Министерство образования и науки Российской Федерации
Государственное образовательное учреждение
высшего профессионального образования
Карельский Государственный Педагогический Университет

Студент 555 группы
А. В. Гарулин

Программно управляемый исполнитель

Дипломная работа

Кафедра информатики
Научный руководитель:
старший преподаватель
А. С. Кюршунов

Петрозаводск
2007

Оглавление

Введение

Необходимость изучения информатики в школе неоспорима и давно доказана, цель изучения этого предмета не должна быть ограничена только подготовкой и изучением базовых знаний для дальнейшего получения профессии. В любой среде программирования реализуются основные алгоритмические конструкции, развивающие алгоритмический стиль мышления, важность которого отмечена Н.М. Амосовым, Н.Н. Моисеевым, А.Н. Лонда и другими учеными. Ими подчеркивалась необходимость разработки алгоритмов для развития школьников. Они показывали, что с помощью алгоритмов можно не только организовывать мыслительную деятельность, но и описывать процессы.

Алгоритмы возникают не только в ходе описания какого-либо процесса (физического, химического, биологического, математического), но и в управлении, воспитании, во всей социальной сфере жизни человека. Именно это и доказывает необходимость их введения в обучение. Некоторые алгоритмы человек осваивает самостоятельно, другие требуют обучения.

Изучать основные базовые конструкции можно по-разному: с помощью реальных языков программирования (C++, Basic, Pascal, Fortran, ADA и др.), специально разработанных учебных алгоритмических языков (Алгоритмический язык Ежова), с помощью исполнителей (Лого Миры, LogoWriter, Кукарача, Кенгуренок, Муравей и тд.) или с помощью программно управляемых конструкторов и исполнителей (ЛЕГО-Лаборатория,).

При этом программно-управляемые исполнители обладают положительными качествами программных исполнителей:

1. Простотой язык описания алгоритмов.
2. Простой и интуитивно понятный интерфейс

Добавляя к ним новые свойственные только им качества:

1. Наглядная демонстрация – укрепление связи между абстрактно изложенным алгоритмом и действиями исполнителя. Таким образом, устраняется одна из проблем – недостаточного уровня развития абстрактного мышления.
2. Присутствие исполнителя «рядом» с программистом.

Цель данной работы теоретическая разработка программно-управляемого исполнителя (ПУИ) и создание рабочей модели. При этом ПУИ должен отвечать следующим требованиям:

1. Простота конструкции
2. Доступность
3. Простота управления и настройки

Задачи:

1. Разработка концепции ПУИ
2. Выбор средств реализации
3. Создание действующей модели

Дипломная работа состоит из введения, трех глав, заключения и одного приложения.

Первая глава данной работы посвящена разработке концепции программно-управляемого исполнителя, внутреннему устройству систем и концепции взаимодействия с компьютером.

В второй главе описаны технологические решения используемые при создании модели исполнителя и обосновывается их применение.

Третья глава, практическая часть, посвящена созданию рабочей модели исполнителя.

1. Концепция программно-управляемого исполнителя

1.1 Концепция ПУИ

Взаимодействие систем компьютера и исполнителя изображено на Рисунке 1. Исполнитель не самостоятельная система и ориентирована на выполнение команд поступающих с LPT порта компьютера в виде шести битового слова, где:

1..4 бит – информация о присутствии или отсутствии сигнала (+5В) на соответствующей ножке регистра. Например: 0 0 0 1 0 1 – говорит о необходимости установки положительного сигнала на 1 и 3 ножке регистра исполнителя;

5,6 биты – указывают на регистр, в который будет происходить запись. Например: шести битовое слово 0 1 0 0 0 0 – говорит о необходимости записи в первый регистр исполнителя.

Запись данных в регистр исполнителя происходит в два шага: первым шагом в порт отправляется слово с указанием, в какой регистр исполнителя необходимо произвести запись; вторым шагом отправляются непосредственно данные. Например, для установки положительного сигнала (+5В) на третьей ножке первого регистра исполнителя необходимо отправить 0 1 0 0 0 0 –для указания регистра исполнителя в который будет происходить запись, следующим шагом отправить 0 0 0 1 0 0 – информация предназначенная для записи.

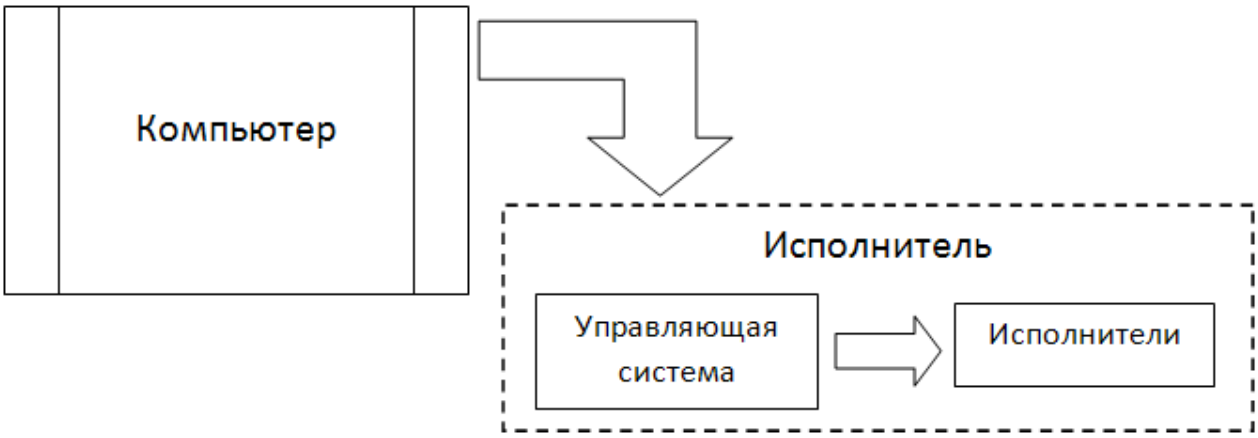

Рисунок 1. Схема взаимодействия системы “компьютер - исполнитель” и внутренней системы исполнителя

На Рисунке 2 изображена схема подключения регистров исполнителя к шине параллельного порта. Цифрами на схеме обозначены выводы соответствующие номерам битов управляющего слова, состоящего из 6 бит. После записи в порт компьютера модулем коммуникатором управляющего слово

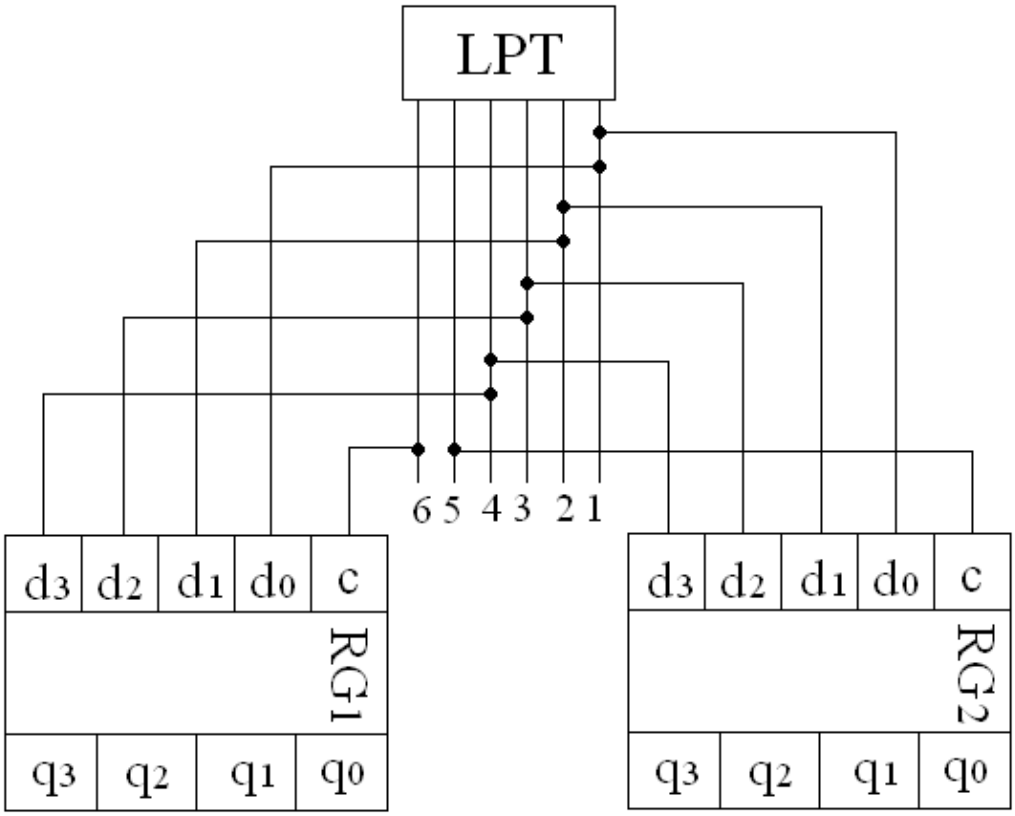

Рисунок 2 функциональная схема подключения регистров к шине LPT

2. Выбор технологических решений

Средства реализации можно разделить на три основные части:

1. Среда программирования. Система, в которой будет создан интерфейс, модуль коммуникации компьютера и устройства, интерпретатор - система обработки пользовательских программ.
2. Цифровые, комбинационные и коммутирующие устройства. Устройства, с помощью которых будет создана действующая модель.
3. Интерфейс подключения. Интерфейс, порт, с помощью которого будет осуществляться сопряжение компьютера и устройства.

2.1 Система программирования

В связи с широким использованием графических операционных систем, в том числе и в школах, для создания пользовательского интерфейса был выбран объектно-ориентированный язык программирования. Из множества существующих систем объектно-ориентированного программирования можно выделить Delphi. В ходе работы данный язык бал выбран по нескольким причинам:

- легкость и удобство создания интерфейса;
- хорошо известен и изучаем во многих ВУЗах;
- большое количество разнообразных подключаемых модулей;
- простая схема создания алгоритмов;
- быстрая работа от компилированных программ.

2.2 Цифровые, комбинационные и коммутирующие устройства

В ходе работы исполнителю потребуется выполнять три основные задачи:

1. запоминать текущую команду компьютера;
2. отображать полученную команду в виде двоичного кода;
3. отправлять сигнал по средствам коммутирующих устройств, двигателям и реле.

Все три задачи будут выполнять регистры К155ИР1. Их устройство позволяет хранить до 4 бит информации, логика устройства легко совместима с логикой компьютера, скорость срабатывания и записи сравнима со скоростью передачи данных портом компьютера, в функции регистров входят все три необходимые задачи.

Отображение текущей команды будет выполняться по средствам светодиодов получающих сигнал от регистра.

Для управления питанием двигателей используются коммутирующие устройства – реле, с напряжением срабатывания 5 вольт и коммутируемым напряжением до 24 вольт. Использование реле позволяет: снизить нагрузку на порт, применить внешний источник питания, использовать двигатель с более высоким потреблением энергии, следовательно, и мощностью. В качестве альтернативы реле можно использовать транзисторы:

1. Биполярные транзисторы: ZTX300, BC108C;
2. Транзисторы Дарлингтона: TIP122, TIP142;
3. Полевые транзисторы: VN10KM, VN66AF.

2.3 Интерфейс подключения

За время развития компьютерной техники широкое распространение получили три достаточно универсальных интерфейсов ввода вывода:

1. Последовательный порт RS-232 или COM-порт. COM-порт основан на использовании микросхемы UART 8250/16450, имеет пропускную способность до 115,2 Кбит/с., для передачи данных

использует четыре провода (нуль-модемное соединение), передача данных осуществляется на расстояние до 50 метров;

2. Последовательный порт USB (Universal Serial Bus). Универсальная последовательная шина USB для передачи данных использует три провода, передача данных осуществляется на расстояние до 10 метров со скоростью до 480 Мбит/с.;

3. Параллельный порт LPT (Centronic). Параллельная шина LPT использует восемь проводов для передачи данных, без дополнительных усилителей сигнала, данные передаются на расстояние до 30 метров со скоростью до 16 Мбит/с.;

Таблица 1 Сравнительные характеристики интерфейсов подключения

Интерфейс	Тип передачи данных	Организация интерфейса	Количество проводов/длинна провода	Скорость передачи данных
COM	Последовательный	Микропроцессор	6/50-100	115,2 Кбит/с
USB	Последовательный	Микропроцессор + программа	3/10	480 Мбит/с
LPT	Параллельный	Непосредственное подключение	9/10-30	16 Мбит/с

Для сопряжения исполнителя с компьютером удобнее всего использовать параллельный порт LPT, скорость передачи данных, которой обладает порт, вполне достаточно для оперативной отправки команды исполнителю и ограничена характеристиками серии «К» используемых регистров. Для организации интерфейса шины LPT не требуется использования дополнительных микросхем или контроллеров. Расстояние, на которое передает данные интерфейс, вполне достаточное для свободного перемещения исполнителя, а параллельный поток данных наиболее легок в обработке и организации функциональной схемы, так как не требует расшифровки.

Глава 3. Описание разработки программно-управляемого исполнителя

3.1 Устройство рабочей модели

На рисунке 3 изображена внутренняя, функциональная схема управления исполнителями.

Контакты RG1, RG2 (микросхемы К155ИР1):

1. Vcc, gnd – питание микросхем;
2. Входы V – выбор способа записи, установлено постоянное подключение к положительной клейме (+5В) – параллельный ввод данных с входов d1..d4;
3. Входы d1..d4 записываемые данные – подключены к шине LPT;
4. Выходы q1..q4 хранимые данные.

Резисторы R1..R8 - 100 Ом. Светодиоды D1..D8 - 3В.

При установке положительного сигнала (+5В) на линиях 5 или 6 (контакты С2 регистров – тактовый вход при параллельной записи) шины LPT, регистр RG1 или RG2 соответственно, переходят в режим записи с входов 1 .. 4 (контакты d1..d4 регистров). После установки нулевого сигнала на линиях 5 или 6 (контакты С2 регистров) регистры переходят в режим отображения полученной информации и выводят ее на выходах q1..q2.

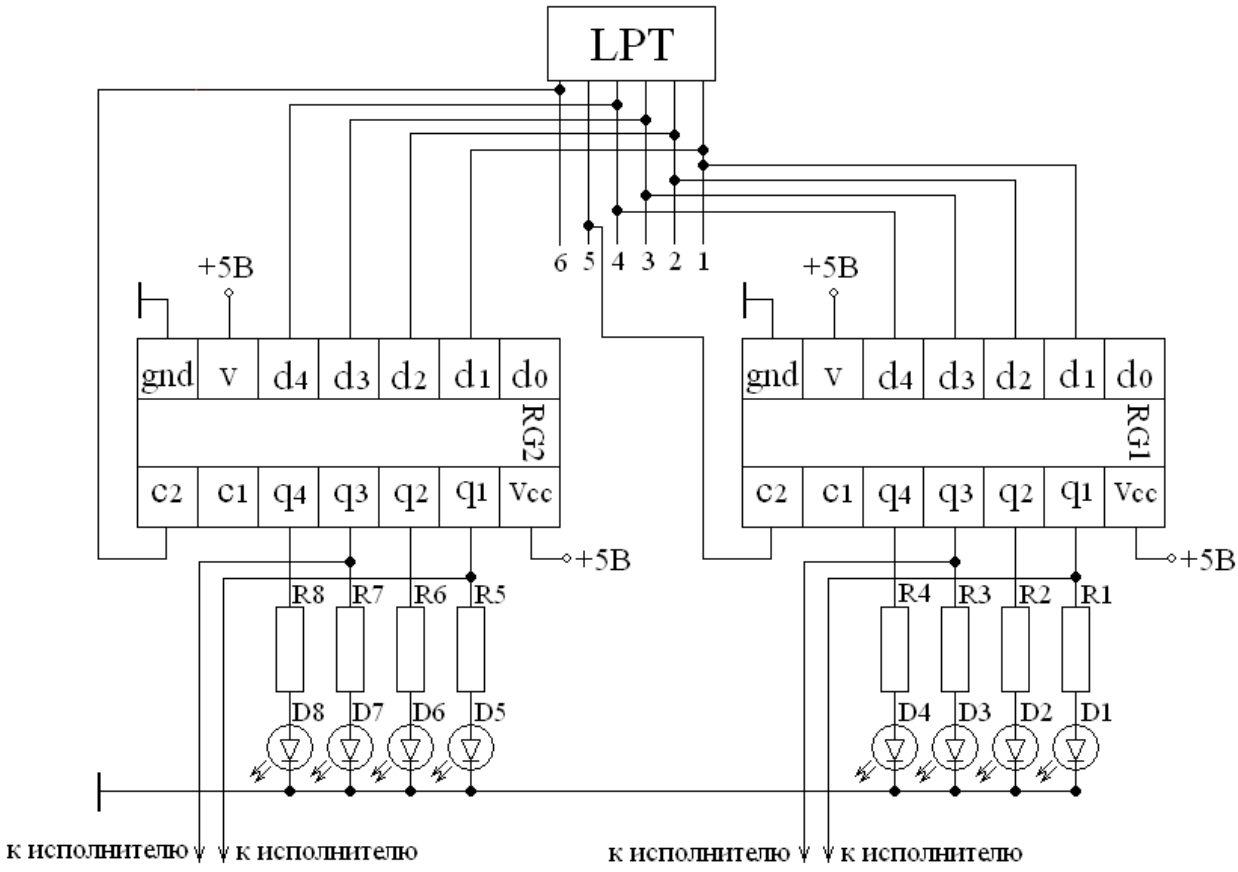

Рисунок 3 функциональная схема управления исполнителями

Напряжение на выходах регистров не хватает для полноценного питания двигателей и электромагнитов, в связи с этим в архитектуру введена система управления питанием исполнителя. Данная система может быть реализована на двух типах коммутирующих устройств:

1. Реле рисунок 4. Вход данной функциональной схемы подключается к выходу системы управления исполнителями, под нагрузкой подразумевается, исполнитель: двигатель или электромагнит, напряжение питания (Vcc) устанавливается в соответствии с типом используемого исполнителя и не должно превышать максимальное коммутируемого напряжения реле;

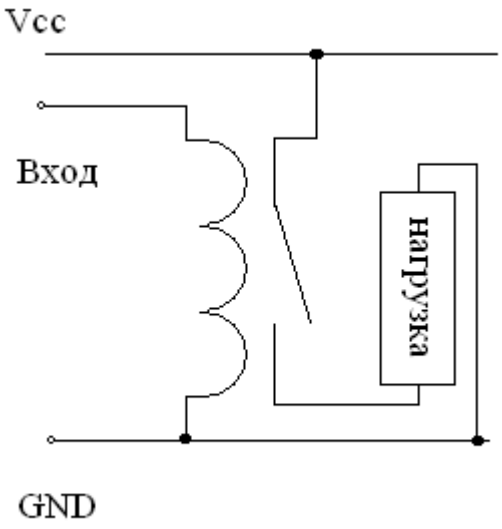

Рисунок 4 система управления питанием на реле

2. Транзисторы, рисунок 5,6,7. Вход функциональных схем подключается к выходу системы управления исполнителями:

a) Биполярный транзистор рисунок 5. Данная функциональная схема позволяет коммутировать ток до 25В (Vcc), силой до 500 mA для ZTX300 и 100 mA для BC108C, резистор R1 4,7 КОм, D1 – 1N4148, D2 – 1N4001;

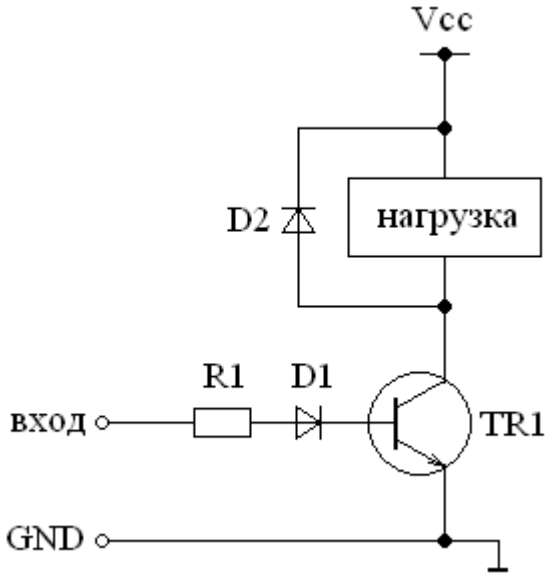

Рисунок 5 схема управления питанием на полевом транзисторе

b) Транзистор Дарлингтона рисунок 6. Данная функциональная схема позволяет коммутировать ток до 100В (Vcc), силой до 5 А для TIP122 и 10 А для TIP142, резистор R1 330 Ом, D1 – 1N4001;

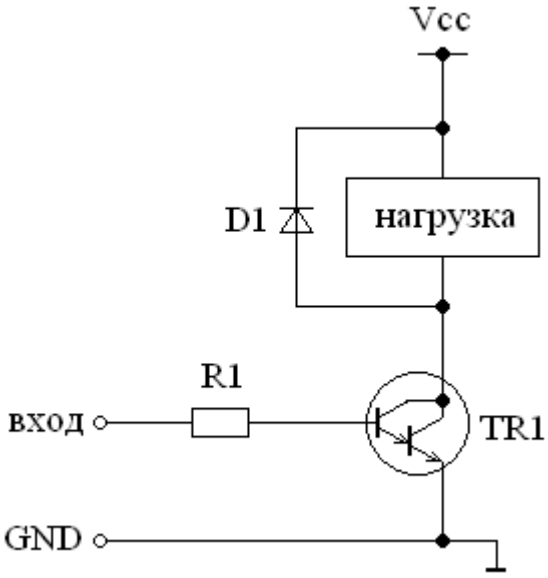

Рисунок 6 схема управления питанием на транзисторе Дарлингтона

а) Полевой транзистор рисунок 7. Данная функциональная схема позволяет коммутировать ток до 60В (Vcc), силой до 0,3 А для VN10KM и 2 А для VN66AF, резистор R1 220 КОм, D1 – 1N4001.

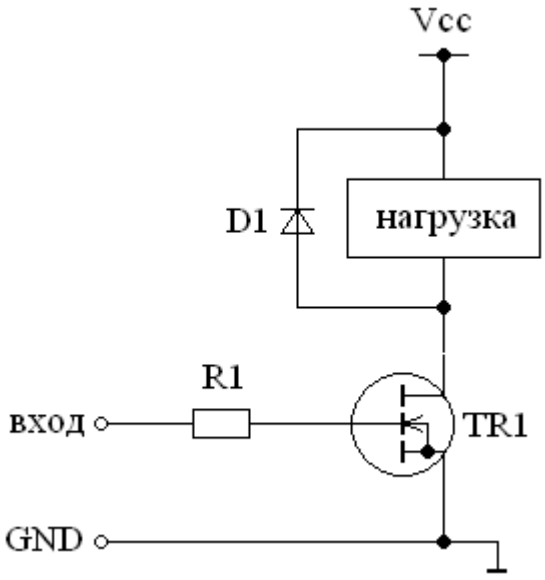

Рисунок 7 схема управления питанием на полевом транзисторе

3.2 Взаимодействие устройства с ПК

Модуль коммуникации, осуществляющий взаимодействие устройства с ПК, вступает в работу после нажатия кнопки «Выполнить». Последовательно перебирая команды в списке, модуль сопоставляет команду ее коду и отправляет в последовательный порт компьютера.

В основе обращения к порту лежит использование драйвера LPTWDMIO.sys по средствам функций библиотеки LPTIO.PAS

3.3 Взаимодействие устройства с пользователем

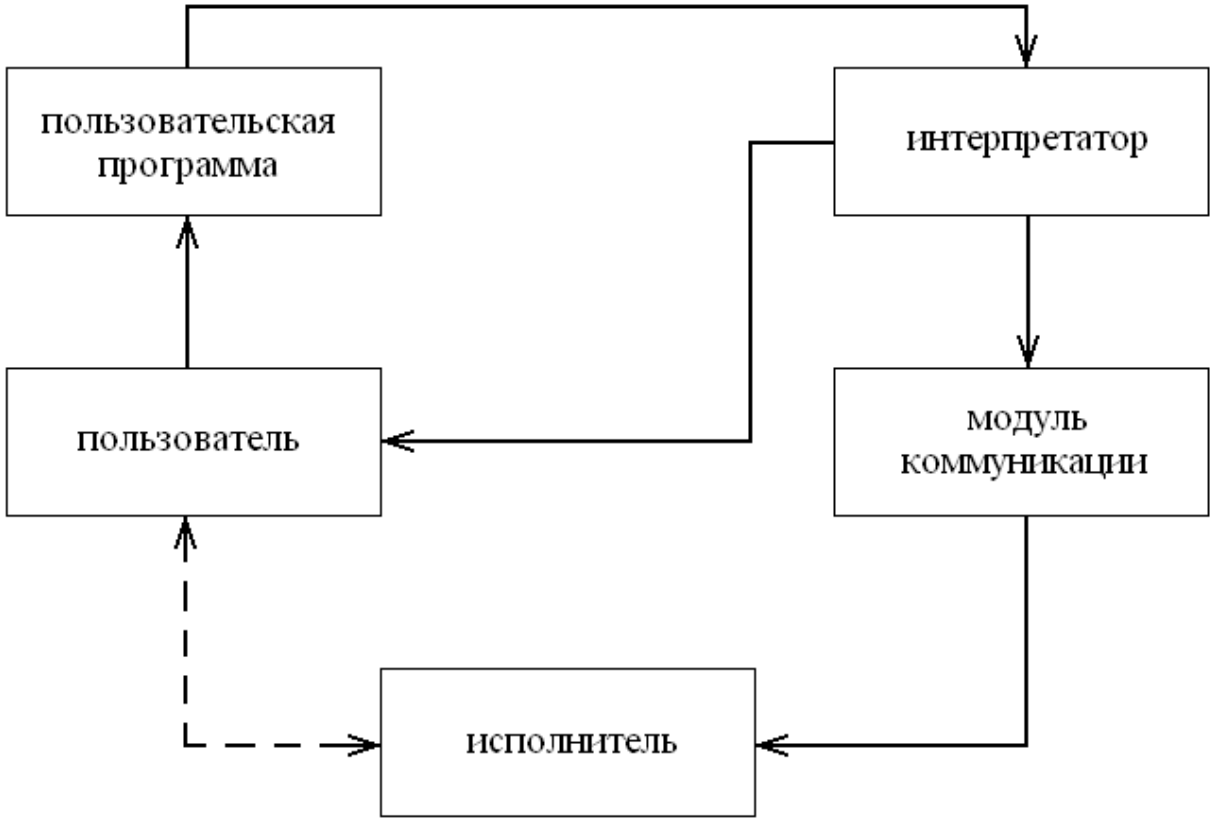

Рисунок 8. Схема работы системы пользователь исполнитель

На Рисунке 8 изображена схема взаимодействия пользователя и исполнителя. Пользователь по средствам интерфейса вводит текст программы, по окончанию ввода программа проверяется и интерпретируется модулем интерпретатором, в случае возникновения ошибок, модуль сообщает о возникшей ситуации пользователю, работа системы прекращается, в случае отсутствия ошибок данные передаются в модуль коммуникации, где передается непосредственно исполнителю.

Внешний интерфейс

Главное окно программы представлено на рисунке 9. Окно программы можно разделить на 6 областей:

1. Главное меню программы, содержит в себе:
    a. Функции работы с файлами, меню Файл,

b. Функции добавления и изменения программы, меню Правка,
c. Функции компиляции, меню Компиляция,
d. Справка по программе, меню справка;

2. Поел отображения программы;
3. Поле отображения линейного образа программы;
4. Поле ввода команды;
5. Кнопки управления вводом, изменением, компиляции и выполнением программы;
6. Поле вывода сообщения об ошибках.

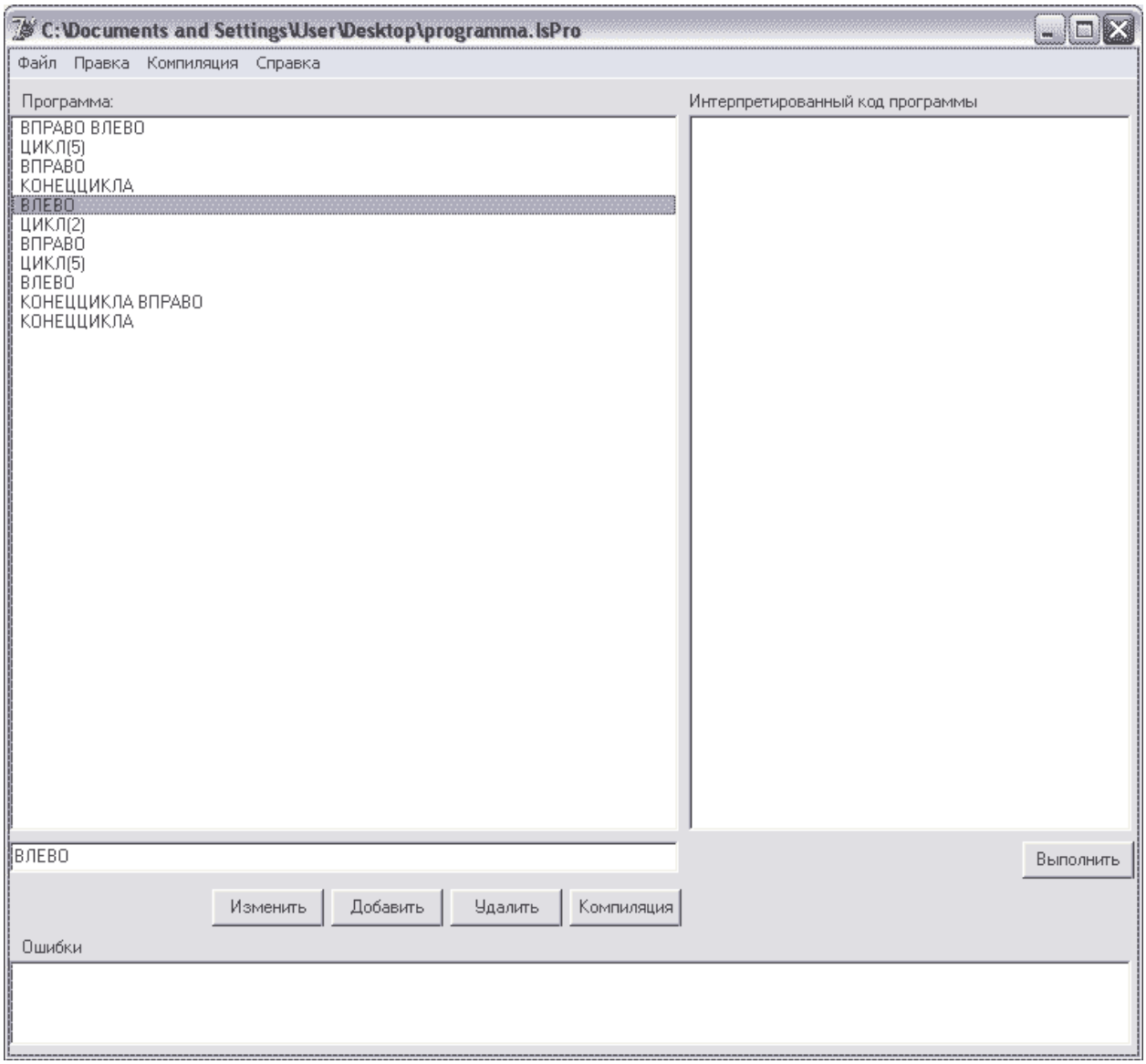

Рисунок 9 Главное окно программы

Программа вводиться через поле ввода команд, после нажатия кнопки Добавить команда уставляется последней строкой в поле отображения программы.

Для внесения изменений, в какую либо из строк программы, строка выделяется мышкой в поле отображения программы. Выбранная строка появляется в поле ввода команд, после ввода изменений нажатием кнопки Изменить. Исходная строка программы заменяется на измененную.

Если строку необходимо удалить, строка выделяется мышкой в поле отображения команд, после нажатия кнопки удалить данная строка будет удалена из списка команд.

С нажатием кнопки Компиляция, программа введенная пользователем проверяется, приводится к линейному виду и выводится в поле отображения линейного образа программы.

После выполнения компиляции становиться доступной кнопка Выполнить, нажатие которой приводит к выполнению линейного алгоритма исполнителем.

Язык описания команд

Действия, описываемые в алгоритме, прежде всего, должны быть понятны самому разработчику алгоритма. Только тогда алгоритм можно преобразовать в форму, понятную тому, кто будет его выполнять.

(Н.В.Макарова)

Конструкция языка должна быть простой, что бы не заострять внимание на архитектуре языка и не отвлекать от решения поставленной задачи. В то же время язык должен сохранять возможность создания более сложных структур и алгоритмов, каскадирования команд, вложенности и рекурсивности отдельных структур используемых на последующих этапах изучения алгоритмизации.

Для начального знакомства с алгоритмизацией и первыми шагами в языках, лучше всего использовать язык по синтаксису близкому к естественному языку, что с успехом реализовано в среде Лого миры.

Система команд, основанная на узкоспециализированной цели, решения определенного круга задач (моделирование движения, создания рисунка и т.д.) может быть реализована в терминах этой проблемы, что

существенно ускорит освоение среды и укрепит логическую связь между действием производимым исполнителем и командой.

Программно управляемый исполнитель ориентирован на выполнение односложных команд. Все конструкции используемого языка формируются из установленных команд, по определенным в языке правилам. Алфавит языка описания команд ограничен использованием цифр: 0 .. 9; и разделителями: « » - пробел, «;» - точка с запятой.

Программа на языке описания команд состоит из множества операторов. Все операторы языка обозначаются ключевым словом, описывающим действие необходимое выполнить исполнителю, и разделены разделителями.

В соответствии с назначением исполнителя операторы можно разделить на три группы:

1. Команды перемещения подвижной части исполнителя;
2. Команды включения и отключения электромагнитов;
3. Операторы организации цикла.

Операторы перемещения подвижной части исполнителя:

1. ВПРАВО – команда перемещения подвижной части исполнителя на один шаг вправо;
2. ВЛЕВО – команда перемещения подвижной части исполнителя на один шаг влево.

Операторы Включения и отключения электромагнитов:

1. ПОДНЯТЬ – команда включения электромагнита;
2. ОТПУСТИТЬ – команда выключения электромагнита.

В языке описания команд предусмотрено использование алгоритмической структуры цикл с параметром, признаком начала тела цикла служит оператор ЦИКЛ(N), где N количество повторений тела цикла. Признаком окончания тела цикла является оператор КОНЕЦЦИКЛА.

Язык описания команд поддерживает как одиночные циклы, так и многократное их вложение друг в друга. Пример программы с использование вложенных циклов приведен в Листинге 3. Приведенная в листинге программа демонстрирует перемещение небольших объектов вправо через одну позицию.

Листинг 5 Пример программы написанный на языке описания команд для программно-управляемого исполнителя.

```
ОТПУСТИТЬ
ЦИКЛ(5)
    ВПРАВО
    ПОДНЯТЬ
    ЦИКЛ(6)
        ВПРАВО
    КОНЕЦЦИКЛА
    ОТПУСТИТЬ
    ЦИКЛ(6)
        ВЛЕВО
    КОНЕЦЦИКЛА
КОНЕЦЦИКЛА
```

Интерпретатор

Программно управляемый исполнитель ориентирован на исполнение команд изложенных в алгоритме, модуль интерпретатор призван перевести текст пользователя, «пользовательскую программу» к линейно-исполняемому алгоритму.

Интерпретация программы проходит в четыре этапа Рисунок 10:

1. Преобразование символов к верхнему регистру;
2. Разделение текста программы на составляющие команды;
3. Проверка наличие не известных команд;
4. Преобразование программы к линейному алгоритму.

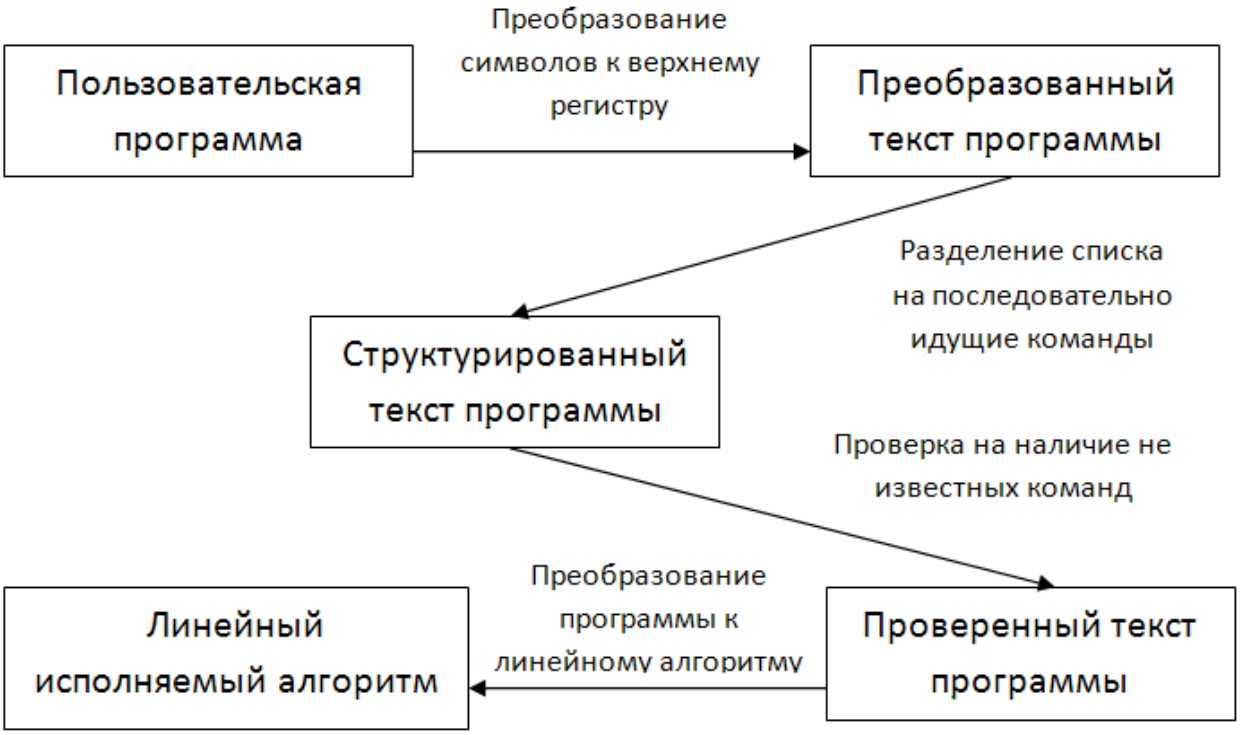

Рисунок 10. Схема преобразования программы

Выполнение первого этапа происходит на стадии ввода строк программы пользователем по средствам свойства CharCase, объекта Edit и служит для более быстрой идентификации команд.

Второй этап выполняется непосредственно при компиляции программы, разделением строки на отдельные команды при нахождении в ней между слов знака « » или «;» см. Листинг 1.

Листинг 1 Процедура разделения строк на отдельные команды.

```
procedure CeparateText;
var
  i,j:integer;
  buf:string;
begin
  for i:=1 to KolVoStr do
    for j:=1 to length(NeodeptText[i])+1 do
      if ((NeodeptText[i][j]<>' ')and(NeodeptText[i][j]<>';'))and
                                    not(j>length(NeodeptText[i])) then
       buf:=buf+NeodeptText[i][j]
      else
       begin
        if buf<>'' Then
          begin
            KolVoKomand:=KolVoKomand+1;
```

```
            SpisokKommand[KolVoKomand]:=buf;
          end;
        buf:='';
      end;
end;
```

Ниже приведен пример программы и результат работы функции CeparateText.

Программа	Результат работы процедуры CeparateText
ОТПУСТИТЬ ЦИКЛ(5) ВПРАВО; ПОДНЯТЬ ЦИКЛ(6) ВПРАВО КОНЕЦЦИКЛА ОТПУСТИТЬ ЦИКЛ(6);ВЛЕВО;КОНЕЦЦИКЛА КОНЕЦЦИКЛА	ОТПУСТИТЬ ЦИКЛ(5) ВПРАВО ПОДНЯТЬ ЦИКЛ(6) ВПРАВО КОНЕЦЦИКЛА ОТПУСТИТЬ ЦИКЛ(6) ВЛЕВО КОНЕЦЦИКЛА КОНЕЦЦИКЛА

Третий этап, этап проверки наличия неизвестных команд в тексте программы, осуществляется сравниванием команд введенных пользователем, со списком заданных команд см. Листинг 2.

Листинг 2. Функция проверки наличия в тексте программы неизвестных команд

```
function ErrorsSpi:boolean;
var
  i:integer;
  errors:boolean;
begin
  errors:=false;
  form1.ListBox3.Clear;
  for i:=1 to KolVoKomand do
    begin
      if not((SpisokKommand[i]='ВПРАВО')or(SpisokKommand[i]='ВЛЕВО')

or(SpisokKommand[i]='ПОДНЯТЬ')or(SpisokKommand[i]='ОТПУСТИТЬ')
```

```
or(SpisokKommand[i]='КОНЕЦЦИКЛА')or(QCikl(SpisokKommand[i])>0))Th
en
       begin
        form1.ListBox3.Items.Add('Неизвестная команда '+SpisokKommand[i]);
        errors:=true;
       end;
   end;
  ErrorsSpi:=errors;
end;
```

Четвертым этапом осуществляется преобразование текста программы к линейному алгоритму. Рекурсивная функция SpisokToIsp (см. Листинг 3) рассматривает текст программы как цикл, повторить который необходимо один раз. Перебирая по очереди все команды, идущие по списку, она переносит их в новый формируемый линейный список, встретив команду «ЦИКЛ» функция вызывается, повторно устанавливая новые параметры: номер строки в которой находится команда цикла и количество повторений. Функция выполняется до тех пор, пока не будут выполнены все итерации, признаком конца итерации служит команда «КОНЕЦЦИКЛА». Возвращаемое функцией значение указывает, на какую строку необходимо вернуться после выполнения внутреннего цикла.

Листинг 3 Функция преобразования циклического списка команд к линейному виду.

```
function SpisokToIsp(Nkomand,KolVoPov:integer):integer;
var
   komanda:integer;
begin
  komanda:=Nkomand;
  while KolVoPov>0 do
    begin
       komanda:=komanda+1;
       if (spisokkommand[komanda]<>'КОНЕЦЦИКЛА')and
          (QCikl(SpisokKommand[komanda])=0) then
         begin
           KolVoKomInIspSpis:=KolVoKomInIspSpis+1;
           IspolnimiySpisok[KolVoKomInIspSpis]:=spisokkommand[komanda];
         end
       else
```

```
        if spisokkommand[komanda]='КОНЕЦЦИКЛА' Then
          begin
            KolVoPov:=KolVoPov-1;
            if KolVoPov>0 then
              komanda:=Nkomand;
          end
        else
          if QCikl(SpisokKommand[komanda])>0 then
            komanda:=SpisokToIsp(komanda,QCikl(SpisokKommand[komanda]));
    end;
  SpisokToIsp:=komanda;
end;
```

Ниже приведен пример программы и результат работы функции

SpisokToIsp.

Программа	Результат работы функции SpisokToIsp

ВПРАВО
ЦИКЛ(2)
ВЛЕВО
  ЦИКЛ(3)
  ВПРАВО
  КОНЕЦЦИКЛА
КОНЕЦЦИКЛА
	ВЛЕВО
	ВПРАВО
	ВПРАВО
	ВПРАВО
	ВЛЕВО
	ВПРАВО
	ВПРАВО
	ВПРАВО

Заключение

В ходе работы была разработана концепция программно-управляемого исполнителя, его структура, схема работы с компьютером и взаимодействия с пользователем. В соответствии с разработанной концепцией были отобраны необходимые средства реализации:

1. Система программирования Delphi;
2. Цифровые устройства;
3. Интерфейс подключения.

В соответствии с концепцией программно управляемого исполнителя были разработаны функциональные схемы:

1. Системы управления исполнителями;
2. Системы управления питанием исполнителя.

Обоснованно их использование в данной системе.

В ходе практической части работы была создана и опробована рабочая модель программно-управляемого исполнителя, разработан язык описания команд и интерфейс среды описания команд.

Список литературы

1. Пей Ан Сопряжение ПК с внешними устройствами; Пер. с англ. Мерещука П.В. – 2-е изд., стер. – М. ЖДМК Пресс; Питер, 2004. - 320с.:ил.
2. Ямпольский В. С. Основы автоматики и электронно-вычислительной техники: Учеб. пособие для студентов физ.-мат. фак. пед. ин-тов. -М.: Просвещение, 1991.-223с.:ил.
3. Безуглов Д. А., И. В. Калиенко. Цифровые устройства и микропроцессоры/ - Ростов н/Д.: Феникс, 2006. – 480 с. –(Высшее образование).
4. Климова Л. М. PASCAL 7.0. Практическое программирование. Решение типовых задач. – М.: КУДИЦ-ОБРАЗ, 2000. – 528 С.
5. М.П.Лапчик, И.Г.Семакин, Е.К.Хеннер; Под общей ред. М.П.Лапчика. Методика преподавания информатики: Учеб. Пособие для студ. пед. вузов. – М.:Издательский центр «Академия», 2002. – 624с.
6. Информатика. Методическое пособие для учителей. 7 класс / Под ред. проф. Н.В.Макаровой. – СПб.: Питер, 2003. – 384 с.: ил.
7. Н.Ф.Софронова. Теория и методика обучения информатике: Учеб. пособие. – М.: Высш. шк., 2004. – 223 с.: ил.
8. Информатика: начальный курс / Под ред. Н.В.Макаровой. – СПб: Питер, 2001. – 160 с.: ил.

9.

Приложение. Исходный код программы