PC-Based Data Acquisition system
Using Modbus protocol Over Serial communication interface

A Report submitted in partial fulfillment of the Requirements for the degree of
B.Sc. (HON)
In
Electrical and Electronic Engineering

Under Supervision of
Dr. Abdelrahman Ali Karar

By
Mohammed Abdeen Mohammed Hassan
(0340\3)

To
Department of Electrical and Electronic Engineering
Faculty of Engineering and Architecture
University of Khartoum

July 2008
Dedication

To the spirits of my father and grand mothers

To my mother,

For her love, encouragement and prayers

To my brothers and sisters,

For their love and support

To my friends,

For their kindness, sincerity and help

To my partner Nasreldin,

For his patience and understanding
Acknowledgment

I WOULD LIKE TO EXPRESS MY PROFOUND GRATITUDE TO ALMIGHTY GOD FOR ALL THE PHYSICAL AND MENTAL STRENGTH HE BESTOWED ON ME DURING ALL THE ACTIVITIES OF THE PROJECT AND FINALLY DURING THE PREPARATION OF THIS THESIS.

I AM INDEBTED TO DR.ABD_ELRAHMAN ALI KARAR, WHO SUPERVISED MY PROJECT WITH ALL SUPPORT, PATIENCE AND DEDICATION.

FINALLY, I WOULD LIKE TO THANK MY FAMILY FOR THEIR UNLIMITED ENCOURAGEMENT AND SUPPORT IN ALL MY ENDEAVORS.
The objective of this project was to build an inexpensive, accurate and acceptable-speed data acquisition and control system, based on computer and serial interface.

The project has made a great use of the features of computer (large memory, processing speed and graphical user interface (GUI)) and of the stand alone logger (reliability and not susceptible noise).

A simulation was used to simulate the communication between master and slave devices using two computers.

Then a real-time process have been develop by using computer holding visual basic program represent master and a stand-alone logger device (A 210 Multifunctional Power Monitor) represent slave.

The serial interface used was RS-232 and the protocol was Modbus.

The project was able to achieve the specified goals.

Recommendations were made for further increase in speeds as well as network scale.
المستخصص

يهدف هذا المشروع لبناء نظام دقيق لجمع البيانات باقل التكاليف وبسرعة مناسبة معتمداً على الحاسوب وكيل (وصلة) متسلسل.

يقوم المشروع بالإستفادة من خصائص الحاسوب المتمثلة في كبر سعة الذاكرة وسرعة المعالجة وخصائص الرسم، وكذلك خصائص المسجل المستقل المتمثلة في إعتمادية الأداء ومقاومته للتشويش. تم تمثيل عملية الإتصال بين جهازين أحدهما خادم والآخر مخدم لتحقيق الهدف المنشود.

صممت العملية الحقيقية باستخدام جهاز حاسوب وبرنامج مكتوب بلغة ال (Visual Basic) باعتباره خادماً بالإضافة إلى مسجل مستقل (جهاز قراءة القدرة الكهربائية) باعتباره مخدم كما أن الوصلة المتواجدة التي أستخدمت في هذه العملية هي (232 – بروتوكول Modbus) برروتوكول RS حفلت أهداف المشروع بنجاح، كما تم وضع توصيات لمزيد من التطوير لزيادة سرعة النظام وتوسيعه ليصبح شبكة متكاملة.
# TABLE OF CONTENTS

Dedication...............................................................................................................................
Acknowledgement .................................................................................................................I
Abstract ................................................................................................................................II
اﻟﻤﺴﺘﺨﻠﺺ................................................................................................................................III
List of contents .......................................................................................................................IV
List of Figures .........................................................................................................................VI
List of Tables ....................................................................................................................... VII
List of Abbreviations .............................................................................................................VIII

Chapter 1: Introduction
  1.1 PC-Based data acquisition systems ............................................................. 1
  1.2 Data acquisition hardware ............................................................................. 1
  1.3 Data acquisition software .............................................................................. 2
  1.4 Fieldbus protocols ........................................................................................ 3
  1.5 Project Objectives .......................................................................................... 3
  1.6 Thesis Layout ................................................................................................... 3

Chapter 2: Theory
  2.1 Data acquisition and control......................................................................... 4
  2.2 Fundamentals of A data acquisition ........................................................... 4
      2.2.1 Sensors and transducers .................................................................... 5
      2.2.2 Field wiring ....................................................................................... 5
      2.2.3 Signal conditioning ............................................................................. 6
      2.2.4 Data acquisition hardware ................................................................. 6
      2.2.5 Data acquisition software ................................................................. 7
      2.2.6 Host computer .................................................................................... 7
  2.3 Data acquisition and control system configuration .................................... 7
      2.3.1 Computer plug-in I/O ...................................................................... 8
      2.3.2 Distributed I/O .................................................................................. 9
      2.3.3 Stand-alone or distributed loggers and controllers ................... 10
  2.4 Considerations in using stand-alone logger/controllers .............................. 11
  2.5 Serial data communication ........................................................................ 12
      2.5.1 Interface standards ......................................................................... 13
          2.5.1.1 RS-232 interface standard ...................................................... 13
          2.5.1.2 RS-485 interface standard ...................................................... 17
          2.5.1.3 Serial interface converter ....................................................... 18
      2.5.2 Protocols ............................................................................................ 19
          2.5.2.1 Field bus Protocols ................................................................. 19
              2.5.2.1.1 Modbus protocol ............................................................. 19
                  2.5.2.1.1.1 Protocol Description .............................................. 19
                  2.5.2.1.1.2 Data Encoding ......................................................... 21
                  2.5.2.1.1.3 Data Model ............................................................. 22
                  2.5.2.1.1.4 Four Main Function Codes ................................... 22
                  2.5.2.1.1.5 Byte format in RTU mode ..................................... 23
                  2.5.2.1.1.6 MODBUS Message RTU Framing ...................... 23
Chapter 3: Design and Implementation
3.1 simulation of master/slave connection Introduction .......................... 25
   3.1.1 Description of the software.......................................................... 25
   3.1.2 Basic Simulation Configuration.................................................... 25
   3.1.3 Simulation Parameters Adjustment ........................................... 25
      3.1.3.1 Communication parameters.............................................. 25
      3.1.3.2 Master/slave definitions ................................................... 26
3.2 Reading of measurands from multifunctional power meter device .. 27
   3.2.1 SINEAX A 210 Multifunctional Power Monitor......................... 28
   3.2.2 Features of A 210 Multifunctional Power Monitor ................... 28
   3.2.3 Extension module (EMMOD 201 module) .................................. 28
   3.2.4 Setting of SINEAX A 210 Multifunctional Power Monitor... 30
   3.2.5 Modbus Master VB software.................................................... 31
   3.2.6 VB program flowchart............................................................... 32
   3.2.7 Application design..................................................................... 33

Chapter 4: Tests and Results
4.1 Results of Monitoring bytes traffic.................................................... 34
   4.1.1 Read coils status (function code 01) RTU mode......................... 34
   4.1.2 Read input status(function code 02) RTU mode......................... 35
   4.1.3 Read holding registers(function code 03) RTU mode............... 36
   4.1.4 Read input registers (function code 04) RTU mode................. 37
4.2 Results of A200 plus software......................................................... 38
4.3 Results of VB master software......................................................... 39

Chapter 5: Conclusions and Recommendations
5.1 Conclusions ..................................................................................... 41
   5.1.1 Benefits .................................................................................... 41
   5.1.2 Limitations .............................................................................. 41
5.2 Recommendations and future work .............................................. 42

References................................................................................................ 43
Appendix A: Devices figures .................................................................... 44
Appendix B: VB program code............................................................... 45
LIST OF FIGURES

Fig. 2-1 Basic elements of DAQ system .................................................................5
Fig. 2-2 Computer plug-in I/O DAQ system .........................................................8
Fig. 2-3 Distributed I/O DAQ system .................................................................10
Fig. 2-4 Stand alone logger/controller DAQ system ...........................................11
Fig. 2-5 RS-232 interface between DTE and DCE .............................................13
Fig. 2-6 Pin description of DTE and DCE serial interface ..............................14
Fig. 2-7 Serial interface repeater ..........................................................................18
Fig. 2-8 Modbus protocol description ............................................................20
Fig. 2-9 RTU Message Frame ..........................................................................23

Fig. 3-1 Master communication parameter setting ........................................26
Fig. 3-2 Slave communication parameter setting .............................................26
Fig. 3-3 Master definitions .................................................................................27
Fig. 3-4 Slave definitions ...................................................................................27
Fig. 3-5 Energy meter setting using A200 plus software ..............................30
Fig. 3-6 VB program flow chart .............................................................32
Fig. 3-7 application circuit ..............................................................................33

Fig. 4-1 coil status of modbus slave .................................................................34
Fig. 4-2 coil status and byte traffic received by modbus Master ................35
Fig. 4-3 discrete input of modbus slave ..........................................................35
Fig. 4-4 discrete input status received by modbus Master ...........................36
Fig. 4-5 holding registers of modbus slave.....................................................36
Fig. 4-6 holding registers received by modbus Master ..................................37
Fig. 4-7 input registers of modbus slave .........................................................37
Fig. 4-8 input registers received by modbus Master .....................................38
Fig. 4-9 A200 plus software interface ..........................................................39
Fig. 4-10 VB program interface ..................................................................40
LIST OF TABLES

Table 2-1 pin description of DB-9 connector ..............................................16
Table 2-2 Modbus Data model .................................................................22
Table.2-3 Modbus function code ..............................................................22
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUI</td>
<td>Graphical User Interface.</td>
</tr>
<tr>
<td>DAQ</td>
<td>Data Acquisition.</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output.</td>
</tr>
<tr>
<td>A/D</td>
<td>Analog to Digital converter.</td>
</tr>
<tr>
<td>D/A</td>
<td>Digital to Analog converter.</td>
</tr>
<tr>
<td>RTU</td>
<td>Remote Terminal Unit.</td>
</tr>
<tr>
<td>ASCII</td>
<td>American Standard Code for Information Interchange.</td>
</tr>
<tr>
<td>DTE</td>
<td>Data Terminal Equipment.</td>
</tr>
<tr>
<td>DCE</td>
<td>Data Communication Equipment.</td>
</tr>
<tr>
<td>EIA</td>
<td>Electronic Industries Association.</td>
</tr>
<tr>
<td>PDU</td>
<td>Protocol Data Unit.</td>
</tr>
<tr>
<td>ADU</td>
<td>Application Data Unit.</td>
</tr>
</tbody>
</table>
1.1 PC-Based data acquisition systems:

Personal computers (PCs) provide high speed, accuracy, flexibility, adaptability, consistency, reliability and mass memory. These features support the mathematics, analysis, storage, display, report generation, control, and communications functions required in today’s applications. However, most real-world signals (temperature, pressure, flow, speed, intensity, position, etc.) cannot be read directly by personal computers without the use of a PC-based data acquisition and control system.

Data acquisition (DAQ) with a PC enables one to display, log and control a wide variety of real world signals. This ability coupled with that of easy interface with various stand-alone instruments and hardware's makes the systems ever more desirable.

1.2 Data acquisition hardware:

Data acquisition and control hardware can be classified as external bus or internal bus systems, depending on the method used to interface the hardware with the PC.

External bus systems may reside in their own enclosures outside the PC. The enclosure provides space for the analog and digital I/O hardware, as well as control circuitry or microcomputer. A power supply may also reside in the enclosure. These systems interface with the PC via a parallel port (standard or enhanced), a serial port (RS-232 or RS422), or an IEEE-488 port. Many external systems also facilitate remote operation. The data acquisition and control system can be placed at a site that is remote from the host computer (and thus close to the field signals). External data acquisition systems can also support a distributed system, in which a large number of parameters can be monitored or controlled, even though they are located physically far from each other and the host computer. The data acquisition system has been broadly used in agricultural application for a few decades. These areas include precision agriculture, weather station, environment control, food processing and safety, air quality, and equipment automation.

The internal bus system is plugged directly into the internal computer expansion bus. There are several advantages associated with internal systems. Because an internal
system can directly use the computer’s resources to achieve its tasks, very high sampling speeds can be reached. Internal systems are lower in cost because the data acquisition and control system can use resources already within the computer. Since the system can use a number of components or functions already in the computer, it does not need its own version of these hardware and software resources. As an example, power can be obtained from the PC power supply; the internal bus system does not need its own power supply. Because an internal bus system is not easy to transfer from one computer to another, the external bus system was selected for the development of a data acquisition and control system. The RS-232 serial port is a universal feature of all types of computers and was selected as the communication route for the new data acquisition system.

1.3 Data acquisition software:

It usually comes with the DAQ hardware or from other vendors, which allows the operating system and various programs to recognize the DAQ hardware and to access the signals being read by the DAQ hardware.

Good Data acquisition software offers high and low level access; one would start out with the high level solutions offered and improves down to assembly instructions in time critical or exotic applications.

Acquired data are displayed, analyzed, and stored on a computer, either using vendor supplied software, or custom displays and control can be developed using various general purpose programming languages such as BASIC, C, Fortran, Java, Lisp, Pascal. Specialized programming languages used for data acquisition include, EPICS used to build large scale data acquisition systems, LabVIEW, which offers a graphical programming environment optimized for data acquisition and MATLAB provides a programming language but also built-in graphical tools and libraries for data acquisition and analysis.

The key to the effective application of PC-based data acquisition is the careful matching of real world requirements with appropriate hardware and software. Depending on your needs, monitoring data can be as simple as connecting a few cables to a plug-in
board and running a menu-driven software package. At the other end of the spectrum, you could design customized sensing and conversion hardware, or perhaps develop application software to optimize a system.

1.4 Fieldbus protocols:

The fieldbuses and their protocols are used today primarily as the communication system for exchange of information between automation systems and distributed field devices. All leading manufacturers of automation technology offer fieldbus interfaces for their devices. That is why the fieldbus systems present a very dynamically growing branch of the industry.

1.5 Project Objectives:

- The main objective of this project is to design and implement PC-based data acquisition systems and to facilitate design, specification, installation, configuration, and programming of data acquisition systems quickly and effectively.
- We have directed our work towards modbus protocol in order to fulfill our objectives. To achieve this objective, work was divided into two sub-objectives:
  - The first objective is to monitor the byte traffic between a master device and slave device as simulation.
  - The second objective is to build a modbus master that can communicate with any other modbus slave and read its holding registers as real-time process.

1.6 Thesis Layout:

This thesis is organized as follows:

Chapter 2: Introduces the concepts of the DAQ systems and it is configurations.

Chapter 3: Describes the design and VB program implementation.

Chapter 4: Describes the results that obtain from our design.

Chapter 5: Presents conclusions and recommended future work.

Appendix A: Contains devices figures.

Appendix B: Contains VB code.
2.1 Data acquisition and control:

Data acquisition is the process by which physical phenomena from the real world are transformed into electrical signals that are measured and converted into a digital format for processing, analysis, and storage by a computer.

In a large majority of applications, the data acquisition system is designed not only to acquire data, but to act on it as well. In defining DAQ systems, it is therefore useful to extend this definition to include the control aspects of the total system.

Control is the process by which digital control signals from the system hardware are convened to a signal format for use by control devices such as actuators and relays. These devices then control a system or process. Where a system is referred to as a data acquisition system, it is possible that it includes control functions as well.

2.2 Fundamentals of A data acquisition:

A data acquisition and control system, built around the power and flexibility of the PC, may consist of a wide variety of diverse hardware building blocks from different equipment manufacturers. It is the task of the system integrator to bring together these individual components into a complete working system. The basic elements of a data acquisition system, as shown in the functional diagram of Figure (2.1), are as follows:

- Sensors and transducers
- Field wiring
- Signal conditioning
- Data acquisition hardware
- Data acquisition software
- PC (operating system)
2.2.1 Sensors and transducers:

Transducers and sensors provide the actual interface between the real world and the data acquisition system by converting physical phenomena into electrical signals that the signal conditioning and/or data acquisition hardware can accept.

2.2.2 Field wiring:

Field wiring represents the physical connection from the transducers and sensors to the signal conditioning hardware and/or data acquisition hardware. When the signal conditioning and/or data acquisition hardware is remotely located from the PC, then the
field wiring provides the physical link between these hardware elements and the host
computer. If this physical link is an RS-232 or RS-485 communications interface, then
this component of the field wiring is often referred to as communications cabling.

2.2.3 Signal conditioning:

Electrical signals generated by transducers often need to be converted to a form
acceptable to the data acquisition hardware, particularly the A/D converter which
converts the signal data to the required digital format. In addition, many transducers
require some form of excitation or bridge completion for proper and accurate operation.
The principal tasks performed by signal conditioning are:
• Filtering
• Amplification
• Linearization
• Isolation
• Excitation

2.2.4 Data acquisition hardware:

Data acquisition and control hardware can be defined as that component of a
complete data acquisition and control system, which performs any of the following
functions:
• Accept analog signal measured from a process, processing and conversion to
digital format, using ADCs, the data is then transferred to a computer for display,
storage and analysis.
• Accept digital signal from computer, processing conversion to analog format
using DACs, the analog control signals are used for controlling a system or
process.
• The input of digital signals, which contain information from a system or process.
• The output of digital control signals.
2.2.5 Data acquisition software:

Data acquisition hardware does not work without software, because it is the software running on the computer that transforms the system into a complete data acquisition, analysis, display, and control system. The application software can be a full screen interactive panel, a dedicated input/output control program, a data logger, a communications handler, or a combination of all of these.

2.2.6 Host computer:

The PC used in a data acquisition system can greatly affect the speeds at which data can be continuously and accurately acquired, processed, and stored for a particular application.

2.3 Data acquisition and control system configuration:

In many applications, and especially for data acquisition and process control, the power and flexibility of the PC, allows DAQ systems to be configured in a number of ways, each with its own distinct advantages. The key to the effective use of the PC is the careful matching of the specific requirements of a particular data acquisition application to the appropriate hardware and software available. The choice of hardware, and the system configuration, is largely dictated by the environment in which the system will operate. The number of sensors and actuators required and their physical location in relation to the host computer, the type of signal conditioning required, and the harshness of the environment, are key factors.

Several of the most common system configurations are as follows:

- Computer plug-in I/O.
- Distributed I/O.
- Stand-alone or distributed loggers and controllers.
2.3.1 Computer plug-in I/O:

Plug-in I/O boards are plugged directly into the computer's expansion bus, are generally compact, and also represent the fastest method of acquiring data to the computer's memory and/or changing outputs. Along with these advantages, plug-in boards often represent the lowest cost alternative for a complete data acquisition and control system and are therefore a commonly utilized item of DAQ hardware.

As shown in Figure (2.2), examples of plug-in I/O boards are, multiple analog input A/D boards, multiple analog output D/A boards, digital I/O boards, counter/timer boards, specialized controller boards (such as stepper/servo motor controllers) or specialized instrumentation boards (such as digital oscilloscopes).

Figure 2.2 computer plug-in I/O DAQ system
Multi-function DAQ boards, containing A/D converters, D/A converters, digital I/O ports and counter timer circuitry, perform all the functions of the equivalent individual specialized boards. Depending on the number of analog inputs/outputs and digital inputs/outputs required for a particular application, multi-function boards represent the most cost effective and flexible solution for DAQ systems.

Plug-in expansion boards are commonly used in applications where the computer is close to the sensors being measured or the actuators being controlled.

2.3.2 Distributed I/O:

Often sensors must be remotely located from the computer in which the processing and storage of the data takes place. This is especially true in industrial environments where sensors and actuators can be located in hostile environments over a wide area, possibly hundreds of meters away. In noisy environments, it is very difficult for very small signals received from sensors such as thermocouples and strain gauges (in the order of mV) to survive transmission over such long distances, especially in their raw form, without the quality of the sensor data being compromised. An alternative to running long and possibly expensive sensor wires, is the use of distributed I/O, which is available in the form of signal conditioning modules remotely located near the sensors to which they are interfaced. One module is required for each sensor used, allowing for high levels of modularity (single point to hundreds of points per location). While this can add reasonable expense to systems with large point counts, the benefits in terms of signal quality and accuracy may be worth it.

One of the most commonly implemented forms of distributed I/O is the digital transmitter. These intelligent devices perform all required signal conditioning functions (amplification, filtering, isolation etc), contain a micro-controller and A/D converter, to perform the digital conversion of the signal within the module itself. Converted data is transmitted to the computer via an RS-232 or RS-485 communications interface. The use of RS-485 multi-drop networks, as shown in Figure (2.3), reduces the amount of cabling required, since each signal-conditioning module shares the same cable pair. Linking up to 32 modules, communicating over distances up to 10 km, is possible when using the RS-
485 multi-drop network. However, since very few computers have built in support for the RS-485 standard, an RS-232 to RS-485 converter is required to allow communications between the computer and the remote modules.

2.3.3 Stand-alone or distributed loggers and controllers:

As well as providing the benefits of intelligent signal conditioning modules, and the ability to make decisions remotely, the use of stand-alone loggers/controllers increases system reliability. This is because once programmed, the stand-alone logger can continue to operate, even when the host computer is not functional or connected. In fact, standalone loggers/controllers are specifically designed to operate independently of the host computer. This makes them especially useful for applications where the unit must be located in a remote or particularly hostile environment, (e.g. a remotely located weather station), or where the application does not allow continuous connection to a computer (e.g. controlling temperatures in a refrigerated truck). Stand-alone loggers/controllers are intelligent powerful and flexible devices, easily interfaced to a wide range of transducers, as well as providing digital inputs and digital control outputs for process control.
The most commonly used serial communications link for direct connection between the computer and the stand-alone logger/controller is the RS-232 serial interface. This allows programming and data logging up to distances of 50 meters, as shown in Figure (2.4) where the stand-alone unit must be located remotely, a portable PC can be taken to the remote location.

![Figure 2.4 stand alone logger/controller DAQ system](image)

Where an application requires more than one logger/controller, each unit is connected within an RS-485 multi-drop network. A RS-232 to RS-485 converter is required to enable this network to connect to the host computer.

2.4 Considerations in using stand-alone logger/controllers:

Data acquisition and control using stand-alone logger/controllers is an orderly process. When designing a system that utilizes these devices, users should consider the following:

- The first is the number of sensors and control outputs required, and their location in relation to the host PC. This determines how many logger/controllers are required and their location.
- The volume of data that is to be logged. Stand-alone logger/controllers only have a limited amount of memory. How quickly the memory becomes full depends on the
number of input channels to be read and their scan rate. If only a limited amount of data is to be stored then the internal memory may be sufficient. Increased capacity can be provided by a memory card.

- How often stored data must be uploaded to the host PC. Equally important as memory considerations, upload of data to a host PC, is largely determined by the type of application. Logging applications in which the data are gathered for analysis over an extended period (e.g. weather monitoring applications), clearly do not require constant uploading. The frequency of uploading would only depend on the memory capacity of the device and the amount of data being logged. Critical applications requiring constant uploading of process data, feedback of alarms and on-line control would need to be directly connected to the host PC.
- The availability of power. Where mains power is not available, the standalone logger/controller is powered from a battery supply. The battery life is determined by a number of factors and is not unlimited. In time, the battery will need to be charged or replaced.

### 2.5 Serial data communication:

All data communications systems have the following components:

- The source of the data (e.g. a computer). Also required is circuitry that converts the signal into one that is compatible with the communications link, called a transmitter or line driver.
- The communications link (twisted-pair cable, coaxial cable, radio, telephone network etc), which transfers the message to the receiver at the other end.
- The receiver of the data where the signal is converted back into a form that can be used by the local electronics circuitry.

Both the receiver and the transmitter must agree on a number of different factors to allow successful communications between them, the most important being:

- The type of electrical signals used to transmit the data
- The type of codes used for each symbol being transmitted
- The meaning of the characters
• How the flow of data is controlled
• How errors are detected and corrected

The hardware rules that apply to the physical interface and its connections are known as interface standards, while the software rules which apply to the format and control of data flow and the detection and correction of errors are generally referred to as the protocol.

2.5.1 Interface standards:

Communications interface standards define the electrical and mechanical details that allow communication equipment from different manufacturers to be connected together and to function efficiently. Two standards are commonly employed for communications between PCs and stand-alone or distributed logger/controllers:
• RS-232 standard
• RS-485 standard

2.5.1.1 RS-232 interface standard:

The Electronic Industries Association (EIA) RS-232 interface standard is probably the most widely known of all serial data interface standards. It was developed for Interfacing between Data Terminal Equipment (DTE) and Data Communications Equipment (DCE) employing serial binary data interchange.

Figure 2.5 RS-232 interface between DTE and DCE
Figure 2.6 pin description of DTE and DCE serial interface

At the RS-232 receiver the following signal voltage levels are defined:

- +3 V to +25 V for transmission of logic 0
- −3 V to −25 V for transmission of logic 1
- +3 V to −3 V for an undefined logic level

To meet these voltage requirements at the receiver and to overcome any voltage drops that occur along the communications lines, the RS-232 transmitter must produce slightly higher voltages.

These are in the range:

- +5 V to +25 V for transmission of logic 0
- −5 V to −25 V for transmission of logic 1
- +5 V to −5 V for an undefined logic level
The EIA-232 standard defines twenty-five (25) electrical connections; these electrical connections are divided into the four groups shown below:

- Data lines (pin2, pin3).
- Timing lines.
- Special secondary functions.
- Control lines: Control lines are used for interactive device control, commonly known as ‘hardware handshaking’, and regulate the way in which data flows across the interface. The four most commonly used control lines are as follows:
  - RTS – request to send
  - CTS – clear to send
  - DSR – data set ready (or DCE ready in EIA-232-D/E)
  - DTR – data terminal ready (or DTE Ready in EIA-232-D/E)

It is important to note that the handshake lines operate in the opposite voltage sense to the data lines. When a control line is active (logic 1), the voltage is in the range +3 to +25 volts and when deactivated (logic 0), the voltage is zero or negative. Hardware handshaking is usually the cause of most of the interfacing problems. Manufacturers sometimes omit certain of these control lines from their EIA-232 equipment or assign unconventional purposes to them. Consequently, many applications do not use hardware handshaking but instead use only the three data lines (transmit, receive and signal common ground) with some form of software handshaking. The control of data flow is then part of the application program. so the smaller DB-9 connector (9 pin D-type) is commonly used.
Main features of the RS-232 interface standard:

- Communication is point-to-point.
- They are suitable for serial, binary, digital, data communication (data is sent bit by bit in sequence).
- Most EIA-232-C communications data is in the ASCII code, although that is not part of the standard.
- Communication is asynchronous (fixed timing between data bits, but variable time between character frames).
- Communication is full-duplex (both directions simultaneously) with a single wire for each direction and a common wire.
- Voltage signals are:
  1. Logic 1: –3 volts to –25 volts
  2. Logic 0: +3 volts to +25 volts

<table>
<thead>
<tr>
<th>Pin no.</th>
<th>DB-9 connector IBM 232 pin assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Received line signal</td>
</tr>
<tr>
<td>2</td>
<td>Received data</td>
</tr>
<tr>
<td>3</td>
<td>Transmitted data</td>
</tr>
<tr>
<td>4</td>
<td>DTE ready</td>
</tr>
<tr>
<td>5</td>
<td>Signal common/ground</td>
</tr>
<tr>
<td>6</td>
<td>DCE ready</td>
</tr>
<tr>
<td>7</td>
<td>Request to send</td>
</tr>
<tr>
<td>8</td>
<td>Clear to send</td>
</tr>
<tr>
<td>9</td>
<td>Ring indicator</td>
</tr>
</tbody>
</table>
• They provide reliable communication up to about 15 m.
• Data rates of up to about 20 kbps are possible.

Main limitation of the RS-232 interface standard:
• The point-to-point restriction is a severe limitation when several ‘smart’ instruments are used.
• The distance limitation of 15 m end to end is too short for most control systems. The 115 kbps rate is too slow for many applications.
• The –3 to –25 volts and +3 to +25 volts signal levels are not directly compatible with the modem standard power supplies in computers of ±5 volts and ±12 volts.

2.5.1.2 RS-485 interface standard:

The EIA RS-485 is the most versatile of the EIA standards, and is an expansion of the RS-422 standard. The RS-485 standard was designed for two-wire, half duplex, balanced multi-drop communications, and allows up to 32 line drivers and 32 line receivers on the same line. It incorporates the advantages of balanced lines with the need for only two wires (plus signal common) cabling.

RS-485 provides reliable serial communications for:
• Distances of up to 1200 m
• Data rates of up to 10 Mbps
• Up to 32 line drivers permitted on the same line
• Up to 32 line receivers permitted on the same line

The line voltages range between –1.5 V to –6 V for logic ‘1’ and +1.5 V to +6 V for logic ‘0’. The line driver for the RS-485 interface produces a 5 V differential voltage on two wires. For full-duplex systems, four wires are required. For a half-duplex system, only two wires are required.

A major enhancement of RS-485 is that a line driver can operate in three states (called tri-state operation), logic ‘0’, and logic ‘1’ and high-impedance. In the high-impedance state, the line driver draws virtually no current and appears to be disconnected from the line. This ‘disabled’ state can be initiated by a control pin on the line driver integrated circuit. This feature allows multi-drop operation where up to 32 line drivers
can be connected on the same line, although only one line driver can be active at any one time. Each terminal in a multi-drop system must therefore be allocated a unique address to avoid any conflict with other devices on the system. RS-485 includes current limiting in cases where contention occurs.

2.5.1.3 Serial interface converter:

Interface converters are increasingly important today with the movement away from RS-232 to the industrial interface standards such as RS-485. Since many industrial devices still use RS-232 ports, it is necessary to use these converters to interface the device to other network standards. In addition, interface converters are sometimes used to increase the effective distance between two RS-232 devices, especially in noisy environments. The block diagram of an RS-232 / RS-485 converter is shown in Figure below.

![Figure 2.7 serial interface repeater](image)
2.5.2 Protocols:

A protocol is essentially a common set of rules governing the exchange of data between the transmitter and receiver of a communications network, and is normally associated with the packaging of data transmitted on the communications interface.

2.5.2.1 Field bus Protocols:
The most popular fieldbus are used in industrial today are:

- Modbus protocol.
- canbus protocol.
- profibus protocol

2.5.2.1.1 Modbus protocol:

Modbus protocol is a messaging structure created by MODICON company to connect PLC to programming tools. It is now widely used to establish master-slave communication between intelligent devices. Modbus is independent of the physical layer. It can be implemented using RS232, RS422, or RS485 or over a variety of media (e.g. fiber, radio, cellular, etc...).

The Modbus protocol comes in 2 versions:

- ASCII transmission mode:
  Each eight-bit (byte) in a message is sent as one ASCII characters.
- RTU transmission mode:
  Each eight-bit in a message is sent as two (four-bit) hexadecimal characters.

- The main advantage of the RTU mode is that it achieves higher throughput.

2.5.2.1.1.1 Protocol Description:
The Modbus protocol defines a simple protocol data unit (PDU) independent of the underlying communication layers. The mapping of Modbus protocol on specific buses or network can introduce some additional fields on the application data unit (ADU).
From the above figure we can see that there are four main fields:

1) Address Field:
   Valid slave device addresses are in the range of (0 ... 247) decimal. The individual slave devices are assigned addresses in the range of (1 ... 247). Value 0 is reserved for broadcast messages.
   - Request:
     A master addresses a slave by placing the slave address in the address field of the message.
   - Response:
     When the slave sends its response, it places its own address in this address field of the response to let the master know which slave is responding.

2) Function code Field:
   The function code field of a Modbus data unit is coded in one byte. Valid codes are in the range of (1 ... 255) decimal (128 – 255 reserved for exception responses). When a message is sent from a Client to a Server device the function code field tells the server what kind of action to perform.
   - Request:
     The function code field tells the slave what kind of action to perform.
   - Response:
     For a normal response, the slave simply echoes the original function code. For an exception response; the slave returns a code that is equivalent to the original function code with its most significant bit set to logic 1.
3) Data Field:

Valid size is in the range of 0 ... 252 Byte.

- Request:
  
  The data field contains additional information which the slave must use to take
  the action defined by the function code.

- This includes items like register addresses, quantity of items, etc...

- Response:

  If no error occurs, the data field contains the data requested. If an error occurs, the
  field contains an exception code that the master application can use to determine the next
  action to be taken.

4) Error check Field:

  It contains the checksum value.

- Request:

  The checksum is calculated by the master and sends to the slave.

- Response:

  The checksum is re-calculated by the slave and compared to the value sent by the
  master. If a difference is detected, the slave will not construct a response to the master.

2.5.2.1.1.2 Data Encoding:

Modbus uses a ‘big-Endian’ representation for addresses and data items. This
means that when a numerical quantity larger than a single byte is transmitted, the most
significant byte is sent first. So for example:

<table>
<thead>
<tr>
<th>Register size</th>
<th>value</th>
<th>the first byte sent is</th>
<th>then 0x34.</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 - Bits</td>
<td>0x1234</td>
<td>0x12</td>
<td>0x12</td>
</tr>
</tbody>
</table>
2.5.2.1.1.3 Data Model:

Modbus bases its data model on a series of tables that have distinguishing characteristics. The four primary tables are:

<table>
<thead>
<tr>
<th>Primary tables</th>
<th>Object type</th>
<th>Type of</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discretes Input</td>
<td>Single bit</td>
<td>Read-Only</td>
<td>This type of data can be provided by an I/O system.</td>
</tr>
<tr>
<td>Coils</td>
<td>Single bit</td>
<td>Read-Write</td>
<td>This type of data can be alterable by an application program.</td>
</tr>
<tr>
<td>Input Registers</td>
<td>16-bit word</td>
<td>Read-Only</td>
<td>This type of data can be provided by an I/O system</td>
</tr>
<tr>
<td>Holding Registers</td>
<td>16-bit word</td>
<td>Read-Write</td>
<td>This type of data can be alterable by an application program.</td>
</tr>
</tbody>
</table>

- For each of the primary tables, the protocol allows individual selection of 65536 data items.
- It’s obvious that all the data handled via Modbus (bits, registers) must be located in device application memory.
- Modus logical reference numbers, which are used in Modbus functions, are unsigned integer indices starting at zero.

2.5.2.1.1.4 Four Main Function Codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Modbus Function</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>01H</td>
<td>Read coil status</td>
<td>0xxxx</td>
</tr>
<tr>
<td>02H</td>
<td>Read input status</td>
<td>1xxxx</td>
</tr>
<tr>
<td>03H</td>
<td>Read holding register</td>
<td>4xxxx</td>
</tr>
<tr>
<td>04H</td>
<td>Read discrete input</td>
<td>3xxxx</td>
</tr>
</tbody>
</table>
2.5.2.1.1.5 Byte format in RTU mode:

The format for each byte (11 bits) in RTU mode is:
- **Coding System:** 8-bit binary
- **Bits per Byte:**
  - 1 start bit
  - 8 data bits, least significant bit sent first
  - 1 bit for parity completion
  - 1 stop bit

Even parity is required; other modes (odd parity, no parity) may also be used. In order to ensure a maximum compatibility with other products.

Remark: the use of no parity requires 2 stop bits.

### 2.5.2.1.1.6 MODBUS Message RTU Framing:

A MODBUS message is placed by the transmitting device into a frame that has a known beginning and ending point. This allows devices that receive a new frame to begin at the start of the message, and to know when the message is completed.

![MODBUS Message Frame](image1)

In RTU mode, message frames are separated by a silent interval of at least 3.5 characters.
- The entire message frame must be transmitted as a continuous stream of characters.
• If a silent interval of more than 1.5 character times occurs between two characters, the message frame is declared incomplete and should be discarded by the receiver.
3.1 simulation of master/slave connection:

One of the objectives of this project is to monitor the byte traffic (frames) between master and slave devices that communicate via modbus protocol. To achieve this objective, two software packages were used:

1. Modbus poll.

Then serial traffic was tested for the four main modbus function codes:

1. Read coil status.
2. Read discrete inputs.
3. Read holding register.
4. Read input register.

3.1.1 Description of the software:

They have some features in common such as:

- Supporting Modbus/TCP, RTU and ASCII modes.
- Monitoring of serial traffic.

➢ We constraint our work in RTU mode because of it is higher throughput as mentioned in chapter two.

3.1.2 Basic Simulation Configuration:

Two computers were connected together by means of a serial interface (RS 232), one of these computers holds the modbus poll software and the other holds the modbus slave software. Then the various parameters of the software were adjusted to perform the required communication.

3.1.3 Simulation Parameters Adjustment:

3.1.3.1 Communication parameters:

The port, baud-rate, byte frame and mode of transmission of Modbus poll and modbus slave software's were set to the values shown in the figures below. All of the previous parameters except for the port must be set to the same values in both the Modbus Poll and Modbus Slave.
3.1.3.2 Master/slave definitions:

These include the slave address (i.e. slave ID) to which the master will connect, function (modbus protocol functions (01…04)), starting address, length (number of bytes) and scan rate (refresh rate). It should be noted that the scan rate parameter is significant only with the Modbus Poll.
3.2 Reading of measurands from multifunctional power meter device:

Another objective is to read measurands from multifunctional power meter which acts as a modbus slave device.

To achieve this we used:

1. SINEAX A 210 Multifunctional Power Monitor.
3.2.1 SINEAX A 210 Multifunctional Power Monitor:

The A 210 power measuring instrument is suitable for control panel mounting and measures all the important measurands in 3-phase and single-phase systems. It displays the measurands with a high contrast 14 mm high LED display. The instrument is also suitable for measurements in high and middle voltage systems because of the freely programmable factors for the current and voltage transformers. It replaces a large number of analog instruments and delivers high-accuracy values.

➢ The device figure is placed in appendix A.

3.2.2 Features of A 210 Multifunctional Power Monitor:

- Measurement of current, voltage, active, reactive and apparent power, active and reactive energy, neutral conductor current, power factor and frequency.
- Flexible power supply with an AC/DC wide-range power supply unit.
- Electrically isolated current inputs (1 A or 5 A).
- Upgrade extension modules with RS232/RS485 interface, load profile memory, Modbus, synchronizing input, analog outputs, Ethernet or Profibus-DP.
- Accurate measured values for U, I ≤ 0.5%, F ≤ 0.02 Hz, others 1%.
- Storage of minimum and maximum values.
- Measurement in single-phase systems, 3-wire and 4-wire systems in 4 quadrant operation.

3.2.3 Extension module (EMMOD 201 module):

The EMMOD 201 extension module extends both the functionality and flexibility of the basic A200 series instruments, and supports programming and communication via the RS 232/485 interfaces. It supports data communication with a control system with the Modbus RTU. It can be added without modifying the basic instrument.

The values that can be recorded with A 210 energy meter are average active and reactive power, and the amount of data recorded at 15 minutes intervals varies from a single value per reading in 166 days up to 14 values per reading in 12 days.
The devices of the A200 series of multi-functional power meters may be equipped with the optional extension module EMMOD201. Hence, three different operating modes may be realized:

- Standalone: The module is used for configuring the device only. There is no permanent connection to a master PC and it is not fixed all the time.
- Bus service: Up to 32 devices may be interconnected via RS485 interface. Measured data are requested permanently via a Modbus master.
- Full device control via bus: Beside the interrogation of measurands, the synchronization of the mean power values, tariff switching and digital output driving (on-site alarming) may be performed via bus interface. The Modbus master is fully responsible for a permanent control of the devices and acquires measurands periodically.

➢ The device figure is placed in appendix A.
➢ The following functions can be performed using the extension module:
  - Interrogation of device features.
  - Modification of device features.
  - Acquisition of present measurands.
  - Acquisition of integrated mean-power values.
  - Interrogation / setting / resetting of meter contents.
  - Resetting minimum / maximum values.
  - Acquisition of mean-power values stored in the logger.

The following points are to be considered:

- For a single line system the measured values are stored in specific registers.
- All measurands need two registers because they are stored as float numbers (32-bit).

The set of registers described bellow are obtained from the device manual:

1. Reg 40100 & 40101 for voltage.
2. Reg 40114 & 40115 for current.
3. Reg 40122 & 40123 for average current.
4. Reg 40138 & 40139 for power.
5. Reg 40146 & 40147 for reactive power.
6. Reg 40154 & 40155 for apparent power.
7. Reg 40156 & 40157 for frequency.
8. Reg 40164 & 40165 for power factor.
3.2.4 Setting of SINEAX A 210 Multifunctional Power Monitor:

Although energy meter setting can be configured manually but by using a PC, a software package A200plus and an interface adapter cable RS232 the configuration settings can be clearly and easily made. This benefit results from EMMOD201.

Besides parameter setting, the user-friendly A200plus software supports reading of the measured values.

A210 device communication parameter has been setting to next values (RTU mode):

(1 unit identifier -9600 Baud rate-Even parity-2 stop bit)
3.2.5 Modbus Master VB software:

In order to acquire the measured values from A210 device, we developed a program written in Visual Basic (VB) language with help of modbus ActiveX control (Mbaxp.ocx).

- Main features of the program include:
  - Acts as Modbus master.
  - Communication interface (RS232).
  - Supports RTU mode.
  - Supports modbus function code 03 (Read holding register).

- Description of the functions used in the software listed by their prototypes:

1. **Open Connection ()**:
   
   Its used to open connection with required com port (1,2,3…etc), some parameters must be set before this function called (port num, baud rate, data bits, parity, stop bits, protocol mode, time out ).

2. **Read Holding Registers ( Handle As Integer, Slave ID As Integer, Start Adder As Long, Quantity, Scan Rate As long)**:
   
   It’s used to send request to slave for reading holding registers specified in the function.

3. **Close Connection ()**:
   
   It’s used to terminate the connection.

4. **Float(Handle As Integer, Index As Integer) As Single**:
   
   It’s used to retrieve the value of the specific index (two registers) as Float num.

- The VB software full code is contained in the Appendix.
3.2.6 VB program flowchart:

Figure 3.6 VB program flow chart
3.2.7 Application design:

A simple circuit (fluorescent lamp as load) is constructed with an A210 device. Then, it is connected with the computer that contained the VB program via an RS232 cable.

![Application circuit diagram]

Figure 3.7 application circuit

As it was mentioned before, A210 device support communication via RS232. It has fixed settings for this mode of communication:

- 9600 Baud rate.
- Even parity.
- 2 stop bit.
- 255 unit identifier

So, these settings were applied to our VB program in order to communicate with the device. Then, the connection was opened and finally the eight holding registers - which were mentioned above – were read.
4.1 Results of Monitoring bytes traffic:

We monitored bytes traffic for each one of the following cases:

1. Read coils status.
2. Read discrete input
3. Read holding registers.
4. Read input registers.

- Arbitrary values were assigned to the slave.

**Read coils status (function code 01) RTU mode:**

Read coil status (1-5) from slave device (1). These five states were set arbitrary to:

- 00001 = 1 (on)
- 00002 = 0 (off)
- 00003 = 1 (on)
- 00004 = 1 (on)
- 00005 = 0 (off)

![Figure 4.1: coil status of modbus slave](image)
Data received by the master:

![Modbus Poll](image)

**Figure 4.** Coil status and byte traffic received by modbus Master

**Read input status(function code 02) RTU mode:**

read discrete input (1-5) from slave device (1).

- Slave discrete inputs:
  - 10001 = 1 (on)  
  - 10002 = 1 (on)  
  - 10003 = 1 (on)  
  - 10004 = 0 (off)  
  - 10005 = 1 (on)

![Modbus Slave](image)

**Figure 4.** Discrete input of modbus slave
Chapter 4

Tests and Results

- Master side:

![Modbus Poll - Mbpoll1](image1)

Figure 4.4 discrete input status received by modbus Master

Read holding registers(function code 03) RTU mode:

- Slave hold registers:

  40001 = 33
  40002 = 152
  40003 = 417
  40004 = 105
  40005 = 789

![Modbus Slave - Mbslave1](image2)

Figure 4.5: holding registers of modbus slave
Master side:

![Figure 4.6: holding registers received by modbus Master](image)

Read input registers (function code 04) RTU mode:

- Slave input register:
  - 30001 = 1285
  - 30002 = 12556
  - 30003 = 8954
  - 30004 = 2456
  - 30005 = 365

![Figure 4.7: input registers of modbus slave](image)
- Master side:

![Modbus Poll - [Mbpoll1]](image)

- Figure 4.8: input registers received by modbus Master
4.2 Results of A200 plus software:

![Image of A200 plus software interface](image)

Figure 4.9 A200 plus software interface

4.3 Results of VB master software:

- VB program was designed to read the holding registers of the A210 device.

- The VB program was tested to read the electrical quantities of single line circuit as mentioned in chapter three, so we sent a request from the software to read the holding registers in the range of (40100…40166) because it contains the basic measured values of a single line AC system.

- The obtained results are shown in VB program interface below.
The VB program was designed with an adjustable scan rate so it can detect simultaneously any changes in the measured values. Also the parameters like Slave definition (slaveID, Reg starting Address, Register Size), so this program can communicate with any Modbus salve and reading its holding registers.

- The VB software full code is contained in Appendix.
5.1 Conclusions:

The main objective of this project was to design and implement pc-based DAQ system. All the major objectives have been accomplished beside some included essential features of any design such as the reliability, speed and accuracy.

5.1.1 Benefits:

➢ The use of stand-alone logger/controllers increases system reliability. This is because the stand-alone or distributed system can continue to operate even when the host computer is not connected or functional. It also increases overall system performance, by distributing the control decisions, algorithms and other analysis functions to localized processors.

➢ stand-alone devices do not necessarily have inbuilt filters, since the units can be located very close to the signal source and therefore not as susceptible to noise.

5.1.2 Limitations:

➢ When stand alone logger/controller connected to the host PC, irrespective of whether there is a single logger/controller or a network of devices, all communications must proceed via the RS-232 interface. The speed at which data can be transmitted to and from the logger/controller(s) is limited by the speed at which data can be transferred across the single RS-232 communications path. This is an important consideration where a number of units send information to be logged by the host PC or there is a large amount of data to be uploaded. In either case, the number of samples that can be taken from each logger/controller is limited by the total amount of information that can be sent via the communications interface.

➢ Unlike specialized high-speed plug-in DAQ boards, stand-alone logger/controllers are not designed for the sampling of high frequency signals. Logger/controllers do not enjoy the benefits of high-speed microprocessors, high-speed data bus and fast memory storage facilities such as DMA that characterize modern PCs.
5.2 Recommendations and future work:

- The data acquisition and control system software was developed in Visual Basic version 6. Although Visual Basic offers a wide range of program support, there are some functions that are not available. Hardware accesses to ports or to memory locations are two examples of functions that Visual Basic does not support. In addition, the execution time of Visual Basic programs is relatively slow for some specific real-time systems. Therefore, it was necessary to use a specialized program to be associated with the hardware and time-saving of data acquisition for this system. A dynamically linked library (DLL) can be used for this purpose written by C++ language in the Microsoft Visual C++ environments.

- The number of devices that can be connected to the PC can be increased by using RS-485 serial interface and RS-232 to RS-485 converter, so a network of devices can be established.

- PLC can be connected to this network by adding EM241 extension module, which is not available in laboratory, but can be applied from external vendors.
References


- Development of a PC-Based Data Acquisition and Control Systems; Y. Lan, X. Lin2, M.F. Kocher and W.C. Hoffman, University of Nebraska-Lincoln.

- [http://www.modbus-ida.org](http://www.modbus-ida.org)
  - Modbus Application Protocol Specification
  - Modbus over serial line specification and implementation guide

- Modicon Modbus Protocol Reference Guide

- SINEAX A 210 Multifunctional Power Monitor user manual

- Interface EMMOD 201 V2.0 data sheets

- [http://www.camillebauer.com](http://www.camillebauer.com)
Figure A.1: SINEAX A 210 Multifunctional Power Monitor:

Figure A.2: Extension module (EMMOD 201 module):
Visual Basic program code:

Dim e As Integer
**************
Private Sub ALLButton1_Click() /*close connection button*/
Mbaxp1.CloseConnection
End Sub
**************
Private Sub ALLButton2_Click() /*Exit button*/
End
End Sub
**************
Private Sub ALLButton3_Click() /*open connection button*/
    Mbaxp1.Connection = 1 'Port 1
    Mbaxp1.BaudRate = 5 '9600 Baud
    Mbaxp1.DataBits = 1 '8 Data bits
    Mbaxp1.Parity = 2 'Even parity
    Mbaxp1.StopBits = 1 '2 Stop bits
    Mbaxp1.ProtocolMode = 0 'RTU Mode
    Mbaxp1.Timeout = 1000
    Mbaxp1.OpenConnection
    If Mbaxp1.GetLastError <> 0 Then
        MsgBox "connection error"
    End If
    e = Mbaxp1.ReadHoldingRegisters(1, 1, 99, 66, 1000)
    Mbaxp1.UpdateEnable (1) 'Start continuously update
End Sub
**************
Private Sub Mbaxp1_ResultOk(ByVal Handle As Integer)
    If Handle = 1 Then
        Text1 = Mbaxp1.Float(1, 0)
        Text2 = Mbaxp1.Float(1, 7)
        Text3 = Mbaxp1.Float(1, 19)
        Text4 = Mbaxp1.Float(1, 32)
        Text5 = Mbaxp1.Float(1, 28)
        Text6 = Mbaxp1.Float(1, 23)
        Text7 = Mbaxp1.Float(1, 27)
        Text8 = Mbaxp1.Float(1, 11)
    End If
End Sub