**Содержание**

Введение

1.Среда Delphi

2.Работа с коммуникационными портами

2.1. Описание COM-портов

2.2. Структура стандартного COM-порта

Список используемой литературы

**Введение**

В кратком изложении история языков программирования такова: изначально вычислительные машины программировались в машинном коде. То есть в их оперативную память напрямую вводили последовательность чисел, являющиеся кодами команд, которые процессор может выполнить. При этом программа составлялась с периодическим заглядыванием в таблицу кодов команд процессора и была отнюдь не наглядной.

Затем появилась идея обозначить коды какими-то короткими, но осмысленными, и потому легко запоминаемыми словами – мнемониками, и создать программу, которая бы, руководствуясь таблицей команд, переводила последовательность мнемоник – мнемокод в последовательность машинных кодов. Такую программу называют ассемблером (assembler - сборочное устройство, транслятор, ассемблер). Программы стали гораздо нагляднее, но решение практических задач требовало написания очень длинных программ (например, файловый менеджер Volkov Commander имеет размер около 64000 байт).

Тогда появились языки программирования высокого уровня. При их создании использовали то обстоятельство, что в программе часто встречаются участки одинакового кода, выполняющие какое либо одно действие: вывод строки, запись в файл, вычисление математической функции и т.д. В языках высокого уровня таким последовательностям кода присвоены имена, и программа составляется на условном языке, каждое, из слов которого заменяет десятки, а то и сотни команд процессора. Таким образом, программа становится еще нагляднее и короче. Существует множество условных языков высокого уровня, для каждого из них написано немало вариантов программы, переводящей условный код в последовательность машинных команд. Одной из таких программ является Delphi.

**1. Среда Delphi**

Система виртуального объектно-ориентированного программирования Delphi предназначена для разработки 32-разрядных приложений Windows. В состав Delphi входит библиотека VCL (Visual Component Library), содержащая компоненты для построения интерфейса приложения, а также структуры для хранения и обработки данных. Библиотека представляет собой иерархию объектов, имеющую общего предка - TObject.

Примерный вид экрана: сверху располагается главное окно, содержащее меню и панели инструментов. Панель, содержащая библиотеку компонентов, называется палитрой. Она состоит из нескольких страниц (вкладок), на которых компоненты сгруппированы по функциональному признаку. Слева располагается окно инспектора объектов. В его верхней части расположен раскрывающийся список компонентов, составляющих приложение. Для выбранного объекта на двух вкладках отражаются его свойства (Properties) и события, на которые он может реагировать (Events). С помощью инспектора объектов задаются свойства компонентов и обработчики событий.

Основную часть экрана занимает заготовка формы, представляющей собой окно будущего приложения. Форма предназначена для размещения компонентов. Под ней находится окно редактора кода, в котором пишется текст программы.

Процесс создания приложения Windows с помощью Delphi состоит из двух основных этапов: виртуального проектирования, то есть задания внешнего облика приложения, и определения его проведения путем написания процедур обработки событий.

*Виртуальное проектирование* состоит в помещении компонентов на форму и задании их свойств с помощью инспектора объектов. Самый простой способ разместить компонент – двойной щелчок на соответствующем значке палитры, при этом компонент помещается в центр формы. Затем его можно переместить и заменить его размеры с помощью мыши. Инспектор объектов отображает свойства выбранного в данный момент компонента (он окружен рамкой).

Задание свойств выполняется либо выбором из имеющихся вариантов, либо вводом требуемого значения с клавиатуры. Если около имени свойства стоит значок +, это означает, что свойство является объектом, содержащим другие свойства. Они становятся доступными с помощью щелчка на «плюсе».

Для определения *поведения программы* необходимо продумать, какие действия должны выполняться при щелчке на кнопках, вводе текста, выборе пункта меню и т.д. Заготовка шаблона обработчика события вызывается двойным щелчком на поле, расположенном слева от имени соответствующего события на вкладке Events инспектора объектов, при этом всплывает окно редактора кода. Для каждого компонента определен свой набор событий, на которые он может реагировать.

Наиболее часто используемые события:

- OnActivate – при получении формой фокуса ввода;

- OnClick, OnDbCluck – при одинарном и двойном щелчке мышью;

- OnClose – при закрытии формы;

- OnСreate - при создании формы;

- OnKeyDown, OnKeyUp – при нажатии и отпускании любой клавиши и их сочетаний;

- OnKeyPress- при нажатии клавиши, имеющей код ASCII;

- OnMouseDown, OnMouseUp – при нажатии и отпускании кнопки мыши;

- OnMouseMove – при перемещении мыши;

- OnPaint – при необходимости отрисовки формы.

Проектируемое с помощью Delphi приложение храниться в нескольких файлах. Описание файлов приведено ниже:

- .dpr - файл проекта. Это главная программа, в которой содержится код инициализации, завершения и запускается цикл обработки событий. Файл содержит ссылки на все формы и модули, входящие в состав проекта. Редактировать его вручную не рекомендуется (все изменения в проекте отслеживаются автоматически).

-.pas – исходный код модуля или формы. Большинство модулей связаны с формами и содержат тексты обработчиков событий элементов, размещенных на формах.

- .dfm – файл формы. Содержит описание внешнего вида формы и расположенных на ней элементов. Может просматриваться (но не редактироваться) с помощью среды Delphi как текстовый файл.

-.cfg, .dsk, .dof – файлы настроек компилятора, среды и проекта. Ручное редактирование этих файлов нежелательно.

-.res – файл ресурсов (создается автоматически). Включает значок программы, номер версии и т.д. Дополнять его своими ресурсами запрещается.

-.exe – исполняемый файл созданного приложения.

- ~dp, ~pa, ~df - резервные файлы проекта, модуля и формы. Создаются, если включен режим резервного копирования.

Среда Delphi оснащена мощной и удобной справочной системой. Чтобы вызвать ее, выбираем команду меню Help >Delphi Help или нажимаем клавишу F1 , установив курсор мыши на интересующем элементе.

**2. Работа с коммуникационными портами**

**2.1 Описание COM-портов**

Коммуникационные порты компьютеров (LPT и COM) широко используются для управления различными периферийными устройствами, такими как принтеры, сканеры, плоттеры, а так же для связи с промышленными объектами в системах автоматизации управления. Через последовательный порт так же осуществляется связь с удаленными объектами через модем, включая и выход в глобальную сеть Internet.

В то же время в учебно-методической литературе схемотехника и методика применения портов компьютера освещена достаточно слабо. В частности не освещены новые режимы работы параллельных портов, такие как EPP и ECP, используемые в современных компьютерах.

В данную работу включено описание принципов организации параллельных и последовательных портов и их схемотехники. Она позволяет изучить программные методы обмена информацией через указанные порты на низком уровне, что важно не только для решения конкретных задач, например, связи с принтером, но и для освещения основных принципов программного управления объектами.

**Архитектура ПК.**

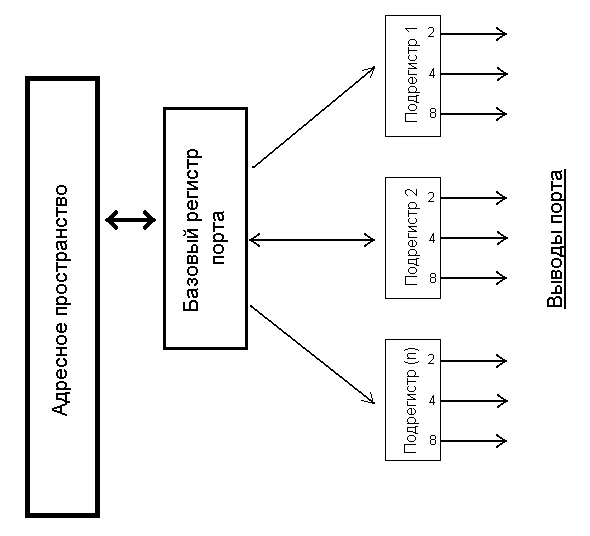
Сначала определимся, что же такое работа на высоком и на низком уровне. Под работой на высоком уровне понимают обращение к порту через ОС и под её контролем.

Дело в том, что в Windows обращение к любому устройству аналогично обращению к одноимённому файлу. То есть, если мы хотим считать данные из порта, например, COM1, то мы должны открыть файл с именем “COM1”. Для этого используются WinAPI функции OpenFile("COM1",,,,,), ReadFile, WriteFile, CreateFile и т.д. Их описание есть в «помощи» любого языка и в соответствующей литературе, в т.ч. и в **[1]**, по этому мы не будем на них подробно останавливаться. Эти функции являются функциями ОС, а не языка, по этому одинаковы во всех языках программирования. Отличается лишь форма их записи.

Под работой на низком уровне понимают непосредственное обращение к регистрам контроллера порта из адресного пространства. АП - это абстрактное понятие, выражающее отправную точку, относительно которой происходит адресация всех устройств компьютера.

Работа на высоком уровне позволяет легко реализовывать протоколы и функции, заложенные разработчиками ОС, но затрудняет реализацию собственных. Работа на низком уровне напротив, открывает широкие возможности для творчества, но реализация более-менее сложного протокола может потребовать знания ассемблера, много времени и, скорее всего, будет нагружать процессор лишними функциями. Кроме того, доступ к некоторым регистрам (а след. и ножкам) порта невозможно (без специальных знаний) получить из адресного пространства и наоборот, некоторые комбинации выводов невозможно «зажечь» с помощью API. Таким образом, работа на высоком и на низком уровне, это не взаимоисключающие понятия, а всего лишь разный подход к управлению портом.

**2.2 Структура стандартного порта**



**Рис.1 Структура стандартного порта.**

Этот макет наиболее наглядно представляет структуру стандартного порта ввода/вывода, к которым относятся COM и LPT.

Как видно, порт содержит базовый регистр – это первый этап на пути к нужной ножке из АП. Базовый регистр содержит в себе несколько подрегистров (ПР) (называемых ещё «смещением адреса»). В зависимости от их типа, связь их с БР может быть двусторонней (запись и чтение) или односторонней (запись или чтение).

Каждому ПР соответствует какое то количество ножек порта. Т.е. разные ножки порта принадлежат разным подрегистрам и для того, чтобы выставить на них какое-то состояние или считать его с какой-нибудь ножки, надо сначала обратится к соотв. ей ПР.

Некоторые ПР вообще не имеют непосредственного отношения к выводам порта, т.к. являются регистрами состояния. Их используют для настройки порта. Например, некоторое количество ножек порта могут работать на передачу/приём. Для того, чтобы установить режим их работы (передача, приём или передача и приём) в соотв. ПР порта записывают определённый код. Ориентируясь на него, контроллер порта устанавливает соотв. режим работы этих ножек.

Каждой ножке в ПР соответствует весовой коэффициент в двоичном коде. Т.е. для того, чтобы выставить “1” на каком то выводе порта, нужно записать в соответствующий ей подрегистр её весовой коэффициент. Напротив, при считывании состояния входной линии порта, порт вернёт не состояние “0” или “1”, а “0” или “N”, где N – весовой коэффициент данной ножки. Возвращение портом “0” означает, что на входе “0”. Возвращение же “N” означает, что на входе “1”.

Указанные действия в языке Паскаль 7.0 выполняет процедура port[]. Формат её записи следующий: **port[b+n]:=m;** Где:

b – адрес БР порта.

n – номер подрегистра.

m – весовой коэффициент вывода.

Попробуем с её помощью установить “1” на выводе RTS(7) COM порта.

Для этого обратимся к **таблице 1**:

**Таблица 1. Назначение контактов COM порта (разъём DB-9S.)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Сигнал. | Контакт разъёма DB-9S. | Направление. | Подрегистр. | Вес. |
| TxD | 3 | Передача. | Х | Х |
| RxD | 2 | Приём. | Х | Х |
| DTR | 4 | Передача. | 4 | 1 |
| DSR | 6 | Приём. | 6 | 32 |
| RTS | 7 | Передача. | 4 | 2 |
| CTS | 8 | Приём. | 6 | 16 |
| DCD | 1 | Приём. | 6 | 8 |
| RI | 9 | Приём. | 6 | 64 |
| GND | 5 | Земля. | X | X |

Как следует из таблицы, выводу RTS соответствует 7 контакт стандартного девятиконтактного COM порта. Он относится к 4 подрегистру и имеет в нём вес равный 2.

Адрес базового регистра, это адрес порта. Для COM1 он равен $3F8 (для COM2 - $2F8, для LPT1 - $378, LPT2 - $278).

Теперь разберёмся, как читать состояние входов.

Как уже говорилось выше, порт возвращает “0”, если на входе “0”, и “N”, если на данном входе “1”. Причём в некоторых случаях (особенно, при работе с LPT) “N” может быть равно сумме кодов ножек, если “1” присутствует не на одном входе.

Осуществляется приём следующим образом: e:=port[b+n] and m;

Где: e – переменная типа byte, в которую сохраняют принятое значение.

b – адрес БР порта.

n – номер подрегистра.

m – весовой коэффициент входа (см. **таблицу 1**).

Для проверки этой записи соединим перемычкой выход RTS(7) и вход CTS(8) и запустим приведённый ниже код: (**пример 1**)

**var**

**aa,e:word;**

**port,base,data:word;**

**{.........}**

**aa:=0;**

**base:=$3F8;**

**begin**

**port:=base+4;**

**data:=2;**

**outport(port,aa,data);**

**sleep(10);**

**port:=base+6;**

**e:=inport(port) and 16;**

**port:=base+4;**

**data:=0;**

**outport(port,aa,data);**

**end;**

Управлять конкретными выводами порта из под Дельфи можно и с помощью API. Это хоть и сложнее, зато такая программа будет работать на любом Windows, в т.ч. на NT, 2000, XP (приведённые выше пример работоспособен лишь в 95/98/МЕ).

**Список использованной литературы**

1) Работа с коммуникационными портами (COM и LPT) в программах для Win32. http://bcb.net.ru/article/hard/index003.html

2) Коммуникационные порты персонального компьютера. http://electronics.org.ua/techinfo/lpt/lpt.htm

3) Основы программирования LPT для DOS и Win9x. http://radiopirat.h11.ru/prog/prog004.htm

4) Примеры работы с АЦП. http://dikoy44.narod.ru/photoalbum.html

5) Вакуленко А. Программа LPTtest. – Радио, 2004, №8, с. 23,24.

6) Патрик Гёлль. Как превратить ПК в измерительный комплекс. ДМКпресс, 2002г.

7) Фаронов В.В. TurboPascal 7.0 практика программирования, учебное пособие, М., изд. Нолидж, 2003 г.

8) Павловская Т.А. Паскаль. Программирование на языке высокого уровня, учебник, П, 2003 г.