Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет ИДиП

Кафедра ИСиТ

Специальность ИСиТ

Специализация 1.40-01.02.03 Информационные системы и технологии

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

КУРСОВОЙ РАБОТЫ

по дисциплине Специализированные информационные системы в медиа, издательском деле и полиграфии

Тема: Разработка программы (DELPHI) преобразования блок-схем алгоритмов (VBA Visio) на язык VBSA

Исполнитель

студент V курса группы 9 Щербацкий А.В. (Ф.И.О)

Руководитель

доцент, к.т.н. \_\_\_\_\_\_\_\_\_\_ С.И. Акунович

Курсовой проект защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель\_\_\_\_\_\_\_\_\_\_ С.И. Акунович

Минск 2012

Оглавление

[Введение 3](#_Toc344018630)

[1. Основные разделы 8](#_Toc344018631)

[1.1 Исходные данные 8](#_Toc344018632)

[1.2 Модель предметной области 8](#_Toc344018633)

[1.3 Добавление компонента в проект 9](#_Toc344018634)

[1.4 Подключение компонентов 12](#_Toc344018635)

[1.5 Реализация программы 12](#_Toc344018636)

[1.6 Язык VBSA 12](#_Toc344018637)

[1.7 Пример работы программы 15](#_Toc344018638)

[2. Графическая часть 17](#_Toc344018639)

[2.1. Структурная схема БУ БКУ 17](#_Toc344018640)

[2.2. Функциональная схема БУ БКУ 18](#_Toc344018641)

[2.3. Модель предметной области 19](#_Toc344018642)

[2.4. Схема программы 20](#_Toc344018643)

[Заключение 21](#_Toc344018644)

[Список использованных источников 22](#_Toc344018645)

# Введение

Современные технологические машины и процессы содержат многочисленные датчики, сигнализирующие об их состояниях. Сигналы с датчиков поступают на вход системы управления, которая должна анализировать текущую ситуацию и вырабатывать адекватные воздействия на исполнительные компоненты оборудования, непосредственно управляющие исполнительными механизмами: переключателями, клапанами, двигателями, насосами, электромоторами. Как сигналы с датчиков, так и управляющие воздействия дискретны, а чаще всего двоичные, т. е. управление становится логическим, и в этом случае системы управления принято называть системами логического управления.

Системы логического управления находят широкое применение не только на производстве, но и на транспорте, в атомной энергетике, сетях связи, информационно-справочных системах, ракетно-космической отрасли, компьютерах и бытовой аппаратуре и т. д.

Разработка любой системы управления всегда начинается с алгоритмического описания процесса управления. Лишь после четкого описания алгоритма, его проверки и отладки средствами моделирования можно перейти к конструированию технической системы, реализующей этот алгоритм. Если в описании алгоритма допущены ошибки, дальнейшая его реализация может оказаться бессмысленной.

Системы управления полиграфическим оборудованием. Многие учёные, в том числе и российские, относят полиграфическое оборудование по его сложности и наукоёмкости к тому же классу, что и космическое оборудование. Главная особенность последних поколений этого оборудования заключается в чрезвычайно высоком уровне его автоматизации, при этом используется только компьютерное управление. Кроме основного управляющего компьютера, который имеется на каждой единице оборудования и может обмениваться данными как с оператором, так и с компьютерами других единиц оборудования, в каждой машине имеются десятки встроенных управляющих микроконтроллеров. Так что системы управления полиграфическим оборудованием только по этой причине являются дискретными.

Сравнительно новым направлением в полиграфии является объёмная печать или, как её ещё называют 3D-печать.

Системы управления спутниками ГЛОНАСС. Полная орбитальная структура системы ГЛОНАСС состоит из 24 спутников, равномерно размещенных трех орбитальных плоскостях (рис. 1).

|  |
| --- |
| uragan1 |
| Рис.1. Полная орбитальная структура системы ГЛОНАСС |

[Спутник ГЛОНАСС](http://www.glonass-center.ru/satellite_w.html) (рис. 2) конструктивно состоит из цилиндрического гермоконтейнера с приборным блоком, рамы антенно-фидерных устройств, приборов системы ориентации, панелей солнечных батарей с приводами, блока двигательной установки и жалюзи системы терморегулирования с приводами.

|  |
| --- |
|  |
| Рис. 2. [Спутник ГЛОНАСС](http://www.glonass-center.ru/satellite_w.html) |

В состав бортовой аппаратуры входят:

* навигационный комплекс;
* комплекс управления;
* система ориентации и стабилизации;
* система коррекции;
* система терморегулирования;
* система электроснабжения.

Управление спутниками ГЛОНАСС осуществляется в автоматизированном режиме.

Создание автоматизированных систем управления. Стандарт [ГОСТ](http://ru.wikipedia.org/wiki/%D0%93%D0%9E%D0%A1%D0%A2) 34.601-90 предусматривает следующие стадии и этапы создания автоматизированной системы (АС):

1. Формирование требований к АС
2. Разработка концепции АС
3. Техническое задание
4. Эскизный проект
5. Технический проект
6. Рабочая документация
7. Ввод в действие
8. Сопровождение АС.

Данный стандарт не вполне подходит для проведения разработок в настоящее время: многие процессы отражены недостаточно, а некоторые положения устарели.

Стандарт ISO/IEC 12207:1995 «Information Technology — Software Life Cycle Processes» является основным нормативным документом, регламентирующим состав процессов жизненного цикла ПО. Он определяет структуру жизненного цикла, содержащую [процессы](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81), [действия](http://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%8F%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C) и [задачи](http://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0), которые должны быть выполнены во время создания ПО.

Каждый процесс разделен на набор действий, каждое действие — на набор задач. Каждый процесс, действие или задача инициируется и выполняется другим процессом по мере необходимости, причем не существует заранее определенных последовательностей выполнения. Связи по входным данным при этом сохраняются.

Модель жизненного цикла ПО — структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении жизненного цикла. Модель жизненного цикла зависит от специфики, масштаба и сложности проекта и специфики условий, в которых система создается и функционирует.

Rational Unified Process (RUP) – одна из лучших методологий разработки программного обеспечения, созданная в компании Rational Software. Основываясь на опыте многих успешных программных проектов, Унифицированный процесс позволяет создавать сложные программные системы, основываясь на индустриальных методах разработки. Одним из основных столпов, на которые опирается RUP, является процесс создания моделей при помощи унифицированного языка моделирования (UML).

Унифицированный процесс можно представить как сумму различных видов деятельности компании-разработчика, необходимых для перевода требований заказчика в программную систему.

Говоря о разработке программного обеспечения, многие пользователи и заказчики подразумевают в первую очередь написание кода приложений и их внедрение. Причиной этого, как правило, служит опыт, приобретенный многими компаниями в 90-е годы. Именно в тот период многие задачи автоматизации решались стихийно и, как правило, силами разработчиков-универсалов, которые, будучи мастерами на все руки, сами проектировали приложения, писали код, готовили проектную документацию, тестировали и внедряли продукт, нередко заодно выполняя и функции системных администраторов.

Для успешной реализации современных проектов требуется весьма широкий спектр знаний и умений, которыми, как правило, обладают разные специалисты, выполняющие в проекте различные функции и задействованные на разных этапах создания продукта.

Моделирования сложных систем визуальными (наглядными) моделями. В последнее десятилетие в компьютерном мире наметилась тенденция моделирования сложных систем визуальными (наглядными) моделями. Причем в новых методах проектирования сложных компьютерных систем, например ООП и ООАП, наглядные модели очень часто связываются с такими зрительными образами как "взгляды", направленные на сложную систему с различных точек зрения. Набор из нескольких наглядных моделей (модельных взглядов) создает в сознании специалистов интегральный образ сложной компьютерной системы, которую они совместно проектируют. Вместе с тем, наглядные модели служат эффективным средством документирования компьютерных систем и их программных обеспечений, а также языком общения между программистами, системными аналитиками и заказчиками систем.

Наиболее известными визуальными моделями, используемыми для проектирования компьютерных систем и их программных обеспечений, являются диаграммы языка UML и стандарта IDEF0, таблицы и диаграммы стандарта IDEF1X. Эти визуальные модели имеют математическую основу в виде теорий графов, множеств и матриц.

Программируемые логические контроллеры (ПЛК, PLC). Программируемый контроллер представляет собой специализированное устройство для автоматизации технологических процессов.

В ПЛК преобладают логические команды над числовыми операциями.

Основные современные ПЛК:

* Siemens
* Овен
* Агава
* Omron
* Mitsubishi
* Wago
* Delta
* GE Fanuc
* Allen Bradley
* Toshiba
* Базис-100

Датчики и исполнительные устройства подключаются к ПЛК двумя основными способами:

1. централизованно, через модули ввода-вывода ПЛК, при этом подключение может выполняться прямыми проводными связями, либо при помощи специальных согласующих модулей;
2. распределенная периферия, в которой связи реализованы посредством каналов связей.

Для программирования ПЛК на языках стандарта IEC 61131-3 создана инструментальная среда IsaGraf. В этой среде поддерживается 7 языков программирования (4 визуальных, 3 текстовых). К текстовым относятся: C, Pascal, Ассемблер.

Среда программирования IsaGraf обеспечивает следующие возможности:

* возможность внесения изменений в код программы во время работы отладчика;
* трассировка рабочих переменных;
* интерактивная модификация переменных;
* запуск/останов программы;
* изменение продолжительности цикла выполнения приложения;
* управление временем выполнения;
* эмуляция входных сигналов.
* Среда IsaGraf работает в двух режимах:
* разработка;
* исполнение.

Курсовой проект разрабатывается в рамках научных исследований, которые проводятся в БГТУ совместно с ОАО «Информационные спутниковые системы» им. академика М.Ф. Решетнева (г. Железногорск, Красноярский край, Российская Федерация) по проблеме создания пакета прикладных программ (ППП «Электрик») для разработки БУ бортовой аппаратуры КА в составе Глобальной навигационной спутниковой системы «ГЛОНАСС».

ППП «Электрик» начал создаваться в период с 1977 по 1992 годы в сотрудничестве НПО «Прикладной механики», Института технической кибернетики АН БССР, Белорусского технологического института, Исследовательского центра станкостроения ГДР и более 20 лет находится в режиме промышленной эксплуатации.

Масштабность и значимость проекта определяют общие трудовые (более 300 чел/лет) и финансовые (около 1 млн. у.е.) затраты.

Состояние и концепции развития ППП «Электрик» опубликованы в журнале «Современная электроника» №6 за 2008г.

Теоретическая база разработки представлена в монографии: Акунович С.И., Гончаров А.А., Петренко Ю.Н. - «Дискретные системы логического управления технологических машин». Минск, «Юнипак», 2006г.

# Основные разделы

## Исходные данные

Для выполнения курсового проекта необходимо было реализовать программу на языке Delphi, выполняющую преобразование блок – схемы алгоритма, выполненной в программе Microsoft Visio, на язык VBSA. Пример алгоритма приведен на рисунке 3.

|  |
| --- |
|  |
| Рис.3. Алгоритм программы, выполненный в виде блок – схемы |

## Модель предметной области

Модель предметной области (рисунок 4) представляет собой UML-диаграмму классов. Она показывает, из каких классов состоит приложение, какие компоненты содержит каждый из классов, а также зависимость между классами и их компонентами.

|  |
| --- |
|  |
| Рис. 4. Модель предметной области |

|  |
| --- |
|  |

## Добавление компонента в проект

Проект был создан в Delphi 2011. Для работы с Microsoft Visio из Delphi была использована объектная модель Visio, путем создания COM – объекта класса Visio.

Для доступа к объектной модели Visio необходимо добавить в проект дополнительные компоненты. Это можно сделать, перейдя в меню в пункт Component – Import component. В появившемся меню следует выбрать пункт “Import a type library” (рис. 5).

|  |
| --- |
|  |
| Рис. 5. Меню добавления компонента. |

В появившемся меню следует выбрать “Microsoft Visio type library” и нажать Next.

|  |
| --- |
|  |
| Рис. 6. Выбор компонента из списка |

В следующем меню можно выбрать папку, в которой будет выполняться поиск подключаемых модулей. Укажем здесь папку проекта.

|  |
| --- |
|  |
| Рис. 7. Выбор директории |

В последнем меню укажем добавить новый модуль в проект.

|  |
| --- |
|  |
| Рис. 8. Добавление модуля в проект |

В папке с проектом будет сгенерирован новый файл Visio\_TLB.pas, который будет содержать необходимые классы для работы с Visio.

## Подключение компонентов

Компонент был добавлен, но еще не используется программой. Перейдем к диррективе uses файла проекта, и добавим в него Visio\_TLB – обьектная модель Visio, COMOBJ – модуль для работы с COM обьектами, и Generics.Collections – стандартные коллекции, которые будут использованы при написании программы. Конечный список импортируемых модулей должен выглядеть примерно так (рис. 9):

|  |
| --- |
|  |
| Рис. 9. Список подключаемых модулей |
|  |

## Реализация программы

При написании программы были реализованы функции, выполняющие считывание информации из файла Visio, и выполняющие их преобразование на язык VBSA. В основном они работают с обьектами класса Shape из обьектной модели Visio. Класс Shape представляет собой одну из фигур документа Visio – процесс, решение, подпроцесс, или соединительная стрелка между фигурами. За тип фигуры в обьектной модели Visio отвечает свойство NameU которое может иметь значения 'Terminator', 'Process', 'Decision', 'Dynamic connector', 'Predefined process', 'Annotation'.

Также важными свойствами класса Shape являются:

1. Index – порядковый уникальный номер фигуры на странице Visio;
2. Connects – массив, содержащий фигуры, которые соединены стрелкой. Доступен только для фигуры – стрелки, 'Dynamic connector';
3. Text – комментарий, записанный для фигуры;
4. CellsU – массив, из которого можно получить доступ к внутренним параметрам фигуры, недоступным из обьектной модели напрямую. Был использован для определения координаты фигуры CellsU['PinX'].ResultIU.

## Язык VBSA

Язык VBSA (Вербальная блок-схема алгоритма) служит для записи блок-схем алгоритмов по «картинке», т.е. по графическому представлению любыми средствами, включая рисунок на бумаге. Запись блок-схемы алгоритма на языке VBSA выполняется с помощью фраз, представленных в виде текстовых строк. Ключевые слова VBSA записываются на русском языке.

Синтаксис языка (Фразы VBSA)

* Нач:<ИМЯ> - блок начала алгоритма;
* Кон:<ИМЯ> - блок конца алгоритма;
* Ком:<ТЕКСТ> - блок комментария;
* <ИМЯ>=<ВЫРАЖЕНИЕ> - блок действия;
* <ИМЯ>(<СПИСОК ПАРАМЕТРОВ>) - блок вызова функции;
* <ИМЯ>() - блок вызова процедуры;
* Если (<УСЛОВИЕ>) То <ЧИСЛО> Иначе <ЧИСЛО> - блок решения;
* <ЧИСЛО> - точка слияния;
* <ЧИСЛО>: - метка блока.

Семантика языка

* Линейная последовательность блоков на схеме алгоритма, заканчивающаяся блоком решение, точкой слияния или блоком конца алгоритма считается одним объединенным блоком;
* Перед каждым объединенным блоком ставится метка блока. Первый объединенный блок (после блока начала алгоритма) должен иметь метку «1:»;
* Параметр <ЧИСЛО> точки слияния должен совпадать с параметром <ЧИСЛО> метки блока, следующего за точкой слияния;
* Если объединенный блок заканчивается точкой слияния, то она записывается в последней строке блока.

###### Пример записи блок-схем алгоритмов

Блок-схемы алгоритмов

|  |
| --- |
|  |
| Рис. 10. Блок – схемы алгоритмов |
|  |

|  |
| --- |
|  |
| Рис. 11. Блок-схемы алгоритмов на языке VBSA |
|  |

## Пример работы программы

В качестве исходных данных выступает документ visio.vsd расположенный в корне диска D. Он содержит блок – схему, которую необходимо преобразовать на язык VBSA.

|  |
| --- |
| E:\Акунович\kourse\1.PNG |
| Рис. 12. Исходные данные |
|  |
|  |

По нажатию кнопки в программе выполняется запуск данного документа в Microsoft Visio, из него извлекаются нужные данные, выполняется преобразование, после чего документ Visio закрывается, а программа заполняется текстом на языке VBSA, полученным из блок – схемы.

|  |
| --- |
| E:\Акунович\kourse\4.PNG |
| Рис. 13. Результат выполнения программы |
|  |
|  |

|  |
| --- |
|  |

# 

# Графическая часть

Изм.

Лист

№ докум.

Подпись

Дата

Лист

1

Разраб.

Драко А.М..

Провер.

Акунович С.И.

Н. Контр.

Утверд.

Структурная схема БУ БКУ

Лит.

ЛистовЛит.

* 1. Структурная схема БУ БКУ



## Описание: D:\акунович\funct.jpgФункциональная схема БУ БКУ

Изм.

Лист

№ докум.

Подпись

Дата

Лист

1

Разраб.

Драко А.М.

Провер.

Акунович С.И.

Н. Контр.

Утверд.

Функциональная схема БУ БКУ

Лит.

Листов

## Модель предметной области

Изм.

Лист

№ докум.

Подпись

Дата

Лист

1

Разраб.

Драко.А.М.

Провер.

Акунович С.И.

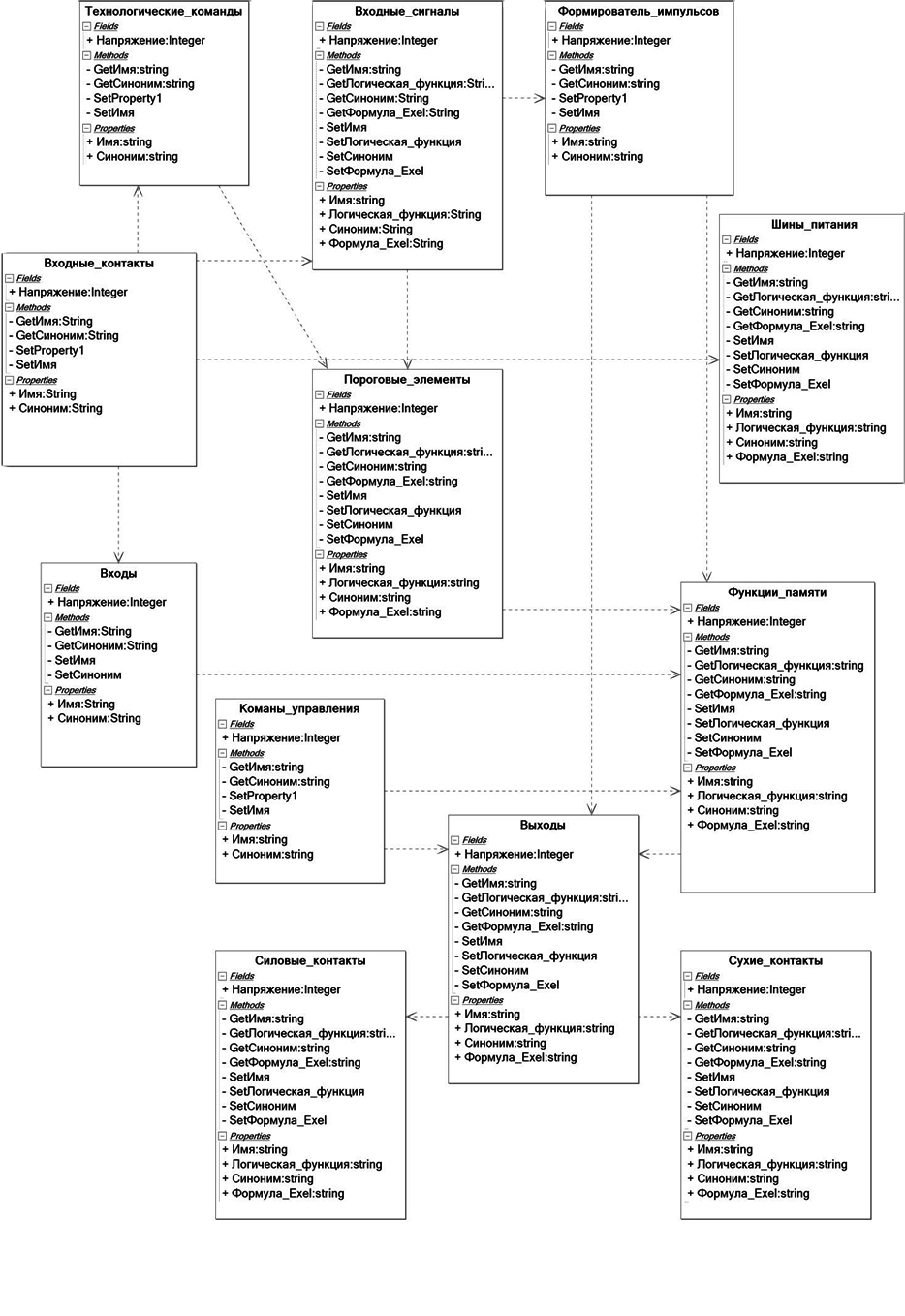
Н. Контр.

Утверд.

Модель предметной области

Лит.

Листов



## Схема программы

Изм.

Лист

№ докум.

Подпись

Дата

Лист

1

Разраб.

Драко.А.М.

Провер.

Акунович С.И.

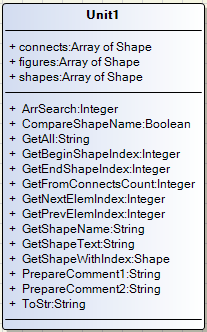
Н. Контр.

Утверд.

Схема программы

Лит.

Листов



# Заключение

При выполнении курсового проекта основным средством разработки была среда Delphi 2011.

Разработанная программа выполняет преобразование блок – схемы алгоритма в документе Visio в текст программы на языке VBSA (Вербальная блок-схема алгоритма). Данный язык является вспомогательным.

Развитием данной программы может быть дальнейшее преобразование полученного кода. В конечном итоге получение работающего текста алгоритма на нужном языке программирования по блок – схеме.

# Список использованных источников

Акунович, С. И. Композиционное конструирование моделей систем дискретного управления в среде DELPHI// Методические указания – Мн.: БГТУ, 2009.

Акунович, С. И. Дискретные системы логического управления технологических машин / С. И. Акунович, А. А. Гончаров, Ю. Н. Петренко. – Минск: Юнипак, 2006.

Уокенбах, Джон. Excel 2007. Профессиональное программирование на VBA. : Пер. с англ.– М.: Издательский дом «Диалектика», 2008.

Микропроцессорные и цифровые устройства полиграфического оборудования. Электронный документ.

А. Леоненков. Самоучитель UML. Электронный документ.

Е. М. Семенов. Проектирование систем управления в среде технологического программирования ISaGRAF. Санкт-Петербург 2007.

ISaGRAF. Version 3.4. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ. Электронный документ.