смещение центра окружности относительно начальной точки по координате Z.

Другой вариант задания дуги – с помощью радиуса дуги:

Пример: N10 G02 X-40. Z-20. R50 F100

G03 – Круговая интерполяция (против часовой стрелки)

G04 – Пауза

Пример: N01 G04 P10 – Пауза 10 секунд.

G25 – Включение контроля Зон запрета перемещений. Зоны должны быть определены через меню «Токарный станок – Зоны запрета»

G26 – Отмена контроля зон запрета.

G28 – Нарезание резьбы с одного прохода.

Пример: N01 G28 Z-30. E1 – Резьба с шагом 1 мм. Перед запуском команды G28 ***обязательно*** должно быть запущено главное движение. Параметром E задается скорость суппорта в мм/об – миллиметров на 1 оборот шпинделя.

G37 – Выход в фиксированную точку

Пример: N01 G37 Pn – Выход в точку, заданную параметром n. См. M100.

Пример: N01 G37 X-20. Z-30. – Выход в точку с координатами X, Z.

G70 – Возврат из подпрограммы

Пример: N01 G70 – Последний кадр подпрограммы.

G71 – Вызов подпрограммы

Пример: N01 G71 P200 – Вызов подпрограммы, которая начинается с кадра

N200. Подпрограмма должна завершаться командой G70.

G72 – Безусловный переход на заданный кадр

Пример: N01 G72 N150 – Переход к кадру N150.

Тоже самое: N01 G72 P150 – Переход к кадру N150.

G92 – Задание смещения центра координатной системы

Пример: N01 G92 Xn Yn Zn

G93 – Отмена смещения центра координатной системы

Пример: N01 G93

G500 – Вывод на экран сообщения с указанным номером. Выполнение УП прерывается. Система ожидает нажатия на кнопку OK.

Пример: N102 G500 P4 – вывод сообщения с номером 4. Редактирование сообщений осуществляется через меню Настройка – Функция G500.

Пример: N102 G500 Px En – вывод сообщения с номером x и значение переменной En.

Пример: N102 G500 P1 #2xxx – вывод сообщения с номером 1 и значение параметра #2xxx.

С функциями продольного точения, торцевой обработки, обработки заданного числа канавок и техникой использования сплайновой интерполяции рекомендуется ознакомиться в пособии [2].

4. ИЗГОТОВЛЕНИЕ РЕАЛЬНОЙ ДЕТАЛИ С ИСПОЛЬЗОВАНИЕМ ГПМ

Лабораторная работа № 4 (6 часов)

4.1. Цели работы

1. Ознакомиться с принципами технологической подготовки при изготовления реального изделия на основе чертежа.
2. Научиться разрабатывать реальную технологическую программу для управления ГПМ.
3. Научиться использовать ГПМ для изготовления реального изделия на основе чертежа.

4.2. Задания к лабораторной работе

1. Получить у преподавателя вариант внешнего вида изделия. Нарисовать чертеж в соответствии с исходными геометрическими размерами заготовки.
2. Разработать технологию изготовления реального изделия. Предварительно с полным списком G и M кодов токарного станка ознакомиться в приложении 3.3.
3. Написать технологическую программу для изготовления изделия.
4. Отладить программу на виртуальной модели.
5. Запустить программу на ГПМ и отладить программу, то есть добиться изготовления изделия, адекватному заданному чертежу.
6. Ответить на контрольные вопросы.
7. Написать отчет, содержащий описание работы и результаты по пп. 2–6 с объяснением допущенных ошибок при разработке и отладке программы.

4.3. Подход к разработке технологии при работе с ГПМ

4.3.1. Некоторые сведения о токарной обработке

Токарная обработка – наиболее распространенный метод изготовления деталей типа тел вращения (валов, дисков, осей, пальцев, фланцев, колец и др.) на токарных станках. На них можно производить обтачивание и растачивание цилиндрических, конических, шаровых и профиль-

ных поверхностей этих деталей, подрезание торцов, вытачивание канавок, нарезание наружных и внутренних резьб, накатывание рифлений, сверление, зенкерование, развертывание отверстий и другие виды токарных работ. Иными словами обработка на токарных станках представляет собой изменение формы и размеров заготовки путем снятия припуска.

Припуском называется слой металла, который необходимо удалить с заготовки для получения детали в окончательно обработанном виде. Слой металла, снимаемый на токарном станке, называется *припуском на токарную обработку.*

Резание металлов осуществляется инструментами, имеющими, как правило, форму клина. Это объясняется способностью клина создавать выигрыш в силе, необходимой для проникновения инструмента в обрабатываемый материал. Причем этот выигрыш возрастает по мере уменьшения угла заострения клина (рис. 4.1).

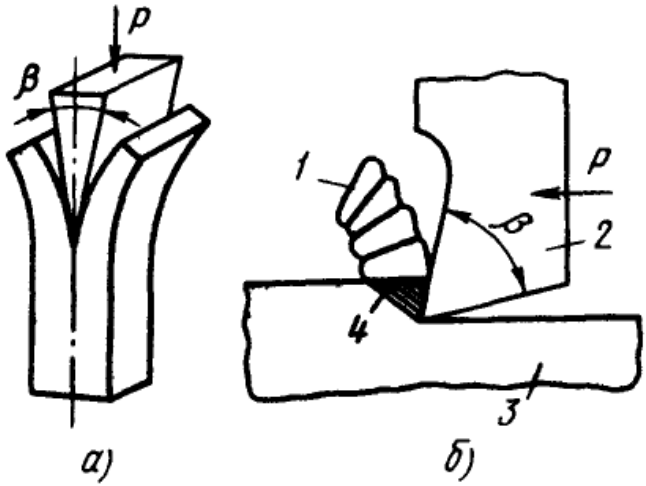


Рис. 4.1. Схема работы клина (*а*) и резца (*б*): 1 – стружка, 2 – резец, 3 – заготовка, 4 – снимаемый слой материала; *P* – сила, действующая на резец и клин при работе, β – угол заострения [4]

На рис. 4.2 схематично показано обтачивание детали 1 резцом 2 [5]. Деталь при этом вращается по стрелке υ, а резец перемещается по стрелке s. Первое из этих движений является *главным*. Второе движение – *движением подачи*.

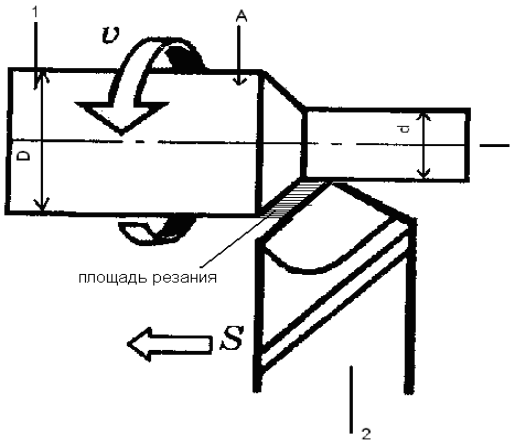


Рис. 4.2. Движения и элементы резания при точении [5]

Процесс резания (стружкообразования) – сложный физический процесс, сопровождающийся большим тепловыделением, деформацией металла, изнашиванием режущего инструмента и наростообразованием на резце. Знание закономерностей процесса резания и сопровождающих его явлений позволяет рационально управлять этим процессом и обрабатывать детали более качественно, производительно и экономично. При резании различных материалов могут образовываться следующие виды стружек: сливные (непрерывные), скалывания (элементные) и надломы (рис. 4.3).

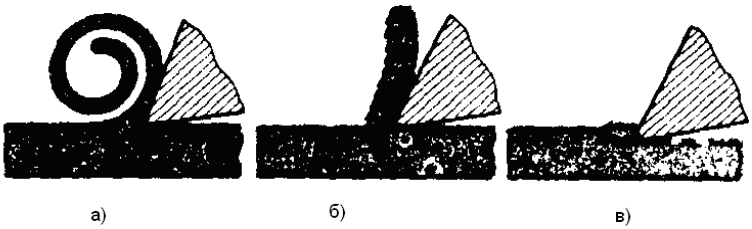


Рис. 4.3. Типы стружек: *а* – сливная, *б* – скалывания, *в* – надлома [4]

Сливная стружка образуется при резании вязких и мягких металлов (мягкая сталь, латунь) с высокой скоростью. Чем больше скорость резания и вязкость обрабатываемого материала, а также меньше угол резания

и толщина среза, и выше качество смазочно-охлаждающей жидкости, тем стружка ближе к сливной.

Стружка надлома образуется при резании хрупких металлов (бронзы, чугуны). Такая стружка состоит из отдельных, почти не связанных между собой элементов. Обработанная поверхность при образовании такой стружки получается шероховатой, с большими впадинами и выступами. В определенных условиях, например при обработке чугунов средней твердости, стружка надлома может получиться в виде колец. Сходство ее со сливной стружкой только внешнее, так как достаточно сжать такую стружку в руке, и она легко разрушится на отдельные элементы.

Стружка скалывания занимает промежуточное положение между сливной стружкой и стружкой надлома и образуется при обработке некоторых сортов латуни и твердых сталей с большими подачами и относительно малыми скоростями резания. С изменением условий резания стружка скалывания может перейти в сливную, и наоборот.

В целях создания наилучших условий для отвода стружки из зоны резания необходимо обеспечить ее дробление или завивание в спираль определенной длины. Дробленую стружку в виде колец и полуколец диаметром 10–15 мм и более следует рассматривать как хорошую. Эта стружка, несмотря на то, что занимает меньший объем и легче транспортируется, снижает стойкость инструмента. Мелкодробленая стружка должна рассматриваться как удовлетворительная.

Помимо снижения стойкости резцов такая стружка, разлетаясь во все стороны, попадает на поверхности станка, нарушает нормальную работу его узлов. Формирование стружки в виде непрерывной спирали, прямой ленты и путаного клубка не удовлетворяет требованиям обработки деталей на станках с ЧПУ и поэтому должно быть исключено.

При некоторых условиях резания на переднюю поверхность режущей кромки налипает обрабатываемый материал, образуя *нарост*. Он имеет клиновидную форму, по твердости в 2–3 раза превышает твердость обрабатываемого металла. Являясь как бы продолжением резца, нарост изменяет его геометрические параметры: участвует в резании металла, влияет на результаты обработки, изнашивание резца и силы, действующие на резец. При обработке нарост периодически разрушается (скалывается) и вновь образуется.

Часть его уходит со стружкой, а часть остается вдавленной в обработанную поверхность (рис. 4.4).

Рис. 4.4. Образование и срыв нароста [5]

Отрыв частиц нароста происходит неравномерно по длине режущего лезвия, что приводит к мгновенному изменению глубины резания. Эти явления, повторяющиеся периодически, ухудшают качество обработанной поверхности, так как вся она оказывается усеянной неровностями. С увеличением пластичности обрабатываемого металла размеры нароста возрастают. При обработке хрупких материалов, например чугуна, нарост может и не образоваться [5].

4.3.2. Рекомендации по планированию технологического процесса

Эффективность эксплуатации станков с ЧПУ во многом определяется их правильным технологическим использованием – рациональным построением технологического процесса и, в частности, выбором режимов резания, обеспечивающих увеличение надежности и производительности обработки [6].

Качество технологии выпуска готового изделия тем выше, чем меньшим количеством инструментов выполняется заданный объем работ и чем меньше времени затрачивается на резание.

Из теории надежности известно, что вероятность безотказной работы нескольких последовательно включаемых элементов равна произведению вероятностей безотказной работы этих элементов. Следовательно, и с точки зрения надежности целесообразно использовать минимально необходимое для заданного объема работ число инструментов. При добавлении инструмента в наладку он должен обеспечивать сохранение надежности работы этой наладки и обладать существенно большей надежностью, чем надежность наладки до его добавления. Отдельные параметры технологического процесса выбирают с учетом того, что содержание операции и требования к ней регламентированы маршрутным технологическим процессом изготовления детали, чертежом детали и ее служебным назначением. Предлагаемые ниже условия существенно облегчают выбор параметров технологического процесса:

Выбор производится в последовательности, соответствующей разработке операционной технологии;

На каждом этапе разработки используют частный критерий оценки качества операционной технологии, исключающий необходимость пересмотра принятого решения на последующих этапах;

Этот частный критерий не противоречит общему критерию оценки качества операционной технологии при всех рассматриваемых условиях.

Инструментальная оснастка. Инструментальная оснастка, применяемая на станке с ЧПУ, должна обеспечивать обработку поверхностей всех форм и размеров деталей, которые обрабатывают на аналогичных станках в конкретных условиях данного производства. Общее число типоразмеров используемого инструмента необходимо минимизировать:

Путем замены фасонного инструмента более простым, но перемещающимся по сложной траектории;

Путем замены мерного инструмента немерным (даже если при этом может потребоваться дополнительный проход);

Применением инструмента, пригодного для различных видов работ (черновых, чистовых; обтачивания, растачивания и т.п.), за счет унификации геометрических параметров режущей части и присоединительных размеров.

Последовательность ввода инструментов в работу определяется требованиями операции. В тех случаях, когда требования к операции могут быть выполнены благодаря различной последовательности ввода инструментов, необходимо выбрать такую последовательность, при которой сумма времен, связанных с выполнением перехода, минимальна.

Выбор траектории движения инструмента. Траектории движения инструмента проектируют так, чтобы минимизировать время обработки и увеличить надежность режущего инструмента. С точки зрения выбора критерия различают траекторию рабочих и траекторию вспомогательных перемещений. Траекторию рабочих перемещений разделяют на участки установившегося резания, врезания и выхода инструмента. На этом этапе разработки операционной технологии уже известны: режущий инструмент, последовательность его ввода и содержание работы каждого инструмента. Поэтому траекторию можно выбирать для каждого инструмента в отдельности.

Надежность инструмента во многом зависит от организации его работы на участках врезания и выхода. На этих участках происходит нагружение технологической системы силами резания, упругая деформация звеньев системы и т.п. Неправильное построение траектории этих участков может привести к снижению производительности, появлению

«зарезов» на обработанной поверхности, поломкам инструмента. Вместе с тем, участки врезания и входа существенно короче участков установившегося резания. Поэтому при выборе траектории врезания варианты следует оценивать по их влиянию на надежность инструмента. Предпочтительны траектории, обеспечивающие плавное увеличение (уменьшение) сил резания до значений, характерных для участка установившегося резания.

О закреплении заготовок. Станочные приспособления оказывают существенное влияние на повышение точности обработки, поскольку погрешность, возникающая при базировании заготовки в приспособлении, является одной из основных составляющих суммарной погрешности обработки. Следовательно, приспособления к станкам с ЧПУ должны обеспечивать большую точность установки заготовок, чем приспособления к универсальным станкам. Для этого необходимо исключить погрешность базирования путем совмещения баз, точки приложения зажимных сил нужно выбирать таким образом, чтобы по возможности полностью исключить деформацию заготовок.

Качество поверхности изготавливаемых деталей. В процессе обработки древесины нельзя получить абсолютно гладкую поверхность вследствие индивидуальных особенностей режущего инструмента, станков, режимов обработки и свойств обрабатываемого материала. В случае обработки древесины на полученной поверхности различают неровности разрушения (риски, сколы, вырывы, ворсистость, мшистость), неровности упругого восстановления и структурные неровности.

Неровности разрушения древесины образуются в результате выколов и вырывов пучков волокон древесины. Выколы и вырывы всегда направлены вдоль волокон и сопутствуют наклону волокон, завиткам и сучкам. Риски – это следы на обработанной поверхности, оставленные рабочими органами режущих инструментов. Они могут иметь форму гребешков и канавок или периодически повторяющихся возвышений и впадин – следствие кинематического процесса резания при цилиндрическом фрезеровании (кинематическая волнистость).

Ворсистость – наличие на поверхности древесины не полностью перерезанных, но отдельных волокон (ворсинок), которые поднимаются и делают поверхность шероховатой. Мшистость – аналогичные неровности, образованные вследствие разрыва волокон древесины в отдельных местах целыми пучками, что также делает поверхность шероховатой.

Неровности упругого восстановления образуются в результате различной твердости годичных слоев древесины и, как следствие, неодинаковой величины упругого смятия режущим инструментом поверхностно-

го слоя древесины. Структурные неровности – это впадины, появившиеся на поверхности деталей, спрессованных из древесных частиц, из-за их различного расположения в деталях.

Любые неровности на поверхности древесины делают ее шероховатой. Шероховатость поверхности древесины характеризуется максимальной высотой неровностей. В процессе обработки древесины различными инструментами можно получить поверхности различной шероховатости. На шероховатость влияют те же факторы, что и на точность обработки. Важнейшие – острота режущего инструмента и режим обработки.

Режимы резания. К элементам режима резания относятся глубина резания, подача и скорость резания.

Глубина резания определяется в основном величиной припуска на обработку.

Например, экспериментально было установлено, что для учебного ГПМ, состоящего из двух токарных станков и робота-манипулятора, в случае использования проходного правого резца со скоростью подачи резца 100 мм/мин и глубиной резания 1,5 мм наблюдается наилучшее качество обрабатываемой поверхности деревянной заготовки с точки зрения шероховатости.

Припуск на обработку выгодно удалять за один проход. Глубина резания оказывает большое влияние на силы резания, поэтому иногда возникает необходимость разделить припуск на несколько проходов. Суммарный припуск разделяется следующим образом: 60% на черновую обработку, 20–30% на получистовую и 10–20% на чистовую.

Для черновой обработки глубину резания принимают t = 3–5 мм, получистовой – 2–3 мм и чистовой – 0,5–1,0 мм.

Величина *подачи* ограничивается силами, действующими в процессе резания; эти силы могут привести к поломке режущего инструмента, деформации и искажению формы заготовки, поломке станка. Целесообразно работать с максимально возможной подачей. Обычно подача назначается из таблиц справочников по режимам резания, составленным на основе специальных исследований и изучения опыта работы машиностроительных заводов. *Для рассматриваемого ГПМ рекомендуется расчитывать величину подачи, не превышающую 0,1 мм/об (например, при частоте вращения шпинделя 1 000 об/мин и скорости перемещения резца 100 мм/мин).*

Заметим, что при одинаковой площади поперечного сечения среза нагрузка на резец меньше при работе с меньшей подачей и большей глубиной резания; нагрузка на станок (по мощности), наоборот, меньше при

работе с большей подачей и меньшей глубиной резания, так как на силу резания глубина оказывает большее влияние, чем подача.

На *скорость резания*, допускаемую резцом, влияют следующие факторы: стойкость режущего инструмента, физико-механические свойства обрабатываемого металла, подача и глубина резания, геометрические элементы режущей части резца, размеры сечения державки резца, смазочно-охлаждающая жидкость, максимально допустимая величина износа резца.

Скорость резания уменьшается с увеличением сопротивления резанию, которое приводит к возникновению больших сил, высокой температуры, интенсивному износу режущего инструмента.

Экспериментально установлено, что для рассматриваемого учебного ГПМ скорость резания на точность позиционирования влияет незначительно, что обусловлено применением шаговых приводов для регулирования скоростей резания в соответствии с заданными в управляющей программе значениями. Одним из главных преимуществ шаговых двигателей является возможность осуществлять точное позиционирование и регулировку скорости без обратной связи.

Подача и глубина резания определяют нагрузку на резец и температуру резания. С увеличением подачи и глубины резания интенсивнее износ резца, что ограничивает скорость резания. Для достижения большей производительности резания выгоднее работать с большими сечениями среза за счет уменьшения скорости резания.

Например, при увеличении подачи в 2 раза скорость резания необходимо уменьшить на 20–25%. При удвоении глубины резания скорость резания должна быть уменьшена на 10–15%. На практике скорость резания увеличивают после того, как достигнуты предельные величины по глубине резания и подаче [7].

4.4. Учет смещения координатной системы при смене инструмента

Данная операция носит название ***Установка вылетов инструмента***. От её выполнения напрямую зависят правильность обработки детали и безаварийная работа станков.

Определение вылетов инструмента следует выполнять следующим образом для каждого конкретного резца:

1. В ручном режиме проточить заготовку до диаметра d на длину b (значения b и d после снятия припуска можно получить измерением непосредственно самой заготовки вручную).

2. Записать текущие координаты резца Z и X.

3. Определить разности (X-d/2) и (Z-b). Это и будут вылеты инструмента по осям X и Z. Заметим, что вышеприведенная формула для определения вылета по оси X действительна при расположении ноля детали на оси вращения шпинделя. Технология назначения ноля детали приведена в гл. 4.5

Если значения вылетов всех инструментов ввести в окно «Параметры инструмента», то при выводе в рабочую позицию соответствующего инструмента, ранее установленного в револьверную головку суппорта станка, будут учитываться соответствующие ему вылеты (рис. 4.5, 4.6).

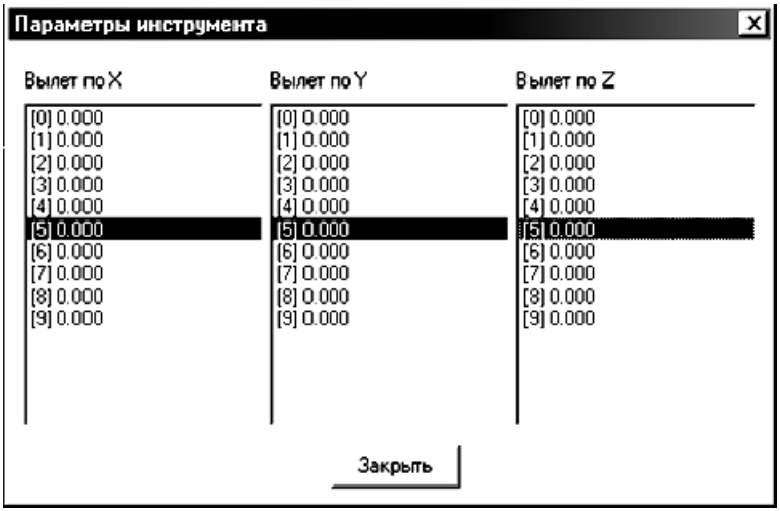


Рис. 4.5. Выбор вкладки вылета для инструмента

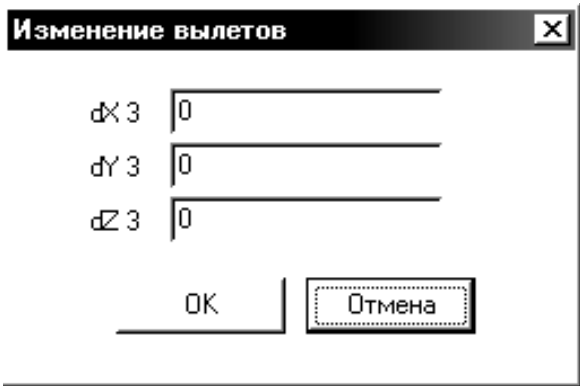


Рис. 4.6. Изменение значений вылетов

Например, введя вылеты всех используемых инструментов в меню «Параметры инструмента», можно программировать управляющую программу не относительно вершин резцов, а относительно известной точки резцедержки, что правильнее, так как данная точка, в отличие от вершины инструмента не меняет своего положения относительно револьверной головки.

4.5. Назначение ноля станка и определение положения ноля детали

Рассмотрим чертеж детали, изображенный на рис. 4.7. В качестве заготовки используется круг диаметром D = 40 мм. В качестве базовой точки (на рисунке – б.т.) принимается пересечение оси шпинделя с плоскостью торца кулачков. Таким образом, вылет заготовки из кулачков патрона относительно базовой точки составляет $L_в$ = 90 мм. В качестве ноля станка (на рисунке – 0ст.) принимается точка, совпадающая с режущей кромкой резца или известной точкой резцедержки (в случае задания вылетов режущего инструмента). В рассматриваемой системе «токарный станок-имитатор» применяется плавающий ноль и в имитаторе он задается относительно базовой точки. На примере рис. 4.7 добавим к вылету заготовки $L_в$ величину ΔZ и примем это расстояние от торца заготовки $L_в+\Delta Z$ за ноль станка по координате Z. Затем к радиусу заготовки D/2 прибавим ΔX и примем значение $D/2+\Delta X$ за расстояние от оси заготовки до нуля станка, т.е. за ноль станка по координате X. Для простоты при-

мем ΔZ и ΔX равными 20мм. Таким образом, получим по оси Z от базовой точки до нуля станка 110 мм, а по оси X – 40 мм.

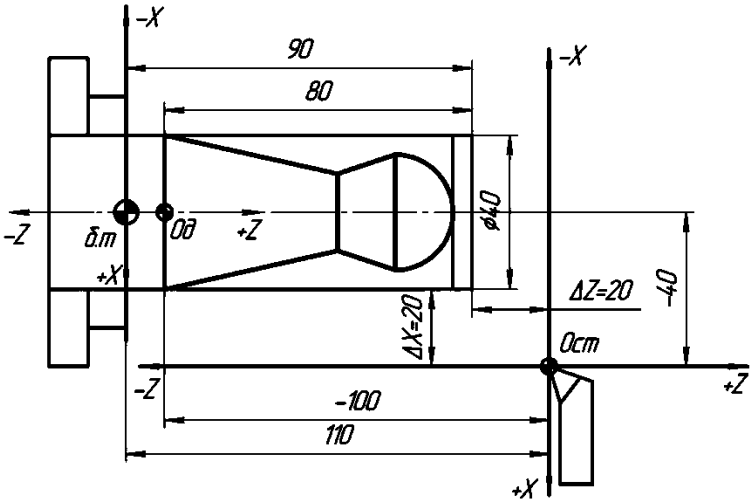


Рис. 4.7. Чертеж детали [2]

Ноль детали назначается технологом, относительно этой точки пишется управляющая программа. Ноль детали (0д) должен быть привязан к нулю станка. Как видно из рис. 4.7, расстояние от нуля станка до нуля детали высчитывается, т.е. в системе координат станка положение ноля детали по оси X = –40, по оси Z = –100.

На рисунке 4.7 ноль детали был назначен рядом с базовой точкой. Однако, для работы на токарном станке, входящем в состав учебной ГПМ, во избежание возникновения аварийных ситуаций настоятельно рекомендуется назначать ноль детали относительно ноля станка в точке пересечения оси шпинделя и плоскости торца заготовки, не зажатого в кулачки патрона.

4.6. Варианты заданий

В табл. 4.1 представлены варианты внешнего вида изделия, которое необходимо изготовить. Информацию о доступных резцах необходимо получить у преподавателя. Настоятельно рекомендуется ознакомиться с

последовательностью работы в режиме «Имитатор» для токарного на примере чертежа обрабатываемой детали в пособии [2].

Приведем пример реализации технологических операций изготовления двух идентичных изделий, геометрические размеры которых представлены на рис. 4.8. Для этого используем деревянную заготовку цилиндрической формы, диаметр которой 40 мм и длина 110 мм.

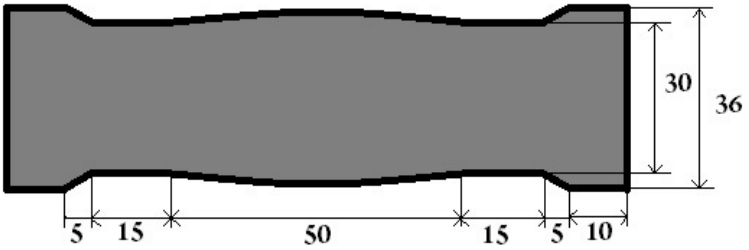


Рис. 4.8. Чертеж готовой детали

Предварительно для станков определяем необходимые параметры, которые для каждого из модулей получаются разные (смотреть расположение модулей ГПМ на рис. 2.1).

Подготовка ТС № 1:

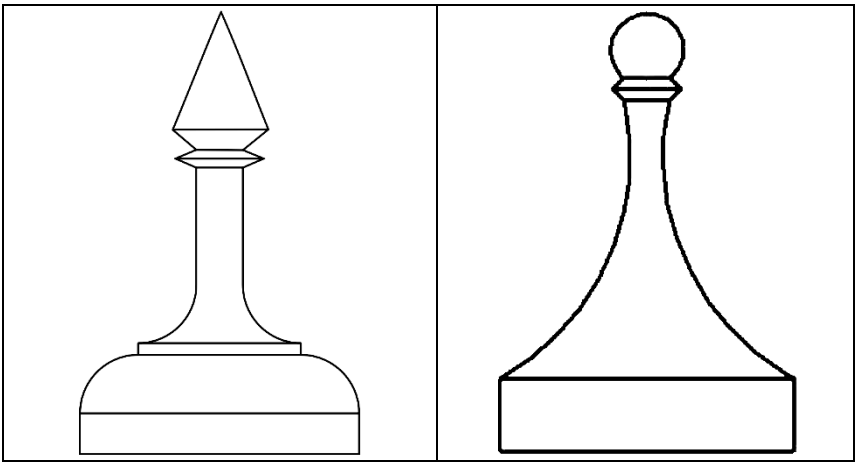
1. Устанавливаем ноль станка, т.е. определяем систему координат станка значениями X = 40, Z = 120.
2. В окне системы координат детали сохраняем значения X = –40, Z = –5.
3. В рабочей позиции револьверной головки устанавливаем резьбовой резец, вылеты которого отражены в технологической программе (ТП) № 1 (в прил. 3.2 – тестовая программа 3 для робота и токарного станка ГПМ), а именно X = 11.14, Z = –0.5.
4. Аналогичная подготовка станка № 2.
5. Устанавливаем ноль станка, т.е. определяем систему координат станка значениями X = 35, Z = 120.
6. В окне системы координат детали сохраняем значения X = –35, Z = –10.
7. В рабочей позиции револьверной головки устанавливаем резьбовой резец, вылеты которого отражены в ТП № 2, а именно X = –0.3, Z = –10.

Затем на каждый станок загружается соответствующая ему технологическая программа. Программа включает в себя черновую и чистовую обработки детали.

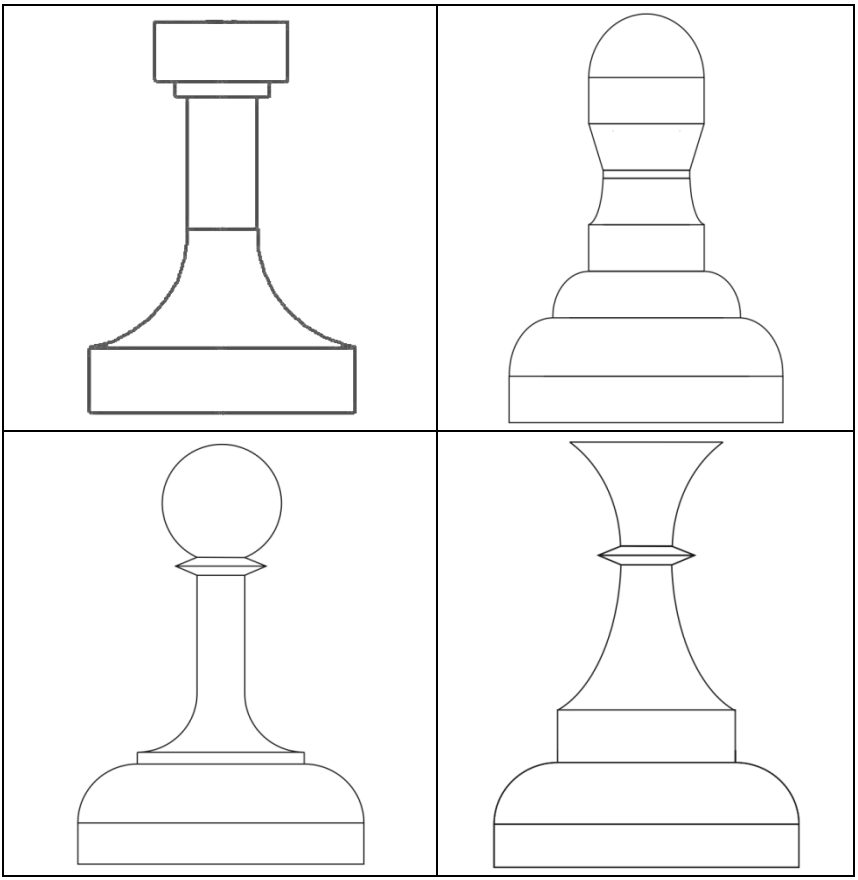
Обращаем внимание, что в технологических программах для станков № 1 и № 2 вследствие использования только одного режущего инструмента значения вылетов, корректирующие местоположение ноля станка относительно ноля детали, задаются вручную во время выполнения программы с помощью функции G92 (см. прил. 3.3), *а не учитываются автоматически, как это было показано в главе 4.4.*

Таблица 4.1

Варианты заданий для самостоятельного выполнения



Окончание табл. 4.1



4.7. Контрольные вопросы

1. Как подготовить токарный станок к работе с заготовкой?
2. Как учитывается размер инструмента на токарном станке?
3. Какие типы *коррекции* используются при работе токарного станка?

5. ИЗУЧЕНИЕ ЛАБОРАТОРНОГО СТЕНДА СИСТЕМЫ ТЕХНИЧЕСКОГО ЗРЕНИЯ РОБОТА

Лабораторная работа № 5 (4 часа)

5.1. Цели работы

1. Изучить структуру и конструкцию стенда робота с техническим зрением [8].
2. Изучить принципы системы технологического программирования робота с техническим зрением.
3. Научиться программировать и отлаживать систему программного управления роботом.

5.2. Задания к лабораторной работе

1. Изучить структуру оборудования, оценить число степеней свободы робота с техническим зрением.
2. Проанализировать особенности исполнительных механизмов (тип датчиков и двигателей, принципы управления ими, точностные характеристики).
3. Научиться включать лабораторный стенд.
4. Научиться составлять технологические программы для робота, предварительно изучив пример УП из прил. 5.1.
5. Научиться настраивать систему фильтрации изображения.
6. Научиться отлаживать программу на виртуальной модели.
7. Разработать программу сборки изделия из предложенных деталей по заданию преподавателя.
8. В режиме управления роботом проверить и отладить разработанные программы.
9. Подготовить ответы на контрольные вопросы.
10. Написать ***РУКОПИСНЫЙ*** отчет по пп. 1–9 с обязательным рассмотрением допущенных ошибок и анализа работы технологических программ.

Для получения допуска к выполнению работы необходимо представить преподавателю первую часть отчета, содержащую титульный лист, и описания по пп. 1–5.

5.3. Описание стенда

Операция сборки является одной из самых трудно автоматизируемых процессов в связи с разнообразием деталей и узлов, трудностями, связанными с идентификацией деталей и их локализацией в пространстве.

В данной работе рассматривается сборочный стенд (рис. 5.1), созданный на базе робота с компьютерной системой ЧПУ.

Робот оснащен web-камерой и системой обработки видеоинформации, т.е. системой технического зрения [8]. Стенд предназначен для изучения программирования робота, системы технического зрения и сборочной операции в целом.

Рис. 5.1. Робот с техническим зрением

5.3.1. Схема компоновки стенда

Сборочный стенд состоит из стола, робота, а также системы управления. Схема расположения указана на рис. 5.2, где показаны расстояния, на которых расположены элементы стенда относительно друг от друга. Данные расстояния устанавливаются предприятием изготовителем и не подлежат изменению.

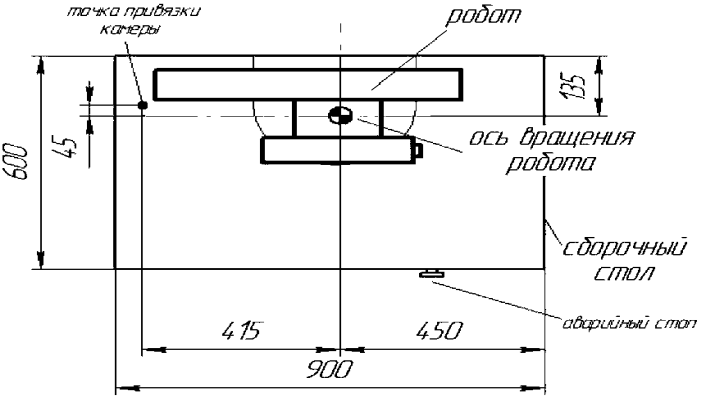


Рис. 5.2. Схема расположения элементов

5.3.2. Интерфейс программы

Программа имеет модульную структуру и обеспечивает возможность управления сборочным роботом с техническим зрением.

Каждый модуль это отдельная программа, позволяющая работать с рабочими единицами стенда. Каждый из модулей может работать автономно, не зависимо от других.

Существует два формата модуля – *сокращенный* и *расширенный*. Функциональность этих форматов отличается лишь только возможностью настройки, а также наличием визуализации робота.

Интерфейс программы в формате *расширенный* представлен на рис. 5.3.

В «*расширенном*» формате есть возможность дополнительной настройки системы, просмотра информации и просмотра управляющей программы (УП). В сокращенном формате есть окно визуализации положения робота в пространстве и расположения объектов. Переключение между вышеперечисленными форматами осуществляется нажатием на соответствующую кнопку.

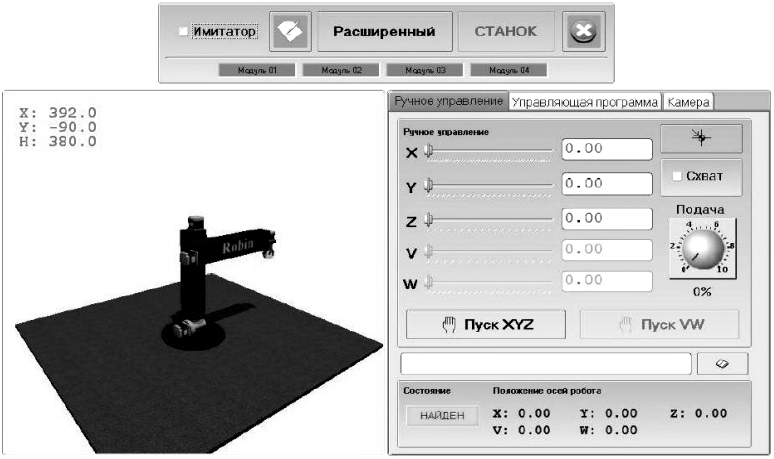


Рис. 5.3. Главная страница интерфейса программы сборочного стенда

Формат модуля – сокращенный представлен на рис.5.4.

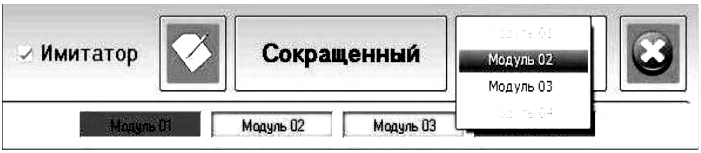


Рис. 5.4. Закладка «Сокращенный формат»

Управляющая программа ***Робот2010 v1.1*** может работать в двух режимах: в режиме имитатора и в режиме реального оборудования.

В режиме ***Имитатор*** можно создавать УП и проводить их отладку виртуально, т.е. не подключая робота. В режиме ***Станок*** появляется возможность управления роботом. Переключение между режимами осуществляется постановкой или снятием галочки в поле Имитатор, расположенном в верхней части окна.

Кнопка ***Выход*** завершает работу в программе «Робот2010 v1.1».

5.4. Управление роботом

Используется модульная структура программного обеспечения. Возможны два вида интерфейса при управлении: ***Сокращенный*** формат и ***Расширенный***.

5.4.1. Сокращенный формат

В сокращенном формате в левом окне имеется два режима – **Ручное управление** и **Управляющая программа**.

Окно ручного управления представлено на рис. 5.5. Из него можно управлять только роботом и только вручную.

Рис. 5.5. Окно ручного управления

Перемещение рабочих органов робота можно осуществлять перетаскиванием бегунков. Бегунки по осям X,Y,Z перемещаются с помощью мыши. В поле расположенном справа от шкалы будет отображаться соответствующее численное значение перемещения.

Внизу рабочего окна (рис. 5.5) отображаются текущие координаты робота.

После задания определенных значений, можно привести в движение звенья робота в режиме ручного управления с помощью соответствующих кнопок.

Движения по координатам X,Y,Z могут выполняться как последовательно, так и одновременно. Подача для всех координат задается в процентах от максимально возможной. **Слишком маленькую подачу задавать не рекомендуется!**

Управление схватом осуществляется при помощи кнопки ***Схват***.

Вывод робота в нулевое положение производится при нажатии кнопки ***Сброс***: 

Данная команда обнуляет все координаты и выводит робот в аппаратный ноль.

Действия всех кнопок дублируются собственными командами. Командная строка расположена под панелью ручного управления.

Например, чтобы послать робота в точку с координатами X10 Y10 Z10, необходимо в командной строке прописать: G01 X10. Y10. Z10. и нажать *Enter* на клавиатуре или кнопку ***Пуск XYZ***.

Кнопка ***История команд***, открывает окно, где отображаются все последние команды. Можно сохранить необходимую последовательность команд и преобразовать ее в УП.

Окно командного управления представлено на рис. 5.6. Здесь можно управлять роботом только посредством УП.

Вызвать из памяти УП можно кнопкой ***Загрузка УП***.

Управляющей программой может быть текстовый файл с расширением txt или prg.

Формирование УП можно производить в стандартной программе ***Блокнот***. Формирование программы можно провести также путем записи истории команд в отдельный файл, а затем, переходя во вкладку *управляющая программа*, нажать кнопку *Сформировать УП* и сохранить файл с расширением ***prg***.

Остановка выполнения УП выполняется нажатием кнопки Стоп. После команды Стоп выполнение УП можно возобновить только с начала.

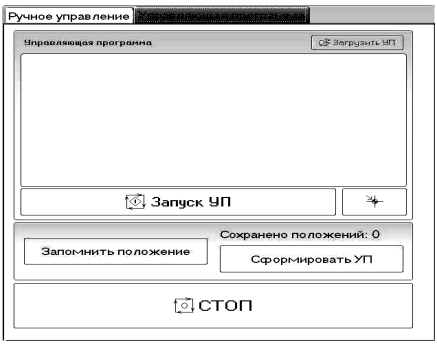


Рис. 5.6. Окно Управляющая программа

5.4.2. Расширенный формат

Расширенный формат (рис. 5.7) целесообразно применять при отладке технологической программы и при работе со стендом в режиме реального времени.

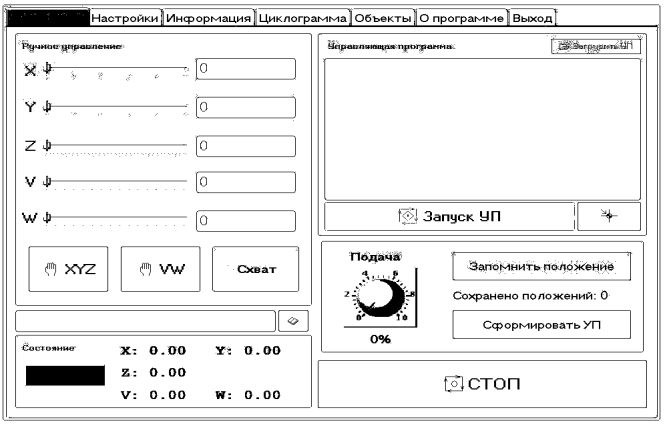


Рис. **5.7.** Окно Расширенного формата

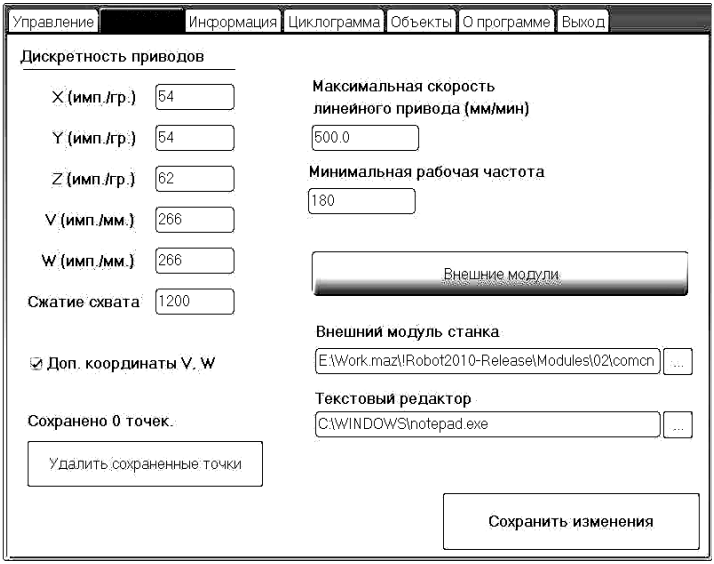


Рис. 5.8. Окно *Настройки*

Вкладка ***Управление*** содержит точно такой же функционал, что и в сокращенной форме.

Вкладка ***Настройки*** показана на рис. 5.8. Все данные, находящиеся в этой вкладке влияют на качество управления роботом.

Пользователям рекомендуется не изменять эти настройки. При изменении каких-либо параметров выполняется их сохранение нажатием на соответствующую кнопку.

5.5. Настройка видеокамеры

Основное окно для работы с видеокамерой системы технического зрения представлено на рис. 5.9 [8]. В данном окне можно проводить как настойку камеры, так и управление роботом. Настройка нужна по следующим причинам. Система технического зрения данного робота (СТЗ) предполагает работу с объектами очень ограниченного формата, а именно с объектами круглой формы в плоскости. Объекты могут отличаться лишь размерами.

В этом случае на изображении сцены на выходе видеокамеры необходимо контрастно выделить объекты в виде окружности (или круга). Для обеспечения точности управления роботом и упрощения алгоритма распознавания крайне желательно работать только с контуром изображения. Но выделение контура связано с дифференцированием изображения, что неизбежно уменьшает помехоустойчивость системы.

Для выхода из этого противоречия в данном стенде используется развитая система фильтрации, которая позволяет компенсировать различные помехи, возникающие в связи с конкретными внешними условиями: типы и параметры освещенности, визуальные особенности элементов сцены, расположение используемых деталей на сцене и т.п.

Рис. 5.9. Окно *Камера*

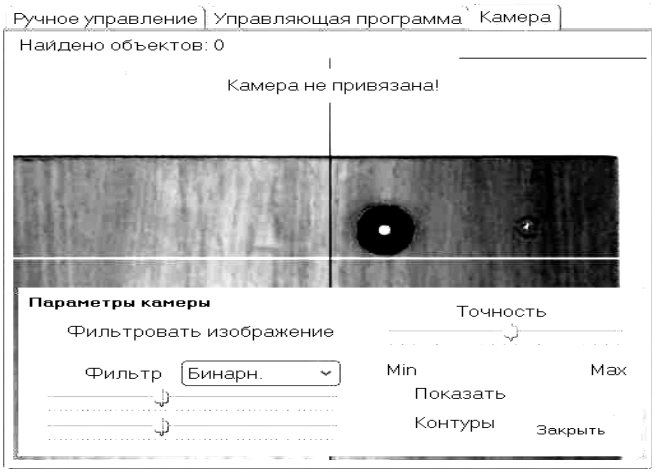


Рис. 5.10. Окно *Камера*, параметры

Рис. 5.11. Окно *Камера*, привязка камеры

5.5.1. Поэтапный алгоритм настройки камеры

Настройка камеры сводится к следующим операциям [8]:

- Привязка видеокамеры и робота к нулевым координатам. При этом учитывается, что оптическая ось камеры и центр схвата взаимно смещены.

- Оптическая настройка камеры с целью получения наиболее контрастного изображения при выделении контура объекта в конкретных условиях работы.
- Установка по заданию преподавателя на сцене объектов сборки.
- Подбор квазиоптимального алгоритма фильтрации.

Поэтапный алгоритм настройки:

1. Вывести робот в нулевое положение.

2. Убрать из зоны видения камеры все объекты за исключением черной метки (точка привязки камеры), как показано на рис. 5.9. Вверху окна будет располагаться надпись *Камера не привязана*.

3. Для привязки камеры к координатам сцены, необходимо нажать правую кнопку мыши и выбрать из всплывающего окна строку *Параметры камеры* (рис. 5.10).

4. Далее необходимо убрать галочку *Фильтровать изображение* и так отрегулировать поле *Точность*, чтобы черная базовая метка распозналась, т.е. выделился контур **красной окружностью**. Настройка камеры и подбор фильтрации осуществляются исключительно вручную при настройке камеры. Информация о различных фильтрах и рекомендации по настройке камеры содержатся ниже.

5. Следующим шагом закрываем *Параметры камеры*. Нажатием правой кнопкой мыши выбираем строку *Привязка камеры* и нажимаем кнопку *Привязать камеру* (рис. 5.11).

6. После успешного завершения предыдущей операции, нажать правую кнопку мыши и выбрать строку *Параметры поиска объектов*. Здесь необходимо настроить следующие параметры в зависимости от собираемых объектов, например:

– минимальный диаметр объекта 100 мм,

– максимальный диаметр объекта 290 мм.

Соотношение сечений 30%.

7. Закрыть окно *Параметры поиска объекта*, нажать правую кнопку мыши и выбрать строку *Командная строка*. Появится командная строка, которая позволяет управлять роботом при помощи команд.

5.5.2. Настройки фильтров

Фильтры используются как способ повышения точности, контрастности изображения между объектом и фоном. Тот или иной фильтр подбирается в зависимости от освещенности зоны съемки, наличия неодно-

родности фона, на котором расположены объекты и т.д. Окно настройки фильтров представлено на рис. 5.12 [8].

Основным критерием выбора фильтров является получение максимально контрастного перехода цвета между фоном и деталью.

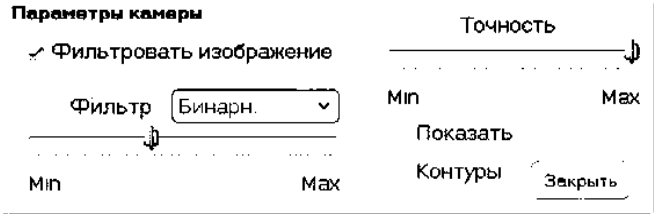


Рис. 5.12. Окно фильтров

Флаг – ***Фильтровать изображение***. Если флаг установлен, то данные с камеры будут обрабатываться выбранным фильтром изображения. Фильтры выбираются из выпадающего списка ***Фильтр***.

Степень фильтрации управляется горизонтальным бегунком, расположенным под списком *Фильтр*.

Установка флага ***Показать*** позволяет увидеть обработанное фильтром изображение. Это тот вид, который видит компьютер.

Установка флага ***Контуры*** позволяет увидеть все найденные системой контуры, даже те, которые не подходят по критериям поиска объектов.

Бегунок ***Точность*** определяет, насколько точно будут определены границы найденного контура, который отвечает критериям поиска объектов. В некоторых случаях точность необходимо загрубить, в каких-то, наоборот, повысить.

Подбор тех или иных параметров осуществляется эмпирически, в зависимости от конкретных внешних условий.

5.6. Программирование стенда

5.6.1. Основные команды стенда

В работе используются следующие команды робота.

Выход в исходное положение

RHOME – Координаты X, Y, Z будут возвращены в исходное положение. Схват будет разжат.

Перемещение робота задается функциями **G00** и **G01**. Формат команды:

G01 X… Y… Z… W… Значение X,Y, Z задается в градусах. Значение W – в миллиметрах. Если указаны какие-либо из параметров X, Y, Z – выполняется одновременное перемещение по указанным координатам в заданную точку со скоростью **F** (если указана). Если параметр F не указан, то перемещение будет выполнено с заданной ранее скоростью.

Например: G01 X10.5 Z15.6 F100

Если в команде G01 указан параметр W, то параметры X, Y, Z игнорируются и выполняется движение только по координате W.

Например: G01 W45.5

CAMPOINT X… Y… F100 – Перемещение камеры в зону расположения элементов сборки. Команда перемещает вертикальную ось камеры в заданную область. Перемещение осуществляется в нулевом положении по оси Z (необходимо для корректного процесса распознавания деталей).

GETOBJECTS F100 – Команда вызова процесса распознавания элементов сборки. Результатом команды будет являться определение объектов и присвоение им порядковых номеров по возрастанию размеров.

GOTOOBJECT P… F100 – Вывод вертикальной оси схвата над объектом.

Например: GOTOOBGECT P2 F100

POINT3D X… Y… Z F100 – Перемещение робота по соответствующим осям. При использовании данной команды необходимо помнить, что начало координат находится на пересечении поворотной оси робота и плоскости стола.

DELAY P… – Технологическая пауза. Параметр P – количество секунд паузы.

Например: DELAY P5.

Команды управления Схватом:

LOCKERON P… – Сжатие схвата на заданное количество шагов. Если параметр P не задан – будет использовано значение, указанное на вкладке «Настройки».

Пример: LOCKERON P500

LOCKEROFF – Разжатие схвата на заданное количество шагов. Если параметр P не задан – будет использовано значение, указанное на вкладке «Настройки».

Пример: LOCKEROFF P500

5.6.2. Пример сборочной операции

Задание: выполняя программирование и наладку стенда необходимо обеспечить сборку узла из трех деталей. На рис. 5.13 показан момент подхода схвата к одной из деталей.

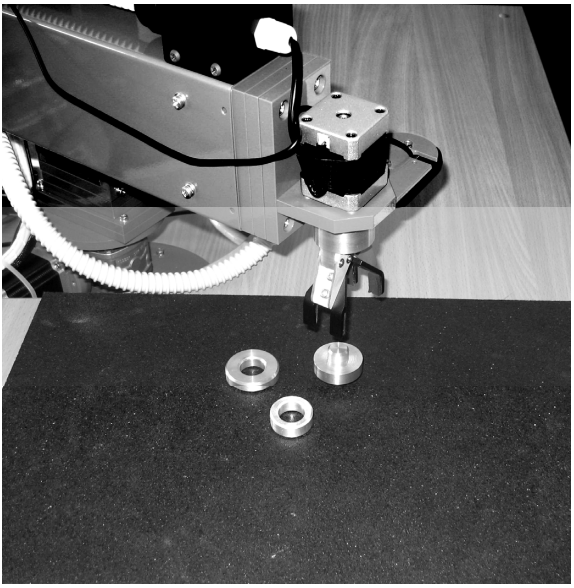


Рис. 5.13. Положение схвата робота перед захватом детали

Последовательность действий робота:

1. Вывод робота в ноль.
2. Настройка видеокамеры.
3. Привязка камеры.
4. Выполнение УП:
   - захват и перемещение детали № 2 в позицию сборки (рис. 5.13),
   - захват и перемещение детали № 1 в позицию сборки,
   - возврат робота в нуль.

5.7. Контрольные вопросы

1. Какие движения могут выполнять звенья робота?
2. Для чего и как производится *выход в ноль*?

3. Тип двигателей приводов робота?
4. Как ограничиваются перемещения звеньев робота?
5. Понятие о техническом зрении?
6. Какие задачи решает система управления?
7. Вид системы управления?
8. Количество управляемых координат?
9. Количество одновременно управляемых координат.
10. Как программируются движения робота.
11. В чем суть алгоритма распознавания изображения, используемого в стенде?
12. Достоинства и недостатки данного метода распознавания?
13. Что такое *бинарная фильтрация*?
14. Чем должны отличаться детали для того, чтобы робот мог их классифицировать?
15. Чем определяется порядок сборки изделия в данном стенде?
16. Для чего нужна система фильтрации изображения?
17. Какие действия должен выполнять наладчик для осуществления сборки по заданной УП?
18. Какие действия должен выполнять оператор?
19. Какие исходные данные нужны для разработки УП?

Приложение 5.1

Пример управляющей программы для робота с техническим зрением.

RHOME	вывод робота в нулевое положение
CAMPOINT X300 Y200 F100	вывод в зону съемки
DELAY P5	технологическая пауза
GETOBJECTS	распознавание объектов
GOTOOBJECT P2 F100	перемещение схвата к объекту № 2
POINT3D Z160 F100	опускание схвата над объектом
LOCKERON	захват объекта
POINT3D Z200 F100	перемещение объекта на безопасную высоту
GOTOOBJECT P3 F100	перемещение схвата к объекту № 3
POIN3D Z165 F100	опускание, сборка объекта № 2 и № 3
LOCKEROFF	разжатие схвата
POINT3D Z200 F100	подъем схвата на безопасную высоту
GOTOOBJECT P1 F100	перемещение схвата к объекту № 1
POINT3D Z200	опускание схвата над объектом
LOCKERON	захват объекта
GOTOOBJECT P3 F100	перемещение схвата к объекту № 3
POINT3D Z175 F100	опускание, сборка объекта № 1 и № 2
LOCKEROFF	разжатие схвата
RHOME	вывод робота в нулевое положение

6. ИЗУЧЕНИЕ АЛГОРИТМОВ РАСПОЗНАВАНИЯ В СИСТЕМЕ ТЕХНИЧЕСКОГО ЗРЕНИЯ РОБОТА

Лабораторная работа № 6 (6 часов)

6.1. Цели работы

Целью данной лабораторной работы является:

1. Получение навыков самостоятельной доработки методики эксперимента.
2. Анализ особенностей алгоритма распознавания сцены системой технического зрения робота [8].
3. Результаты анализа должны быть получены на основе проведенных экспериментальных исследований.

6.2. Задания к лабораторной работе

1. Данная работа может быть выполнена только после выполнения работы № 5 «Изучение лабораторного стенда системы технического зрения робота».
2. Для выполнения исследований набрать и отладить учебную технологическую программу.
3. Провести эксперименты № 1–3.
4. На основе результатов экспериментов сделать следующие выводы.

- Определить рабочую зону робота, в которой он может распознавать и работать с деталями.
- С какой точностью схват выходит к заданной детали? Оценить среднее значение ошибки и дисперсию.
- Зависит ли точность выхода схвата к детали от ее расположения на сцене?
- Как система технического зрения (СТЗ) различает похожие детали?
- По каким параметрам и насколько количественно должны отличаться детали, чтобы СТЗ воспринимала их, как детали разного класса?
- Как система выбирает конкретные детали для сборки одного изделия из нескольких комплектов исходных деталей?
- Как ведет себя СТЗ, если ей предложено собрать два одинаковых изделия из двух одинаковых комплектов? Почему именно так?

6.3. Методика экспериментов

6.3.1. Эксперимент № 1. Определение рабочей зоны робота

Предполагается, что рабочее пространство имеет форму круга. Чтобы убедиться в этом:

- Выводим схват робота в точку, в которой происходит запоминание сцены, и в центре рабочего поля камеры ставим деталь минимальных размеров;
- Располагаем диаметрально (по горизонтали) две других одинаковых детали на границе предполагаемой рабочей зоны;
- Проверяем, может ли СТЗ распознать в этих точках детали;
- Проверяем, может ли робот взять в этих точках детали.

Если робот не может взять детали, то рабочая зона меньше предполагаемой и ее надо сузить, и провести эксперимент заново. В противном случае зону надо расширить. Когда граница уверенной работы робота найдена, следует замерить на рабочей поверхности координаты точек для дальнейшего построения границы реальной зоны.

Далее эксперимент повторить по вертикальному диаметру и, желательно, под углами, кратными 45°.

В отчете отразить:

- Особенности методики, проведенного Вами эксперимента;
- Статистически оценить точность определения границ;
- Оценить, совпадают ли границы области распознавания детали СТЗ и границы рабочей зоны робота.

6.3.2. Эксперимент № 2. Анализ потенциальной точности сборки

Точность сборки изделия зависит от точности определения координат детали на сцене и точности подвода схвата к детали. В начале эксперимента требуется статистически оценить точность подвода схвата к детали, расположенной в различных местах сцены.

Для этого помещаем одну и ту же деталь в пяти точках, расположенных по горизонтальному диаметру рабочей зоны и в четыре точках – по вертикальному. Для каждой выбранной точки находим среднее значение ошибки подвода схвата и дисперсию ошибки. Величину ошибки можно определить как разность координат схвата и координат центра детали.

Для получения статистически значимых оценок требуется проводить не менее **десяти** измерений в каждой точке.

Учтите, что эти измерения можно сделать без дополнительного инструмента, только используя информацию, снимаемую с робота.

Требуется выяснить:

- Зависит ли точность подвода схвата к детали от ее положения на сцене?
- Зависит ли точность подвода схвата к детали от ее размера?
- Зависит ли точность подвода схвата к детали от высоты положения камеры при запоминании сцены?
- Если при запоминании сцены высоту положения камеры уменьшить в два раза, как изменится точность подвода схвата к детали в зависимости от ее положения? Если изменится, то почему?

В отчете отразить:

- Особенности методики, проведенного Вами эксперимента.
- Результаты точности определения положения детали на сцене.
- Результаты точности подвода схвата к детали.
- Соответствующие графики.

6.3.3. Эксперимент № 3. Анализ возможности классификации деталей СТЗ робота

Предполагается, что имеется два идентичных (с точностью до изготовления) комплекта деталей. Необходимо выяснить:

- Может ли робот различить две одинаковые детали, если они лежат рядом?
- Может ли робот различить две одинаковые детали, если они лежат далеко друг от друга?
- В какой последовательности он будет брать детали для построения двух изделий?
- При многократном запуске ТП изменится ли порядок использования деталей при сборке двух изделий?
- Чем определяются его действия?

Методику эксперимента разработайте самостоятельно.

В отчете отразить:

- Особенности методики, проведенного Вами эксперимента.
- Ваши ответы на поставленные вопросы, подкрепленные результатами проведенного эксперимента.

7. НАХОЖДЕНИЕ ОПТИМАЛЬНОЙ ТРАЕКТОРИИ ДВИЖЕНИЯ РОБОТА ПРИ ОБХОДЕ ЗАДАННОГО НАБОРА КОНТРОЛЬНЫХ ТОЧЕК

Лабораторная работа № 7 (6 часов)

7.1. Цели работы

1. Ознакомиться с методами оптимизации.
2. Изучить некоторые методы линейного и дискретного программирования.
3. Научиться применять методы дискретного программирования для выбора оптимальной траектории движения робота в заданных условиях.

7.2. Задания к лабораторной работе

1. Ознакомиться с теоретическим материалом. Подробно изучить алгоритм нахождения кратчайшего пути методом ветвей и границ.
2. Ответить на контрольные вопросы.
3. Научиться работать с программой ***Course***.
4. Решить задачу аналитически (методом ветвей и границ) для одного из вариантов задания (указанного преподавателем) и сравнить своё решение с программным.
5. В отчете о проделанной работе отразить:
   - Что такое «оптимизация»?
   - Что такое линейное, нелинейное, дискретное программирование?
   - В чем суть метода ветвей и границ?
   - Описать методику решения Вашей задачи.
   - Описать детально процесс решения поставленной задачи и полученные результаты.
   - Оценить, является ли Ваше решение единственным оптимальным.
   - Оценить время решения Вашей задачи компьютером при использовании разных методов оптимизации.
   - Измените длину одного ребра на 1, 5, 10, 30 единиц и проанализируйте полученные результаты. Обратите внимание, что и как изменяется в решении задачи?

7.3. Элементы теории оптимизации

При управлении транспортным роботом возникает ряд специфических задач:

- управление по заданной траектории с заданной точностью;
- организация прохождения траектории за минимальное время или с минимальными затратами ресурсов;
- обход всех контрольных точек по произвольной траектории за минимальное время;
- нахождение минимального пути при обходе всех контрольных точек при заданной сетке дорог.

Все эти задачи связаны с **оптимизацией управления по некоторому критерию**.

7.3.1. Критерий качества управления

В отличие от многих других математических задач задача управления имеет ту особенность, что допускает не одно, а множество различных решений [9, 10]. Как правило, имеется множество способов организации какого-либо процесса, приводящие к достижению поставленной цели.

Если имеется множество решений какой-либо задачи, то следует вести речь о выборе такого решения, которое с какой-либо точки зрения является наилучшим. Можно привести много примеров подобных задач. Из одного города в другой можно приехать, пользуясь различными видами транспорта: воздушным, водным, автобусным, автомобильным. Решением задачи будет выбор наиболее выгодного вида транспорта с точки зрения времени проезда, стоимости, удобств и т.п.

В тех случаях, когда цель управления может быть достигнута несколькими различными способами, на способ управления можно наложить добавочные требования, степень выполнения которых может служить основанием для выбора способа управления.

Во многих случаях реализация процесса управления требует затрат каких-либо ресурсов: времени, материалов, топлива, электроэнергии. Следовательно, при выборе способа управления следует говорить не только о том, достигается ли поставленная цель, но и о том, какие ресурсы придется затратить для ее достижения. В этом случае задача управления состоит в том, чтобы из множества решений, обеспечивающих достижение цели, выбрать одно, которое требует наименьшей затраты ресурсов.

Математическое выражение, дающее количественную оценку степени выполнения наложенных на способ управления требований, называют ***критерием качества управления***.

Наиболее предпочтительным или ***оптимальным*** способом управления будет такой, при котором критерий качества управления достигает минимального (максимального) значения. При выборе, например, режима движения робота за критерий качества управления можно принять расход энергии или затраченное время на достижение цели.

7.3.2. Ограничения, накладываемые на процесс управления

Задачу нахождения оптимального управления роботом, следует считать несуществующей, если на характер движения системы не наложено никаких ограничений (например, нет ограничений на мощность двигателя).

В общем случае имеются два вида ограничений на выбор способа управления [9, 10]. Ограничениями первого вида являются сами законы внешней среды, в соответствии с которыми происходит движение управляемой системы. При математической формулировке задачи управления эти ограничения представляются обычно *алгебраическими, дифференциальными или разностными уравнениями* связи.

Второй вид ограничений вызван ограниченностью ресурсов, используемых при управлении. Математически ограничения этого вида выражаются обычно в виде *систем алгебраических уравнений или неравенств*, связывающих переменные, описывающие состояние системы.

7.3.3. Постановка задачи оптимизации

Задачу оптимального управления можно считать сформулированной математически, если:

- сформулирована цель управления, выраженная через критерий качества управления,
- определены ограничения первого вида, представляющие собой системы дифференциальных или разностных уравнений, ограничивающих возможные способы движения системы,
- определены ограничения второго вида, представляющие сбой систему алгебраических уравнений или неравенств, выражающих ограниченность ресурсов или иных величин, пользуемых при управлении.

Способ управления, который удовлетворяет всем поставленным ограничениям и обращает в минимум (максимум) критерий качества управления, называют ***оптимальным управлением***.

Задачи, связанные с выбором оптимального маршрута для робота среди некоторых контрольных точек, могут решаться методами ***линейного или дискретного программирования***.

7.4. Нахождение оптимального пути транспортного робота

7.4.1. Методы решения задачи коммивояжёра

Круг задач, связанных с нахождением кратчайшего пути при обходе заданных контрольных точек на местности с последующим возвратом в исходную точку, известен, как **задача коммивояжера** [11]. Для решения этих задач робот должен обладать ***искусственным интеллектом***.

В 1859 г. ирландский математик У. Гамильтон придумал игру «Кругосветное путешествие» по додекаэдру, узловые вершины которого символизировали крупнейшие города Земли, а ребра – соединяющие их дороги [12]. Задача состоит в отыскании такого пути, проходящего через все вершины (города, пункты назначения) графа, чтобы посетить каждую вершину однократно и возвратиться в исходную. Пути, обладающие таким свойством, называются ***гамильтоновыми циклами***.

Задачи коммивояжера решаются посредством различных методов, выведенных в результате теоретических исследований. Однако они требуют значительных вычислительных ресурсов (особенно – времени) при большой размерности пространства допустимых решений.

Все эффективные методы являются эвристическими. В большинстве таких методов находится не самый эффективный маршрут, а субоптимальное решение. Зачастую востребованы итерационные алгоритмы.

Выделяют следующие группы методов решения задач коммивояжера, которые относятся к простейшим:

Полный перебор (или метод «грубой силы») – метод решения задачи путем перебора всех возможных вариантов. Сложность полного перебора зависит от количества всех возможных решений задачи. Если пространство решений очень велико, то полный перебор может не дать результатов в течение нескольких лет.

Случайный перебор. Обычно выбор решения можно представить последовательностью выборов. Если делать эти выборы с помощью како-

го-либо случайного механизма, то решение находится очень просто, но не быстро, так что можно находить решение многократно и запоминать «рекорд», т.е. наилучшее из встретившихся решений. Этот наивный подход существенно улучшается, когда удается учесть в случайном механизме перспективность тех или иных выборов, т.е. комбинировать случайный поиск с эвристическим методом и методом локального поиска. Такие методы применяются, например, при составлении расписаний для Аэрофлота [11].

Жадный алгоритм – алгоритм нахождения наикратчайшего расстояния путём выбора самого короткого, ещё не выбранного ребра, при условии, что оно не образует цикла с уже выбранными рёбрами. При решении задачи коммивояжера жадный алгоритм превратится в стратегию «иди в ближайший (еще не посещенный) город». Жадный алгоритм, очевидно, бессилен в задаче коммивояжера. Рассмотрим для примера сеть (рис. 7.1), представляющую узкий ромб.

Коммивояжер стартует из города 1. Алгоритм «иди в ближайший город» выведет его в город 2, затем 3, далее 4; на последнем шаге придется платить за жадность, возвращаясь по длинной диагонали ромба. В результате получится не кратчайший, а длиннейший тур.

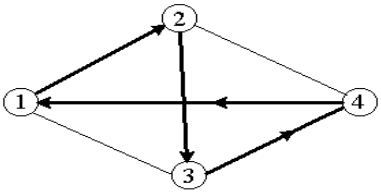


Рис. 7.1. Жадный алгоритм

В основе алгоритма лежит утверждение: «Если справедливо неравенство треугольника, то для каждой цепи верно, что расстояние от начала до конца цепи меньше (или равно) суммарной длины всех ребер цепи». Это обобщение расхожего убеждения, что прямая короче кривой.

Деревянный алгоритм для решения задачи коммивояжера будет следующим: на входной сети задачи коммивояжера строится кратчайшее остовное дерево и удваиваются все его ребра. В результате получаем граф, связный с вершинами, имеющими только четные степени. Затем строится Эйлеров цикл, начиная с вершины 1, цикл задается перечнем вершин. Просматривается перечень вершин, начиная с 1, и зачеркивается каждая вершина, которая повторяет уже встреченную в последовательности. Останется тур, который и является результатом алгоритма.

Доказано, что деревянный алгоритм ошибается менее чем в два раза, поэтому такие алгоритмы называют приблизительными, а не просто эвристическими.

Метод имитации отжига. Экзотическое название данного алгоритма связано с методами имитационного моделирования в статистической физике, основанными на технике Монте-Карло. Исследование кристаллической решетки и поведения атомов при медленном остывании тела привело к появлению на свет вероятностных алгоритмов, которые оказались чрезвычайно эффективными в комбинаторной оптимизации. Впервые это было замечено в 1983 году. Сегодня этот алгоритм является популярным как среди практиков благодаря своей простоте, гибкости и эффективности, так и среди теоретиков, поскольку для данного алгоритма удается аналитически исследовать его свойства и доказать асимптотическую сходимость.

Алгоритм имитации отжига относится к классу пороговых алгоритмов локального поиска. На каждом шаге этого алгоритма для текущего решения i_k в его окрестности $N(i_k)$ выбирается некоторое решение j и, если разность по целевой функции между новым и текущим решением не превосходит заданного порога t_k, то новое решение j заменяет текущее. В противном случае выбирается новое соседнее решение [11].

На практике применяются различные модификации более эффективных методов:

В основе **метода ветвей и границ** лежит идея последовательного разбиения множества допустимых решений на подмножества [12]. На каждом шаге метода элементы разбиения подвергаются проверке для выяснения, содержит данное подмножество оптимальное решение или нет. Проверка осуществляется посредством вычисления оценки снизу для целевой функции на данном подмножестве. Если оценка снизу не меньше *рекорда* – наилучшего из найденных решений, то подмножество может быть отброшено. Проверяемое подмножество может быть отброшено еще и в том случае, когда в нем удается найти наилучшее решение. Если значение целевой функции на найденном решении меньше рекорда, то происходит смена рекорда. По окончанию работы алгоритма рекорд является результатом его работы.

Если удается отбросить все элементы разбиения, то рекорд – оптимальное решение задачи. В противном случае, из не отброшенных подмножеств выбирается наиболее перспективное (например, с наименьшим значением нижней оценки), и оно подвергается разбиению. Новые подмножества вновь подвергаются проверке и т.д.

Генетический алгоритм – это эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, напоминающих биологическую эволюцию.

Генетические алгоритмы служат, главным образом, для поиска решений в многомерных пространствах поиска.

Алгоритмы муравья, или оптимизация по принципу муравьиной колонии (название было придумано изобретателем алгоритма, Марко Дориго), основаны на применении специфических свойств, присущим муравьям, и используют их для ориентации в физическом пространстве. Алгоритмы муравья особенно интересны потому, что их можно использовать для решения не только статичных, но и динамических проблем, например, в изменяющихся сетях [12].

7.4.2. Математическая постановка задачи

Имеется *n* городов с номерами *1,2,…n*, для каждой пары городов *i* и *j* задано расстояние *c[i, j]* между ними. Выезжая из города *1*, коммивояжер должен побывать во всех остальных городах по одному разу и вернуться в исходный город *1*. Определить, в каком порядке следует объезжать города, чтобы суммарное пройденное расстояние было наименьшим.

Пусть $1, p_1, p_2, \ldots, p_{n-1}, 1$ – номера городов, записанные в порядке их обхода. То есть p_i – номер города, посещаемого на *i*-м шаге, $i = 0, \ldots, n$, $p_0 = p_n = 1$. Тогда пройденное расстояние равно:

$$\sum_{j=0}^{n-1} c[p_j, p_{j+1}] \longrightarrow \min .$$

Среди чисел $p_1, p_2, \ldots, p_{n-1}$ ровно по одному разу встречается каждое число из интервала *2…n*. Таким образом, в задаче ищется перестановка целых чисел от *2* до *n*, доставляющая минимум целевой функции.

Пример задачи. Пусть n=5, матрица расстояний между городами:

$$C = \begin{pmatrix} 0 & 5 & 3 & 1 & 2 \\ 5 & 0 & 1 & 3 & 6 \\ 3 & 1 & 0 & 4 & 2 \\ 1 & 3 & 4 & 0 & 7 \\ 2 & 6 & 2 & 7 & 0 \end{pmatrix}$$

Оптимальный маршрут выглядит так: $1 \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 1$.

Пройденное коммивояжером расстояние равно 9.

7.5. Термины и определения

Сформулируем задачу коммивояжера для метода ветвей и границ: ***Имеется несколько городов, соединенных некоторым образом дорогами с известной длиной; требуется установить, имеется ли путь, двигаясь по которому можно побывать в каждом городе только один раз и при этом вернуться в город, откуда путь был начат («обход коммивояжера»), и, если таковой путь имеется, установить кратчайший из таких путей.***

Формализуем условие в терминах теории графов. Города будут вершинами графа, а дороги между городами – ребрами графа, на каждом из которых задана весовая функция: ***вес ребра*** – это длина соответствующей дороги. В общем случае граф может быть ориентированным, т.е. *вес* выбранной дороги может зависеть от направления, например, велосипедист едет в гору или с горы.

Путь, который требуется найти, это – ориентированный остовной простой цикл минимального веса в орграфе; такие циклы называются также ***гамильтоновыми.***

Некоторые определения:

- Цикл называется ***остовным***, если он проходит по всем вершинам графа.
- Цикл называется ***простым***, если он проходит по каждой своей вершине только один раз.
- Цикл называется ***ориентированным***, если начало каждого последующего ребра совпадает с концом предыдущего.
- ***Вес*** цикла – это сумма весов его ребер.
- Орграф называется ***полным***, если в нем имеются все возможные ребра.

Введем некоторые термины. Пусть имеется некоторая числовая матрица. ***Привести строку этой матрицы*** означает выделить в строке минимальный элемент (его называют ***константой приведения***) и вычесть его из всех элементов этой строки. Очевидно, в результате в этой строке на месте минимального элемента окажется ноль, а все остальные элементы будут неотрицательными. Аналогичный смысл имеют слова ***привести столбец*** матрицы.

Слова «привести матрицу по строкам» означают, что все строки матрицы приводятся. Аналогичный смысл имеют слова ***привести матрицу по столбцам.***

Наконец, слова ***привести матрицу*** означают, что матрица сначала приводится по строкам, а потом приводится по столбцам.

Весом элемента матрицы называют сумму констант приведения матрицы, которая получается из данной матрицы заменой обсуждаемого элемента на ∞. Следовательно, слова «самый тяжелый нуль в матрице» означают, что в матрице подсчитан вес каждого нуля, а затем фиксирован нуль с максимальным весом. Самый тяжелый нуль определяется суммированием наименьшей строки и столбца.

Приступим теперь к описанию метода ветвей и границ для решения задачи о коммивояжере.

Метод ветвей и границ – это один из методов организации полного перебора. Он применим не всегда, а только тогда, когда выполняются специфические дополнительные условия:

а) Расстояния между пунктами назначения должны быть неотрицательными;

б) Запрет на петли в туре, т.е. не должно быть внутренних циклов (исключение: вернуться в исходную точку).

Данный алгоритм используется для поиска оптимального гамильтонова контура в графе, имеющем N вершин, причем каждая вершина i связана с любой другой вершиной j двунаправленной дугой. Каждой дуге приписан вес $C_{i,j}$, причем веса дуг строго положительны ($C_{i,j} \geq 0$). Веса дуг образуют матрицу стоимости. Все элементы по диагонали матрицы приравнивают к бесконечности ($C_{j,j} = \infty$).

В случае, если пара вершин i и j не связана между собой (граф не полносвязный), то соответствующему элементу матрицы стоимости приписываем вес, равный длине минимального пути между вершинами i и j. Если в итоге дуга (i, j) войдет в результирующий контур, то ее необходимо заменить соответствующим ей путем.

Общая идея тривиальна: нужно разделить огромное число перебираемых вариантов на классы и получить оценки (снизу – в задаче минимизации, сверху – в задаче максимизации) для этих классов, чтобы иметь возможность отбрасывать варианты не по одному, а целыми классами. Трудность состоит в том, чтобы найти такое разделение на классы (ветви) и такие оценки (границы), чтобы процедура была эффективной.

7.6. Алгоритм выполнения работы

1. В каждой строке исходной матрицы стоимости найдем минимальный элемент и вычтем его из всех элементов строки. Сделаем это и для столбцов, не содержащих нуля. Получим матрицу стоимости, каждая

строка и каждый столбец которой содержат хотя бы один нулевой элемент.

2. Для каждого нулевого элемента матрицы *Cij* рассчитаем коэффициент $\Gamma_{i,j}$, который равен сумме наименьшего элемента *i* строки (исключая элемент $C_{i,j} = 0$) и наименьшего элемента *j* столбца. Из всех коэффициентов $\Gamma_{i,j}$ выберем такой, который является максимальным

$$\Gamma_{k,l} = max\{\Gamma_{i,j}\}.$$

В гамильтонов контур вносится соответствующая дуга (*k,l*).

3. Удаляем *k*-тую строку и столбец *l*, поменяем на бесконечность значение элемента $C_{l,k}$ (поскольку дуга (*k,l*) включена в контур, то обратный путь из вершины *l* в *k* недопустим).

4. Повторяем алгоритм шага 1, пока порядок матрицы не станет равным двум.

5. Затем в текущий ориентированный граф вносим две недостающие дуги, определяющиеся однозначно матрицей прядка 2. Получаем гамильтонов контур.

В ходе решения ведется постоянный подсчет текущего значения **нижней границы**. Нижняя граница равна сумме всех вычтенных элементов в строках и столбцах. Итоговое значение нижней границы должно совпасть с длиной результирующего контура.

Рассмотрим конкретный пример реализации метода ветвей и границ.

Итак, требуется найти **легчайший простой основный ориентированный цикл в полном взвешенном ориентированном графе на пяти вершинах со следующей весовой матрицей:**

	1	2	3	4	5
1	∞	9	8	4	10
2	6	∞	4	5	7
3	5	3	∞	6	2
4	1	7	2	∞	8
5	2	4	5	2	∞

Верхняя строка и левый столбец, выделенные затемненным фоном, содержат номера вершин графа; символ ∞ (компьютерная *бесконечность*), стоящий на главной диагонали, означает отсутствие ребер-петель выбранной вершины.

Подсчитаем φ(I) в нашем примере.

	1	2	3	4	5		
1	∞	9	8	4	1	4	← min в строке 1
2	6	∞	4	5	7	4	← min в строке 2
3	5	3	∞	6	2	2	← min в строке 3
4	1	7	2	∞	8	1	← min в строке 4
5	2	4	5	2	∞	2	← min в строке 5

Результат приведения по строкам:

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

Получим константы приведения и выделим минимумы по столбцам:

	1	2	3	4	5
1	∞	5	4	0	6
2	2	∞	0	1	3
3	3	1	∞	4	0
4	0	6	1	∞	7
5	0	2	3	0	∞
	min в столбце 1 0	min в столбце 2 1	min в столбце 3 0	min в столбце 4 0	min в столбце 5 0

Результат приведения матрицы:

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

Сумма констант приведения φ(I) = 4+4+2+1+2+1=14.

Обозначим эту матрицу через $M1$; найдем в ней самый тяжелый нуль. В полученной матрице укажем рядом с каждым нулем в скобках его вес.

	1	2	3	4	5
1	∞	4	4	0(4)	6
2	2	∞	0(2)	1	3
3	3	0(1)	∞	4	0(3)
4	0(1)	5	1	∞	7
5	0(0)	1	3	0(0)	∞

Самым тяжелым оказывается нуль в клетке (1,4). Следовательно, множество Γ разбивается на $\Gamma_{\{(1,4)\}}$ (все циклы, проходящие через ребро (1,4)) и $\Gamma_{\overline{\{(1,4)\}}}$ (все циклы, не проходящие через ребро (1,4)).

Построим для множества $\Gamma_{\{(1,4)\}}$ соответствующую ему матрицу и значение оценочной функции.

Условимся о следующем действии: перед тем, как в очередной матрице вычеркнуть строку и столбец, в ней надо заменить на ∞ числа во всех тех клетках, которые соответствуют ребрам, заведомо не принадлежащим тем гамильтоновым циклам, которые проходят через уже отобранные ранее ребра.

Учитывая это напоминание, элемент с номером (4,1) заменим на ∞ и вычеркнем строку номер 1 и столбец номер 4:

	1	2	3	5
2	2	∞	0	3
3	3	0	∞	0
4	∞	5	1	7
5	0	1	3	∞

Приведем теперь эту матрицу:

	1	2	3	5
2	2	∞	0	3
3	3	0	∞	0
4	∞	4	0	6
5	0	1	3	∞

Это – матрица $M1,1$; сумма констант приведения здесь равна 1, поэтому $\varphi_{\{1,4\}} = 14+1= 15$. Для $M1,2$ заменяем на ∞ элемент (1,4) в $M1$:

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

После этого приводим полученную матрицу:

	1	2	3	4	5
1	∞	0	0	∞	2
2	2	∞	0	1	3
3	3	0	∞	4	0
4	0	5	1	∞	7
5	0	1	3	0	∞

Это – матрица $M1,2$; сумма констант последнего приведения равна 4, так что $\varphi_{\{\overline{1,4}\}} = 14+4=18$. Следовательно, дальнейшей разработке подвергается множество $\Gamma_{\{1,4\}}$. Вот веса нулей матрицы $M1,1$ (они указаны в скобках):

	1	2	3	5
2	2	∞	0(2)	3
3	3	0(1)	∞	0(3)
4	∞	4	0(4)	6
5	0(3)	1	3	∞

Самым тяжелым является нуль с номером (4,3), так что теперь следует рассматривать множества $\Gamma_{\{1,4\}\{4,3\}}$ и $\Gamma_{\{1,4\}\{\overline{4,3}\}}$.

Обратимся к первому из них.

Условимся о следующем действии: перед тем, как в очередной матрице вычеркнуть строку и столбец, в ней надо заменить на ∞ числа во всех тех клетках, которые соответствуют ребрам, заведомо не принадлежащим тем гамильтоновым циклам, которые проходят через уже отобранные ранее ребра.

Следовательно, клетки с номерами (4,2), (4,5) и (3,1) надо заполнить символом ∞; после этого строку номер 4 и столбец номер 3 следует вычеркнуть; получим:

	1	2	5
2	2	∞	3
3	∞	0	0
5	0	1	∞

Приведение этой матрицы:

	1	2	5
2	0	∞	1
3	∞	0	0
5	0	1	∞

Для оценочной функции: $\varphi_{\{1,4\}\{4,3\}}$=15+2=17.

Матрица для множества $\Gamma_{\{1,4\}\{\overline{4,3}\}}$:

	1	2	3	5
2	2	∞	0	3
3	3	0	∞	0
4	∞	4	∞	6
5	0	1	3	∞

Результат ее приведения:

	1	2	3	5
2	2	∞	0	3
3	3	0	∞	0
4	∞	0	∞	2
5	0	1	3	∞

Оценочная функция: $\varphi_{\{1,4\}\{\overline{4,3}\}}$=15+4=19. Следовательно, дальнейшей разработке подлежит $\Gamma_{\{1,4\}\{4,3\}}$. Поскольку, вычеркнув строку 4 и столбец 3 в матрице $M1,1$, нужно также заменить на ∞ числа в определённых клетках так, чтобы не получалось коротких циклов (длиной меньше N), то в клетке с номером (3,1) надо поставить символ ∞. «Взвешиваем» нули:

	1	2	5
2	0(1)	∞	1
3	∞	0(1)	0(1)
5	0(1)	1	∞

Выбираем любую из соответствующих клеток; для определенности – клетку (2,1).

Теперь речь пойдет о множествах $\Gamma_{\{1,4\}\{4,3\}\{2,1\}}$ и $\Gamma_{\{1,4\}\{4,3\}\overline{\{2,1\}}}$.

Условимся о следующем действии: перед тем, как в очередной матрице вычеркнуть строку и столбец, в ней надо заменить на значение ∞ во всех тех клетках, которые соответствуют ребрам, заведомо не принадлежащим тем гамильтоновым циклам, которые проходят через уже отобранные ранее ребра.

Поэтому для первого множества положим в последней матрице элемент с номером (3,2) равным ∞, вычеркнем строку номер 2 и столбец номер 1:

	2	5
3	∞	0
5	1	∞

Приведем эту матрицу:

	2	5
3	∞	0
5	0	∞

Получаем для оценочной функции: $\varphi_{\{1,4\}\{4,3\}\{2,1\}}=17+1=18$.

Для множества $\Gamma_{\{1,4\}\{4,3\}\overline{\{2,1\}}}$ матрица такова:

	1	2	5
2	∞	∞	1
3	∞	0	0
5	0	1	∞

Приведение этой матрицы дает:

	1	2	5
2	∞	∞	0
3	∞	0	0
5	0	1	∞

Для оценочной функции: $\varphi_{\{1,4\}\{4,3\}\{\overline{2,1}\}}$ =17+1=18.

Получилось, что для дальнейшей разработки можно брать любое из множеств $\Gamma_{\{1,4\}\{4,3\}\{2,1\}}$ и $\Gamma_{\{1,4\}\{4,3\}\{\overline{2,1}\}}$. В первом случае уже получена матрица размером 2×2; ее нулевые клетки дают те ребра, которые с найденными ранее составляют обход коммивояжера, причем вес этого обхода равен значению оценочной функции – 18. Вот этот обход:

$$(1,4)(4,3)(2,1)(5,2)(3,5) \text{ или } 1\rightarrow4\rightarrow3\rightarrow5\rightarrow2\rightarrow1.$$

Найденный рекорд на самом деле является искомым оптимумом, потому что значения оценочной функции на всех оборванных ветвях (на границах) больше или равны весу рекорда.

При ином варианте выборов по ходу разбиений можно было получить другой оптимум: $1\rightarrow4\rightarrow3\rightarrow2\rightarrow5\rightarrow1$.

7.7. Описание модели

Главное окно программы представлено на рис. 7.2.

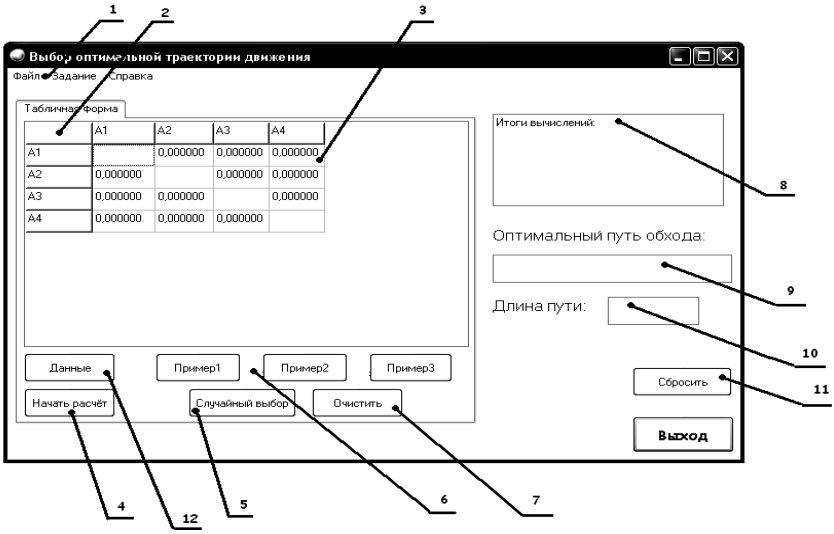


Рис. 7.2. Главное окно программы

На нем расположены следующие основные элементы.

1. **Основное меню программы**, которое включает в себя следующие пункты:

Меню **«Файл»** представлено на рис. 7.3.

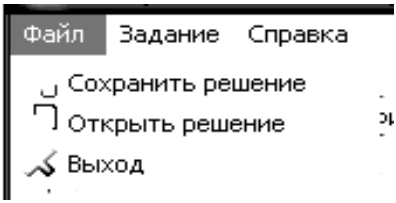


Рис. 7.3. Меню *Файл*

Открыть решение – открывает файл с ранее сохраненным решением.

Пункт главного меню **«Задание»** представлен на рис. 7.4.

Лабораторная работа – на этой форме отображается задание к лабораторной работе и алгоритм её выполнения;

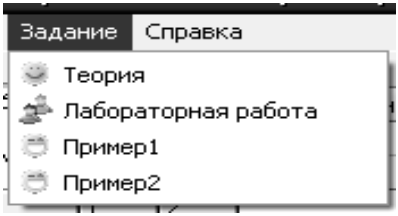


Рис. 7.4. Меню *Задание*

Пример1/Пример2/Пример3 – на рис. 7.5 показана таблица расстояний *Примера 1*, вызванная соответствующей кнопкой.

Меню **Справка** – выдает краткую информацию о моделирующей программе и ее разработчиках.

2. **Фиксированные ячейки** – отображают количество пунктов назначения, через которые необходимо пройти транспортному роботу.

3. **Табличная форма** – содержит матрицу расстояний между городами (в соответствии с формулировкой задачи коммивояжера).

4. Кнопка **Начать расчёт** запускает программу поиска оптимального решения и вывод его на экран.

5. Кнопка **Случайный выбор** генерирует случайные величины от 0 до 20 в матрице расстояний.

6. Кнопки **Пример1**, **Пример2**, **Пример3** – открывают 1-й, 2-й и 3-й варианты заданий для лабораторной работы, которые требуется решить. Здесь матрицы исходных данных фиксированы.

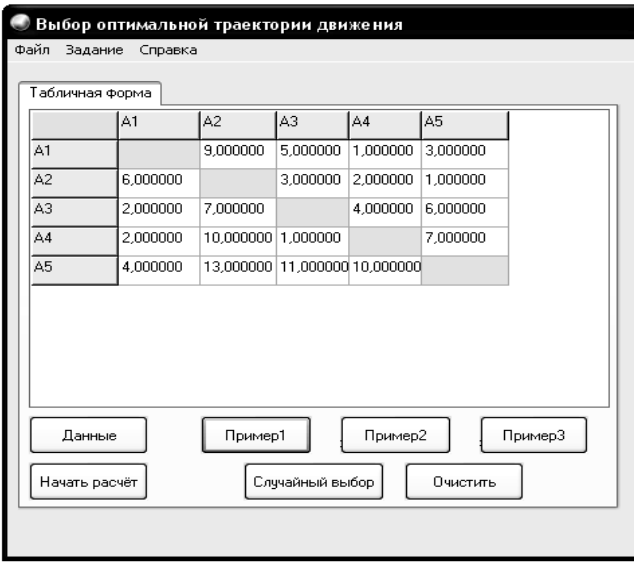


Рис. 7.5. Пример таблицы расстояний

7. Кнопка «Очистить» освобождает поле матрицы расстояний для ввода новых значений.

8. Этот элемент предназначен для отображения **итогов вычислений**. Он позволяет вводить многострочный текст с клавиатуры, загружать его из файла, редактировать и сохранять в файл текстового формата.

9, 10. Поля для вывода найденного решения: **оптимальный путь обхода** и **длину пути**.

11. Кнопка **Сбросить** очищает элемент (8) и предоставляет возможность проводить последующие вычисления.

12. Нажатие кнопки **Данные** выводит на экран окно, предназначенное для ввода исходных данных в таблицу расстояний (рис. 7.6).

Заполнение таблицы производится двумя способами:

- **Ввод вручную**: задается размер матрицы расстояний, т.е. изменяется количество пунктов назначения (рис. 7.6).

- **Ввод из файла:** в поле «**Файл данных**» вводится адрес ранее сохранённого файла.

Замечание: для записи матрицы расстояний при «Вводе вручную», необходимо производить двойной щелчок левой кнопкой мышки на каждой ячейке. В появившемся окне (рис. 7.7) прописываем необходимое значение

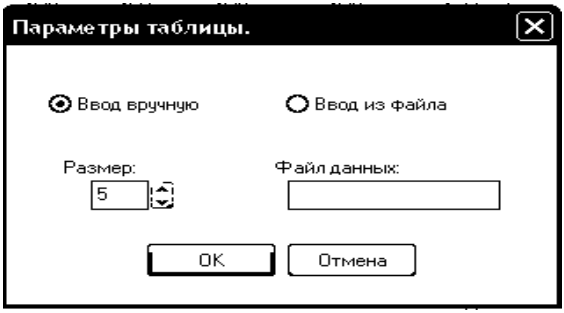


Рис. 7.6. Параметры таблицы

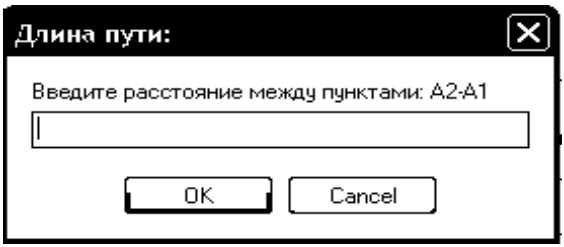
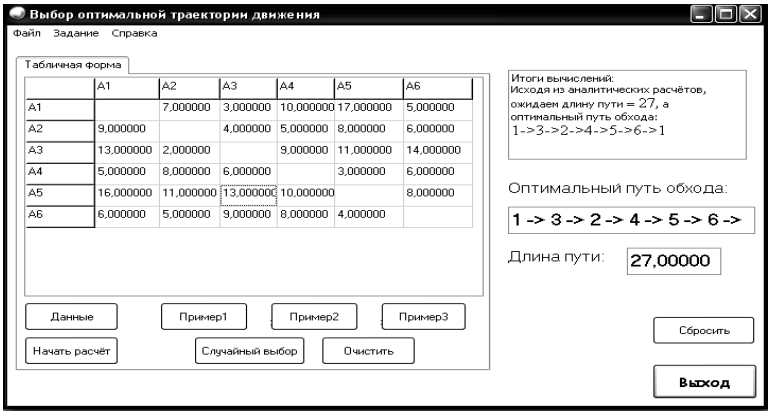


Рис. 7.7. Окно ввода данных

После ввода данных можно запустить программу на поиск оптимального решения. Найденное решение выдается в главное окно с комментариями (рис. 7.8).

Надо помнить, что метод ветвей и границ дает квазиоптимальное решение, т.е. не гарантирует нахождения единственного глобального экстремума. Кроме того, функция может быть многоэкстремальной и в зависимости от конкретной реализации данного алгоритма, могут быть найдены другие квазиоптимальные решения.

В гл. 7.9 приведены ряд заданий для самостоятельного решения **СТУДЕНТАМИ** задачи нахождения оптимального пути. Полученный ими результат можно проверить на моделирующей программе.



	A1	A2	A3	A4	A5	A6
A1		7,000000	3,000000	10,000000	17,000000	5,000000
A2	9,000000		4,000000	5,000000	8,000000	6,000000
A3	13,000000	2,000000		9,000000	11,000000	14,000000
A4	5,000000	8,000000	6,000000		3,000000	6,000000
A5	16,000000	11,000000	13,000000	10,000000		8,000000
A6	6,000000	5,000000	9,000000	8,000000	4,000000	

Рис. 7.8. Вывод оптимального решения

Надо помнить, что метод ветвей и границ дает квазиоптимальное решение, т.е. не гарантирует нахождения единственного глобального экстремума. Кроме того, функция может быть многоэкстремальной и в зависимости от конкретной реализации данного алгоритма, могут быть найдены другие квазиоптимальные решения.

7.8. Контрольные вопросы

1. Адаптивные системы управления. Целевая функция.
2. Системы программного управления. Целевая функция.
3. Содержательные постановки задач оптимизации управления транспортным роботом.
4. Критерии качества и ограничения, например, в системе управления транспортным роботом.
5. Классическая задача оптимизации.
6. Что такое *Оптимальная система*?
7. Что такое *Квазиоптимальная система*?
8. Что такое *Экстремальная система*?
9. Что такое *Глобальный экстремум*, *Локальный экстремум*?
10. Что такое *Линейное программирование*?
11. Что такое *Нелинейное программирование*?
12. Что такое *Дискретное программирование*?
13. Сформулируйте задачу коммивояжера.
14. В чем заключается метод вервей и границ?

15. Что содержит в себе матрица весов?
16. Что означает привести матрицу по столбцам/по строкам?
17. Как определить самый тяжелый нуль?
18. Почему *метод ветвей и границ* не гарантирует нахождение глобального экстремума?
19. Почему пути, найденные моделирующей системой и рассчитанный студентом, могут отличаться, но при этом критерий качества будет одинаковый?
20. В каких случаях нельзя применять метод ветвей и границ?

7.9. Варианты задания для проведения лабораторной работы

Задание: Найти оптимальный маршрут транспортного робота:

Вариант 1

0	*9*	*1*	*3*	*14*	**1**
1	*0*	*1*	*4*	*6*	**2**
4	*3*	*0*	*12*	*8*	**3**
2	*6*	*1*	*0*	*2*	**4**
5	*7*	*9*	*1*	*0*	**5**
1	**2**	**3**	**4**	**5**	

Вариант 2

0	*2*	*3*	*8*	*3*	**1**
2	*0*	*16*	*2*	*4*	**2**
9	*1*	*0*	*2*	*1*	**3**
5	*4*	*8*	*0*	*6*	**4**
7	*7*	*8*	*7*	*0*	**5**
1	**2**	**3**	**4**	**5**	

Вариант 3

0	*10*	*10*	*9*	*2*	**1**
34	*0*	*1*	*14*	*9*	**2**
6	*6*	*0*	*6*	*22*	**3**
7	*25*	*36*	*0*	*3*	**4**
31	*5*	*6*	*9*	*0*	**5**
1	**2**	**3**	**4**	**5**	

Вариант 4

0	*3*	*7*	*2*	*8*	**1**
10	*0*	*5*	*6*	*10*	**2**
6	*1*	*0*	*9*	*5*	**3**
2	*4*	*9*	*0*	*3*	**4**
4	*7*	*8*	*7*	*0*	**5**
1	**2**	**3**	**4**	**5**	

Вариант 5

0	*2*	*6*	*1*	*2*	**1**
8	*0*	*5*	*4*	*6*	**2**
1	*5*	*0*	*3*	*5*	**3**
8	*6*	*2*	*0*	*5*	**4**
7	*3*	*3*	*6*	*0*	**5**
1	**2**	**3**	**4**	**5**	

Вариант 6

0	*11*	*8*	*4*	*2*	**1**
8	*0*	*5*	*4*	*6*	**2**
10	*5*	*0*	*3*	*5*	**3**
9	*6*	*2*	*0*	*5*	**4**
12	*3*	*3*	*6*	*0*	**5**
1	**2**	**3**	**4**	**5**	

Вариант 7

0	*15*	*33*	*30*	*14*	**1**
47	*0*	*31*	*61*	*81*	**2**
48	*13*	*0*	*14*	*33*	**3**
66	*82*	*55*	*0*	*17*	**4**
74	*27*	*34*	*24*	*0*	**5**
1	**2**	**3**	**4**	**5**	

Вариант 8

0	*12*	*8*	*20*	*5*	**1**
11	*0*	*10*	*9*	*18*	**2**
17	*17*	*0*	*19*	*4*	**3**
3	*13*	*16*	*0*	*6*	**4**
16	*11*	*9*	*8*	*0*	**5**
1	**2**	**3**	**4**	**5**	

Вариант 9

0	*13*	*16*	*3*	*6*	**1**
8	*0*	*13*	*8*	*12*	**2**
15	*16*	*0*	*10*	*12*	**3**
6	*4*	*8*	*0*	*13*	**4**
3	*19*	*7*	*6*	*0*	**5**
1	**2**	**3**	**4**	**5**	

8. ИСПОЛЬЗОВАНИЕ ПАКЕТА SOLIDWORKS ДЛЯ МОДЕЛИРОВАНИЯ РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

Лабораторная работа № 8 (4 часа)

8.1. Цель работы

Цель – изучить конструкторские возможности программного пакета SolidWorks для моделирования роботов.

8.2. Задания к лабораторной работе

1. Изучить теоретический материал, изложенный в описании данной лабораторной работы.
2. Выполнить проектирование заданного преподавателем узла робота РБ-241 [1, 9]. Общий вид робота представлен на рис. 8.1, модель робота – на рис. 8.2, набор некоторых узлов – на рис. 8.3.

Рис. 8.1. Общий вид робота РБ-241

3. Запустить автоматизированную обучающую систему (АОС) по изучению конструкторских возможностей данного пакета и выполнить все задания, указанные в ней.
4. Ответить на все вопросы АОС.

5. Изучить и построить предложенную преподавателем деталь робота.
6. Ответить на контрольные вопросы.
7. Написать отчет о проделанной работе.

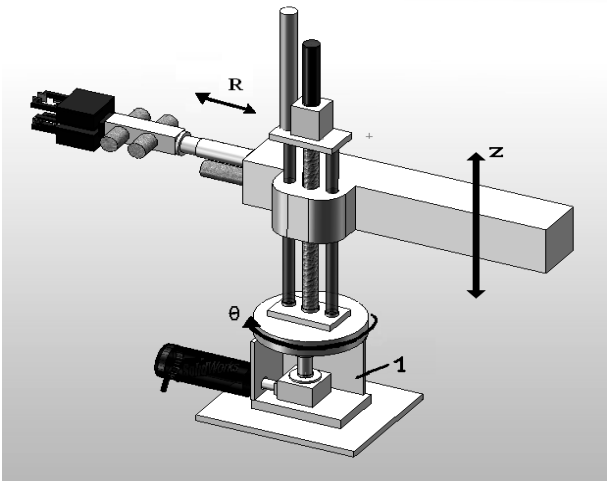


Рис. 8.2. Модель робота

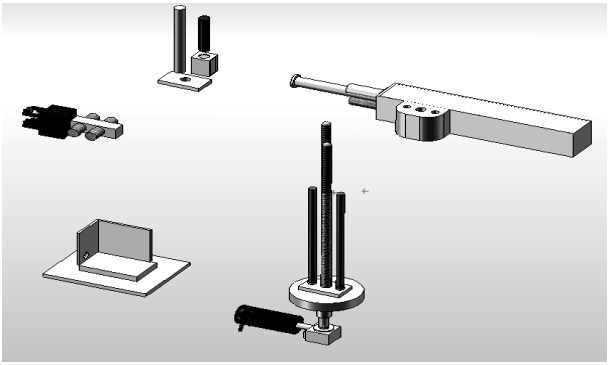


Рис. 8.3. Некоторые узлы робота

8.3. Описание моделирующего конструкторского пакета SolidWorks

В качестве примера при изучении пакета SolidWorks будем использовать внешний вид робота РБ-241 (рис. 8.1). Это – промышленный робот, работающий в цилиндрической системе координат, который используется для автоматизирования процессов машиностроения. Он может устанавливать и менять детали для обработки, менять рабочие инструменты, удалять стружку из зоны инструмента при помощи воздушной струи, вести подсчет обработанных деталей и обеспечивать разные другие обслуживающие операции на одной или двух металлообрабатывающих машинах [1].

Робот РБ-241 состоит из механической системы и управляющего устройства.

8.3.1. Описание пакета

Пакет SolidWorks, разработанный корпорацией SolidWorks (США), представляет собой полнофункциональное приложение для автоматизированного механико-машиностроительного конструирования, базирующееся на параметрической объектно-ориентированной методологии [13, 14]. Это позволяет легко получать твердотельную модель из двумерного эскиза, применяя очень простые и эффективные инструменты моделирования.

Однако представление проектируемого изделия не ограничивается трехмерным твердотельным моделированием – имеются средства ассоциативного конструирования. Это означает, что можно создать прототип класса деталей, например изготавливаемых штамповкой из листового металла, а затем использовать параметрическую модель при проектировании формы заготовки.

Кроме того, пакет SolidWorks упрощает проектирование полостных деталей, изготавливаемых литьем или в пресс-формах. С помощью SolidWorks можно создавать также поверхностные параметрические модели.

Графический интерфейс Windows позволяет конструктору усовершенствовать свои решения и реализовать их в виде виртуального прототипа или твердотельной модели, больших сборок, сборочных узлов, а также выполнить деталировку и получить необходимую чертежную документацию. Особенности работы с пакетом следующие.

- Модель SolidWorks состоит из трехмерной геометрии твердотельных элементов в документах деталей и сборок.

- Чертежи создаются из моделей или путем черчения видов в документе чертежа.
- Обычно сначала рисуется эскиз, создается основание, а затем в модель добавляются многочисленные элементы.
- Можно усовершенствовать проект путем добавления, редактирования или переупорядочивания элементов.
- Связь между деталями, сборками и чертежами гарантирует, что изменения, сделанные в одном документе или виде, автоматически выполняются во всех других документах и видах.
- Чертежи или сборки можно создавать на любом этапе в процессе проектирования.

Пакет SolidWorks – это лишь один из продуктов корпорации SolidWorks, входящей в состав Dassault Systems. Пакет SolidWorks может также служить программной платформой для некоторых приложений. Таким образом, в окне этой программы можно запускать совместимые приложения, разработанные корпорацией SolidWorks как программное обеспечение, встраиваемое в SolidWorks. Назовем некоторые программы, работающие на платформе SolidWorks:

- SolidWorks Animator – создание видеороликов;
- PhotoWorks – средства для получения фотореалистичного изображения модели;
- Feature Works – распознавание геометрии импортированных элементов;
- COSMOSWorks – инженерные расчеты;
- COSMOSMotion – динамический анализ механизмов;
- COSMOSFlow – модуль для анализа поведения жидкостей и газов в широком диапазоне чисел Рейнольдса;
- eDrawing – средство коллективной работы над проектом;
- SolidWorks Piping – проектирование трубопроводов;
- Toolbox – библиотека стандартных изделий;
- Mold Base – библиотека пресс-форм.

Особености работы с пакетом подробно представлены в книге [15] и учебном пособии [16].

8.3.2. Основные режимы работы

Программа SolidWorks имеет несколько режимов работы (рис. 8.4).

Режим Part (Деталь) представляет собой параметрическую объектно-ориентированную среду, позволяющую строить твердотельные модели. По умолчанию имеется три плоскости: Plane 1, Plane 2 и Plane 3.

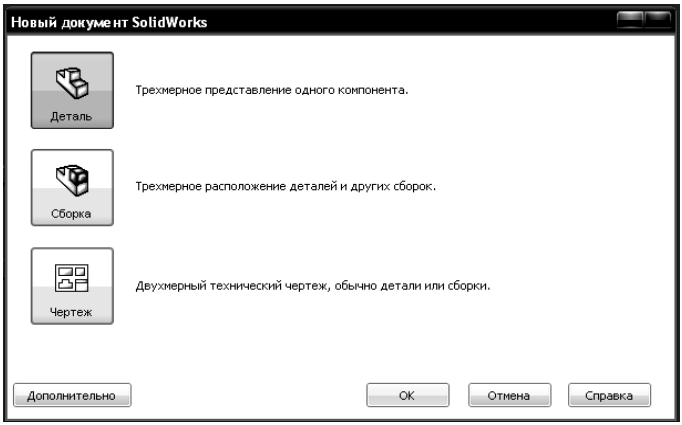


Рис. 8.4. Режимы работы SolidWorks

Сначала необходимо выделить плоскость, в которой будет строиться эскиз базового элемента. После этого открывается эскизная среда, располагающая всеми необходимыми инструментами для построения чертежей. Построив эскиз (рис. 8.5), нужно нанести размеры и установить требуемые взаимосвязи между его элементами, находясь все в той же среде построений. Добавление взаимосвязей, уравнений и расчетных таблиц помогает конструктору предельно четко выразить свой замысел.

В режиме Part (Деталь) доступна библиотека стандартных отверстий HOLE WIZARD. В ней представлены простые и секционные отверстия, а также расточенные, конические и т.д. Библиотека поддерживает стандарты ISO, ANSI и другие. В режиме Part создаются такие элементы чертежа, как обозначения сварных швов, геометрических допусков, базовых поверхностей, чистоты обработки поверхности. В пакет SolidWorks также входит панель библиотечных элементов Feature Palette, которая включает стандартные детали машин и изделия.

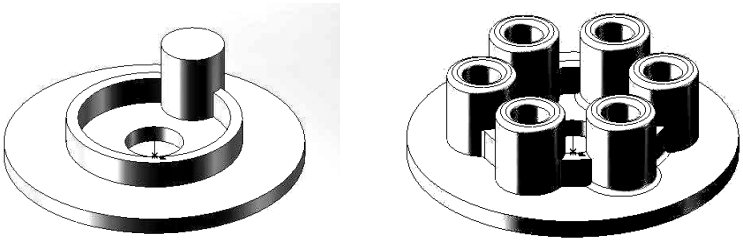


Рис. 8.5. Примеры работы режима Part и Assembly

Режим Assembly (Сборка). В режиме Assembly (Сборка) с помощью соответствующих инструментов выполняется объединение компонентов в сборку (рис. 8.6). Сборка компонентов может осуществляться двумя методами:

- сборка **снизу вверх**,
- сборка **сверху вниз**.

Рис. 8.6. Режим Assembly (Сборка)

При подходе **снизу вверх** сборка формируется путем интеграции ранее созданных компонентов с сохранением всех конструкторских решений. Подход **сверху вниз** подразумевает создание компонентов в режиме сборки: можно начать с каких-то готовых изделий и далее в контексте сборки создавать другие компоненты. При этом можно задавать зависимость размеров одних компонентов от размеров других.

Специальный режим **SmartMates** (Умное сопряжение) позволяет составить сборку из компонентов всего одним щелчком мыши. В процессе

добавления компонентов в сборку в SolidWorks можно использовать операцию перетаскивания, а также проверять *собираемость* полученной сборки.

Ценной возможностью SolidWorks является обнаружение противоречий в сборке, что позволяет конструктору при повороте и перемещении деталей видеть возникающие несоответствия между объединяемыми компонентами. Благодаря поддержке динамических свойств конструкции в программе SolidWorks можно получить анимационную модель процесса сборки. Имитация движения механизма выполняется с учетом воздействия двигателей, сил упругости и силы тяжести.

Режим **Drawing (Чертеж)** предназначен для формирования технической документации на созданные ранее детали и сборки в виде чертежных видов и их деталировок (рис. 8.7).

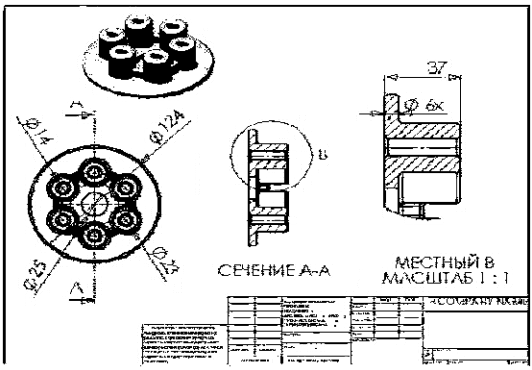


Рис. 8.7. Режим Drawing (Чертеж)

В SolidWorks составление документации осуществляется двумя способами.

Во-первых, можно получить чертежи автоматически на основе созданных деталей или изделий. На чертежах отображаются все размеры и обозначения, добавленные к компоненту в режиме Part (Деталь). При этом сохраняется свойство двусторонней ассоциативности. Чертеж сборки может быть также дополнен спецификацией и текстовыми примечаниями.

Во-вторых, построить чертежи изделия и нанести размеры можно *вручную* с использованием традиционных инструментов компьютерной инженерной графики.

8.3.3. Основные термины

Перед тем как продолжить дальнейшее изучение SolidWorks, необходимо познакомиться с некоторыми терминами, которые широко используются в лабораторной работе.

Объектно-ориентированное конструирование

Под *элементом* (feature) понимается наименьший стандартный блок, разрабатываемый индивидуально. В SolidWorks твердотельная модель создается путем интеграции некоторого числа таких стандартных блоков. Созданная в SolidWorks модель представляет собой комбинацию отдельных элементов, связанных между собой прямо или косвенно. Эти элементы должным образом *понимают* свои размеры и функции, и, следовательно, их можно легко модифицировать в процессе конструирования.

Если при конструировании полностью заданы все связи, то изменение какого-либо параметра или переопределение связей приводит к автоматическому изменению геометрии модели. Благодаря этому процесс конструирования становится более гибким.

Параметрическое конструирование

Параметрическая сущность пакета состоит в том, что в нем поддерживается возможность использовать стандартные свойства и параметры при определении размеров и формы модели. Это позволяет свободно модифицировать форму и размеры геометрии. Другими словами, можно модифицировать или переопределить форму и размеры любого элемента на любом этапе конструирования, что существенно облегчает выполнение конструкторских работ. Рассмотрим это на примере модели детали трубопровода (рис. 8.9).

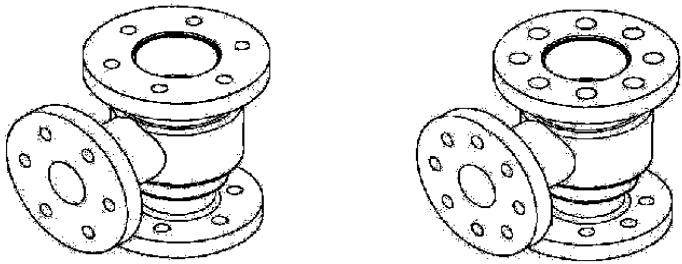


Рис. 8.9. Исходная и модифицированная детали трубопровода

Для того чтобы модифицировать эту конструкцию, например, изменить диаметр отверстий и их количество на передней, верхней и нижней поверхностях, нужно просто выделить элемент (отверстие) и изменить параметры его прототипа – диаметр и количество экземпляров.

Двунаправленная ассоциативность

Как уже отмечалось, в SolidWorks имеются различные режимы работы: Part (Деталь), Assembly (Сборка) и Drawing (Чертеж). Между ними поддерживается единая двусторонняя взаимосвязь. Таким образом, все изменения, внесенные конструктором в модель в одном из режимов, немедленно будут отражены и в других режимах.

Как уже отмечалось, в SolidWorks имеются различные режимы работы: Part, Assembly и Drawing (см. рис. 8.10). Между ними поддерживается единая двусторонняя взаимосвязь. Таким образом, все изменения, внесенные конструктором в модель в одном из режимов, немедленно будут отражены и в других режимах. На чертеже (рис. 8.10) изображены различные виды детали трубопровода, модель которой приводилась на рис. 8.9.

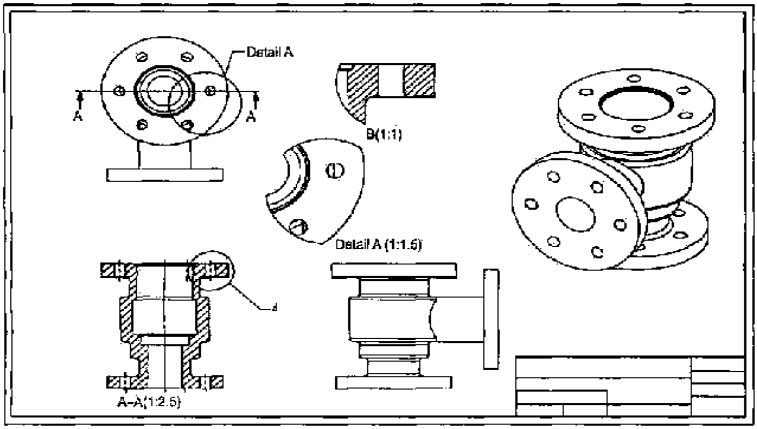


Рис. 8.10. Виды детали трубопровода до модификации

Например, если изменить размеры детали в режиме Part (Деталь), размеры изменятся и в режиме Assembly (Сборка), и в режиме Drawing (Чертеж). Аналогично, если вы измените размеры детали на чертеже, автоматически сгенерированном в режиме Drawing (Чертеж), то в режимах

Part (Деталь) и Drawing (Чертеж) эти изменения также появятся. Поясним это на примере.

Если модифицировать модель этой детали в режиме Part (Деталь), это повлечет за собой автоматическое изменение чертежей в режиме Drawing (Чертеж). Как изменились виды этой детали после увеличения диаметра и количества отверстий, показано на рис. 8.11.

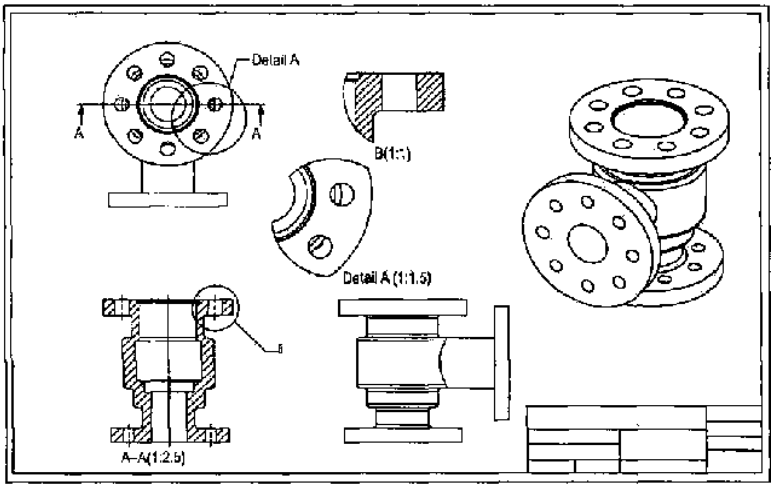


Рис. 8.11. Виды детали трубопровода после модификации

8.4. Пример конструирования детали

Рассмотрим методику создания в пакете SolidWorks детали, изображенной на рис. 8.12.

Это упражнение включает:

- создание основания;
- добавление элемента – бобышка;
- добавление выреза.

8.4.1. Создание основания

Открытие нового документа детали.

1. Нажмите кнопку ***Создать*** на панели инструментов ***«Стандартная»***. Появится диалоговое окно ***Новый документ SolidWorks***.

2. Выберите параметр ***Деталь***, затем нажмите ***ОК***.

3. Появится окно новой детали.

Первым элементом в детали является коробка, вытянутая из эскизного прямоугольного профиля. Начните с рисования прямоугольника.

1. Нажмите кнопку ***Вытянутая бобышка/основание*** на панели инструментов ***«Элементы»***. Появятся ***Передняя***, ***Верхняя*** и ***Правая*** плоскости, и указатель примет форму: 

2. Поместите указатель на ***переднюю*** плоскость и выберите ее. Изображение на дисплее изменится таким образом, что ***Передняя*** плоскость будет обращена прямо на пользователя.

3. Выберите ***Прямоугольник*** на панели инструментов ***«Инструменты эскиза»***.

4. Переместите указатель в исходную точку эскиза. При помещении указателя на исходную точку он принимает следующую форму  .

5. Нажмите на исходную точку, а затем переместите указатель, чтобы создать прямоугольник (рис. 8.13). При перемещении рядом с ним отображается размер прямоугольника.

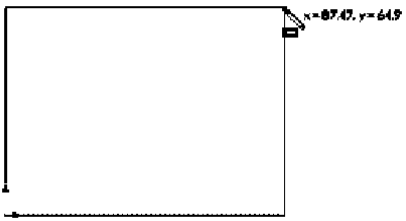


Рис. 8.13. Нарисованный прямоугольник

6. Кнопкой мыши завершите построение прямоугольника.

7. Нажмите кнопку ***Выберите*** на панели инструментов ***«Стандартная»***.

8. Чтобы изменить размеры прямоугольника, перетащите одну из сторон синего цвета или вершину.

Добавление размеров

1. Выберите ***Инструменты***, ***Параметры***, ***Настройки пользователя***, ***Общие***.

2. Отключите параметр ***Ввести значение размера***, затем нажмите ***OK***.

3. Выберите ***Автоматическое нанесение размеров*** на панели инструментов ***«Эскиз»***.

4. Нажмите на верхнюю кромку прямоугольника, затем нажмите в том месте, где требуется нанести размер.

5. Убедитесь, что выбран параметр ***Автоматическое нанесение размеров***, и нажмите на правую сторону прямоугольника, чтобы нанести ее размер (рис. 8.14).

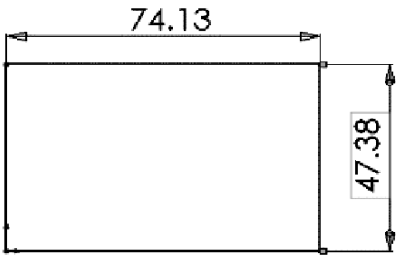


Рис. 8.14. Изображение размеров детали

Изменение значений размеров

1. Дважды нажмите на один из размеров. Появится диалоговое окно ***Изменить*** (рис. 8.15). Текущий размер выделен.

Рис. 8.15. Задание размеров детали

2. Установите значение, равное ***120***, затем нажмите кнопку с галочкой. Размер на эскизе изменится в соответствии с новым размером. Значение размера теперь составляет ***120 мм***.

3. Нажмите кнопку ***Изменить в размер экрана*** на панели инструментов ***«Вид»***, чтобы отобразить весь прямоугольник в полный размер и поместить его по центру в графической области.

4. Дважды нажмите на другой размер и введите значение ***120***.

Вытяжка основания

Первый элемент в любой детали называется основанием. Этот элемент создается путем вытяжки нарисованного прямоугольника.

1. Выберите ***Выход из эскиза*** на панели инструментов ***«Эскиз»*** или на панели инструментов ***«Стандартная»***. Появится диалоговое окно ***Вытянуть*** PropertyManager (Менеджера свойств) в дереве конструирования FeatureManager (на левой панели), вид эскиза будет показан в тримметрии (рис. 8.16), а в графической области появится предварительный вид вытяжки.

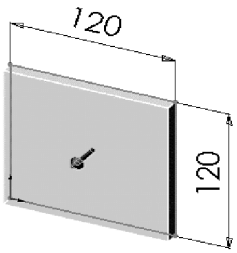


Рис. 8.16. Вид эскиза панели

2. В окне PropertyManager (Менеджере свойств) в окне группы ***Направление 1***: Установите для параметра ***Граничное условие*** значение ***На заданное расстояние***. Введите значение ***30*** для параметра ***Глубина***.

3. Нажмите ***OK*** для создания вытяжки.

Сохранение детали

Нажмите кнопку ***Сохранить*** на панели инструментов *«Стандартная»*. Файл будет сохранен с заданным именем и расширением ***.sldprt***.

8.4.2. Создание бобышки

Рисование бобышки.

Для создания дополнительных элементов на детали (например, бобышек или вырезов) можно рисовать их на гранях или плоскостях модели, а затем вытягивать эскизы.

Можно рисовать за один раз только на одной грани или плоскости.

1. Нажмите кнопку ***Скрыть невидимые линии*** на панели инструментов ***«Вид»***.

2. Нажмите кнопку ***Вытянутая бобышка / Основание***.

3. Переместите указатель по лицевой грани детали. Границы грани высветятся, указывая на то, что эта грань выбрана.

4. Нажмите на лицевую грань детали для ее выбора. В диспетчере команд появятся команды панели инструментов ***«Эскизы»***.

5. Нажмите кнопку ***Окружность*** на панели инструментов ***«Эскиз»***. Указатель примет форму: 

6. Нажмите рядом с центром грани и, перемещая указатель, нарисуйте окружность (рис. 8.17). Нажмите снова для завершения построения окружности.

Нанесение размеров и вытяжка бобышки

Для установления местоположения и размера окружности добавьте необходимые размеры.

1. Выберите Автоматическое нанесение размеров.

2. Нажмите на верхнюю кромку грани, нажмите на окружность, затем в том месте, где необходимо нанести центр окружности.

.3. Дважды нажмите на *размер*, установите значение ***60*** в диалоговом окне ***Изменить*** и нажмите ***ОК***.

4. Повторите операцию, чтобы задать расстояние от центра окружности до боковой кромки грани. Установите это значение, также равное ***60***.

5. Пользуясь все тем же инструментом ***Автоматическое нанесение размеров***, нажмите на окружность для определения ее диаметра.

6. Нажмите в том месте, где нужно нанести размер диаметра. Установите значение диаметра, равное ***70*** (рис. 8.17). Окружность становится черной, а в строке состояния указывается, что эскиз полностью определен.

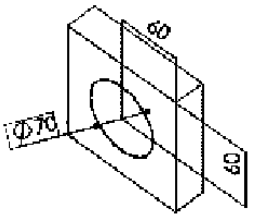


Рис. 8.17. Эскиз детали

7. Нажмите ***Выход из эскиза*** . Появится диалоговое окно ***Вытянуть*** PropertyManager (Менеджера свойств).

8. В окне PropertyManager (Менеджер свойств) в окне группы ***Направление 1***, установите для параметра ***Глубина*** значение ***25***, а для других элементов оставьте параметры по умолчанию и нажмите ***ОК*** для вытяжки элемента бобышки.

Элемент ***Extrude2 (Вытянуть 2)*** появится в дереве конструирования FeatureManager (рис. 8.18).

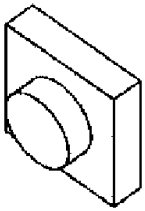


Рис. 8.18. Вытягивание бобышки

8.4.3. Создание выреза

Сначала нарисуем вырез и укажем для него размеры.

1. Нажмите кнопку ***Закрасить с кромками*** на панели инструментов ***«Вид»***.

2. Нажмите кнопку ***Вытянутый вырез*** на панели инструментов ***«Элемент»***.

3. Нажмите на лицевую грань круговой бобышки для ее выбора.

4. Нажмите ***Стандартные виды*** и выберите ***Перпендикулярно***. Деталь поворачивается выбранной гранью модели к смотрящему.

5. Нарисуйте окружность рядом с центром бобышки, как показано на рисунке 8.19.

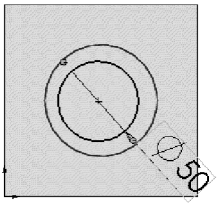


Рис. 8.19. Создание *выреза*

Выберите ***Автоматическое нанесение размеров*** и установите размер диаметра окружности равный ***50 мм.***

Затем добавьте взаимосвязь ***«концентричность».***

1. Нажмите кнопку ***Добавить взаимосвязи*** на панели инструментов ***«Эскиз»***. Появится окно ***Добавить взаимосвязи*** PropertyManager (Менеджера свойств).

2. Выберите нарисованную окружность (внутреннюю окружность) и кромку бобышки (наружную окружность).

3. В разделе ***Добавить взаимосвязи*** выберите ***Концентричность***.

4. Нажмите кнопку ***ОК***.

Далее завершите вырез.

1. Нажмите ***Выход из эскиза***. Появится диалоговое окно ***Вырез-Вытянуть*** PropertyManager (Менеджера свойств).

2. В окне PropertyManager (Менеджер свойств) в окне группы ***Направление 1*** выберите ***Через все*** в списке ***Граничное условие***.

3. Нажмите кнопку ***ОК***.

4. Нажмите ***Стандартные виды*** и выберите ***Триметрия***.

5. Нажмите кнопку ***Сохранить,*** чтобы сохранить деталь. Законченный вид детали представлен на рис. 8.20.

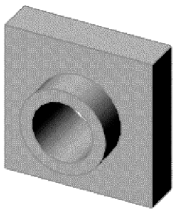


Рис. 8.20. Законченный вид детали

8.5. Дополнительные возможности SolidWorks

Стандарт нанесения размеров и единицы измерения

После установки пакета SolidWorks можно выбрать единицы измерения и стандарт, которые будут использоваться при нанесении размеров. SolidWorks поддерживает множество распространенных стандартов, например ANSI, ISO, DIN и ГОСТ. Кроме того, в программе имеется большой набор различных единиц измерения: миллиметры, сантиметры, дюймы и т.д.

Библиотечные элементы

Обычно при выполнении проектно-конструкторских работ некоторые элементы используются очень часто. В большинстве других программ твердотельного моделирования приходится каждый раз создавать эти элементы заново. Но пакет Solid Works позволяет **сохранить** нужный тип элемента в библиотеке и использовать его многократно. Это экономит время конструктора и повышает эффективность работы.

Панель библиотечных элементов

Панель библиотечных элементов Feature Palette также является уникальной возможностью SolidWorks. Используя панель элементов библиотеки, можно выбирать из раскрывающихся списков готовые детали, такие как валы, фрезы, пресс-формы для листового металла и т.д. [16].

Уравнения

Уравнения – это аналитические и численные формулы, которые определяют взаимосвязи или ограничения размеров элементов в процессе построения эскиза элемента или после него. Уравнения могут быть применены и к готовым элементам, вставленным в эскиз.

Обнаружение противоречий в сборке

Встроенные в программу средства обнаружения противоречий позволяют определить наличие наложений и конфликтов между узлами сборки в процессе движения сборки. Во время создания сборки непротиворечивость взаимосвязей между ее узлами можно проверять, перемещая и поворачивая узлы.

Поиск ошибок

В процессе создания элемента модели или после редактирования элемента, созданного ранее, его геометрия может оказаться недопустимой. Система не сможет создать такой элемент, и для поиска допущенной ошибки будут использоваться средства обнаружения дефектов ***What's Wrong? (Поиск ошибки)***.

COSMOSXpress

В SolidWorks имеется специальный компонент ***COSMOSXpress*** – инструмент для статического анализа и анализа напряжений. В *COSMOSXpress* доступен только один вид анализа – линейный статический анализ. С его помощью можно оценить смещение, растяжение и напряжения,

прилагаемые к компоненту, с учетом материала, различных условий нагружения и закрепления модели.

При достижении предельных напряжений **элемент разрушается**.

8.6. Контрольные вопросы

1. Сколько режимов работы имеет SolidWorks? Назовите их.
2. Что понимается под элементом в SolidWorks?
3. Можно ли чертежи создать из моделей или только путем черчения видов в документе чертежа?
4. Изменения, сделанные в одном документе или виде, автоматически выполняются во всех других документах и видах? Если да, то с помощью чего?
5. При каком подходе сборка формируется путем интеграции ранее созданных компонентов?
6. Что такое двунаправленная ассоциативность?

9. РАЗРАБОТКА МОДЕЛИ ТЕХНОЛОГИЧЕСКОГО ОБЪЕКТА

Лабораторная работа № 9

9.1. Цель работы

Научиться создавать модель реального технологического объекта (робота или станка), используя возможности пакета SolidWorks.

9.2. Задания к лабораторной работе

1. Изучить теоретический материал, изложенный в описании данной лабораторной работы.
2. Запустить АОС по изучению конструкторских возможностей данного пакета и выполнить все задания, указанные в ней.
3. Ответить на все вопросы АОС.
4. Смоделировать один из предложенных преподавателем технологических объектов (см. рис. 9.11–9.16).
5. Провести исследование виртуальной модели на адекватность моделируемому объекту.
6. Ответить на контрольные вопросы.
7. Написать отчет о проделанной работе.

9.3. Основные методы сборки

Синтез модели любого технологического объекта осуществляется на основе ***проекта сборки***.

Проект сборки определяется как проект, состоящий из двух или более компонентов, собранных вместе соответственно их рабочему положению. Компоненты собираются вместе в режиме ***Assembly (Сборка)*** при помощи параметрических соотношений. В SolidWorks такие соотношения называются ***сопряжениями***. Сопряжение дает возможность ограничивать степень свободы компонентов согласно их рабочему положению.

Сборка компонентов может осуществляться двумя методами:

- сборка **снизу вверх**,
- сборка **сверху вниз**.

В данной лабораторной работе будет рассмотрено проектирование сборки методом *снизу вверх*. Этот метод является традиционным, наибо-

лее распространенным и предпочтительным методом проектирования. При помощи этого метода создаются документы отдельных деталей, которые затем размещаются в сборке. На них имеются ссылки как на внешние компоненты.

При использовании данного метода проектирования детали создаются в режиме Part (Деталь) и хранятся в файлах с расширением .sldprt.

Для примера рассмотрим этапы создания схвата робота РБ-241 (см. рис. 8.1, 8.2). Он состоит из 12 простых элементов (рис. 9.1), которые объединяются в сборку с помощью наложения взаимосвязей.

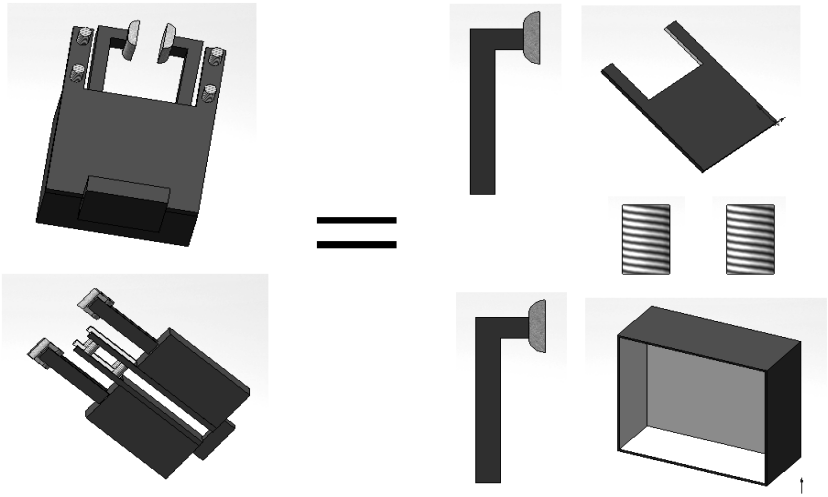


Рис. 9.1. Синтез схвата робота РБ-241 с двумя сменными кистями на основе проекта сборки

Построение элемента начинается с использования менеджера команд ***Sketch (Эскиз)***. Этот менеджер команд предназначен для перехода в среду двумерных или трехмерных построений. Сначала выделяем плоскость, в которой будем строить эскиз базового элемента – Plane 1, Plane 2 или Plane 3.

После этого, используя инструменты для построения чертежей, создаем первый элемент схвата. Это будет основание коробки. С помощью команды ***Вытянутая бобышка/основание*** строим эскиз прямоугольника, который потом преобразуем в элемент, используя, менеджер команд ***Features (Элементы)***. Преобразование эскиза в элемент «Основание коробки» представлено на рис. 9.2.

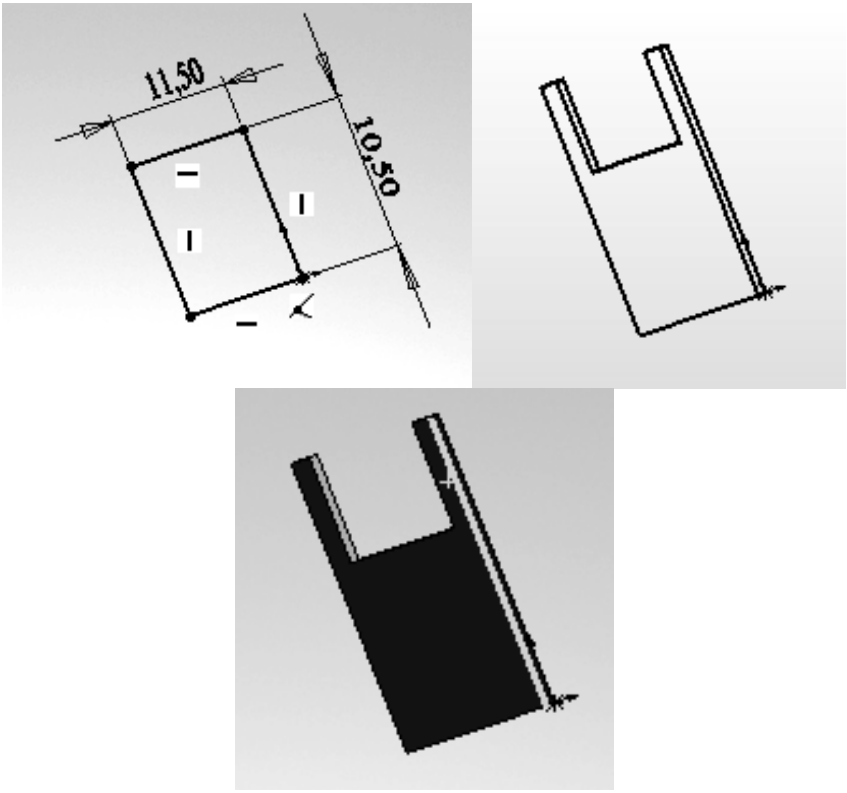


Рис. 9.2. Преобразование эскиза в элемент с помощью команды Features

Когда все детали сборки спроектированы, открывается документ сборки (.sldasm), в который переносятся смоделированные детали при помощи инструментов режима Assembly (Сборка). Размещаем первый компонент в начале координат сборки. Это способствует тому, что плоскости и детали сборки будут по умолчанию совпадать, а детали будут иметь ту же ориентацию, что и в режиме Part (Деталь). В начале работы с новым документом SolidWorks в режиме *Assembly (Сборка)* на экране появляется менеджер свойств ***Insert Component (Вставить компонент)***, который изображен на рис. 9.3.

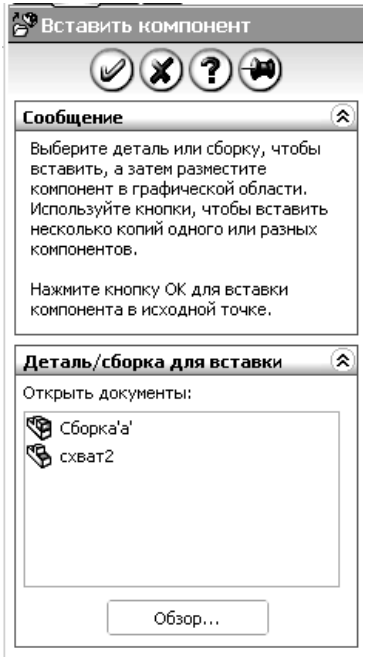


Рис. 9.3. Менеджер свойств ***Insert*** Component

При щелчке на кнопке Обзор, доступной в раздвижной панели ***Деталь/Сборка*** для вставки, на экране появится диалоговое окно ***Открыть***, в котором необходимо выбрать папку, где был сохранен компонент. Далее выделяем компонент и щелкаем на кнопке *Открыть*. Курсор будет замещен курсором компонента, и, кроме того, на экране появится предварительное изображение компонента.

Сопряжения.

После того как компоненты размещены в документе сборки, необходимо собрать их. При сборке компонентов ограничивается степень их свободы. Как уже упоминалось, компоненты собираются при помощи ***сопряжения***. Сопряжение помогает точно разместить компонент и определить его положение относительно других компонентов и деталей сборки.

Сопряжения состоят из набора логических операций, которые определяют отношение (например, касание или перпендикулярность) между элементами эскиза модели, плоскостями, ребрами или вершинами.

Существуют следующие виды сопряжений:

- Сопряжение ***Perpendicular (Перпендикулярность)*** – два выделенных сегмента линий становятся перпендикулярными друг другу (рис. 9.4).

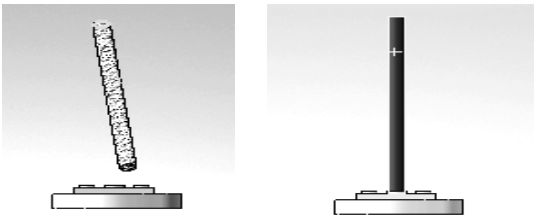


Рис. 9.4. Сопряжение ***Перпендикулярност***ь

- Сопряжение ***Parallel (Параллельность)*** – два выделенных сегмента линий становятся параллельными друг другу (рис. 9.5).

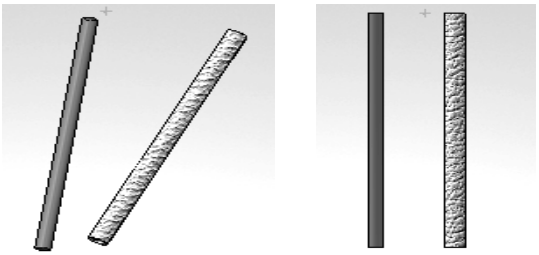


Рис. 9.5. Сопряжение ***параллельность***

3. Сопряжение ***Concentric (Концентричность)*** – для двух выделенных дуг или окружностей эта взаимосвязь означает совмещение их центров (рис. 9.6).

При сборке схвата для создания сопряжения ***Coincident (Совпадение)*** использовался менеджера свойств ***Mate (Сопряжение)*** (рис. 9.7).

Чтобы создать сопряжение между элементами основания коробки и самой коробкой необходимо выделить плоскую поверхность на первом компоненте, а затем элемент из второго компонента. Элементы будут окрашены зеленым цветом.

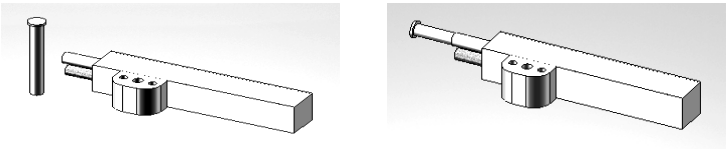


Рис. 9.6. Сопряжение ***концентричность***

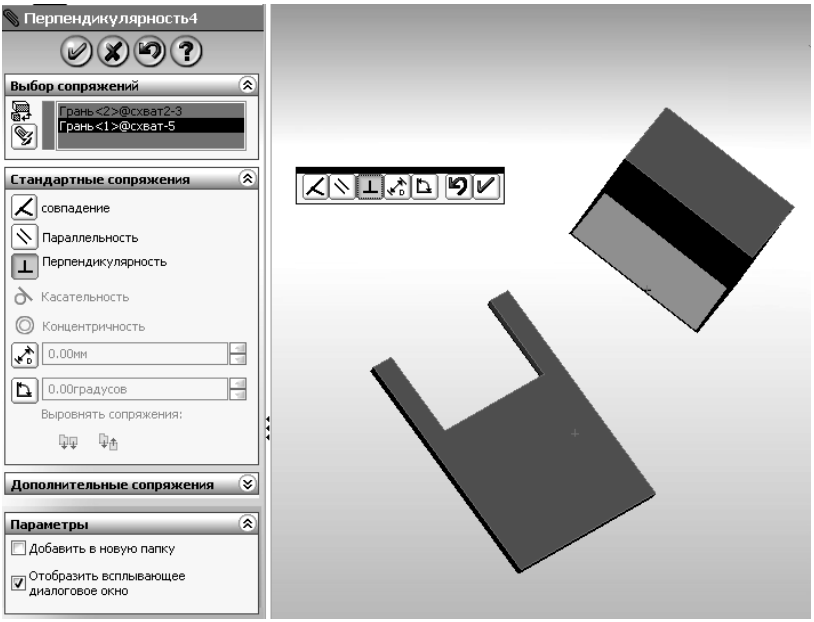


Рис. 9.7. Менеджер свойств ***Mate***

Имена выбранных элементов отобразятся в области выделения ***Entities to Mate (Элементы для сопряжения)*** раздвижной панели ***Mate Selections (Выбор сопряжения)***. На экране также появится панель инструментов ***Mate (Сопряжение)***, как показано на рис. 9.7. Наиболее подходящее сопряжение для данного набора выделенных компонентов отображается на панели менеджера свойств ***Mate (Сопряжение)*** и в раздвижной панели ***Standard Mates (Стандартные сопряжения)***. По умолчанию выбирается наиболее подходящее сопряжение. В данном случае использовалось сопряжение перпендикулярность.

После того, как два эти элемента связали, в сборку добавляются 2 кисти схвата (рис. 9.8), которые располагаются параллельно друг другу с помощью сопряжения Parallel (Параллельность).

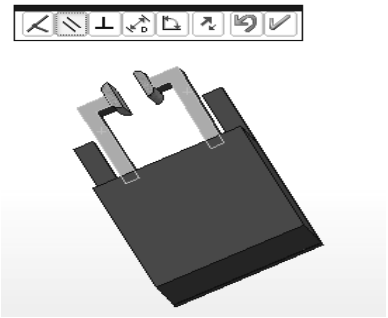


Рис. 9.8. Сборка схвата

Таким образом, была спроектирована верхняя часть схвата. Для проектирования нижней части проделываем аналогичные операции. Две части схвата соединяются между собой с помощью четырех пружин (рис. 9.9).

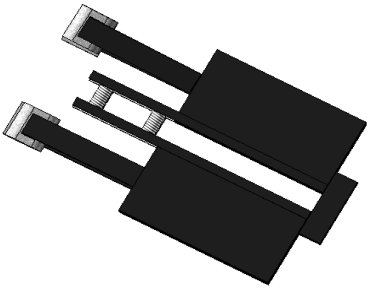


Рис. 9.9. Схват робота РБ-241

Основное преимущество данного метода проектирования заключается в том, что компоненты проектируются независимо и между ними можно легко поддерживать взаимосвязи. Следовательно, данный метод проектирования позволяет уделить больше внимания отдельным компонентам. Кроме того, этот метод проектирования наиболее предпочтителен при работе с большими сборками.

Аналогичным способом были построены следующие части робота (рис. 9.10): вертикальная стойка 1 с двумя направляющими колоннами и приводами вертикального (ось Z) и кругового (координата θ) перемещений руки робота, сама рука робота 2, обеспечивающая перемещения объекта в горизонтальном направлении (ось R).

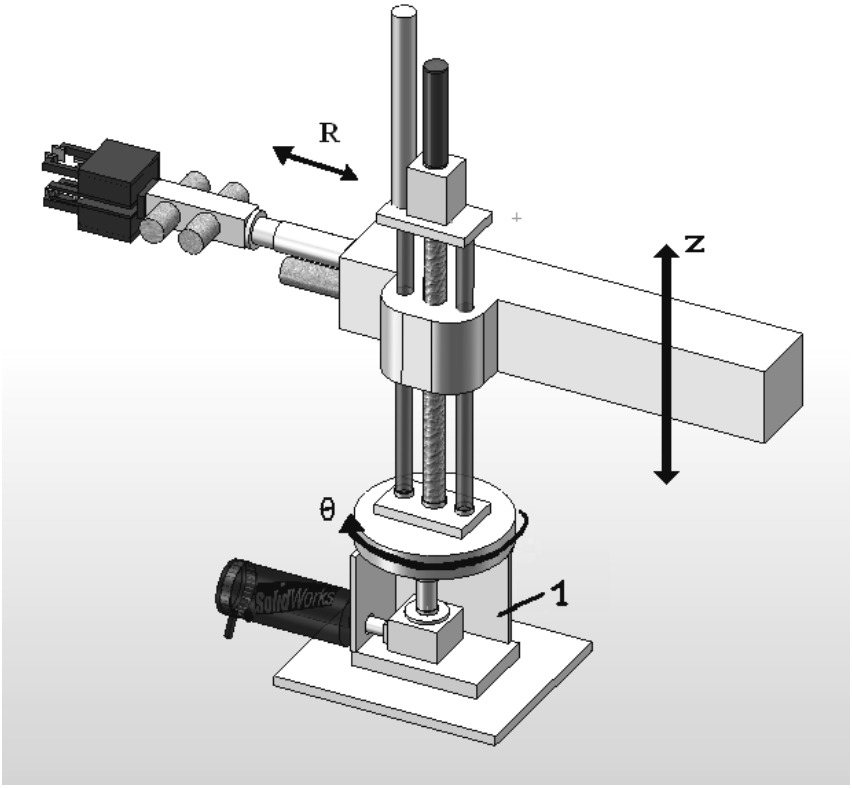


Рис. 9.10. Синтез робота РБ-241 на основе проекта сборки

9.4. Варианты объектов для моделирования

На рис. 9.11–9.16 приведены фотографии некоторых технологических объектов для построения виртуальных моделей.

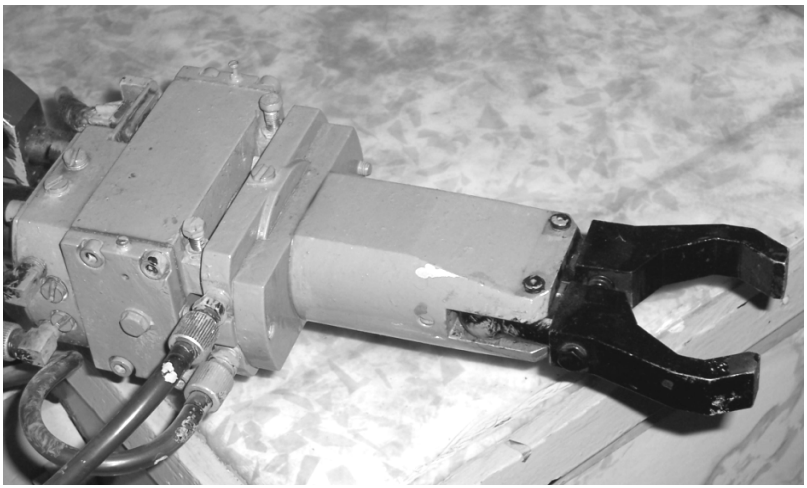


Рис. 9.11. Схват робота

Рис. 9.12. Ручной пресс серии AP

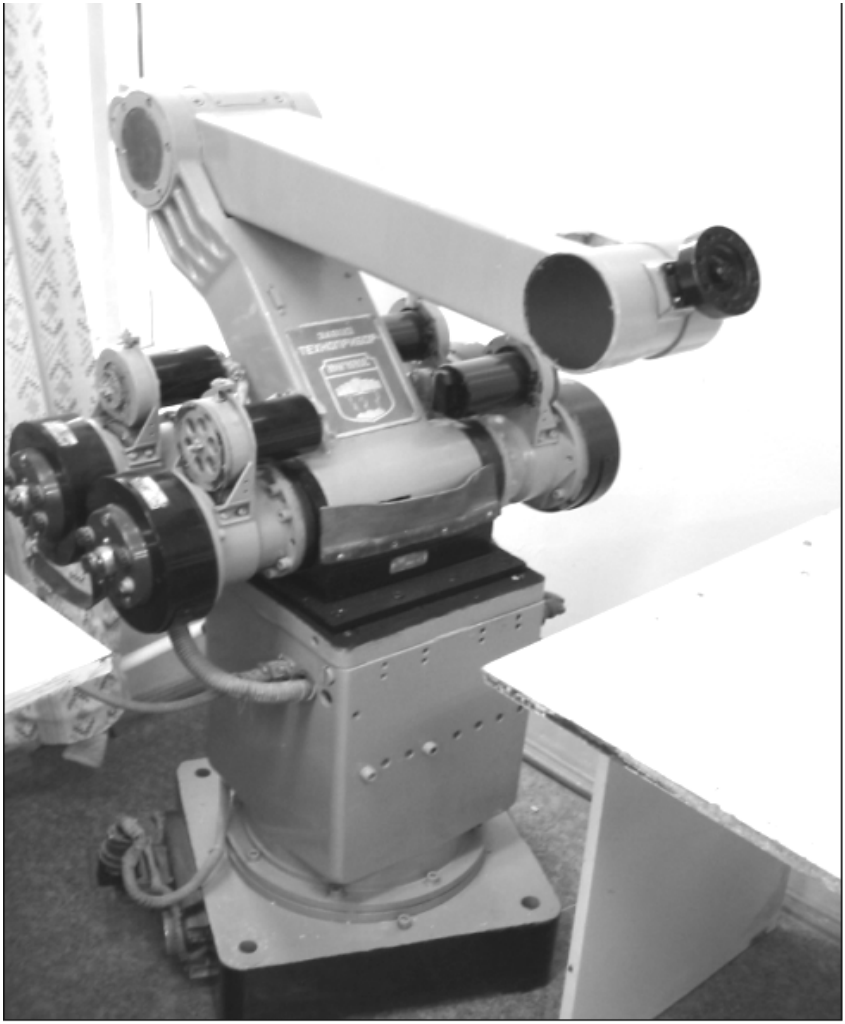


Рис. 9.13. Робот ТУР-10

Рис. 9.14. Сборочный робот ЦПР-1П

Рис. 9.15. Робот – элемент ГПМ

Рис. 9.16. Токарный станок **WM180Ф3**

9.5. Контрольные вопросы

1. Что такое *Базовый объект модели*?
2. Назовите основные методы сборки. В чем их сущность?
3. Какая среда используется для построения элементов?
4. В чем заключено преимущество метода «снизу вверх»?
5. Как производится поворот эскиза в SolidWorks?
6. Для чего первый элемент в сборке размещают в начале координат?
7. Назовите основные виды сопряжений. В чем их отличие?
8. Для чего в сборке применяются сопряжения?
9. Какое сопряжение привязывает центральную точку выделенной дуги или окружности к центральной точке другой дуги или окружности?

10. АНИМАЦИЯ ВИРТУАЛЬНОЙ МОДЕЛИ

Лабораторная работа № 10

10.1. Цели работы

1. Научиться отображать динамику работы элементов объекта.
2. Научиться отображать динамику работы объекта в целом.
3. Научиться создавать анимированные изображения.

10.2. Задания к лабораторной работе

1. Изучить теоретический материал, изложенный в описании данной лабораторной работы.
2. Изучить и организовать один из предложенных видов анимации.
3. Используя модель, созданную при выполнении лабораторной работы № 9, продемонстрировать возможность пермещения механических элементов объекта по всем управляемым координатам.
4. Продемонстрировать возможность визуализации сборки и разненсения узлов модели.
5. Ответить на контрольные вопросы.
6. Написать отчет о проделанной работе.

10.3. SolidWorks Animator

Для создания анимации (оживленных изображений) и передачи движения в сборках используют пакет ***SolidWorks Animator***. Он используется для выполнения следующих действий:

- перемещения модели;
- вращения модели;
- создания анимации с помощью вида с разнесенными частями;
- указания пошагового порядка анимации;
- добавления дополнительных шагов в анимацию.

10.3.1. Перемещение модели

В SolidWorks существует возможность перемещения отдельных не связанных ограничениями компонентов в документе сборки, без оказания влияния на положение других компонентов. Существует три способа перемещения компонентов.

Перемещение компонентов при помощи перетаскивания.

Одним из главных преимуществ режима ***Assembly (Сборка)*** SolidWorks 2007 является возможность перемещать компонент, размещенный в сборке, не вызывая никаких инструментов. Просто выделите компонент и перетаскивайте. Отпустите левую кнопку мыши, чтобы расположить компонент в требуемой области.

Перемещение отдельных компонентов при помощи инструмента *Move Component (Перемещение компонента).*

Также можно перемещать индивидуальные компоненты при помощи инструмента *Move Component* (Перемещение компонента).

Щелкните на кнопке *Move Component* в менеджере команд. На экране появится менеджер свойств *Move Component*. Вам будет предложено выделить компонент и перетащить его. Курсор выделения заменится курсором перемещения; выделите компонент и перетащите его. Отпустите левую кнопку мыши, чтобы оставить компонент в требуемой точке.

10.3.2. Вращение модели

SolidWorks позволяет выполнять вращение индивидуального компонента в документе сборки, не влияя на расположение других компонентов. Инструмент ***Rotate Component (Вращение компонента)*** используется для вращения компонента. Существует три способа вращения индивидуального компонента.

Вращение компонента при помощи перетаскивания.

В SolidWorks можно вращать компонент, размещенный в сборке, не вызывая никаких инструментов. Чтобы повернуть компонент, просто выделите компонент при помощи правой кнопки мыши и перетащите курсор для вращения компонента. Отпустите правую кнопку мыши после получения требуемой ориентации индивидуального компонента.

Вращение компонентов при помощи инструмента *Rotate Component (Вращение компонента).*

Поворачивать индивидуальные компоненты можно также при помощи инструмента ***Rotate Component***. Чтобы повернуть компонент при помощи данного инструмента, щелкните на кнопке *Rotate Component* в менеджере команд. На экране отобразится менеджер свойств *Rotate Component*. Далее вам будет предложено выделить компонент и перетаскивать его для выполнения вращения.

Вращение компонента при помощи *Тройки.*

Одним из усовершенствований режима сборки в SolidWorks является инструмент ***Triad (Тройка)***. При помощи *Тройки* можно перемещать индивидуальный компонент.

Для этого необходимо выделить компонент, а затем щелкнуть правой кнопкой мыши, чтобы появилось контекстное меню. В появившемся контекстном меню выберите параметр ***Move with Triad (Переместить с помощью тройки)***. На компоненте появится трехмерное изображение системы координат. При помощи этой *Тройки* вы можете перемещать и вращать компонент. Различные компоненты *Тройки* показаны на рис. 10.1.

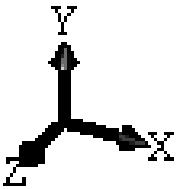


Рис. 10.1. Вид инструмента Тройка

Также можно перемещать компонент в плоскости XY. Для этого выберите крыло тройки XY и перетащите курсор, чтобы переместить компонент. Для перемещения компонента в плоскости YZ выберите YZ-крыло и перетащите курсор. Аналогично, при помощи крыла ZX вы можете перетащить компонент в плоскости ZX.

10.3.3. Разнесение модели

В SolidWorks можно отображать сборку в разнесенном виде (рис. 10.2). Есть два способа создания разнесенного вида сборки. Первый способ – ***автоматический***.

Чтобы автоматически создать разнесенный вид сборки, необходимо использовать команду ***Auto Explode (Авторазнесение)*** в диалоговом окне ***Assembly Exploder (Разнесение сборки)***. Компоненты сборки будут разнесены и помещены в произвольные места в документе сборки.

Второй способ – создание ***систематического разнесенного вида***. Вид с разнесенными частями состоит из одного или нескольких шагов разнесения, которые задаются пользователем.

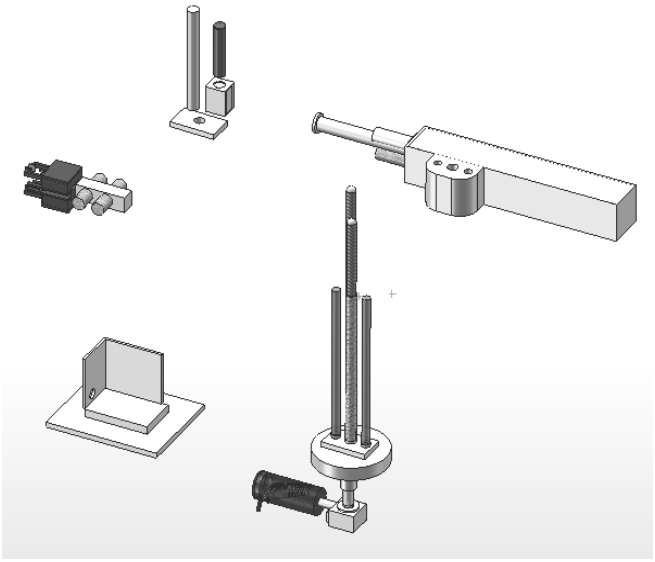


Рис. 10.2. Разнесение сборки

Чтобы создать систематический разнесенный вид сборки, щелкните на кнопке ***New (Создать)*** в диалоговом окне ***Assembly Exploder (Разнесение сборки)***. Размер диалогового окна увеличится, и появятся некоторые новые элементы управления, как показано на рис. 10.3. В этом окне будет предложено создать новый шаг разнесения. Область выделения ***Direction to explode along (В каком направлении разнести)*** будет актив-

ной. Выделите направление, в котором вы хотите разнести компоненты. Роль направляющей для разнесения может играть кромка, грань, плоскость, ось или эскиз.

Когда направление разнесения выбрано, активной становится область выделения ***Components to explode (Какие компоненты разнести)***. Выделите компоненты, которые требуется разнести в выбранном в настоящий момент направлении.

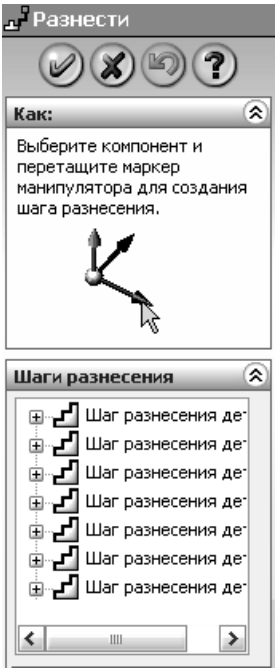


Рис. 10.3. Окно Assembly Exploder

Задайте дистанцию разнесения с помощью счетчика ***Distance (Расстояние)***. Можно также изменить направление разнесения на противоположное, установив флажок ***Reverse Direction (Реверс направления)***. При установленном флажке ***Explode related components together (Разнести взаимосвязанные компоненты вместе)*** разносятся все компоненты, связанные с выделенным компонентом посредством сборочных сопряжений.

После того как разнесены компоненты на одно и то же расстояние в выбранном направлении, необходимо разнести другие компоненты.

Для этого необходимо создать новый шаг разнесения. Щелкните на кнопке ***New (Создать)*** в диалоговом окне *Assembly Exploder*. Теперь, следуя той же процедуре, разнесите следующую совокупность компонентов. Можно создавать столько шагов разнесения, сколько пожелаете.

Создав все шаги разнесения, щелкните на кнопке ОК в диалоговом окне *Assembly Exploder*, чтобы завершить создание разнесенного вида.

10.4. Контрольные вопросы

1. Для чего используют SolidWorks Animator?
2. Назовите основные способы перемещения компонентов.
3. Для чего используют инструмент Triad (Тройка)?
4. Сколько существует способов разнесения сборки? Назовите их.
5. Какой флажок необходимо установить, чтобы изменить направление разнесения на противоположное?
6. Как организовать анимацию объекта?

11. МОДЕЛИРОВАНИЕ ОПЕРАЦИОННОЙ СИСТЕМЫ РЕАЛЬНОГО ВРЕМЕНИ СЕТЯМИ ПЕТРИ

Лабораторная работа № 11

11.1. Цели работы

1. Научиться моделировать динамические системы сетями Петри.
2. Изучить алгоритм функционирования операционной системы реального времени (ОС РВ).
3. Научиться моделировать фрагменты ОС РВ.

11.2. Задания к лабораторной работе

1. Изучить теоретическую часть.
2. Научиться работать с пакетом HPSim.
3. Построить в пакете HPSim учебные модели.
4. Построить в пакете HPSim фрагменты ОС РВ.
5. Проверить адекватность моделей и исследовать их.
6. Ответить на контрольные вопросы.

11.3. Теоретическое описание моделирования сетями Петри

Сети Петри представляют собой математическую модель для представления структуры и анализа динамики функционирования систем в терминах «условие-событие». Эта модель может быть успешно использована для описания, так называемых, динамических дискретных систем различных классов, таких как: вычислительные процессы и программы, технологические процессы, информационные, экономические, биологические, социальные и технические системы [1, 9].

Модели сетей Петри позволяют исследовать работоспособность моделируемых систем, оптимальность их структуры, эффективность процесса их функционирования, а также возможность достижения в процессе функционирования определенных состояний. Сети Петри и их обобщения являются удобным и мощным средством моделирования асинхронных, параллельных, распределенных и недетерминированных процессов, позволяют наглядно представить динамику функционирования систем и составляющих их элементов.

11.3.1. Структура сети Петри

Сеть Петри *(С)* представляет собой двудольный, направленный, помеченный, мультиграф (ориентированный граф, в котором существует пара вершин, соединённых между собой более чем одной дугой; вершинам присвоены символы латинского алфавита).

Его можно представить в виде четверки элементов (рис. 11.1 (а, б)):

$$C = (P, T, I, O),$$

где:

$P = \{p_1, p_2, \ldots, p_n\}$ – конечное множество позиций, $n \geq 0$.

$T = \{t_1, t_2, \ldots, t_m\}$ – конечное множество переходов, $m \geq 0$.

Множество позиций и множество переходов не пересекаются:

$$P \cap T = \varnothing,$$

где:

$I: T \rightarrow P^{\infty}$ – входная функция, отображение множества переходов во входные позиции,

$O: T \rightarrow P^{\infty}$ – выходная функция, отображение множества переходов в выходные позиции.

Позиция p_i – входная позиция перехода t_j, если $p_i \in I(t_j)$;

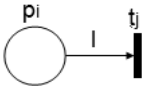


Рис. 11.1 (а). Входная позиция

p_i – выходная позиция перехода t_j, если $p_i \in O(t_j)$.

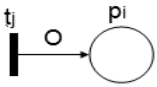


Рис. 11.1 (б). Выходная позиция

Входы и выходы переходов представляют собой комплекты позиций. Комплект является обобщением множества, в которое включены многократно повторяющиеся элементы – тиражированные элементы. Использование комплектов, а не множеств для входов и выходов перехода позволяет позиции быть кратным входом либо кратным выходом перехода.

Кратность входной позиции p_i для перехода t_i есть число появлений позиции во входном комплекте перехода, $\#(p_i, I(t_j))$.

Аналогично кратность выходной позиции p_i для перехода t_j есть число появлений позиции в выходном комплекте перехода, $\#(p_i, O(t_j))$.

Если входная и выходная функции являются множествами (а не комплектами), то кратность каждой позиции есть либо *0*, либо *1*.

11.3.2. Графическое представление сетей Петри

Структура сети Петри представляет собой совокупность позиций и переходов. В соответствии с этим граф сети Петри обладает двумя типами узлов (рис. 11.2). Кружок ○ является позицией, а планка | – переходом.

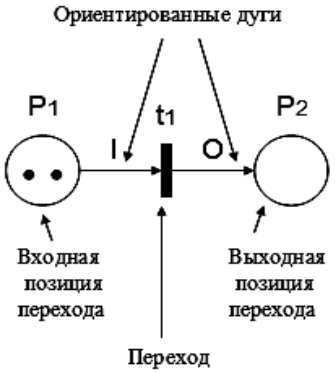


Рис. 11.2. Структура сети Петри

Ориентированные дуги (стрелки) соединяют позиции и переходы, при этом некоторые дуги направлены от позиций к переходам, а другие – от переходов к позициям. Дуга, направленная от позиции p_1 к переходу t_1, определяет позицию, которая является входом перехода. Кратные входы в переход указываются кратными дугами из входных позиций в переход. Выходная позиция указывается дугой от перехода к позиции. Кратные выходы также представлены кратными дугами.

11.3.3. Правила работы с сетями Петри

Выполнением сети Петри управляют количество и распределение фишек в сети. Сеть Петри выполняется посредством запусков переходов. Переход запускается удалением фишек из его входных позиций и образованием новых фишек, помещаемых в его выходные позиции.

Переход запускается, если он разрешен. Переход называется разрешенным, если каждая из его входных позиций имеет число фишек, по крайней мере, равное числу дуг из позиции в переход. Фишки во входной позиции, которые разрешают переход, называются его разрешающими фишками.

Для поддержки моделирования сетей Петри на компьютере разработан программный продукт ***HPSim***. Программа представляет сети типа позиция / переход, стохастические сети и сети Петри со временем. Моделирование может быть выполнено как один шаг или непрерывным способом.

11.4. Алгоритм работы операционной системы реального времени

11.4.1. Особенности структуры ОС РВ

В главе 10 работы [1] рассмотрена система программного управления транспортным роботом. Для управления всеми программными задачами в ней используется операционная система реального времени (ОС РВ).

Главная особенность ОС РВ состоит в том, чтобы заданный класс задач решался в реальном масштабе времени, т.е. результат решения был бы получен к заданному моменту времени. При управлении реальными инерционными объектами это требование приводит к дефициту времени.

Целесообразно все задачи разделить на задачи реального времени и фоновые задачи [9], выполнение которых не критично во времени. Работа всей программной системы осуществляется циклично во времени, интервал которого задается прерываниями от таймера. Из-за дефицита времени в каждом таймерном периоде необходимо решать только те задачи, которые необходимы в этот момент.

Нас интересуют лишь принципиальные особенности построения системы управления программами. Наиболее существенные особенности коснутся этапа загрузки системы и выхода из нее и организация управления программными задачами с учетом наличия режима реального времени [1, рис. 10.16 и 10.17]. Управлением задачами будет заниматься пла-

нировщик, состоящий из двух диспетчеров – реального времени и фонового.

Тогда интересующую нас структуру можно изобразить в следующем виде (рис. 11.3). В основе системы положен *монитор*, который является основной, управляющей частью.

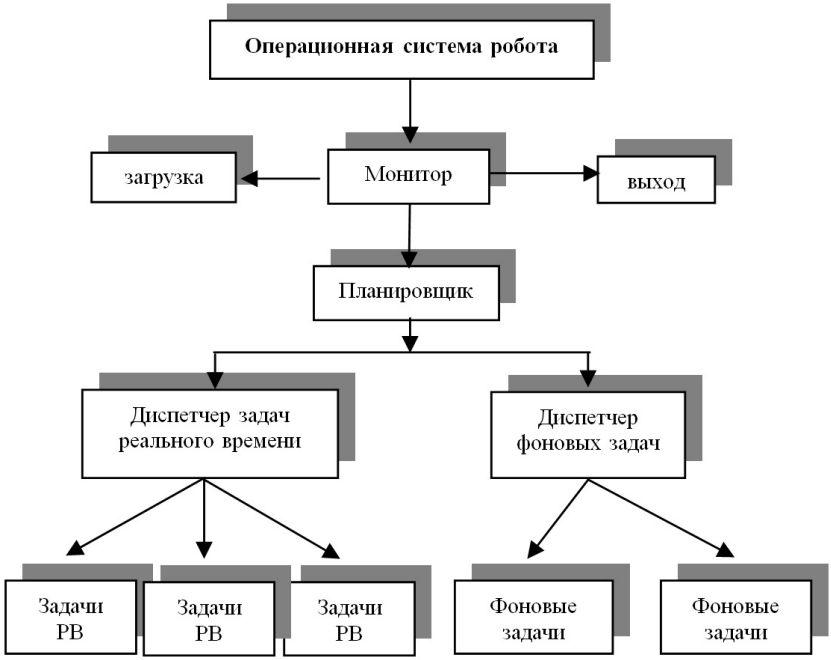


Рис. 11.3. Оригинальная часть операционной системы

11.4.2. Алгоритм работы монитора ОС РВ

Монитор должен обеспечивать следующие состояния операционной системы (рис. 11.3):

- загрузку системы,
- работу планировщика,
- выход из системы.

Рассмотрим алгоритм работы монитора (рис. 11.4).

После запуска монитора проводится загрузка ОС РВ, которая начинается со смены системных векторов прерывания исходной ОС [1], затем создаются необходимые буфера для команд и различных данных. Следующим этапом вводятся различные параметры и установки для технологического объекта.

При запуске системы монитор реализует загрузку операционной системы, а затем запускает планировщик. Последний через соответствующие диспетчеры управляет задачами РВ и фоновыми. По команде «Конец работы» монитор должен обеспечить выход из системы реального времени.

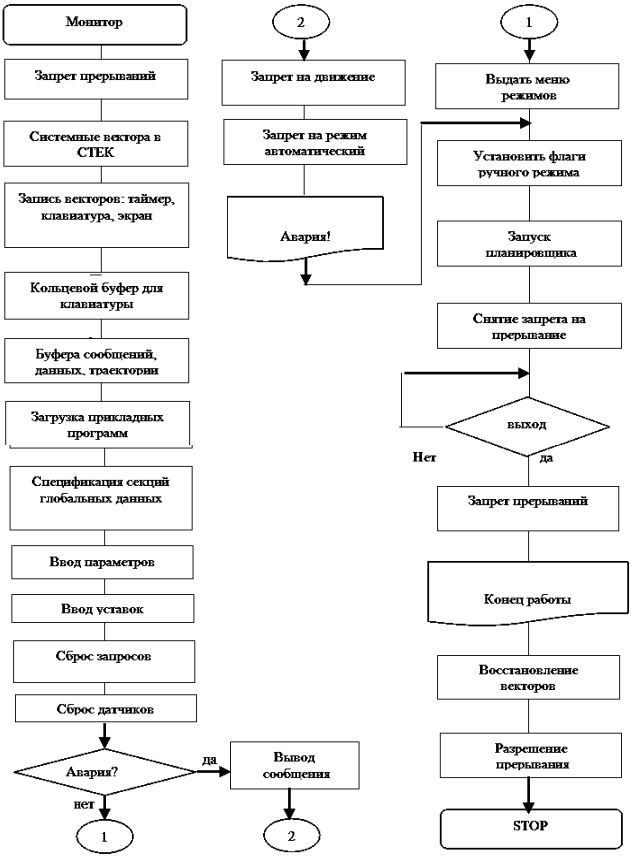


Рис. 11.4. Алгоритм работы монитора

Использован принцип семафоров. Его суть: пользователи, внешние устройства и задачи могут проявлять инициативу и выставлять запросы на обслуживание, т.е. на выполнение необходимых программ. Роль монитора сводится к тому, чтобы эти запросы удовлетворить в соответствии с приоритетами. В мониторе заложен алгоритм не функционирования всей системы управления, а только **алгоритм обслуживания запросов**. Следовательно, работа монитора слабо зависит от конкретного объекта управления для достаточно широкого класса объектов, что придает весьма значительную гибкость системе.

Действительно, для изменения алгоритма функционирования объекта достаточно изменить наполнение соответствующих прикладных задач или изменить последовательность их исполнения за счет изменения приоритетов. Такой подход используется в сложных универсальных системах или в системах автоматического управления.

Завершается загрузка анализом аварийных ситуаций. Если аварии нет, то запускается таймер и начинается циклическая работа ОС РВ.

Планировщик начинает работать по таймерной отметке, запуская, диспетчер реального времени (рис.11.5), работа которого проходит при установленном запрете на прерывания. Поскольку диспетчер не знает, какая конкретно задача снимается с выполнения, он должен сохранить в стеке содержимое всех регистров.

В качестве рабочих регистров выделяем планировщику регистры:

- R0 для хранения адресов паспортов задач,
- R1 назначаем счетчиком обработанных задач,
- R2 используем как адресный регистр для подпрограмм.

После занесения в регистр R0 адреса паспорта первой задачи РВ и в регистр R1 число задач реального времени N, производится проверка наличия флага запроса по адресу (R0). Если флаг сброшен, требуется перейти к следующему паспорту, для чего выполняется операция:

$$R0 + 4 \rightarrow R0.$$

В противном случае диспетчер должен организовать запуск задачи РВ, но при этом необходимо сохранить в стеке содержимое регистров R0 и R1.

При инициализации задачи диспетчер снимает флаг запроса, заносит в регистр R2 адрес задачи и запускает ее как подпрограмму. Задачи РВ решаются в текущем цикле управления до конца и по последней команде «выход из прерывания» возвращают управление диспетчеру РВ, восстановив из стека содержимое регистров R0 и R1. Последней задачей, запускаемой каждый цикл, является диспетчер фоновых задач, структурно составляющий вторую часть планировщика (рис. 11.6).

Работа диспетчера ФЗ во многом схожа с работой диспетчера РВ: запуск фоновых задач производится аналогично задачам РВ. Прежде всего, проверяется признак инициализации ФЗ, установленного в предыдущем цикле по адресу (R0). Если флаг установлен, диспетчер ФЗ должен обеспечить продолжение, запущенной ранее ФЗ.

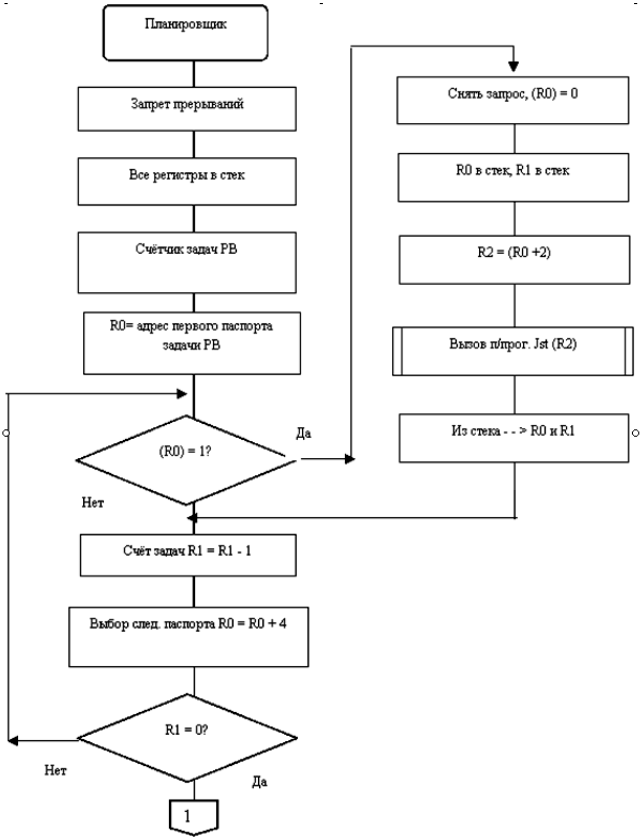


Рис. 11.5. Алгоритм планировщика, Диспетчер РВ

По окончанию работы текущей ФЗ проверяется состояние счетчика фоновых задач R1. Если R1 = 0, т.е. все задачи обработаны, планировщик переводит систему в состояние ожидания прерывания от таймера или ввода команды «конец работы».

Более подробное описание можно найти в книге [9].

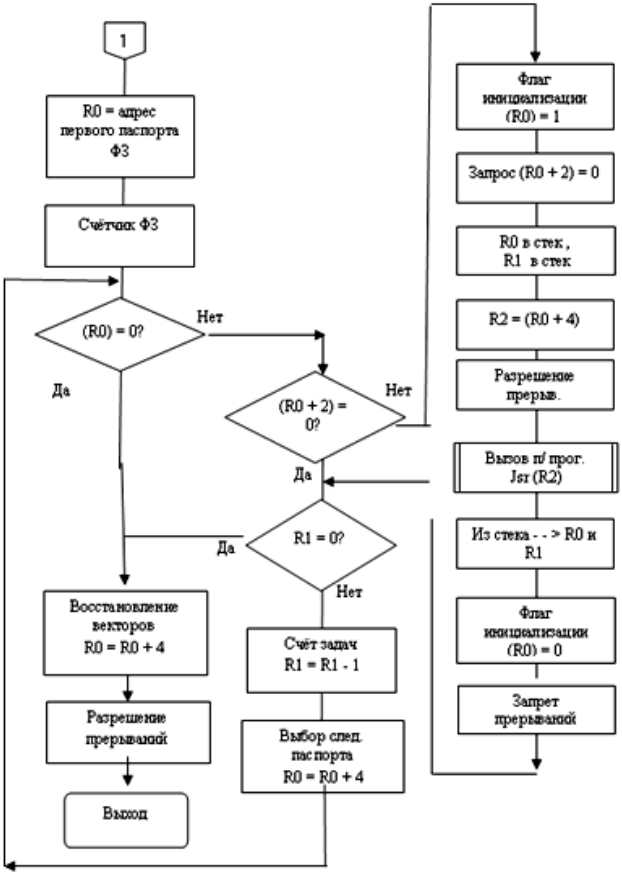


Рис. 11.6. Алгоритм планировщика, Диспетчер ФЗ

11.5. Учебное задание

11.5.1. Экспериментальное исследование влияния свойств вершин

Свойства переходов (Time Mode) определяет динамические свойства переходов во времени:

- Immediate (немедленный, мгновенный),

- Deterministic (детерминированный),
- Exponential (экспоненциальный, стремительный),
- Uniform Distraction (нормальный закон распределения).

Свойства позиций определяют исходные ресурсы и их изменение во времени:

- Initial Tokens – количество фишек в начале моделирования,
- Current Tokens – текущее количество фишек,
- Capacity –допустимая емкость позиции: максимальное количество фишек,
- Tokens Count – счётчик фишек, проходивших через позицию, при моделировании.

Методика исследований свойств переходов. Определить экспериментальным способом, как будет влиять на данную схему изменение свойств переходов (T0 и T1), для этого необходимо в вершину P0 установить 30 меток (рис. 11.7). Метки через два перехода T0 и T1 поступают в соответствующие вершины P2 и P3. Изменяя свойства переходов T0 и T1, проанализируйте характер движения меток.

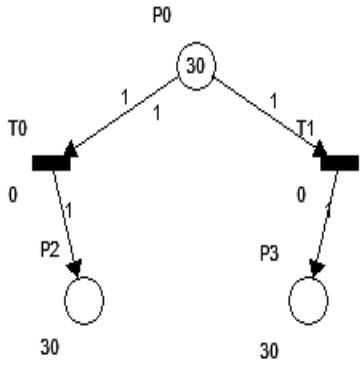


Рис. 11.7. Схема для исследования свойств вершин

Исследование свойств дуг. Требуется экспериментальным способом определить, как будет влиять на данную схему, изменение свойств одной из дуг (P2), для этого необходимо в вершину P0 установить 30 меток (рис. 11.8). Метки через переход T0 поступают в соответствующие вершины P1 и P2. Изменяя свойство перехода T0, проанализируйте характер движения меток. Результат исследований изложите в отчете.

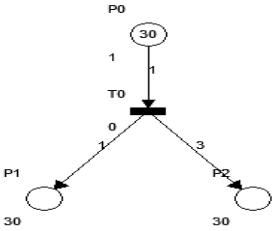


Рис. 11.8. Схема для исследования свойства дуг

11.5.2. Экспериментальное исследование счетчика по модулю два

Счетчик по модулю два реализуется в виде ***триггера*** – устройства с двумя устойчивыми состояниями 0 и 1. В исходном состоянии на выходе счетчика – **0**, при подаче на вход сигнала, он переходит в состояние **1**. при подаче еще одного сигнала, счетчик переходит в состояние **0**.

1. Постройте в программном пакете ***HPSim*** граф, представленный на рис. 11.9 и ответьте на вопрос: какие элементы графа описывают работу счетчика по модулю 2?

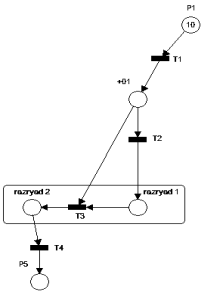


Рис. 11.9. Счётчик по модулю 2

2. На примере графа, разберитесь с принципом работы счётчика по модулю 2 в двоичной системе.

3. Опишите в отчете, как он работает по шагам.

4. Определите, что функционально представляет граф на рис. 11.9.

5. Используя счётчик по модулю 2, собрать модель одноразрядного восьмеричного счётчика.

11.5.3. Моделирование операционной системы реального времени

Задание

1. В графе *Алгоритм монитора* (рис. 11.10), представленном в сетях Петри, расставить метки, таким образом, чтобы данная система работала в соответствии с ее описанием [9].

2. Описать из каких частей она состоит и как работает.

3. Разобраться, как работает модель *Планировщика*.

4. Как и при каких условиях осуществляется запуск *Планировщика*?

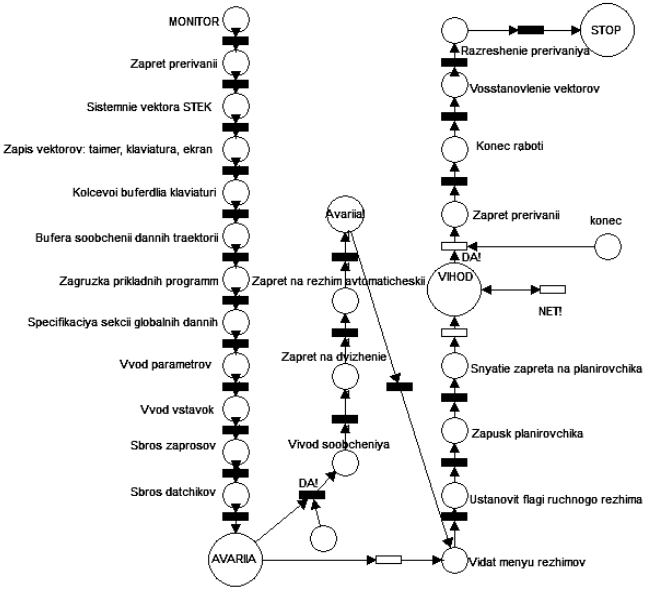


Рис. 11.10. Алгоритм монитора в сетях Петри

На рис. 11.11 представлен граф, описывающий *монитор, планировщик задач* и *диспетчер реального времени*.

5. При каких условиях запускается диспетчер РВ?

6. Разобраться, как работает модель диспетчера РВ.

7. Расставить метки так, чтобы запустилось две задачи реального времени.

8. Когда все задачи обработаны, до какого момента планировщик будет находиться в состоянии ожидания?

9. Как организовано состояние ожидания планировщика?

10. По какому условию начинается таймерный цикл?

11. Что происходит в графе, если все ЗРВ не успевают обработаться в текущем цикле?

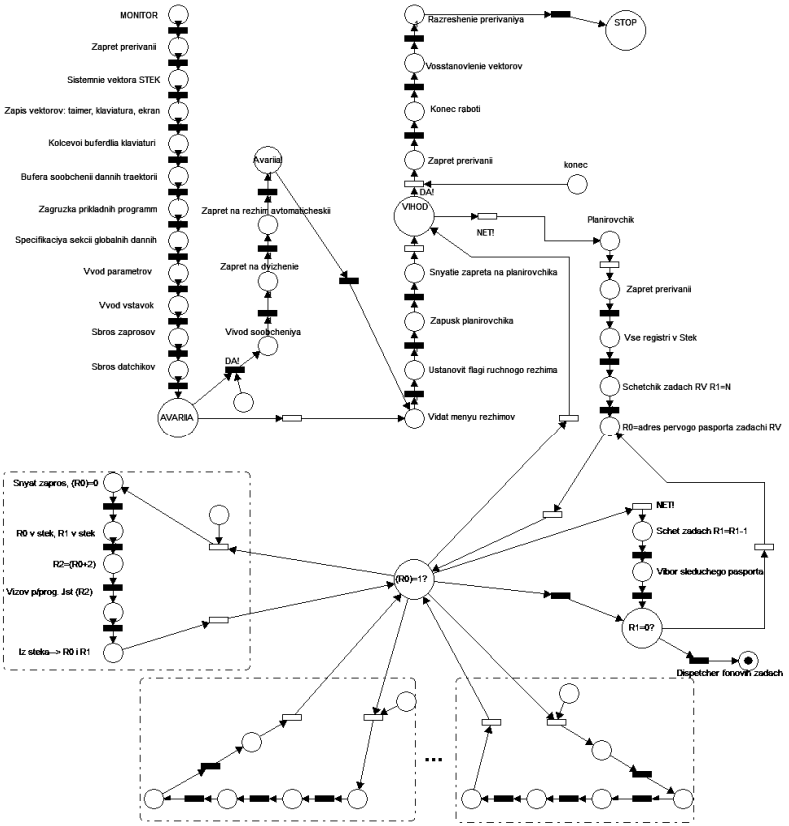


Рис. 11.11. Алгоритм монитора, планировщика, диспетчера РВ

На рис. 11.12 представлен граф **диспетчера фоновых задач**.

12. Разобраться, как работает модель диспетчера ФЗ.

13. На основе графа синтезировать и нарисовать блок-схему диспетчера ФЗ.

14. Для чего используются метки 1, 2, 3, 4, 5, 6?

15. Что характеризует их значение?
16. Что будет, если количество запрошенных задач будет меньше общего числа ФЗ?
17. Что будет, если количество запрошенных задач будет больше общего числа ФЗ?
18. Что происходит в графе, если все ФЗ не успевают обработаться в текущем цикле?

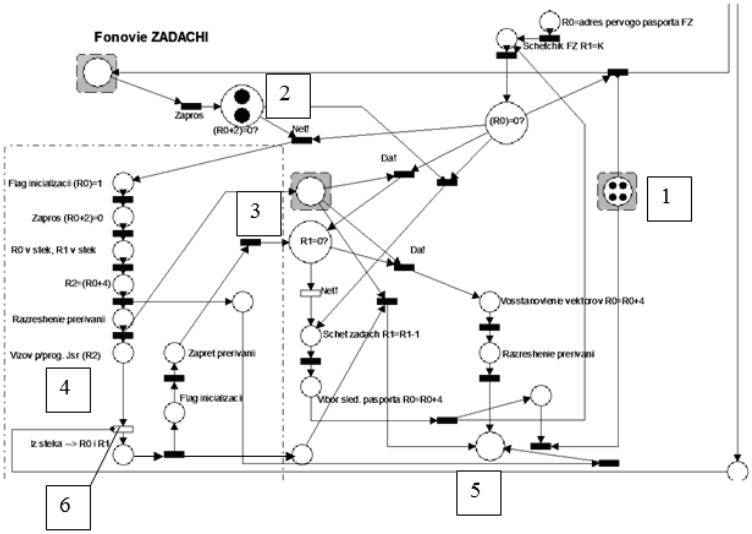


Рис. 11.12. Алгоритм работы диспетчера ФЗ

На рис. 11.13 представлен граф, описывающий Монитор, Планировщик, ДРВ и ДФЗ в двухпроцессорной системе.

19. Разобраться, как работает данная модель.
20. Как в графе отражено наличие второго процессора?
21. В чём преимущество использования двухпроцессорных систем над однопроцессорной?
22. Какие задачи (ЗРВ или ФЗ) обслуживает второй процессор в данном графе?
23. Какие задачи рациональнее поручить второму процессору? Почему?

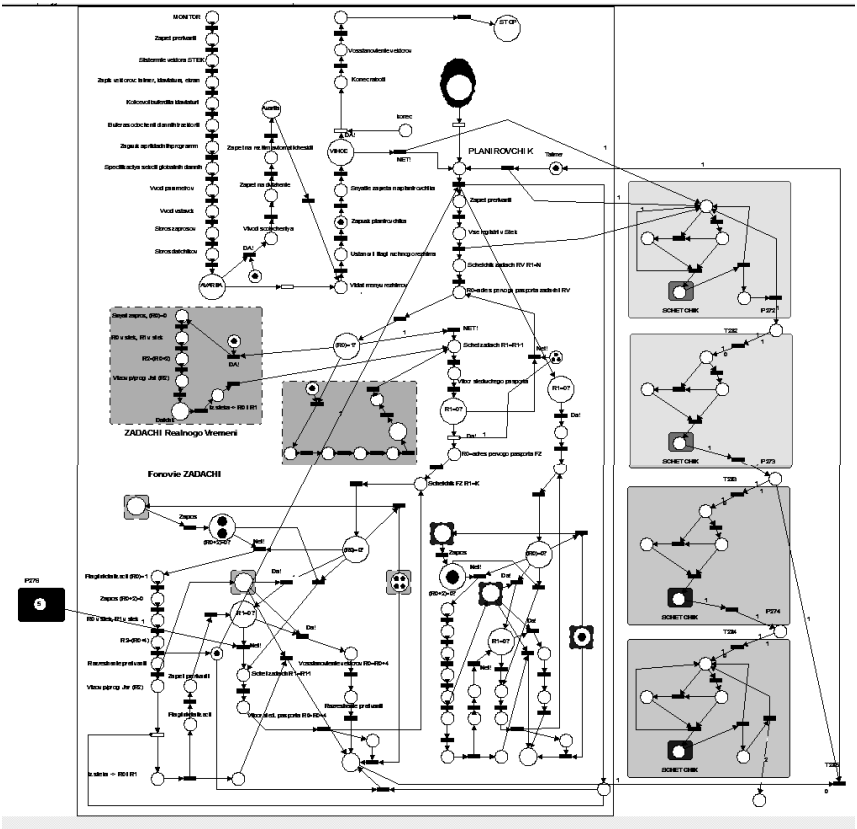


Рис. 11.13. Полная схема. Алгоритм работы двухпроцессорного планировщика, диспетчера РВ и диспетчера ФЗ

11.6. Пакет HPSim 1.1. для моделирования в сетях Петри

Процесс имитационного моделирования предназначен для обеспечения функционирования сложной динамической системы заданной структуры, накопления статистической информации в процессе работы, подготовки данных для анализа, а также для динамического отображения информации о моделировании на экран в режиме, установленном пользователем.

Программный продукт HPSim был разработан для поддержки моделирования и проектирования сетей Петри. Программа представляет сети типа позиция / переход и сети Петри со временем.

Размер и сложность разрабатываемой сети Петри ограничены только возможностями используемого компьютера. Например, объем памяти даже в 128 MB позволяет редактировать и имитировать проекты с несколькими сетями и каждой сетью в 1000 объектов.

Пакет моделирования сетей Петри обеспечивает следующие функциональные возможности:

- Автоматизированное построение имитационных моделей сложных объектов с помощью графического редактора и конструктора сетей.
- Проведение имитационного моделирования в режимах пошагового исполнения (для отладки моделей) и сбора информации для статистики.
- Формальный анализ и диагностирование систем, представленных сетями Петри, и исследование их структурных и поведенческих свойств.

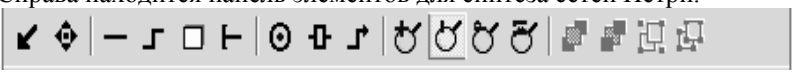
Запускается эмулятор сетей Петри с помощью файла *HPSim.exe*. Затем необходимо или открыть нужный нам прикладной файл с расширением *.hps*, или создать новый. Для запуска загруженной программы требуется нажать на иконку  (*ON/OFF*). А затем закрыть выскочившее окно. Нажать на кнопку *Step*  для пошагового моделирования или нажать на кнопку *Run*  для непрерывного моделирования.

Чтобы вернуть фишки в исходное состояние используется команда *Reset*  .

Для временной остановки при моделировании используется команда *Pause*  .

Элемент  используется для регулировки скорости моделирования.

Справа находится панель элементов для синтеза сетей Петри.



Слева находится панель, отражающая свойства элемента. Например, свойства позиции представлены окном:

Property	Value
Name	P1
Size	Normal
Show Name	TRUE
Show Capacity	FALSE
Initial Tokens	1
Current Tokens	0
Capacity	2
Tokens Count	11

Здесь:

Size – размер позиции.

Show Name – показать название позиции.

Show Capacity – показать текущее состояние позиции.

Initial Tokens – количество фишек в начале моделирования.

Current Tokens – текущее количество фишек.

Capacity – вместимость (сколько вмещает в себя позиция).

Tokens Count – счетчик фишек, проходивших через позицию при моделировании.

Свойства перехода:

Property	Value
Name	T5
Size	Normal
Show Name	TRUE
Show Delay	FALSE
Time Mode	Immediate
Initial Delay	0
Range Delay	0
Current Delay	0
Tokens Fired	1

Здесь:

Show Delay – задержка времени.

Time Mode – режим времени перехода.

Initial Delay – начальная задержка.

Range Delay – задержка диапазона.

Current Delay – текущая задержка.

Задержка используется для сетей Петри с учетом текущего времени.

Tokens Fired – сколько фишек прошло через переход.

11.7. Контрольные вопросы

1. Что представляют собой сети Петри?
2. Что позволяют исследовать сети Петри?
3. Дать определение графа.
4. Типы узлов сети Петри, их соединительные элементы.
5. Программный продукт, использующийся для поддержки моделирования сетями Петри.
6. Перечислить основные части Операционной системы.
7. Какие состояния Операционной системы должен обеспечивать монитор?
8. Из каких частей состоит планировщик?
9. В каких состояниях могут находиться задачи РВ?
10. В каких состояниях могут находиться ФЗ?
11. Перечислить задачи РВ.
12. Перечислить ФЗ.
13. Что такое ***прерывание***?
14. Что дает использование механизма прерываний?
15. Что такое ***Счётчик команд***?
16. Что такое ***Регистр флагов***?
17. Как организуются прерывания?

12. УПРАВЛЕНИЕ ПИТАНИЕМ СВЕТОДИОДА С ПОМОЩЬЮ МИКРОКОНТРОЛЛЕРА ATMEGA8

Лабораторная работа № 12

12.1. Цель работы

Цель – получить первоначальные навыки программирования работы микроконтроллера (МК).

12.2. Задания к лабораторной работе

1. Изучить теоретические основы архитектуры микроконтроллера Atmega8.
2. Изучить основы программирования работы его портов.
3. Научиться создавать виртуальные модели в симуляторе электрических цепей **Proteus**, создавать проект в среде программирования **AVR Studio**.
4. Разработать программу управления светодиодом с помощью функции задержки ***delay*** и протестировать ее работу на виртуальной модели.
5. Собрать реальную схему на макетной плате, запрограммировать МК и проверить ее работу.
6. Ответить на контрольные вопросы преподавателя.
7. написать РУКОПИСНЫЙ отчет по пп. 1–5 с обязательным рассмотрением допущенных ошибок и анализа работы синтезированных программ.

12.3. Ход работы

1. Изучить теоретический материал.
2. На изложенном ниже примере, разобраться в работе AVR Studio.
3. Внимательно изучить пример программы.
4. Выполнить следующее задание: написать программу, реализующую мигания светодиодом с заданной частотой, используя команду delay.
5. Отчет по лабораторной работе должен содержать:

- цель работы;
- основы теории;

- условие индивидуального задания;
- листинг программы с комментариями;
- описание методики и результаты проверки правильности функционирования программы (в какой последовательности подавались входные сигналы, что визуально наблюдалось при этом и т.п.);
- выводы по работе.

12.4. Микроконтроллер, основные понятия и назначение портов ввода/вывода

Микроконтроллер (англ. Micro Controller Unit, MCU) – микросхема, предназначенная для управления электронными устройствами (рис. 12.1).

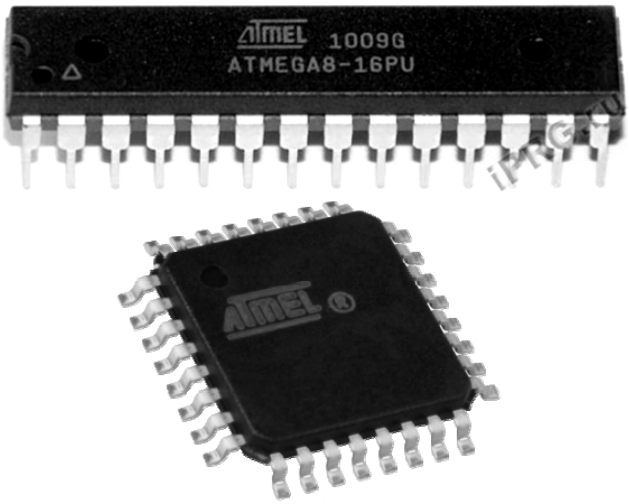


Рис. 12.1. Микроконтроллер **Atmega 8**

Типичный микроконтроллер сочетает на одном кристалле функции процессора и периферийных устройств, содержит ОЗУ и (или) ПЗУ. По сути, это миниатюрный однокристальный компьютер, способный выполнять простые задачи.

Микроконтроллер фирмы **AVR** содержит: быстрый **RISC**-процессор, два типа энергонезависимой памяти (**Flash**-память программ и память

данных **EEPROM**), оперативную память RAM, порты ввода/вывода и различные периферийные интерфейсные схемы [17–18]. В этой работе мы рассмотрим порты ввода/вывода микроконтроллера *ATMEGA8* (рис. 12.1).

Порты микроконтроллера – это устройства ввода/вывода, позволяющие микроконтроллеру передавать или принимать данные. Стандартный порт микроконтроллера AVR имеет восемь разрядов данных, по которым информация может передаваться или приниматься в параллельном режиме. Для обозначения портов используются латинские буквы A, B, C и т.д. Количество портов ввода/вывода варьируется в зависимости от модели микроконтроллера.

Любой порт микроконтроллера можно сконфигурировать как вход или как выход. Для того чтобы это сделать, следует записать в соответствующим порту регистра DDRx соответствующее значение. Кроме того, в качестве входа или выхода можно сконфигурировать отдельно любой вывод (пин) порта. В любом случае, хотите вы сконфигурировать весь порт или отдельный вывод, вам необходимо будет работать с регистрами[1] DDRx, PORTx, PINx.

DDRx – регистр направления передачи данных. При этом разряды любого регистра обычно нумеруются справа налево, начина нумерацию с нуля. Каждому разряду (или биту) соответствует вывод (ножка) микроконтроллера. Этот регистр определяет, является тот или иной вывод порта входом или выходом. Ножки микроконтроллера также называют пинами. Если некоторый разряд регистра DDRx содержит логическую единицу, то соответствующий вывод порта сконфигурирован как выход, в противном случае – как вход. Буква x в данном случае должна обозначать имя порта, с которым вы работаете. Таким образом, для порта A это будет регистр DDRA, для порта B – регистр DDRB и т.д.

Используя компилятор AVR GCC, входящий в состав среды программирования AVR Studio, записать в необходимый регистр то или иное значение можно одним из следующих способов. Для всего порта сразу:

DDRD = 0xff,

где 0xff – шестнадцатеричное представление числа ff, где 0x является префиксом, используемым для записи шестнадцатеричных чисел. В десятичном представлении это будет число 255, а в двоичном виде оно будет выглядеть как 11111111. То есть во всех битах регистра DDRD будут записаны логические единицы. При получении первых навыков програм-

[1] Регистр – последовательное или параллельное логическое устройство, используемое для хранения n-разрядных двоичных чисел и выполнения преобразований над ними.

мирования рекомендуется использовать двоичное представление чисел, где каждый разряд числа соответствует разряду регистра. Опытный программист использует шестнадцатеричное представление в силу краткости записи.

В AVR GCC для представления двоичных чисел используется префикс 0b. Таким образом, число 11111111 должно представляться в программе как 0b11111111. Мы можем записать предыдущую команду в более читабельном виде.

DDRD = 0b11111111.

Это означает, что все выводы порта D будут сконфигурированы как выходы, так как каждому разряду порта соответствует логическая единица.

Для того чтобы сконфигурировать все выводы порта D как входы, следует записать во все биты регистра DDRD логические нули.

DDRD = 0x00 или DDRD = 0b0000000.

В регистр DDRD можно записать и другие числа. Например:

DDRD = 0xb3 или DDRD = 0b10110011,

где 0xb3 – шестнадцатеричное представление числа 179. В двоичном виде оно будет выглядеть как 10110011. То есть часть выводов порта D будет сконфигурирована как выходы, а часть – как входы:

PD0 – 1 (выход)
PD1 – 1 (выход)
PD2 – 0 (вход)
PD3 – 0 (вход)
PD4 – 1 (выход)
PD5 – 1 (выход)
PD6 – 0 (вход)
PD7 – 1 (выход)

Каждый бит регистров DDRx может быть установлен отдельно. Например, чтобы сконфигурировать отдельно вывод PD2 как выход, нам необходимо в соответствующий бит регистра DDRD записать 1. Для этого применяют следующую конструкцию языка программирования C++.

DDRD |= 1<<2;

где 1<<2 – осуществляет сдвиг единички влево на 2 бита, то есть справа добавляются два нулевых бита и получается 100, а знак «|», стоящий перед знаком присваивания «=», осуществляет операцию побитного логического сложения. При логическом сложении 0+0=0, 0+1=1, 1+1=1. Операцию логического сложения по-другому называют операцией ИЛИ (английское название OR).

Таким образом, к битам, хранящимся в регистре DDRD, прибавляется двоичное 100, представленное в 8-битном регистре микроконтроллера как 00000100, и результат записывается обратно в регистр DDRD.

Чтобы сконфигурировать отдельно вывод PD2 как вход, нам необходимо в соответствующий бит регистра DDRD записать 0. Для этого применяют следующую конструкцию:

DDRD &= ~(1<<2).

В данном случае результат сдвига единицы на две позиции влево инвертируется с помощью операции побитного инвертирования, обозначаемой значком «~». При инверсии мы получаем вместо нулей единички, а вместо единичек – нули. Эта логическая операция иначе называется операцией НЕ (английское название NOT). Таким образом, при побитном инвертировании 00000100 мы получаем 11111011.

Получившееся число с помощью операции побитного логического умножения & умножается на число, хранящееся в регистре DDRD, и результат записывается в регистр DDRD. При логическом умножении 0*0=0, 0*1=0, 1*1=1. Операцию логического умножения иначе называют операцией И (английское название AND). То есть сдвинутая нами влево на две позиции единичка превращается при инвертировании в ноль и умножается на соответствующий бит, хранящийся в регистре DDRD. При умножении на ноль мы получаем ноль. Таким образом, бит PD2 становится равным нулю.

Кроме логических операций И, ИЛИ, НЕ существует также операция «исключающее ИЛИ» (английское название XOR). Она обозначается значком ^. При исключающем ИЛИ значение бита, к которому «прибавляется» единичка, изменяется на противоположное. Например,

110011 ^ 011010 = 101001.

Следует добавить, что работа с числами в 8-битном микроконтроллере проходит с использованием 8-битных регистров. Перед вычислениями аргумент помещается в один из специальных регистров, с которыми напрямую может работать арифметико-логическое устройство (АЛУ). Например, перед выполнением команды DDRD &= ~(1<<2) аргумент помещается во вспомогательный регистр микроконтроллера. Содержимое такого регистра будет выглядеть как 11111011. После этого осуществляется операция побитного умножения, что дает во втором бите регистра DDRD значение 0.

Регистр **PORTx**, управляет состоянием выводов порта «x». В зависимости от выбранного направления работы (вход или выход) порта или отдельно взятого вывода (при помощи регистра DDRx), значение записанное в регистр PORTx того же порта, может присваивать выводу раз-

личные состояния. Логические элементы оперируют сигналами двух типов: «высокий логический уровень» (1) и «низкий логический уровень» (0), которые характеризуются различным уровнем напряжения: полное напряжение питания принимается в качестве уровня «логической единицы» (для Atmega8 оно составляет интервалу 3,5-5V), а нулевое напряжение (0-0,5V) – в качестве уровня «логического нуля».

Когда вывод настроен как выход, то любое значение, записанное в соответствующий бит регистра PORTx, поступает на выход:

- PORTxn = 1 – состояние вывода соответствует лог. единице,
- PORTxn = 0 – состояние вывода соответствует лог. нулю,

т.е. состояние вывода (когда он настроен на выход) может изменяться только между лог. единицей и лог. нулем.

Когда вывод настроен как вход, то значение, поступающие на него, записывается в соответствующий бит регистра PORTx. В этом режиме, текущее состояние выводов, можно прочитать при помощи регистра PINx этого же порта. Чтение состояния вывода можно производить при любых настройках вывода: будь то вход, выход или альтернативная функция вывода. Регистр PINx можно только читать.

Функция delay_ms() приостанавливает исполнение программы на величину времени time, заданного в миллисекундах. Описание работы функции _delay_ms() содержится в библиотеке delay.h, поэтому нам будет необходимо подключить ее к программе.

12.5. Практическая часть

В данном разделе описывается процесс создания виртуальной схемы, состоящей из МК Atmega8, к одному из портов которого подключен светодиод.

Моделирование схемы в Proteus.

Для того чтобы начать нашу работу нам необходимо смоделировать схему в симуляторе электрических цепей Proteus. Симулятор позволяет нам без сборки реального устройства отладить работу схемы, найти ошибки, полученные на стадии проектирования, снять необходимые характеристики и многое другое, что поможет избежать ошибок на практике в виде сгоревших деталей.

Собираем нашу схему. Открываем программу и создаем новый проект, используя меню FILE > NEW DESIGN. Это можно не делать, если вы только что открыли программу, так как при запуске PROTEUS автоматически создаст новый проект с именем UNTITLED.DSN – безымянный. Установим для удобства свои размеры листа схемы. Откроем меню SYS-

TEM > SET SHEET SIZE. Выберем вариант USER, в окошках введем 6 in 4 in (высота и ширина в дюймах). После этого нажмите F8, чтобы уместить размер листа схемы под окно редактирования.

Соберем схему согласно рис. 12.2.

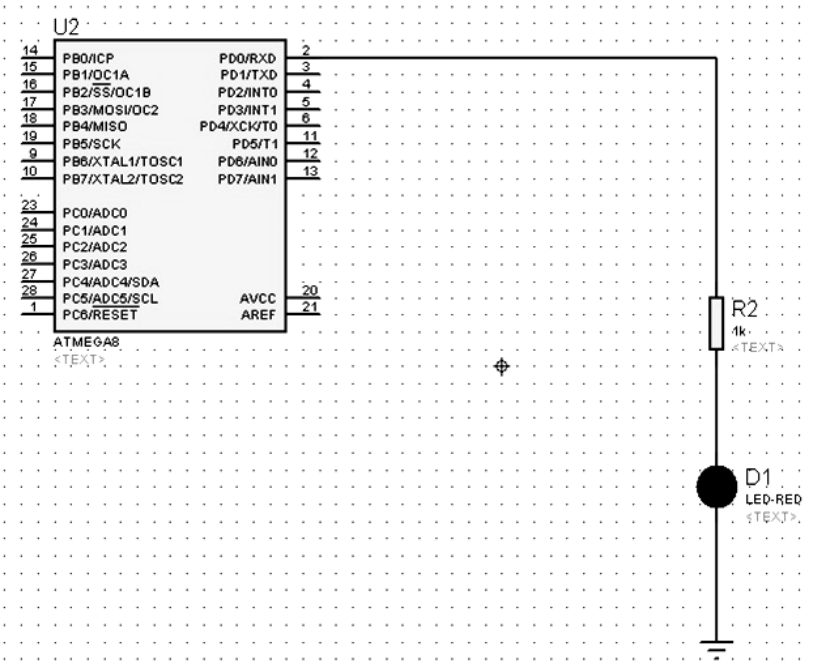


Рис. 12.2. Схема для управления светодиодом

Итак, нам нужны один микроконтроллер ATmega 8, один светодиод (берем красный) и один резистор на 0,5 кОм. Откройте библиотеку компонентов (рис. 12.3). Далее нажав на пиктограмму P, либо дважды щелкнув левой кнопкой по полю объектов, на экране появится окно со списком имеющихся компонентов. Чтобы долго не искать, наберите в окне KEYWORDS Atmega8. Теперь либо дважды нажав enter, но тогда закроется окно и придется заново его открывать, либо дважды щелкнув левой кнопкой по строке с описанием компонента, появившейся в окне results, вы переместите выбранный вами компонент в список Object Selector. Выберите подобным образом резистор, набрав RES и светодиод LED-RED.

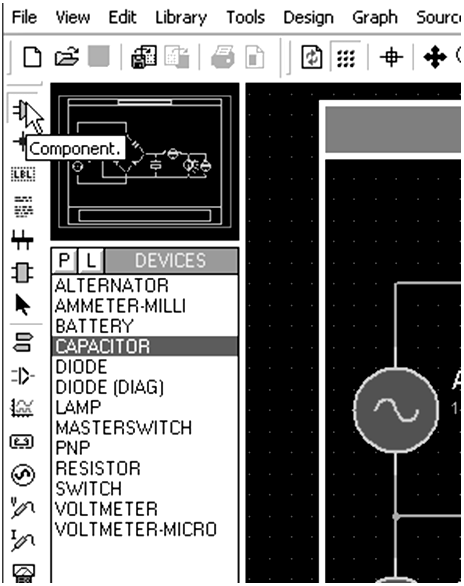


Рис. 12.3. Библиотека компонент

Компоненты набираются по одному экземпляру, размножить их можно будет уже потом, просто выбирая в списке Object Selector. Закройте библиотеку, нажав OK или же закрыв окно. Если все прошло без ошибок, то разместим наши компоненты, щелкнув левой кнопкой сначала по названию компонента в списке, а затем в нужном нам месте на пустой еще схеме.

Для полноты электрической схемы нам не хватает еще двух важных элементов – «земля» (общий провод) и «питание» (источник электрической энергии). Элементы такого типа выбираются в режиме INTER SHEET TERMINAL (рис. 12.4).

Выберите элемент GROUND (земля) и поместите его на схеме под светодиодом. Теперь соедините компоненты между собой. Поместим курсор на верхний вывод сопротивления, на конце курсора должен появиться крестик, показывающий, что соединение возможно. Щелкнув левую кнопку, перетащим курсор на вывод МК PD0 (вывод № 2), при этом должна появиться тонкая линия, показывающая, что соединение возможно. Щёлкнем левой кнопкой еще раз. Аналогично соедините остальные элементы согласно схеме.

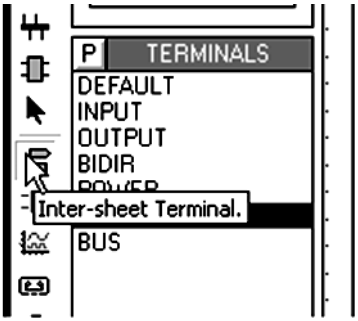


Рис. 12.4. Режим INTER SHEET TERMINAL

Измените сопротивление резистора на 0,5 кОм, вызвав меню Edit Components двойным щелчком по соответствующему элементу. Так же измените тип модели резистора на digital, чтобы симулятор не тратил время на обсчет аналоговых свойств резистора (рис. 12.5). Осталось запитать микроконтроллер, для этого необходимо к 20 выводу подсоединить источник энергии POWER, а к 21 – GROUND.

Однако при моделировании работы микроконтроллера в среде Proteus можно его не запитывать, так как программа сама смоделирует это действие. Сохраните проект в свою папку, чтобы не путаться под именем LED.DSN.

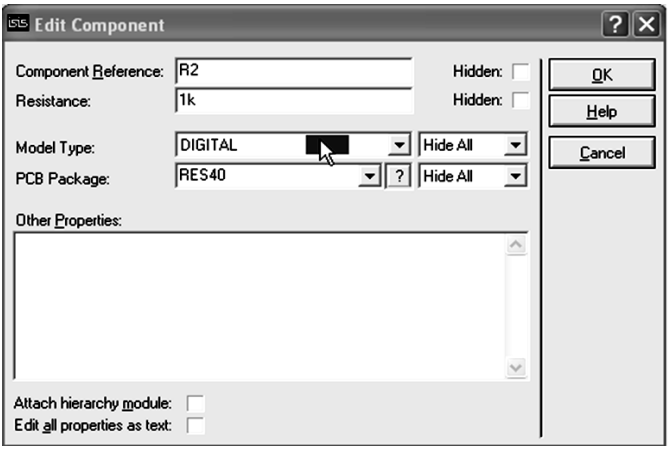


Рис. 12.5. Настройки резистора

12.5.1. Создание проекта в AVRStudio

Теперь, когда наша схема собрана, займемся написанием программного кода. Мигание светодиодом мы будем осуществлять через команду задержки, назовем эту программу *withdelay*. Открываем программу *AVRStudio* и создаем новый проект (рис. 12.6), выбираем *New Project* и нажимаем *Next*. На этом этапе (рис. 12.7) мастер проектов предлагает выбрать тип проекта (*Project type*), выберем *AVR GCC* и укажем имя проекта (*Project name*), в нашем случае (*withdelay*). Расширения для файлов указывать не надо – они присваиваются автоматически.

После того, как имя проекта задано – активируется кнопка *Next* >>, позволяя перейти к следующей странице мастера. Здесь предлагается выбрать отладочную платформу и используемый микроконтроллер (*Select debug platform and device*) (рис. 12.8). В окне слева перечислены все доступные платформы для отладки (*Debug platform*). Эмулятор (*AVR Simulator*) поддерживает практически все существующие микроконтроллеры, поэтому для начала следует пользоваться им.

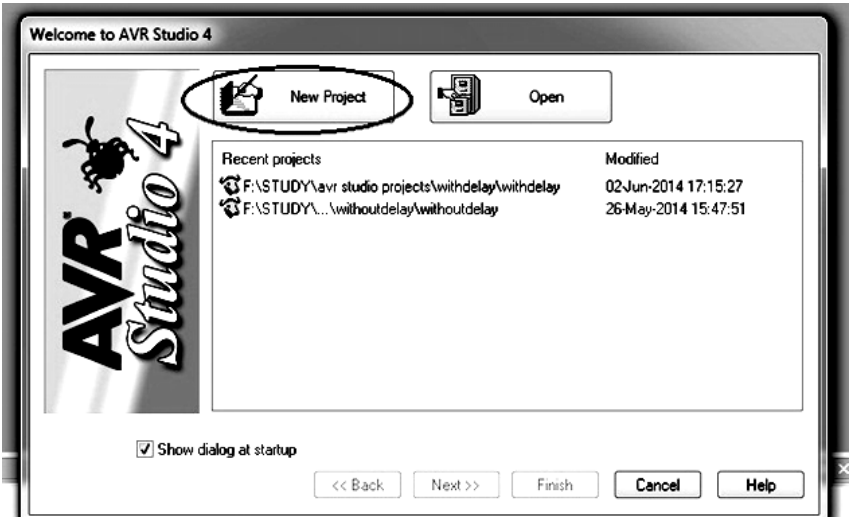


Рис. 12.6. Создание нового проекта в AVRStudio

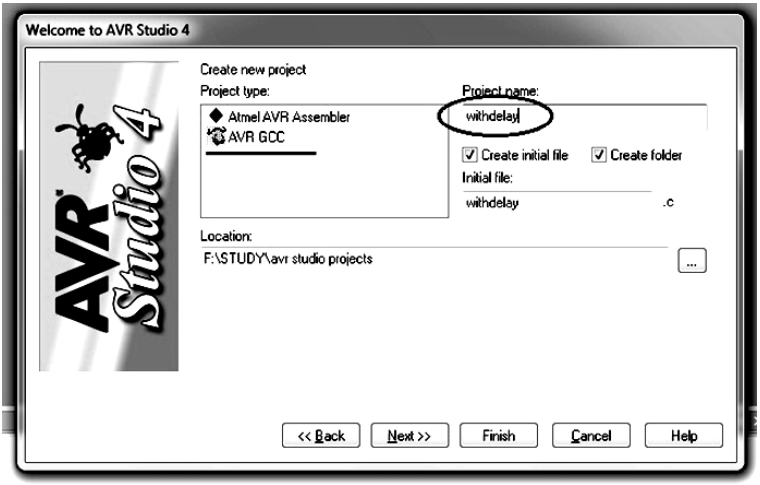


Рис. 12.7. Выбор типа и название проекта

Выбрав платформу и модель микроконтроллера, вы можете завершить работу мастера нажатием кнопки «Finish». Теперь мы видим окно, в котором и будем писать нашу программу (рис. 12.9).

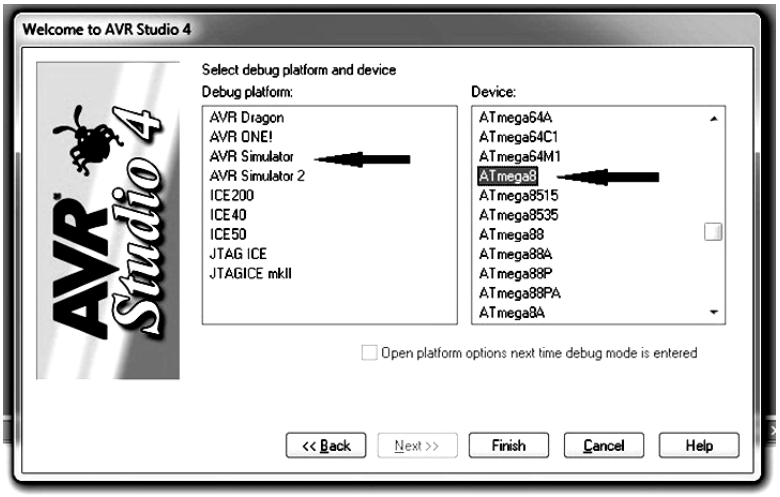


Рис. 12.8. Выбор отладочной платформы и микроконтроллера

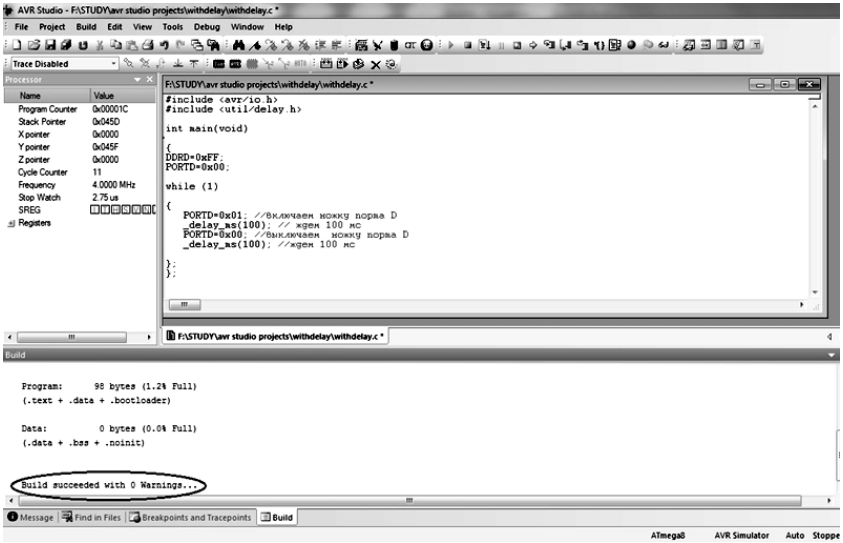


Рис. 12.9. Окно набора и компиляция программы

Листинг нашей программы с комментариями приведен ниже.

```
#include <avr/io.h>
#include <util/delay.h>
int main(void)
{
    DDRD=0xFF;          // настраиваем порт D на выход
    PORTD=0x00;         // при первом старте МК на всех выводах порта
                        // D будет логический ноль
    while (1)           //бесконечный цикл, “1” соответствует “true”
    { PORTD=0x01;       //устанавливаем логическую единицу на нулевом
                        // выводе порта D
      _delay_ms(1000);  //задержка 1000 мс
      PORTD=0x00;       //устанавливаем логический ноль
      _delay_ms(1000);  //задержка 1000 мс
}};
```

13. ИЗУЧЕНИЕ ПРОГРАММИРОВАНИЯ ТАЙМЕРОВ И СЧЕТЧИКОВ С ИСПОЛЬЗОВАНИЕМ СИСТЕМЫ ПРЕРЫВАНИЙ

Лабораторная работа № 13

13.1. Цель работы

Цель лабораторной работы – изучить организацию прерываний и работу таймеров / счетчиков в микроконтроллере ATmega8 на примере управления светодиодом через прерывания

13.2. Задания к лабораторной работе

1. Изучить теоретический материал.
2. На примере, изложенном ниже, разобраться в работе прерываний и таймеров / счетчиков.
3. Внимательно изучить пример программы.
4. Выполнить следующее задание: подключить второй светодиод другого цвета к схеме и реализовать поочередное мигание двух светодиодов с разной скоростью.
5. Отчет по лабораторной работе должен содержать:
    - цель работы;
    - условие индивидуального задания;
    - листинг программы на языке Си;
    - описание методики и результаты проверки правильности функционирования программы (в какой последовательности подавались входные сигналы, что визуально наблюдалось при этом и т.п.);
    - выводы по работе.

13.3. Теоритический материал

Применение функции задержки не позволяет нам полностью воспользоваться всеми возможностями микроконтроллера. На практике задачи, решаемые программой, выглядят немного сложнее. При формировании мигания светодиода через функцию delay, микроконтроллер не может выполнять больше никаких операций, весь его рабочий потенциал

уходит лишь на простое мигание, что не рационально. Хотя достаточно часто подобный способ задержки имеет право на жизнь, однако ещё чаще принято формировать интервалы при помощи таймеров [17, 18].

С целью улучшения работы микроконтроллера, мы организуем то же самое мигание светодиода, но через прерывания, с помощью таймера/счетчика. Способность таймеров вызывать прерывания по переполнению значения соответствующего регистра позволяет нам выполнять определённые действия через определённые интервалы времени.

В данном случае под прерыванием понимаем следующее. Останавливается обычный ход выполнения нашей программы и запускается на выполнение заданный код, с последующим возвратом в прерванную программу.

13.3.1. Таймеры/счетчики (TIMER/COUNTERS)

Микроконтроллеры AVR имеют в своем составе от 1 до 4 таймеров/счетчиков с разрядностью 8 или 16 бит, которые могут работать и как таймеры от внутреннего источника тактовой частоты, и как счетчики внешних событий.

Их можно использовать для точного формирования временных интервалов, подсчета импульсов на выводах микроконтроллера, формирования последовательности импульсов, тактирования приемопередатчика последовательного канала связи.

Таймеры/счетчики способны вырабатывать запросы прерываний, переключая процессор на их обслуживание по событиям и освобождая его от необходимости периодического опроса состояния таймеров. Поскольку основное применение микроконтроллеры находят в системах реального времени, таймеры/счетчики являются одним из наиболее важных элементов.

У таймера-счетчика есть **регистр TCNT**, из которого можно прочитать, сколько времени прошло с момента запуска таймера. Значение в этом регистре не в минутах или секундах, а в «тиках» таймера. Чему равен один тик – зависит от тактовой частоты, на которой работает микроконтроллер AVR, и от настроек таймера.

Так же у таймера-счетчика есть настраиваемый делитель частоты, который определяет, на какую величину будет поделена тактовая частота микроконтроллера перед тем, как будет подана на таймер-счетчик. Длительность тика таймера является обратной величиной от частоты, полученной в результате деления.

Например:

Тактовая частота МК=11059200 Гц (11.0592 МГц)

Делитель = 1024

Частота таймера = Тактовая частота/Делитель = 11059200/1024 = 10800 Гц.

Длительность периода таймера = 1/10800 = $92.593*10^{-6}$с = 92.593 мкс

Для настройки делителя используются биты CS регистра TCCR.

Например, для таймера 1 в atmega8 используются регистр TCCR1B и биты CS10, CS11, CS12.

Таблица 13.1

Зависимость делителя от состояния битов CS для таймера 1

S2	CS11	CS10	Делитель
0	0	0	0
0	0	1	1
0	1	0	8
0	1	1	64
1	0	0	256
1	0	1	1024

Пример кода настройки делителя для таймера 1 avr atmega8:

```
TCCR1B = (1<<CS12)|(0<<CS11)|(1<<CS10); // 1024
```

В avr таймеры могут быть 8 или 16 разрядными. Разрядность определяет максимальное количество тиков, которые может сосчитать таймер до переполнения. После этого сработает прерывание и таймер обнулится. Для 8-разрядного – это 256 (2^8) тиков, для 16 – это 65536 (2^{16}).

В нашем случае один тик равен 92.593 мкс, соответственно максимальное значение, которое может измерить 16-битный таймер, это $65536*92.593*10^{-6}$ = 6.068 с, 8-битный – 0.0237 с.

После того как таймер досчитает до максимального значения, он переполняется, т.е. начинает считать с нуля. Эту ситуацию можно обрабатывать при помощи прерываний. Для этого надо разрешить прерывание по переполнению таймера и выставить бит общего разрешения прерываний.

13.3.2. Система прерываний в МК

Система прерываний – одна из важнейших частей микроконтроллера. Все микроконтроллеры AVR имеют многоуровневую систему прерываний. Прерывание прекращает нормальный ход программы для выполнения приоритетной задачи, определяемой внутренним или внешним событием.

Для каждого такого события разрабатывается отдельная программа, которую называют подпрограммой обработки запроса на прерывание (для краткости – подпрограммой прерывания), и размещается в памяти программ.

При возникновении события, вызывающего прерывание, микроконтроллер сохраняет содержимое счетчика команд, прерывает выполнение центральным процессором текущей программы и переходит к выполнению подпрограммы обработки прерывания.

После выполнения подпрограммы прерывания осуществляется восстановление предварительно сохраненного счетчика команд и процессор возвращается к выполнению прерванной программы.

Для каждого события может быть установлен приоритет. Понятие приоритет означает, что выполняемая подпрограмма прерывания может быть прервана другим событием только при условии, что оно имеет более высокий приоритет, чем текущее. В противном случае центральный процессор перейдет к обработке нового события только после окончания обработки предыдущего.

Прерывания делятся на внутренние и внешние. К источникам внутренних прерываний относятся встроенные модули микроконтроллера (таймеры, приёмопередатчик USART). Внешние прерывания возникают при поступлении внешних сигналов на выводы микроконтроллера (например, сигналы на выводы RESET и INT). Характер сигналов, приводящих к возникновению прерывания задаётся в регистре управления **MCUCR**, в частности в разрядах ISC00 (бит 0) и ISC01 (бит 1) для входа INT 0; ISC10 (бит 2) и ISC11 (бит 3) для входа INT1.

Таблица 13.2

Настройка срабатывания прерывания INT0

ISC00	ISC01	Значение
0	0	Прерывание вызывается по уровню лог. 0 на входе INT0
0	1	Прерывание вызывается по ниспадающему фронту сигнала на входе INT0
1	1	Прерывание вызывается по возрастающему фронту сигнала на входе INT0

Таблица 13.3

Настройка срабатывания прерывания INT1

SC10	SC11	Значение
0	0	Прерывание вызывается по уровню лог. 0 на входе INT1
0	1	Прерывание вызывается по возрастающему фронту сигнала на входе INT1
1	1	Прерывание вызывается по ниспадающему фронту сигнала на входе INT1

Таблица 13.4

Векторы прерываний в Atmega8

Адрес	Источник прерывания	Описание
0x0000	RESET	Сигнал сброса
0x0001	INT0	Внешний запрос на прерывание по входу INT0
0x0002	INT1	Внешний запрос на прерывание по входу INT1
0x0003	T/C1	Захват по таймеру T/C1
0x0004	T/C1	Совпадение с регистром сравнения A таймера T/C1
0x0005	T/C1	Совпадение с регистром сравнения B таймера T/C1
0x0006	T/C1	Переполнение счётчика T/C1
0x0007	T/C0	Переполнение счётчика T/C0
0x0008	SPI	Передача данных по интерфейсу SPI завершена
0x0009	UART	Приём данных приемопередатчиком UART завершен
0x000A	UART	Регистр данных UART пуст
0x000B	UART	Передача данных приёмопередатчиком UART завершена
0x000C	ANA_COMP	Прерывание от аналогового компаратора

В микроконтроллере Atmega8 каждому прерыванию соответствует свой вектор прерывания (адрес в начале области памяти программ, в которой хранится команда для перехода к заданной подпрограмме обработки прерывания). В Atmega8 все прерывания имеют одинаковый приоритет.

В случае одновременного возникновения нескольких прерываний первым будет обрабатываться прерывание с меньшим номером вектора.

13.3.3. Управления прерываниями

За управление прерываниями в ATmega8 отвечают 4 регистра:

GIMSK (он же GICR) – запрет/разрешение прерываний по сигналам на входах INT0, INT1;

GIFR – управление всеми внешними прерываниями;

TIMSK, TIFR – управление прерываниями от таймеров/счётчиков;

Регистр GIMSK(GICR)

7	6	5	4	3	2	1	0
INT1	INT0	-	-	-	-	-	-

INTx=1: прерывания по сигналу на входы INTx разрешены.

INTx=0: прерывания по сигналу на входы INTx запрещены.

Регистр GIFR (состояния)

7	6	5	4	3	2	1	0
INTF1	INTF0	-	-	-	-	-	-

INTFx=1: произошло прерывание на входе INTx. При входе в подпрограмму обработки прерывания INTFx автоматически сбрасывается в состояние логический ноль.

Регистр TIMSK

7	6	5	4	3	2	1	0
TOIE1	OCIE1A	OCIE1B	-	TICIE	-	TOIE0	-

TOIE1=1: прерывание по переполнению T/C1 разрешено.

OCIE1A=1: прерывание при совпадении регистра сравнения A с содержимым счётчика T/C1 разрешено.

OCIE1B=1: прерывание при совпадении регистра сравнения B с содержимым счётчика T/C1 разрешено.

TICIE=1: разрешено прерывание при выполнении условия захвата.

TOIE0=1: прерывание по переполнению T/C0 разрешено.

Регистр TIFR (состояния)

7	6	5	4	3	2	1	0
TOV1	OCF1A	OCF1B	-	ICF1	-	TOV0	-

TOV1=1: произошло переполнение T/C1.

OCF1A=1: произошло совпадение регистра сравнения A с содержимым счётчика T/C1 разрешено.

OCF1B=1: произошло совпадение регистра сравнения B с содержимым счётчика T/C1 разрешено.

ICF=1: выполнилось условие захвата.

TOV0=1: произошло переполнение T/C0.

При входе в подпрограмму обработки прерывания соответствующий прерыванию флаг регистра TIFR автоматически сбрасывается в состояние логический ноль.

Прерывания работают только тогда, когда в регистре состояния SREG разрешены общие прерывания (бит 7 = 1). В случае наступления

прерывания этот бит автоматически сбрасывается в 0, блокируя выполнение последующих прерываний.

13.4. Разработка программы

Рассмотрим пример программы по управлению светодиодом с помощью таймера/счетчика. Создаем проект в AVR Studio, как в Лабораторной работе № 12, называем его «withoutdelay» и пишем следующий код мигания светодиодом через прерывания, который выглядит следующим образом:

```
#include <avr/io.h>
#include <avr/interrupt.h>
SIGNAL(SIG_OVERFLOW0)                  //подпрограмма обработки
                                       //прерывания
                                       //по переполнению таймера
{
    TCNT0=0x00;                        //сбрасываем счетный регистр
                                       //таймера/счетчика T0
    if (PIND & (1<<PIND0))             //сравнение нулевого бита порта
                                       //D с единицей
    PORTD &= (0<<PD0);                 //установка 0 на нулевом выводе
                                       //порта D
    else
    PORTD |= (1<<PD1);                 //установка 1
}

int main(void)                         //основная программа
{
    DDRD = 0xff;                       //порт D сконфигурировать
                                       //как выход
    TCCR0=0b00000101;                  //установка делителя
    PORTD = 0x00;                      //установка 0 на всех
                                       //линиях порта D
    TIMSK =0b00000001;                 //записываем 1 в бит
                                       //TOIE0 регистра
                                       //TIMSK – разрешение
                                       //прерывания по переполнению
                                       //таймера/счетчика T0
    TCNT0=0x00;                        //сброс счетного
                                       //регистра таймера T0
    sei();                             //разрешаем прерывания;
    while (1) {}                       //бесконечный цикл
}
```

Подпрограмма прерывания по таймеру изменяет состояния вывода PB0, к которому подключен светодиод, что приводит к миганию. Когда таймер досчитает до максимального значения, он переполняется, т.е. начинает считать с нуля. В нашем случае тактовая частота 1 МГц, делитель 1024, частота таймера счетчика = 976,5625 Гц, длительность тика 0,001024 с, соответственно максимальное значение, которое может измерить 8-битный таймер – 256 · 0,001024 с = 0,262144 секунды.

Обрабатываем эту ситуацию при помощи прерываний. Для этого надо разрешить прерывание по переполнению таймера (записываем 1 в бит TOIE0 регистра TIMSK, TIMSK = 0b00000001) и выставить бит общего разрешения прерываний sei().

Чтобы таймер считал всегда с нуля при инициализации и при каждом срабатывании прерывания сбрасываем счетный регистр таймера TCNT0.

ЛИТЕРАТУРА

1. Раводин О. М. Моделирование робототехнических систем: Лабораторный практикум / О. М. Раводин, В. А. Бейнарович. – Томск: Томский государственный университет, 2009. – Ч. 1. – 104 с.
2. Учебный настольный токарный станок с компьютерным управлением: Учебное пособие / П.Г. Мазеин [и др.]. – Челябинск: Южно-Уральский государственный университет, 2010. – 125 с.
3. Гибкий производственный модуль (ГПМ): Учебное пособие / П. Г. Мазеин [и др.]. – Челябинск: Южно-Уральский государственный университет, 2010. – 29 с.
4. Фещенко В. Н. Токарная обработка: Учебник / В. Н. Фещенко, Р. Х. Махмутов. – 6-е изд., стер. – М.: Высш. шк., 2005. – 303 с.
5. Оглоблин А. Н. Основы токарного дела / А. Н. Оглоблин ; под ред, проф. Г. А. Глазкова. – 3-е изд., перераб. – Л.: Машиностроение, 1974. – 328 с.
6. Автоматизация выбора режущего инструмента для станков с ЧПУ / В. И. Аверченков [и др.]. – 2-е изд., стереотип. – М. : ФЛИНТА, 2011. – 151 с.
7. Фещенко В. Н. Обработка на токарно-револьверных станках: Учеб. пособие для техн. училищ / В. Н. Фещенко. – 2-е изд., перераб. и доп. – М.: Высшая школа, 1989. – 256 с.
8. Роботизированный стенд с техническим зрением и компьютерным управлением для автоматизированной сборки узлов или сортировки изделий: Учебное пособие / П. Г. Мазеин [и др.]. – Челябинск: Южно-Уральский государственный университет, 2010. – 30 с.
9. Раводин О. М. Гибкие автоматизированные производственные системы и робототехника: Учебное пособие / О. М. Раводин, В. А. Бейнарович. – Томск: Томский государственный университет, 2008. – 214 с.
10. Коршунов Ю. М. Математические основы кибернетики: Учеб. пособ. для вузов / М. Ю. Коршунов – 3-е изд., перераб. и доп. – М. : Энергоатомиздат, 1987. – 496 с.
11. Задача коммивояжёра. URL: https://ru.wikipedia.org/wiki/Задача_коммивояжёра (дата обращения: 01.11.2014).
12. Окулов С. М. Программирование в алгоритмах / С. М. Окулов. – М.: БИНОМ. Лаборатория знаний, 2002. – 341 с.
13. Dassault Systèmes SOLIDWORKS. URL: http://www.solidworks.com (дата обращения: 01.10.2014).

14. Учебник SolidWorks для вузов. URL: http://ru.scribd.com/doc/116061389/Учебник-SolidWorks-для-вузов-2004#scribd (дата обращения: 01.10.2014).
15. Тику Ш. Эффективная работа: SolidWorks 2004 / Ш. Тику. – СПб.: Питер, 2005. – 768 с.
16. Жигалова Е. Ф. Автоматизация конструкторского и технологического проектирования: Учебное пособие / Е. Ф. Жигалова. – Томск: Томский межвузовский центр Д.О., 2002. – 80 с.
17. Электроника для начинающих. AVR. Учебный курс. URL: http://www.avr-start.ru (дата обращения 20.05.14).
18. AVR. Учебный курс. Интернет издание. URL: http://libbib.org/avr-uchebnyj-kurs-di-halt (дата обращения 20.05.2014).