

**NEAR EAST UNIVERSITY**



**Faculty of Engineering**

**Department of Computer Engineering**

**DEVELOPMENT DELPHI PROGRAM  
FOR EXCHANGE AND CHANGE SYSTEM**

**Graduation Project  
COM- 400**

**Student: Ismail Mekki**

**Supervisor: Mr. Ümit ILHAN**

**Nicosia 2003 - 2004**

## ACKNOWLEDGEMENTS



*"No one can deny the role of the university in upgrading the mentality of the student to be capable to cover the requirements of the working life. This role is concerning with qualifying the student up to the level that the society needs. However, this role cannot be accomplished unless there is a qualified leader and sophisticated coach.*

*Fortunately, Mr. Umit ILHAN was the main reason of my success in this project, and thus, he deserves my all thanks, gratitude, and my respect due to his support and wise advice.*

*I appreciate all the effort that he provided during the preparation of this project.*

*Therefore, firstly, I would like to dedicate this project my family because of their unlimited support during my life.*

*Secondly, I would like to dedicate it to my supervisor Mr. Umit ILHAN because of his wise supervision on this project and for his wide knowledge.*

*Finally, I appreciate all the effort aids of my friends during the preparation of this project."*

*With all due respect*

## ABSTRACT

This program is designed for the change and exchange markets where the individuals deal with the currencies and barter them. This system is based on BORLAND DELPHI 6 programming language. All the criterions of this system are taken according upon the request of the Near East Bank and all the features of this software can be adjusted according to the desire of the customer.

All the screens that will appear in the usage of this program will be illustrated in the coming chapters in details. The calculations and the mathematical operations that this program applies are explained in details in the appendix at the end of this report. This program is designed to find out the exact profit of the company by using graphical charts and by showing statues reports at any time. Three hard currencies are taken into consideration in this system and they are compared with the Turkish lira, which is the local currency. The values of these hard currencies will be taken from the stock markets that the government decides daily according o the daily economic level. The decision maker in NEB will determine the amount of the profit that he desires and the total revenue will be calculating by the software. A navigation bar is located in the bottom of each screen for adding, deleting, saving... etc.

## TABLE OF CONTENTS

<b>ACKNOWLEDGEMENTS</b>	<b>I</b>
<b>ABSTRACT</b>	<b>II</b>
<b>TABLE OF CONTENTS</b>	<b>III</b>
<b>LIST OF FIGURES</b>	<b>V</b>
<b>CHAPTER ONE: INTRODUCTION</b>	<b>1</b>
1.1. Overview	1
1.2. Economic Analysis of Floating Exchange Rate System	2
1.2 a Introduction	2
1.2 b Opportunities From Around World	2
1.3 Delphi Programming	5
1.4. Database Programming	7
1.5. Borland Database Engine (BDE)	8
1.6. Graphical Data-Aware Control	8
1.7. The clintDataSet Component	9
1.8 Classic BDE component	10
1.9 Tables and Queries	10
1.10 DBNavigator and Dataset Actions	11
1.11 Text-Based Data-Aware Control	12
1.12 Navigator a Dataset	12
 <b>CHAPTER TWO: THE DESCRIPTION OF THE SYSTEM</b>	 <b>14</b>
2.1 Main Menu Screen	14
2.2 Change System	18
2.3. Exchange curve	19
2.4. Exchange System	21
2.5. Exchange Bar Graphs	22



2.6. Adding the Exchange rates	23
2.7. Database Desktop	25
<b>CHAPTER THREE: DATA SOURCE</b>	26
3.1. Data Flow Diagram	26
3.2 . Main Menu	27
3.3. Buy Change System Preview	48
3.4. Exchange System Sterling	51
3.5. Change System Sterling	52
3.6. Exchange System Sterling Dollar	52
3.7. Exchange System Dollar Euro	54
3.8. Change System Dollar Euro	55
3.9. Exchange System Euro	55
3.10. Exchange System Dollar Sterling	57
3.11. Change System Dollar Sterling	58
3.12. Sell Exchange System	58
3.13. Exchange Euro to Sterling	60
3.14. Exchange Euro	61
3.15. Exchange System Dollar to Euro	61
3.16. Buy Change System preview	63
3.17. Buy Change System	66
3.18. Buy Change System	67
3.19. Change System Sterling	68
3.20. Change System	69
3.21. Report Dollar	71
3.22. Sell Change System Preview	72
<b>CONCLUSION</b>	75
<b>SYSTEM REQUIREMENTS</b>	75
<b>REFERENCES</b>	76

## LIST OF FIGURES

### CHAPTER II

#### CHANGE AND EXCHANGE SYSTEMS

2.1 Main Menu Screen	14
2.2 Buy change system preview	15
2.3 Exchange report	15
2.4 Exchange rates	16
2.5 Change System Preview	17
2.6 Exchange system Preview	17
2.7 Change System	18
2.8 Change Curve	19
2.9 Change Curve	20
2.10 Change Curve	20
2.11 Exchange System	21
2.12 Exchange Bar Graphs a	22
2.13 Exchange Bar Graphs b	23
2.14 Adding the Exchange Rates (Selling Rates)	23
2.15 Adding the Exchange Rates (Foreign Currency Rates)	24
2.16 Database Desktop	25

### CHAPTER III

#### DATA SOURCE

3.1 Data Flow Diagram	26
-----------------------	----

# INTRODUCTION

## 1.1 Overview

Many activities are done daily in the stock market. Most of these activities are concerning with the transactions and exchanging the goods and services. Since the market is the place that the buyers and sellers exchange goods and services, then the currencies are considered as a good that the people buy and sell within the market.

As it is known, the most currencies that are used widely are those currencies that are called the hard currencies such as US Dollar, UK sterling, and Euro. Therefore, as we are concerning with controlling the activities in the market by issuing the required software programs that include all features of controlling the business by means of scientific methods in order to be used in a proper and easy way.

Using any programming language to create certain software gives you the opportunity of gaining a good experience and it will promote your approaches of analyzing the aspects of any given project.

Therefore, any software needs an adequate and sufficient programming language that helps us to set up all the applications and all the functions that include the features of organizing and sustaining our business.

The role of a computer engineer requires an adequate ability of analyzing and dealing with all the aspects of any project in order to be capable to create and issue certain software that controls the business. Therefore, our role as an engineer is to not only deal with the hardware, but also deal with creating a software programs. Thus, the role of an engineer is to control the technical specifications of the firm and to know every single aspect that is necessary to measure the level of the success.



## **1.2 Economic Analysis of Floating Exchange Rate Systems**

### **1.2.a Introduction**

Associated foreign exchange is looking for experienced foreign exchange sales professionals! Successful candidates will be strongly motivated individuals who can identify and cultivate potential corporate clients with international payment needs. Strong communication skills, experience in the foreign exchange field and experience in relationship based selling are a must. Successful candidates will be responsible for opening, maintaining and growing accounts through relationship building, identifying customer needs and cross-selling appropriate services.

### **1.2 b Opportunities from Around the World**

Over the last three decades the foreign exchange market has become the world's largest financial market, with over \$1.5 trillion USD traded daily. The primary market for currencies is the 24-hour Interbank market. The Interbank market literally follows the sun around the world, moving from major banking centers of the United States to Australia and New Zealand to the Far East, to Europe and finally back to the United States. With the large minimum transaction sizes and often-stringent financial requirements, banks, hedge funds, major currency dealers and the occasional high net-worth individual speculator were the principal participants. These large traders were able to take advantage of the many benefits offered by the forex market vs. other markets including fantastic liquidity and the strong trending nature of the world's primary currency exchange rates.

The business section of any newspaper will have a table of spot exchange rates. These are the rates at which a person could have bought other currencies or foreign Exchange, such as the English Pound, French Franc, or the new European Euro. The Prices of foreign currencies can be determined in two major types of exchange rate Systems. In the United States, the dollar's exchange rates are determined by the Marketplace, i.e., by supply and demand. This type of system is called a floating Exchange rate system. In other countries, governments set the price of their currencies With respect to other countries. They then buy or sell foreign exchange at the prices They've set. This is called a fixed exchange rate system. The economic effects of these Two systems can be very different. However, in either system the underlying forces Influencing the value of a country's currency remain the same. Due to possible



confusion of being able to quote different currencies in terms of Each other, e.g., \$/£ or £/\$, we need to explicitly define an exchange rate. An **exchange Rate** is, therefore, the domestic cost of a unit of foreign exchange. For example, from the US perspective the price of the English Pound would be denominated as the number of US dollars per pound, or \$/£. As noted above, the exchange rate in a floating exchange rate system is Determined by market forces. Our definition of the exchange rate defines the market as The market for foreign exchange. In this market we have demanders and suppliers of Foreign currencies willing to pay and accept dollars in return for these currencies. We Will in turn discuss the demand and supply of foreign exchange. Foreign Exchange Demand The demand for foreign exchange is a derived demand. With the exception of currency Collectors, the demand for foreign exchange is due to people's desire to use it in the Purchase of foreign goods or financial assets. Foreign exchange demand is, therefore, Highly sensitive to changes in these desires.

In order to understand changes in the demand for foreign exchange, we will need to Discuss its underlying forces. These are the demand for foreign goods and services and the demand for foreign financial assets. The supply of foreign exchange has at its roots the same conceptual basis as Demand, only it is from the foreign perspective. Foreign currency is supplied to the Foreign exchange market when foreigners exchange their currency for dollars in order to Buy US goods or financial assets. Equivalently, the supply of foreign exchange is Nothing more than a mirror image of the foreign demand for US currency. Exchange is the mirror of the supply of dollars to the foreign exchange market. One question which might arise is which foreign exchange Market. New York, London, Frankfurt and Tokyo are Major financial centers with large foreign exchange markets. The answer as to which market is all of them. The first rule of business is to buy low and sell high. Should exchange rates be different across different financial centers, then the opportunity for arbitrage profits occurs. Currency dealers will buy low in one center and sell high in another, driving exchange rates into equality Across the different markets. For example, should the Swiss Franc be at a lower price (in terms of \$) in London and at a higher price in New York, then the dealers will increase the demand for the Swiss Franc in London, driving up its price, and increase its supply in New York, driving down its price there. This continues until the price is the same in both places. The major questions to be addressed are how exchange rates determined are and what the forces which influence them are. In Figure 1, the equilibrium exchange rate (e) is the one where the quantity demanded is equal to the quantity supplied for

foreign exchange. As with most markets, the price changes in order to equilibrate the market. When quantity demanded exceeds quantity supplied, and then the exchange rate will rise. If the quantity supplied is greater than quantity demanded, the exchange rate falls. What does it mean when the exchange rate rises or falls? As we have defined the Exchange rate (\$/£), when the exchange rate rises, the value of the dollar decreases or depreciates. It now takes more dollars to buy an English pound than it did before the Change in the exchange rate. Fewer foreign goods can now be purchased for a given number of dollars. The reverse is also true. As the exchange rate falls, the dollar cost of foreign exchange falls, increasing the dollar's value. This is termed an **appreciation** of the dollar. More foreign exchange rate \$/£ Foreign exchange Sfx or D\$ Dfx or S\$. Market should force lead to a change in either the supply or demand for foreign exchange then the exchange will change accordingly to re-equilibrate the market. The basic notion is that exchange rates are sensitive to differential inflation rates across Countries. Should the domestic inflation rate rise at a rate greater than our trading Partners, then at a given exchange rate, the price of domestic goods will be rising relative To foreign goods. This will, in turn, increase the demand for foreign goods (imports are Now cheaper in domestic currency terms) and decrease the demand for domestic exports (Domestic exports are now more expensive in foreign currency terms). This results in an Increase in the demand for foreign exchange, as well as a decrease in the supply of Foreign exchange.

This is a long-run effect because of the Law of One Price. This concept states that in the Long run the price of tradable goods must be the same across countries. If this was not The case, then the opportunity for arbitrage profits, buying low in one country and selling High in another, would result in a movement in the exchange rate bringing about the Equalization. For example, suppose Argentine wheat, at the prevailing exchange rate, is cheap in US Dollars. As North Americans buy more and more Argentine wheat, they increase the Demand for the Argentine currency, driving up its value, thus making wheat more Expensive in dollar terms. The exchange rates which would prevail under the Law of One Price are called purchasing power parity exchange rates (PPP). While these do not exist in reality (there are many other factors affecting exchange rates) there is an underlying pressure moving exchange rates in this fashion. PPP exchange rates are used in comparing the economic performance between countries. The World Bank compares countries in their *World Development Report* using a PPP exchange rate. Medium Term - Differential Growth Rates As an economy



grows, its demand for imports will also grow. As income increases, some portion of that increase will be spent on imported goods. In the jargon of macroeconomics, the proportion of the additional dollar of income spent on imports is called the marginal propensity to import. Assume that the marginal propensity to import is the same across countries. Should a country's economy grow faster than its trading partners, then its demand for imports will also be growing faster? In the context of Figure 4, this is represented by increases in both the demand and supply of foreign exchange, but the demand would increase by more. This would result in a slight depreciation of the domestic currency. Short-Run - Differential Interest Rates This factor has become extremely important as countries have liberalized their economies, allowing the flow of financial capital into and out of their countries. It has played an important role in the East Asian and Mexican Peso financial crises. Exchange rate \$/£ Foreign exchange.

### 1.3 Delphi Programming

Delphi 5 provided new features to the Object Inspector, and Delphi 6 includes even more additions to it. As this is a tool programmer's use all the time, along with the editor and the Form Designer, its improvements are really significant.

The most important change in Delphi 6 is the ability of the Object Inspector to expand component references in-place. Properties referring to other components are now displayed in a different color and can be expanded by selecting the + symbol on the left, as it happens with internal subcomponents. You can then modify the properties of that other component without having to select it.

**NOTE** This interface-expansion feature also supports subcomponents, as demonstrated by the new Labeled Edit control. The Form Designer

**TIP** A related feature of the Object Inspector is the ability to select the component referenced by a property. To do this, double-click the property value with the left mouse button while keeping the Ctrl key pressed. For example, if you have a Main Menu component in a form and you are looking at the properties of the form in the Object Inspector, you can select the Main Menu component by moving to the Main Menu property of the form and Ctrl+double-clicking the value of this property. This selects the main menu indicated as the value of the property in

the Object Inspector. Here are some other relevant changes of the Object Inspector: The list at the top of the Object Inspector shows the type of the object and

can be removed to save some space (and considering the presence of the Object Tree View). The properties that reference an object are now a different color and may be expanded without changing the selection. You can optionally also view read-only properties in the Object Inspector. Of course, they are grayed out. The Object Inspector has a new Properties dialog box which allows you to customize the colors of the various types of properties and the overall behavior of this window.

The Project Manager doesn't provide a way to set the options of two different projects at one time. What you can do instead is invoke the Project Options dialog from the Project Manager for each project. The first page of Project Options (Forms) lists the forms that should be created automatically at program startup and the forms that are created manually by the program.

The next page (Application) is used to set the name of the application and the name of its Help file, and to choose its icon. Other Project Options choices relate to the Delphi compiler and linker, version information, and the use of run-time packages.

There are two ways to set compiler options. One is to use the Compiler page of the Project Options dialog. The other is to set or remove individual options in the source code with the `{SX+}` or `{SX-}` commands, where you'd replace *X* with the option you want to set. This second approach is more flexible, since it allows you to change an option only for a specific source-code file, or even for just a few lines of code. The source-level options override the compile-level options.

All project options are saved automatically with the project, but in a separate file with a .DOF extension. This is a text file you can easily edit. You should not delete this file if you have changed any of the default options. Delphi also saves the compiler options in another format in a CFG file, for command-line compilation. The two files have similar content but a different format: The *dcc* command-line compiler, in fact, cannot use .DOF files, but needs the .CFG format. Another alternative for saving compiler options is to press Ctrl+O+O (press the O key twice while keeping Ctrl pressed). This inserts, at the top of the current unit, compiler directives that correspond to the current project options, as in the following listing: `{SA+,B-,C+,D+,E-,F-,G+,H+,I+,J+,K-,L+,M-,N+,O+,P+,Q-,R-,S-,T-,U-,V+,W-,X+,Y+,Zl}`

Memory management in Delphi is subject to three rules: Every object must be created before it can be used; every object must be destroyed after it has been used; and every object must be destroyed only once. Whether you have to do these operations in



your code, or you can let Delphi handle memory management for you, depends on the model you choose among the different approaches provided by Delphi.

Delphi supports three types of memory management for dynamic elements (that is, elements not in the stack and the global memory area):

- Every time you create an object explicitly, in the code of your application, you should also free it. If you fail to do so, the memory used by that object won't be released for other objects until the program terminates.

- When you create a component, you can specify an owner component, passing the owner to the component constructor. The owner component (often a form) becomes responsible for destroying all the objects it owns. In other words, when you free the form, it frees all the components it owns. So, if you create a component and give it an owner, you don't have to remember to destroy it. This is the standard behavior of the components you create at design time by placing them on a form or data module.

- When you allocate memory for strings, dynamic arrays, and objects referenced by interface variables, Delphi automatically frees the memory when the reference goes out of scope. You don't need to free a string: when it becomes unreachable, its memory is released.

## **1.4 Database Programming**

Delphi's support for database applications is one of the key features of the programming environment. Many programmers spend most of their time writing data-access code, which needs to be the most robust portion of a database application. This chapter provides an overview of Delphi's extensive support for database programming. What you will find here is a discussion of the theory of database design. I am assuming that you already know the fundamentals of database design and have already designed the structure of a database. I will not look into database-specific problems; my goal is to help you understand how Delphi supports database access. I will begin with an explanation of the alternatives Delphi offers in terms of data access, and then I will provide an overview of the database components that I have used in my program. This chapter includes an overview of the `TDataSet` class, an in-depth analysis of the `TField` components, and the use of data-aware controls. The following chapters will provide information on more advanced database programming topics, such as client/server programming, the use of `dbGo`, `dbExpress`, and `InterBase Express`.

## 1.5 Borland Database Engine (BDE)

The BDE originated with Paradox, well before Delphi existed, and was extended by Borland to support other local databases and many SQL servers. The BDE has direct access to dBASE, Paradox, ASCII, FoxPro, and Access tables. A series of drivers (called SQL Links and available only in Delphi Enterprise) allows access to some SQL servers, including Oracle, Sybase, Microsoft, Informix, InterBase, and DB2 servers. If you need access to a different database, the BDE can also interface with ODBC drivers.

## 1.6 Graphical Data-Aware Controls

Finally, Delphi includes two graphical data-aware controls:

- **DBImage**, which is an extension of an Image component that shows a picture stored in a BLOB field (provided the database use a graphic format that the Image component supports, such as BMP and JPEG). The output of the Cust- Lookup example, with the BLookupComboBox showing multiple fields in its drop-down list.

- **DBChart** is a data-aware business graphic component or the data-aware version of the TeeChart control built by David Berneda. To demonstrate the use of the DBChart control, I have added this component to a simple example showing a data grid. The application, called ChartDB, shows a pie chart with the surface of each country of the COUNTRY.DB table. The program has almost no code, as all the settings can be done using the specific component editor, which has several options but is quite easy to use. Here are some of the key properties of the component, taken from the form description:

**object** DBChart1: TDBChart

Legend.Visible = False

Align = alClient

**object** Series1: TPieSeries

Marks.ArrowLength = 8

Marks.Visible = True

DataSource = Table1

XLabelsSource = 'Name'

ExplodeBiggest = 3

OtherSlice.Style = poBelowPercent

OtherSlice.Text = 'Others'



```

OtherSlice.Value = 2
PieValues.ValueSource = 'Area'
end;
end.

```

What I have done is show the area field as the data source for the pie chart (the PieValues ValueSource property of the series), use the name field for the labels (the XLabelsSource property of the series), and condense all the countries with a value below 2 percent in a single section indicated as Others (the OtherSlide subproperties). As a minor addition to the code, I have added two radio buttons you can use to toggle between the area and the population. The code of the two radio buttons simply sets the source of the series, after casting it to the proper series type, as in:

```

procedure TForm1.RadioPopulationClick(Sender: TObject);
begin
  DBChart1.Title.Text [0] := 'Population of Countries';
  (DBChart1.Series [0] as TPieSeries).PieValues.ValueSource := 'Population';
end;

```

## 1.7 The ClientDataSet Component

Finally, there is a component derived from TDataSet that has a peculiar behavior and can be combined with other data-access components. The ClientDataSet component, in fact, is a dataset accessing data kept in memory. The in-memory data can be totally temporary (lost as you exit the program), saved to a local file as a snapshot, and imported by another dataset using a Provider component. This last situation is certainly the most common: You can hook a ClientDataSet to any other local dataset, or use Borland's multitier support (discussed in Accessing a Database: BDE, dbExpress, and other alternatives "Multitier Database Applications with DataSnap") to retrieve data from a dataset hosted by a different application, possibly running on a separate computer. The ClientDataSet component becomes particularly useful if the data-access components you are using provide limited or no caching. This is particularly true of the new dbExpress engine, but can equally help you when using the BDE or other native components.

On the other hand, ADO already provides most of the services of the `ClientDataSet` component and using these two at the same time can be useful only in limited situations

## 1.8 Classic BDE Components

Each of the database-access solutions discussed above has its own set of data-access, database connection, and extra utility components on a specific page of the Component palette. The classic BDE components have been moved to the new BDE page and include the Table, Query, and StoredProc components. The ADO, dbExpress, and InterBase Express components are each in specific pages, and all include specific dataset components and others that tend to mimic the BDE components, simplifying the porting of existing applications.

The Data Access page of the Component palette includes only the Data Source Component and others not specifically related with any single data access technology. Besides the data-access component of your choice, a Delphi visual application generally uses some data-aware controls (in the Data Controls page) and the DataSource component. Data-aware controls are visual components used to view and edit the data in a form and are extensions of standard components such as edit and list boxes, radio buttons, images, and the

Grid. The DataSource component has the role of connector between the data-aware controls and a dataset component.

## 1.9 Tables and Queries

The simplest traditional way to specify data access in Delphi was to use the BDE Table component. A Table object simply refers to a database table. When you use a Table component, you need to indicate the name of the database you want to use in its DatabaseName property. You can enter an alias or the path of the directory with the table files. The Object Inspector lists the available names, which depend on the aliases installed in the BDE. You also need to indicate a proper value in the TableName property. The Object Inspector lists the available tables of the current database (or directory), so you should generally select the DatabaseName property first. Another classic dataset is the BDE Query component. A query requires a SQL language



command. You can customize a query using SQL more easily than you can customize a table (as long as you know at least the basic elements of SQL, of course). The Query component has a `DatabaseName` property like the Table component, but it does not have a `TableName` property. The table is indicated in the SQL statement, stored in the SQL property. For example, you can write a simple SQL statement like this:

**Select \* from** Country where Country is the name of a table and the asterisk (\*) indicates that you want to use all of the fields in the table.

The efficiency of a table or a query varies depending on the database you are using. In general, we can say that the Table component tends to be faster on local tables, while the Query component tends to be faster on SQL servers, although this is just a very general rule, and in many cases you might have the opposite effect. We'll see some efficiency issues while discussing

client/server development in the third BDE dataset component is `StoredProc`, which refers to stored procedures of a SQL server database. You can run these procedures and get the results in the form of a database table. Stored procedures can only be used with SQL servers.

## 1.10 DBNavigator and Dataset Actions

DBNavigator is a collection of buttons used to navigate and perform actions on the database. You can disable some of the buttons of the DBNavigator control, by removing some of the elements of the `VisibleButtons` set. The buttons perform basic actions on the connected dataset, so you can easily replace them

With your own toolbar, particularly if you use an `ActionList` component with the predefined database actions provided by Delphi. In this case, in fact, you get all the standard behaviors, but you'll also see the various buttons enabled only when their action is legitimate. **TIP** If you use the standard actions, you can avoid connecting them to a specific `DataSource` component, and the actions will be applied to the dataset connected to the visual control that currently has the input focus. This way a single toolbar can be used for multiple datasets displayed by a form.

## 1.11 Text-Based Data-Aware Controls

There are multiple text-oriented components:

DBText displays the contents of a field that cannot be modified by the user. It is a data aware Label graphical control. It can be very useful, but users might confuse this control with the plain labels that indicate the content of each field-based control. DBEdit lets the user edit a field (change the current value) using an Edit control. At times, you might want to disable editing and use a DBEdit as if it were a DBText, but highlighting the fact that this is data coming from the database. DBMemo lets the user see and modify a large text field, eventually stored in a memo or BLOB (binary large object) field. It resembles the Memo component and has full editing capabilities, but all the text is rendered in a single font.

DBRichEdit is a component that lets the user edit a formatted text file; it is based on a RichEdit Windows common control and, in contrast to DBMemo, it allows text with multiple fonts and paragraph styles.

## 1.12 Navigating a Dataset

We've seen that a dataset has only one active record, and you can imagine that the active record changes often, in response of user actions or because of internal commands given to the dataset. To move around the dataset and change the active record, there are methods of the TDataSet class, particularly in the section commented as "position, movement." You can move to the next or previous record, jump back and forth by

A given number of records (with MoveBy), or go directly to the first or last record of the dataset. These operations of the dataset are generally available in the DBNavigator component or in the standard dataset actions, and they are not particularly complex to understand. What is not obvious, though, is how a dataset handles the extreme positions. If you open any dataset with a navigator attached, you can see that as you move on record by record, the Next button remains enabled even when you've reached the last record. It's only when you try to move forward after the last record that the current record apparently doesn't change and the button is disabled. This is because the Eof test (end of file) succeeds only when the cursor has been moved to a special position after the last record. If you jump to the end with the Last button, instead, you'll immediately be at the very end. You'll see exactly the same behavior for the first record



(and the Bof test). As we'll see in a while, this approach is very handy, as we can scan a dataset testing for Eof to be True and, at this point, we know we've also already processed the last record of the dataset.

**NOTE** Handling this special record positions before the beginning and after the end of the dataset, which are called *cracks*, is very important (and quite confusing) when you write a custom dataset. Besides moving around record by record or by a given number of records, programs might need to jump to specific records or positions. Some datasets support the RecordCount property and allow movement to a record at a given position in the dataset using the RecNo property. These properties can be used only for datasets that support positions natively, which basically excludes all client/server architectures, unless you grab all of the records in a local cache (something you'll generally want to avoid) and then navigate on the cache. As we'll see in the next chapter, when you open a query on a SQL server you fetch only the records you are using, so Delphi doesn't know the record count, at least not in advance. There are two alternatives you can use to refer to a record in a dataset, regardless of its type. You can save a reference to the current record and then jump back to it after moving around. This is accomplished by using bookmarks, either in the TBookmark or the more modern TBookmarkStr form. You can locate a record of the dataset matching given criteria, using the Locate method. This even works after you close and reopen the dataset, because you're working at a logical (and not physical) level. This approach is presented in the next section.

## 2.1 Main Menu Screen

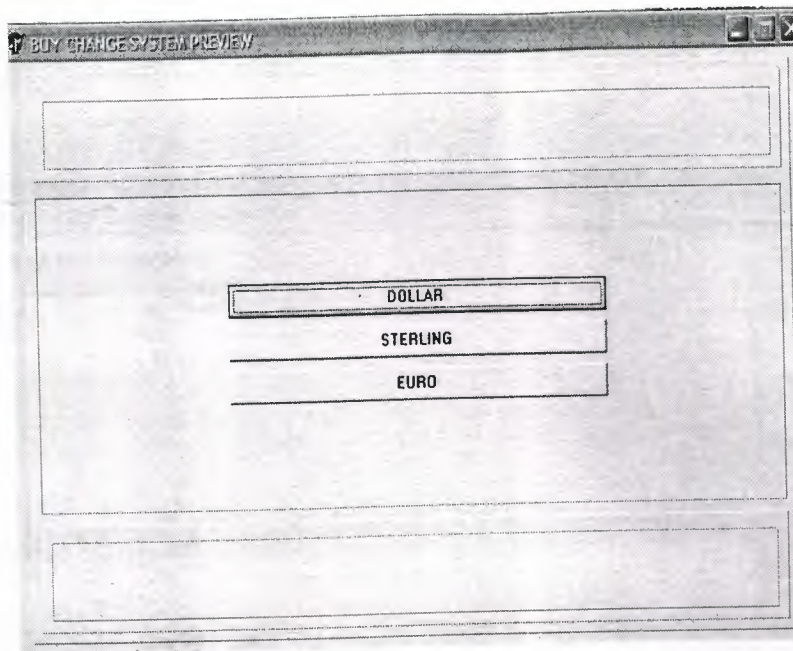


Figure 2.1 Main Menu Screen

The figure above shows the first screen which occurs when you first start running the program. This screen contains the following:

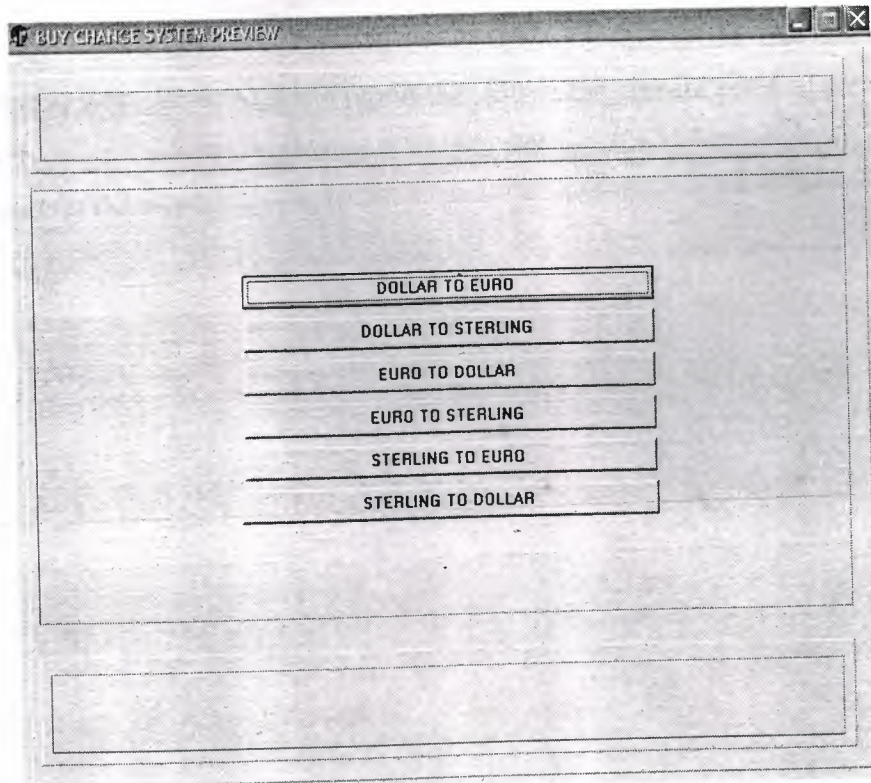
- ◆ The calendar: This automatically obtains the current date month and year of the system which the program is running on.
- ◆ Buy change system preview: which shows the current selling price of currencies that you have entered in the change system, which will be explained later.





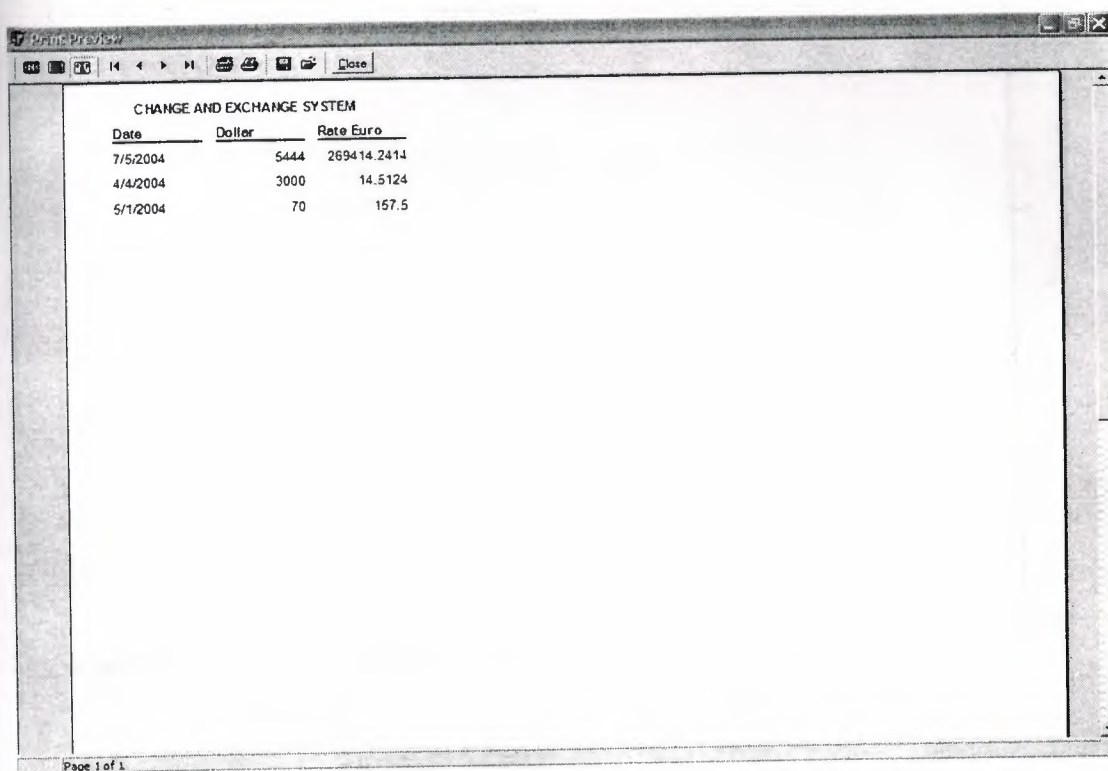
**Figure 2.2** Buy change system preview

- ◆ Sell exchange system preview: which shows the current reports of the exchange system. The following screen will appear and allow you to Select the type of exchange report that you wish to preview.



**Figure 2.3** Exchange report

The following report preview will show the exchange rates related to which form you choose:



Date	Dollar	Rate Euro
7/5/2004	5444	269414.2414
4/4/2004	3000	14.5124
5/1/2004	70	157.5

Figure 2.4 Exchange rates

- ◆ Change system preview: it is a report that shows the current price of the currencies in the stuck exchange related to selling currencies, which can be printed by the button next to it.



Print Preview

CHANGE AND EXCHANGE SYSTEM

Turkish	Sterling	Dollar	Euro	Date
1430000	2400000	1400000	2700000	1/1/2004

Page 1 of 1

Figure 2.5 Change System Preview

- ◆ Exchange system: it is a report that shows the current price of the currencies in the stuck exchange related to buying currencies, which can be printed by the button next to it.

Print Preview

CHANGE AND EXCHANGE SYSTEM

Turkish	Sterling	Dollar	Euro	Date
1200000	1400000	1700000	2000000	1/1/2004

Page 1 of 1

Figure 2.6 Exchange System Preview

## 2.2 Change System

MAIN MENU CHANGE EXCHANGE SYSTEM

NEAR EAST UNIVERSITY  
DEPARTMENT OF COMPUTER ENGINEERING  
DEVELOPMENT DELPHI PROGRAM FOR EXCHANGE AND CHANGE SYSTEM

CHANGE SYSTEM REPORT CHANGE SYSTEM

TURKISH TO DOLLAR | TURKISH TO STERLING | **TURKISH TO EURO**

TL Turkish	£ Sterling	\$ Dollar	€ Euro
1430000	2400000	1400000	2700000

Date  
1/1/2004

ADD TO CHANGE SYSTEM:

Date  
7/5/2004

EXCHANGE RATE	EXCHANGE AMOUNT	EXCHANGE VALUE
27000000	100	270000000

Figure 2.7 Change System

This screen consists of three sub-menus, which will allow you to complete your transaction processes. In addition, the forms contain the following:

The first part of each form grapes the exchange rates of the currencies from your database that you have entered.

The second part deals with the current amount of currency that you are exchanging at the time being. It consists of a navigation bar that allows you to scroll in the fields of your transactions.



## 2.3 Exchange Curves

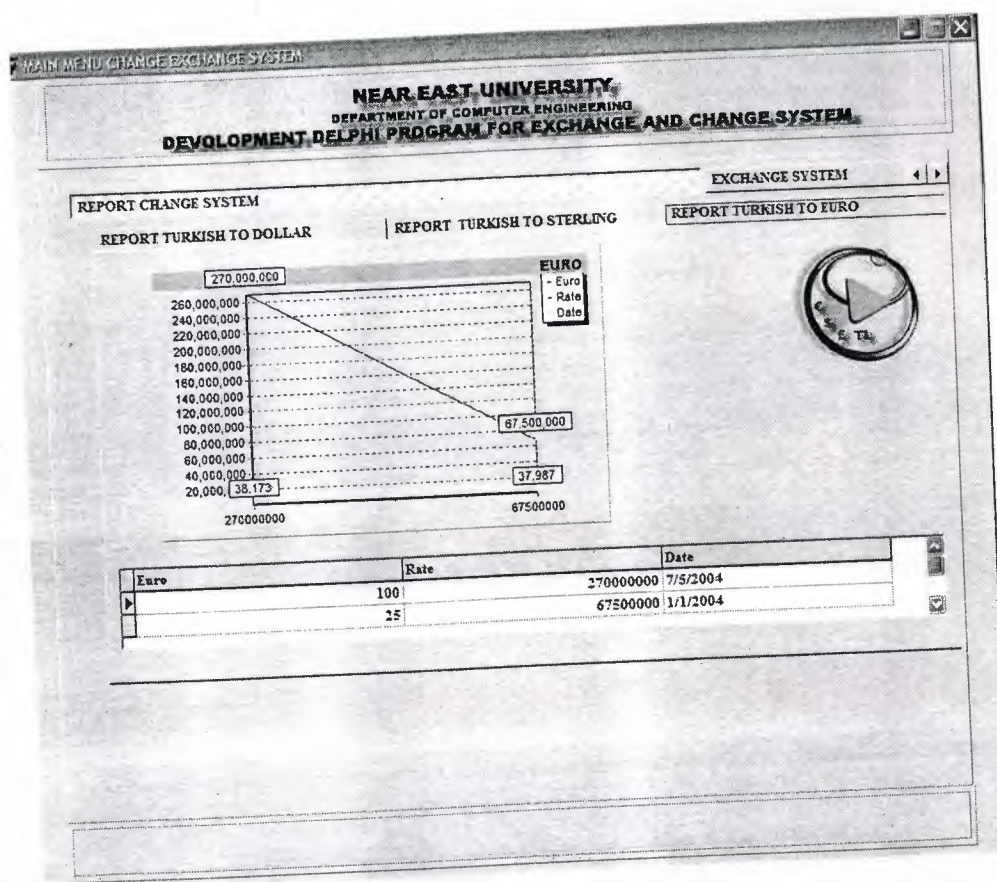


Figure 2.8 Change Curve

It consists of three chart reports that are all linked to the data base and change in respect to the processes that you make, that each transaction made is added or deleted and presented in the as a curve in respect to the money amount you have processed. The rest of the graphs are shown below

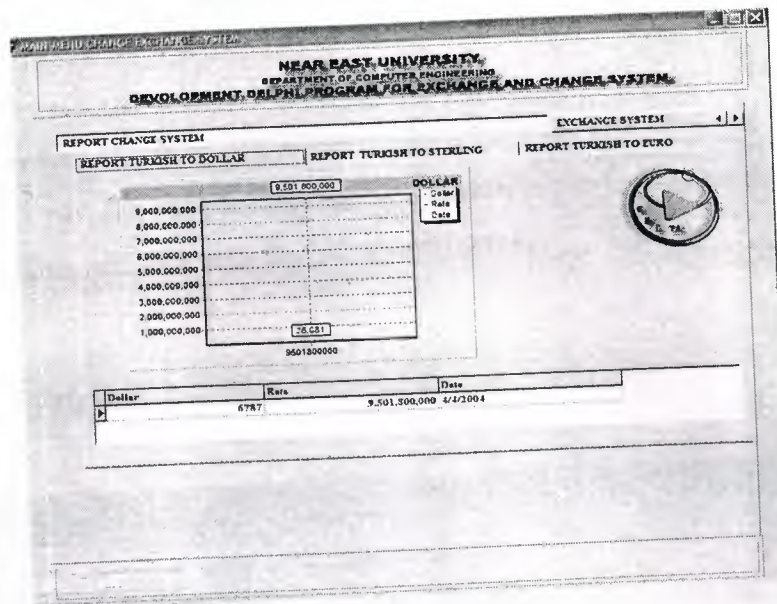


Figure 2.9 Change Curve

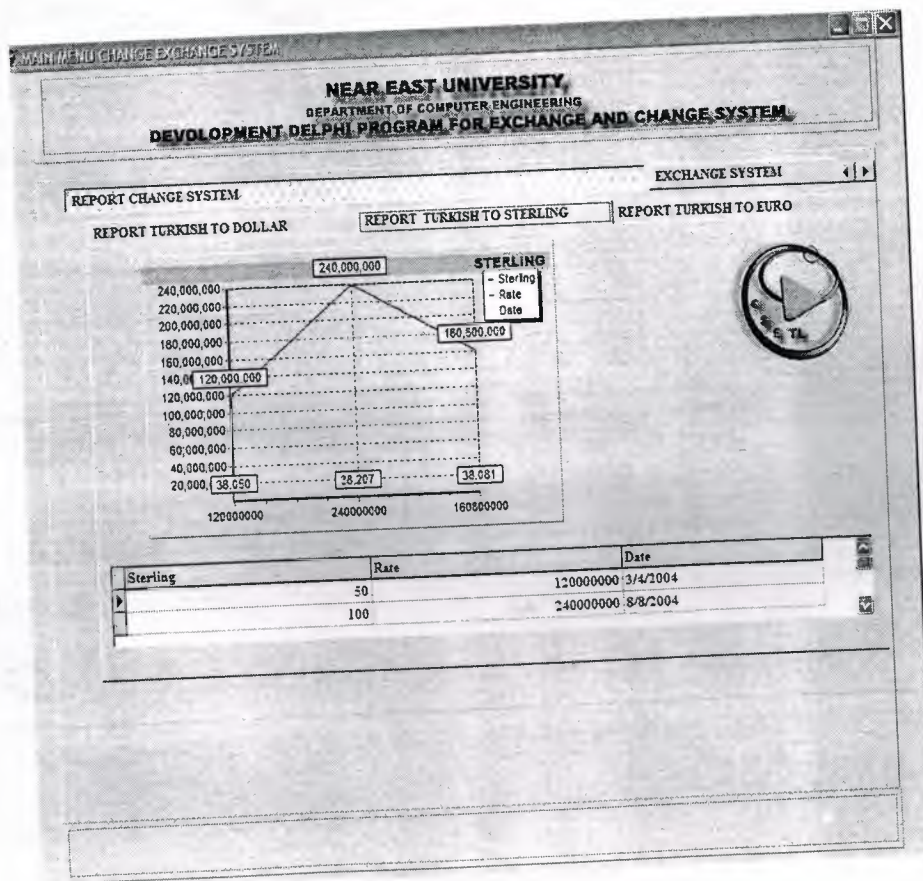


Figure 2.10 Change Curve



These graphs are just to keep track of you exchange processes in terms of money, and also shows you which of the currencies is have the most demand in the market, at this point you can analyze the current situation of the foreign currencies flow in you exchange office and take your decisions among that.

## 2.4 Exchange System

MAIN MENU CHANGE EXCHANGE SYSTEM

**NEAR EAST UNIVERSITY**  
DEPARTMENT OF COMPUTER ENGINEERING  
DEVELOPMENT DELPHI PROGRAM FOR EXCHANGE AND CHANGE SYSTEM

EXCHANGE SYSTEM INFO CHANGE SYSTEM

EURO TO DOLLAR

CHANGE	[TL] Turkish	[£] Sterling	[\$] Dollar	[€] Euro	Date
	1430000	2400000	1400000	2700000	1/1/2004

EXCHANGE

EXCHANGE	[TL] Turkish	[£] Sterling	[\$] Dollar	[€] Euro	Date
	1200000	1400000	1700000	2000000	1/1/2004

EXCHANGE

EXCHANGE	[\$] Dollars given amount	[\$] DOLLAR PRICE	[€] woned ratio	equal amount	Date
	54345			40081.7156	7/8/2004




Figure 2.11 Exchange System

The exchange system consists of three fields that allow you to deal with the exchange of the foreign currencies each one consist of selling and buying prices of the currencies and another field for your processes, where you can add delete or modify your processes.

## 2.5 Exchange Bar Graphs

The following bar graphs illustrate the change in the currencies by date and respectively with the database.

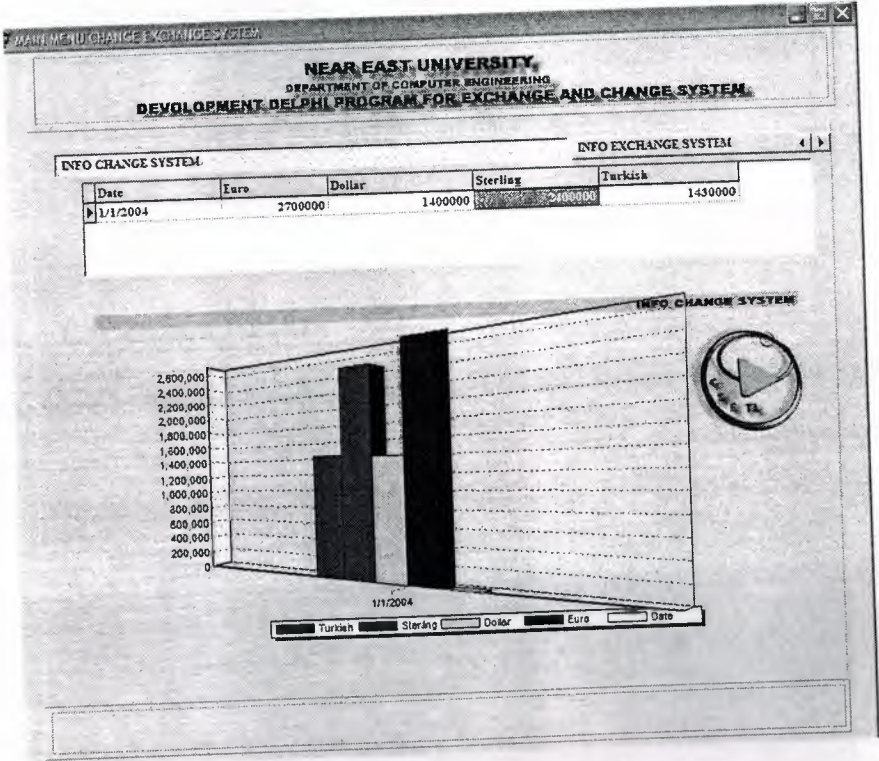


Figure 2.12 Exchange Bar Graphs a



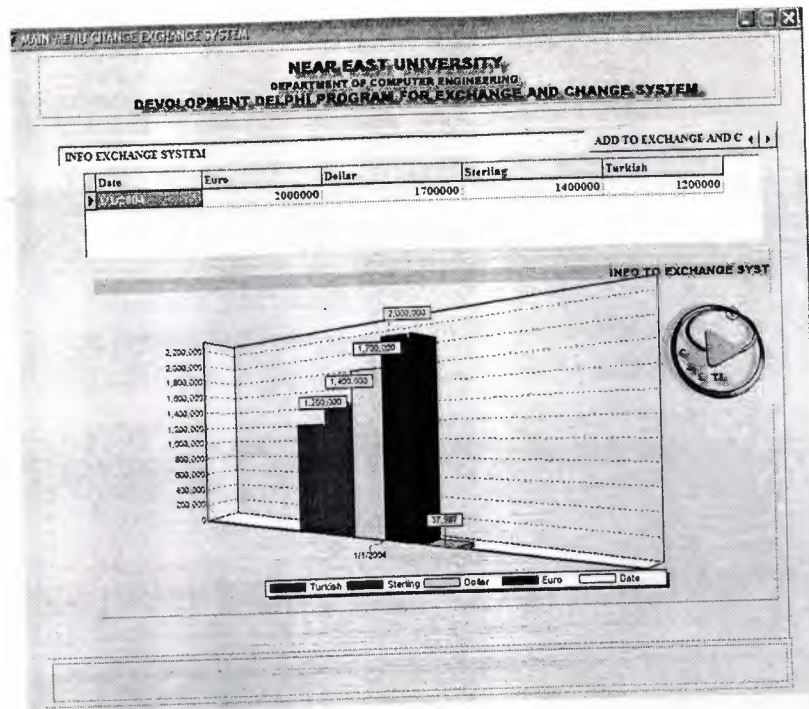


Figure 2.13 Exchange Bar Graphs b

## 2.6 Adding the Exchange Rates

a. **Selling Rates:** here in this form you can you can add the current rates of the local currency.

The screenshot displays a software interface titled "NEAR EAST UNIVERSITY DEPARTMENT OF COMPUTER ENGINEERING DEVELOPMENT DELPHI PROGRAM FOR EXCHANGE AND CHANGE SYSTEM". It features a section labeled "ADD TO EXCHANGE AND CRANGE SYSTEM" with a form for entering exchange rates. The form includes fields for "EXCHANGE SYSTEM", "Date", and a table for entering rates for Turkish Lira, Sterling, Dollar, and Euro. The date entered is 1/1/2004.

EXCHANGE SYSTEM	Date	(TL) Turkish	(£) Sterling	(\$ ) Dollar	(€) Euro
	1/1/2004	1200000	1400000	1700000	2000000

Figure 2.14 Adding the Exchange Rates (Selling Rates)

## b. Foreign Currency Rates

Here in the following form you can add the foreign exchange rates

NEAR EAST UNIVERSITY.  
DEPARTMENT OF COMPUTER ENGINEERING  
DEVELOPMENT DELPHI PROGRAM FOR EXCHANGE AND CHANGE SYSTEM

ADD TO EXCHANGE AND CHANGE SYSTEM

CHANGE SYSTEM: EXCHANGE SYSTEM

TL Turkish	(£) Sterling	(\$ ) Dollar	(€) Euro
1430000	3400000	1400000	3700000

Date: 1/1/2004

Logo: A circular logo with a stylized 'N' and 'E' and the text 'NEAR EAST UNIVERSITY' around it.

Figure 2.15 Adding the Exchange Rates (Foreign Currency Rates)



**NEAR EAST UNIVERSITY**



**Faculty of Engineering**

**Department of Computer Engineering**

**DEVELOPMENT DELPHI PROGRAM  
FOR EXCHANGE AND CHANGE SYSTEM**

**Graduation Project  
COM- 400**

**Student: Ismail Mekki**

**Supervisor: Mr. Ümit ILHAN**

**Nicosia 2003 - 2004**

## ACKNOWLEDGEMENTS



*"No one can deny the role of the university in upgrading the mentality of the student to be capable to cover the requirements of the working life. This role is concerning with qualifying the student up to the level that the society needs. However, this role cannot be accomplished unless there is a qualified leader and sophisticated coach.*

*Fortunately, Mr. Umit ILHAN was the main reason of my success in this project, and thus, he deserves my all thanks, gratitude, and my respect due to his support and wise advice.*

*I appreciate all the effort that he provided during the preparation of this project.*

*Therefore, firstly, I would like to dedicate this project my family because of their unlimited support during my life.*

*Secondly, I would like to dedicate it to my supervisor Mr. Umit ILHAN because of his wise supervision on this project and for his wide knowledge.*

*Finally, I appreciate all the effort aids of my friends during the preparation of this project."*

*With all due respect*



## ABSTRACT

This program is designed for the change and exchange markets where the individuals deal with the currencies and barter them. This system is based on BORLAND DELPHI 6 programming language. All the criterions of this system are taken according upon the request of the Near East Bank and all the features of this software can be adjusted according to the desire of the customer.

All the screens that will appear in the usage of this program will be illustrated in the coming chapters in details. The calculations and the mathematical operations that this program applies are explained in details in the appendix at the end of this report. This program is designed to find out the exact profit of the company by using graphical charts and by showing statues reports at any time. Three hard currencies are taken into consideration in this system and they are compared with the Turkish lira, which is the local currency. The values of these hard currencies will be taken from the stock markets that the government decides daily according o the daily economic level. The decision maker in NEB will determine the amount of the profit that he desires and the total revenue will be calculating by the software. A navigation bar is located in the bottom of each screen for adding, deleting, saving... etc.

## TABLE OF CONTENTS

<b>ACKNOWLEDGEMENTS</b>	<b>I</b>
<b>ABSTRACT</b>	<b>II</b>
<b>TABLE OF CONTENTS</b>	<b>III</b>
<b>LIST OF FIGURES</b>	<b>V</b>
<b>CHAPTER ONE: INTRODUCTION</b>	<b>1</b>
1.1. Overview	1
1.2. Economic Analysis of Floating Exchange Rate System	2
1.2 a Introduction	2
1.2 b Opportunities From Around World	2
1.3 Delphi Programming	5
1.4. Database Programming	7
1.5. Borland Database Engine (BDE)	8
1.6. Graphical Data-Aware Control	8
1.7. The clintDataSet Component	9
1.8 Classic BDE component	10
1.9 Tables and Queries	10
1.10 DBNavigator and Dataset Actions	11
1.11 Text-Based Data-Aware Control	12
1.12 Navigator a Dataset	12
 <b>CHAPTER TWO: THE DESCRIPTION OF THE SYSTEM</b>	 <b>14</b>
2.1 Main Menu Screen	14
2.2 Change System	18
2.3. Exchange curve	19
2.4. Exchange System	21
2.5. Exchange Bar Graphs	22



2.6. Adding the Exchange rates	23
2.7. Database Desktop	25
<b>CHAPTER THREE: DATA SOURCE</b>	26
3.1. Data Flow Diagram	26
3.2 . Main Menu	27
3.3. Buy Change System Preview	48
3.4. Exchange System Sterling	51
3.5. Change System Sterling	52
3.6. Exchange System Sterling Dollar	52
3.7. Exchange System Dollar Euro	54
3.8. Change System Dollar Euro	55
3.9. Exchange System Euro	55
3.10. Exchange System Dollar Sterling	57
3.11. Change System Dollar Sterling	58
3.12. Sell Exchange System	58
3.13. Exchange Euro to Sterling	60
3.14. Exchange Euro	61
3.15. Exchange System Dollar to Euro	61
3.16. Buy Change System preview	63
3.17. Buy Change System	66
3.18. Buy Change System	67
3.19. Change System Sterling	68
3.20. Change System	69
3.21. Report Dollar	71
3.22. Sell Change System Preview	72
<b>CONCLUSION</b>	75
<b>SYSTEM REQUIREMENTS</b>	75
<b>REFERENCES</b>	76

## LIST OF FIGURES

### CHAPTER II

#### CHANGE AND EXCHANGE SYSTEMS

2.1 Main Menu Screen	14
2.2 Buy change system preview	15
2.3 Exchange report	15
2.4 Exchange rates	16
2.5 Change System Preview	17
2.6 Exchange system Preview	17
2.7 Change System	18
2.8 Change Curve	19
2.9 Change Curve	20
2.10 Change Curve	20
2.11 Exchange System	21
2.12 Exchange Bar Graphs a	22
2.13 Exchange Bar Graphs b	23
2.14 Adding the Exchange Rates (Selling Rates)	23
2.15 Adding the Exchange Rates (Foreign Currency Rates)	24
2.16 Database Desktop	25

### CHAPTER III

#### DATA SOURCE

3.1 Data Flow Diagram	26
-----------------------	----



# INTRODUCTION

## 1.1 Overview

Many activities are done daily in the stock market. Most of these activities are concerning with the transactions and exchanging the goods and services. Since the market is the place that the buyers and sellers exchange goods and services, then the currencies are considered as a good that the people buy and sell within the market.

As it is known, the most currencies that are used widely are those currencies that are called the hard currencies such as US Dollar, UK sterling, and Euro. Therefore, as we are concerning with controlling the activities in the market by issuing the required software programs that include all features of controlling the business by means of scientific methods in order to be used in a proper and easy way.

Using any programming language to create certain software gives you the opportunity of gaining a good experience and it will promote your approaches of analyzing the aspects of any given project.

Therefore, any software needs an adequate and sufficient programming language that helps us to set up all the applications and all the functions that include the features of organizing and sustaining our business.

The role of a computer engineer requires an adequate ability of analyzing and dealing with all the aspects of any project in order to be capable to create and issue certain software that controls the business. Therefore, our role as an engineer is to not only deal with the hardware, but also deal with creating a software programs. Thus, the role of an engineer is to control the technical specifications of the firm and to know every single aspect that is necessary to measure the level of the success.

## **1.2 Economic Analysis of Floating Exchange Rate Systems**

### **1.2.a Introduction**

Associated foreign exchange is looking for experienced foreign exchange sales professionals! Successful candidates will be strongly motivated individuals who can identify and cultivate potential corporate clients with international payment needs. Strong communication skills, experience in the foreign exchange field and experience in relationship based selling are a must. Successful candidates will be responsible for opening, maintaining and growing accounts through relationship building, identifying customer needs and cross-selling appropriate services.

### **1.2 b Opportunities from Around the World**

Over the last three decades the foreign exchange market has become the world's largest financial market, with over \$1.5 trillion USD traded daily. The primary market for currencies is the 24-hour Interbank market. The Interbank market literally follows the sun around the world, moving from major banking centers of the United States to Australia and New Zealand to the Far East, to Europe and finally back to the United States. With the large minimum transaction sizes and often-stringent financial requirements, banks, hedge funds, major currency dealers and the occasional high net-worth individual speculator were the principal participants. These large traders were able to take advantage of the many benefits offered by the forex market vs. other markets including fantastic liquidity and the strong trending nature of the world's primary currency exchange rates.

The business section of any newspaper will have a table of spot exchange rates. These are the rates at which a person could have bought other currencies or foreign Exchange, such as the English Pound, French Franc, or the new European Euro. The Prices of foreign currencies can be determined in two major types of exchange rate Systems. In the United States, the dollar's exchange rates are determined by the Marketplace, i.e., by supply and demand. This type of system is called a floating Exchange rate system. In other countries, governments set the price of their currencies With respect to other countries. They then buy or sell foreign exchange at the prices They've set. This is called a fixed exchange rate system. The economic effects of these Two systems can be very different. However, in either system the underlying forces Influencing the value of a country's currency remain the same. Due to possible



confusion of being able to quote different currencies in terms of Each other, e.g., \$/£ or £/\$, we need to explicitly define an exchange rate. An **exchange Rate** is, therefore, the domestic cost of a unit of foreign exchange. For example, from the US perspective the price of the English Pound would be denominated as the number of US dollars per pound, or \$/£. As noted above, the exchange rate in a floating exchange rate system is Determined by market forces. Our definition of the exchange rate defines the market as The market for foreign exchange. In this market we have demanders and suppliers of Foreign currencies willing to pay and accept dollars in return for these currencies. We Will in turn discuss the demand and supply of foreign exchange. Foreign Exchange Demand The demand for foreign exchange is a derived demand. With the exception of currency Collectors, the demand for foreign exchange is due to people's desire to use it in the Purchase of foreign goods or financial assets. Foreign exchange demand is, therefore, Highly sensitive to changes in these desires.

In order to understand changes in the demand for foreign exchange, we will need to Discuss its underlying forces. These are the demand for foreign goods and services and the demand for foreign financial assets. The supply of foreign exchange has at its roots the same conceptual basis as Demand, only it is from the foreign perspective. Foreign currency is supplied to the Foreign exchange market when foreigners exchange their currency for dollars in order to Buy US goods or financial assets. Equivalently, the supply of foreign exchange is Nothing more than a mirror image of the foreign demand for US currency. Exchange is the mirror of the supply of dollars to the foreign exchange market. One question which might arise is which foreign exchange Market. New York, London, Frankfurt and Tokyo are Major financial centers with large foreign exchange markets. The answer as to which market is all of them. The first rule of business is to buy low and sell high. Should exchange rates be different across different financial centers, then the opportunity for arbitrage profits occurs. Currency dealers will buy low in one center and sell high in another, driving exchange rates into equality Across the different markets. For example, should the Swiss Franc be at a lower price (in terms of \$) in London and at a higher price in New York, then the dealers will increase the demand for the Swiss Franc in London, driving up its price, and increase its supply in New York, driving down its price there. This continues until the price is the same in both places. The major questions to be addressed are how exchange rates determined are and what the forces which influence them are. In Figure 1, the equilibrium exchange rate (e) is the one where the quantity demanded is equal to the quantity supplied for

foreign exchange. As with most markets, the price changes in order to equilibrate the market. When quantity demanded exceeds quantity supplied, and then the exchange rate will rise. If the quantity supplied is greater than quantity demanded, the exchange rate falls. What does it mean when the exchange rate rises or falls? As we have defined the Exchange rate (\$/£), when the exchange rate rises, the value of the dollar decreases or depreciates. It now takes more dollars to buy an English pound than it did before the Change in the exchange rate. Fewer foreign goods can now be purchased for a given number of dollars. The reverse is also true. As the exchange rate falls, the dollar cost of foreign exchange falls, increasing the dollar's value. This is termed an **appreciation** of the dollar. More foreign exchange rate \$/£ Foreign exchange Sfx or D\$ Dfx or S\$. Market should force lead to a change in either the supply or demand for foreign exchange then the exchange will change accordingly to re-equilibrate the market. The basic notion is that exchange rates are sensitive to differential inflation rates across Countries. Should the domestic inflation rate rise at a rate greater than our trading Partners, then at a given exchange rate, the price of domestic goods will be rising relative To foreign goods. This will, in turn, increase the demand for foreign goods (imports are Now cheaper in domestic currency terms) and decrease the demand for domestic exports (Domestic exports are now more expensive in foreign currency terms). This results in an Increase in the demand for foreign exchange, as well as a decrease in the supply of Foreign exchange.

This is a long-run effect because of the Law of One Price. This concept states that in the Long run the price of tradable goods must be the same across countries. If this was not The case, then the opportunity for arbitrage profits, buying low in one country and selling High in another, would result in a movement in the exchange rate bringing about the Equalization. For example, suppose Argentine wheat, at the prevailing exchange rate, is cheap in US Dollars. As North Americans buy more and more Argentine wheat, they increase the Demand for the Argentine currency, driving up its value, thus making wheat more Expensive in dollar terms. The exchange rates which would prevail under the Law of One Price are called purchasing power parity exchange rates (PPP). While these do not exist in reality (there are many other factors affecting exchange rates) there is an underlying pressure moving exchange rates in this fashion. PPP exchange rates are used in comparing the economic performance between countries. The World Bank compares countries in their *World Development Report* using a PPP exchange rate. Medium Term - Differential Growth Rates As an economy



grows, its demand for imports will also grow. As income increases, some portion of that increase will be spent on imported goods. In the jargon of macroeconomics, the proportion of the additional dollar of income spent on imports is called the marginal propensity to import. Assume that the marginal propensity to import is the same across countries. Should a country's economy grow faster than its trading partners, then its demand for imports will also be growing faster? In the context of Figure 4, this is represented by increases in both the demand and supply of foreign exchange, but the demand would increase by more. This would result in a slight depreciation of the domestic currency. Short-Run - Differential Interest Rates This factor has become extremely important as countries have liberalized their economies, allowing the flow of financial capital into and out of their countries. It has played an important role in the East Asian and Mexican Peso financial crises. Exchange rate \$/£ Foreign exchange.

### 1.3 Delphi Programming

Delphi 5 provided new features to the Object Inspector, and Delphi 6 includes even more additions to it. As this is a tool programmer's use all the time, along with the editor and the Form Designer, its improvements are really significant.

The most important change in Delphi 6 is the ability of the Object Inspector to expand component references in-place. Properties referring to other components are now displayed in a different color and can be expanded by selecting the + symbol on the left, as it happens with internal subcomponents. You can then modify the properties of that other component without having to select it.

**NOTE** This interface-expansion feature also supports subcomponents, as demonstrated by the new Labeled Edit control. The Form Designer

**TIP** A related feature of the Object Inspector is the ability to select the component referenced by a property. To do this, double-click the property value with the left mouse button while keeping the Ctrl key pressed. For example, if you have a Main Menu component in a form and you are looking at the properties of the form in the Object Inspector, you can select the Main Menu component by moving to the Main Menu property of the form and Ctrl+double-clicking the value of this property. This selects the main menu indicated as the value of the property in

the Object Inspector. Here are some other relevant changes of the Object Inspector: The list at the top of the Object Inspector shows the type of the object and

can be removed to save some space (and considering the presence of the Object Tree View). The properties that reference an object are now a different color and may be expanded without changing the selection. You can optionally also view read-only properties in the Object Inspector. Of course, they are grayed out. The Object Inspector has a new Properties dialog box which allows you to customize the colors of the various types of properties and the overall behavior of this window.

The Project Manager doesn't provide a way to set the options of two different projects at one time. What you can do instead is invoke the Project Options dialog from the Project Manager for each project. The first page of Project Options (Forms) lists the forms that should be created automatically at program startup and the forms that are created manually by the program.

The next page (Application) is used to set the name of the application and the name of its Help file, and to choose its icon. Other Project Options choices relate to the Delphi compiler and linker, version information, and the use of run-time packages.

There are two ways to set compiler options. One is to use the Compiler page of the Project Options dialog. The other is to set or remove individual options in the source code with the `{SX+}` or `{SX-}` commands, where you'd replace *X* with the option you want to set. This second approach is more flexible, since it allows you to change an option only for a specific source-code file, or even for just a few lines of code. The source-level options override the compile-level options.

All project options are saved automatically with the project, but in a separate file with a .DOF extension. This is a text file you can easily edit. You should not delete this file if you have changed any of the default options. Delphi also saves the compiler options in another format in a CFG file, for command-line compilation. The two files have similar content but a different format: The *dcc* command-line compiler, in fact, cannot use .DOF files, but needs the .CFG format. Another alternative for saving compiler options is to press Ctrl+O+O (press the O key twice while keeping Ctrl pressed). This inserts, at the top of the current unit, compiler directives that correspond to the current project options, as in the following listing: `{SA+,B-,C+,D+,E-,F-,G+,H+,I+,J+,K-,L+,M-,N+,O+,P+,Q-,R-,S-,T-,U-,V+,W-,X+,Y+,Zl}`

Memory management in Delphi is subject to three rules: Every object must be created before it can be used; every object must be destroyed after it has been used; and every object must be destroyed only once. Whether you have to do these operations in



your code, or you can let Delphi handle memory management for you, depends on the model you choose among the different approaches provided by Delphi.

Delphi supports three types of memory management for dynamic elements (that is, elements not in the stack and the global memory area):

- Every time you create an object explicitly, in the code of your application, you should also free it. If you fail to do so, the memory used by that object won't be released for other objects until the program terminates.

- When you create a component, you can specify an owner component, passing the owner to the component constructor. The owner component (often a form) becomes responsible for destroying all the objects it owns. In other words, when you free the form, it frees all the components it owns. So, if you create a component and give it an owner, you don't have to remember to destroy it. This is the standard behavior of the components you create at design time by placing them on a form or data module.

- When you allocate memory for strings, dynamic arrays, and objects referenced by interface variables, Delphi automatically frees the memory when the reference goes out of scope. You don't need to free a string: when it becomes unreachable, its memory is released.

## **1.4 Database Programming**

Delphi's support for database applications is one of the key features of the programming environment. Many programmers spend most of their time writing data-access code, which needs to be the most robust portion of a database application. This chapter provides an overview of Delphi's extensive support for database programming. What you will find here is a discussion of the theory of database design. I am assuming that you already know the fundamentals of database design and have already designed the structure of a database. I will not look into database-specific problems; my goal is to help you understand how Delphi supports database access. I will begin with an explanation of the alternatives Delphi offers in terms of data access, and then I will provide an overview of the database components that I have used in my program. This chapter includes an overview of the TDataSet class, an in-depth analysis of the TField components, and the use of data-aware controls. The following chapters will provide information on more advanced database programming topics, such as client/server programming, the use of dbGo, dbExpress, and InterBase Express

## 1.5 Borland Database Engine (BDE)

The BDE originated with Paradox, well before Delphi existed, and was extended by Borland to support other local databases and many SQL servers. The BDE has direct access to dBASE, Paradox, ASCII, FoxPro, and Access tables. A series of drivers (called SQL Links and available only in Delphi Enterprise) allows access to some SQL servers, including Oracle, Sybase, Microsoft, Informix, InterBase, and DB2 servers. If you need access to a different database, the BDE can also interface with ODBC drivers.

## 1.6 Graphical Data-Aware Controls

Finally, Delphi includes two graphical data-aware controls:

- **DBImage**, which is an extension of an Image component that shows a picture stored in a BLOB field (provided the database use a graphic format that the Image component supports, such as BMP and JPEG). The output of the Cust- Lookup example, with the BLookupComboBox showing multiple fields in its drop-down list.

- **DBChart** is a data-aware business graphic component or the data-aware version of the TeeChart control built by David Berneda. To demonstrate the use of the DBChart control, I have added this component to a simple example showing a data grid. The application, called ChartDB, shows a pie chart with the surface of each country of the COUNTRY.DB table. The program has almost no code, as all the settings can be done using the specific component editor, which has several options but is quite easy to use. Here are some of the key properties of the component, taken from the form description:

**object** DBChart1: TDBChart

Legend.Visible = False

Align = alClient

**object** Series1: TPieSeries

Marks.ArrowLength = 8

Marks.Visible = True

DataSource = Table1

XLabelsSource = 'Name'

ExplodeBiggest = 3

OtherSlice.Style = poBelowPercent

OtherSlice.Text = 'Others'



```

OtherSlice.Value = 2
PieValues.ValueSource = 'Area'
end;
end.

```

What I have done is show the area field as the data source for the pie chart (the PieValues ValueSource property of the series), use the name field for the labels (the XLabelsSource property of the series), and condense all the countries with a value below 2 percent in a single section indicated as Others (the OtherSlide subproperties). As a minor addition to the code, I have added two radio buttons you can use to toggle between the area and the population. The code of the two radio buttons simply sets the source of the series, after casting it to the proper series type, as in:

```

procedure TForm1.RadioPopulationClick(Sender: TObject);
begin
  DBChart1.Title.Text [0] := 'Population of Countries';
  (DBChart1.Series [0] as TPieSeries).PieValues.ValueSource := 'Population';
end;

```

## 1.7 The ClientDataSet Component

Finally, there is a component derived from TDataSet that has a peculiar behavior and can be combined with other data-access components. The ClientDataSet component, in fact, is a dataset accessing data kept in memory. The in-memory data can be totally temporary (lost as you exit the program), saved to a local file as a snapshot, and imported by another dataset using a Provider component. This last situation is certainly the most common: You can hook a ClientDataSet to any other local dataset, or use Borland's multitier support (discussed in Accessing a Database: BDE, dbExpress, and other alternatives "Multitier Database Applications with DataSnap") to retrieve data from a dataset hosted by a different application, possibly running on a separate computer. The ClientDataSet component becomes particularly useful if the data-access components you are using provide limited or no caching. This is particularly true of the new dbExpress engine, but can equally help you when using the BDE or other native components.

On the other hand, ADO already provides most of the services of the ClientDataSet component and using these two at the same time can be useful only in limited situations

## 1.8 Classic BDE Components

Each of the database-access solutions discussed above has its own set of data-access, database connection, and extra utility components on a specific page of the Component palette. The classic BDE components have been moved to the new BDE page and include the Table, Query, and StoredProc components. The ADO, dbExpress, and InterBase Express components are each in specific pages, and all include specific dataset components and others that tend to mimic the BDE components, simplifying the porting of existing applications.

The Data Access page of the Component palette includes only the Data Source Component and others not specifically related with any single data access technology. Besides the data-access component of your choice, a Delphi visual application generally uses some data-aware controls (in the Data Controls page) and the DataSource component. Data-aware controls are visual components used to view and edit the data in a form and are extensions of standard components such as edit and list boxes, radio buttons, images, and the

Grid. The DataSource component has the role of connector between the data-aware controls and a dataset component.

## 1.9 Tables and Queries

The simplest traditional way to specify data access in Delphi was to use the BDE Table component. A Table object simply refers to a database table. When you use a Table component, you need to indicate the name of the database you want to use in its DatabaseName property. You can enter an alias or the path of the directory with the table files. The Object Inspector lists the available names, which depend on the aliases installed in the BDE. You also need to indicate a proper value in the TableName property. The Object Inspector lists the available tables of the current database (or directory), so you should generally select the DatabaseName property first. Another classic dataset is the BDE Query component. A query requires a SQL language



command. You can customize a query using SQL more easily than you can customize a table (as long as you know at least the basic elements of SQL, of course). The Query component has a `DatabaseName` property like the Table component, but it does not have a `TableName` property. The table is indicated in the SQL statement, stored in the SQL property. For example, you can write a simple SQL statement like this:

**Select \* from** Country where Country is the name of a table and the asterisk (\*) indicates that you want to use all of the fields in the table.

The efficiency of a table or a query varies depending on the database you are using. In general, we can say that the Table component tends to be faster on local tables, while the Query component tends to be faster on SQL servers, although this is just a very general rule, and in many cases you might have the opposite effect. We'll see some efficiency issues while discussing

client/server development in the third BDE dataset component is `StoredProc`, which refers to stored procedures of a SQL server database. You can run these procedures and get the results in the form of a database table. Stored procedures can only be used with SQL servers.

## 1.10 DBNavigator and Dataset Actions

DBNavigator is a collection of buttons used to navigate and perform actions on the database. You can disable some of the buttons of the DBNavigator control, by removing some of the elements of the `VisibleButtons` set. The buttons perform basic actions on the connected dataset, so you can easily replace them

With your own toolbar, particularly if you use an `ActionList` component with the predefined database actions provided by Delphi. In this case, in fact, you get all the standard behaviors, but you'll also see the various buttons enabled only when their action is legitimate. **TIP** If you use the standard actions, you can avoid connecting them to a specific `DataSource` component, and the actions will be applied to the dataset connected to the visual control that currently has the input focus. This way a single toolbar can be used for multiple datasets displayed by a form.

## 1.11 Text-Based Data-Aware Controls

There are multiple text-oriented components:

DBText displays the contents of a field that cannot be modified by the user. It is a data aware Label graphical control. It can be very useful, but users might confuse this control with the plain labels that indicate the content of each field-based control. DBEdit lets the user edit a field (change the current value) using an Edit control. At times, you might want to disable editing and use a DBEdit as if it were a DBText, but highlighting the fact that this is data coming from the database. DBMemo lets the user see and modify a large text field, eventually stored in a memo or BLOB (binary large object) field. It resembles the Memo component and has full editing capabilities, but all the text is rendered in a single font.

DBRichEdit is a component that lets the user edit a formatted text file; it is based on a RichEdit Windows common control and, in contrast to DBMemo, it allows text with multiple fonts and paragraph styles.

## 1.12 Navigating a Dataset

We've seen that a dataset has only one active record, and you can imagine that the active record changes often, in response of user actions or because of internal commands given to the dataset. To move around the dataset and change the active record, there are methods of the TDataSet class, particularly in the section commented as "position, movement." You can move to the next or previous record, jump back and forth by

A given number of records (with MoveBy), or go directly to the first or last record of the dataset. These operations of the dataset are generally available in the DBNavigator component or in the standard dataset actions, and they are not particularly complex to understand. What is not obvious, though, is how a dataset handles the extreme positions. If you open any dataset with a navigator attached, you can see that as you move on record by record, the Next button remains enabled even when you've reached the last record. It's only when you try to move forward after the last record that the current record apparently doesn't change and the button is disabled. This is because the Eof test (end of file) succeeds only when the cursor has been moved to a special position after the last record. If you jump to the end with the Last button, instead, you'll immediately be at the very end. You'll see exactly the same behavior for the first record



(and the Bof test). As we'll see in a while, this approach is very handy, as we can scan a dataset testing for Eof to be True and, at this point, we know we've also already processed the last record of the dataset.

**NOTE** Handling this special record positions before the beginning and after the end of the dataset, which are called *cracks*, is very important (and quite confusing) when you write a custom dataset. Besides moving around record by record or by a given number of records, programs might need to jump to specific records or positions. Some datasets support the RecordCount property and allow movement to a record at a given position in the dataset using the RecNo property. These properties can be used only for datasets that support positions natively, which basically excludes all client/server architectures, unless you grab all of the records in a local cache (something you'll generally want to avoid) and then navigate on the cache. As we'll see in the next chapter, when you open a query on a SQL server you fetch only the records you are using, so Delphi doesn't know the record count, at least not in advance. There are two alternatives you can use to refer to a record in a dataset, regardless of its type. You can save a reference to the current record and then jump back to it after moving around. This is accomplished by using bookmarks, either in the TBookmark or the more modern TBookmarkStr form. You can locate a record of the dataset matching given criteria, using the Locate method. This even works after you close and reopen the dataset, because you're working at a logical (and not physical) level. This approach is presented in the next section.

## 2.1 Main Menu Screen

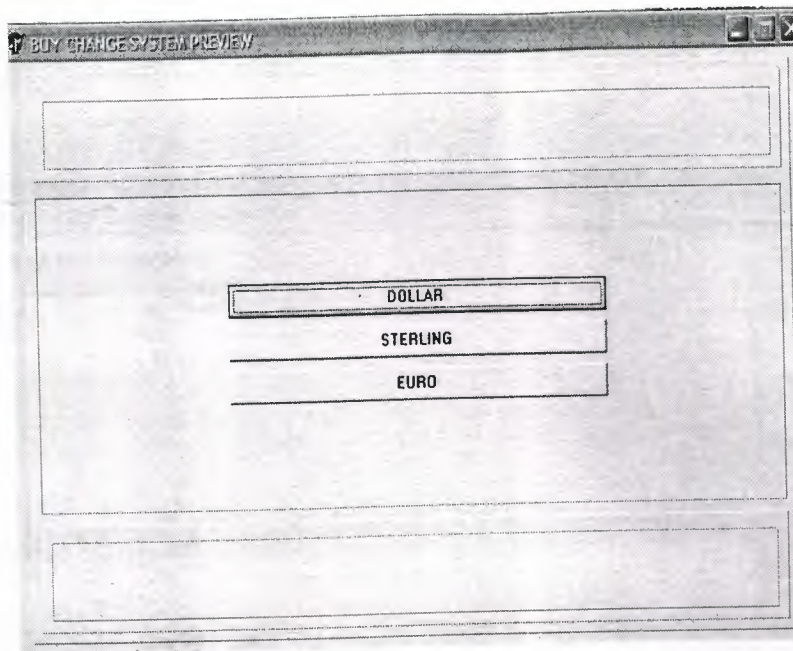


Figure 2.1 Main Menu Screen

The figure above shows the first screen which occurs when you first start running the program. This screen contains the following:

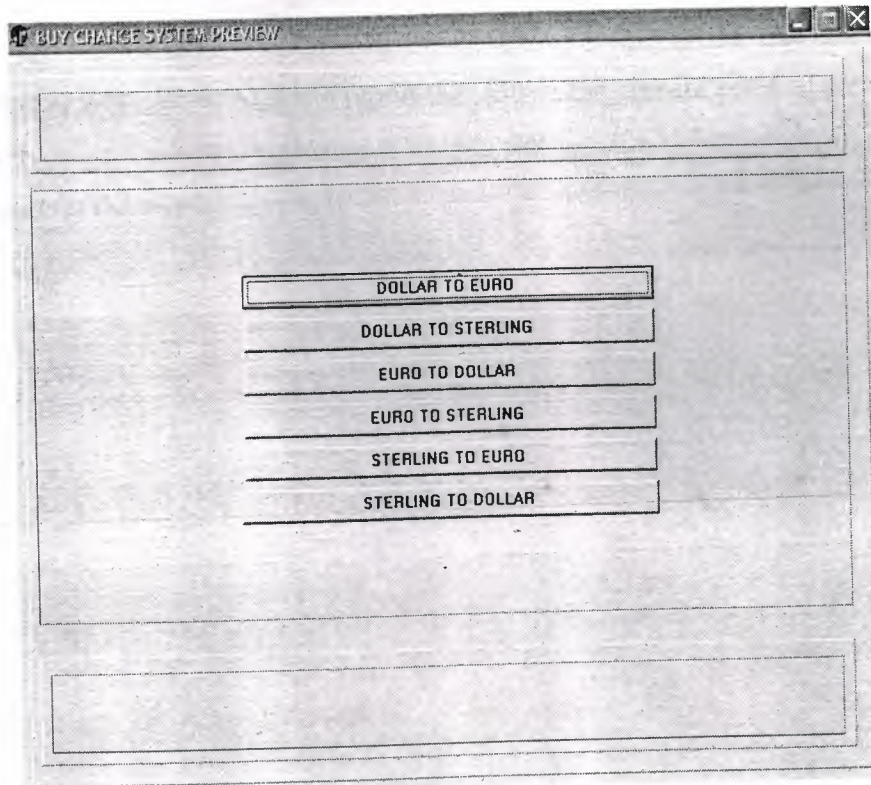
- ◆ The calendar: This automatically obtains the current date month and year of the system which the program is running on.
- ◆ Buy change system preview: which shows the current selling price of currencies that you have entered in the change system, which will be explained later.





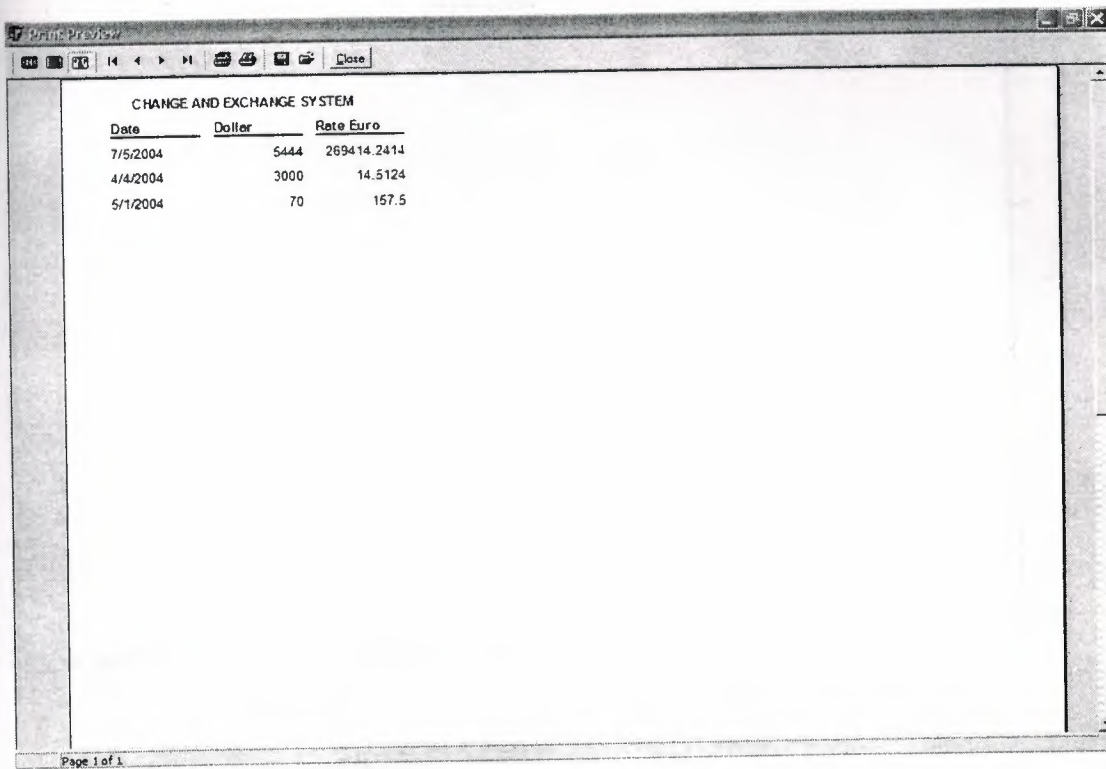
**Figure 2.2** Buy change system preview

- ◆ Sell exchange system preview: which shows the current reports of the exchange system. The following screen will appear and allow you to Select the type of exchange report that you wish to preview.



**Figure 2.3** Exchange report

The following report preview will show the exchange rates related to which form you choose:



Date	Dollar	Rate Euro
7/5/2004	5444	269414.2414
4/4/2004	3000	14.5124
5/1/2004	70	157.5

Figure 2.4 Exchange rates

- ◆ Change system preview: it is a report that shows the current price of the currencies in the stuck exchange related to selling currencies, which can be printed by the button next to it.



Print Preview

CHANGE AND EXCHANGE SYSTEM

Turkish	Sterling	Dollar	Euro	Date
1430000	2400000	1400000	2700000	1/1/2004

Page 1 of 1

Figure 2.5 Change System Preview

- ◆ Exchange system: it is a report that shows the current price of the currencies in the stuck exchange related to buying currencies, which can be printed by the button next to it.

Print Preview

CHANGE AND EXCHANGE SYSTEM

Turkish	Sterling	Dollar	Euro	Date
1200000	1400000	1700000	2000000	1/1/2004

Page 1 of 1

Figure 2.6 Exchange System Preview

## 2.2 Change System

MAIN MENU CHANGE EXCHANGE SYSTEM

NEAR EAST UNIVERSITY  
DEPARTMENT OF COMPUTER ENGINEERING  
DEVELOPMENT DELPHI PROGRAM FOR EXCHANGE AND CHANGE SYSTEM

CHANGE SYSTEM

TURKISH TO DOLLAR | TURKISH TO STERLING | **TURKISH TO EURO**

TL Turkish	£ Sterling	\$ Dollar	€ Euro
1430000	2400000	1400000	2700000

Date  
1/1/2004

REPORT CHANGE SYSTEM

ADD TO CHANGE SYSTEM:

Date  
7/5/2004

EXCHANGE RATE	EXCHANGE AMOUNT	EXCHANGE VALUE
27000000	100	270000000

Figure 2.7 Change System

This screen consists of three sub-menus, which will allow you to complete your transaction processes. In addition, the forms contain the following:

The first part of each form grapes the exchange rates of the currencies from your database that you have entered.

The second part deals with the current amount of currency that you are exchanging at the time being. It consists of a navigation bar that allows you to scroll in the fields of your transactions.



## 2.3 Exchange Curves

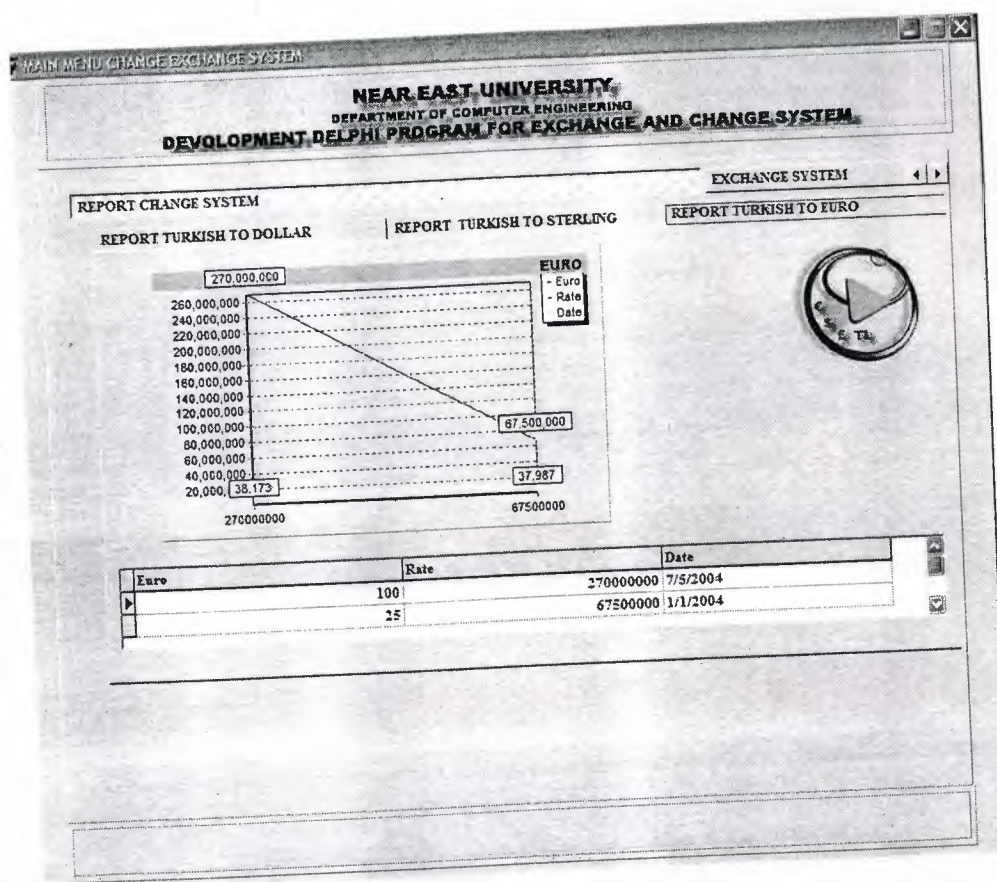


Figure 2.8 Change Curve

It consists of three chart reports that are all linked to the data base and change in respect to the processes that you make, that each transaction made is added or deleted and presented in the as a curve in respect to the money amount you have processed. The rest of the graphs are shown below

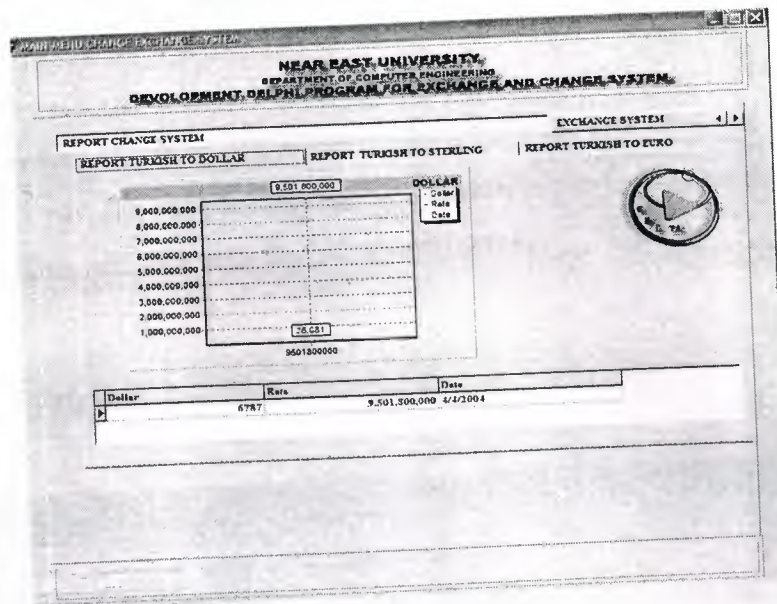


Figure 2.9 Change Curve

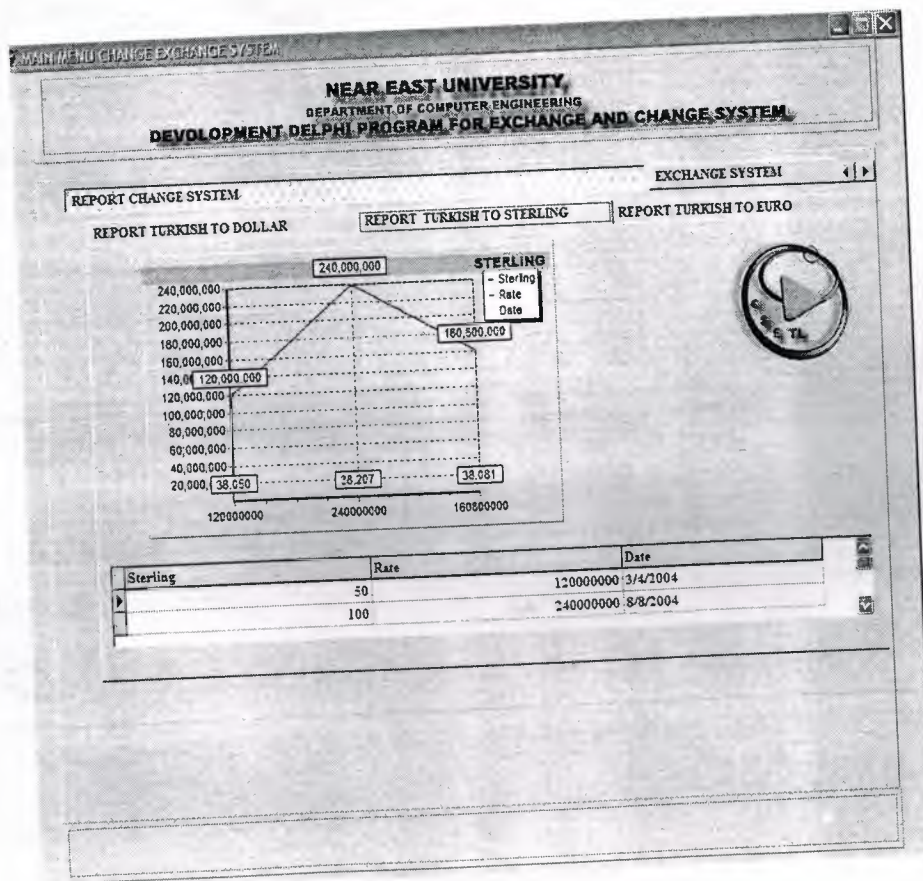


Figure 2.10 Change Curve



These graphs are just to keep track of you exchange processes in terms of money, and also shows you which of the currencies is have the most demand in the market, at this point you can analyze the current situation of the foreign currencies flow in you exchange office and take your decisions among that.

## 2.4 Exchange System

MAIN MENU CHANGE EXCHANGE SYSTEM

**NEAR EAST UNIVERSITY**  
DEPARTMENT OF COMPUTER ENGINEERING  
DEVELOPMENT DELPHI PROGRAM FOR EXCHANGE AND CHANGE SYSTEM

EXCHANGE SYSTEM INFO CHANGE SYSTEM

EURO TO DOLLAR

CHANGE	[TL] Turkish	[£] Sterling	[\$] Dollar	[€] Euro	Date
	1430000	2400000	1400000	2700000	1/1/2004

EXCHANGE

EXCHANGE	[TL] Turkish	[£] Sterling	[\$] Dollar	[€] Euro	Date
	1200000	1400000	1700000	2000000	1/1/2004

EXCHANGE

EXCHANGE	[\$] Dollars given amount	[\$] DOLLAR PRICE	[€] woned ratio	equal amount	Date
	54345			40081.7156	7/8/2004




Figure 2.11 Exchange System

The exchange system consists of three fields that allow you to deal with the exchange of the foreign currencies each one consist of selling and buying prices of the currencies and another field for your processes, where you can add delete or modify your processes.

## 2.5 Exchange Bar Graphs

The following bar graphs illustrate the change in the currencies by date and respectively with the database.

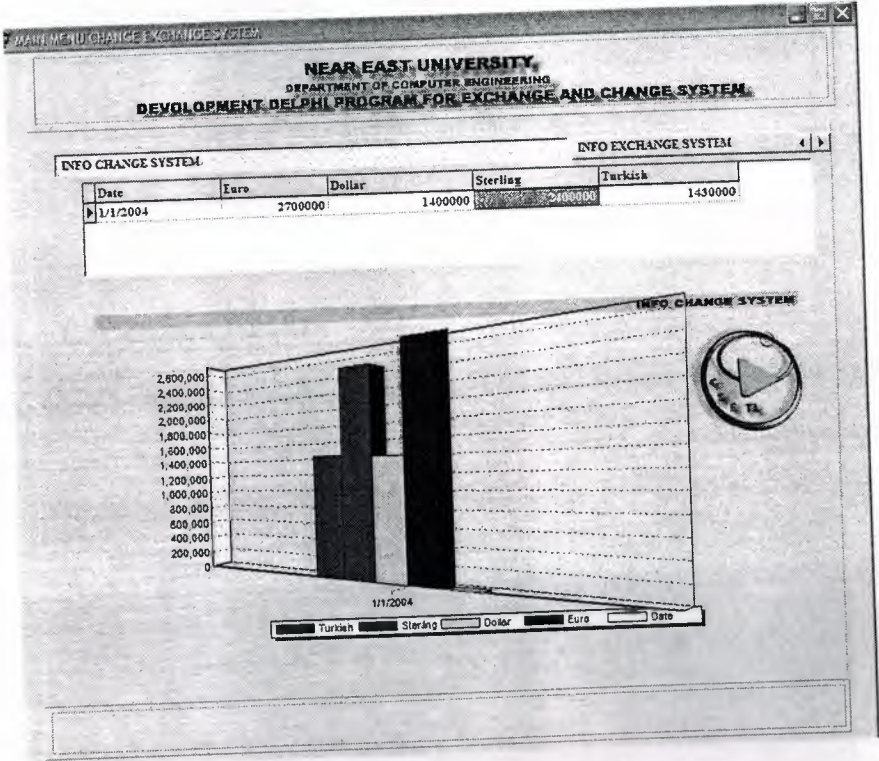


Figure 2.12 Exchange Bar Graphs a



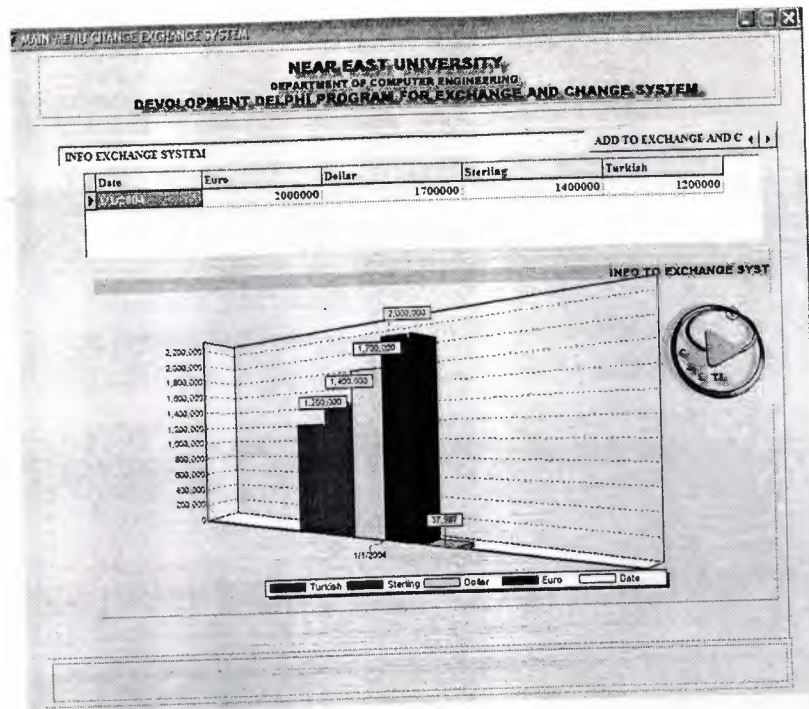


Figure 2.13 Exchange Bar Graphs b

## 2.6 Adding the Exchange Rates

a. **Selling Rates:** here in this form you can add the current rates of the local currency.

The screenshot shows the "ADD TO EXCHANGE AND CHANGE SYSTEM" form. It includes a section for "EXCHANGE SYSTEM" with a table of exchange rates for the date 1/1/2004. The table has columns for Turkish (TL), Sterling (£), Dollar (\$), and Euro (€). Below the table is a date field set to 1/1/2004. A legend at the bottom identifies the currencies: Turkish (light blue), Sterling (dark blue), Dollar (light green), Euro (dark green), and Date (white).

TL Turkish	£ Sterling	\$ Dollar	€ Euro
1200000	1400000	1700000	2000000

Figure 2.14 Adding the Exchange Rates (Selling Rates)

## b. Foreign Currency Rates

Here in the following form you can add the foreign exchange rates

NEAR EAST UNIVERSITY.  
DEPARTMENT OF COMPUTER ENGINEERING  
DEVELOPMENT DELPHI PROGRAM FOR EXCHANGE AND CHANGE SYSTEM

ADD TO EXCHANGE AND CHANGE SYSTEM

CHANGE SYSTEM: EXCHANGE SYSTEM

TL Turkish	(£) Sterling	(\$) Dollar	(€) Euro
1430000	3400000	1400000	3700000

Date: 1/1/2004

Logo: A circular logo with a stylized 'N' and 'E' and the text 'NEAR EAST UNIVERSITY' around it.

Figure 2.15 Adding the Exchange Rates (Foreign Currency Rates)



## 2.7 Database Desktop

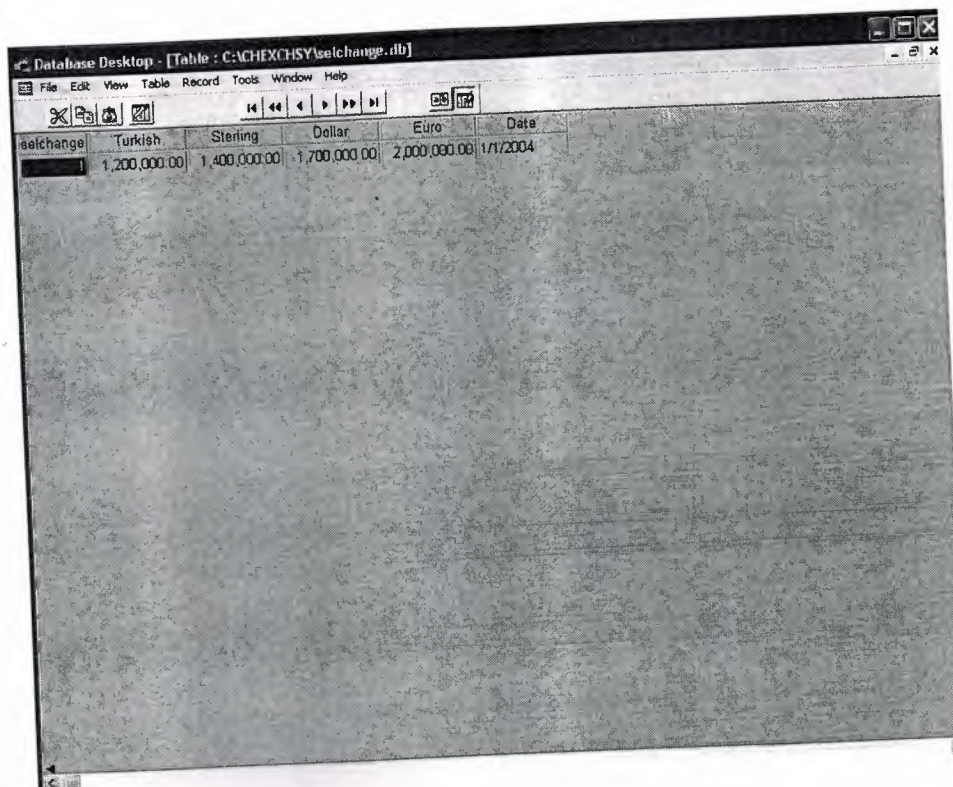


Figure 2.16 Database Desktop

Database Desktop is a database tool where you can create or restructure database tables, or browse and edit their data. You can work with tables in Paradox, dBase, and SQL formats.

## DATA FLOW DIAGRAM (DFD)

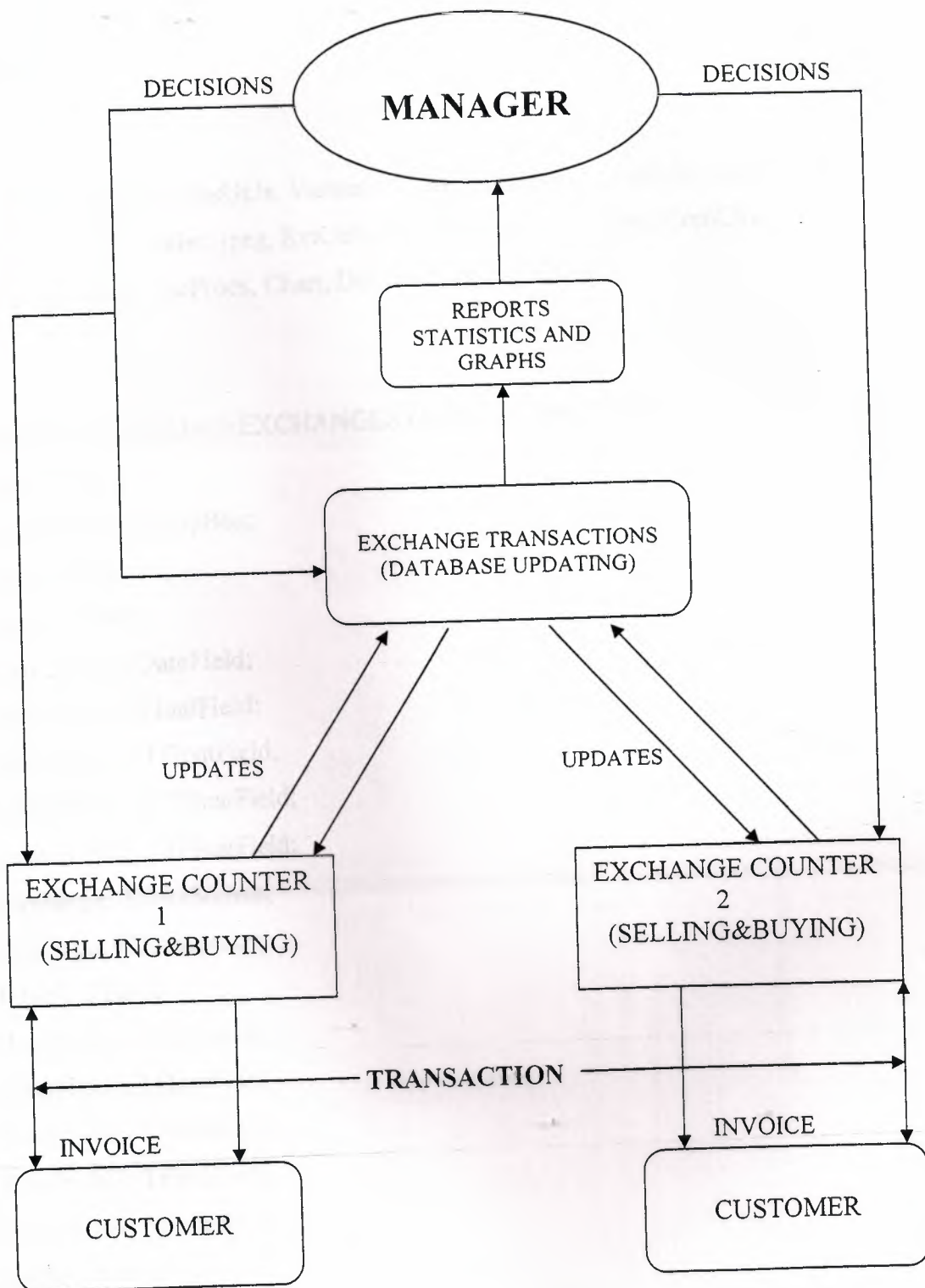


Figure 3.1 Data Flow Diagram



## 3.2 MAIN MENU

unit CHEXSU;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, jpeg, ExtCtrls, DBCtrls, StdCtrls, Mask, ComCtrls,  
TeEngine, Series, TeeProcs, Chart, DbChart, Grids, DBGrids;

type

TMAINMENUCHANGEEXCHANGESYSTEM = class(TForm)

Panel1: TPanel;

GroupBox1: TGroupBox;

Image1: TImage;

Table1: TTable;

Table1Date: TDateField;

Table1Euro: TFloatField;

Table1Dollar: TFloatField;

Table1Sterling: TFloatField;

Table1Turkish: TFloatField;

buychange: TDataSource;

selchange: TDataSource;

Table2: TTable;

DateField1: TDateField;

FloatField1: TFloatField;

FloatField2: TFloatField;

FloatField3: TFloatField;

FloatField4: TFloatField;

dollar: TDataSource;

Table3: TTable;

Table3Dollar: TFloatField;

Table3Rate: TFloatField;

Table3Date: TDateField;  
sterlaing: TDataSource;  
Table4: TTable;  
euro: TDataSource;  
Table5: TTable;  
Table9: TTable;  
exdollarE: TDataSource;  
Table10: TTable;  
exeuroD: TDataSource;  
Table11: TTable;  
Table12: TTable;  
exeuroS: TDataSource;  
exdollarS: TDataSource;  
Table14: TTable;  
exsterlingD: TDataSource;  
exsterlingE: TDataSource;  
Table15: TTable;  
Table17: TTable;  
Table18: TTable;  
Table6: TTable;  
Table7: TTable;  
Table8: TTable;  
Panel2: TPanel;  
GroupBox32: TGroupBox;  
Panel3: TPanel;  
PageControl1: TPageControl;  
TabSheet1: TTabSheet;  
Image2: TImage;  
TabControl1: TTabControl;  
Panel6: TPanel;  
MonthCalendar1: TMonthCalendar;  
Button1: TButton;  
Button3: TButton;  
Button4: TButton;



Button5: TButton;  
TabSheet2: TTabSheet;  
PageControl3: TPageControl;  
TabSheet11: TTabSheet;  
GroupBox6: TGroupBox;  
Label16: TLabel;  
Label17: TLabel;  
Label18: TLabel;  
Label19: TLabel;  
Label20: TLabel;  
DBEdit12: TDBEdit;  
DBEdit13: TDBEdit;  
DBEdit14: TDBEdit;  
DBEdit15: TDBEdit;  
DBEdit16: TDBEdit;  
DBNavigator2: TDBNavigator;  
GroupBox8: TGroupBox;  
Label26: TLabel;  
GroupBox9: TGroupBox;  
Label27: TLabel;  
Label28: TLabel;  
Label29: TLabel;  
Label30: TLabel;  
DBEdit25: TDBEdit;  
DBEdit22: TDBEdit;  
DBEdit23: TDBEdit;  
DBEdit24: TDBEdit;  
DBNavigator5: TDBNavigator;  
TabSheet12: TTabSheet;  
GroupBox5: TGroupBox;  
Label11: TLabel;  
Label12: TLabel;  
Label13: TLabel;  
Label14: TLabel;

Label15: TLabel;  
DBEdit7: TDBEdit;  
DBEdit8: TDBEdit;  
DBEdit9: TDBEdit;  
DBEdit10: TDBEdit;  
DBEdit11: TDBEdit;  
DBNavigator4: TDBNavigator;  
GroupBox10: TGroupBox;  
Label31: TLabel;  
GroupBox11: TGroupBox;  
Label32: TLabel;  
Label33: TLabel;  
Label34: TLabel;  
Label35: TLabel;  
DBEdit26: TDBEdit;  
DBEdit28: TDBEdit;  
DBEdit29: TDBEdit;  
DBEdit27: TDBEdit;  
DBNavigator6: TDBNavigator;  
TabSheet13: TTabSheet;  
GroupBox7: TGroupBox;  
Label21: TLabel;  
Label22: TLabel;  
Label23: TLabel;  
Label24: TLabel;  
Label25: TLabel;  
DBEdit17: TDBEdit;  
DBEdit18: TDBEdit;  
DBEdit19: TDBEdit;  
DBEdit20: TDBEdit;  
DBEdit21: TDBEdit;  
DBNavigator3: TDBNavigator;  
GroupBox12: TGroupBox;  
Label36: TLabel;



GroupBox13: TGroupBox;  
Label37: TLabel;  
Label38: TLabel;  
Label39: TLabel;  
Label40: TLabel;  
DBEdit30: TDBEdit;  
DBEdit31: TDBEdit;  
DBEdit32: TDBEdit;  
DBEdit33: TDBEdit;  
DBNavigator7: TDBNavigator;  
TabSheet3: TTabSheet;  
PageControl4: TPageControl;  
TabSheet14: TTabSheet;  
DBGrid1: TDBGrid;  
DBChart1: TDBChart;  
Series1: TLineSeries;  
Series2: TLineSeries;  
Series3: TLineSeries;  
TabSheet15: TTabSheet;  
DBGrid2: TDBGrid;  
DBChart2: TDBChart;  
LineSeries1: TLineSeries;  
LineSeries2: TLineSeries;  
LineSeries3: TLineSeries;  
TabSheet16: TTabSheet;  
DBGrid3: TDBGrid;  
DBChart3: TDBChart;  
LineSeries4: TLineSeries;  
LineSeries5: TLineSeries;  
LineSeries6: TLineSeries;  
TabSheet4: TTabSheet;  
Label71: TLabel;  
PageControl5: TPageControl;  
TabSheet17: TTabSheet;

GroupBox14: TGroupBox;  
Label41: TLabel;  
Label42: TLabel;  
Label43: TLabel;  
Label44: TLabel;  
Label45: TLabel;  
Label77: TLabel;  
DBEdit34: TDBEdit;  
DBEdit35: TDBEdit;  
DBEdit36: TDBEdit;  
DBEdit37: TDBEdit;  
DBEdit38: TDBEdit;  
DBNavigator8: TDBNavigator;  
GroupBox20: TGroupBox;  
Label78: TLabel;  
Label79: TLabel;  
Label80: TLabel;  
Label81: TLabel;  
Label82: TLabel;  
Label113: TLabel;  
DBEdit64: TDBEdit;  
DBEdit65: TDBEdit;  
DBEdit66: TDBEdit;  
DBEdit67: TDBEdit;  
DBEdit68: TDBEdit;  
DBNavigator14: TDBNavigator;  
GroupBox27: TGroupBox;  
Label120: TLabel;  
Label121: TLabel;  
Label122: TLabel;  
Label123: TLabel;  
Label124: TLabel;  
Label125: TLabel;  
DBEdit97: TDBEdit;



DBEdit98: TDBEdit;  
DBEdit99: TDBEdit;  
DBNavigator21: TDBNavigator;  
Edit89: TEdit;  
Edit90: TEdit;  
TabSheet18: TTabSheet;  
GroupBox18: TGroupBox;  
Label61: TLabel;  
Label62: TLabel;  
Label63: TLabel;  
Label64: TLabel;  
Label65: TLabel;  
Label76: TLabel;  
DBEdit54: TDBEdit;  
DBEdit55: TDBEdit;  
DBEdit56: TDBEdit;  
DBEdit57: TDBEdit;  
DBEdit58: TDBEdit;  
DBNavigator12: TDBNavigator;  
GroupBox21: TGroupBox;  
Label83: TLabel;  
Label84: TLabel;  
Label85: TLabel;  
Label86: TLabel;  
Label87: TLabel;  
Label109: TLabel;  
DBEdit69: TDBEdit;  
DBEdit70: TDBEdit;  
DBEdit71: TDBEdit;  
DBEdit72: TDBEdit;  
DBEdit73: TDBEdit;  
DBNavigator15: TDBNavigator;  
GroupBox28: TGroupBox;  
Label126: TLabel;

Label127: TLabel;  
Label128: TLabel;  
Label129: TLabel;  
Label130: TLabel;  
Label131: TLabel;  
DBNavigator22: TDBNavigator;  
DBEdit100: TDBEdit;  
DBEdit101: TDBEdit;  
DBEdit102: TDBEdit;  
Edit3: TEdit;  
Edit5: TEdit;  
TabSheet19: TTabSheet;  
GroupBox15: TGroupBox;  
Label46: TLabel;  
Label47: TLabel;  
Label48: TLabel;  
Label49: TLabel;  
Label50: TLabel;  
Label75: TLabel;  
DBEdit39: TDBEdit;  
DBEdit40: TDBEdit;  
DBEdit41: TDBEdit;  
DBEdit42: TDBEdit;  
DBEdit43: TDBEdit;  
DBNavigator9: TDBNavigator;  
GroupBox22: TGroupBox;  
Label88: TLabel;  
Label89: TLabel;  
Label90: TLabel;  
Label91: TLabel;  
Label92: TLabel;  
Label110: TLabel;  
DBEdit74: TDBEdit;  
DBEdit75: TDBEdit;



DBEdit76: TDBEdit;  
DBEdit77: TDBEdit;  
DBEdit78: TDBEdit;  
DBNavigator16: TDBNavigator;  
GroupBox29: TGroupBox;  
Label132: TLabel;  
Label133: TLabel;  
Label134: TLabel;  
Label135: TLabel;  
Label136: TLabel;  
Label137: TLabel;  
DBEdit103: TDBEdit;  
Edit7: TEdit;  
DBEdit104: TDBEdit;  
DBEdit105: TDBEdit;  
DBNavigator23: TDBNavigator;  
Edit93: TEdit;  
TabSheet20: TTabSheet;  
GroupBox17: TGroupBox;  
Label56: TLabel;  
Label57: TLabel;  
Label58: TLabel;  
Label59: TLabel;  
Label60: TLabel;  
Label74: TLabel;  
DBEdit49: TDBEdit;  
DBEdit50: TDBEdit;  
DBEdit51: TDBEdit;  
DBEdit52: TDBEdit;  
DBEdit53: TDBEdit;  
DBNavigator11: TDBNavigator;  
GroupBox23: TGroupBox;  
Label93: TLabel;  
Label94: TLabel;

Label95: TLabel;  
Label96: TLabel;  
Label97: TLabel;  
Label111: TLabel;  
DBEdit79: TDBEdit;  
DBEdit80: TDBEdit;  
DBEdit81: TDBEdit;  
DBEdit82: TDBEdit;  
DBEdit83: TDBEdit;  
DBNavigator17: TDBNavigator;  
GroupBox30: TGroupBox;  
Label138: TLabel;  
Label139: TLabel;  
Label140: TLabel;  
Label141: TLabel;  
Label142: TLabel;  
Label143: TLabel;  
DBEdit106: TDBEdit;  
Edit9: TEdit;  
DBEdit107: TDBEdit;  
DBEdit108: TDBEdit;  
DBNavigator24: TDBNavigator;  
Edit92: TEdit;  
TabSheet21: TTabSheet;  
GroupBox16: TGroupBox;  
Label51: TLabel;  
Label52: TLabel;  
Label53: TLabel;  
Label54: TLabel;  
Label55: TLabel;  
Label73: TLabel;  
DBEdit44: TDBEdit;  
DBEdit45: TDBEdit;  
DBEdit46: TDBEdit;



DBEdit47: TDBEdit;  
DBEdit48: TDBEdit;  
DBNavigator10: TDBNavigator;  
GroupBox24: TGroupBox;  
Label98: TLabel;  
Label99: TLabel;  
Label100: TLabel;  
Label101: TLabel;  
Label102: TLabel;  
Label112: TLabel;  
DBEdit84: TDBEdit;  
DBEdit85: TDBEdit;  
DBEdit86: TDBEdit;  
DBEdit87: TDBEdit;  
DBEdit88: TDBEdit;  
DBNavigator18: TDBNavigator;  
GroupBox31: TGroupBox;  
Label144: TLabel;  
Label145: TLabel;  
Label146: TLabel;  
Label147: TLabel;  
Label148: TLabel;  
Label149: TLabel;  
DBEdit109: TDBEdit;  
DBEdit112: TDBEdit;  
DBEdit113: TDBEdit;  
DBNavigator25: TDBNavigator;  
Edit110: TEdit;  
Edit111: TEdit;  
TabSheet22: TTabSheet;  
GroupBox19: TGroupBox;  
Label66: TLabel;  
Label67: TLabel;  
Label68: TLabel;

Label69: TLabel;  
Label70: TLabel;  
Label72: TLabel;  
DBEdit59: TDBEdit;  
DBEdit60: TDBEdit;  
DBEdit61: TDBEdit;  
DBEdit62: TDBEdit;  
DBEdit63: TDBEdit;  
DBNavigator13: TDBNavigator;  
GroupBox25: TGroupBox;  
Label103: TLabel;  
Label104: TLabel;  
Label105: TLabel;  
Label106: TLabel;  
Label107: TLabel;  
Label108: TLabel;  
DBEdit89: TDBEdit;  
DBEdit90: TDBEdit;  
DBEdit91: TDBEdit;  
DBEdit92: TDBEdit;  
DBEdit93: TDBEdit;  
DBNavigator19: TDBNavigator;  
GroupBox26: TGroupBox;  
Label118: TLabel;  
Label119: TLabel;  
Label114: TLabel;  
Label115: TLabel;  
Label116: TLabel;  
Label117: TLabel;  
DBEdit94: TDBEdit;  
Edit1: TEdit;  
DBEdit95: TDBEdit;  
DBEdit96: TDBEdit;  
DBNavigator20: TDBNavigator;



Edit66: TEdit;  
TabSheet6: TTabSheet;  
DBGrid5: TDBGrid;  
DBChart4: TDBChart;  
BarSeries1: TBarSeries;  
BarSeries2: TBarSeries;  
BarSeries3: TBarSeries;  
Series4: TBarSeries;  
Series5: TBarSeries;  
TabSheet7: TTabSheet;  
DBGrid4: TDBGrid;  
DBChart5: TDBChart;  
BarSeries4: TBarSeries;  
BarSeries5: TBarSeries;  
BarSeries6: TBarSeries;  
BarSeries7: TBarSeries;  
BarSeries8: TBarSeries;  
TabSheet8: TTabSheet;  
GroupBox2: TGroupBox;  
PageControl2: TPageControl;  
TabSheet9: TTabSheet;  
GroupBox3: TGroupBox;  
Label5: TLabel;  
Label4: TLabel;  
Label3: TLabel;  
Label2: TLabel;  
Label6: TLabel;  
EditTurkish: TDBEdit;  
EditSterling: TDBEdit;  
EditDollar: TDBEdit;  
EditEuro: TDBEdit;  
DBEdit6: TDBEdit;  
DBNavigator: TDBNavigator;  
TabSheet10: TTabSheet;

```

GroupBox4: TGroupBox;
Label1: TLabel;
Label7: TLabel;
Label8: TLabel;
Label9: TLabel;
Label10: TLabel;
DBEdit1: TDBEdit;
DBEdit2: TDBEdit;
DBEdit3: TDBEdit;
DBEdit4: TDBEdit;
DBEdit5: TDBEdit;
DBNavigator1: TDBNavigator;
Button2: TButton;
Button6: TButton;
Image3: TImage;
Image4: TImage;
Image5: TImage;
Image6: TImage;
Image7: TImage;
Image8: TImage;
Image9: TImage;
Image10: TImage;
Image11: TImage;
Image12: TImage;
procedure DBEdit23Change(Sender: TObject);
procedure DBEdit29Change(Sender: TObject);
procedure DBEdit32Change(Sender: TObject);
procedure Edit111Change(Sender: TObject);
procedure Edit90Change(Sender: TObject);
procedure Edit5Change(Sender: TObject);
procedure Edit93Change(Sender: TObject);
procedure Edit92Change(Sender: TObject);
procedure Edit66Change(Sender: TObject);
procedure Button1Click(Sender: TObject);

```



```

procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
procedure Button6Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure FormCreate(Sender: TObject);

```

```
private
```

```
    { Private declarations }
```

```
public
```

```
    { Public declarations }
```

```
end;
```

```
var
```

```
    MAINMENUCHANGEEXCHANGESYSTEM:
```

```
    TMAINMENUCHANGEEXCHANGESYSTEM;
```

```
implementation
```

```
uses REPORT, MENU, MENU2, REPORT4, REPORT5;
```

```
{ $R *.dfm }
```

```

procedure TMAINMENUCHANGEEXCHANGESYSTEM.DBEdit23Change(Sender:
TObject);

```

```
    var
```

```
    a,b,c:Currency;
```

```
begin
```

```
if (DBEdit22.gettextlen=0) or (DBEdit23.gettextlen=0) then
```

```
    begin
```

```
        showmessage('please enter value on edits');
```

```
        exit;
```

```
    end;
```

```
    begin
```

```

a:=strtoFloat(DBEdit22.text);
b:=strtoFloat(DBEdit23.text);
c:=a*b;
DBEdit24.text:=floattostr(c);
end;
end;

```

```

procedure TMAINMENUCHANGEEXCHANGESYSTEM.DBEdit29Change(Sender:
TObject);

```

```

    var
    a,b,c:Currency;

```

```

begin

```

```

if (DBEdit28.gettextlen=0) or (DBEdit29.gettextlen=0) then

```

```

    begin

```

```

        showmessage('please enter value on edits');

```

```

        exit;

```

```

    end;

```

```

    begin

```

```

a:=strtoFloat(DBEdit28.text);

```

```

b:=strtoFloat(DBEdit29.text);

```

```

c:=a*b;

```

```

DBEdit27.text:=floattostr(c);

```

```

end;

```

```

end;

```

```

procedure TMAINMENUCHANGEEXCHANGESYSTEM.DBEdit32Change(Sender:
TObject);

```

```

    var

```

```

    a,b,c:Currency;

```

```

begin

```

```

if (DBEdit31.gettextlen=0) or (DBEdit32.gettextlen=0) then

```

```

    begin

```

```

        showmessage('please enter value on edits');

```

```

    exit;
end;
begin
a:=strtoFloat(DBEdit31.text);
b:=strtoFloat(DBEdit32.text);
c:=a*b;
DBEdit33.text:=floattostr(c);
end;
end;

```

```

procedure TMAINMENUCHANGEEXCHANGESYSTEM.Edit111Change(Sender:
TObject);
    VAR
a,b,c,d:CURRENCY;
begin
    if (DBEdit109.gettextlen=0)OR (Edit110.gettextlen=0) or (Edit111.gettextlen=0) then
        begin
            showmessage('please enter value on edits');
            exit;
            end;
            begin
a:=strtoFloat(DBEdit109.text);
b:=strtoFloat(Edit110.text);
d:=strtoFloat(Edit111.text);
c:=(a * b)/d;

DBEdit112.text:=floattostr(c);
end;
end;

```

```

procedure TMAINMENUCHANGEEXCHANGESYSTEM.Edit90Change(Sender:
TObject);

```



```

VAR
a,b,c,d:CURRENCY;
begin
  if (DBEdit97.gettextlen=0)OR (Edit89.gettextlen=0) or (Edit90.gettextlen=0) then
    begin
      showmessage('please enter value on edits');
      exit;
      end;
      begin
a:=strtoFloat(DBEdit97.text);
b:=strtoFloat(Edit89.text);
d:=strtoFloat(Edit90.text);
c:= (a * b)/d;

DBEdit98.text:=floattostr(c);
end;
end;

//procedure TMAINMENUCHANGEEXCHANGESYSTEM.Edit4Change(Sender:
TObject);

procedure TMAINMENUCHANGEEXCHANGESYSTEM.Edit5Change(Sender:
TObject);
  VAR
a,b,c,d:CURRENCY;
begin
  if (DBEdit100.gettextlen=0)OR (Edit3.gettextlen=0) or (Edit5.gettextlen=0) then
    begin
      showmessage('please enter value on edits');
      exit;
      end;
      begin
a:=strtoFloat(DBEdit100.text);
b:=strtoFloat(Edit3.text);

```

**NEAR EAST UNIVERSITY**



**Faculty of Engineering**

**Department of Computer Engineering**

**DEVELOPMENT DELPHI PROGRAM  
FOR EXCHANGE AND CHANGE SYSTEM**

**Graduation Project  
COM- 400**

**Student: Ismail Mekki**

**Supervisor: Mr. Ümit ILHAN**

**Nicosia 2003 - 2004**

## ACKNOWLEDGEMENTS



*"No one can deny the role of the university in upgrading the mentality of the student to be capable to cover the requirements of the working life. This role is concerning with qualifying the student up to the level that the society needs. However, this role cannot be accomplished unless there is a qualified leader and sophisticated coach.*

*Fortunately, Mr. Umit ILHAN was the main reason of my success in this project, and thus, he deserves my all thanks, gratitude, and my respect due to his support and wise advice.*

*I appreciate all the effort that he provided during the preparation of this project.*

*Therefore, firstly, I would like to dedicate this project my family because of their unlimited support during my life.*

*Secondly, I would like to dedicate it to my supervisor Mr. Umit ILHAN because of his wise supervision on this project and for his wide knowledge.*

*Finally, I appreciate all the effort aids of my friends during the preparation of this project."*

*With all due respect*



## ABSTRACT

This program is designed for the change and exchange markets where the individuals deal with the currencies and barter them. This system is based on BORLAND DELPHI 6 programming language. All the criterions of this system are taken according upon the request of the Near East Bank and all the features of this software can be adjusted according to the desire of the customer.

All the screens that will appear in the usage of this program will be illustrated in the coming chapters in details. The calculations and the mathematical operations that this program applies are explained in details in the appendix at the end of this report. This program is designed to find out the exact profit of the company by using graphical charts and by showing statues reports at any time. Three hard currencies are taken into consideration in this system and they are compared with the Turkish lira, which is the local currency. The values of these hard currencies will be taken from the stock markets that the government decides daily according o the daily economic level. The decision maker in NEB will determine the amount of the profit that he desires and the total revenue will be calculating by the software. A navigation bar is located in the bottom of each screen for adding, deleting, saving... etc.

## TABLE OF CONTENTS

<b>ACKNOWLEDGEMENTS</b>	<b>I</b>
<b>ABSTRACT</b>	<b>II</b>
<b>TABLE OF CONTENTS</b>	<b>III</b>
<b>LIST OF FIGURES</b>	<b>V</b>
<b>CHAPTER ONE: INTRODUCTION</b>	<b>1</b>
1.1. Overview	1
1.2. Economic Analysis of Floating Exchange Rate System	2
1.2 a Introduction	2
1.2 b Opportunities From Around World	2
1.3 Delphi Programming	5
1.4. Database Programming	7
1.5. Borland Database Engine (BDE)	8
1.6. Graphical Data-Aware Control	8
1.7. The clintDataSet Component	9
1.8 Classic BDE component	10
1.9 Tables and Queries	10
1.10 DBNavigator and Dataset Actions	11
1.11 Text-Based Data-Aware Control	12
1.12 Navigator a Dataset	12
 <b>CHAPTER TWO: THE DESCRIPTION OF THE SYSTEM</b>	 <b>14</b>
2.1 Main Menu Screen	14
2.2 Change System	18
2.3. Exchange curve	19
2.4. Exchange System	21
2.5. Exchange Bar Graphs	22

2.6. Adding the Exchange rates	23
2.7. Database Desktop	25
<b>CHAPTER THREE: DATA SOURCE</b>	26
3.1. Data Flow Diagram	26
3.2 . Main Menu	27
3.3. Buy Change System Preview	48
3.4. Exchange System Sterling	51
3.5. Change System Sterling	52
3.6. Exchange System Sterling Dollar	52
3.7. Exchange System Dollar Euro	54
3.8. Change System Dollar Euro	55
3.9. Exchange System Euro	55
3.10. Exchange System Dollar Sterling	57
3.11. Change System Dollar Sterling	58
3.12. Sell Exchange System	58
3.13. Exchange Euro to Sterling	60
3.14. Exchange Euro	61
3.15. Exchange System Dollar to Euro	61
3.16. Buy Change System preview	63
3.17. Buy Change System	66
3.18. Buy Change System	67
3.19. Change System Sterling	68
3.20. Change System	69
3.21. Report Dollar	71
3.22. Sell Change System Preview	72
<b>CONCLUSION</b>	75
<b>SYSTEM REQUIREMENTS</b>	75
<b>REFERENCES</b>	76



## LIST OF FIGURES

### CHAPTER II

#### CHANGE AND EXCHANGE SYSTEMS

2.1 Main Menu Screen	14
2.2 Buy change system preview	15
2.3 Exchange report	15
2.4 Exchange rates	16
2.5 Change System Preview	17
2.6 Exchange system Preview	17
2.7 Change System	18
2.8 Change Curve	19
2.9 Change Curve	20
2.10 Change Curve	20
2.11 Exchange System	21
2.12 Exchange Bar Graphs a	22
2.13 Exchange Bar Graphs b	23
2.14 Adding the Exchange Rates (Selling Rates)	23
2.15 Adding the Exchange Rates (Foreign Currency Rates)	24
2.16 Database Desktop	25

### CHAPTER III

#### DATA SOURCE

3.1 Data Flow Diagram	26
-----------------------	----

# INTRODUCTION

## 1.1 Overview

Many activities are done daily in the stock market. Most of these activities are concerning with the transactions and exchanging the goods and services. Since the market is the place that the buyers and sellers exchange goods and services, then the currencies are considered as a good that the people buy and sell within the market.

As it is known, the most currencies that are used widely are those currencies that are called the hard currencies such as US Dollar, UK sterling, and Euro. Therefore, as we are concerning with controlling the activities in the market by issuing the required software programs that include all features of controlling the business by means of scientific methods in order to be used in a proper and easy way.

Using any programming language to create certain software gives you the opportunity of gaining a good experience and it will promote your approaches of analyzing the aspects of any given project.

Therefore, any software needs an adequate and sufficient programming language that helps us to set up all the applications and all the functions that include the features of organizing and sustaining our business.

The role of a computer engineer requires an adequate ability of analyzing and dealing with all the aspects of any project in order to be capable to create and issue certain software that controls the business. Therefore, our role as an engineer is to not only deal with the hardware, but also deal with creating a software programs. Thus, the role of an engineer is to control the technical specifications of the firm and to know every single aspect that is necessary to measure the level of the success.

## **1.2 Economic Analysis of Floating Exchange Rate Systems**

### **1.2.a Introduction**

Associated foreign exchange is looking for experienced foreign exchange sales professionals! Successful candidates will be strongly motivated individuals who can identify and cultivate potential corporate clients with international payment needs. Strong communication skills, experience in the foreign exchange field and experience in relationship based selling are a must. Successful candidates will be responsible for opening, maintaining and growing accounts through relationship building, identifying customer needs and cross-selling appropriate services.

### **1.2 b Opportunities from Around the World**

Over the last three decades the foreign exchange market has become the world's largest financial market, with over \$1.5 trillion USD traded daily. The primary market for currencies is the 24-hour Interbank market. The Interbank market literally follows the sun around the world, moving from major banking centers of the United States to Australia and New Zealand to the Far East, to Europe and finally back to the United States. With the large minimum transaction sizes and often-stringent financial requirements, banks, hedge funds, major currency dealers and the occasional high net-worth individual speculator were the principal participants. These large traders were able to take advantage of the many benefits offered by the forex market vs. other markets including fantastic liquidity and the strong trending nature of the world's primary currency exchange rates.

The business section of any newspaper will have a table of spot exchange rates. These are the rates at which a person could have bought other currencies or foreign Exchange, such as the English Pound, French Franc, or the new European Euro. The Prices of foreign currencies can be determined in two major types of exchange rate Systems. In the United States, the dollar's exchange rates are determined by the Marketplace, i.e., by supply and demand. This type of system is called a floating Exchange rate system. In other countries, governments set the price of their currencies With respect to other countries. They then buy or sell foreign exchange at the prices They've set. This is called a fixed exchange rate system. The economic effects of these Two systems can be very different. However, in either system the underlying forces Influencing the value of a country's currency remain the same. Due to possible



confusion of being able to quote different currencies in terms of Each other, e.g., \$/£ or £/\$, we need to explicitly define an exchange rate. An **exchange Rate** is, therefore, the domestic cost of a unit of foreign exchange. For example, from the US perspective the price of the English Pound would be denominated as the number of US dollars per pound, or \$/£. As noted above, the exchange rate in a floating exchange rate system is Determined by market forces. Our definition of the exchange rate defines the market as The market for foreign exchange. In this market we have demanders and suppliers of Foreign currencies willing to pay and accept dollars in return for these currencies. We Will in turn discuss the demand and supply of foreign exchange. Foreign Exchange Demand The demand for foreign exchange is a derived demand. With the exception of currency Collectors, the demand for foreign exchange is due to people's desire to use it in the Purchase of foreign goods or financial assets. Foreign exchange demand is, therefore, Highly sensitive to changes in these desires.

In order to understand changes in the demand for foreign exchange, we will need to Discuss its underlying forces. These are the demand for foreign goods and services and the demand for foreign financial assets. The supply of foreign exchange has at its roots the same conceptual basis as Demand, only it is from the foreign perspective. Foreign currency is supplied to the Foreign exchange market when foreigners exchange their currency for dollars in order to Buy US goods or financial assets. Equivalently, the supply of foreign exchange is Nothing more than a mirror image of the foreign demand for US currency. Exchange is the mirror of the supply of dollars to the foreign exchange market. One question which might arise is which foreign exchange Market. New York, London, Frankfurt and Tokyo are Major financial centers with large foreign exchange markets. The answer as to which market is all of them. The first rule of business is to buy low and sell high. Should exchange rates be different across different financial centers, then the opportunity for arbitrage profits occurs. Currency dealers will buy low in one center and sell high in another, driving exchange rates into equality Across the different markets. For example, should the Swiss Franc be at a lower price (in terms of \$) in London and at a higher price in New York, then the dealers will increase the demand for the Swiss Franc in London, driving up its price, and increase its supply in New York, driving down its price there. This continues until the price is the same in both places. The major questions to be addressed are how exchange rates determined are and what the forces which influence them are. In Figure 1, the equilibrium exchange rate (e) is the one where the quantity demanded is equal to the quantity supplied for

foreign exchange. As with most markets, the price changes in order to equilibrate the market. When quantity demanded exceeds quantity supplied, and then the exchange rate will rise. If the quantity supplied is greater than quantity demanded, the exchange rate falls. What does it mean when the exchange rate rises or falls? As we have defined the Exchange rate (\$/£), when the exchange rate rises, the value of the dollar decreases or depreciates. It now takes more dollars to buy an English pound than it did before the Change in the exchange rate. Fewer foreign goods can now be purchased for a given number of dollars. The reverse is also true. As the exchange rate falls, the dollar cost of foreign exchange falls, increasing the dollar's value. This is termed an **appreciation** of the dollar. More foreign exchange rate \$/£ Foreign exchange Sfx or D\$ Dfx or S\$. Market should force lead to a change in either the supply or demand for foreign exchange then the exchange will change accordingly to re-equilibrate the market. The basic notion is that exchange rates are sensitive to differential inflation rates across Countries. Should the domestic inflation rate rise at a rate greater than our trading Partners, then at a given exchange rate, the price of domestic goods will be rising relative To foreign goods. This will, in turn, increase the demand for foreign goods (imports are Now cheaper in domestic currency terms) and decrease the demand for domestic exports (Domestic exports are now more expensive in foreign currency terms). This results in an Increase in the demand for foreign exchange, as well as a decrease in the supply of Foreign exchange.

This is a long-run effect because of the Law of One Price. This concept states that in the Long run the price of tradable goods must be the same across countries. If this was not The case, then the opportunity for arbitrage profits, buying low in one country and selling High in another, would result in a movement in the exchange rate bringing about the Equalization. For example, suppose Argentine wheat, at the prevailing exchange rate, is cheap in US Dollars. As North Americans buy more and more Argentine wheat, they increase the Demand for the Argentine currency, driving up its value, thus making wheat more Expensive in dollar terms. The exchange rates which would prevail under the Law of One Price are called purchasing power parity exchange rates (PPP). While these do not exist in reality (there are many other factors affecting exchange rates) there is an underlying pressure moving exchange rates in this fashion. PPP exchange rates are used in comparing the economic performance between countries. The World Bank compares countries in their *World Development Report* using a PPP exchange rate. Medium Term - Differential Growth Rates As an economy



grows, its demand for imports will also grow. As income increases, some portion of that increase will be spent on imported goods. In the jargon of macroeconomics, the proportion of the additional dollar of income spent on imports is called the marginal propensity to import. Assume that the marginal propensity to import is the same across countries. Should a country's economy grow faster than its trading partners, then its demand for imports will also be growing faster? In the context of Figure 4, this is represented by increases in both the demand and supply of foreign exchange, but the demand would increase by more. This would result in a slight depreciation of the domestic currency. Short-Run - Differential Interest Rates This factor has become extremely important as countries have liberalized their economies, allowing the flow of financial capital into and out of their countries. It has played an important role in the East Asian and Mexican Peso financial crises. Exchange rate \$/£ Foreign exchange.

### 1.3 Delphi Programming

Delphi 5 provided new features to the Object Inspector, and Delphi 6 includes even more additions to it. As this is a tool programmer's use all the time, along with the editor and the Form Designer, its improvements are really significant.

The most important change in Delphi 6 is the ability of the Object Inspector to expand component references in-place. Properties referring to other components are now displayed in a different color and can be expanded by selecting the + symbol on the left, as it happens with internal subcomponents. You can then modify the properties of that other component without having to select it.

**NOTE** This interface-expansion feature also supports subcomponents, as demonstrated by the new Labeled Edit control. The Form Designer

**TIP** A related feature of the Object Inspector is the ability to select the component referenced by a property. To do this, double-click the property value with the left mouse button while keeping the Ctrl key pressed. For example, if you have a Main Menu component in a form and you are looking at the properties of the form in the Object Inspector, you can select the Main Menu component by moving to the Main Menu property of the form and Ctrl+double-clicking the value of this property. This selects the main menu indicated as the value of the property in

the Object Inspector. Here are some other relevant changes of the Object Inspector: The list at the top of the Object Inspector shows the type of the object and



can be removed to save some space (and considering the presence of the Object Tree View). The properties that reference an object are now a different color and may be expanded without changing the selection. You can optionally also view read-only properties in the Object Inspector. Of course, they are grayed out. The Object Inspector has a new Properties dialog box which allows you to customize the colors of the various types of properties and the overall behavior of this window.

The Project Manager doesn't provide a way to set the options of two different projects at one time. What you can do instead is invoke the Project Options dialog from the Project Manager for each project. The first page of Project Options (Forms) lists the forms that should be created automatically at program startup and the forms that are created manually by the program.

The next page (Application) is used to set the name of the application and the name of its Help file, and to choose its icon. Other Project Options choices relate to the Delphi compiler and linker, version information, and the use of run-time packages.

There are two ways to set compiler options. One is to use the Compiler page of the Project Options dialog. The other is to set or remove individual options in the source code with the `{SX+}` or `{SX-}` commands, where you'd replace *X* with the option you want to set. This second approach is more flexible, since it allows you to change an option only for a specific source-code file, or even for just a few lines of code. The source-level options override the compile-level options.

All project options are saved automatically with the project, but in a separate file with a .DOF extension. This is a text file you can easily edit. You should not delete this file if you have changed any of the default options. Delphi also saves the compiler options in another format in a CFG file, for command-line compilation. The two files have similar content but a different format: The *dcc* command-line compiler, in fact, cannot use .DOF files, but needs the .CFG format. Another alternative for saving compiler options is to press Ctrl+O+O (press the O key twice while keeping Ctrl pressed). This inserts, at the top of the current unit, compiler directives that correspond to the current project options, as in the following listing: `{SA+,B-,C+,D+,E-,F-,G+,H+,I+,J+,K-,L+,M-,N+,O+,P+,Q-,R-,S-,T-,U-,V+,W-,X+,Y+,Zl}`

Memory management in Delphi is subject to three rules: Every object must be created before it can be used; every object must be destroyed after it has been used; and every object must be destroyed only once. Whether you have to do these operations in

your code, or you can let Delphi handle memory management for you, depends on the model you choose among the different approaches provided by Delphi.

Delphi supports three types of memory management for dynamic elements (that is, elements not in the stack and the global memory area):

- Every time you create an object explicitly, in the code of your application, you should also free it. If you fail to do so, the memory used by that object won't be released for other objects until the program terminates.

- When you create a component, you can specify an owner component, passing the owner to the component constructor. The owner component (often a form) becomes responsible for destroying all the objects it owns. In other words, when you free the form, it frees all the components it owns. So, if you create a component and give it an owner, you don't have to remember to destroy it. This is the standard behavior of the components you create at design time by placing them on a form or data module.

- When you allocate memory for strings, dynamic arrays, and objects referenced by interface variables, Delphi automatically frees the memory when the reference goes out of scope. You don't need to free a string: when it becomes unreachable, its memory is released.

## **1.4 Database Programming**

Delphi's support for database applications is one of the key features of the programming environment. Many programmers spend most of their time writing data-access code, which needs to be the most robust portion of a database application. This chapter provides an overview of Delphi's extensive support for database programming. What you will find here is a discussion of the theory of database design. I am assuming that you already know the fundamentals of database design and have already designed the structure of a database. I will not look into database-specific problems; my goal is to help you understand how Delphi supports database access. I will begin with an explanation of the alternatives Delphi offers in terms of data access, and then I will provide an overview of the database components that I have used in my program. This chapter includes an overview of the TDataSet class, an in-depth analysis of the TField components, and the use of data-aware controls. The following chapters will provide information on more advanced database programming topics, such as client/server programming, the use of dbGo, dbExpress, and InterBase Express



## 1.5 Borland Database Engine (BDE)

The BDE originated with Paradox, well before Delphi existed, and was extended by Borland to support other local databases and many SQL servers. The BDE has direct access to dBASE, Paradox, ASCII, FoxPro, and Access tables. A series of drivers (called SQL Links and available only in Delphi Enterprise) allows access to some SQL servers, including Oracle, Sybase, Microsoft, Informix, InterBase, and DB2 servers. If you need access to a different database, the BDE can also interface with ODBC drivers.

## 1.6 Graphical Data-Aware Controls

Finally, Delphi includes two graphical data-aware controls:

- **DBImage**, which is an extension of an Image component that shows a picture stored in a BLOB field (provided the database use a graphic format that the Image component supports, such as BMP and JPEG). The output of the Cust- Lookup example, with the BLookupComboBox showing multiple fields in its drop-down list.

- **DBChart** is a data-aware business graphic component or the data-aware version of the TeeChart control built by David Berneda. To demonstrate the use of the DBChart control, I have added this component to a simple example showing a data grid. The application, called ChartDB, shows a pie chart with the surface of each country of the COUNTRY.DB table. The program has almost no code, as all the settings can be done using the specific component editor, which has several options but is quite easy to use. Here are some of the key properties of the component, taken from the form description:

**object** DBChart1: TDBChart

Legend.Visible = False

Align = alClient

**object** Series1: TPieSeries

Marks.ArrowLength = 8

Marks.Visible = True

DataSource = Table1

XLabelsSource = 'Name'

ExplodeBiggest = 3

OtherSlice.Style = poBelowPercent

OtherSlice.Text = 'Others'



```

OtherSlice.Value = 2
PieValues.ValueSource = 'Area'
end;
end.

```

What I have done is show the area field as the data source for the pie chart (the PieValues ValueSource property of the series), use the name field for the labels (the XLabelsSource property of the series), and condense all the countries with a value below 2 percent in a single section indicated as Others (the OtherSlide subproperties). As a minor addition to the code, I have added two radio buttons you can use to toggle between the area and the population. The code of the two radio buttons simply sets the source of the series, after casting it to the proper series type, as in:

```

procedure TForm1.RadioPopulationClick(Sender: TObject);
begin
  DBChart1.Title.Text [0] := 'Population of Countries';
  (DBChart1.Series [0] as TPieSeries).PieValues.ValueSource := 'Population';
end;

```

## 1.7 The ClientDataSet Component

Finally, there is a component derived from TDataSet that has a peculiar behavior and can be combined with other data-access components. The ClientDataSet component, in fact, is a dataset accessing data kept in memory. The in-memory data can be totally temporary (lost as you exit the program), saved to a local file as a snapshot, and imported by another dataset using a Provider component. This last situation is certainly the most common: You can hook a ClientDataSet to any other local dataset, or use Borland's multitier support (discussed in Accessing a Database: BDE, dbExpress, and other alternatives "Multitier Database Applications with DataSnap") to retrieve data from a dataset hosted by a different application, possibly running on a separate computer. The ClientDataSet component becomes particularly useful if the data-access components you are using provide limited or no caching. This is particularly true of the new dbExpress engine, but can equally help you when using the BDE or other native components.

On the other hand, ADO already provides most of the services of the ClientDataSet component and using these two at the same time can be useful only in limited situations

## 1.8 Classic BDE Components

Each of the database-access solutions discussed above has its own set of data-access, database connection, and extra utility components on a specific page of the Component palette. The classic BDE components have been moved to the new BDE page and include the Table, Query, and StoredProc components. The ADO, dbExpress, and InterBase Express components are each in specific pages, and all include specific dataset components and others that tend to mimic the BDE components, simplifying the porting of existing applications.

The Data Access page of the Component palette includes only the Data Source Component and others not specifically related with any single data access technology. Besides the data-access component of your choice, a Delphi visual application generally uses some data-aware controls (in the Data Controls page) and the DataSource component. Data-aware controls are visual components used to view and edit the data in a form and are extensions of standard components such as edit and list boxes, radio buttons, images, and the

Grid. The DataSource component has the role of connector between the data-aware controls and a dataset component.

## 1.9 Tables and Queries

The simplest traditional way to specify data access in Delphi was to use the BDE Table component. A Table object simply refers to a database table. When you use a Table component, you need to indicate the name of the database you want to use in its DatabaseName property. You can enter an alias or the path of the directory with the table files. The Object Inspector lists the available names, which depend on the aliases installed in the BDE. You also need to indicate a proper value in the TableName property. The Object Inspector lists the available tables of the current database (or directory), so you should generally select the DatabaseName property first. Another classic dataset is the BDE Query component. A query requires a SQL language



command. You can customize a query using SQL more easily than you can customize a table (as long as you know at least the basic elements of SQL, of course). The Query component has a `DatabaseName` property like the Table component, but it does not have a `TableName` property. The table is indicated in the SQL statement, stored in the SQL property. For example, you can write a simple SQL statement like this:

**Select \* from** Country where Country is the name of a table and the asterisk (\*) indicates that you want to use all of the fields in the table.

The efficiency of a table or a query varies depending on the database you are using. In general, we can say that the Table component tends to be faster on local tables, while the Query component tends to be faster on SQL servers, although this is just a very general rule, and in many cases you might have the opposite effect. We'll see some efficiency issues while discussing

client/server development in the third BDE dataset component is `StoredProc`, which refers to stored procedures of a SQL server database. You can run these procedures and get the results in the form of a database table. Stored procedures can only be used with SQL servers.

## 1.10 DBNavigator and Dataset Actions

DBNavigator is a collection of buttons used to navigate and perform actions on the database. You can disable some of the buttons of the DBNavigator control, by removing some of the elements of the `VisibleButtons` set. The buttons perform basic actions on the connected dataset, so you can easily replace them

With your own toolbar, particularly if you use an `ActionList` component with the predefined database actions provided by Delphi. In this case, in fact, you get all the standard behaviors, but you'll also see the various buttons enabled only when their action is legitimate. **TIP** If you use the standard actions, you can avoid connecting them to a specific `DataSource` component, and the actions will be applied to the dataset connected to the visual control that currently has the input focus. This way a single toolbar can be used for multiple datasets displayed by a form.



## 1.11 Text-Based Data-Aware Controls

There are multiple text-oriented components:

DBText displays the contents of a field that cannot be modified by the user. It is a data aware Label graphical control. It can be very useful, but users might confuse this control with the plain labels that indicate the content of each field-based control. DBEdit lets the user edit a field (change the current value) using an Edit control. At times, you might want to disable editing and use a DBEdit as if it were a DBText, but highlighting the fact that this is data coming from the database. DBMemo lets the user see and modify a large text field, eventually stored in a memo or BLOB (binary large object) field. It resembles the Memo component and has full editing capabilities, but all the text is rendered in a single font.

DBRichEdit is a component that lets the user edit a formatted text file; it is based on a RichEdit Windows common control and, in contrast to DBMemo, it allows text with multiple fonts and paragraph styles.

## 1.12 Navigating a Dataset

We've seen that a dataset has only one active record, and you can imagine that the active record changes often, in response of user actions or because of internal commands given to the dataset. To move around the dataset and change the active record, there are methods of the TDataSet class, particularly in the section commented as "position, movement." You can move to the next or previous record, jump back and forth by

A given number of records (with MoveBy), or go directly to the first or last record of the dataset. These operations of the dataset are generally available in the DBNavigator component or in the standard dataset actions, and they are not particularly complex to understand. What is not obvious, though, is how a dataset handles the extreme positions. If you open any dataset with a navigator attached, you can see that as you move on record by record, the Next button remains enabled even when you've reached the last record. It's only when you try to move forward after the last record that the current record apparently doesn't change and the button is disabled. This is because the Eof test (end of file) succeeds only when the cursor has been moved to a special position after the last record. If you jump to the end with the Last button, instead, you'll immediately be at the very end. You'll see exactly the same behavior for the first record

(and the Bof test). As we'll see in a while, this approach is very handy, as we can scan a dataset testing for Eof to be True and, at this point, we know we've also already processed the last record of the dataset.

**NOTE** Handling this special record positions before the beginning and after the end of the dataset, which are called *cracks*, is very important (and quite confusing) when you write a custom dataset. Besides moving around record by record or by a given number of records, programs might need to jump to specific records or positions. Some datasets support the RecordCount property and allow movement to a record at a given position in the dataset using the RecNo property. These properties can be used only for datasets that support positions natively, which basically excludes all client/server architectures, unless you grab all of the records in a local cache (something you'll generally want to avoid) and then navigate on the cache. As we'll see in the next chapter, when you open a query on a SQL server you fetch only the records you are using, so Delphi doesn't know the record count, at least not in advance. There are two alternatives you can use to refer to a record in a dataset, regardless of its type. You can save a reference to the current record and then jump back to it after moving around. This is accomplished by using bookmarks, either in the TBookmark or the more modern TBookmarkStr form. You can locate a record of the dataset matching given criteria, using the Locate method. This even works after you close and reopen the dataset, because you're working at a logical (and not physical) level. This approach is presented in the next section.



## 2.1 Main Menu Screen



Figure 2.1 Main Menu Screen

The figure above shows the first screen which occurs when you first start running the program. This screen contains the following:

- ◆ The calendar: This automatically obtains the current date month and year of the system which the program is running on.
- ◆ Buy change system preview: which shows the current selling price of currencies that you have entered in the change system, which will be explained later.



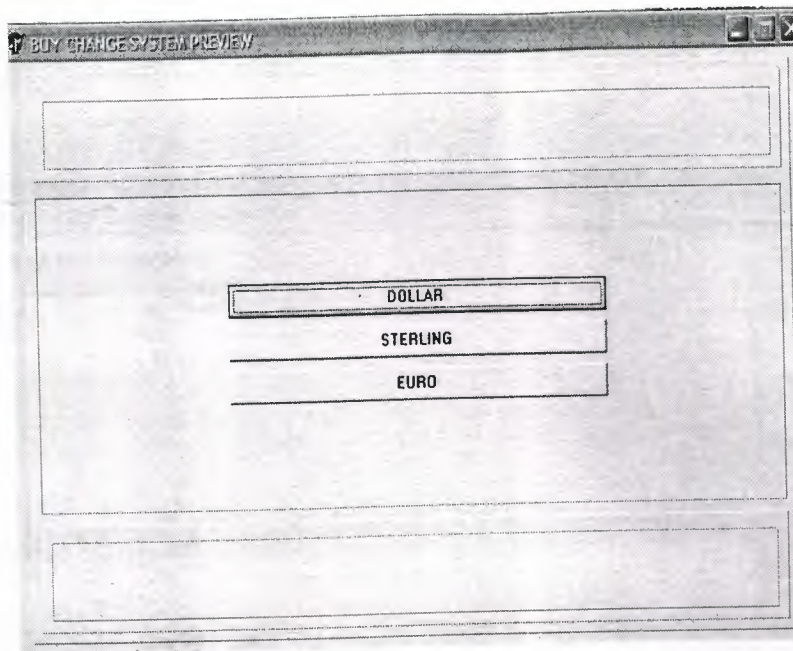


Figure 2.2 Buy change system preview

- ◆ Sell exchange system preview: which shows the current reports of the exchange system. The following screen will appear and allow you to Select the type of exchange report that you wish to preview.

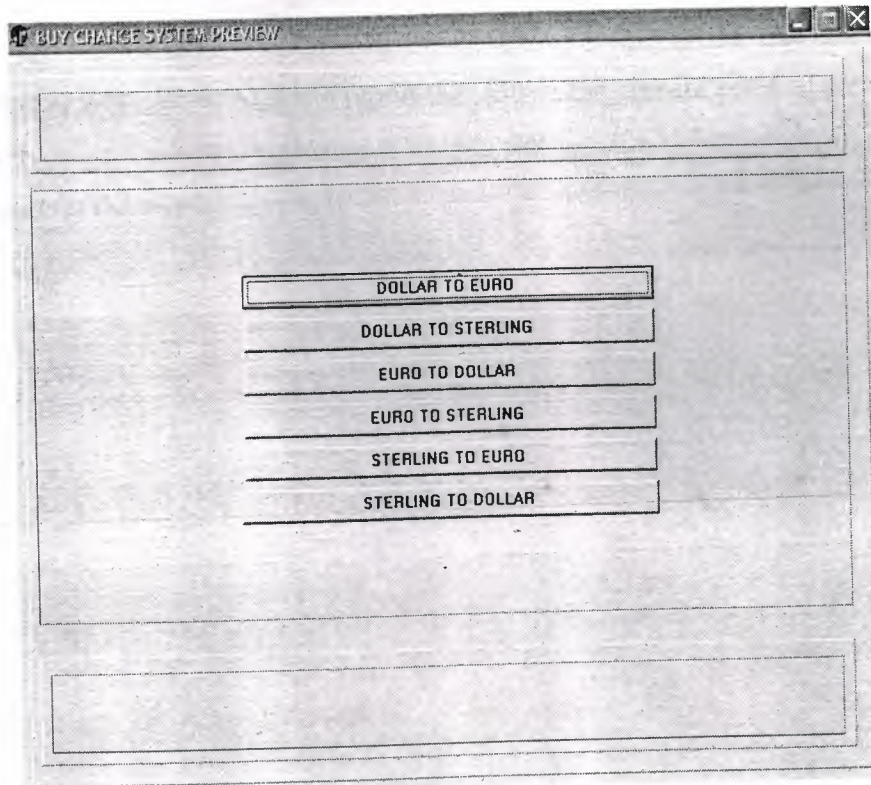
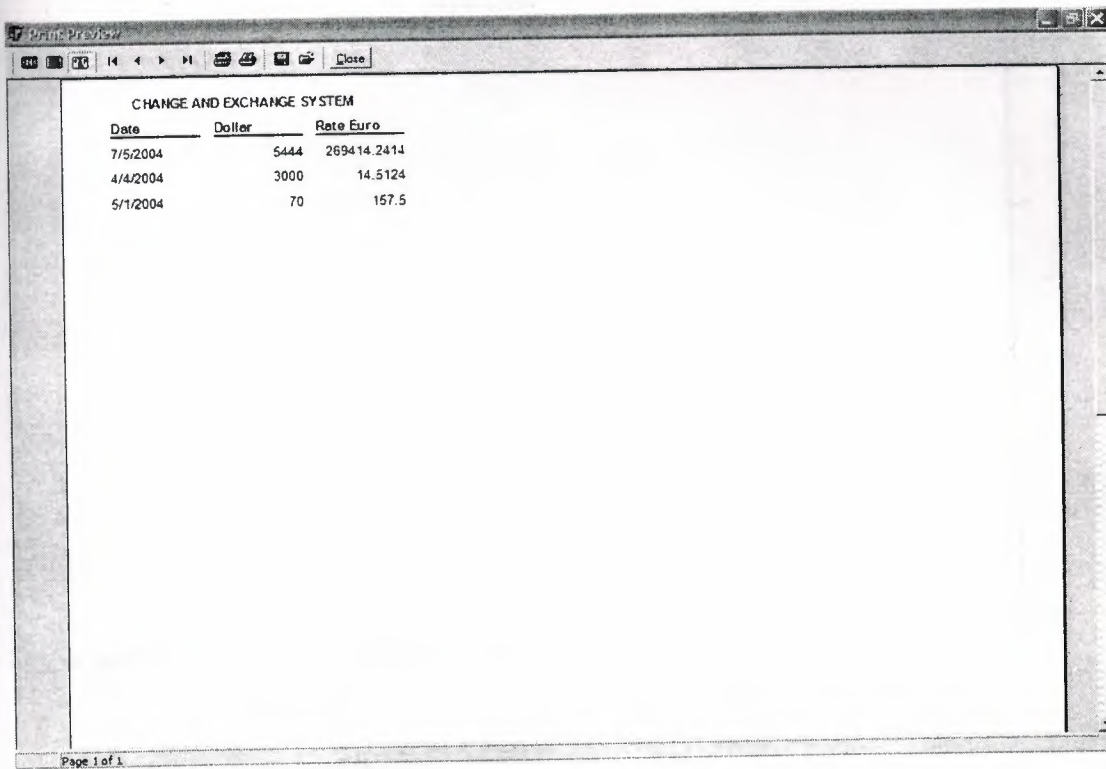


Figure 2.3 Exchange report

The following report preview will show the exchange rates related to which form you choose:



Date	Dollar	Rate Euro
7/5/2004	5444	269414.2414
4/4/2004	3000	14.5124
5/1/2004	70	157.5

Figure 2.4 Exchange rates

- ◆ Change system preview: it is a report that shows the current price of the currencies in the stuck exchange related to selling currencies, which can be printed by the button next to it.

Turkish	Sterling	Dollar	Euro	Date
1430000	2400000	1400000	2700000	1/1/2004

Figure 2.5 Change System Preview

- ◆ Exchange system: it is a report that shows the current price of the currencies in the stuck exchange related to buying currencies, which can be printed by the button next to it.

Turkish	Sterling	Dollar	Euro	Date
1200000	1400000	1700000	2000000	1/1/2004

Figure 2.6 Exchange System Preview



## 2.2 Change System

MAIN MENU CHANGE EXCHANGE SYSTEM

NEAR EAST UNIVERSITY  
DEPARTMENT OF COMPUTER ENGINEERING  
DEVELOPMENT DELPHI PROGRAM FOR EXCHANGE AND CHANGE SYSTEM

CHANGE SYSTEM REPORT CHANGE SYSTEM

TURKISH TO DOLLAR | TURKISH TO STERLING | **TURKISH TO EURO**

TL Turkish	£ Sterling	\$ Dollar	€ Euro
1430000	2400000	1400000	2700000

Date  
1/1/2004

ADD TO CHANGE SYSTEM:

Date  
7/5/2004

EXCHANGE RATE	EXCHANGE AMOUNT	EXCHANGE VALUE
27000000	100	270000000

Figure 2.7 Change System

This screen consists of three sub-menus, which will allow you to complete your transaction processes. In addition, the forms contain the following:

The first part of each form grapes the exchange rates of the currencies from your database that you have entered.

The second part deals with the current amount of currency that you are exchanging at the time being. It consists of a navigation bar that allows you to scroll in the fields of your transactions.

## 2.3 Exchange Curves

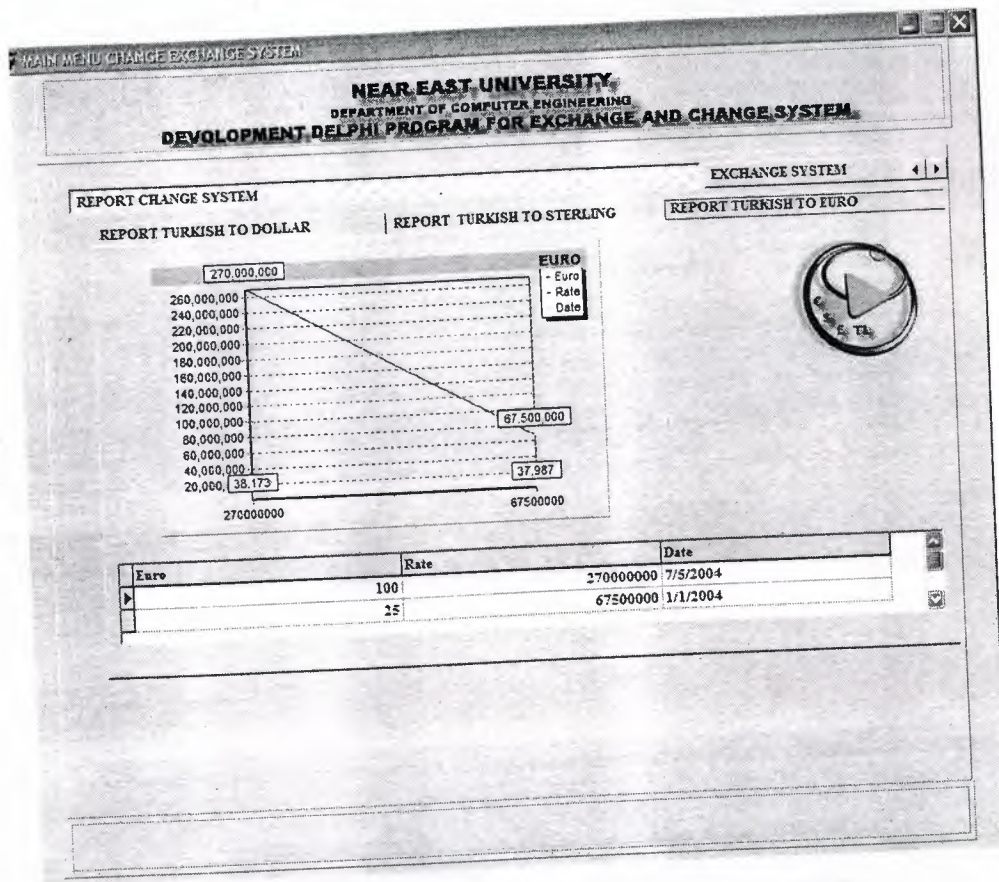


Figure 2.8 Change Curve

It consists of three chart reports that are all linked to the data base and change in respect to the processes that you make, that each transaction made is added or deleted and presented in the as a curve in respect to the money amount you have processed. The rest of the graphs are shown below



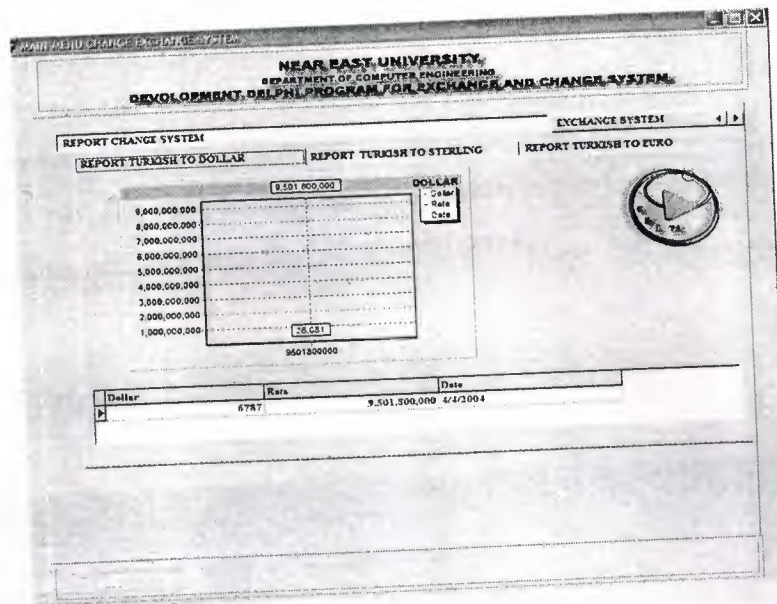


Figure 2.9 Change Curve

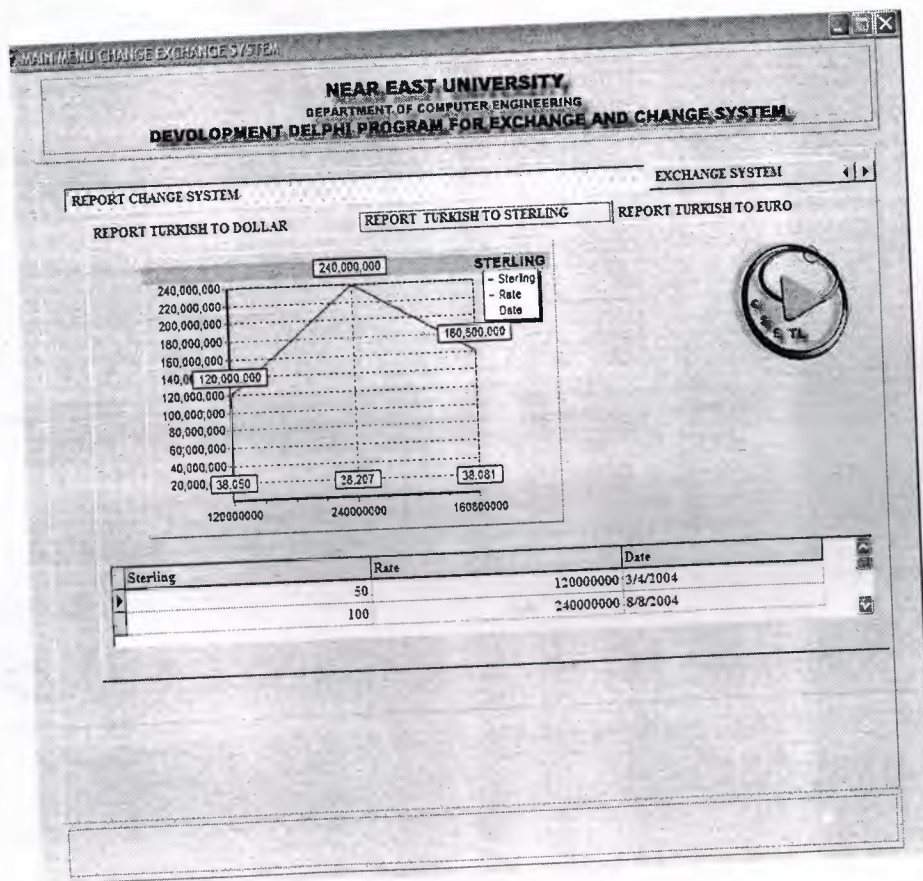


Figure 2.10 Change Curve



These graphs are just to keep track of you exchange processes in terms of money, and also shows you which of the currencies is have the most demand in the market, at this point you can analyze the current situation of the foreign currencies flow in you exchange office and take your decisions among that.

## 2.4 Exchange System

MAIN MENU CHANGE EXCHANGE SYSTEM

**NEAR EAST UNIVERSITY**  
DEPARTMENT OF COMPUTER ENGINEERING  
DEVELOPMENT DELPHI PROGRAM FOR EXCHANGE AND CHANGE SYSTEM

EXCHANGE SYSTEM INFO CHANGE SYSTEM

EURO TO DOLLAR

CHANGE	[TL] Turkish	[£] Sterling	[\$] Dollar	[€] Euro	Date
	1430000	2400000	1400000	2700000	1/1/2004

EXCHANGE

EXCHANGE	[TL] Turkish	[£] Sterling	[\$] Dollar	[€] Euro	Date
	1200000	1400000	1700000	2000000	1/1/2004

EXCHANGE

EXCHANGE	[\$] Dollars given amount	[\$] DOLLAR PRICE	[€] woned ratio	equal amount	Date
	54345			40081.7156	7/8/2004




Figure 2.11 Exchange System

The exchange system consists of three fields that allow you to deal with the exchange of the foreign currencies each one consist of selling and buying prices of the currencies and another field for your processes, where you can add delete or modify your processes.

## 2.5 Exchange Bar Graphs

The following bar graphs illustrate the change in the currencies by date and respectively with the database.

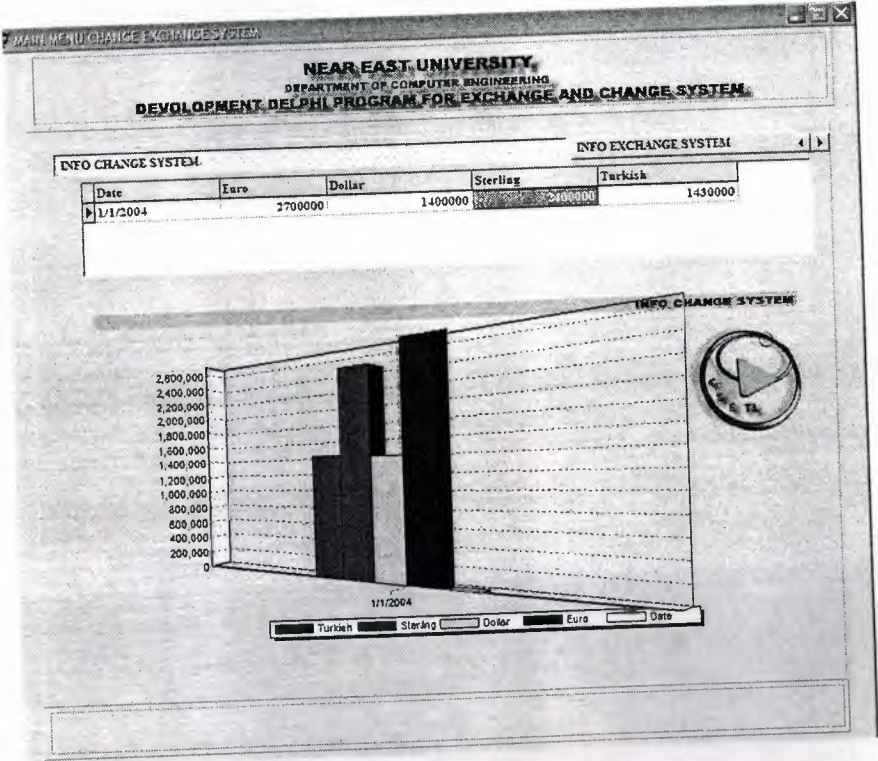


Figure 2.12 Exchange Bar Graphs a



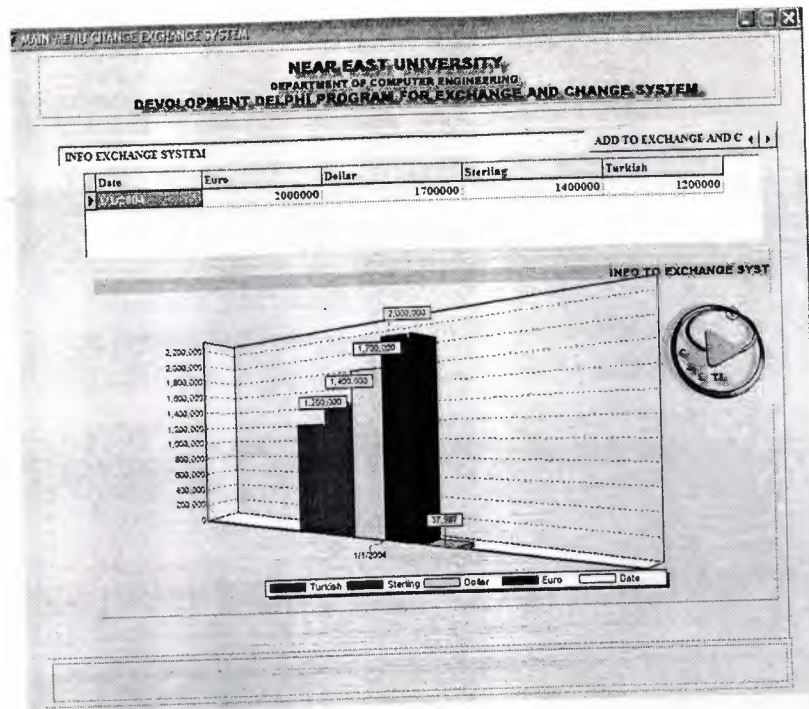


Figure 2.13 Exchange Bar Graphs b

## 2.6 Adding the Exchange Rates

a. **Selling Rates:** here in this form you can add the current rates of the local currency.

The screenshot shows the "ADD TO EXCHANGE AND CHANGE SYSTEM" form. It includes a section for "EXCHANGE SYSTEM" with a table of exchange rates for the date 1/1/2004. The table has columns for Turkish (TL), Sterling (£), Dollar (\$), and Euro (€). Below the table is a date field set to 1/1/2004. A legend at the bottom identifies the currencies: Turkish (light blue), Sterling (dark blue), Dollar (light green), Euro (dark green), and Date (white).

TL Turkish	£ Sterling	\$ Dollar	€ Euro
1200000	1400000	1700000	2000000

Figure 2.14 Adding the Exchange Rates (Selling Rates)



## b. Foreign Currency Rates

Here in the following form you can add the foreign exchange rates

NEAR EAST UNIVERSITY.  
DEPARTMENT OF COMPUTER ENGINEERING  
DEVELOPMENT DELPHI PROGRAM FOR EXCHANGE AND CHANGE SYSTEM

ADD TO EXCHANGE AND CHANGE SYSTEM

CHANGE SYSTEM: EXCHANGE SYSTEM

TL Turkish	(£) Sterling	(\$ ) Dollar	(€) Euro
1430000	3400000	1400000	3700000

Date: 1/1/2004

Logo: A circular logo with a stylized 'N' and 'E' and the text 'NEAR EAST UNIVERSITY' around it.

Figure 2.15 Adding the Exchange Rates (Foreign Currency Rates)

## 2.7 Database Desktop

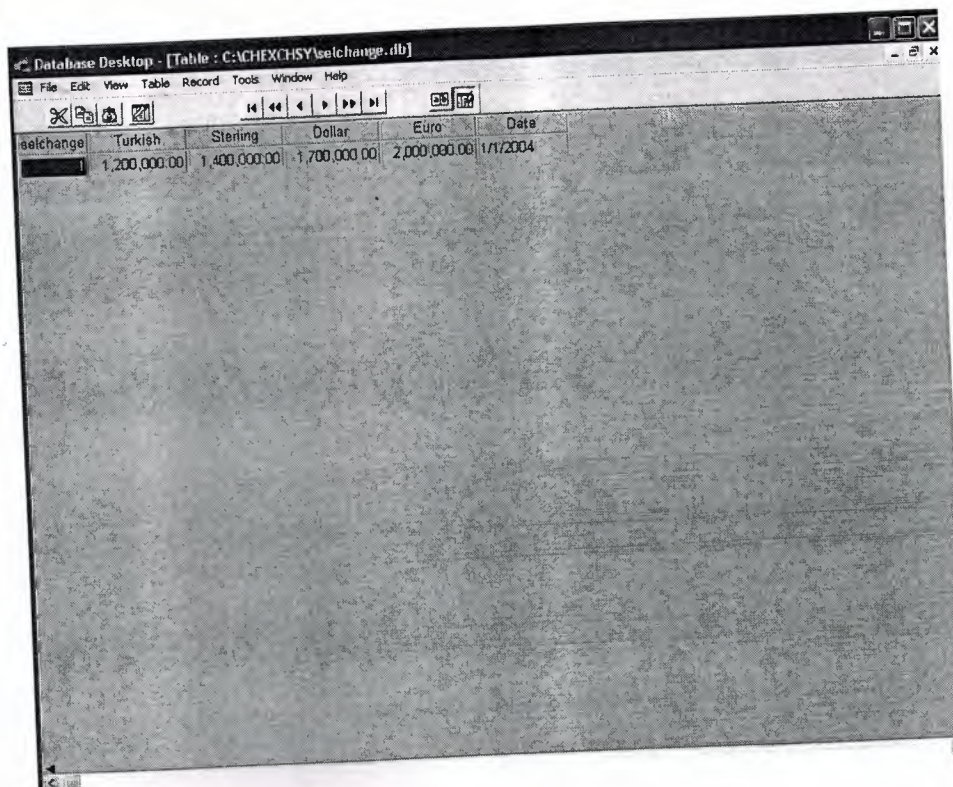


Figure 2.16 Database Desktop

Database Desktop is a database tool where you can create or restructure database tables, or browse and edit their data. You can work with tables in Paradox, dBase, and SQL formats.



## DATA FLOW DIAGRAM (DFD)

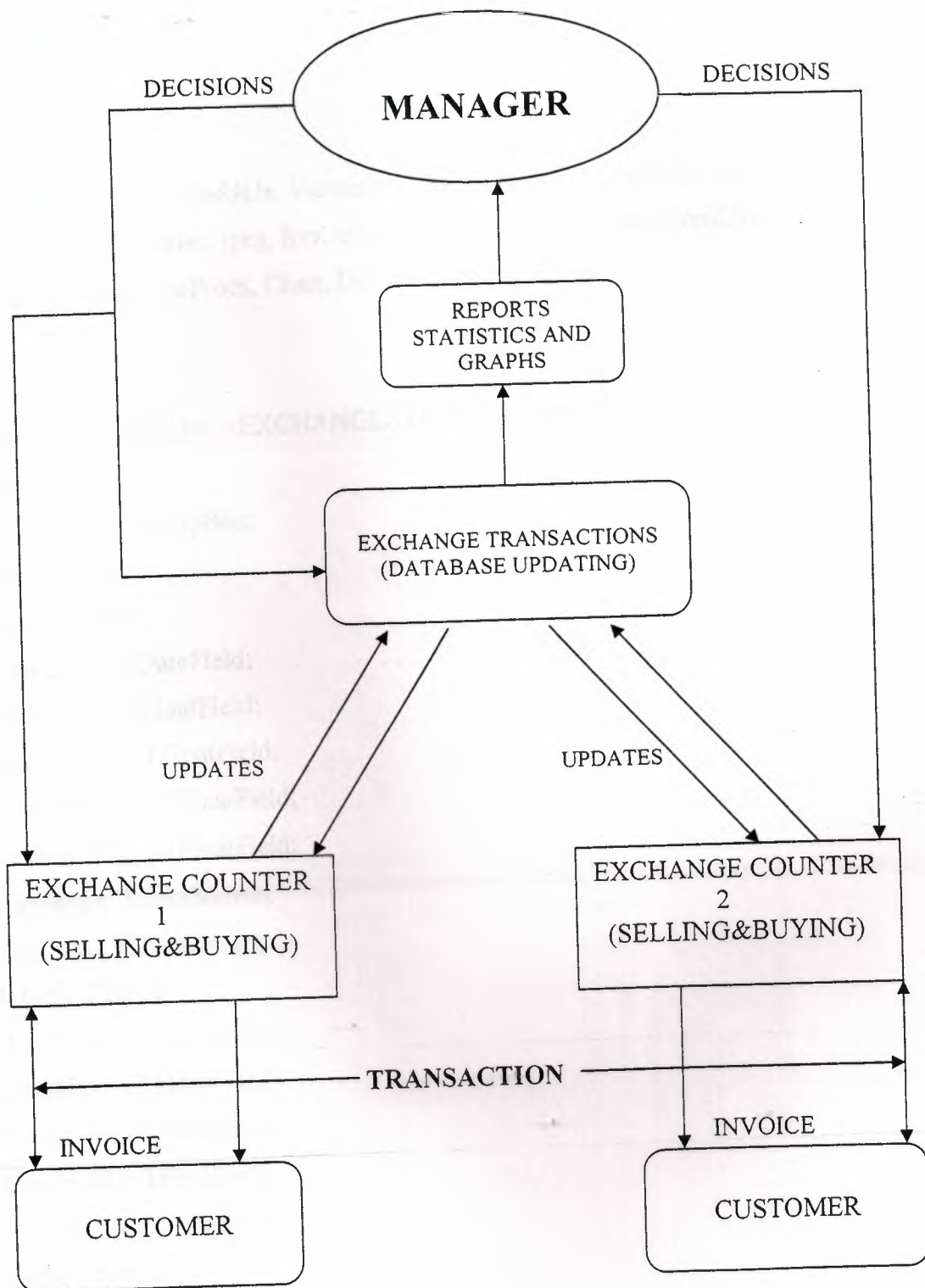


Figure 3.1 Data Flow Diagram

## 3.2 MAIN MENU

unit CHEXSU;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, jpeg, ExtCtrls, DBCtrls, StdCtrls, Mask, ComCtrls,  
TeEngine, Series, TeeProcs, Chart, DbChart, Grids, DBGrids;

type

TMAINMENUCHANGEEXCHANGESYSTEM = class(TForm)

Panel1: TPanel;

GroupBox1: TGroupBox;

Image1: TImage;

Table1: TTable;

Table1Date: TDateField;

Table1Euro: TFloatField;

Table1Dollar: TFloatField;

Table1Sterling: TFloatField;

Table1Turkish: TFloatField;

buychange: TDataSource;

selchange: TDataSource;

Table2: TTable;

DateField1: TDateField;

FloatField1: TFloatField;

FloatField2: TFloatField;

FloatField3: TFloatField;

FloatField4: TFloatField;

dollar: TDataSource;

Table3: TTable;

Table3Dollar: TFloatField;

Table3Rate: TFloatField;



Table3Date: TDateField;  
sterlaing: TDataSource;  
Table4: TTable;  
euro: TDataSource;  
Table5: TTable;  
Table9: TTable;  
exdollarE: TDataSource;  
Table10: TTable;  
exeuroD: TDataSource;  
Table11: TTable;  
Table12: TTable;  
exeuroS: TDataSource;  
exdollarS: TDataSource;  
Table14: TTable;  
exsterlingD: TDataSource;  
exsterlingE: TDataSource;  
Table15: TTable;  
Table17: TTable;  
Table18: TTable;  
Table6: TTable;  
Table7: TTable;  
Table8: TTable;  
Panel2: TPanel;  
GroupBox32: TGroupBox;  
Panel3: TPanel;  
PageControl1: TPageControl;  
TabSheet1: TTabSheet;  
Image2: TImage;  
TabControl1: TTabControl;  
Panel6: TPanel;  
MonthCalendar1: TMonthCalendar;  
Button1: TButton;  
Button3: TButton;  
Button4: TButton;

Button5: TButton;  
TabSheet2: TTabSheet;  
PageControl3: TPageControl;  
TabSheet11: TTabSheet;  
GroupBox6: TGroupBox;  
Label16: TLabel;  
Label17: TLabel;  
Label18: TLabel;  
Label19: TLabel;  
Label20: TLabel;  
DBEdit12: TDBEdit;  
DBEdit13: TDBEdit;  
DBEdit14: TDBEdit;  
DBEdit15: TDBEdit;  
DBEdit16: TDBEdit;  
DBNavigator2: TDBNavigator;  
GroupBox8: TGroupBox;  
Label26: TLabel;  
GroupBox9: TGroupBox;  
Label27: TLabel;  
Label28: TLabel;  
Label29: TLabel;  
Label30: TLabel;  
DBEdit25: TDBEdit;  
DBEdit22: TDBEdit;  
DBEdit23: TDBEdit;  
DBEdit24: TDBEdit;  
DBNavigator5: TDBNavigator;  
TabSheet12: TTabSheet;  
GroupBox5: TGroupBox;  
Label11: TLabel;  
Label12: TLabel;  
Label13: TLabel;  
Label14: TLabel;

Label15: TLabel;  
DBEdit7: TDBEdit;  
DBEdit8: TDBEdit;  
DBEdit9: TDBEdit;  
DBEdit10: TDBEdit;  
DBEdit11: TDBEdit;  
DBNavigator4: TDBNavigator;  
GroupBox10: TGroupBox;  
Label31: TLabel;  
GroupBox11: TGroupBox;  
Label32: TLabel;  
Label33: TLabel;  
Label34: TLabel;  
Label35: TLabel;  
DBEdit26: TDBEdit;  
DBEdit28: TDBEdit;  
DBEdit29: TDBEdit;  
DBEdit27: TDBEdit;  
DBNavigator6: TDBNavigator;  
TabSheet13: TTabSheet;  
GroupBox7: TGroupBox;  
Label21: TLabel;  
Label22: TLabel;  
Label23: TLabel;  
Label24: TLabel;  
Label25: TLabel;  
DBEdit17: TDBEdit;  
DBEdit18: TDBEdit;  
DBEdit19: TDBEdit;  
DBEdit20: TDBEdit;  
DBEdit21: TDBEdit;  
DBNavigator3: TDBNavigator;  
GroupBox12: TGroupBox;  
Label36: TLabel;



GroupBox13: TGroupBox;  
Label37: TLabel;  
Label38: TLabel;  
Label39: TLabel;  
Label40: TLabel;  
DBEdit30: TDBEdit;  
DBEdit31: TDBEdit;  
DBEdit32: TDBEdit;  
DBEdit33: TDBEdit;  
DBNavigator7: TDBNavigator;  
TabSheet3: TTabSheet;  
PageControl4: TPageControl;  
TabSheet14: TTabSheet;  
DBGrid1: TDBGrid;  
DBChart1: TDBChart;  
Series1: TLineSeries;  
Series2: TLineSeries;  
Series3: TLineSeries;  
TabSheet15: TTabSheet;  
DBGrid2: TDBGrid;  
DBChart2: TDBChart;  
LineSeries1: TLineSeries;  
LineSeries2: TLineSeries;  
LineSeries3: TLineSeries;  
TabSheet16: TTabSheet;  
DBGrid3: TDBGrid;  
DBChart3: TDBChart;  
LineSeries4: TLineSeries;  
LineSeries5: TLineSeries;  
LineSeries6: TLineSeries;  
TabSheet4: TTabSheet;  
Label71: TLabel;  
PageControl5: TPageControl;  
TabSheet17: TTabSheet;

GroupBox14: TGroupBox;  
Label41: TLabel;  
Label42: TLabel;  
Label43: TLabel;  
Label44: TLabel;  
Label45: TLabel;  
Label77: TLabel;  
DBEdit34: TDBEdit;  
DBEdit35: TDBEdit;  
DBEdit36: TDBEdit;  
DBEdit37: TDBEdit;  
DBEdit38: TDBEdit;  
DBNavigator8: TDBNavigator;  
GroupBox20: TGroupBox;  
Label78: TLabel;  
Label79: TLabel;  
Label80: TLabel;  
Label81: TLabel;  
Label82: TLabel;  
Label113: TLabel;  
DBEdit64: TDBEdit;  
DBEdit65: TDBEdit;  
DBEdit66: TDBEdit;  
DBEdit67: TDBEdit;  
DBEdit68: TDBEdit;  
DBNavigator14: TDBNavigator;  
GroupBox27: TGroupBox;  
Label120: TLabel;  
Label121: TLabel;  
Label122: TLabel;  
Label123: TLabel;  
Label124: TLabel;  
Label125: TLabel;  
DBEdit97: TDBEdit;

DBEdit98: TDBEdit;  
DBEdit99: TDBEdit;  
DBNavigator21: TDBNavigator;  
Edit89: TEdit;  
Edit90: TEdit;  
TabSheet18: TTabSheet;  
GroupBox18: TGroupBox;  
Label61: TLabel;  
Label62: TLabel;  
Label63: TLabel;  
Label64: TLabel;  
Label65: TLabel;  
Label76: TLabel;  
DBEdit54: TDBEdit;  
DBEdit55: TDBEdit;  
DBEdit56: TDBEdit;  
DBEdit57: TDBEdit;  
DBEdit58: TDBEdit;  
DBNavigator12: TDBNavigator;  
GroupBox21: TGroupBox;  
Label83: TLabel;  
Label84: TLabel;  
Label85: TLabel;  
Label86: TLabel;  
Label87: TLabel;  
Label109: TLabel;  
DBEdit69: TDBEdit;  
DBEdit70: TDBEdit;  
DBEdit71: TDBEdit;  
DBEdit72: TDBEdit;  
DBEdit73: TDBEdit;  
DBNavigator15: TDBNavigator;  
GroupBox28: TGroupBox;  
Label126: TLabel;



Label127: TLabel;  
Label128: TLabel;  
Label129: TLabel;  
Label130: TLabel;  
Label131: TLabel;  
DBNavigator22: TDBNavigator;  
DBEdit100: TDBEdit;  
DBEdit101: TDBEdit;  
DBEdit102: TDBEdit;  
Edit3: TEdit;  
Edit5: TEdit;  
TabSheet19: TTabSheet;  
GroupBox15: TGroupBox;  
Label46: TLabel;  
Label47: TLabel;  
Label48: TLabel;  
Label49: TLabel;  
Label50: TLabel;  
Label75: TLabel;  
DBEdit39: TDBEdit;  
DBEdit40: TDBEdit;  
DBEdit41: TDBEdit;  
DBEdit42: TDBEdit;  
DBEdit43: TDBEdit;  
DBNavigator9: TDBNavigator;  
GroupBox22: TGroupBox;  
Label88: TLabel;  
Label89: TLabel;  
Label90: TLabel;  
Label91: TLabel;  
Label92: TLabel;  
Label110: TLabel;  
DBEdit74: TDBEdit;  
DBEdit75: TDBEdit;

DBEdit76: TDBEdit;  
DBEdit77: TDBEdit;  
DBEdit78: TDBEdit;  
DBNavigator16: TDBNavigator;  
GroupBox29: TGroupBox;  
Label132: TLabel;  
Label133: TLabel;  
Label134: TLabel;  
Label135: TLabel;  
Label136: TLabel;  
Label137: TLabel;  
DBEdit103: TDBEdit;  
Edit7: TEdit;  
DBEdit104: TDBEdit;  
DBEdit105: TDBEdit;  
DBNavigator23: TDBNavigator;  
Edit93: TEdit;  
TabSheet20: TTabSheet;  
GroupBox17: TGroupBox;  
Label56: TLabel;  
Label57: TLabel;  
Label58: TLabel;  
Label59: TLabel;  
Label60: TLabel;  
Label74: TLabel;  
DBEdit49: TDBEdit;  
DBEdit50: TDBEdit;  
DBEdit51: TDBEdit;  
DBEdit52: TDBEdit;  
DBEdit53: TDBEdit;  
DBNavigator11: TDBNavigator;  
GroupBox23: TGroupBox;  
Label93: TLabel;  
Label94: TLabel;

Label95: TLabel;  
Label96: TLabel;  
Label97: TLabel;  
Label111: TLabel;  
DBEdit79: TDBEdit;  
DBEdit80: TDBEdit;  
DBEdit81: TDBEdit;  
DBEdit82: TDBEdit;  
DBEdit83: TDBEdit;  
DBNavigator17: TDBNavigator;  
GroupBox30: TGroupBox;  
Label138: TLabel;  
Label139: TLabel;  
Label140: TLabel;  
Label141: TLabel;  
Label142: TLabel;  
Label143: TLabel;  
DBEdit106: TDBEdit;  
Edit9: TEdit;  
DBEdit107: TDBEdit;  
DBEdit108: TDBEdit;  
DBNavigator24: TDBNavigator;  
Edit92: TEdit;  
TabSheet21: TTabSheet;  
GroupBox16: TGroupBox;  
Label51: TLabel;  
Label52: TLabel;  
Label53: TLabel;  
Label54: TLabel;  
Label55: TLabel;  
Label73: TLabel;  
DBEdit44: TDBEdit;  
DBEdit45: TDBEdit;  
DBEdit46: TDBEdit;



DBEdit47: TDBEdit;  
DBEdit48: TDBEdit;  
DBNavigator10: TDBNavigator;  
GroupBox24: TGroupBox;  
Label98: TLabel;  
Label99: TLabel;  
Label100: TLabel;  
Label101: TLabel;  
Label102: TLabel;  
Label112: TLabel;  
DBEdit84: TDBEdit;  
DBEdit85: TDBEdit;  
DBEdit86: TDBEdit;  
DBEdit87: TDBEdit;  
DBEdit88: TDBEdit;  
DBNavigator18: TDBNavigator;  
GroupBox31: TGroupBox;  
Label144: TLabel;  
Label145: TLabel;  
Label146: TLabel;  
Label147: TLabel;  
Label148: TLabel;  
Label149: TLabel;  
DBEdit109: TDBEdit;  
DBEdit112: TDBEdit;  
DBEdit113: TDBEdit;  
DBNavigator25: TDBNavigator;  
Edit110: TEdit;  
Edit111: TEdit;  
TabSheet22: TTabSheet;  
GroupBox19: TGroupBox;  
Label66: TLabel;  
Label67: TLabel;  
Label68: TLabel;

Label69: TLabel;  
Label70: TLabel;  
Label72: TLabel;  
DBEdit59: TDBEdit;  
DBEdit60: TDBEdit;  
DBEdit61: TDBEdit;  
DBEdit62: TDBEdit;  
DBEdit63: TDBEdit;  
DBNavigator13: TDBNavigator;  
GroupBox25: TGroupBox;  
Label103: TLabel;  
Label104: TLabel;  
Label105: TLabel;  
Label106: TLabel;  
Label107: TLabel;  
Label108: TLabel;  
DBEdit89: TDBEdit;  
DBEdit90: TDBEdit;  
DBEdit91: TDBEdit;  
DBEdit92: TDBEdit;  
DBEdit93: TDBEdit;  
DBNavigator19: TDBNavigator;  
GroupBox26: TGroupBox;  
Label118: TLabel;  
Label119: TLabel;  
Label114: TLabel;  
Label115: TLabel;  
Label116: TLabel;  
Label117: TLabel;  
DBEdit94: TDBEdit;  
Edit1: TEdit;  
DBEdit95: TDBEdit;  
DBEdit96: TDBEdit;  
DBNavigator20: TDBNavigator;

Edit66: TEdit;  
TabSheet6: TTabSheet;  
DBGrid5: TDBGrid;  
DBChart4: TDBChart;  
BarSeries1: TBarSeries;  
BarSeries2: TBarSeries;  
BarSeries3: TBarSeries;  
Series4: TBarSeries;  
Series5: TBarSeries;  
TabSheet7: TTabSheet;  
DBGrid4: TDBGrid;  
DBChart5: TDBChart;  
BarSeries4: TBarSeries;  
BarSeries5: TBarSeries;  
BarSeries6: TBarSeries;  
BarSeries7: TBarSeries;  
BarSeries8: TBarSeries;  
TabSheet8: TTabSheet;  
GroupBox2: TGroupBox;  
PageControl2: TPageControl;  
TabSheet9: TTabSheet;  
GroupBox3: TGroupBox;  
Label5: TLabel;  
Label4: TLabel;  
Label3: TLabel;  
Label2: TLabel;  
Label6: TLabel;  
EditTurkish: TDBEdit;  
EditSterling: TDBEdit;  
EditDollar: TDBEdit;  
EditEuro: TDBEdit;  
DBEdit6: TDBEdit;  
DBNavigator: TDBNavigator;  
TabSheet10: TTabSheet;



```

GroupBox4: TGroupBox;
Label1: TLabel;
Label7: TLabel;
Label8: TLabel;
Label9: TLabel;
Label10: TLabel;
DBEdit1: TDBEdit;
DBEdit2: TDBEdit;
DBEdit3: TDBEdit;
DBEdit4: TDBEdit;
DBEdit5: TDBEdit;
DBNavigator1: TDBNavigator;
Button2: TButton;
Button6: TButton;
Image3: TImage;
Image4: TImage;
Image5: TImage;
Image6: TImage;
Image7: TImage;
Image8: TImage;
Image9: TImage;
Image10: TImage;
Image11: TImage;
Image12: TImage;
procedure DBEdit23Change(Sender: TObject);
procedure DBEdit29Change(Sender: TObject);
procedure DBEdit32Change(Sender: TObject);
procedure Edit111Change(Sender: TObject);
procedure Edit90Change(Sender: TObject);
procedure Edit5Change(Sender: TObject);
procedure Edit93Change(Sender: TObject);
procedure Edit92Change(Sender: TObject);
procedure Edit66Change(Sender: TObject);
procedure Button1Click(Sender: TObject);

```

```

procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
procedure Button6Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure FormCreate(Sender: TObject);

```

```
private
```

```
    { Private declarations }
```

```
public
```

```
    { Public declarations }
```

```
end;
```

```
var
```

```
    MAINMENUCHANGEEXCHANGESYSTEM:
```

```
    TMAINMENUCHANGEEXCHANGESYSTEM;
```

```
implementation
```

```
uses REPORT, MENU, MENU2, REPORT4, REPORT5;
```

```
{ $R *.dfm }
```

```

procedure TMAINMENUCHANGEEXCHANGESYSTEM.DBEdit23Change(Sender:
TObject);

```

```
    var
```

```
    a,b,c:Currency;
```

```
begin
```

```
if (DBEdit22.gettextlen=0) or (DBEdit23.gettextlen=0) then
```

```
begin
```

```
    showmessage('please enter value on edits');
```

```
    exit;
```

```
end;
```

```
begin
```

```

a:=strtoFloat(DBEdit22.text);
b:=strtoFloat(DBEdit23.text);
c:=a*b;
DBEdit24.text:=floattostr(c);
end;
end;

```

```

procedure TMAINMENUCHANGEEXCHANGESYSTEM.DBEdit29Change(Sender:
TObject);

```

```

    var
    a,b,c:Currency;
begin
if (DBEdit28.gettextlen=0) or (DBEdit29.gettextlen=0) then
    begin
        showmessage('please enter value on edits');
        exit;
    end;
    begin
a:=strtoFloat(DBEdit28.text);
b:=strtoFloat(DBEdit29.text);
c:=a*b;
DBEdit27.text:=floattostr(c);
end;
end;

```

```

procedure TMAINMENUCHANGEEXCHANGESYSTEM.DBEdit32Change(Sender:
TObject);

```

```

    var
    a,b,c:Currency;
begin
if (DBEdit31.gettextlen=0) or (DBEdit32.gettextlen=0) then
    begin
        showmessage('please enter value on edits');

```



```

    exit;
end;

begin
a:=strtoFloat(DBEdit31.text);
b:=strtoFloat(DBEdit32.text);
c:=a*b;
DBEdit33.text:=floattostr(c);
end;
end;

```

```

procedure TMAINMENUCHANGEEXCHANGESYSTEM.Edit111Change(Sender:
TObject);
    VAR
a,b,c,d:CURRENCY;
begin
    if (DBEdit109.gettextlen=0)OR (Edit110.gettextlen=0) or (Edit111.gettextlen=0) then
        begin
            showmessage('please enter value on edits');
            exit;
        end;
        begin
a:=strtoFloat(DBEdit109.text);
b:=strtoFloat(Edit110.text);
d:=strtoFloat(Edit111.text);
c:= (a * b)/d;

DBEdit112.text:=floattostr(c);
end;
end;

```

```

procedure TMAINMENUCHANGEEXCHANGESYSTEM.Edit90Change(Sender:
TObject);

```

```

VAR
a,b,c,d:CURRENCY;
begin
  if (DBEdit97.gettextlen=0)OR (Edit89.gettextlen=0) or (Edit90.gettextlen=0) then
    begin
      showmessage('please enter value on edits');
      exit;
      end;
      begin
a:=strtoFloat(DBEdit97.text);
b:=strtoFloat(Edit89.text);
d:=strtoFloat(Edit90.text);
c:= (a * b)/d;

DBEdit98.text:=floattostr(c);
end;
end;

//procedure TMAINMENUCHANGEEXCHANGESYSTEM.Edit4Change(Sender:
TObject);

procedure TMAINMENUCHANGEEXCHANGESYSTEM.Edit5Change(Sender:
TObject);
  VAR
a,b,c,d:CURRENCY;
begin
  if (DBEdit100.gettextlen=0)OR (Edit3.gettextlen=0) or (Edit5.gettextlen=0) then
    begin
      showmessage('please enter value on edits');
      exit;
      end;
      begin
a:=strtoFloat(DBEdit100.text);
b:=strtoFloat(Edit3.text);

```

```
d:=strtoFloat(Edit5.text);
```

```
c:= (a * b)/d;
```

```
DBEdit101.text:=Floattostr(c);
```

```
end;
```

```
end;
```

```
procedure TMAINMENUCHANGEEXCHANGESYSTEM.Edit93Change(Sender:  
TObject);
```

```
VAR
```

```
a,b,c,d:CURRENCY;
```

```
begin
```

```
if (DBEdit103.gettextlen=0)OR (Edit7.gettextlen=0) or (Edit93.gettextlen=0) then
```

```
begin
```

```
showmessage('please enter value on edits');
```

```
exit;
```

```
end;
```

```
begin
```

```
a:=strtoFloat(DBEdit103.text);
```

```
b:=strtoFloat(Edit7.text);
```

```
d:=strtoFloat(Edit93.text);
```

```
c:= (a * b)/d;
```

```
DBEdit104.text:=Floattostr(c);
```

```
end;
```

```
end;
```

```
procedure TMAINMENUCHANGEEXCHANGESYSTEM.Edit92Change(Sender:  
TObject);
```

```
VAR
```

```
a,b,c,d:CURRENCY;
```

```
begin
```

```
if (DBEdit106.gettextlen=0)OR (Edit9.gettextlen=0) or (Edit92.gettextlen=0) then
```



```

begin
  showmessage('please enter value on edits');
exit;
end;

begin
a:=strtoFloat(DBEdit106.text);
b:=strtoFloat(Edit9.text);
d:=strtoFloat(Edit92.text);
c:=(a * b)/d;
DBEdit107.text:=Floattostr(c);
end;
end;

```

```

procedure TMAINMENUCHANGEEXCHANGESYSTEM.Edit66Change(Sender:
TObject);
VAR
a,b,c,d:CURRENCY;
begin
  if (DBEdit94.gettextlen=0)OR (Edit1.gettextlen=0) or (Edit66.gettextlen=0) then
  begin
    showmessage('please enter value on edits');
    exit;
  end;
  begin
a:=strtoFloat(DBEdit94.text);
b:=strtoFloat(Edit1.text);
d:=strtoFloat(Edit66.text);
c:=(a * b)/d;
DBEdit95.text:=Floattostr(c);
end;
end;

```

```
procedure TMAINMENUCHANGEEXCHANGESYSTEM.Button1Click(Sender:
TObject);
begin
//REPORTDOLLAR.QuickRep1.Preview;
SELCHANGESYSTEMPREVIEW.SHOW;
end;
```

```
procedure TMAINMENUCHANGEEXCHANGESYSTEM.Button3Click(Sender:
TObject);
begin
BUYCHANGESYSTEMPREVIEW.SHOW;
end;
```

```
procedure TMAINMENUCHANGEEXCHANGESYSTEM.Button4Click(Sender:
TObject);
begin
BUYCHANGESYSTEM.QuickRep1.Preview;
end;
```

```
procedure TMAINMENUCHANGEEXCHANGESYSTEM.Button5Click(Sender:
TObject);
begin
SELLEXCHANGESYSTEM.QuickRep1.Preview;
end;
```

```
procedure TMAINMENUCHANGEEXCHANGESYSTEM.Button6Click(Sender:
TObject);
begin
SELLEXCHANGESYSTEM.QuickRep1.Print;
end;
```

```
procedure TMAINMENUCHANGEEXCHANGESYSTEM.Button2Click(Sender:
TObject);
begin
```

```
BUYCHANGESYSTEM.QuickRep1.Print;  
end;
```

```
procedure TMAINMENUCHANGEEXCHANGESYSTEM.FormCreate(Sender:  
TObject);  
begin  
  
end;  
  
end.
```

### 3.3 BUY CHANGE SYSTEM PREVIEW

```
unit MENU2;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, ExtCtrls;
```

```
type
```

```
TBUYCHANGESYSTEMPREVIEW = class(TForm)
```

```
  Panel1: TPanel;
```

```
  GroupBox1: TGroupBox;
```

```
  Button3: TButton;
```

```
  Button2: TButton;
```

```
  Button1: TButton;
```

```
  Panel2: TPanel;
```

```
  GroupBox2: TGroupBox;
```

```
  Panel3: TPanel;
```

```
  GroupBox3: TGroupBox;
```



```

Button4: TButton;
Button5: TButton;
Button6: TButton;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
procedure Button6Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    BUYCHANGESYSTEMPREVIEW: TBUYCHANGESYSTEMPREVIEW;

implementation

uses REPORT5DU, REPORT5DS, REPORT5E, REPORT5ES, REPORT5SE,
    REPORT5SD;

{$R *.dfm}
procedure TBUYCHANGESYSTEMPREVIEW.Button1Click(Sender: TObject);
begin
    EXCHANGSYSTEMDOLLARTOEURO.QuickRep1.Preview;
end;

procedure TBUYCHANGESYSTEMPREVIEW.Button2Click(Sender: TObject);
begin
    EXCHANGSYSTEMDOLLARSTERLING.QuickRep1.Preview;
end;

```

```
procedure TBUYCHANGESYSTEMPREVIEW.Button3Click(Sender: TObject);  
begin  
EXCHANGEEUROSYSTEM.QuickRep1.Preview;  
end;
```

```
procedure TBUYCHANGESYSTEMPREVIEW.Button4Click(Sender: TObject);  
begin  
EXCHANGEEUROS.QuickRep1.Preview;  
end;
```

```
procedure TBUYCHANGESYSTEMPREVIEW.Button5Click(Sender: TObject);  
begin  
EXCHANGESYSTEMSTERLINGSTERLING.QuickRep1.Preview;  
end;
```

```
procedure TBUYCHANGESYSTEMPREVIEW.Button6Click(Sender: TObject);  
begin  
EXCHANGESYSTEMSTERLINGDOLLAR.QuickRep1.Preview;  
end;
```

```
procedure TBUYCHANGESYSTEMPREVIEW.FormCreate(Sender: TObject);  
begin  
  
end;  
  
end.
```

### 3.4 EXCHANGE SYSTEM STERLING

```
unit REPORT5SE;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;
```

```
type
```

```
TEXCHANGESYSTEMSTERLINGSTERLING = class(TForm)
```

```
QuickRep1: TQuickRep;
```

```
PageFooterBand1: TQRBand;
```

```
QRExpr1: TQRExpr;
```

```
ColumnHeaderBand1: TQRBand;
```

```
QRLabel1: TQRLabel;
```

```
QRLabel2: TQRLabel;
```

```
QRLabel3: TQRLabel;
```

```
DetailBand1: TQRBand;
```

```
QRExpr2: TQRExpr;
```

```
QRExpr3: TQRExpr;
```

```
QRExpr4: TQRExpr;
```

```
QRLabel4: TQRLabel;
```

```
Table1: TTable;
```

```
procedure FormCreate(Sender: TObject);
```

```
private
```

```
{ Private declarations }
```

```
public
```

```
{ Public declarations }
```

```
end;
```

```
var
```



### 3.5 CHANGE SYSTEM STERLING

implementation

{ \$R \*.dfm }

procedure TEXCHANGESYSTEMSTERLINGSTERLING.FormCreate(Sender:

TObject);

begin

end;

end.

### 3.6 EXCHANGE SYSTEM STERLING DOLLAR

unit REPORT5SD;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;

type

TEXCHANGESYSTEMSTERLINGDOLLAR = class(TForm)

QuickRep1: TQuickRep;

PageFooterBand1: TQRBand;

QRExpr1: TQRExpr;

ColumnHeaderBand1: TQRBand;

QRLabel1: TQRLabel;

QRLabel2: TQRLabel;

```

QRLabel3: TQRLabel;
DetailBand1: TQRBand;
QRExpr2: TQRExpr;
QRExpr3: TQRExpr;
QRExpr4: TQRExpr;
QRLabel4: TQRLabel;
Table1: TTable;
procedure FormCreate(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  EXCHANGESYSTEMSTERLINGDOLLAR:
  TEXCHANGESYSTEMSTERLINGDOLLAR;
implementation

{$R *.dfm}
procedure TEXCHANGESYSTEMSTERLINGDOLLAR.FormCreate(Sender:
TObject);
begin

end;

end.

```

### 3.7 EXCHANGE SYSTEM DOLLAR EURO



unit REPORT5ED;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;

type

TEXCHANGESYSTEMDOLLAREURO = class(TForm)

QuickRep1: TQuickRep;

PageFooterBand1: TQRBand;

QRExpr1: TQRExpr;

ColumnHeaderBand1: TQRBand;

QRLabel1: TQRLabel;

QRLabel2: TQRLabel;

QRLabel3: TQRLabel;

DetailBand1: TQRBand;

QRExpr2: TQRExpr;

QRExpr3: TQRExpr;

QRExpr4: TQRExpr;

QRLabel4: TQRLabel;

Table1: TTable;

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var



### 3.8 CHANGE SYSTEM DOLLAR EURO

TEXCHANGESYSTEMDOLLAREURO;

implementation

{ \$R \*.dfm }

procedure TEXCHANGESYSTEMDOLLAREURO.FormCreate(Sender: TObject);

begin

end;

end.

### 3.9 EXCHANGE EURO SYSTEM

unit REPORT5E;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;

type

TEXCHANGEEUROSYSTEM = class(TForm)

QuickRep1: TQuickRep;

PageFooterBand1: TQRBand;

QRExpr1: TQRExpr;

ColumnHeaderBand1: TQRBand;

QRLabel1: TQRLabel;

```

QRLabel2: TQRLabel;
QRLabel3: TQRLabel;
DetailBand1: TQRBand;
QRExpr2: TQRExpr;
QRExpr3: TQRExpr;
QRExpr4: TQRExpr;
QRLabel4: TQRLabel;
Table1: TTable;
procedure FormCreate(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    EXCHANGEEUROSYSTEM: TEXCHANGEEUROSYSTEM;

implementation

    {$R *.dfm}

    procedure TEXCHANGEEUROSYSTEM.FormCreate(Sender: TObject);
    begin
        end;

    end.

```

### 3.10 EXCHANGE SYSTEM DOLLAR STERLING

unit REPORT5DS;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;

type

TEXCHANGESYSTEMDOLLARSTERLING = class(TForm)

QuickRep1: TQuickRep;

PageFooterBand1: TQRBand;

QRExpr1: TQRExpr;

ColumnHeaderBand1: TQRBand;

QRLabel1: TQRLabel;

QRLabel2: TQRLabel;

QRLabel3: TQRLabel;

DetailBand1: TQRBand;

QRExpr2: TQRExpr;

QRExpr3: TQRExpr;

QRExpr4: TQRExpr;

QRLabel4: TQRLabel;

Table1: TTable;

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var



### 3.11 CHANGE SYSTEM DOLLAR STERLING

TEXCHANGESYSTEMDOLLARSTERLING;

implementation

{ \$R \*.dfm }

procedure TEXCHANGESYSTEMDOLLARSTERLING.FormCreate(Sender:

TObject);

begin

end;

end.

### 3.12 SELL EXCHANGE SYSTEM

unit REPORT5;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;

type

TSELLEXCHANGESYSTEM = class(TForm)

QuickRep1: TQuickRep;

PageFooterBand1: TQRBand;

QRExpr1: TQRExpr;

ColumnHeaderBand1: TQRBand;

```

QRLabel1: TQRLabel;
QRLabel2: TQRLabel;
QRLabel3: TQRLabel;
QRLabel4: TQRLabel;
QRLabel5: TQRLabel;
DetailBand1: TQRBand;
QRExpr2: TQRExpr;
QRExpr3: TQRExpr;
QRExpr4: TQRExpr;
QRExpr5: TQRExpr;
QRExpr6: TQRExpr;
QRLabel6: TQRLabel;
Table1: TTable;
procedure FormCreate(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  SELLEXCHANGESYSTEM: TSELLEXCHANGESYSTEM;

implementation

  {$R *.dfm}

  procedure TSELLEXCHANGESYSTEM.FormCreate(Sender: TObject);
  begin

  end;

end.

```

### 3.13 EXCHANGE EURO TO STERLING

unit REPORT5ES;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;

type

TEXCHANGEEUROS = class(TForm)

QuickRep1: TQuickRep;

PageFooterBand1: TQRBand;

QRExpr1: TQRExpr;

ColumnHeaderBand1: TQRBand;

QRLabel1: TQRLabel;

QRLabel2: TQRLabel;

QRLabel3: TQRLabel;

DetailBand1: TQRBand;

QRExpr3: TQRExpr;

QRExpr4: TQRExpr;

QRLabel4: TQRLabel;

Table1: TTable;

QRExpr2: TQRExpr;

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var



**NEAR EAST UNIVERSITY**



**Faculty of Engineering**

**Department of Computer Engineering**

**DEVELOPMENT DELPHI PROGRAM  
FOR EXCHANGE AND CHANGE SYSTEM**

**Graduation Project  
COM- 400**

**Student: Ismail Mekki**

**Supervisor: Mr. Ümit ILHAN**

**Nicosia 2003 - 2004**

## ACKNOWLEDGEMENTS



*"No one can deny the role of the university in upgrading the mentality of the student to be capable to cover the requirements of the working life. This role is concerning with qualifying the student up to the level that the society needs. However, this role cannot be accomplished unless there is a qualified leader and sophisticated coach.*

*Fortunately, Mr. Umit ILHAN was the main reason of my success in this project, and thus, he deserves my all thanks, gratitude, and my respect due to his support and wise advice.*

*I appreciate all the effort that he provided during the preparation of this project.*

*Therefore, firstly, I would like to dedicate this project my family because of their unlimited support during my life.*

*Secondly, I would like to dedicate it to my supervisor Mr. Umit ILHAN because of his wise supervision on this project and for his wide knowledge.*

*Finally, I appreciate all the effort aids of my friends during the preparation of this project."*

*With all due respect*

## ABSTRACT

This program is designed for the change and exchange markets where the individuals deal with the currencies and barter them. This system is based on BORLAND DELPHI 6 programming language. All the criterions of this system are taken according upon the request of the Near East Bank and all the features of this software can be adjusted according to the desire of the customer.

All the screens that will appear in the usage of this program will be illustrated in the coming chapters in details. The calculations and the mathematical operations that this program applies are explained in details in the appendix at the end of this report. This program is designed to find out the exact profit of the company by using graphical charts and by showing statues reports at any time. Three hard currencies are taken into consideration in this system and they are compared with the Turkish lira, which is the local currency. The values of these hard currencies will be taken from the stock markets that the government decides daily according o the daily economic level. The decision maker in NEB will determine the amount of the profit that he desires and the total revenue will be calculating by the software. A navigation bar is located in the bottom of each screen for adding, deleting, saving... etc.



## TABLE OF CONTENTS

<b>ACKNOWLEDGEMENTS</b>	<b>I</b>
<b>ABSTRACT</b>	<b>II</b>
<b>TABLE OF CONTENTS</b>	<b>III</b>
<b>LIST OF FIGURES</b>	<b>V</b>
<b>CHAPTER ONE: INTRODUCTION</b>	<b>1</b>
1.1. Overview	1
1.2. Economic Analysis of Floating Exchange Rate System	2
1.2 a Introduction	2
1.2 b Opportunities From Around World	2
1.3 Delphi Programming	5
1.4. Database Programming	7
1.5. Borland Database Engine (BDE)	8
1.6. Graphical Data-Aware Control	8
1.7. The clintDataSet Component	9
1.8 Classic BDE component	10
1.9 Tables and Queries	10
1.10 DBNavigator and Dataset Actions	11
1.11 Text-Based Data-Aware Control	12
1.12 Navigator a Dataset	12
 <b>CHAPTER TWO: THE DESCRIPTION OF THE SYSTEM</b>	 <b>14</b>
2.1 Main Menu Screen	14
2.2 Change System	18
2.3. Exchange curve	19
2.4. Exchange System	21
2.5. Exchange Bar Graphs	22

2.6. Adding the Exchange rates	23
2.7. Database Desktop	25
<b>CHAPTER THREE: DATA SOURCE</b>	26
3.1. Data Flow Diagram	26
3.2 . Main Menu	27
3.3. Buy Change System Preview	48
3.4. Exchange System Sterling	51
3.5. Change System Sterling	52
3.6. Exchange System Sterling Dollar	52
3.7. Exchange System Dollar Euro	54
3.8. Change System Dollar Euro	55
3.9. Exchange System Euro	55
3.10. Exchange System Dollar Sterling	57
3.11. Change System Dollar Sterling	58
3.12. Sell Exchange System	58
3.13. Exchange Euro to Sterling	60
3.14. Exchange Euro	61
3.15. Exchange System Dollar to Euro	61
3.16. Buy Change System preview	63
3.17. Buy Change System	66
3.18. Buy Change System	67
3.19. Change System Sterling	68
3.20. Change System	69
3.21. Report Dollar	71
3.22. Sell Change System Preview	72
<b>CONCLUSION</b>	75
<b>SYSTEM REQUIREMENTS</b>	75
<b>REFERENCES</b>	76

## LIST OF FIGURES

### CHAPTER II

#### CHANGE AND EXCHANGE SYSTEMS

2.1 Main Menu Screen	14
2.2 Buy change system preview	15
2.3 Exchange report	15
2.4 Exchange rates	16
2.5 Change System Preview	17
2.6 Exchange system Preview	17
2.7 Change System	18
2.8 Change Curve	19
2.9 Change Curve	20
2.10 Change Curve	20
2.11 Exchange System	21
2.12 Exchange Bar Graphs a	22
2.13 Exchange Bar Graphs b	23
2.14 Adding the Exchange Rates (Selling Rates)	23
2.15 Adding the Exchange Rates (Foreign Currency Rates)	24
2.16 Database Desktop	25

### CHAPTER III

#### DATA SOURCE

3.1 Data Flow Diagram	26
-----------------------	----



# INTRODUCTION

## 1.1 Overview

Many activities are done daily in the stock market. Most of these activities are concerning with the transactions and exchanging the goods and services. Since the market is the place that the buyers and sellers exchange goods and services, then the currencies are considered as a good that the people buy and sell within the market.

As it is known, the most currencies that are used widely are those currencies that are called the hard currencies such as US Dollar, UK sterling, and Euro. Therefore, as we are concerning with controlling the activities in the market by issuing the required software programs that include all features of controlling the business by means of scientific methods in order to be used in a proper and easy way.

Using any programming language to create certain software gives you the opportunity of gaining a good experience and it will promote your approaches of analyzing the aspects of any given project.

Therefore, any software needs an adequate and sufficient programming language that helps us to set up all the applications and all the functions that include the features of organizing and sustaining our business.

The role of a computer engineer requires an adequate ability of analyzing and dealing with all the aspects of any project in order to be capable to create and issue certain software that controls the business. Therefore, our role as an engineer is to not only deal with the hardware, but also deal with creating a software programs. Thus, the role of an engineer is to control the technical specifications of the firm and to know every single aspect that is necessary to measure the level of the success.

## **1.2 Economic Analysis of Floating Exchange Rate Systems**

### **1.2.a Introduction**

Associated foreign exchange is looking for experienced foreign exchange sales professionals! Successful candidates will be strongly motivated individuals who can identify and cultivate potential corporate clients with international payment needs. Strong communication skills, experience in the foreign exchange field and experience in relationship based selling are a must. Successful candidates will be responsible for opening, maintaining and growing accounts through relationship building, identifying customer needs and cross-selling appropriate services.

### **1.2 b Opportunities from Around the World**

Over the last three decades the foreign exchange market has become the world's largest financial market, with over \$1.5 trillion USD traded daily. The primary market for currencies is the 24-hour Interbank market. The Interbank market literally follows the sun around the world, moving from major banking centers of the United States to Australia and New Zealand to the Far East, to Europe and finally back to the United States. With the large minimum transaction sizes and often-stringent financial requirements, banks, hedge funds, major currency dealers and the occasional high net-worth individual speculator were the principal participants. These large traders were able to take advantage of the many benefits offered by the forex market vs. other markets including fantastic liquidity and the strong trending nature of the world's primary currency exchange rates.

The business section of any newspaper will have a table of spot exchange rates. These are the rates at which a person could have bought other currencies or foreign Exchange, such as the English Pound, French Franc, or the new European Euro. The Prices of foreign currencies can be determined in two major types of exchange rate Systems. In the United States, the dollar's exchange rates are determined by the Marketplace, i.e., by supply and demand. This type of system is called a floating Exchange rate system. In other countries, governments set the price of their currencies With respect to other countries. They then buy or sell foreign exchange at the prices They've set. This is called a fixed exchange rate system. The economic effects of these Two systems can be very different. However, in either system the underlying forces Influencing the value of a country's currency remain the same. Due to possible



confusion of being able to quote different currencies in terms of Each other, e.g., \$/£ or £/\$, we need to explicitly define an exchange rate. An **exchange Rate** is, therefore, the domestic cost of a unit of foreign exchange. For example, from the US perspective the price of the English Pound would be denominated as the number of US dollars per pound, or \$/£. As noted above, the exchange rate in a floating exchange rate system is Determined by market forces. Our definition of the exchange rate defines the market as The market for foreign exchange. In this market we have demanders and suppliers of Foreign currencies willing to pay and accept dollars in return for these currencies. We Will in turn discuss the demand and supply of foreign exchange. Foreign Exchange Demand The demand for foreign exchange is a derived demand. With the exception of currency Collectors, the demand for foreign exchange is due to people's desire to use it in the Purchase of foreign goods or financial assets. Foreign exchange demand is, therefore, Highly sensitive to changes in these desires.

In order to understand changes in the demand for foreign exchange, we will need to Discuss its underlying forces. These are the demand for foreign goods and services and the demand for foreign financial assets. The supply of foreign exchange has at its roots the same conceptual basis as Demand, only it is from the foreign perspective. Foreign currency is supplied to the Foreign exchange market when foreigners exchange their currency for dollars in order to Buy US goods or financial assets. Equivalently, the supply of foreign exchange is Nothing more than a mirror image of the foreign demand for US currency. Exchange is the mirror of the supply of dollars to the foreign exchange market. One question which might arise is which foreign exchange Market. New York, London, Frankfurt and Tokyo are Major financial centers with large foreign exchange markets. The answer as to which market is all of them. The first rule of business is to buy low and sell high. Should exchange rates be different across different financial centers, then the opportunity for arbitrage profits occurs. Currency dealers will buy low in one center and sell high in another, driving exchange rates into equality Across the different markets. For example, should the Swiss Franc be at a lower price (in terms of \$) in London and at a higher price in New York, then the dealers will increase the demand for the Swiss Franc in London, driving up its price, and increase its supply in New York, driving down its price there. This continues until the price is the same in both places. The major questions to be addressed are how exchange rates determined are and what the forces which influence them are. In Figure 1, the equilibrium exchange rate ( $e$ ) is the one where the quantity demanded is equal to the quantity supplied for



foreign exchange. As with most markets, the price changes in order to equilibrate the market. When quantity demanded exceeds quantity supplied, and then the exchange rate will rise. If the quantity supplied is greater than quantity demanded, the exchange rate falls. What does it mean when the exchange rate rises or falls? As we have defined the Exchange rate (\$/£), when the exchange rate rises, the value of the dollar decreases or depreciates. It now takes more dollars to buy an English pound than it did before the Change in the exchange rate. Fewer foreign goods can now be purchased for a given number of dollars. The reverse is also true. As the exchange rate falls, the dollar cost of foreign exchange falls, increasing the dollar's value. This is termed an **appreciation** of the dollar. More foreign exchange rate \$/£ Foreign exchange Sfx or D\$ Dfx or S\$. Market should force lead to a change in either the supply or demand for foreign exchange then the exchange will change accordingly to re-equilibrate the market. The basic notion is that exchange rates are sensitive to differential inflation rates across Countries. Should the domestic inflation rate rise at a rate greater than our trading Partners, then at a given exchange rate, the price of domestic goods will be rising relative To foreign goods. This will, in turn, increase the demand for foreign goods (imports are Now cheaper in domestic currency terms) and decrease the demand for domestic exports (Domestic exports are now more expensive in foreign currency terms). This results in an Increase in the demand for foreign exchange, as well as a decrease in the supply of Foreign exchange.

This is a long-run effect because of the Law of One Price. This concept states that in the Long run the price of tradable goods must be the same across countries. If this was not The case, then the opportunity for arbitrage profits, buying low in one country and selling High in another, would result in a movement in the exchange rate bringing about the Equalization. For example, suppose Argentine wheat, at the prevailing exchange rate, is cheap in US Dollars. As North Americans buy more and more Argentine wheat, they increase the Demand for the Argentine currency, driving up its value, thus making wheat more Expensive in dollar terms. The exchange rates which would prevail under the Law of One Price are called purchasing power parity exchange rates (PPP). While these do not exist in reality (there are many other factors affecting exchange rates) there is an underlying pressure moving exchange rates in this fashion. PPP exchange rates are used in comparing the economic performance between countries. The World Bank compares countries in their *World Development Report* using a PPP exchange rate. Medium Term - Differential Growth Rates As an economy

grows, its demand for imports will also grow. As income increases, some portion of that increase will be spent on imported goods. In the jargon of macroeconomics, the proportion of the additional dollar of income spent on imports is called the marginal propensity to import. Assume that the marginal propensity to import is the same across countries. Should a country's economy grow faster than its trading partners, then its demand for imports will also be growing faster? In the context of Figure 4, this is represented by increases in both the demand and supply of foreign exchange, but the demand would increase by more. This would result in a slight depreciation of the domestic currency. Short-Run - Differential Interest Rates This factor has become extremely important as countries have liberalized their economies, allowing the flow of financial capital into and out of their countries. It has played an important role in the East Asian and Mexican Peso financial crises. Exchange rate \$/£ Foreign exchange.

### 1.3 Delphi Programming

Delphi 5 provided new features to the Object Inspector, and Delphi 6 includes even more additions to it. As this is a tool programmer's use all the time, along with the editor and the Form Designer, its improvements are really significant.

The most important change in Delphi 6 is the ability of the Object Inspector to expand component references in-place. Properties referring to other components are now displayed in a different color and can be expanded by selecting the + symbol on the left, as it happens with internal subcomponents. You can then modify the properties of that other component without having to select it.

**NOTE** This interface-expansion feature also supports subcomponents, as demonstrated by the new Labeled Edit control. The Form Designer

**TIP** A related feature of the Object Inspector is the ability to select the component referenced by a property. To do this, double-click the property value with the left mouse button while keeping the Ctrl key pressed. For example, if you have a Main Menu component in a form and you are looking at the properties of the form in the Object Inspector, you can select the Main Menu component by moving to the Main Menu property of the form and Ctrl+double-clicking the value of this property. This selects the main menu indicated as the value of the property in

the Object Inspector. Here are some other relevant changes of the Object Inspector: The list at the top of the Object Inspector shows the type of the object and



can be removed to save some space (and considering the presence of the Object Tree View). The properties that reference an object are now a different color and may be expanded without changing the selection. You can optionally also view read-only properties in the Object Inspector. Of course, they are grayed out. The Object Inspector has a new Properties dialog box which allows you to customize the colors of the various types of properties and the overall behavior of this window.

The Project Manager doesn't provide a way to set the options of two different projects at one time. What you can do instead is invoke the Project Options dialog from the Project Manager for each project. The first page of Project Options (Forms) lists the forms that should be created automatically at program startup and the forms that are created manually by the program.

The next page (Application) is used to set the name of the application and the name of its Help file, and to choose its icon. Other Project Options choices relate to the Delphi compiler and linker, version information, and the use of run-time packages.

There are two ways to set compiler options. One is to use the Compiler page of the Project Options dialog. The other is to set or remove individual options in the source code with the `{SX+}` or `{SX-}` commands, where you'd replace *X* with the option you want to set. This second approach is more flexible, since it allows you to change an option only for a specific source-code file, or even for just a few lines of code. The source-level options override the compile-level options.

All project options are saved automatically with the project, but in a separate file with a .DOF extension. This is a text file you can easily edit. You should not delete this file if you have changed any of the default options. Delphi also saves the compiler options in another format in a CFG file, for command-line compilation. The two files have similar content but a different format: The *dcc* command-line compiler, in fact, cannot use .DOF files, but needs the .CFG format. Another alternative for saving compiler options is to press Ctrl+O+O (press the O key twice while keeping Ctrl pressed). This inserts, at the top of the current unit, compiler directives that correspond to the current project options, as in the following listing: `{SA+,B-,C+,D+,E-,F-,G+,H+,I+,J+,K-,L+,M-,N+,O+,P+,Q-,R-,S-,T-,U-,V+,W-,X+,Y+,Zl}`

Memory management in Delphi is subject to three rules: Every object must be created before it can be used; every object must be destroyed after it has been used; and every object must be destroyed only once. Whether you have to do these operations in



your code, or you can let Delphi handle memory management for you, depends on the model you choose among the different approaches provided by Delphi.

Delphi supports three types of memory management for dynamic elements (that is, elements not in the stack and the global memory area):

- Every time you create an object explicitly, in the code of your application, you should also free it. If you fail to do so, the memory used by that object won't be released for other objects until the program terminates.

- When you create a component, you can specify an owner component, passing the owner to the component constructor. The owner component (often a form) becomes responsible for destroying all the objects it owns. In other words, when you free the form, it frees all the components it owns. So, if you create a component and give it an owner, you don't have to remember to destroy it. This is the standard behavior of the components you create at design time by placing them on a form or data module.

- When you allocate memory for strings, dynamic arrays, and objects referenced by interface variables, Delphi automatically frees the memory when the reference goes out of scope. You don't need to free a string: when it becomes unreachable, its memory is released.

## **1.4 Database Programming**

Delphi's support for database applications is one of the key features of the programming environment. Many programmers spend most of their time writing data-access code, which needs to be the most robust portion of a database application. This chapter provides an overview of Delphi's extensive support for database programming. What you will find here is a discussion of the theory of database design. I am assuming that you already know the fundamentals of database design and have already designed the structure of a database. I will not look into database-specific problems; my goal is to help you understand how Delphi supports database access. I will begin with an explanation of the alternatives Delphi offers in terms of data access, and then I will provide an overview of the database components that I have used in my program. This chapter includes an overview of the `TDataSet` class, an in-depth analysis of the `TField` components, and the use of data-aware controls. The following chapters will provide information on more advanced database programming topics, such as client/server programming, the use of `dbGo`, `dbExpress`, and `InterBase Express`.

## 1.5 Borland Database Engine (BDE)

The BDE originated with Paradox, well before Delphi existed, and was extended by Borland to support other local databases and many SQL servers. The BDE has direct access to dBASE, Paradox, ASCII, FoxPro, and Access tables. A series of drivers (called SQL Links and available only in Delphi Enterprise) allows access to some SQL servers, including Oracle, Sybase, Microsoft, Informix, InterBase, and DB2 servers. If you need access to a different database, the BDE can also interface with ODBC drivers.

## 1.6 Graphical Data-Aware Controls

Finally, Delphi includes two graphical data-aware controls:

- **DBImage**, which is an extension of an Image component that shows a picture stored in a BLOB field (provided the database use a graphic format that the Image component supports, such as BMP and JPEG). The output of the Cust- Lookup example, with the BLookupComboBox showing multiple fields in its drop-down list.

- **DBChart** is a data-aware business graphic component or the data-aware version of the TeeChart control built by David Berneda. To demonstrate the use of the DBChart control, I have added this component to a simple example showing a data grid. The application, called ChartDB, shows a pie chart with the surface of each country of the COUNTRY.DB table. The program has almost no code, as all the settings can be done using the specific component editor, which has several options but is quite easy to use. Here are some of the key properties of the component, taken from the form description:

**object** DBChart1: TDBChart

Legend.Visible = False

Align = alClient

**object** Series1: TPieSeries

Marks.ArrowLength = 8

Marks.Visible = True

DataSource = Table1

XLabelsSource = 'Name'

ExplodeBiggest = 3

OtherSlice.Style = poBelowPercent

OtherSlice.Text = 'Others'



```

OtherSlice.Value = 2
PieValues.ValueSource = 'Area'
end;
end.

```

What I have done is show the area field as the data source for the pie chart (the PieValues ValueSource property of the series), use the name field for the labels (the XLabelsSource property of the series), and condense all the countries with a value below 2 percent in a single section indicated as Others (the OtherSlide subproperties). As a minor addition to the code, I have added two radio buttons you can use to toggle between the area and the population. The code of the two radio buttons simply sets the source of the series, after casting it to the proper series type, as in:

```

procedure TForm1.RadioPopulationClick(Sender: TObject);
begin
  DBChart1.Title.Text [0] := 'Population of Countries';
  (DBChart1.Series [0] as TPieSeries).PieValues.ValueSource := 'Population';
end;

```

## 1.7 The ClientDataSet Component

Finally, there is a component derived from TDataSet that has a peculiar behavior and can be combined with other data-access components. The ClientDataSet component, in fact, is a dataset accessing data kept in memory. The in-memory data can be totally temporary (lost as you exit the program), saved to a local file as a snapshot, and imported by another dataset using a Provider component. This last situation is certainly the most common: You can hook a ClientDataSet to any other local dataset, or use Borland's multitier support (discussed in Accessing a Database: BDE, dbExpress, and other alternatives "Multitier Database Applications with DataSnap") to retrieve data from a dataset hosted by a different application, possibly running on a separate computer. The ClientDataSet component becomes particularly useful if the data-access components you are using provide limited or no caching. This is particularly true of the new dbExpress engine, but can equally help you when using the BDE or other native components.



On the other hand, ADO already provides most of the services of the ClientDataSet component and using these two at the same time can be useful only in limited situations

## 1.8 Classic BDE Components

Each of the database-access solutions discussed above has its own set of data-access, database connection, and extra utility components on a specific page of the Component palette. The classic BDE components have been moved to the new BDE page and include the Table, Query, and StoredProc components. The ADO, dbExpress, and InterBase Express components are each in specific pages, and all include specific dataset components and others that tend to mimic the BDE components, simplifying the porting of existing applications.

The Data Access page of the Component palette includes only the Data Source Component and others not specifically related with any single data access technology. Besides the data-access component of your choice, a Delphi visual application generally uses some data-aware controls (in the Data Controls page) and the DataSource component. Data-aware controls are visual components used to view and edit the data in a form and are extensions of standard components such as edit and list boxes, radio buttons, images, and the

Grid. The DataSource component has the role of connector between the data-aware controls and a dataset component.

## 1.9 Tables and Queries

The simplest traditional way to specify data access in Delphi was to use the BDE Table component. A Table object simply refers to a database table. When you use a Table component, you need to indicate the name of the database you want to use in its DatabaseName property. You can enter an alias or the path of the directory with the table files. The Object Inspector lists the available names, which depend on the aliases installed in the BDE. You also need to indicate a proper value in the TableName property. The Object Inspector lists the available tables of the current database (or directory), so you should generally select the DatabaseName property first. Another classic dataset is the BDE Query component. A query requires a SQL language

command. You can customize a query using SQL more easily than you can customize a table (as long as you know at least the basic elements of SQL, of course). The Query component has a `DatabaseName` property like the Table component, but it does not have a `TableName` property. The table is indicated in the SQL statement, stored in the SQL property. For example, you can write a simple SQL statement like this:

**Select \* from** Country where Country is the name of a table and the asterisk (\*) indicates that you want to use all of the fields in the table.

The efficiency of a table or a query varies depending on the database you are using. In general, we can say that the Table component tends to be faster on local tables, while the Query component tends to be faster on SQL servers, although this is just a very general rule, and in many cases you might have the opposite effect. We'll see some efficiency issues while discussing

client/server development in the third BDE dataset component is `StoredProc`, which refers to stored procedures of a SQL server database. You can run these procedures and get the results in the form of a database table. Stored procedures can only be used with SQL servers.

## 1.10 DBNavigator and Dataset Actions

DBNavigator is a collection of buttons used to navigate and perform actions on the database. You can disable some of the buttons of the DBNavigator control, by removing some of the elements of the `VisibleButtons` set. The buttons perform basic actions on the connected dataset, so you can easily replace them

With your own toolbar, particularly if you use an `ActionList` component with the predefined database actions provided by Delphi. In this case, in fact, you get all the standard behaviors, but you'll also see the various buttons enabled only when their action is legitimate. **TIP** If you use the standard actions, you can avoid connecting them to a specific `DataSource` component, and the actions will be applied to the dataset connected to the visual control that currently has the input focus. This way a single toolbar can be used for multiple datasets displayed by a form.



## 1.11 Text-Based Data-Aware Controls

There are multiple text-oriented components:

DBText displays the contents of a field that cannot be modified by the user. It is a data aware Label graphical control. It can be very useful, but users might confuse this control with the plain labels that indicate the content of each field-based control. DBEdit lets the user edit a field (change the current value) using an Edit control. At times, you might want to disable editing and use a DBEdit as if it were a DBText, but highlighting the fact that this is data coming from the database. DBMemo lets the user see and modify a large text field, eventually stored in a memo or BLOB (binary large object) field. It resembles the Memo component and has full editing capabilities, but all the text is rendered in a single font.

DBRichEdit is a component that lets the user edit a formatted text file; it is based on a RichEdit Windows common control and, in contrast to DBMemo, it allows text with multiple fonts and paragraph styles.

## 1.12 Navigating a Dataset

We've seen that a dataset has only one active record, and you can imagine that the active record changes often, in response of user actions or because of internal commands given to the dataset. To move around the dataset and change the active record, there are methods of the TDataSet class, particularly in the section commented as "position, movement." You can move to the next or previous record, jump back and forth by

A given number of records (with MoveBy), or go directly to the first or last record of the dataset. These operations of the dataset are generally available in the DBNavigator component or in the standard dataset actions, and they are not particularly complex to understand. What is not obvious, though, is how a dataset handles the extreme positions. If you open any dataset with a navigator attached, you can see that as you move on record by record, the Next button remains enabled even when you've reached the last record. It's only when you try to move forward after the last record that the current record apparently doesn't change and the button is disabled. This is because the Eof test (end of file) succeeds only when the cursor has been moved to a special position after the last record. If you jump to the end with the Last button, instead, you'll immediately be at the very end. You'll see exactly the same behavior for the first record



(and the Bof test). As we'll see in a while, this approach is very handy, as we can scan a dataset testing for Eof to be True and, at this point, we know we've also already processed the last record of the dataset.

**NOTE** Handling this special record positions before the beginning and after the end of the dataset, which are called *cracks*, is very important (and quite confusing) when you write a custom dataset. Besides moving around record by record or by a given number of records, programs might need to jump to specific records or positions. Some datasets support the RecordCount property and allow movement to a record at a given position in the dataset using the RecNo property. These properties can be used only for datasets that support positions natively, which basically excludes all client/server architectures, unless you grab all of the records in a local cache (something you'll generally want to avoid) and then navigate on the cache. As we'll see in the next chapter, when you open a query on a SQL server you fetch only the records you are using, so Delphi doesn't know the record count, at least not in advance. There are two alternatives you can use to refer to a record in a dataset, regardless of its type. You can save a reference to the current record and then jump back to it after moving around. This is accomplished by using bookmarks, either in the TBookmark or the more modern TBookmarkStr form. You can locate a record of the dataset matching given criteria, using the Locate method. This even works after you close and reopen the dataset, because you're working at a logical (and not physical) level. This approach is presented in the next section.

## 2.1 Main Menu Screen

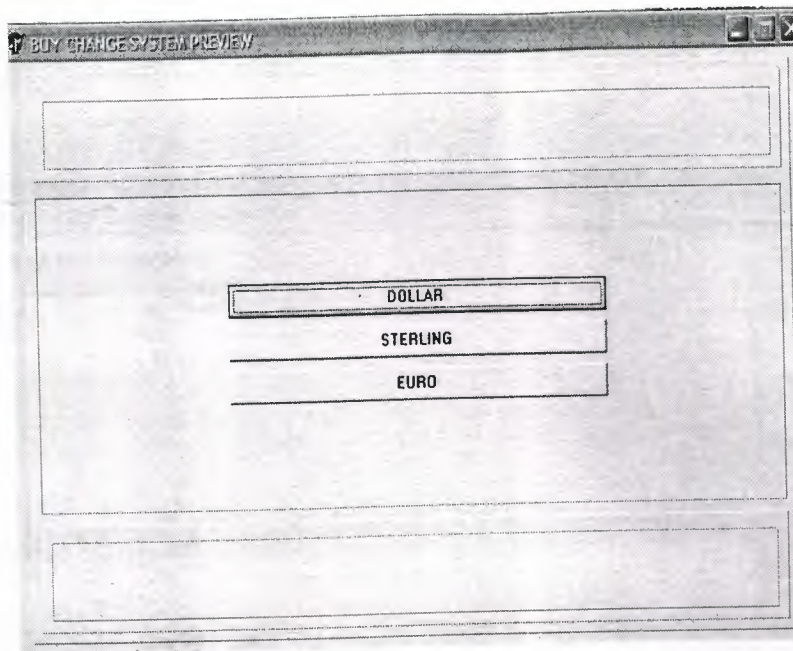


Figure 2.1 Main Menu Screen

The figure above shows the first screen which occurs when you first start running the program. This screen contains the following:

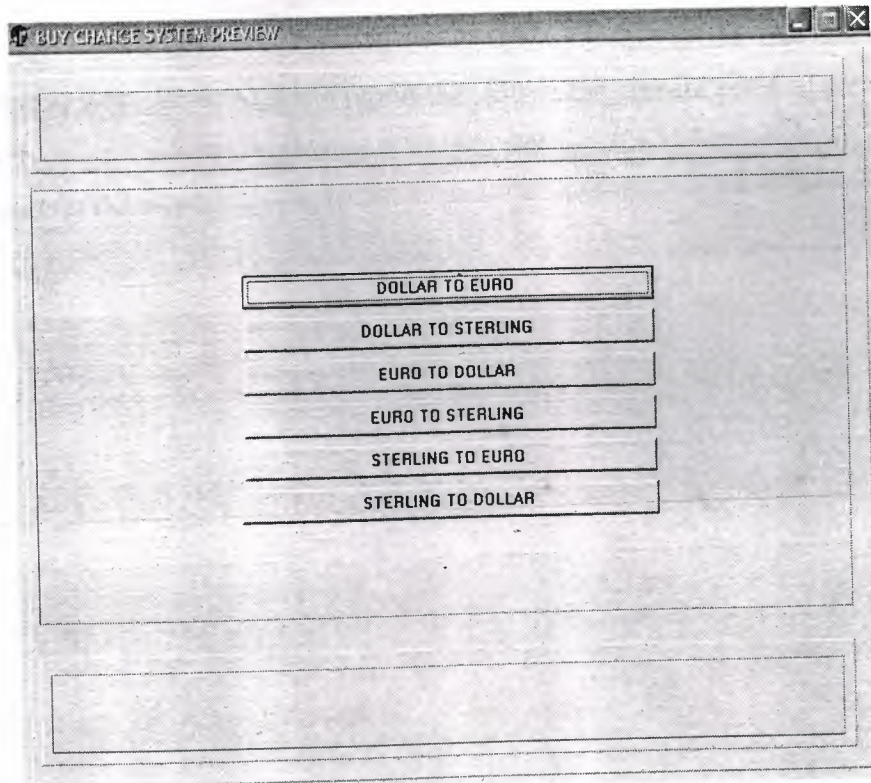
- ◆ The calendar: This automatically obtains the current date month and year of the system which the program is running on.
- ◆ Buy change system preview: which shows the current selling price of currencies that you have entered in the change system, which will be explained later.





**Figure 2.2** Buy change system preview

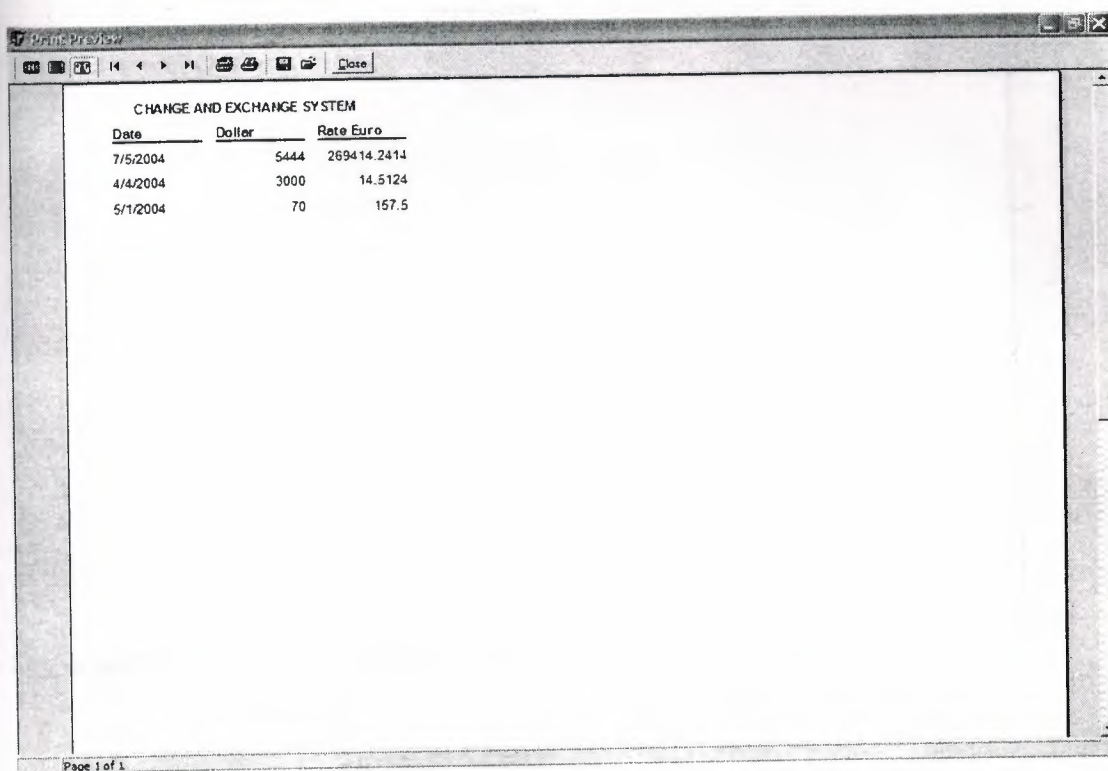
- ◆ Sell exchange system preview: which shows the current reports of the exchange system. The following screen will appear and allow you to Select the type of exchange report that you wish to preview.



**Figure 2.3** Exchange report



The following report preview will show the exchange rates related to which form you choose:



Date	Dollar	Rate Euro
7/5/2004	5444	269414.2414
4/4/2004	3000	14.5124
5/1/2004	70	157.5

Figure 2.4 Exchange rates

- ◆ Change system preview: it is a report that shows the current price of the currencies in the stuck exchange related to selling currencies, which can be printed by the button next to it.

CHANGE AND EXCHANGE SYSTEM

Turkish	Sterling	Dollar	Euro	Date
1430000	2400000	1400000	2700000	1/1/2004

Page 1 of 1

Figure 2.5 Change System Preview

- ◆ Exchange system: it is a report that shows the current price of the currencies in the stuck exchange related to buying currencies, which can be printed by the button next to it.

CHANGE AND EXCHANGE SYSTEM

Turkish	Sterling	Dollar	Euro	Date
1200000	1400000	1700000	2000000	1/1/2004

Page 1 of 1

Figure 2.6 Exchange System Preview

## 2.2 Change System

MAIN MENU CHANGE EXCHANGE SYSTEM

NEAR EAST UNIVERSITY  
DEPARTMENT OF COMPUTER ENGINEERING  
DEVELOPMENT DELPHI PROGRAM FOR EXCHANGE AND CHANGE SYSTEM

CHANGE SYSTEM REPORT CHANGE SYSTEM

TURKISH TO DOLLAR | TURKISH TO STERLING | **TURKISH TO EURO**

TL Turkish	£ Sterling	\$ Dollar	€ Euro
1430000	2400000	1400000	2700000

Date  
1/1/2004

ADD TO CHANGE SYSTEM:

Date  
7/5/2004

EXCHANGE RATE	EXCHANGE AMOUNT	EXCHANGE VALUE
27000000	100	270000000

Figure 2.7 Change System

This screen consists of three sub-menus, which will allow you to complete your transaction processes. In addition, the forms contain the following:

The first part of each form grapes the exchange rates of the currencies from your database that you have entered.

The second part deals with the current amount of currency that you are exchanging at the time being. It consists of a navigation bar that allows you to scroll in the fields of your transactions.



## 2.3 Exchange Curves

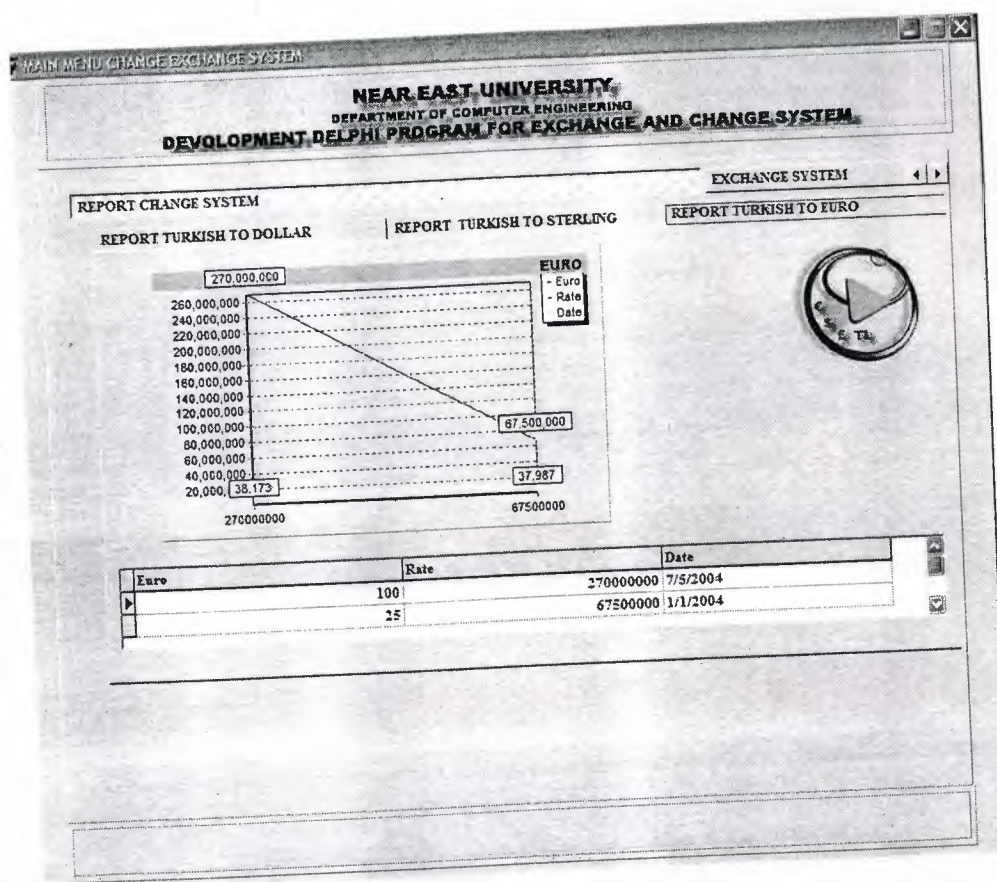


Figure 2.8 Change Curve

It consists of three chart reports that are all linked to the data base and change in respect to the processes that you make, that each transaction made is added or deleted and presented in the as a curve in respect to the money amount you have processed. The rest of the graphs are shown below

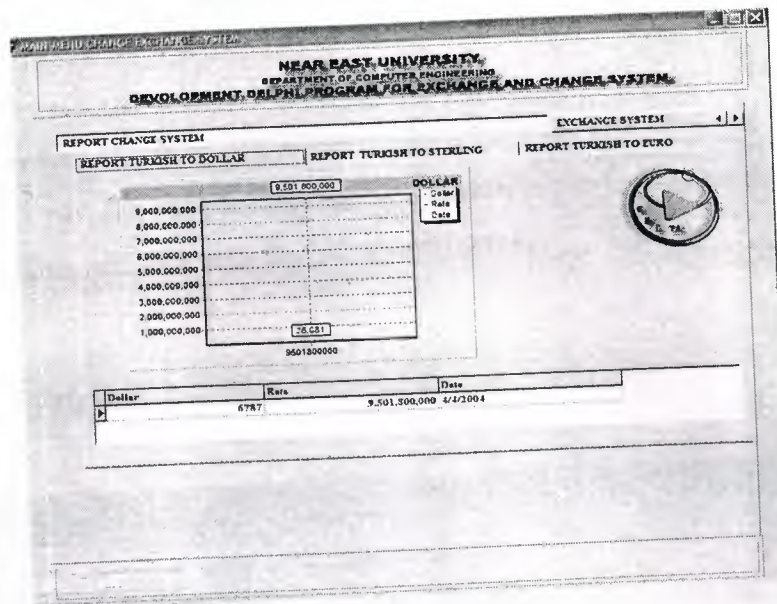


Figure 2.9 Change Curve

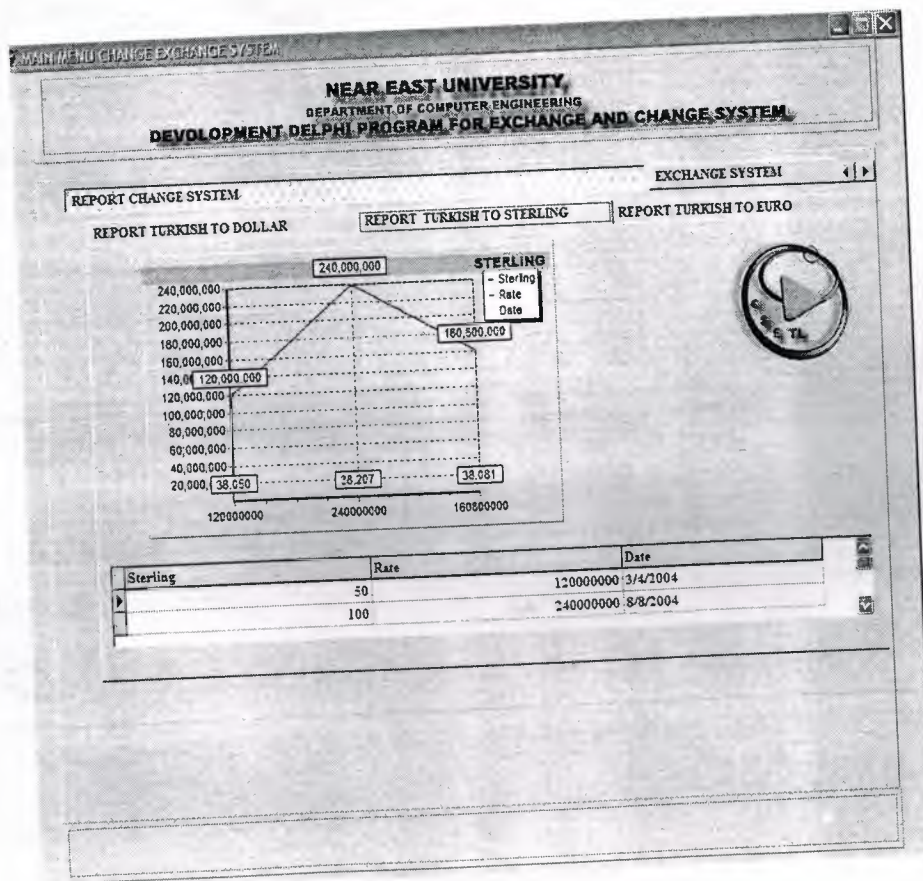


Figure 2.10 Change Curve



These graphs are just to keep track of you exchange processes in terms of money, and also shows you which of the currencies is have the most demand in the market, at this point you can analyze the current situation of the foreign currencies flow in you exchange office and take your decisions among that.

## 2.4 Exchange System

MAIN MENU CHANGE EXCHANGE SYSTEM

**NEAR EAST UNIVERSITY**  
DEPARTMENT OF COMPUTER ENGINEERING  
DEVELOPMENT DELPHI PROGRAM FOR EXCHANGE AND CHANGE SYSTEM

EXCHANGE SYSTEM INFO CHANGE SYSTEM

EURO TO DOLLAR

CHANGE	[TL] Turkish	[£] Sterling	[\$] Dollar	[€] Euro	Date
	1430000	2400000	1400000	2700000	1/1/2004

EXCHANGE

EXCHANGE	[TL] Turkish	[£] Sterling	[\$] Dollar	[€] Euro	Date
	1200000	1400000	1700000	2000000	1/1/2004

EXCHANGE

EXCHANGE	[\$] Dollars given amount	[\$] DOLLAR PRICE	[€] woned ratio	equal amount	Date
	54345			40081.7156	7/8/2004




Figure 2.11 Exchange System



The exchange system consists of three fields that allow you to deal with the exchange of the foreign currencies each one consist of selling and buying prices of the currencies and another field for your processes, where you can add delete or modify your processes.

## 2.5 Exchange Bar Graphs

The following bar graphs illustrate the change in the currencies by date and respectively with the database.

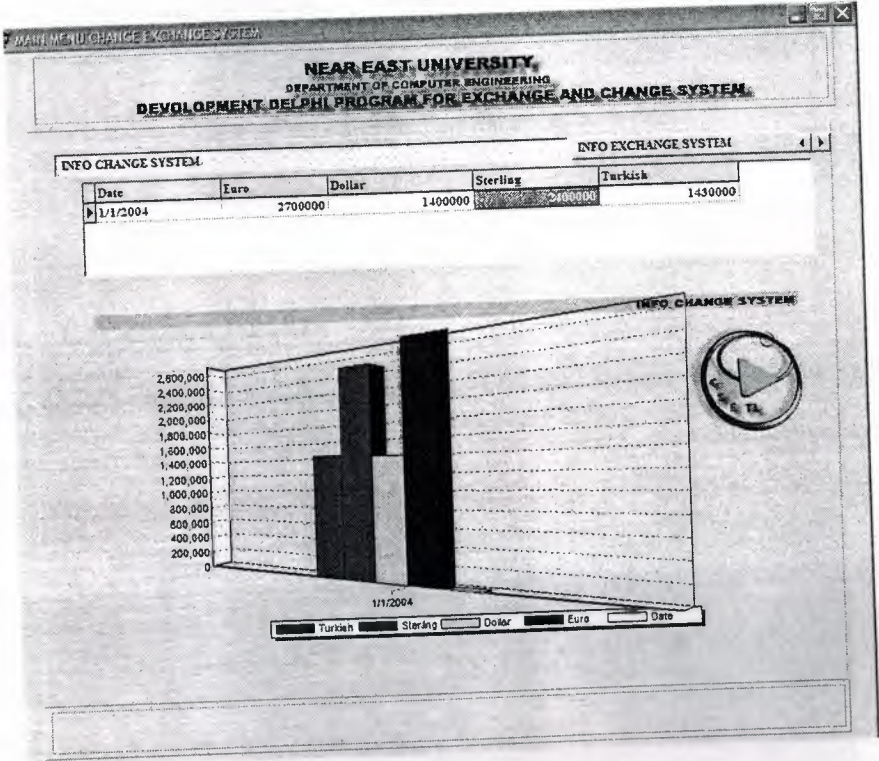


Figure 2.12 Exchange Bar Graphs a

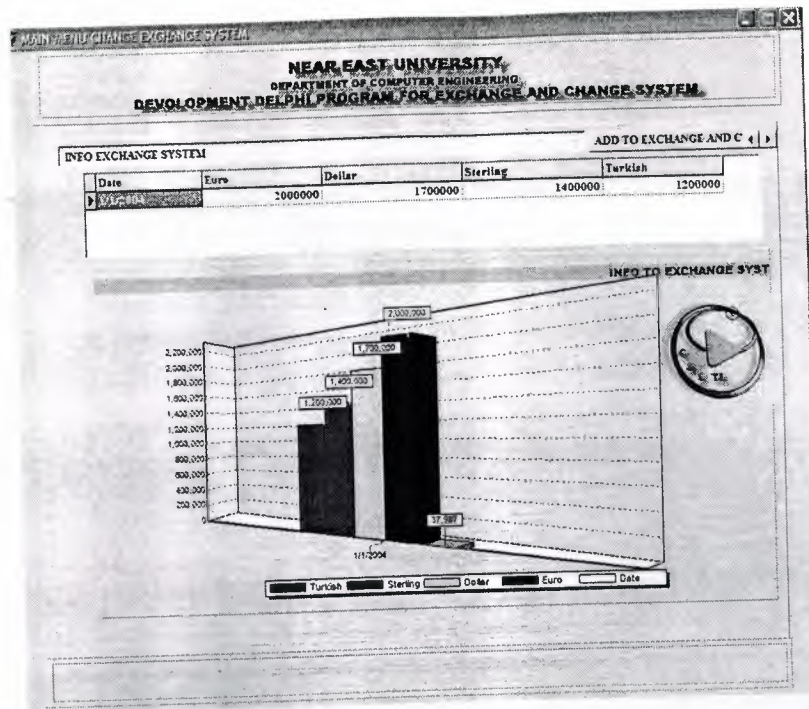


Figure 2.13 Exchange Bar Graphs b

## 2.6 Adding the Exchange Rates

a. **Selling Rates:** here in this form you can you can add the current rates of the local currency.

The screenshot shows the "ADD TO EXCHANGE AND CRANGE SYSTEM" form. It includes a table for entering exchange rates for the date 1/1/2004. The table has columns for Turkish (TL), Sterling (£), Dollar (\$), and Euro (€). A legend at the bottom identifies the currencies: Turkish (TL), Sterling (£), Dollar (\$), Euro (€), and Date.

TL Turkish	£ Sterling	\$ Dollar	€ Euro
1200000	1400000	1700000	2000000

Figure 2.14 Adding the Exchange Rates (Selling Rates)



## b. Foreign Currency Rates

Here in the following form you can add the foreign exchange rates

MAIN MENU CHANGE EXCHANGE SYSTEM

NEAR EAST UNIVERSITY.  
DEPARTMENT OF COMPUTER ENGINEERING  
DEVELOPMENT DELPHI PROGRAM FOR EXCHANGE AND CHANGE SYSTEM

ADD TO EXCHANGE AND CHANGE SYSTEM

CHANGE SYSTEM

TL Turkish	(£) Sterling	(\$ ) Dollar	(€) Euro
1430000	3400000	1400000	3700000

Date  
1/1/2004

Logo: A circular logo with a stylized 'N' and 'E' and the text 'NEAR EAST UNIVERSITY' around it.

Figure 2.15 Adding the Exchange Rates (Foreign Currency Rates)



## 2.7 Database Desktop

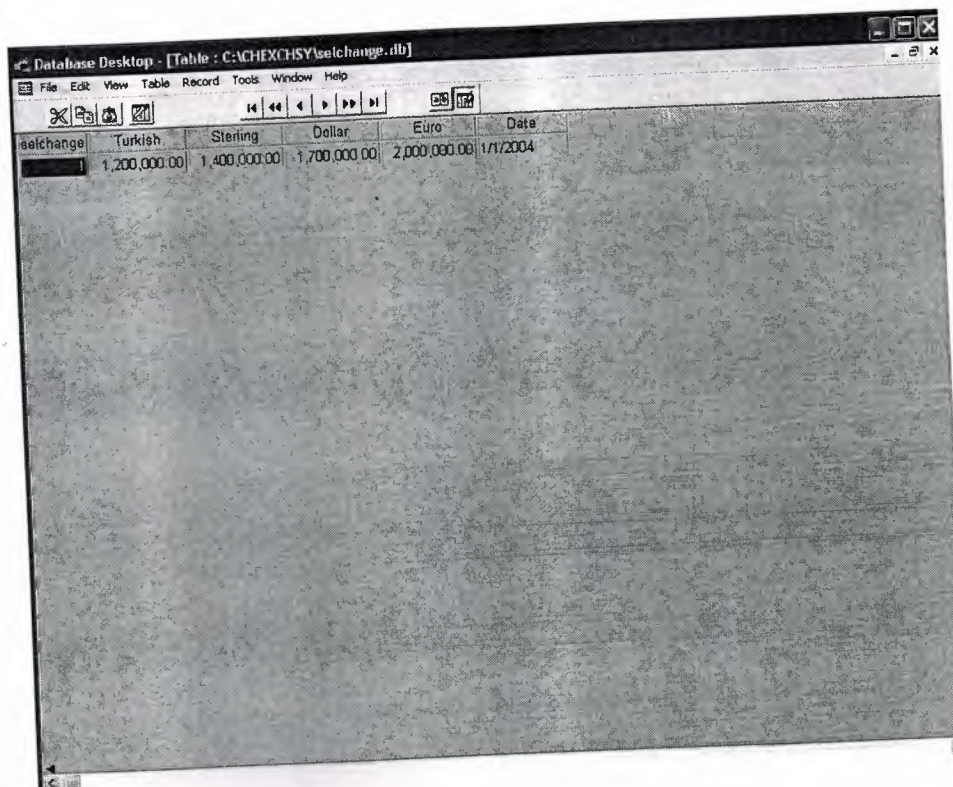


Figure 2.16 Database Desktop

Database Desktop is a database tool where you can create or restructure database tables, or browse and edit their data. You can work with tables in Paradox, dBase, and SQL formats.

## DATA FLOW DIAGRAM (DFD)

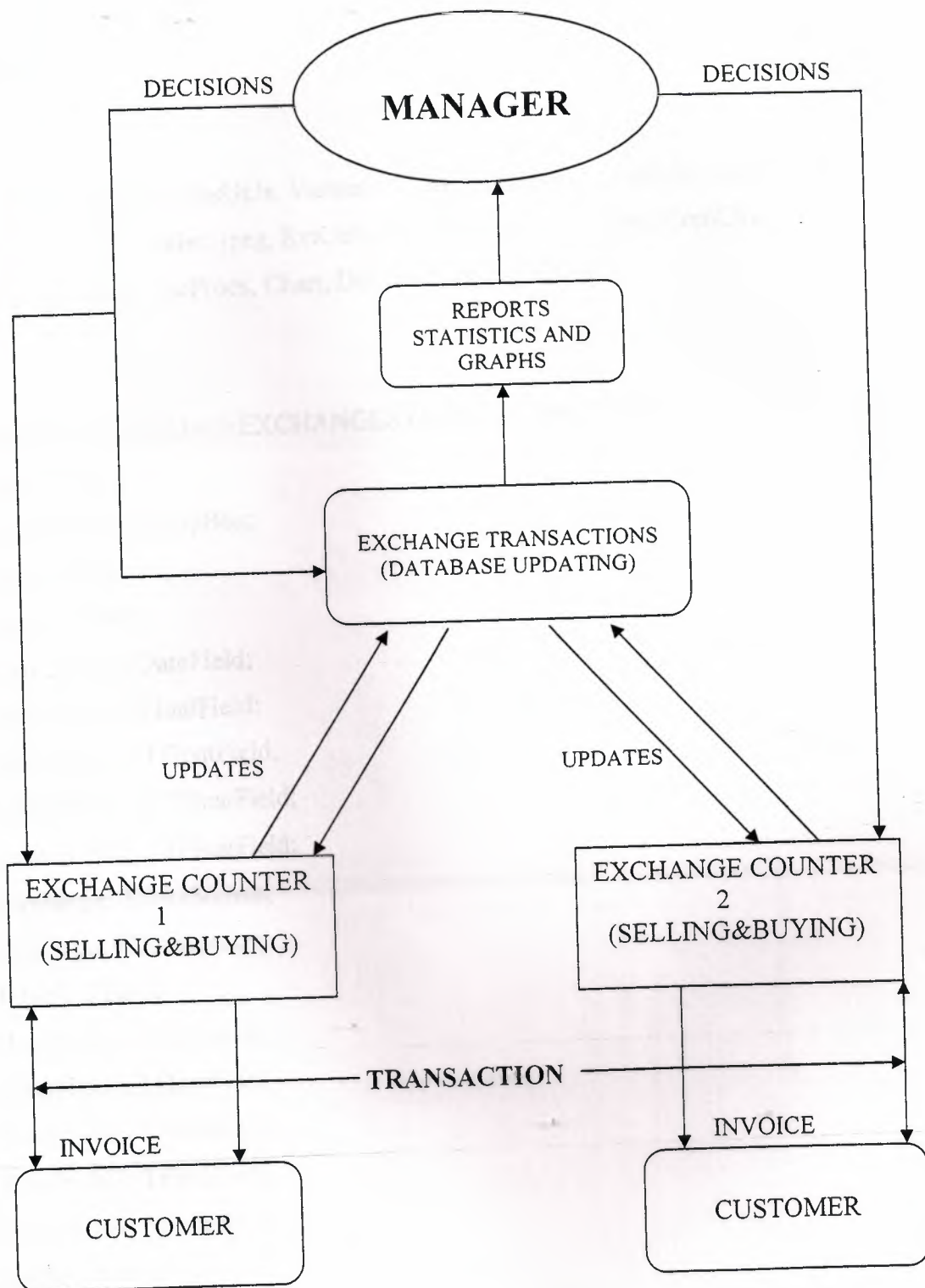


Figure 3.1 Data Flow Diagram

## 3.2 MAIN MENU

unit CHEXSU;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, jpeg, ExtCtrls, DBCtrls, StdCtrls, Mask, ComCtrls,  
TeEngine, Series, TeeProcs, Chart, DbChart, Grids, DBGrids;

type

TMAINMENUCHANGEEXCHANGESYSTEM = class(TForm)

Panel1: TPanel;

GroupBox1: TGroupBox;

Image1: TImage;

Table1: TTable;

Table1Date: TDateField;

Table1Euro: TFloatField;

Table1Dollar: TFloatField;

Table1Sterling: TFloatField;

Table1Turkish: TFloatField;

buychange: TDataSource;

selchange: TDataSource;

Table2: TTable;

DateField1: TDateField;

FloatField1: TFloatField;

FloatField2: TFloatField;

FloatField3: TFloatField;

FloatField4: TFloatField;

dollar: TDataSource;

Table3: TTable;

Table3Dollar: TFloatField;

Table3Rate: TFloatField;



Table3Date: TDateField;  
sterlaing: TDataSource;  
Table4: TTable;  
euro: TDataSource;  
Table5: TTable;  
Table9: TTable;  
exdollarE: TDataSource;  
Table10: TTable;  
exeuroD: TDataSource;  
Table11: TTable;  
Table12: TTable;  
exeuroS: TDataSource;  
exdollarS: TDataSource;  
Table14: TTable;  
exsterlingD: TDataSource;  
exsterlingE: TDataSource;  
Table15: TTable;  
Table17: TTable;  
Table18: TTable;  
Table6: TTable;  
Table7: TTable;  
Table8: TTable;  
Panel2: TPanel;  
GroupBox32: TGroupBox;  
Panel3: TPanel;  
PageControl1: TPageControl;  
TabSheet1: TTabSheet;  
Image2: TImage;  
TabControl1: TTabControl;  
Panel6: TPanel;  
MonthCalendar1: TMonthCalendar;  
Button1: TButton;  
Button3: TButton;  
Button4: TButton;

Button5: TButton;  
TabSheet2: TTabSheet;  
PageControl3: TPageControl;  
TabSheet11: TTabSheet;  
GroupBox6: TGroupBox;  
Label16: TLabel;  
Label17: TLabel;  
Label18: TLabel;  
Label19: TLabel;  
Label20: TLabel;  
DBEdit12: TDBEdit;  
DBEdit13: TDBEdit;  
DBEdit14: TDBEdit;  
DBEdit15: TDBEdit;  
DBEdit16: TDBEdit;  
DBNavigator2: TDBNavigator;  
GroupBox8: TGroupBox;  
Label26: TLabel;  
GroupBox9: TGroupBox;  
Label27: TLabel;  
Label28: TLabel;  
Label29: TLabel;  
Label30: TLabel;  
DBEdit25: TDBEdit;  
DBEdit22: TDBEdit;  
DBEdit23: TDBEdit;  
DBEdit24: TDBEdit;  
DBNavigator5: TDBNavigator;  
TabSheet12: TTabSheet;  
GroupBox5: TGroupBox;  
Label11: TLabel;  
Label12: TLabel;  
Label13: TLabel;  
Label14: TLabel;

Label15: TLabel;  
DBEdit7: TDBEdit;  
DBEdit8: TDBEdit;  
DBEdit9: TDBEdit;  
DBEdit10: TDBEdit;  
DBEdit11: TDBEdit;  
DBNavigator4: TDBNavigator;  
GroupBox10: TGroupBox;  
Label31: TLabel;  
GroupBox11: TGroupBox;  
Label32: TLabel;  
Label33: TLabel;  
Label34: TLabel;  
Label35: TLabel;  
DBEdit26: TDBEdit;  
DBEdit28: TDBEdit;  
DBEdit29: TDBEdit;  
DBEdit27: TDBEdit;  
DBNavigator6: TDBNavigator;  
TabSheet13: TTabSheet;  
GroupBox7: TGroupBox;  
Label21: TLabel;  
Label22: TLabel;  
Label23: TLabel;  
Label24: TLabel;  
Label25: TLabel;  
DBEdit17: TDBEdit;  
DBEdit18: TDBEdit;  
DBEdit19: TDBEdit;  
DBEdit20: TDBEdit;  
DBEdit21: TDBEdit;  
DBNavigator3: TDBNavigator;  
GroupBox12: TGroupBox;  
Label36: TLabel;



GroupBox13: TGroupBox;  
Label37: TLabel;  
Label38: TLabel;  
Label39: TLabel;  
Label40: TLabel;  
DBEdit30: TDBEdit;  
DBEdit31: TDBEdit;  
DBEdit32: TDBEdit;  
DBEdit33: TDBEdit;  
DBNavigator7: TDBNavigator;  
TabSheet3: TTabSheet;  
PageControl4: TPageControl;  
TabSheet14: TTabSheet;  
DBGrid1: TDBGrid;  
DBChart1: TDBChart;  
Series1: TLineSeries;  
Series2: TLineSeries;  
Series3: TLineSeries;  
TabSheet15: TTabSheet;  
DBGrid2: TDBGrid;  
DBChart2: TDBChart;  
LineSeries1: TLineSeries;  
LineSeries2: TLineSeries;  
LineSeries3: TLineSeries;  
TabSheet16: TTabSheet;  
DBGrid3: TDBGrid;  
DBChart3: TDBChart;  
LineSeries4: TLineSeries;  
LineSeries5: TLineSeries;  
LineSeries6: TLineSeries;  
TabSheet4: TTabSheet;  
Label71: TLabel;  
PageControl5: TPageControl;  
TabSheet17: TTabSheet;

GroupBox14: TGroupBox;  
Label41: TLabel;  
Label42: TLabel;  
Label43: TLabel;  
Label44: TLabel;  
Label45: TLabel;  
Label77: TLabel;  
DBEdit34: TDBEdit;  
DBEdit35: TDBEdit;  
DBEdit36: TDBEdit;  
DBEdit37: TDBEdit;  
DBEdit38: TDBEdit;  
DBNavigator8: TDBNavigator;  
GroupBox20: TGroupBox;  
Label78: TLabel;  
Label79: TLabel;  
Label80: TLabel;  
Label81: TLabel;  
Label82: TLabel;  
Label113: TLabel;  
DBEdit64: TDBEdit;  
DBEdit65: TDBEdit;  
DBEdit66: TDBEdit;  
DBEdit67: TDBEdit;  
DBEdit68: TDBEdit;  
DBNavigator14: TDBNavigator;  
GroupBox27: TGroupBox;  
Label120: TLabel;  
Label121: TLabel;  
Label122: TLabel;  
Label123: TLabel;  
Label124: TLabel;  
Label125: TLabel;  
DBEdit97: TDBEdit;

DBEdit98: TDBEdit;  
DBEdit99: TDBEdit;  
DBNavigator21: TDBNavigator;  
Edit89: TEdit;  
Edit90: TEdit;  
TabSheet18: TTabSheet;  
GroupBox18: TGroupBox;  
Label61: TLabel;  
Label62: TLabel;  
Label63: TLabel;  
Label64: TLabel;  
Label65: TLabel;  
Label76: TLabel;  
DBEdit54: TDBEdit;  
DBEdit55: TDBEdit;  
DBEdit56: TDBEdit;  
DBEdit57: TDBEdit;  
DBEdit58: TDBEdit;  
DBNavigator12: TDBNavigator;  
GroupBox21: TGroupBox;  
Label83: TLabel;  
Label84: TLabel;  
Label85: TLabel;  
Label86: TLabel;  
Label87: TLabel;  
Label109: TLabel;  
DBEdit69: TDBEdit;  
DBEdit70: TDBEdit;  
DBEdit71: TDBEdit;  
DBEdit72: TDBEdit;  
DBEdit73: TDBEdit;  
DBNavigator15: TDBNavigator;  
GroupBox28: TGroupBox;  
Label126: TLabel;



Label127: TLabel;  
Label128: TLabel;  
Label129: TLabel;  
Label130: TLabel;  
Label131: TLabel;  
DBNavigator22: TDBNavigator;  
DBEdit100: TDBEdit;  
DBEdit101: TDBEdit;  
DBEdit102: TDBEdit;  
Edit3: TEdit;  
Edit5: TEdit;  
TabSheet19: TTabSheet;  
GroupBox15: TGroupBox;  
Label46: TLabel;  
Label47: TLabel;  
Label48: TLabel;  
Label49: TLabel;  
Label50: TLabel;  
Label75: TLabel;  
DBEdit39: TDBEdit;  
DBEdit40: TDBEdit;  
DBEdit41: TDBEdit;  
DBEdit42: TDBEdit;  
DBEdit43: TDBEdit;  
DBNavigator9: TDBNavigator;  
GroupBox22: TGroupBox;  
Label88: TLabel;  
Label89: TLabel;  
Label90: TLabel;  
Label91: TLabel;  
Label92: TLabel;  
Label110: TLabel;  
DBEdit74: TDBEdit;  
DBEdit75: TDBEdit;

DBEdit76: TDBEdit;  
DBEdit77: TDBEdit;  
DBEdit78: TDBEdit;  
DBNavigator16: TDBNavigator;  
GroupBox29: TGroupBox;  
Label132: TLabel;  
Label133: TLabel;  
Label134: TLabel;  
Label135: TLabel;  
Label136: TLabel;  
Label137: TLabel;  
DBEdit103: TDBEdit;  
Edit7: TEdit;  
DBEdit104: TDBEdit;  
DBEdit105: TDBEdit;  
DBNavigator23: TDBNavigator;  
Edit93: TEdit;  
TabSheet20: TTabSheet;  
GroupBox17: TGroupBox;  
Label56: TLabel;  
Label57: TLabel;  
Label58: TLabel;  
Label59: TLabel;  
Label60: TLabel;  
Label74: TLabel;  
DBEdit49: TDBEdit;  
DBEdit50: TDBEdit;  
DBEdit51: TDBEdit;  
DBEdit52: TDBEdit;  
DBEdit53: TDBEdit;  
DBNavigator11: TDBNavigator;  
GroupBox23: TGroupBox;  
Label93: TLabel;  
Label94: TLabel;

Label95: TLabel;  
Label96: TLabel;  
Label97: TLabel;  
Label111: TLabel;  
DBEdit79: TDBEdit;  
DBEdit80: TDBEdit;  
DBEdit81: TDBEdit;  
DBEdit82: TDBEdit;  
DBEdit83: TDBEdit;  
DBNavigator17: TDBNavigator;  
GroupBox30: TGroupBox;  
Label138: TLabel;  
Label139: TLabel;  
Label140: TLabel;  
Label141: TLabel;  
Label142: TLabel;  
Label143: TLabel;  
DBEdit106: TDBEdit;  
Edit9: TEdit;  
DBEdit107: TDBEdit;  
DBEdit108: TDBEdit;  
DBNavigator24: TDBNavigator;  
Edit92: TEdit;  
TabSheet21: TTabSheet;  
GroupBox16: TGroupBox;  
Label51: TLabel;  
Label52: TLabel;  
Label53: TLabel;  
Label54: TLabel;  
Label55: TLabel;  
Label73: TLabel;  
DBEdit44: TDBEdit;  
DBEdit45: TDBEdit;  
DBEdit46: TDBEdit;



DBEdit47: TDBEdit;  
DBEdit48: TDBEdit;  
DBNavigator10: TDBNavigator;  
GroupBox24: TGroupBox;  
Label98: TLabel;  
Label99: TLabel;  
Label100: TLabel;  
Label101: TLabel;  
Label102: TLabel;  
Label112: TLabel;  
DBEdit84: TDBEdit;  
DBEdit85: TDBEdit;  
DBEdit86: TDBEdit;  
DBEdit87: TDBEdit;  
DBEdit88: TDBEdit;  
DBNavigator18: TDBNavigator;  
GroupBox31: TGroupBox;  
Label144: TLabel;  
Label145: TLabel;  
Label146: TLabel;  
Label147: TLabel;  
Label148: TLabel;  
Label149: TLabel;  
DBEdit109: TDBEdit;  
DBEdit112: TDBEdit;  
DBEdit113: TDBEdit;  
DBNavigator25: TDBNavigator;  
Edit110: TEdit;  
Edit111: TEdit;  
TabSheet22: TTabSheet;  
GroupBox19: TGroupBox;  
Label66: TLabel;  
Label67: TLabel;  
Label68: TLabel;

Label69: TLabel;  
Label70: TLabel;  
Label72: TLabel;  
DBEdit59: TDBEdit;  
DBEdit60: TDBEdit;  
DBEdit61: TDBEdit;  
DBEdit62: TDBEdit;  
DBEdit63: TDBEdit;  
DBNavigator13: TDBNavigator;  
GroupBox25: TGroupBox;  
Label103: TLabel;  
Label104: TLabel;  
Label105: TLabel;  
Label106: TLabel;  
Label107: TLabel;  
Label108: TLabel;  
DBEdit89: TDBEdit;  
DBEdit90: TDBEdit;  
DBEdit91: TDBEdit;  
DBEdit92: TDBEdit;  
DBEdit93: TDBEdit;  
DBNavigator19: TDBNavigator;  
GroupBox26: TGroupBox;  
Label118: TLabel;  
Label119: TLabel;  
Label114: TLabel;  
Label115: TLabel;  
Label116: TLabel;  
Label117: TLabel;  
DBEdit94: TDBEdit;  
Edit1: TEdit;  
DBEdit95: TDBEdit;  
DBEdit96: TDBEdit;  
DBNavigator20: TDBNavigator;

Edit66: TEdit;  
TabSheet6: TTabSheet;  
DBGrid5: TDBGrid;  
DBChart4: TDBChart;  
BarSeries1: TBarSeries;  
BarSeries2: TBarSeries;  
BarSeries3: TBarSeries;  
Series4: TBarSeries;  
Series5: TBarSeries;  
TabSheet7: TTabSheet;  
DBGrid4: TDBGrid;  
DBChart5: TDBChart;  
BarSeries4: TBarSeries;  
BarSeries5: TBarSeries;  
BarSeries6: TBarSeries;  
BarSeries7: TBarSeries;  
BarSeries8: TBarSeries;  
TabSheet8: TTabSheet;  
GroupBox2: TGroupBox;  
PageControl2: TPageControl;  
TabSheet9: TTabSheet;  
GroupBox3: TGroupBox;  
Label5: TLabel;  
Label4: TLabel;  
Label3: TLabel;  
Label2: TLabel;  
Label6: TLabel;  
EditTurkish: TDBEdit;  
EditSterling: TDBEdit;  
EditDollar: TDBEdit;  
EditEuro: TDBEdit;  
DBEdit6: TDBEdit;  
DBNavigator: TDBNavigator;  
TabSheet10: TTabSheet;



```

GroupBox4: TGroupBox;
Label1: TLabel;
Label7: TLabel;
Label8: TLabel;
Label9: TLabel;
Label10: TLabel;
DBEdit1: TDBEdit;
DBEdit2: TDBEdit;
DBEdit3: TDBEdit;
DBEdit4: TDBEdit;
DBEdit5: TDBEdit;
DBNavigator1: TDBNavigator;
Button2: TButton;
Button6: TButton;
Image3: TImage;
Image4: TImage;
Image5: TImage;
Image6: TImage;
Image7: TImage;
Image8: TImage;
Image9: TImage;
Image10: TImage;
Image11: TImage;
Image12: TImage;
procedure DBEdit23Change(Sender: TObject);
procedure DBEdit29Change(Sender: TObject);
procedure DBEdit32Change(Sender: TObject);
procedure Edit111Change(Sender: TObject);
procedure Edit90Change(Sender: TObject);
procedure Edit5Change(Sender: TObject);
procedure Edit93Change(Sender: TObject);
procedure Edit92Change(Sender: TObject);
procedure Edit66Change(Sender: TObject);
procedure Button1Click(Sender: TObject);

```

```

procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
procedure Button6Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure FormCreate(Sender: TObject);

```

```
private
```

```
    { Private declarations }
```

```
public
```

```
    { Public declarations }
```

```
end;
```

```
var
```

```
    MAINMENUCHANGEEXCHANGESYSTEM:
```

```
    TMAINMENUCHANGEEXCHANGESYSTEM;
```

```
implementation
```

```
uses REPORT, MENU, MENU2, REPORT4, REPORT5;
```

```
{ $R *.dfm }
```

```

procedure TMAINMENUCHANGEEXCHANGESYSTEM.DBEdit23Change(Sender:
TObject);

```

```
    var
```

```
    a,b,c:Currency;
```

```
begin
```

```
if (DBEdit22.gettextlen=0) or (DBEdit23.gettextlen=0) then
```

```
    begin
```

```
        showmessage('please enter value on edits');
```

```
        exit;
```

```
    end;
```

```
    begin
```

```

a:=strtoFloat(DBEdit22.text);
b:=strtoFloat(DBEdit23.text);
c:=a*b;
DBEdit24.text:=Floattostr(c);
end;
end;

```

```

procedure TMAINMENUCHANGEEXCHANGESYSTEM.DBEdit29Change(Sender:
TObject);

```

```

    var
    a,b,c:Currency;
begin
if (DBEdit28.gettextlen=0) or (DBEdit29.gettextlen=0) then
    begin
        showmessage('please enter value on edits');
        exit;
    end;
    begin
a:=strtoFloat(DBEdit28.text);
b:=strtoFloat(DBEdit29.text);
c:=a*b;
DBEdit27.text:=Floattostr(c);
end;
end;

```

```

procedure TMAINMENUCHANGEEXCHANGESYSTEM.DBEdit32Change(Sender:
TObject);

```

```

    var
    a,b,c:Currency;
begin
if (DBEdit31.gettextlen=0) or (DBEdit32.gettextlen=0) then
    begin
        showmessage('please enter value on edits');

```



```

    exit;
end;
begin
a:=strtoFloat(DBEdit31.text);
b:=strtoFloat(DBEdit32.text);
c:=a*b;
DBEdit33.text:=floattostr(c);
end;
end;

```

```

procedure TMAINMENUCHANGEEXCHANGESYSTEM.Edit111Change(Sender:
TObject);
    VAR
a,b,c,d:CURRENCY;
begin
    if (DBEdit109.gettextlen=0)OR (Edit110.gettextlen=0) or (Edit111.gettextlen=0) then
        begin
            showmessage('please enter value on edits');
            exit;
            end;
            begin
a:=strtoFloat(DBEdit109.text);
b:=strtoFloat(Edit110.text);
d:=strtoFloat(Edit111.text);
c:=(a * b)/d;

DBEdit112.text:=floattostr(c);
end;
end;

```

```

procedure TMAINMENUCHANGEEXCHANGESYSTEM.Edit90Change(Sender:
TObject);

```

```

VAR
a,b,c,d:CURRENCY;
begin
  if (DBEdit97.gettextlen=0)OR (Edit89.gettextlen=0) or (Edit90.gettextlen=0) then
    begin
      showmessage('please enter value on edits');
      exit;
      end;
    begin
a:=strtoFloat(DBEdit97.text);
b:=strtoFloat(Edit89.text);
d:=strtoFloat(Edit90.text);
c:= (a * b)/d;

DBEdit98.text:=floattostr(c);
end;
end;

//procedure TMAINMENUCHANGEEXCHANGESYSTEM.Edit4Change(Sender:
TObject);

procedure TMAINMENUCHANGEEXCHANGESYSTEM.Edit5Change(Sender:
TObject);
  VAR
a,b,c,d:CURRENCY;
begin
  if (DBEdit100.gettextlen=0)OR (Edit3.gettextlen=0) or (Edit5.gettextlen=0) then
    begin
      showmessage('please enter value on edits');
      exit;
      end;
    begin
a:=strtoFloat(DBEdit100.text);
b:=strtoFloat(Edit3.text);

```

```
d:=strtoFloat(Edit5.text);
```

```
c:= (a * b)/d;
```

```
DBEdit101.text:=Floattostr(c);
```

```
end;
```

```
end;
```

```
procedure TMAINMENUCHANGEEXCHANGESYSTEM.Edit93Change(Sender:  
TObject);
```

```
VAR
```

```
a,b,c,d:CURRENCY;
```

```
begin
```

```
if (DBEdit103.gettextlen=0)OR (Edit7.gettextlen=0) or (Edit93.gettextlen=0) then
```

```
begin
```

```
showmessage('please enter value on edits');
```

```
exit;
```

```
end;
```

```
begin
```

```
a:=strtoFloat(DBEdit103.text);
```

```
b:=strtoFloat(Edit7.text);
```

```
d:=strtoFloat(Edit93.text);
```

```
c:= (a * b)/d;
```

```
DBEdit104.text:=Floattostr(c);
```

```
end;
```

```
end;
```

```
procedure TMAINMENUCHANGEEXCHANGESYSTEM.Edit92Change(Sender:  
TObject);
```

```
VAR
```

```
a,b,c,d:CURRENCY;
```

```
begin
```

```
if (DBEdit106.gettextlen=0)OR (Edit9.gettextlen=0) or (Edit92.gettextlen=0) then
```



```

begin
  showmessage('please enter value on edits');
exit;
end;

```

```

begin
a:=strtoFloat(DBEdit106.text);
b:=strtoFloat(Edit9.text);
d:=strtoFloat(Edit92.text);
c:=(a * b)/d;
DBEdit107.text:=Floattostr(c);
end;
end;

```

```

procedure TMAINMENUCHANGEEXCHANGESYSTEM.Edit66Change(Sender:
TObject);

```

```

  VAR

```

```

a,b,c,d:CURRENCY;

```

```

begin

```

```

  if (DBEdit94.gettextlen=0)OR (Edit1.gettextlen=0) or (Edit66.gettextlen=0) then

```

```

    begin

```

```

      showmessage('please enter value on edits');

```

```

    exit;

```

```

    end;

```

```

      begin

```

```

a:=strtoFloat(DBEdit94.text);

```

```

b:=strtoFloat(Edit1.text);

```

```

d:=strtoFloat(Edit66.text);

```

```

c:=(a * b)/d;

```

```

DBEdit95.text:=Floattostr(c);

```

```

end;

```

```

end;

```

```
procedure TMAINMENUCHANGEEXCHANGESYSTEM.Button1Click(Sender:
TObject);
begin
//REPORTDOLLAR.QuickRep1.Preview;
SELCHANGESYSTEMPREVIEW.SHOW;
end;
```

```
procedure TMAINMENUCHANGEEXCHANGESYSTEM.Button3Click(Sender:
TObject);
begin
BUYCHANGESYSTEMPREVIEW.SHOW;
end;
```

```
procedure TMAINMENUCHANGEEXCHANGESYSTEM.Button4Click(Sender:
TObject);
begin
BUYCHANGESYSTEM.QuickRep1.Preview;
end;
```

```
procedure TMAINMENUCHANGEEXCHANGESYSTEM.Button5Click(Sender:
TObject);
begin
SELLEXCHANGESYSTEM.QuickRep1.Preview;
end;
```

```
procedure TMAINMENUCHANGEEXCHANGESYSTEM.Button6Click(Sender:
TObject);
begin
SELLEXCHANGESYSTEM.QuickRep1.Print;
end;
```

```
procedure TMAINMENUCHANGEEXCHANGESYSTEM.Button2Click(Sender:
TObject);
begin
```

```
BUYCHANGESYSTEM.QuickRep1.Print;  
end;
```

```
procedure TMAINMENUCHANGEEXCHANGESYSTEM.FormCreate(Sender:  
TObject);  
begin  
  
end;  
  
end.
```

### 3.3 BUY CHANGE SYSTEM PREVIEW

```
unit MENU2;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, ExtCtrls;
```

```
type
```

```
TBUYCHANGESYSTEMPREVIEW = class(TForm)
```

```
  Panel1: TPanel;
```

```
  GroupBox1: TGroupBox;
```

```
  Button3: TButton;
```

```
  Button2: TButton;
```

```
  Button1: TButton;
```

```
  Panel2: TPanel;
```

```
  GroupBox2: TGroupBox;
```

```
  Panel3: TPanel;
```

```
  GroupBox3: TGroupBox;
```



```

Button4: TButton;
Button5: TButton;
Button6: TButton;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
procedure Button6Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    BUYCHANGESYSTEMPREVIEW: TBUYCHANGESYSTEMPREVIEW;

implementation

uses REPORT5DU, REPORT5DS, REPORT5E, REPORT5ES, REPORT5SE,
    REPORT5SD;

{$R *.dfm}
procedure TBUYCHANGESYSTEMPREVIEW.Button1Click(Sender: TObject);
begin
    EXCHANGSYSTEMDOLLARTOEURO.QuickRep1.Preview;
end;

procedure TBUYCHANGESYSTEMPREVIEW.Button2Click(Sender: TObject);
begin
    EXCHANGSYSTEMDOLLARSTERLING.QuickRep1.Preview;
end;

```

```
procedure TBUYCHANGESYSTEMPREVIEW.Button3Click(Sender: TObject);  
begin  
EXCHANGEEUROSYSTEM.QuickRep1.Preview;  
end;
```

```
procedure TBUYCHANGESYSTEMPREVIEW.Button4Click(Sender: TObject);  
begin  
EXCHANGEEUROS.QuickRep1.Preview;  
end;
```

```
procedure TBUYCHANGESYSTEMPREVIEW.Button5Click(Sender: TObject);  
begin  
EXCHANGESYSTEMSTERLINGSTERLING.QuickRep1.Preview;  
end;
```

```
procedure TBUYCHANGESYSTEMPREVIEW.Button6Click(Sender: TObject);  
begin  
EXCHANGESYSTEMSTERLINGDOLLAR.QuickRep1.Preview;  
end;
```

```
procedure TBUYCHANGESYSTEMPREVIEW.FormCreate(Sender: TObject);  
begin  
  
end;  
  
end.
```

### 3.4 EXCHANGE SYSTEM STERLING

```
unit REPORT5SE;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;
```

```
type
```

```
TEXCHANGESYSTEMSTERLINGSTERLING = class(TForm)
```

```
QuickRep1: TQuickRep;
```

```
PageFooterBand1: TQRBand;
```

```
QRExpr1: TQRExpr;
```

```
ColumnHeaderBand1: TQRBand;
```

```
QRLabel1: TQRLabel;
```

```
QRLabel2: TQRLabel;
```

```
QRLabel3: TQRLabel;
```

```
DetailBand1: TQRBand;
```

```
QRExpr2: TQRExpr;
```

```
QRExpr3: TQRExpr;
```

```
QRExpr4: TQRExpr;
```

```
QRLabel4: TQRLabel;
```

```
Table1: TTable;
```

```
procedure FormCreate(Sender: TObject);
```

```
private
```

```
{ Private declarations }
```

```
public
```

```
{ Public declarations }
```

```
end;
```

```
var
```



### 3.5 CHANGE SYSTEM STERLING

implementation

{ \$R \*.dfm }

procedure TEXCHANGESYSTEMSTERLINGSTERLING.FormCreate(Sender:  
TObject);

begin

end;

end.

### 3.6 EXCHANGE SYSTEM STERLING DOLLAR

unit REPORT5SD;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;

type

TEXCHANGESYSTEMSTERLINGDOLLAR = class(TForm)

QuickRep1: TQuickRep;

PageFooterBand1: TQRBand;

QRExpr1: TQRExpr;

ColumnHeaderBand1: TQRBand;

QRLabel1: TQRLabel;

QRLabel2: TQRLabel;

```

QRLabel3: TQRLabel;
DetailBand1: TQRBand;
QRExpr2: TQRExpr;
QRExpr3: TQRExpr;
QRExpr4: TQRExpr;
QRLabel4: TQRLabel;
Table1: TTable;
procedure FormCreate(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    EXCHANGESYSTEMSTERLINGDOLLAR:
    TEXCHANGESYSTEMSTERLINGDOLLAR;
implementation

{$R *.dfm}
procedure TEXCHANGESYSTEMSTERLINGDOLLAR.FormCreate(Sender:
TObject);
begin

end;

end.

```

### 3.7 EXCHANGE SYSTEM DOLLAR EURO



unit REPORT5ED;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;

type

TEXCHANGESYSTEMDOLLAREURO = class(TForm)

QuickRep1: TQuickRep;

PageFooterBand1: TQRBand;

QRExpr1: TQRExpr;

ColumnHeaderBand1: TQRBand;

QRLabel1: TQRLabel;

QRLabel2: TQRLabel;

QRLabel3: TQRLabel;

DetailBand1: TQRBand;

QRExpr2: TQRExpr;

QRExpr3: TQRExpr;

QRExpr4: TQRExpr;

QRLabel4: TQRLabel;

Table1: TTable;

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var



### 3.8 CHANGE SYSTEM DOLLAR EURO

TEXCHANGESYSTEMDOLLAREURO;

implementation

{ \$R \*.dfm }

procedure TEXCHANGESYSTEMDOLLAREURO.FormCreate(Sender: TObject);

begin

end;

end.

### 3.9 EXCHANGE EURO SYSTEM

unit REPORT5E;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;

type

TEXCHANGEEUROSYSTEM = class(TForm)

QuickRep1: TQuickRep;

PageFooterBand1: TQRBand;

QRExpr1: TQRExpr;

ColumnHeaderBand1: TQRBand;

QRLabel1: TQRLabel;

```

QRLabel2: TQRLabel;
QRLabel3: TQRLabel;
DetailBand1: TQRBand;
QRExpr2: TQRExpr;
QRExpr3: TQRExpr;
QRExpr4: TQRExpr;
QRLabel4: TQRLabel;
Table1: TTable;
procedure FormCreate(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    EXCHANGEEUROSYSTEM: TEXCHANGEEUROSYSTEM;

implementation

    {$R *.dfm}

    procedure TEXCHANGEEUROSYSTEM.FormCreate(Sender: TObject);
    begin
        end;

    end.

```

### 3.10 EXCHANGE SYSTEM DOLLAR STERLING

unit REPORT5DS;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;

type

TEXCHANGESYSTEMDOLLARSTERLING = class(TForm)

QuickRep1: TQuickRep;

PageFooterBand1: TQRBand;

QRExpr1: TQRExpr;

ColumnHeaderBand1: TQRBand;

QRLabel1: TQRLabel;

QRLabel2: TQRLabel;

QRLabel3: TQRLabel;

DetailBand1: TQRBand;

QRExpr2: TQRExpr;

QRExpr3: TQRExpr;

QRExpr4: TQRExpr;

QRLabel4: TQRLabel;

Table1: TTable;

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var



### 3.11 CHANGE SYSTEM DOLLAR STERLING

TEXCHANGESYSTEMDOLLARSTERLING;

implementation

{ \$R \*.dfm }

procedure TEXCHANGESYSTEMDOLLARSTERLING.FormCreate(Sender:

TObject);

begin

end;

end.

### 3.12 SELL EXCHANGE SYSTEM

unit REPORT5;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;

type

TSELLEXCHANGESYSTEM = class(TForm)

QuickRep1: TQuickRep;

PageFooterBand1: TQRBand;

QRExpr1: TQRExpr;

ColumnHeaderBand1: TQRBand;

```

QRLabel1: TQRLabel;
QRLabel2: TQRLabel;
QRLabel3: TQRLabel;
QRLabel4: TQRLabel;
QRLabel5: TQRLabel;
DetailBand1: TQRBand;
QRExpr2: TQRExpr;
QRExpr3: TQRExpr;
QRExpr4: TQRExpr;
QRExpr5: TQRExpr;
QRExpr6: TQRExpr;
QRLabel6: TQRLabel;
Table1: TTable;
procedure FormCreate(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  SELLEXCHANGESYSTEM: TSELLEXCHANGESYSTEM;

implementation

{$R *.dfm}

procedure TSELLEXCHANGESYSTEM.FormCreate(Sender: TObject);
begin

end;

end.

```

### 3.13 EXCHANGE EURO TO STERLING

unit REPORT5ES;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;

type

TEXCHANGEEUROS = class(TForm)

QuickRep1: TQuickRep;

PageFooterBand1: TQRBand;

QRExp1: TQRExp;

ColumnHeaderBand1: TQRBand;

QRLabel1: TQRLabel;

QRLabel2: TQRLabel;

QRLabel3: TQRLabel;

DetailBand1: TQRBand;

QRExp3: TQRExp;

QRExp4: TQRExp;

QRLabel4: TQRLabel;

Table1: TTable;

QRExp2: TQRExp;

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var



### 3.14 EXCHANGE EURO

TEXCHANGEEUROS;

implementation

{ \$R \*.dfm }

procedure TEXCHANGEEUROS.FormCreate(Sender: TObject);

begin

end;

end.

### 3.15 EXCHANGE SYSTEM DOLLAR TO EURO

unit REPORT5DU;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;

type

TEXCHANGSYSTEMDOLLARTOEURO = class(TForm)

QuickRep1: TQuickRep;

PageFooterBand1: TQRBand;

QRExpr1: TQRExpr;

ColumnHeaderBand1: TQRBand;

QRLabel1: TQRLabel;

```

QRLabel2: TQRLabel;
QRLabel3: TQRLabel;
DetailBand1: TQRBand;
QRExpr2: TQRExpr;
QRExpr3: TQRExpr;
QRExpr4: TQRExpr;
QRLabel4: TQRLabel;
Table1: TTable;
procedure FormCreate(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  EXCHANGSYSTEMDOLLARTOEURO:
  TEXCHANGSYSTEMDOLLARTOEURO;

implementation

{$R *.dfm}

procedure TEXCHANGSYSTEMDOLLARTOEURO.FormCreate(Sender: TObject);
begin

end;

end.

```

### 3.16 BUY CHANGE SYSTEM PREVIEW

unit MENU2;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, ExtCtrls;

type

TBUYCHANGESYSTEMPREVIEW = class(TForm)

Panel1: TPanel;

GroupBox1: TGroupBox;

Button3: TButton;

Button2: TButton;

Button1: TButton;

Panel2: TPanel;

GroupBox2: TGroupBox;

Panel3: TPanel;

GroupBox3: TGroupBox;

Button4: TButton;

Button5: TButton;

Button6: TButton;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure Button5Click(Sender: TObject);

procedure Button6Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

private

{ Private declarations }



public

{ Public declarations }

end;

var

BUYCHANGESYSTEMPREVIEW: TBUYCHANGESYSTEMPREVIEW;

implementation

uses REPORT5DU, REPORT5DS, REPORT5E, REPORT5ES, REPORT5SE,  
REPORT5SD;

{ \$R \*.dfm }

procedure TBUYCHANGESYSTEMPREVIEW.Button1Click(Sender: TObject);

begin

EXCHANGSYSTEMDOLLARTOEURO.QuickRep1.Preview;

end;

procedure TBUYCHANGESYSTEMPREVIEW.Button2Click(Sender: TObject);

begin

EXCHANGESYSTEMDOLLARSTERLING.QuickRep1.Preview;

end;

procedure TBUYCHANGESYSTEMPREVIEW.Button3Click(Sender: TObject);

begin

EXCHANGEEUROSYSTEM.QuickRep1.Preview;

end;

procedure TBUYCHANGESYSTEMPREVIEW.Button4Click(Sender: TObject);

begin

EXCHANGEEUROS.QuickRep1.Preview;

end;

procedure TBUYCHANGESYSTEMPREVIEW.Button5Click(Sender: TObject);

```

begin
EXCHANGESYSTEMSTERLINGSTERLING.QuickRep1.Preview;
end;

procedure TBUYCHANGESYSTEMPREVIEW.Button6Click(Sender: TObject);
begin
EXCHANGESYSTEMSTERLINGDOLLAR.QuickRep1.Preview;
end;
procedure TBUYCHANGESYSTEMPREVIEW.FormCreate(Sender: TObject);
begin

end;

end.

end;

procedure TBUYCHANGESYSTEMPREVIEW.Button2Click(Sender: TObject);
begin
EXCHANGESYSTEMDOLLARSTERLING.QuickRep1.Preview;
end;

procedure TBUYCHANGESYSTEMPREVIEW.Button3Click(Sender: TObject);
begin
EXCHANGEEUROSYSTEM.QuickRep1.Preview;
end;

procedure TBUYCHANGESYSTEMPREVIEW.Button4Click(Sender: TObject);
begin
EXCHANGEEUROS.QuickRep1.Preview;
end;

procedure TBUYCHANGESYSTEMPREVIEW.Button5Click(Sender: TObject);
begin

```

```
EXCHANGESYSTEMSTERLINGSTERLING.QuickRep1.Preview;  
end;
```

```
procedure TBUYCHANGESYSTEMPREVIEW.Button6Click(Sender: TObject);  
begin  
EXCHANGESYSTEMSTERLINGDOLLAR.QuickRep1.Preview;  
end;
```

```
procedure TBUYCHANGESYSTEMPREVIEW.FormCreate(Sender: TObject);  
begin  
  
end;  
  
end.
```

### 3.17 BUY CHANGE SYSTEM

```
unit REPORT4;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;
```

```
type
```

```
TBUYCHANGESYSTEM = class(TForm)
```

```
QuickRep1: TQuickRep;
```

```
PageFooterBand1: TQRBand;
```

```
QRExpr1: TQRExpr;
```

```
ColumnHeaderBand1: TQRBand;
```

```
QRLabel1: TQRLabel;
```



```

QRLabel2: TQRLabel;
QRLabel3: TQRLabel;
QRLabel4: TQRLabel;
QRLabel5: TQRLabel;
DetailBand1: TQRBand;
QRExpr2: TQRExpr;
QRExpr3: TQRExpr;
QRExpr4: TQRExpr;
QRExpr5: TQRExpr;
QRExpr6: TQRExpr;
QRLabel6: TQRLabel;
Table1: TTable;
procedure FormCreate(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var

```

### 3.18 BUY CHANGE SYSTEM

```
TBUYCHANGESYSTEM;
```

```
implementation
```

```
{ $R *.dfm }
```

```

procedure TBUYCHANGESYSTEM.FormCreate(Sender: TObject);
begin

```

end;

end.

### 3.19 CHANGE SYSTEM STERLING

unit REPORT3;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;

type

TCHANGESYSTEMSTERLING = class(TForm)

QuickRep1: TQuickRep;

PageFooterBand1: TQRBand;

QRExpr1: TQRExpr;

ColumnHeaderBand1: TQRBand;

QRLabel1: TQRLabel;

QRLabel2: TQRLabel;

QRLabel3: TQRLabel;

DetailBand1: TQRBand;

QRExpr2: TQRExpr;

QRExpr3: TQRExpr;

QRExpr4: TQRExpr;

QRLabel4: TQRLabel;

Table1: TTable;

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

```

    { Public declarations }
end;

var
    CHANGESYSTEMSTERLING: TCHANGESYSTEMSTERLING;
implementation

{$R *.dfm}

procedure TCHANGESYSTEMSTERLING.FormCreate(Sender: TObject);
begin
end;

end.

```

### 3.20 CHANGE SYSTEM

```

unit REPORT2;

interface

uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
    Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;

type
    TCHANGESYSTEM = class(TForm)
        QuickRep1: TQuickRep;
        PageFooterBand1: TQRBand;
        QRExpr1: TQRExpr;
        ColumnHeaderBand1: TQRBand;
        QRLabel1: TQRLabel;
    end;

```



```

QRLabel2: TQRLabel;
QRLabel3: TQRLabel;
DetailBand1: TQRBand;
QRExpr2: TQRExpr;
QRExpr3: TQRExpr;
QRExpr4: TQRExpr;
QRLabel4: TQRLabel;
Table1: TTable;
Table1Euro: TFloatField;
Table1Rate: TFloatField;
Table1Date: TDateField;
procedure FormCreate(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    CHANGESYSTEM: TCHANGESYSTEM;

implementation

{$R *.dfm}

procedure TCHANGESYSTEM.FormCreate(Sender: TObject);
begin

end;

end.

```

### 3.21 REPORT DOLLAR

unit REPORT;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls;

type

TREPORTDOLLAR = class(TForm)

QuickRep1: TQuickRep;

PageFooterBand1: TQRBand;

QRExpr1: TQRExpr;

ColumnHeaderBand1: TQRBand;

QRLabel1: TQRLabel;

QRLabel2: TQRLabel;

QRLabel3: TQRLabel;

DetailBand1: TQRBand;

QRExpr2: TQRExpr;

QRExpr3: TQRExpr;

QRExpr4: TQRExpr;

Table1: TTable;

QRLabel4: TQRLabel;

Table1Dollar: TFloatField;

Table1Rate: TFloatField;

Table1Date: TDateField;

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

```

end;

var
    REPORTDOLLAR: TREPORTDOLLAR;

implementation

{$R *.dfm}

procedure TREPORTDOLLAR.FormCreate(Sender: TObject);
begin

end;

end.

```

### 3.22 SELL CHANGE SYSTEM PREVIEW

```

unit MENU;

interface

uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
    Dialogs, StdCtrls, ExtCtrls;

type
    TSELCHANGESYSTEMPREVIEW = class(TForm)
        Panel1: TPanel;
        GroupBox1: TGroupBox;
        Button3: TButton;
        Button2: TButton;
        Button1: TButton;
        Panel2: TPanel;
    end;

```



```

GroupBox2: TGroupBox;
Panel3: TPanel;
GroupBox3: TGroupBox;
procedure Button1Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  SELCHANGESYSTEMPREVIEW: TSELCHANGESYSTEMPREVIEW;

implementation

uses REPORT, REPORT2, REPORT3;

{$R *.dfm}

procedure TSELCHANGESYSTEMPREVIEW.Button1Click(Sender: TObject);
begin
  REPORTDOLLAR.QuickRep1.Preview;
end;

procedure TSELCHANGESYSTEMPREVIEW.Button3Click(Sender: TObject);
begin
  CHANGESYSTEM.QuickRep1.Preview;
end;

procedure TSELCHANGESYSTEMPREVIEW.Button2Click(Sender: TObject);
begin

```

CHANGESYSTEMSTERLING.QuickRep1.Preview;

end;

procedure TSELCHANGESYSTEMPREVIEW.FormCreate(Sender: TObject);

begin

end;

end.

## **CONCLUSION**

Due to my own aim of this project, this program may improve and its features will be promoted later and it will be capable to the usage on network in the future. I have designed this program with using database, which is available in the main features of DELPHI Program. This database is very simple and it can be understood easily with comparing with the other types of database. Later on, I can apply password option, which is not required in the project requirements and thus, it will be more controllable and more efficient. I appreciate all the experience that I have gained from learning and practicing on this programming language and I feel that I will not stop at this point.

## **SYSTEM REQUIREMENTS**

In order to run this program and to setup the settings of this program from any computer, you should have the following specifications:

- ◆ Pentium® II
- ◆ Operating system: Windows 98, 2000 NT, ME, XP Professional
- ◆ Ram: 128 MB
- ◆ Processor: 1000 MHz at least
- ◆ VGA card: 32MB



## REFERENCES

1. DELPHI 4
2. DELPHI 5
3. BORLAND DELPHI 6
4. [www.google.com](http://www.google.com)
5. [www.borland.com](http://www.borland.com)
6. [www.webferret.com](http://www.webferret.com)
7. [www.alltheweb.com](http://www.alltheweb.com)
8. [www.whatis.com](http://www.whatis.com)