



NEAR EAST UNIVERSITY

**GRADUATE SCHOOL OF APPLIED
AND SOCIAL SCIENCES**

**DESIGN OF A MICROCONTROLLER BASED DATA
LOGGING SYSTEM**

Mohammad Fadel Klaib

MASTER THESIS

DEPARTMENT OF COMPUTER ENGINEERING

Nicosia 2004

NEU

JURY REPORT

**DEPARTMENT OF
COMPUTER ENGINEERING**

Academic Year: 2003-2004

STUDENT INFORMATION

Full Name	Mohammad Fadel Klaib		
Undergraduate degree	BSc.	Date Received	Spring 1997-2001
University	Al_Quds University	CGPA	3.07

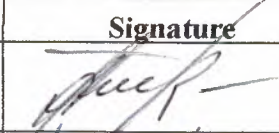
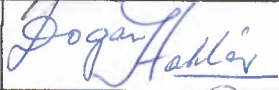

THESIS

Title			
Title	Design Of A Microcontroller Based Data Logging System		
Description: This thesis investigates how real-time environmental data collected by a microcontroller can be sent to the PHP script language program running on a PC and also how this data can be analyzed and stored on MySQL Database storage.			
Supervisor	Prof.Dr. Doğan İbrahim	Department	Computer Engineering

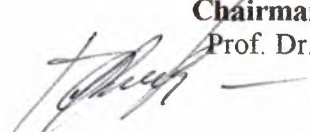
DECISION OF EXAMINING COMMITTEE

The jury has decided to accept / reject the student's thesis. The decision was taken unanimously / by majority .	
-------------------------------------------------------------------------------------------------------------------------------------------	--

COMMITTEE MEMBERS

Number Attending	3	Date	05/03/2004
Name		Signature	
Assoc. Prof. Dr. Rahib Abiyev, Chairman of the jury			
Assist. Prof. Dr. Doğan Haktanır, Member			
Assoc. Prof. Dr. İlham Huseynov, Member			

APPROVALS

Date 05/03/2004	Chairman of Department Prof. Dr. Doğan Ibrahim 
---------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------

DEPARTMENT OF COMPUTER ENGINEERING
DEPARTMENTAL DECISION

Date:05/03/2004

Subject: Completion of M.Sc. Thesis

Participants: Prof. Dr. Doğan İbrahim, Assist.Prof. Dr. Doğan Haktanir, Assoc.Prof. Dr. İlham Huseynov, Assoc. Prof. Dr. Rahib Abiyev, Alaa Eleyan, Mazen Eleyan.

DECISION

We certify that the student whose number and name are given below, has fulfilled all the requirements for a M .Sc. Degree in Computer Engineering.

CGPA

20021354

Mohammad Fadel Klaib

4.00

Assoc. Prof. Dr. Rahib Abiyev, Committee Chairman, Computer Engineering
Department, NEU

Assist. Prof. Dr. Doğan Haktanir, Committee Member, Electrical and Electronic
Engineering Department, NEU

Assoc. Prof. Dr. İlham Huseynov, Committee Member, Computer Information System
Department, NEU

Prof. Dr. Doğan İbrahim, Supervisor, Chairman of Computer
Engineering Department, NEU

Chairman of Department
Prof. Dr. Doğan İbrahim

Mohammad Klaib : **DESIGN OF A MICROCONTROLLER BASED DATA
LOGGING SYSTEM**

**Approval of the Graduate School of Applied and
Social Sciences**

Prof. Dr. Fakhraddin Mamedov
Director



**We certify this thesis is satisfactory for the award of the
Degree of Master of Science in Computer Engineering**

Examining Committee in charge:

A black ink signature of Assoc. Prof. Dr. Rahib Abiyev.

Assoc. Prof. Dr. Rahib Abiyev, Chairman, Computer Engineering
Department, NEU

A blue ink signature of Assist. Prof. Dr. Doğan Haktanir.

Assist. Prof. Dr. Doğan Haktanir, Member, Electrical and
Electronic Engineering Department, NEU

A blue ink signature of Assoc. Prof. Dr. İlham Huseynov.

Assoc. Prof. Dr. İlham Huseynov, Member, Computer
Information System Department, NEU

A black ink signature of Prof. Dr. Doğan İbrahim.

Prof. Dr. Doğan İbrahim, Supervisor, Chairman of
Computer Engineering Department, NEU

ACKNOWLEDGMENTS

The work in this thesis was done under the supervision of Prof. Dr. Dogan Ibrahim (The Chairman of Computer Engineering Department), to whom I am grateful for his support, his interest in the progress of the project, and for his insightful and critical comments.

I am also wish to thank My best friend Mr. Hussam Saeed Musa, Software Engineer from Gaza, who gave me his ever devotion and all valuable information which I really needed to complete my project.

I am also thankful to Mr. Abdullah Al_Kayed, Electronic Engineer in Jordan. Al_kayed has helped me through many helpful and enjoyable discussions.

Also Thanks to all my friends which they support me in Cyprus.

Further I am thankful to Near East University academic staff and all those persons who helped me or encouraged me for the completion of my project. Thanks!

Finally, my thanks go to whom my love will never end, to my father and my mother, to my brothers and sisters, that helped me a lot and gave their lasting encouragement in my studies, so that I could be successful in my life time.

ABSTRACT

This Thesis describes how real-time environmental data collected by a microcontroller can be sent to the PHP script language program running on a PC and also how this data can be analyzed and stored on the permanent MySQL database storage. The thesis also shows how data can be displayed to the users through the Internet using PHP script.

The system designed and developed by the author is based upon Microchip's PIC 16F877 microcontroller, and digitizes and records an analog voltage from a sensor at programmable sample periods, and is readily adaptable for use with a variety of sensors, and it is inexpensive. Individual data loggers cost less than \$20 per unit to build and can be assembled in less than an hour on a printed circuit board, it is permanent storage in MySQL database, then displaying the values read through internet using PHP script language, at any time and in any area of the world. The importance of gathering data accurately and the ability to analyze this data efficiently is discussed in the Thesis.

The author has designed and developed a microcontroller based data logger system and the hardware and software details of this data logger system are described in detail in the Thesis.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	i
ABSTRACT	ii
TABLE OF CONTENTS	iii
INTRODUCTION	1
1 DATA LOGGERS	4
1.1. Overview	4
1.2. What is a data logger?	4
1.3. What are the advantages of using a data logger to collect data?	5
1.4. What are data loggers used for?	6
1.5. Analyzing the data	12
1.6. Features to look for in a data logger	13
1.7. How to choose a data logger?	14
1.8. The different types of data loggers, and how they operate?	15
1.9. How can networking be used for extensive acquisition requirements?	15
1.10. What is a microcontroller?	16
1.11. Summary	18
2 SOME TYPICAL DATA LOGGERS IN THE MARKET	20
2.1. Overview	20
2.2. Popular data loggers	20
2.2.1. Lighting Logger	20
2.2.2. Four Channel Pressure Data Logger	21
2.2.3. Single Channel Temperature Data Logger SmartButton	22
2.2.4. SmartReader Plus 4, Pressure Measurement Solution	24
2.2.5. Validatable Temperature Logger	25
2.2.6. Precision Temperature Datalogger	27
2.2.7. ADC-16 High resolution data logger	29
2.2.8. PT-104 Platinum Resistance Thermometer Data Logger	30

2.2.9. Multi Channel Data Acquisition	32
2.2.10. ADC-10 & ADC-12 –single channel, low cost data loggers	34
2.3. Comparison between the above systems:	36
2.4. Summary	36
3 DESIGNING A DATA LOGGER	37
3.1. Overview	37
3.2. Description Of The Data Logger Designed By The Author	37
3.3. Hardware of the data logger	39
3.3.1. Power	40
3.3.2. Sensors	41
3.3.2.1. Magnetometer	42
3.3.2.2. Accelerometers	43
3.3.2.3. Light Sensor	44
3.3.2.4. Temperature Sensor	45
3.3.2.5. Pressure Sensor	46
3.3.2.6. Humidity Sensor	48
3.3.3. The PIC16F877 Microcontroller	48
3.3.3.1. Overview of the File Registers	51
3.3.3.2. Overview of the 8-Channel 10-bit ADC	52
3.3.3.3. Overview of the USART Hardware	53
3.3.3.3.1. Baud-Rate Generator, BRG	53
3.3.4 Interface by Serial / RS232 Port	54
3.3.4.1 RS232 Converter Chips	55
3.3.4.2. RS232D (9 pin Connector)	57
3.3.4.3. Serial Port's Registers (PC's)	57
3.4. Summary	57
4 DEVELOPMENT OF THE MICROCONTROLLER C PROGRAM USING PICC COMPILER	58
4.1. Overview	58
4.2. Starting HTLPIC	58

4.3. Using HTLPIC	59
4.4. HTLPIC menus	60
4.4.1. <>> menu	60
4.4.2. File menu	60
4.4.3. Edit menu	62
4.4.4 Options menu	65
4.4.5 Compile menu	67
4.4.6 Make menu	70
4.4.7 Run menu	75
4.4.8 Utility menu	76
4.4.9 Help menu	78
4.5 Creating and Compiling the Data Logger Program	81
4.5.1 Flow Chart Of PIC_PROGRAM.C	81
4.5.2. PIC_PROGRAM Discussion	83
4.5.3. Compiling The Data Logger Program Using PICC Compiler	87
4.5.4. Download PROGRAM.HEX To PIC16F877 Chip	91
4.6. Testing	95
5 PHP SCRIPTING LANGUAGE	96
5.1. Overview	96
5.2. Some of PHP's Strengths	96
5.3. Conventions in PHP	98
5.4. Embedding PHP into HTML Pages	99
5.5. Operators	102
5.5.1. Arithmetic Operators	102
5.5.2. String Operators	103
5.5.3. Assignment Operators	103
5.5.4. Pre- and Post-increment and Decrement	104
5.5.5. Comparison Operators	105
5.5.6. Logical operators	106

5.6. Control structure	106
5.6.1. Making Decisions with Conditionals	107
5.6.2. Iteration: Repeating Actions	109
5.7. Summary	113
6 DESIGNING DATABASE FOR DATA LOGGER USING MySQL	114
6.1. Overview	114
6.2. Some of MYSQL Strengths	115
6.3. Users and Privileges	115
6.4. Designing Database for Data Logger	116
6.4.1. Create database	116
6.4.2. Open database	116
6.4.3. Creating database tables	116
6.4.4. Looking at the Database with SHOW and DESCRIBE	118
6.5. SQL Queries Used In Data Logger Database	119
6.5.1. Inserting Data	119
6.5.2. Modifying Data	119
6.5.3. Deleting Data	120
6.5.4. Retrieving Data from the Database	121
6.5.4.1. Retrieving Data with Specific Criteria	122
6.5.5. Altering Tables After Creation	123
6.5.6. Dropping Tables	124
6.5.7. Dropping a Whole Database	125
6.6. Summary	125
7 USING PHP AND MYSQL TO COLLECT AND ANALYZE THE DATA	126
7.1. Overview	126
7.2. Web database Architectures	126
7.3 Putting content into your database with PHP	127
7.3.1. Setting Up a Connection	128
7.3.2. Choosing a Database to Use	129

7.3.3. Querying the Database	129
7.4. Using PHP To read data from database	130
7.4.1. Querying the Database	130
7.4.2. Retrieving the Query Results	130
7.5. Other PHP Database Interfaces	131
7.6. PHP Script Files	132
7.6.1. READ.PHP Script File	132
7.6.1.1. Flow Chart of READ.PHP Script	133
7.6.1.2. READ.PHP Script File Discussion	135
7.6.2. HOME.PHP Script File	137
7.6.3. ADMIN.PHP Script File	138
7.6.4. DISPLAY.PHP SCRIPT File	140
7.6.4.1. Flow Chart of DISPLAY.PHP Script	141
7.6.4.2. DISPLAY.PHP Script Screens	143
7.7. Summary	145
CONCLUSION	146
REFERENCES	148
APPENDIX A	151
APPENDIX B	171

INTRODUCTION

Efforts to understand and model environmental variables require data sets that track both temporal and spatial changes. Networks of recording instruments or instruments connected to commercially available data loggers can gather these data, but the cost of such networks is typically several thousands to tens of thousands of dollars and may require the development of software to interface between the data logger and a host computer. The high cost of these networks together with the risks of vandalism and theft associated with their deployment in remote locations prohibit their general use, even in studies that could clearly benefit from them. This thesis describes an inexpensive, easy-to-build, microcontroller based data logger system. The low cost of this system coupled with the increasing availability and decreasing cost of a wide variety of compatible sensors should enable more researchers and educators to systematically gather temporal and spatial data without the necessity of a large budget.

The system designed and developed by the author is based upon Microchip's PIC 16F877 microcontroller, and: (1) digitizes and records an analog voltage from a sensor at programmable sample periods, (2) is readily adaptable for use with a variety of sensors, (3) is inexpensive. Individual data loggers cost less than \$20 per unit to build and can be assembled in less than an hour on a printed circuit board, (4) permanent storage in MySQL database, (5) displaying the values read through internet using PHP script language, at any time and in any area of the world.

The data logger system developed by the author is composed of two units, a "logger" and a "reader" which work with a host computer. The logger is build on breadboard circuit with an 8-bit microcontroller chip (PIC 16F877, Microchip Technology, Inc.), and supporting components.

Power is supplied through 5V charger and it can be powered directly from the PC. The logger converts an analog voltage signal from an external sensor into a digital value, and

stores the digital value in the logger's registers. The "reader" is in the same microcontroller, a RS-232 transceiver interface chip (MAX232A, Microchip Technology), and supporting components. The reader enables communication between the logger and a "host" PC-compatible computer using one cable to the RS-232 serial port of the host computer. The host computer runs a Windows-based program written to do that, through commands sent to reading data from serial port COM1, that have the values from the logger (microcontroller chip). Typically, a logger is connected to the sensor and deployed in the field for data collection, and finally linked to the serial port to download and store the data on the host computer.

Operation of the system depends upon coordinated software running on the host computer, and the data logger and all parts of the system hardware and software have been tested by the user.

The aim of this thesis is to build a low cost data logger system, that's read 8 analog signal, and 16 digital signals. Then the logger sends these values to the PC using the standard RS232 communications line. PHP script running on the PC opens the serial communications channel and reads the serial data. This data is then stored in the MySQL database for future analysis.

This thesis includes seven chapters, the first three chapters covering the hardware topics, and the other four chapters covering the software topics, organized in the following structure:

Chapter 1, This chapter explains the benefits of using low-cost data loggers for monitoring the performance of various environmental systems. The importance of gathering accurate data and the ability to analyze this information efficiently are described. Various data logging applications will also be explained. The following topics are discussed in this chapter: What is a data logger?, What are the advantages of using data logger for collecting data?, What are data loggers used for?, Analyzing the data, Features to look for in a data logger, How to choose a data logger?, What are the different types of data logger, and how

do they operate?, How can networking be used for extensive acquisition requirements?, and finally, What is a microcontroller?.

Chapter 2, In this chapter the features of various commercially available data loggers are analyzed and compared with each other.

Chapter 3, In this chapter the details of the data logger system designed and developed by the author is described. Briefly, the following topics are described in this chapter: the hardware of the data logger (power, sensors, the PIC16F877 microcontroller, interface by Serial / RS232 Port), and the software of the data logger.

Chapter 4, This chapter covers the details of the HI-TECH C compiler. This is the C compiler used by the author during the development of the microcontroller software.

Chapter 5, This chapter is about the PHP script language. In this chapter the following topics are discussed: what is PHP, some of PHP's strengths, conventions in PHP, how to embed the PHP code into your HTML pages, operators, and control structure.

Chapter 6, This chapter is about the MySQL database server. In this chapter the following topics are discussed: what is MySQL?, what is SQL?, some of MYSQL strengths, users and privileges, how to setting up the database, basic SQL queries.

Chapter 7, This chapter shows how the PHP and MySQL can be linked together. In this chapter the following topics are discussed: why use PHP and MySQL?, web database architectures, how to putting data into your database with PHP, how to getting data from database by using PHP, other PHP Database Interfaces.

Finally, Appendix A contains the source code of the microcontroller program and the PHP script file listings. Appendix B gives the description of the database tables.

1. DATA LOGGERS

1.1 OVERVIEW

This chapter explains the benefits of using low-cost data loggers for monitoring the performance of various environmental systems. The importance of gathering accurate data and the ability to analyze this information efficiently will be discussed. Various data logging applications will also be explained.

In this chapter the following topics will be discussed:

- What is a data logger?
- What are the advantages of using data logger for collecting data?
- What are data loggers used for?
- Analyzing the data
- Features to look for in a data logger
- How do I choose a data logger?
- What are the different types of data logger, and how do they operate?
- How can networking be used for extensive acquisition requirements?
- What is a microcontroller?

1.2 WHAT IS A DATA LOGGER?

A data logger [1] is an electronic device that records the output of sensors. Data loggers are small, stand-alone, battery-powered devices that are equipped with a microprocessor, memory for data storage and sensor's. The sensors can measure many different physical quantities, ranging from speed, acceleration and position to temperature, relative humidity, rainfall, light intensity, on/off and open/closed state changes, voltage and events over extended periods of time. Data loggers can also control other devices according to the measurements made by the sensors. For example, a data logger in a greenhouse could control ventilation and humidifiers according to the sensor readings.

Typically, Most data loggers interface with a personal computer and utilize software to activate the logger and view/analyze the collected data.

Data collected can be displayed in tables and graphs and the software available allows for sophisticated analysis.

Data logging technology allows the user to collect and store data in a short period of time and to focus on its analysis.

1.3 WHAT ARE THE ADVANTAGES OF USING A DATA LOGGER TO COLLECT DATA?

A data logger is an attractive alternative to either a recorder or data acquisition system in many applications [2].

When compared to a recorder, data loggers have the ability to accept a greater number of input channels, with better resolution and accuracy. Also, data loggers usually have some form of on-board intelligence, which provides the user with diverse capabilities. For example, raw data can be analyzed to give flow rates, differential temperatures, and other interpreted data that otherwise would require manual analysis by the operator.

The major difference between a data logger and a recorder, however, is the way the data itself is stored, analyzed and recorded. A common recorder accepts an input, and compares it to a full-scale value. The pen arm is then deflected across the recording width, to produce The appropriate ratio of the actual input to the full-scale input. For example, using a recorder with a 1 Volt full scale, an input of 0.5 Volts would move the pen $0.5/1$ or 50% of the distance across the recording width. In comparison, a data logger accepts an input, which is fed into an analog-to-digital converter prior to analysis and storage. This method has advantages in accuracy and resolution, while only a recorder can provide a truly continuous trend recording.

One of the primary benefits of using data logging for systems monitoring is the ability to collect data on a 24-hour, around-the-clock basis. Upon activation, the loggers are deployed and left unattended to gather and record information for the duration of the monitoring period. This allows for a more complete and accurate picture of the target system's overall performance. Because some condition changes take place over long periods of time or when no one is present, the use of data loggers is much more efficient and feasible than performing "spot checks" to gather data manually.

Another important benefit is the ability to collect data in hard-to-reach areas through the use of built-in or external sensors. Temperature, relative humidity, CO₂, voltage, amperage, pressure, light, many other types of data can be monitored in areas that would otherwise be too difficult or cumbersome to access such as electrical junction boxes, air return vents, water recovery tanks, motor and fan housings, to name a few. Loggers with external input capabilities may be wired to existing gauges and sensors that have voltage output terminals, thereby allowing these devices to be monitored and recorded, also.

Data loggers incorporate the latest in digital technology making them smaller, less expensive, more accurate and more reliable than chart recorders. Furthermore, data is stored in digital format, which allows for more convenient analysis, presentation and storage. There is no need for costly and inconvenient chart supplies because the data is viewed on a PC or laptop computer. The small, unobtrusive size of some data loggers makes them easy to hide for inconspicuous data collection.

Additionally, many benefits can be realized from choosing data loggers over data acquisition systems for performance monitoring. Data acquisition systems often require the installation of extra circuit boards and controls in the PC along with the connection and wiring of cables and sensors. These systems work well in permanently configured, on-line applications but can be very expensive and difficult to implement, especially when short-term monitoring is all that is required. Data loggers significantly reduce the price per channel for most logging applications, are easier to deploy and can be placed in areas that permanent, digital systems cannot reach. They can also be easily removed and reused in other application studies, because they require no extra wiring and are stand-alone devices.

1.4 WHAT ARE DATA LOGGERS USED FOR?

Data loggers are used in a huge variety of applications requiring rugged, portable data acquisition and monitoring.

Although PC-based measurement systems are now very common, they aren't always appropriate. For example, data loggers have been used in tunnels, on vehicles, on bridges, in fields, on mountain-tops and even in the Space Shuttle! In these situations PC-based data acquisition systems may not offer the right temperature range, or be rugged or compact

enough. Also, data loggers are often used in remote locations where there is no mains electricity and can be battery-powered. In these situations low power consumption is critical.

Data loggers aren't just for making measurements. They can also control external devices, carry out calculations and send messages. That's why some data loggers are called "Measurement and Control Systems".

Data loggers are used in many different applications to monitor and collect specific types of information. The following are just a few examples of "real world" applications where data loggers can make the task of gathering information a lot easier and much more efficient.

One of the common environmental complaints in residential properties is that the heating or air conditioning system is not working properly. It's either always too hot or always too cold. These types of comfort level complaints are very easy to monitor and verify with the use of temperature data loggers.

Temperature data loggers employ a sensor that reacts to changes in temperature. The logger monitors and records these changes at preset intervals and stores the data with date and time information in its memory for later retrieval. Many temperature loggers are small enough that they can be placed in hidden, "out-of-the-way" locations to gather information without being seen or disturbed. Depending on the amount of built-in memory and the interval for taking readings, the loggers can realistically collect data for several months at a time, before reaching their full capacity.

This temperature data, upon later analysis, offers a much more accurate and complete picture of the actual temperature activity that occurred throughout the entire monitoring period. "Spot checks" or manual readings would not provide this type of extensive coverage. The data can then be used to ascertain where problems in the heating system might exist, before making an informed recommendation to the customer or client. The savings in manpower alone are enormous, especially if the study is being done in a multi-unit residential or commercial building.

Temperature loggers can also be used in many other applications. Here are a few examples:

Typical applications for data loggers include:

- Agricultural research.
- Automotive testing.
- Civil engineering.
- Environmental monitoring.
- Process control.
- Structural monitoring.
- Weather & meteorology.
- Verify nighttime temperature setback strategies.
- Monitor air handler temperature fluctuations.
- Analyze gradient temperature changes.
- Monitor equipment-operating temperatures.
- Compare outdoor conditions to indoor comfort levels.
- Assess operating temperatures of motors.

Logging Motor/Compressor Cycling Performance

“State” loggers [3] are another very useful type of data logger and compressors. These loggers record the “on” and “off” run times along with time and date stamps. This information is very useful in detecting short cycling and recording the overall usage of the equipment being monitored.

Motor On/Off loggers with built-in vibration sensors are often used to monitor the start and stop times of compressors and pumps. The adjustable sensor detects the vibration when the compressor is turned on and records this event with a time and date stamp. When the compressor or pump is shut down, the logger will also record this event with another time and date stamp. This data can be used to show the total amount of time the equipment was operational. Further analysis of the data can show where short cycling may have occurred, the peak usage times and percentage of time that the equipment was on or off.

For electrical devices that do not produce measurable vibrations (small fans, motors, etc...), AC-field sensor loggers are used to monitor On/Off state changes. These loggers detect the

electromagnetic field produced when the device is turned on, through the use of a built-in sensor. The loggers can be placed directly on the outside of electric motor housings, and may also be mounted on one of the phases of power being supplied to the motor or device being monitored.

There are “State” loggers that also record “open” and “closed” events as well as contact closures. These loggers can be wired to passive relay-switches and dry contacts. When a contact is made, or when the relay switch is “tripped”, the logger records this data with a time and date stamp for each On/Off or Open/Closed state change.

When evaluating motor, fan or compressor cycling and performance, “State” loggers are an invaluable tool. These loggers can detect and log events that might otherwise go unnoticed, or would be difficult to monitor manually. They offer the capability to capture events, whenever they may occur.

Monitoring Electric Light Usage

When performing baseline studies on power consumption in buildings [3], knowing how much electricity being used for lighting is very important. There are data loggers available that detect and record light usage.

Light On/Off loggers, are similar to “State” loggers in that they record data only when a “state” change occurs. These loggers employ a sensor that detects changes in light intensity. There should be a sensitivity control so that the threshold between light and darkness can be adjusted by the user. This will allow the logger to know the difference between “on” and “off”. Lighting conditions vary from room to room, and having the ability to adjust the sensor is important. Care should also be taken to make sure that the light sensor directly faces the source of light being monitored. Sunlight and other reflected light sources may trigger these loggers giving inaccurate readings. When the lights are turned on, the logger will record this state change along with date and time stamps. When the lights are turned off, this information is also recorded.

This type of data is valuable in determining how often-electric lighting is being used. Peak hours and periods of non-usage can also be analyzed. This information is very useful,

especially when making recommendations for retrofits and estimating the electrical demand for lighting.

Power Usage Logging

Tracking power consumption [3] and usage is essential in evaluating a system's overall performance. The efficiency and energy demands of equipment can be monitored and recorded through the use of data loggers.

Readings of current, amperage and voltage may be taken manually with hand-held meters and probes. These methods of collecting data are common and work well, when short monitoring periods are required. However, when longer studies are necessary, data loggers can gather information for extended, uninterrupted periods of time.

Loggers with the ability to accept external inputs are an excellent choice for collecting power data. These loggers are programmed by the user to take readings at preset intervals. The external inputs on these types of loggers usually accept voltage inputs from external sensors and preexisting gauges. Current transformers, meters and gauges can be used with external input loggers, provided that these sensors have voltage output characteristics compatible with the input voltage requirements of the data logger. Current transformers are simply attached around the phases of power being monitored. (In most cases, no hard wiring is necessary). The logger, upon activation, will take readings at the preset intervals and record the amperage, voltage or current information with time and date stamps. Some of these loggers have multi-channel capabilities and can collect data from various different sensors simultaneously. Recording data from all 3 phases of power, allows for a more complete picture of the total power being used as well as the peak hours of usage.

By relying on manual readings, events such as power spikes, surges and fluctuations may go undetected. Data loggers with external current probes and sensors attached can perform unattended studies for days, weeks and months around the clock. Data can be gathered at times that would not otherwise be practical or efficient.

Air Quality Monitoring

The performance of air quality systems can also be monitored and recorded [3] using data loggers. Temperature readings for supply and return vents can be recorded and compared to help identify air balance problems. CO₂ levels and relative humidity information can also be monitored to obtain important air quality data.

Various data loggers are available that have built in temperature and RH sensors so that these types of data can be collected simultaneously in the same location. Other loggers with external input capabilities can be effectively used by attaching external temperature, RH and CO₂ sensors for placement in hard to reach areas. Hybrids of these two types of loggers are also available. They offer built in sensors as well as the capability to attach external sensors, for even more monitoring possibilities.

External temperature sensors can be placed inside air vents, and attached to heating coils and pipes for accurate readings. Temperature and RH readings can be taken inside and outside of buildings for comparison. CO₂ levels can be monitored when the building is occupied and unoccupied for later evaluation of the ventilation system.

By collecting data from various locations simultaneously, valuable information about overall air system performance can be obtained. Later analysis of the data will identify energy saving opportunities in over-ventilated spaces. Complaints about air quality or insufficient ventilation can be studied objectively. Performance of heating and air conditioning systems can be evaluated. Temperature changes can be analyzed to verify proper heating and cooling operation.

Through the use of data loggers, more accurate and usable data can be obtained with much less expense and over longer periods of time, than can be obtained by manual, hand-held monitor readings.

1.5 ANALYZING THE DATA

Most data loggers operate with some type of proprietary software [2]. This software allows the user to set certain operating parameters within the logger. Sampling/reading intervals, delayed start times, alarm ranges, external sensor selection, a descriptive header for the logger are a few examples. Information about the logger's battery status, clock synchronization and memory capacity may also be available. There should also be a software command to activate the logger. All of this is accessed through a serial port connection between the logger and the computer.

Software features vary among data logger manufacturers. At a minimum, the user should be able to view the data files with date and time information in graphical and tabular formats. The capability to export data files into other spreadsheet programs is also a very useful feature found in some data logging software programs. Other software features to look for are:

- 1) Capability to display multiple data files on one graph.
- 2) Zoom in on data of interest.
- 3) Copy and paste graphs into other applications.
- 4) Compare multiple parameters, from multiple loggers.
- 5) Multiple value axes on one graph. Set axis ranges.
- 6) Calculate on/off run times and percentages.
- 7) View and overlay data from successive deployments.
- 8) Capability to customize graphs, colors, legends, etc.
- 9) Good Technical support.

Software packages with these features allow for easier analysis of the collected data. Specific data points can be filtered out for more accurate comparisons and study. Graphs and tabular data can be printed for use in reports and evaluation of long-term studies. A

major advantage of using data loggers over hand-held monitoring devices and chart recorders are the capability to define, target and store massive amounts of information relevant to the study- something not possible with chart recorders. Good, comprehensive analytical software is as important as the data logger itself, when evaluating systems performance.

1.6 FEATURES TO LOOK FOR IN A DATA LOGGER

Listed below are some of the basic features that one should look for when choosing data loggers for use in performance studies [2].

- 1) **Ease of use...** The logger should be user-friendly, easy to set up, activate and off load data.
- 2) **Size...** The logger should be small and easy to install in hard to reach locations. Easy to carry. Portable.
- 3) **Self-powered...** Data loggers that are self-powered can be placed in areas and locations where electricity is not readily available.
- 4) **Flexibility...** You should be able to define reading intervals, select external sensors, customize and modify data easily. Be able to deploy the loggers where needed.
- 5) **Reliability...** Choose loggers that have a proven track record for accuracy, reliability and dependability.
- 6) **Durability...** Look for loggers that are designed for rugged indoor/outdoor use. They should be able to withstand the rigours of harsh environments (extreme heat and cold, excessive moisture, pressure, etc.)
- 7) **Technical Support...** Even in the best situations, there will always be questions about software. Questions about whether or not the logger is suitable for certain applications. Questions on how to change the batteries. Choose your loggers from a manufacturer that provides good, prompt, informative and useful answers to your

questions. There may be unforeseen problems that crop up, and having good technical support is a must.

- 8) **Comprehensive Software...** The data that you have collected is useless if you can't analyze it. Make sure that the logger software is easy to use. You should be able to access and customize data files easily.
- 9) **Data Shuttle Capability...** Data shuttles are convenient, hand-held devices that allow you to off load and relaunch data loggers, while in the field. There is no need to bring the loggers back to the computer. The shuttle holds the data and is easily off loaded to the computer instead. This allows the loggers to remain in place. Not all data loggers are shuttle compatible, however, this is a very nice feature that makes data collection even easier.
- 10) **Cost...** Collecting data can be expensive, especially if it is not done manually. Data loggers can range in price from less than \$100 to over \$1,000. Although factors like accuracy, flexibility and reliability are very important so is the cost of the logger itself. Memory and computer processor prices have fallen sharply over the past few years making data loggers more accessible than ever before. The technology is readily available, inexpensively, for small companies and large firms alike. When choosing data loggers, their cost is an important factor to consider when planning a building systems performance study.

1.7 HOW TO CHOOSE A DATA LOGGER?

Here's a quick summary of the points you'll need to consider:

1. What physical measurements do you need to make?
2. What type of sensors will you need and what kind of signal (output) does that sensor give? For example, many sensors give an analogue voltage proportional to the quantity you are measuring (e.g. temperature). Others send out streams of electronic pulses, which the data logger needs to count. Still others transmit the readings by generating digital pulses, which the data logger interprets.
3. How frequently do you need to make measurements?

4. Do you need to control any external devices?
5. Do you need to process the data to reduce the number of measurements stored?
6. Where will the data logger be located? Does it need environmental protection?
What about power?
7. How will you communicate with the data logger?

1.8 THE DIFFERENT TYPES OF DATA LOGGERS, AND HOW THEY OPERATE?

The differences between various data loggers are based on the way that data is recorded and stored. The basic difference between the two data logger types is that one type allows the data to be stored in a memory, to be retrieved at a later time, while the other type automatically records the data on paper, for immediate viewing and analysis. Many data loggers combine these two functions, usually unequally, with the emphasis on either the ability to transfer the data or to provide a printout of it.

The advantages of the local hard copy data loggers are that 1, the operator has a permanent recording on paper, 2, no other external or peripheral equipment is required for operation, and 3, many data loggers of this type also have the ability to record data trends, in addition to simple digital data recording.

In comparison, units with internal data storage tend to be more compact, due to the fact that no paper and recording equipment are required, and because they are much simpler electronically and mechanically.

Data storage units are usually more economical. These units can also be operated in a stand-alone mode, with the ability to feed or download data to a host computer system.

1.9 HOW CAN NETWORKING BE USED FOR EXTENSIVE ACQUISITION REQUIREMENTS?

For users who must acquire data over many locations, and wish to have a single collection/recording point, networking is a truly viable solution. With a network, one central location is responsible for data storage and recording; data is collected by remote units in various locations, and then fed to this "master" unit for storage/recording. This is a

great convenience, in that an operator can retrieve the data from one location, rather than having to go to each individual site for collection.

1.10 WHAT IS A MICROCONTROLLER?

A microcontroller [5] is a computer on a chip. All computers -- whether we are talking about a personal desktop computer or a large mainframe computer or a microcontroller -- have several things in common:

- All computers have a CPU (central processing unit) that executes programs. If you are sitting at a desktop computer right now reading this article, the CPU in that machine is executing a program that implements the Web browser.
- The CPU loads the program from somewhere. On your desktop machine, the browser program is loaded from the hard disk.
- The computer has some RAM (random-access memory) where it can store "variables."
- And the computer has some input and output devices so it can talk to people. On your desktop machine, the keyboard and mouse are input devices and the monitor and printer are output devices. A hard disk is an I/O device -- it handles both input and output.

The desktop computer you are using is a "general purpose computer" that can run any of thousands of programs. Microcontrollers are "special purpose computers." Microcontrollers do one thing well. There are a number of other common characteristics that define microcontrollers. If a computer matches a majority of these characteristics, then you can call it a "microcontroller":

- Microcontrollers are "embedded" inside some other device (often a consumer product) so that they can control the features or actions of the product. Another name for a microcontroller, therefore, is "embedded controller."

- Microcontrollers are dedicated to one task and run one specific program. The program is stored in ROM (read-only memory) and generally does not change.
- Microcontrollers are often low-power devices. A desktop computer is almost always plugged into a wall socket and might consume 50 watts of electricity. A battery-operated microcontroller might consume 50 milliwatts.
- A microcontroller has a dedicated input device and often (but not always) has a small LED or LCD display for output. A microcontroller also takes input from the device it is controlling and controls the device by sending signals to different components in the device.

For example, the microcontroller inside a TV takes inputs from the remote control unit and displays outputs on the TV screen. The controller controls the channel selector, the speaker system and certain adjustments on the picture tube electronics such as tint and brightness. The engine controller in a car takes inputs from sensors such as the oxygen and knock sensors and controls things like fuel mix and spark plug timing. A microwave oven controller takes input from a keypad, displays outputs on an LCD display and controls a relay that turns the microwave generator on and off.

- A microcontroller is often small and low cost. The components are chosen to minimize size and to be as inexpensive as possible.
- A microcontroller is often, but not always, ruggedized in some way.

The microcontroller controlling a car's engine, for example, has to work in temperature extremes that a normal computer generally cannot handle. A car's microcontroller in Alaska has to work fine in -30 degree F (-34 C) weather, while the same microcontroller in Nevada might be operating at 120 degrees F (49 C). When you add the heat naturally generated by the engine, the temperature can go as high as 150 or 180 degrees F (65-80 C) in the engine compartment.

On the other hand, a microcontroller embedded inside a VCR hasn't been ruggedized at all.

The actual processor used to implement a microcontroller can vary widely. For example, the cell phone shown on Inside a Digital Cell Phone contains a Z-80 processor. [4] The Z-80 is an 8-bit microprocessor developed in the 1970s and originally used in home computers of the time. The Garmin GPS contains a low-power version of the Intel 80386. The 80386 was originally used in desktop computers.

In many products, such as microwave ovens, the demand on the CPU is fairly low and price is an important consideration. In these cases, manufacturers turn to dedicated microcontroller chips -- chips that were originally designed to be low-cost, small, low-power, embedded CPUs. The Motorola 6811 and Intel 8051 are both good examples of such chips. There is also a line of popular controllers called "PIC microcontrollers" created by a company called Microchip. By today's standards, these CPUs are incredibly minimalistic; but they are extremely inexpensive when purchased in large quantities and can often meet the needs of a device's designer with just one chip.

Any microcontroller chip might have X bytes of ROM and Y bytes of RAM on the chip, along with Z I/O pins. In large quantities, the cost of these chips can sometimes be just pennies. You certainly are never going to run Microsoft Word on such a chip -- Microsoft Word requires perhaps 30 megabytes of RAM and a processor that can run millions of instructions per second. But then, you don't need Microsoft Word to control a microwave oven, either. With a microcontroller, you have one specific task you are trying to accomplish, and low-cost, low-power performance is what is important.

1.11 SUMMARY

Data loggers are effective, inexpensive tools ideal for monitoring and measuring building systems performance. They can be used to collect important data from many types of devices and equipment, whether you are there or not. They take the guesswork out of trying to evaluate the efficiency of a particular system, by providing accurate information. The benefits obtained from being able to print and easily store graphic and tabular data for analysis and evaluation are numerous. Data loggers are inexpensive and they offer huge savings in manpower alone. They log data non-stop, around the clock, day and night,

gathering information that might otherwise go undetected. Loggers can be placed in hard-to-reach, out-of-the-way areas. The inexpensive cost of computer memory has put data loggers within the reach of everyone. By using data loggers to collect the information, your valuable time can be spent elsewhere, analyzing and evaluating the overall performance of the system.

2. SOME TYPICAL DATA LOGGERS IN THE MARKET

2.1 OVERVIEW

The differences between various data loggers are based on many factors. One of these factors is the way that data is recorded and stored. In this factor the basic difference between data logger systems is that some type allows the data to be stored in a memory, to be retrieved at a later time, while the other type automatically records the data on paper, for immediate viewing and analysis. And the other differences are the type of the chip or microcontroller used to collect data and the number of sensors that can read the various data. So in this chapter we will look at the details and the features of several popular data loggers that exist in the market place.

2.2 POPULAR DATA LOGGERS

2.2.1 Lighting Logger

Lighting Logger [6] is a battery powered, portable logger for lighting energy audits. It has a built-in photocell to monitor lighting and a sensitivity adjustment to eliminate day lighting and ambient lighting. On/Off transition times are recorded and stored

Uses and application:

- Evaluate retrofits and verify savings.
- Answer peak impact questions.
- Calculate energy consumption.

Features:

- No Wiring.
- Battery operated, 10-year life.
- Magnetic strip for easy mounting in fixtures.

Specifications at a Glance:

- Size: 1" x 2" x 4".
- Weight: 4 ounces.

- Mounting: Lighting Logger: - Magnetic Strip with 32 oz.
- Display: 2-digit LCD display ("ON" when recording, blank when "OFF").
- Units: Time stamps record in month, day, year, hour, minute, and second internally in the logger.
- Power: Internal Lithium Battery, factory installed with 10-year lifetime.
- Operating: Environment 0°C to 60°C, non-condensing humidity.
- Memory: Capacity 8,192 records, ring memory.
- Communication: RS-232 using the Smart Logger™ Interface Cable.
- Resolution: One second.

2.2.2 Four Channel Pressure Data Logger

SmartReader Plus 4 data logger [7] provides easy monitoring and troubleshooting of pneumatic control systems. The unit is completely self-contained with a 1/4" quick-release pressure fitting to allow a fast and trouble-free connection. As an added bonus: SmartReader Plus 4 can also monitor temperature internally, or temperature and RH externally. (RH sensors/probes and thermistor temperature probes sold separately).

Features:

- 12-bit resolution.
- 10-year battery life.
- Accessible remotely by modem.
- Network capable.
- Continuous sampling.
- User selectable rates.

Specifications at a Glance:

- 4 Channels.
- Memory Sizes: 32 KB and 128 KB.
- Size: 107 x 74 x 22 mm (4.2" x 2.9" x 0.9").
- Weight: 110 grams (3.75 ounces).
- Case Material: Noryl Plastic.

- Operating Limits: -40°C to 70°C (-40°F to 158°F) and 0 to 95% Relative Humidity (non-condensing).
- Clock Accuracy: +/-2 seconds per day.
- Battery: 3.6 volt Lithium, 1 Amp-Hour.
- Power Consumption: 5 to 10 micro amps (continuous).

2.2.3 Single Channel Temperature Data Logger SmartButton

Features:

- Single-Channel temperature.
- One-year warranty.
- Variable sample rate (1 minute to 255 minutes).
- Record reading up to one year.
- “Plug and play” ease of use.
- Zooming graph capabilities.

The ACR SmartButton [7] comes with a one-year warranty, has a 10-year battery life, is NIST traceable, has the ability to record over 2000 data points, and operates in a temperature range of -10 to 85°C. The ACR SmartButton Reader™ software is packed full of features including user selectable sampling rates, programmable alarm thresholds, graph zooming and start delay.

Product Specifications:

- Size: 17 mm diameter x 6 mm height.
- Weight: 4 g.
- Battery: 3.0 volt Lithium (typical life - 10 years).
- Resolution: 8 bits – 0.5°C (0.9°F).
- Mounting: User selectable.
- Clock Accuracy: +/-2 seconds per month.
- Sampling Methods: Continuous (First-in, First-out), Stop when full.

- Case Material: Stainless steel.
- Memory Size: Capable of storing 2048 readings.
- Sampling Rates: User selectable rates from 1 to 255 minutes.
- Start Delay: Up to 255 minutes.
- Number of Channels: One internal temperature sensor.
- Alarm Thresholds: User selectable, event storage.
- Warranty: 1 year.

PC Requirements:

IBM PC or 100% compatible Pentium 75 (minimum) with at least one free serial port, 16 MB of RAM memory, colors monitor & pointing device.

- Software Requirements: ACR SmartButton Reader™ for Windows® versions 95, 98, 2000, NT or ME.
- Communication: RS232 Serial/ACR SmartButton® interface.

Sensor Specifications:

- Sensor: -10 to 85°C (14 to 185°F).
- Sensor Accuracy: +/-0.5°C from 0 to 70°C, +/-0.9°F from 32 to 158°F).

Applications:

Food processing verification, pharmaceutical storage, Laboratories, transportation of temperature sensitive Goods, equipment runs time, HVAC system testing and balancing, etc.

Accessories

Interface cable, plastic plate mount, angled blue hard plastic and adhesive.

2.2.4 SmartReader Plus 4, Pressure Measurement Solution



FIG 2.1 Four Channel Pressure, Temperature and RH Data Logger

Features:

- 12-bit resolution.
- 4 channels.
- 10 years battery life.
- Up to 128 KB memory.
- User-configurable.
- Accessible remotely by modem
- Network capable.
- Up to 25 samples per second.

The SmartReader Plus 4 data logger [7] provides easy monitoring and troubleshooting of pneumatic control systems. The unit is completely self-contained with a 1/4" quick-release pressure fitting to allow a fast and trouble-free connection. As an added bonus, SmartReader Plus 4 can also monitor temperature internally, or temperature and RH externally. (RH sensors/probes and thermostat temperature probes sold separately).

Product Specifications:

- Number of Channels: Four. (One for ambient temperature, one for pressure, and 2 for the optional Relative Humidity/Temperature robe).
- Calibrated Accuracy: $\pm 0.5\%$ of full scale @ 25°C (77°F).
- Memory Sizes: 32 KB and 128 KB.

2.2.5 Validatable Temperature Logger



FIG 2.2 Validatable Temperature Logger[8]

- Portable self-contained temperature data logger for critical pharmaceutical validation applications.
- Equipped with internal temperature sensor, memory and ten-year battery.
- Tamper-proof operation produces reliable and secure documentation for validation records.
- Password-protected setup, download and calibration functions ensure integrity of collected data.
- Records temperature without wires, power cords or connections.
- Traceable instrument-grade temperature measurement accuracy.
- Adjustable logging intervals with multi-year data recording capacity.
- High-stability temperature sensor offers exceptional long-term data logger accuracy and in-calibration performance.
- External input for remote temperature probe flexibility

Technical Specifications

- Size: 2.8 x 2.1 x 0.7" (71x53x18mm), 60g (2.2 oz).
- Operating Range: -40 to 85°C (-40 to 185°F) and 0 to 100% RH.
- Interfaces: RS-232 serial port, half-duplex, 19,200 baud. USB interface available through USB Logger Cable. Ethernet network interface available through Ethernet-to-serial adapter. Mounting: Magnetic strips, Velcro optional.
- PC Software: Designed for use with log Software.
- Clock: Accuracy: +/- 1 min./month at 0 to 50°C.

- Electromagnetic Interference: Meets FCC Part 15 for digital devices; meets CE requirements for radiated emissions, electrostatic discharge, & radiated susceptibility.
- Power Source: Internal lithium battery with life of 10 years at 1 min. sampling rate.
- Temperature Accuracy: See table 2.1.

Table 2.1 : Temperature Accuracy In Validatable Temperature Logger

Sensor	Measuring Range	Precision	One year Accuracy	Sensor Stability
Internal	-40 to 80°C	0.05°C at 25°C	0.15°C ¹ over 20 to 30°C	< 0.1°C over 10 years at 75°C
Remote	-50 to 150°C	0.05°C at 25°C	25°C +/- 0.25°C at 25°C	< 0.1°C over 10 years at 75°C

Temperature Precision Graph

FIG 2.3 shows the details VL-1000 temperature precision over rated measuring ranges (as specified in above table). Solid graph line represents values for internal temperature sensor. Dotted line represents values for remote temperature sensor.

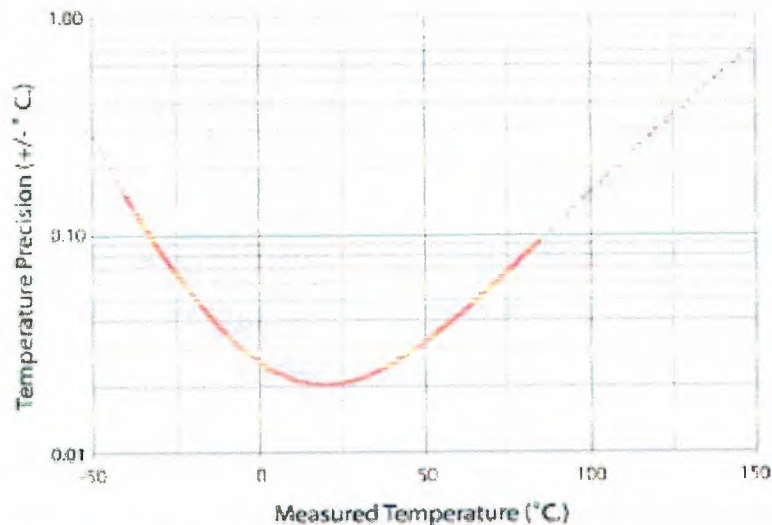


FIG 2.3 Temperature Precision Graph

Internal Temperature Sensor Type of Sensor:

- Precision-tolerance epoxy-encapsulated NTC thermostat.
- Response Time: 10 seconds in moving air (thermostat only).

Remote Temperature Probe Input

- Compatibility: Accepts one EPT-010 temperature probe or any 100K ohm thermostat probe compatible with Betatherm 100K6A1. Contact factory for resistance-temperature curve information.
- Input Connection: Pluggable screw-type terminal block accepts bare leads or EPT-010 - compatible temperature probe connector.

Memory:

- Memory Type: Non-volatile 32K x 8 EEROM.
- Data Sample Capacity: 21,500 12-bit samples.
- Memory Modes: User-selectable: Wrap when memory full, or Stop when memory full.
- Memory Protection: Data retention > 20 years backup without power.
- Sampling Rates: User-selectable (in intervals of 10 seconds) from once every 10 seconds to once a day.
- Recording Span: Recording span depends on sample interval selected and number of channels enabled. Table 2.2 below details typical sampling rates and length of time logger will retain data in memory before wrapping around or stopping.

Table 2.2: Recording Span In Validatable Temperature Logger

Sample interval	Recording Span	
	1 channel	2 channel
10 seconds	60 hours	30 hours
1 minute	14.9 days	7.4 days
15 minutes	7.5 months	3.7 months
1 hour	2.4 years	1.2 years
3 hours	7.3 years	3.6 years

2.2.6 Precision Temperature Datalogger

Specifications for the Veriteq Spectrum 1000 Precision Temperature Datalogger[9]

General

- Size: 2.8 x 2.1 x 0.7" (71x53x18mm); 60g (2.2 oz).
- Operating Range: -40 to 85°C (-40 to 185°F) and 0 to 100% RH.

- Interfaces: RS-232 serial port, half-duplex, 19,200 baud. USB interface available through USB-to-serial adapter.
- Mounting: Magnetic strips, Velcro optional.
- PC Software:
 - Compatible with the following software packages: Veriteq Spectrum, Veriteqs Go (for Palm OS), Veriteq viewLinc (for on-going monitoring & alarming), Spectrum Excel Add-in, and Spectrum API (for custom applications).
- Clock: Accuracy: +/- 1 min./month at 0 to 50°C.
- Electromagnetic Interference: Meets FCC Part 15 for digital devices; meets CE requirements for radiated emissions, electrostatic discharge, & radiated susceptibility.
- Power Source: Internal lithium battery with life of 10 years at 1 minute sampling rate.

Temperature Accuracy

Table 2.3: Temperature Accuracy In Precision Temperature Datalogger

Sensor	Measuring Range	Precision ³	One year Accuracy	Sensor Stability
Internal	-40 to 80°C	0.05°C at 25°C	0.15°C ¹ over 20 to 30°C	< 0.1°C over 10 years at 75°C
Remote ²	-50 to 150°C	0.05°C at 25°C	25°C +/- 0.25°C at 25°C	< 0.1°C over 10 years at 75°C

¹ Over -20 to 70°C: +/-0.25°C With optional EPT-010 remote temperature probe

Internal Temperature Sensor

- Type of Sensor: Precision-tolerance epoxy-encapsulated NTC thermostat.
- Response Time: 10 seconds in moving air (thermostat only).

Remote Temperature Probe Input

- Compatibility: Accepts one EPT-010 temperature probe or any 100K ohm thermostat probe compatible with Betatherm 100K6A1. Contact factory for resistance-temperature curve information.

- Input Connection: Pluggable screw-type terminal block accepts bare leads or EPT-010 - compatible temperature probe connector.

Memory

- Memory Type: Non-volatile 32K x 8 EEROM.
- Data Sample Capacity: 21,500 12-bit samples.
- Memory Modes: User-selectable: Wrap when memory full, or Stop when memory full.
- Memory Protection: Data retention > 20 years backup without power.
- Sampling Rates: User-selectable (in intervals of 10 seconds) from once every 10 seconds to once a day.
- Recording Span: Recording span depends on sample interval selected and number of channels enabled. Table 2.4 details typical sampling rates and length of time logger will retain data in memory before wrapping around or stopping.

Table 2.4: Recording Span In Precision Temperature Datalogger

Sample Interval	Recording Span	
	1 channel	2 channels
10 seconds	60 hours	30 hours
1 minute	14.9 days	7.4 days
15 minutes	7.5 months	3.7 months
1 hour	2.4 years	1.2 years
3 hours	7.3 years	3.6 years

2.2.7 ADC-16 High resolution data logger

Features

- High resolution.
- No power supply required.
- Compact design.

Data logging [10] software included The ADC-16 data logger offers high resolution (16 bits + sign) and is capable of detecting signal changes as small as 40 μ V. It provides a PC

with 8 highly accurate input channels. Pairs of channels can be used differentially to reject noise. Reference outputs can be used to directly power sensors.

An optional terminal board provides screw terminals to simplify connection to the D type connectors.

The ADC-16 can be used with our signal conditioning range to enable connection of a wide range of sensors and transducers.

Specification

- No of channels: 8.
- Resolution: 16 bit + sign.
- Input range: $\pm 2.5\text{V}$.
- Overload protection: $\pm 30\text{V}$.
- Sampling rate: 1.5Hz.
- Accuracy: 0.2%.
- Input impedance: $1\text{M}\Omega$.
- Input connector: D25 female.
- Output connector: D9 male to PC serial port.
- Outputs: 2 (fixed $\pm 5\text{V}$ references).
- Supplied software: PicoLog for Windows (3.x, 95/98, NT/2000) Drivers & examples C, Pascal, Delphi, Visual Basic, HP VEE and LabView. A macro is also provided to collect data directly into an Excel spreadsheet.

2.2.8 PT-104 Platinum Resistance Thermometer Data Logger

The PT-104 is [11] a four-channel temperature measuring device. It offers the ultimate in resolution ($0.001\text{ }^{\circ}\text{C}$) and accuracy ($0.01\text{ }^{\circ}\text{C}$) when used with PT100 sensors. It can also be used to measure resistance and voltage, Measures temperature, resistance and voltage, High resolution and accuracy, No power supply required, For use with PT100 and PT1000 sensors.

Accuracy and Resolution

Although accurate temperature sensors are widely available, it has been difficult to take advantage of them due to errors caused by the measuring device. The PT104 however, is inherently accurate due to its novel design. Rather than relying on voltage references (which tend to be temperature sensitive) it uses 'reference' resistors, which are extremely stable (low temperature co-efficient and drift). The exact value of each resistor is stored in an EEPROM to provide the ultimate in accuracy (yearly re-calibration is recommended). To achieve the 0.001 °C resolution a state of the art ADC is used that can resolve to better than 1 part in 16 million.

Temperature

The PT104 measures temperature using platinum resistance thermometers (PRTs). Both common industry standards (PT100 and PT1000) are supported. The unit is compatible with 2, 3 and 4 wire sensors (4 wire PT100 sensors are recommended for accurate measurements).

Resistance

When measuring resistance, the unit uses a four-wire circuit to give the greatest possible accuracy.

Two resistance ranges are available (0 to 375 ohms and 0 to 10,000 ohms). The unit is calibrated for 0 to 375 ohms so this range should be used for accurate measurements.

Voltage

For voltage measurements, each input connector can be treated as a differential input with ground, or two single ended inputs. Both inputs must be zero volts or above, though it does not matter which input has the higher voltage.

Two voltage ranges are available (0 to 115mV and 0 to 2500mV). For accurate measurements use the 0 to 2500mV range.

Software

The PT-104 is supplied with PicoLog data logging software for advanced data logging. PicoLog will automatically detect which sensor is connected and will display readings in the correct units. For users who wish to write their own custom software, drivers and examples are supplied.

Remote Data Collection

The PT104 is normally connected directly to a PC, but it is also possible to communicate with the PT104 using a modem (radio or telephone) so that you can collect data from a remote site.

Table 2.5: PT-104 Specification

	Temperature	Resistance	Voltage
Sensor	PT100*, PT1000	N/A	N/A
Range	-200..800°C	0..375 ohms* 0..10k ohms	0..115mV 0..2.5V*
Linearity	10ppm	10ppm	10ppm
Accuracy @25°C	0.01°C*	20ppm*	0.2%*
Temperature coefficient	3ppm/°C	3ppm/°C	100ppm/°C
RMS Noise (using filter)	0.01°C	10ppm	10ppm
Resolution	0.001°C	1uohm	0.156uV
Conversion time per channel	720mS **	720mS **	180mS
Number of inputs	4		
Connectors	4-pin miniDIN		
Input impedance	>>1Mohm		
Overvoltage protection	+/-100V		
Output	RS232, D9 female		
Environmental	20 to 30°C for stated accuracy 0 to 70°C overall 20 to 90% RH		
Software	Drivers for Windows 3.1/95/98/NT/2000/ME/Linux Examples for C, Delphi, Excel, Labview, Labwindows, HP-VEE		

2.2.9 Multi Channel Data Acquisition

Pico Technology supplies [12] a range of easy to use PC based data acquisition products. The units simply plug into the parallel or serial port of a PC (desktop or laptop) and require no power supply. Used with PicoLog data acquisition software, they replace costly chart recorders and complicated 'plug in' data acquisition boards. These PC based data

acquisition products allow your computer to display and record voltages. By connecting suitable sensors, they can be used to measure temperature, pressure, humidity, light, resistance, current, power, speed, vibration... in fact, anything that you need to measure. In addition, the parallel port products are supplied with PicoScope software, which allows them to be used as oscilloscopes, spectrum analyzers and multimeters, Easy to install and use - up and running in minutes, Supplied with PicoLog data acquisition software, Parallel port products are also supplied with PicoScope virtual instrument software, Low cost data acquisition systems.

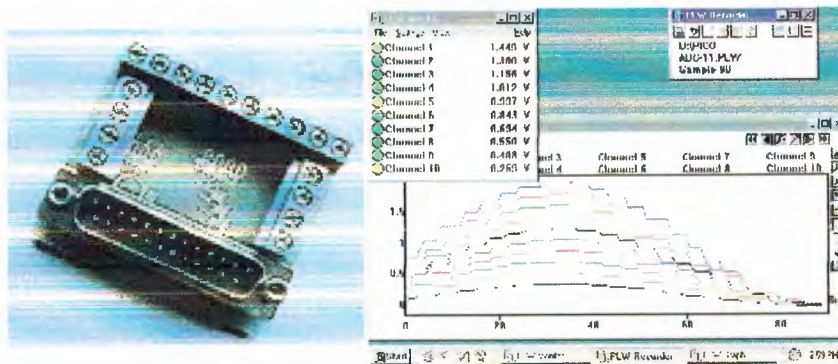


FIG 2.4 Multi channel data acquisition products

Features

- Large number of channels.
- Compact design.
- Digital output for control purposes.
- Data acquisition software included.
- The ADC-11 connects to the PC parallel port and provides 11 channels of analog input in a case slightly larger than a matchbox.
- The ADC-11/10 has 10-bit resolution, whilst the ADC-11/12 has 12-bit resolution making it more suitable for detecting small signal changes.
- The ADC-11 terminal block provides screw terminals to simplify connection to the inputs.
- The ADC-22 connects to the PC parallel port and is similar to the ADC-11/10, but provides 22 channels.

- A digital output is provided for control/alarm functions, in addition this output can also be used to power sensors such as thermostats.

The ADC-11s and ADC-22 are ideal for many different applications. Some examples listed below:

- Measuring the speed of a model car - shows the use of light gates.
- Measuring the swing of a pendulum.
- Wind resistance and terminal velocity

Specifications:

- No of channels: 11 (ADC-11), 22 (ADC-22).
- Resolution: 10-bit (ADC-11/10 and ADC-22), 12-bit (ADC-11/12).
- Input range: 0 to 2.5V.
- Overload protection: $\pm 30V$.
- Sampling rate: 10kHz.
- Accuracy: $\pm 1\%$ (ADC-11/10 and ADC-22), $\pm 0.5\%$ (ADC-11/12).
- Input connector: D25 female.
- Input impedance: $>1MW$.
- Output connector: D25male to PC parallel port.
- Outputs: 1 (digital).
- Supplied software: (The ADC-11/12 is not supported under DOS), PicoLog for Windows 3.x, 95/98, NT/2000). PicoScope: DOS and Windows (3.x, 95/98, NT/2000).
- Drivers & examples: C, Pascal, Delphi, Visual Basic, HP VEE and LabView. A macro is also provided to collect data directly into an Excel spreadsheet.

2.2.10 ADC-10 & ADC-12 -single channel, low cost data loggers

Features:

- Lowest Cost.
- No power supply required.

- Compact design.
- Data logging software included.

The ADC-10 and ADC-12 are [13] low cost data loggers that provide your PC with a single channel of analog input. They plug directly into the PC parallel port and require no external power, so they are ideal for use with both portable and desktop PCs.

The ADC-10 has 8-bit resolution, whilst the ADC-12 has 12-bit resolution: it is more suitable for detecting small signal changes.

Both units are supplied with both PicoLog software (for data logging) and with PicoScope virtual instrument software.

Specifications:

- No of channels: 1.
- Resolution: 8-bit (ADC-10), 12-bit (ADC-12).
- Input range: 0 to 5V.
- Overload protection: $\pm 30V$.
- Sampling rate: 20kS/s (ADC-10), 15kS/s (ADC-12).
- Accuracy: 1%.
- Input impedance: 200KW.
- Input connector: BNC.
- Output connector: D25 to PC parallel port.
- Supplied software: PicoLog for Windows (3.x, 95/98, NT/2000) PicoScope: DOS and Windows (3.x, 95/98, NT/2000)
- Drivers & examples: C, Pascal, Delphi, Visual Basic, HP VEE and LabView. A macro is also provided to collect data directly into an Excel spreadsheet.

2.3 COMPARISON BETWEEN THE ABOVE SYSTEMS:

Table 2.6: Comparison Between The Above Systems

Product	1	2	3	4	5
Channels	1	4	1	4	2
Resolution	1 sec	12 bit	8 bit	12 bit	-----
Input Ranges	-----	5.5 V	3 V	-----	-----
Sampling Rate	-----	-----	1-255S/m	25 S/s	1 S/s
Outputs	-----	-----	-----	-----	-----
Accuracy	-----	-----	±05%	-----	-----
Power	10-year battery	10 year	10 year	10 year	10 year
Operating	0°C to 60°C	-40 to 70°C	-10 to 85°C	-----	-40 to 85°C
Memory	192k	32k	2k	128k	32k

Product	6	7	8	9	10
Channels	2	8	4	11 or 22	1
Resolution	-----	16 bit	-----	10 bit	8 bit
Input Ranges	-----	± 2.5 V	0-2.5 V	0-2.5 V	0-5 V
Sampling Rate	1 S/s	1.5 Hz	-----	10 KHz	20 KS/s
Outputs	-----	± 5V	-----	1 (TTL)	-----
Accuracy	±1%	0.2%	-----	± 1%	1%
Power	10 year	-----	-----	From PC	From PC
Operating	-20 to 70°C	-----	-200to500°C	-----	-----
Memory	32k	-----	-----	-----	-----

2.4 Summary

This chapter has described the features of various data loggers in the market.

3. DESIGNING A DATA LOGGER SYSTEM

3.1 Overview

The previous chapter has described the features of the popular commercially available data loggers in the market.

This chapter will give the details of the data logger system designed and developed by the author. Briefly, the following topics will be described in this chapter:

- Description of the Data Logger designed by the author
- Hardware of the Data Logger
 - 1- Power supply
 - 2- Sensors
 - 3- The PIC16F877 Microcontroller
 - 4- Interface using the Serial / RS232 Port

3.2 Description Of The Data Logger Designed By The Author

Figure 3.1 shows the block diagram of the data logger designed by the author. The data logger basically consists of a microcontroller, a serial communication level converter, and a PC. The operation of the system is as follows:

The microcontroller used in the thesis has 8 analog input channels and 16 digital input channels, organised as two ports, each ports being 8 bits wide. The microcontroller is self-operating under a program developed by the author. Basically, analog sensors are connected to the analog inputs, and digital sensors and digital status signals are connected to the digital inputs. The program inside the microcontroller reads the analog and the digital sensor data every second. This data is then organized in the form of a packet and is sent to the serial output port (RS232) of the microcontroller. The PC is connected to the microcontroller through the RS232 serial communications line. A PHP script based

program, developed by the author runs on the PC and this program collects the serial data sent by the data logger. The data received by the PC is stored in a database for further analysis.

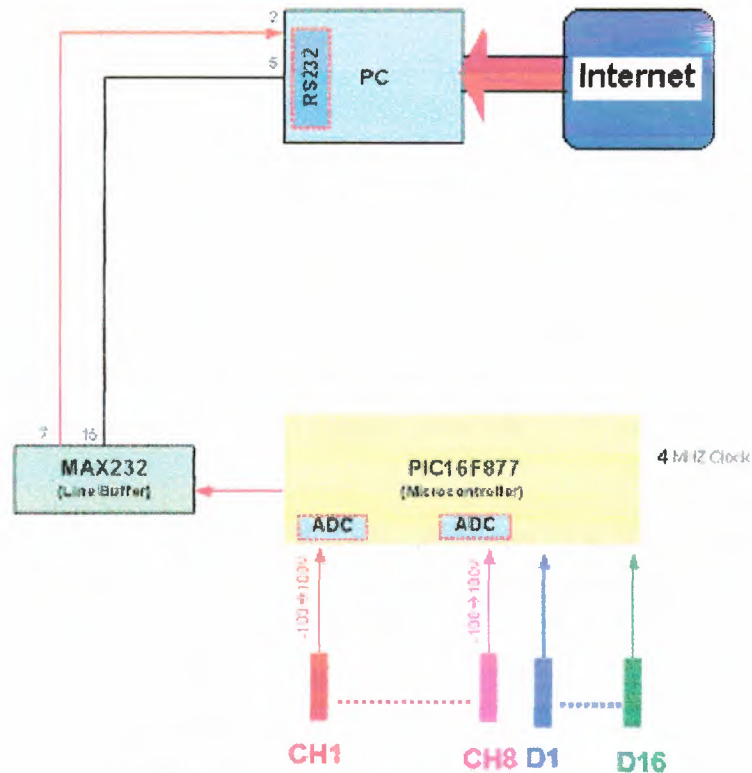


FIG 3.1 Block Diagram Of The Data Logger

Features of the Data Logger Designed By the Author:

- Low-cost.
- 10-bit resolution.
- 8 Analog channels.
- 16 Digital channels.
- Have 368 bytes of RAM.
- The Universal Synchronous Asynchronous Receiver Transmitter (USART) is also known as the Serial Communications.
- Interface or SCI.

Data Logger Specifications:

- Number of Analog Channels: Eight (One for temperature, one for pressure, and 6 for the other's Analog signals).
- Number of Digital channels: 16.
- Memory Sizes: 386 byte.
- Resolution: 8 bits.
- Input range: 0 - 5V.
- Overload protection: $\pm 30V$.
- Baud rate: 9600 bit /sec.
- Accuracy: 0.2%.
- Input impedance: $1M\Omega$.
- Power Source: Internal lithium battery with life of 10 years at 1 second sampling rate.
- Supplied software: MPLAB (PICC) for Windows (3.x, 95/98, NT/2000) Drivers & examples C, Tabling and Storing Data using PHP & MySQL.

3.3 Hardware Of The Data Logger

Figure 3.2 shows the complete hardware circuit diagram of the data logger. As can be seen from this figure, the circuit is based on a PIC16F877 type microcontroller. AN0-AN7 are the analog input channels of the microcontroller. Similarly, PORTC and PORT D are the digital input/output channels of the microcontroller and they are configured as digital inputs in this project. PIC16F877 has another input/output port called PORT B and bit 0 of this port is used as the RS232 serial communications line. A MAX232 type RS232-TTL level converter chip is connected to this port. Serial data has been generated under software control. The output of the MAX232 chip is connected to a 9-pin D-type RS232 connector which is then plugged-in to the COM1 serial port of a PC. A laptop PC is used in this project, operating under the Windows 2000 operating system.

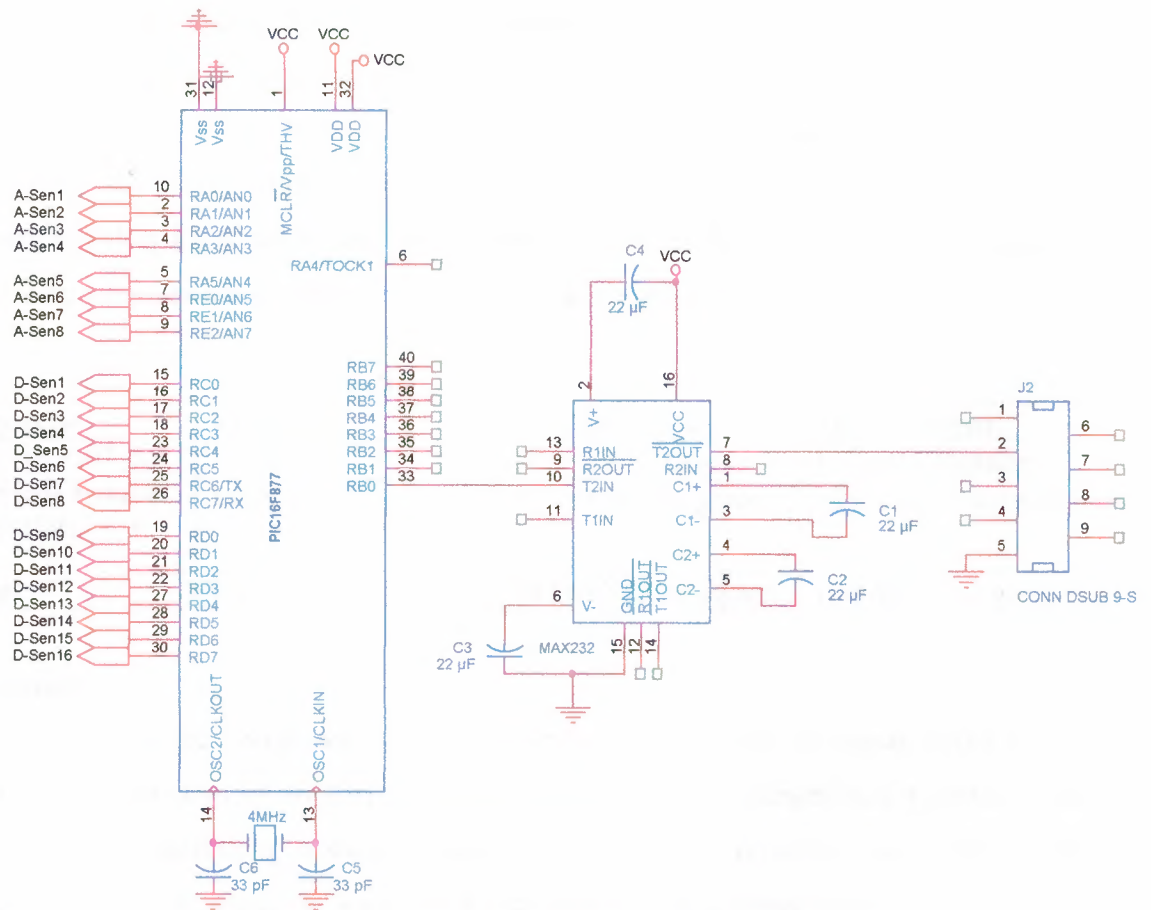


FIG 3.2 Circuit Diagram Of The Data Logger

More information about the various hardware components used in the design are given below:

3.3.1 Power

One of the tightest constraints [14] on the circuit design was the operating voltage. Assuming the device would be operated using batteries, careful attention was paid to the available operating voltage. Because the chip sets in the design required a very narrow operating range centered at 5 Volts, a 5V portable power source was used. Table 3.1 compares three types of batteries. Lithium batteries not only offered the greatest energy density but also operated at the desired Voltage. Even though Lithium batteries had the highest energy density, they did not, however, have the highest current capacity. The Nickel-metal hybrid batteries could output over 260 times more current per unit mass.

Aside from the fact that adding a voltage regulator would complicate the design of the Circuit, voltage regulators required additional power to run, added noise to the system, and increased the parts count for the Circuit design. Virtually all microprocessors on the market can go into low power modes. However, the author found that very few processors consumed less than 1 μ W while sleeping. It was important to have a very low sleep current because long sleep periods could drain a significant amount of the battery.

Table 3.1: Battery types

Battery type	Voltage	Energy Density	Max Current
Alkaline AA P107_ND	1.5 V	90 mWatt-hrs/gram	130mA (24grams)
Nickel_Metal Hybrid (rechargeable) P014-ND	1.2 V	55 mWatt-hrs/gram	>2600mA(26 grams)
Lithium	3.0 V	285 mWatt-hrs/gram	10mA(10.5 grams)

3.3.2 Sensors

The sensor is [15] the component of an instrument that converts an input signal into a quantity that is measured by another part of the instrument and changed into a useful signal for an information-gathering system or we can also define the sensor as a device that responds to a physical stimulus, such as thermal energy, electromagnetic energy, acoustic energy, pressure, magnetism, or motion, by producing a signal, usually electrical.

There are multiple ways of sensing and estimating just about every physical attribute of the earth, atmosphere, and aircraft. Physical sensors that provide the raw data vary in quality, reliability, and the extent to which their values must be filtered and combined with others to obtain useful estimates.

Sensors provided the raw data that needed to gain a high level understanding of the environment. With that in mind, I chose sensors which extracted telling information from their surrounding.

The sensors themselves could be classified into two groups:

- Weather monitors.
- Motion detectors.

The following sections elaborate further on the distinction between the two groups. Table 3.2 summarizes the sensors that can be used .

Table 3.2: Sensor Specifications

	Current consumption	Voltage range	Min range/ Max range	Accuracy	Temperature dependence	Product
Magnetometer	650μA	2.7-5.25V	-/+0.5 Gauss	2 mGauss	1.4mG / °C	AA002-02 NVE
Accelerometer	600μA	3-5.25V	-/+2g	25 mg	Negligible	ADXL202 Analog Devices
Light sensor	200μA	2.7-5.5V	0 mW/m2 / 26 mW/m2	6 mW/m2	Negligible	H53371 Edmund Scientific Silicon Detector
Temperature Sensor	600μA	2.7-5.5V	-20°C/100°C	0.25°C	not applicable	AD7418 Analog Devices
Pressure sensor	650μA	2.7-5.5V	0.6 PSI gauge range @ 14.4 PSI absolute	2.4 mPSI	10 mPSI/°C	SM5310 SMI
Humidity sensor	200μA@5V	4-9V	0-100% relative humidity	+/-2% RH	Negligible	HHH-3605 Hy-Cal

3.3.2.1 Magnetometer

Magnetometers [16] are sensors that measure magnetic fields. They can measure the 60 Hz fluctuations from power lines or the Earth's naturally occurring magnetic field.

To implement magnetometers into my project, The circuit shown in Fig. 3.3 is used. The magnetometers were followed by a two-stage amplifier design with a total voltage gain of 284. The analog signal was then digitized by a 10 bit ADC.

Because a Lithium battery's voltage dropped from 3.3V to 2.7V over its lifetime, the circuit was designed to withstand power supply fluctuations. The output of the circuit was described by Equation 1:

$$\frac{V_{out}}{V_{cc}} = G_2 \left(\frac{\Delta R}{2R} G_1 + \frac{1}{2} - \frac{R_{variable}}{R_{totalpot}} \right)$$

If V_{cc} fluctuated, so did V_{out} , but since the output was sampled by an A/D converter whose reference point was V_{cc} , the digital value was left unchanged, at least to first order.

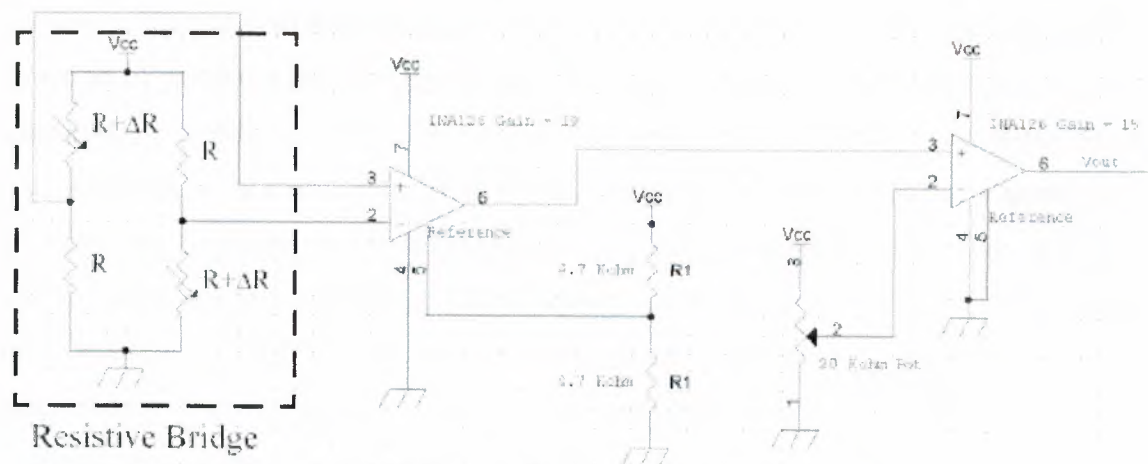


Fig. 3.3. Circuit diagram of magnetometers.

The magnetometers are designed such that they can detect the Earth's magnetic field. Since each magnetometer can measure the magnetic intensity along a single axis only, three mutually perpendicular magnetometers were needed to extract the full vector of the Earth's magnetic field. The Earth vector rotated in the frame of the device, allowing the device to detect rotational movement. Going one step further, since the Earth's magnetic field changes both in intensity and direction over the globe, the device could theoretically be able to extract the geographic latitude.

Though the magnetometers are accurate to 2 mGauss, they were vulnerable to large magnetic fields. Fields in excess of 15 Gauss permanently changed the sensitivity of the sensors, requiring a recalibration.

3.3.2.2 Accelerometers

The Analog Devices [17] ADXL202 is an MEMS type accelerometer which uses capacitive sensing to measure distance between a reference mass and a proof mass. The output of the ADXL 202 is a pulse-width modulated signal whose duty cycle is proportional to

acceleration. The microprocessor measures the period of the pulses to determine the correct acceleration measurement.

The word, accelerometer, is a bit of a misnomer because force is the unit really being measured. The most striking example is that accelerometers can measure the magnitude and direction of gravity. Like the magnetometers, three mutually perpendicular accelerometers were needed to fully resolve the magnitude and direction of any force. In a static situation (i.e. when the sensor was immobile), the gravity vector was the only force acting on the accelerometers. Partial orientation information could be obtained based on the gravity vector's relationship to the frame of the sensor, much the same way the Earth's magnetic field yielded orientation information. The gravity vector in conjunction with the magnetometers could provide exact orientation information. Of course, under actual acceleration, the gravity vector could not be extracted.

3.3.2.3 Light Sensor

The light sensor [18] is designed with a photodiode and transresistance amplifier (Fig. 3.4). Depending on the resistor chosen, the light sensor's sensitivity can be adjusted for either sunlight or room light. The output of the amplifier was then digitized by a 10 bit ADC.

The bandwidth of the light sensor on the Laser Mote was over 1KHz. The light sensor was a communications receiver where modulated light was used to send binary data.

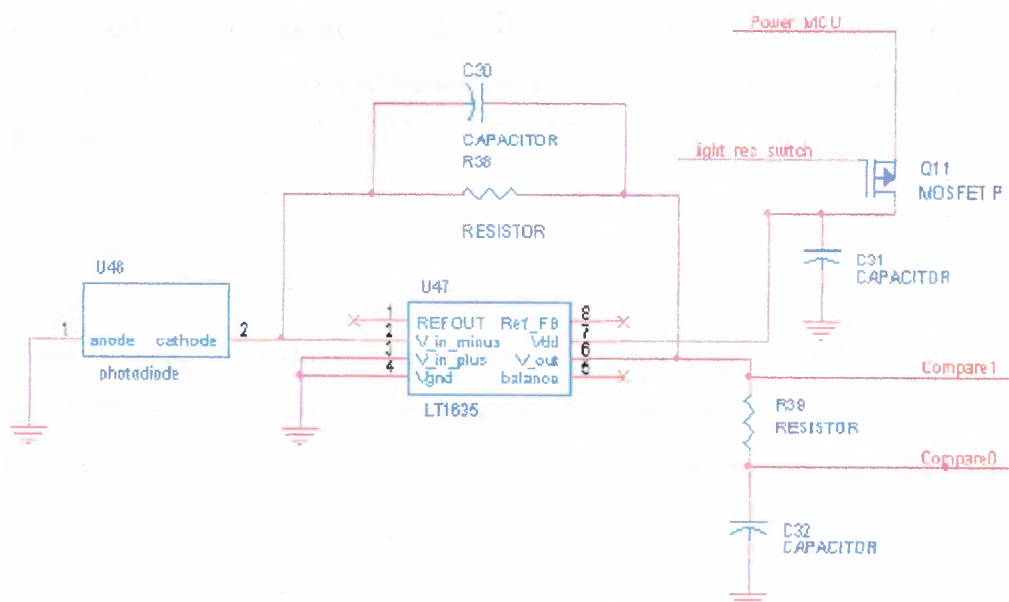


Fig. 3.4 Circuit diagram of light sensor.

3.3.2.4 Temperature Sensor

A large distinction can be made [19] between temperature sensor types. They can be classified in many ways. One perspective is to simply classify them into two large groups, contact and non-contact. Because the contact-vs-noncontact approach seems to be a very obvious difference due to the physical nature of the sensor's interaction with the object of measurement.

How to choose one type over another? Often that is not clear and sometimes not very significant. It depends greatly on the measurement context and precision or accuracy requirements. So, it is wise to understand those at the outset, if at all possible.

Both types of sensors require some assumptions and inferences to measure temperature. Many, many well-known uses of these sensors are very straightforward and few, if any, assumptions are required. Other uses require some careful analysis to determine the controlling aspects of influencing factors that can produce errors and which, in turn, can make the apparent temperature quite different from the true temperature.

All sensor readings have built-in errors. One secret to high quality measurement results is to make error estimates with a high level of confidence. Neglecting to make a realistic and careful error analysis often results in errors much larger than the assumed values, because assumptions are often unrealistic, based on lore and not fact. Further, they can lack one or more significant contributions or not have individual errors correctly combined.

Also, it is worth noting that all realistic error analyses start with the uncertainties assigned to the traceable calibration of the sensor itself. Without traceable calibration, one is forced to make assumptions.

Contact Sensors:

Contact temperature sensors measure their own temperature. One infers the temperature of the object to which the sensor is in contact by assuming or knowing that the two are in thermal equilibrium, that is, there is no heat flow between them.

Contact sensors come in a wide array of types, sizes, measurement capabilities and prices. There are numerous types but perhaps the best-known are the thermometers used in clinical or human body temperature measurements. Even these have a largely increased number of subvariants today. From the glass with silvery mercury filling (going away rapidly because

of environmental and health damage that mercury can cause), to the IR ear thermometer made popular by Braun's Thermoscan.

Noncontact Sensors

Most commercial and scientific non-contact temperature sensors [20] measure the thermal radiant power of the Infrared or Optical radiation that they receive and one then infers the temperature of an object from which the radiant power is assumed to be emitted. IR thermometers dominate this group but they have their variations, too. One of the major differences are the Spot versus Area-measuring IR thermometers. The latter are better known as quantitative or radiometric thermal imaging cameras and they are usually used by skilled and trained operators called "Thermographers".

The output of the temperature sensor used the I²C protocol to transmit digital information to the microcontroller. With a full range from -20°C/100°C the AD7418 required no calibration and had an accuracy down to 0.25°C.

3.3.2.5 Pressure Sensor

The SM5310 [21] is an uncompensated silicon sensor chip whose resistance changes as a function of pressure. The device comes in a resistive H bridge package, and its output is amplified by a two-stage amplifier design similar to that of the magnetometer (Fig. 3.5). The output is then digitized by a 10 bit ADC.

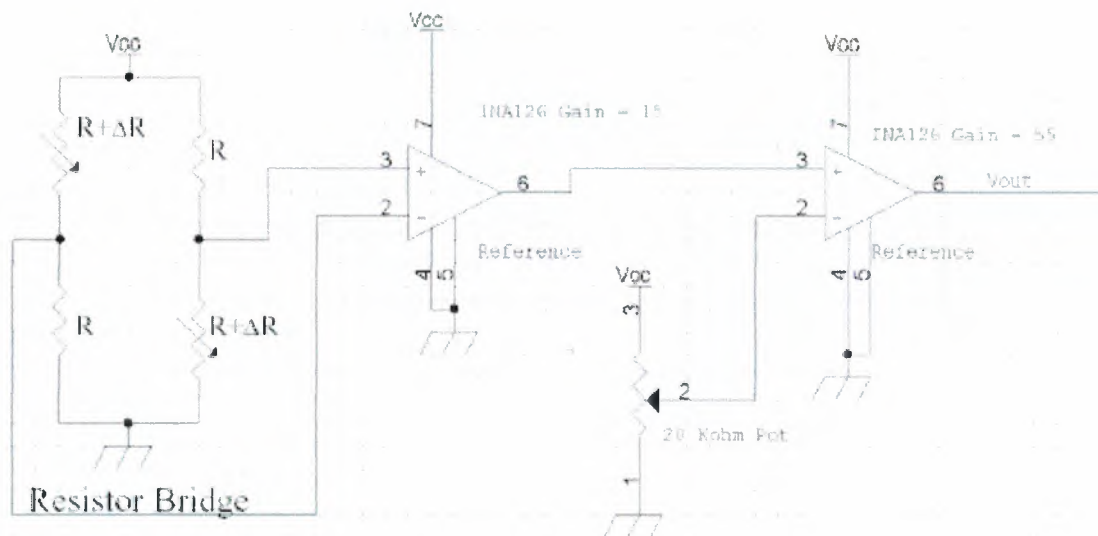


Fig. 3.5 Circuit diagram of pressure sensor.

In addition to providing weather information, the pressure sensor could be used as a motion sensor. Since pressure decreases at higher altitudes, the sensor could act as an altimeter. Referring to Table 3.2, the accuracy of the pressure sensor circuit was 2.4 mPSI, and according to Table 3.3, the change in pressure as a function of altitude is approximately 0.5 mPSI/foot, suggesting the sensor could detect altitude changes of 5 feet or greater.

While the circuit could detect small gradations in altitude, it had a strong dependence on temperature, namely 10 mPSI/°C. In other words, in order for the pressure sensor to maintain its sensitivity of 2.4 mPSI, the temperature would have to be accurately measured to within 0.25 C for calibration purposes. Further work needs to be conducted to verify the true accuracy of the pressure sensor.

Table 3.3: Pressure vs. Altitude

Altitude Above Sea Level (feet)	Equivalent Pressure (PSI)	Approximate Slope mPSI/ foot
0	14.7	-0.52
3000	13.17	-0.48
6000	11.78	-0.44
8000	10.91	-0.43

3.3.2.6 Humidity Sensor

The capacitive humidity sensor [22] is a fully integrated circuit whose output voltage is linearly proportional to relative humidity. Its low power design (200 μ A) and ease of integration made it an almost ideal candidate to the suite of sensors used. However, its specified voltage range of 3.9V-9V made it difficult to integrate with the Lithium batteries. Qualitatively, the humidity sensors operated at 3 Volts, but quantitative analysis was not performed to determine the degradation in accuracy.

3.3.3 The PIC16F877 Microcontroller

The important features [23] specific to picking a microprocessor were 1) power, 2)memory size, 3) fast reprogrammability, 4) A/D channels, and 5) a 5 volt operating supply.

Table 3.4 compares the most suitable microprocessors on the market. Though not an 8 bit processor, the Strong Thumb processor was included both because of its commercial popularity and low power consumption. At the time the Atmel AT90LS8535 offered the best performance even though newer designs like the Microchip PIC16F877 offered lower power consumption and greater functionality.

Table 3.4: Comparison of Microprocessors

	Atmel AVR AT90LS8535	Microchip PIC16F877 (preliminary)	MC68H(R)C 908JL3	Atmel AT91M404000 16/32 bit Strong Thumb
Flash Memory	4K	8Kx14	4K	external memory
Endurance	1k	1k	10k	N/A
MIPS/mA	1.25 (min)	1.66 (preliminary)	0.1 (typical)	0.6 MIPS/mA (1.35 mA static current)
A/D channels	8 (10 bit)	8 (10 bit)	12 (8 bit)	0
In-application programming (IAP)	No	Yes	Yes	Yes
Operating voltage	2.7-5.5V	2.0 – 5.5V	2.7-3.3V	2.7V-3.6V
I/O pins	35	40	23	100

The PIC16F877 is a high-performance FLASH microcontroller that provides engineers with the highest design flexibility possible. In addition to 8192x14 words of FLASH program memory, 256 data memory bytes, and 368 bytes of user RAM, PIC16F877 also features an integrated 8-channel 10-bit Analogue-to-Digital converter. Peripherals include two 8-bit timers, one 16-bit timer, a Watchdog timer, Brown-Out-Reset (BOR), In-Circuit-Serial Programming™, RS-485 type UART for multi-drop data acquisition applications, and I²C™ or SPI™ communications capability for peripheral expansion. Precision timing interfaces are accommodated through two CCP modules and two PWM modules.

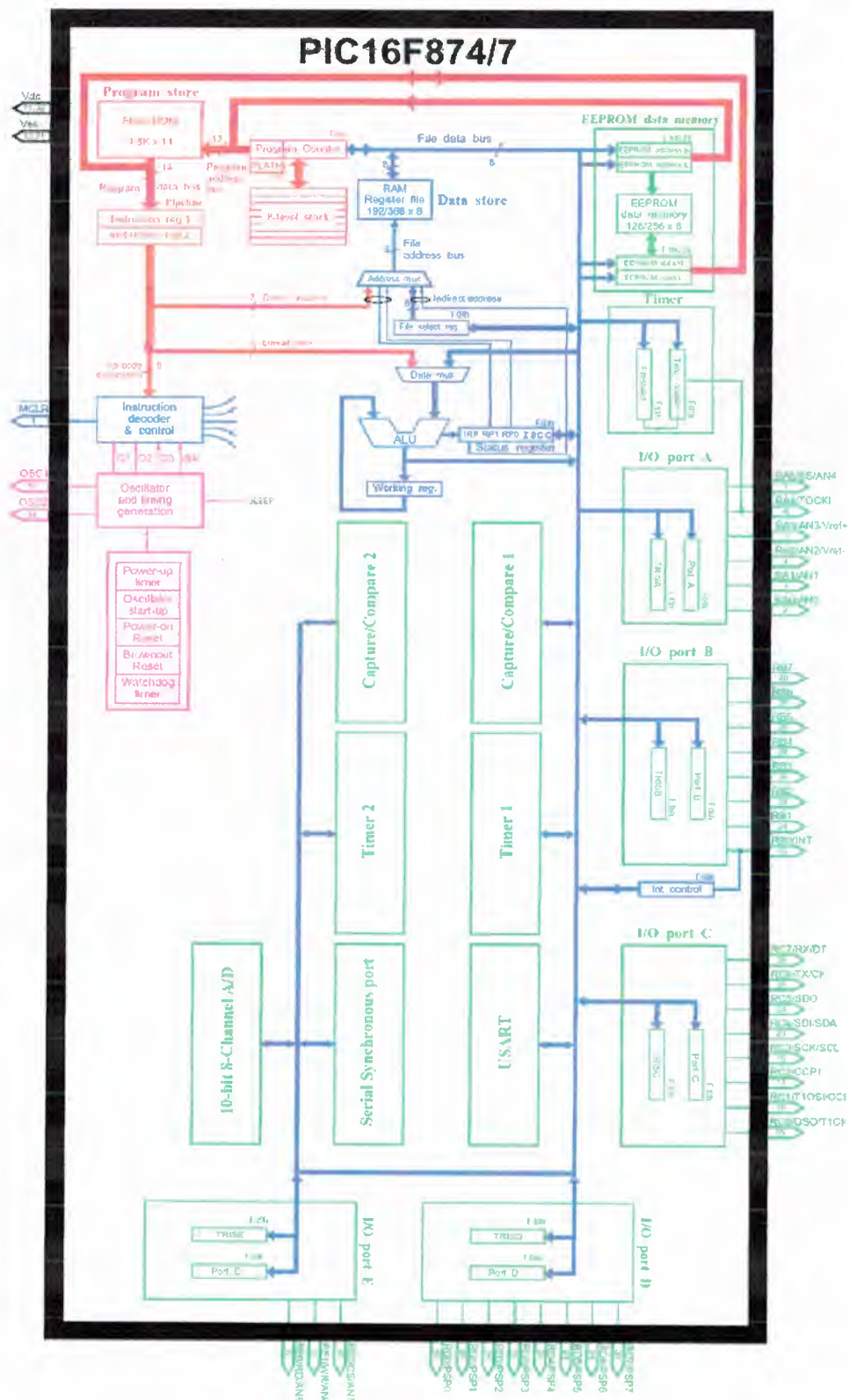


Fig 3.6 Architecture of the PIC16F877 microcontroller

3.3.3.1 Overview of the File Registers

The data memory is partitioned into multiple banks which contain the general purpose registers and the special function registers. Bits RP1 and RP0 are the bank select bits, these bits are found in the STATUS register (b6 & b5).

00h	Indirect addr. (*)	80h	Indirect addr. (*)	100h	Indirect addr. (*)	180h	Indirect addr. (*)
01h	TMR0	81h	OPTION_REG	101h	TMR0	181h	OPTION_REG
02h	PCL	82h	PCL	102h	PCL	182h	PCL
03h	STATUS	83h	STATUS	103h	STATUS	183h	STATUS
04h	FSR	84h	FSR	104h	FSR	184h	FSR
05h	PORTA	85h	TRISA	105h	Unimplemented	185h	Unimplemented
06h	PORTB	86h	TRISB	106h	PORTB	186h	TRISB
07h	PORTC	87h	TRISC	107h	Unimplemented	187h	Unimplemented
08h	PORTD	88h	TRISD	108h	Unimplemented	188h	Unimplemented
09h	PORTE	89h	TRISE	109h	Unimplemented	189h	Unimplemented
0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah	PCLATH
0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh	INTCON
0Ch	PIR1	8Ch	PIR1	10Ch	EECON1	18Ch	EECON1
0Dh	PIR2	8Dh	PIR2	10Dh	EEADR	18Dh	EECON2
0Eh	TMR1L	8Eh	PCON	10Eh	EEADTH	18Eh	RESERVED
0Fh	TMR1H	8Fh	Unimplemented	10Fh	EEADRH	18Fh	RESERVED
10h	T1CON	90h	Unimplemented	110h	General Purpose Register 96 Bytes	190h	General Purpose Register 96 Bytes
11h	TMR2	91h	SSPCON2	.		.	
12h	T2CON	92h	PR2	.		.	
13h	SSPBUF	93h	SSPAD0	.		.	
14h	SSPCON	94h	SSPSTAT	.		.	
15h	CCPR1L	95h	Unimplemented	.		.	
16h	CCPR1H	96h	Unimplemented	.		.	
17h	CCP1CON	97h	Unimplemented	.		.	
18h	RCSTA	98h	TXSTA	.		.	
19h	TXREG	99h	SPBRG	.		.	
1Ah	RCREG	9Ah	Unimplemented	.		.	
1Bh	CCPR2L	9Bh	Unimplemented	.		.	
1Ch	CCPR2H	9Ch	Unimplemented	.		.	
1Dh	CCP2CON	9Dh	Unimplemented	.		.	
1Eh	ADRESH	9Eh	ADRESL	.		1EFh	
1Fh	ADCON0	9Fh	ADCON1	.		1F0h	
20h	General Purpose Register 80 Bytes	A0h	General Purpose Register 80 Bytes	16Fh	Accesses Global Register 70h-7Fh	1FFh	Accesses Global Register 70h-7Fh
6Fh		EFh		170h		.	
70h		F0h		.	Accesses Global Register 70h-7Fh	.	
7Fh		FFh		.		.	
.		.		.	Accesses Global Register 70h-7Fh	.	
.		.		.		.	

Fig 3.7 PIC16F877 register file map

Each bank extends up to 7Fh (128 bytes). The lower locations of each bank are reserved for the special function registers (shown in yellow). Above the special function registers are general purpose registers (shown in blue), implemented as static RAM. All implemented banks contain special function registers. Some “high use” special function registers from one bank may be mirrored in another bank for code reduction and quicker access. Also notice that there are 16 general purpose global registers (shown in green), these registers can be accessed from any bank.

3.3.3.2 Overview of the 8-Channel 10-bit ADC

At first it appears that the PIC16F877 has 8 built-in ADCs, but this is not the case. Figure 3.8 shows a simplified block diagram of the analogue-to-digital converter module, clearly there is only one 10-bit ADC which can be connected to only one of eight input pins at any one time.

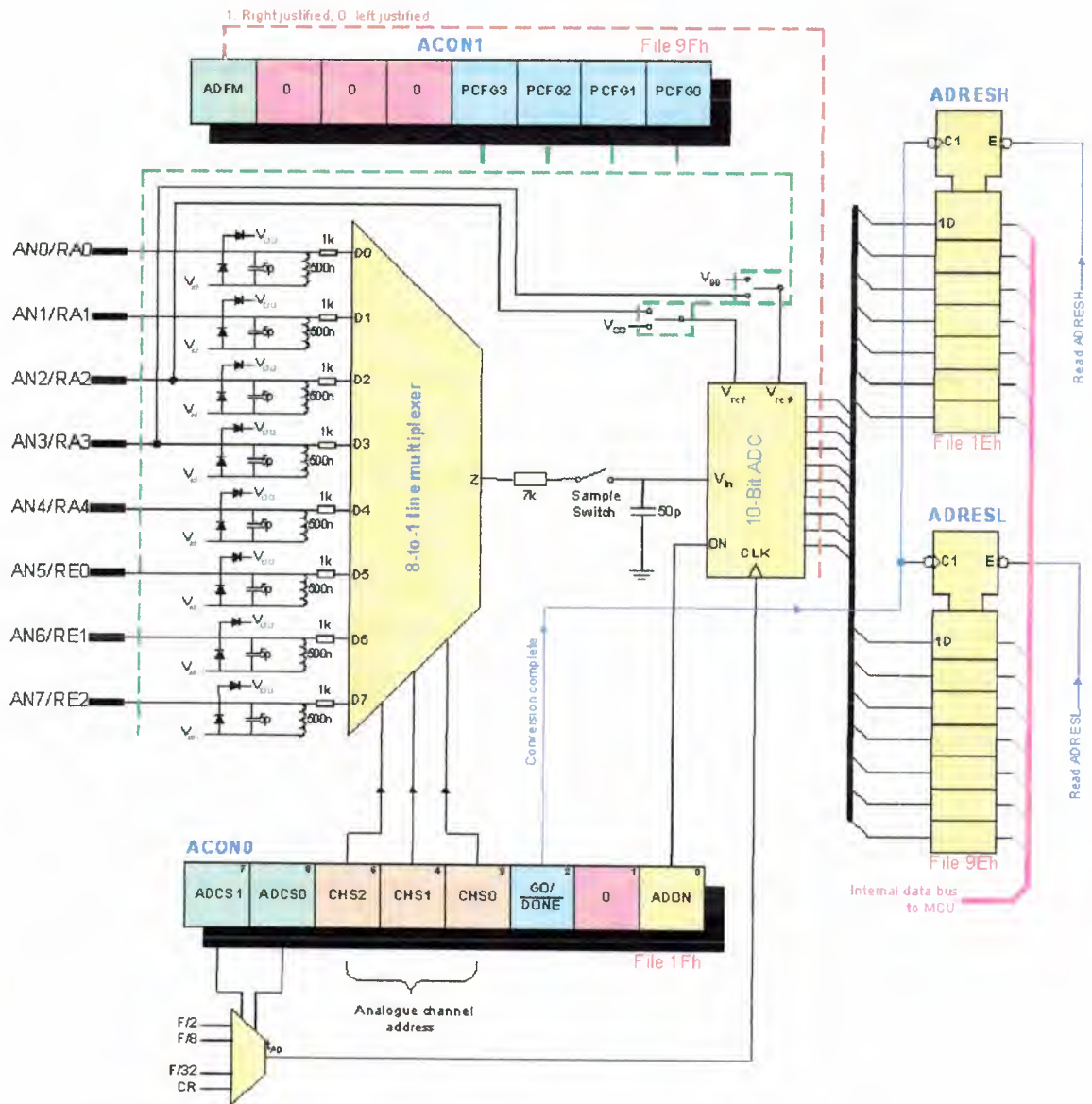


Fig 3.8 Simplified block diagram of the PIC16F877 ADC module

The input analogue channels AN4..0 are shared with port A, and channels AN7..5 are shared with port E. If less than eight analogue channels are required then some of the pins can be assigned as digital I/O port lines using PCFG3..0 bits. For example, if PCFG3..0 = 0010 then AN4..0 are configured as analogue inputs, while AN7..5 are digital (port E free), with VDD used as the reference.

On reset all pins are set to accept analogue signals. Pins that are reconfigured as digital I/O should never be connected to an analogue signal. Such voltage may bias the digital input buffer into its linear range and the resulting large current could cause irreversible damage.

The 10-bit ADC uses a technique known as 'successive approximation', the following mechanical analogy will help explain how it works.

The electronic equivalent to this successive approximation technique uses a network of precision capacitors configured to allow consecutive halving of a fixed voltage VREF to be switched in to an analogue comparator, which acts as the balance scale.

Generally the network of capacitors are valued in powers of two to subdivide the analogue reference voltage (e.g. 1,2,4,8,16, etc...). This sampling acquisition process takes a finite time due to the charging time constant and is specified in the datasheet as 19.72 μ S.

3.3.3.3 Overview of the USART Hardware

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules. The USART can be configured for asynchronous operation (UART) for communication with a PC or synchronous operation for communicating with peripheral devices such as DAC or DAC integrated circuit.

Note bit SPEN (RCSTA:7) and bits TRISC:7..6 have to be set in order to configure pin PC6/TX/CK and RC7/RX/DT for USART operation. CSS (C compiler) will automatically configure these bits, but it is important to be aware that if using fast_io(C) mode to manually configure port C, bits 7 & 6 must also be manually set if using the hardware UART.

3.3.3.3.1. Baud-Rate Generator, BRG

This is basically a programmable 8-bit counter followed by a switchable frequency flip flop chain which can be set up to give the appropriate sampling and shifting rates for the

desired baud rate, based on the PIC's crystal frequency XTAL (e.g. for 4MHz, XTAL = 4) giving: -

$$\text{Baud Rate (Low Speed Mode)} = \frac{XTAL}{64 \times (X + 1)}$$

$$\text{Baud Rate (Low Speed Mode)} = \frac{XTAL}{64 \times (X + 1)}$$

$$X = \frac{XTAL \times 10^6}{64 \times BAUD}$$

It may be advantageous to use the high baud rate (BRGH = 1) even for slower baud clocks as this may reduce baud rate error in some cases.

3.3.4 Interface By Serial / RS232 Port

The Serial Port [24] is harder to interface than the Parallel Port. In most cases, any device you connect to the serial port will need the serial transmission converted back to parallel so that it can be used. This can be done using a UART. On the software side of things, there are many more registers that you have to attend to than on a Standard Parallel Port (SPP).

Advantages of using serial data transfer rather than parallel :

Serial Cables can be longer than Parallel cables. The serial port transmits a '1' as -3 to -25 volts and a '0' as +3 to +25 volts where as a parallel port transmits a '0' as 0v and a '1' as 5v. Therefore the serial port can have a maximum swing of 50V compared to the parallel port which has a maximum swing of 5 Volts. Therefore cable loss is not going to be as much of a problem for serial cables than they are for parallel.

You don't need as many wires than parallel transmission. If your device needs to be mounted a far distance away from the computer then 3 core cable (Null Modem Configuration) is going to be a lot cheaper than running 19 or 25 core cable. However you must take into account the cost of the interfacing at each end.

Infra Red devices have proven quite popular recently. You may of seen many electronic diaries and palm top computers which have infra red capabilities build in. However could

you imagine transmitting 8 bits of data at the one time across the room and being able to (from the devices point of view) decipher which bits are which? Therefore serial transmission is used where one bit is sent at a time. IrDA-1 (The first infra red specifications) was capable of 115.2k baud and was interfaced into a UART. The pulse length however was cut down to $3/16^{\text{th}}$ of a RS232 bit length to conserve power considering these devices are mainly used on diaries, laptops and palmtops.

Microcontroller's have also proven to be quite popular recently. Many of these have in built SCI (Serial Communications Interfaces) which can be used to talk to the outside world. Serial Communication reduces the pin count of these MPU's. Only two pins are commonly used, Transmit Data (TXD) and Receive Data (RXD) compared with at least 8 pins if you use a 8 bit Parallel method.

3.3.4.1 RS232 Converter Chips

A standard serial interfacing for PC, RS232C, [25] requires negative logic, i.e., logic '1' is -3V to -12V and logic '0' is +3V to +12V. To convert a TTL logic, say, TxD and RxD pins of the uC chips, thus need a converter chip. A MAX232 chip has long been using in many uC boards. It provides 2-channel RS232C port and requires external 22uF capacitors. Carefully checking the polarity of capacitor when soldering the board. A DS275, however, no need external capacitor and smaller. Either circuit can be used without any problems.

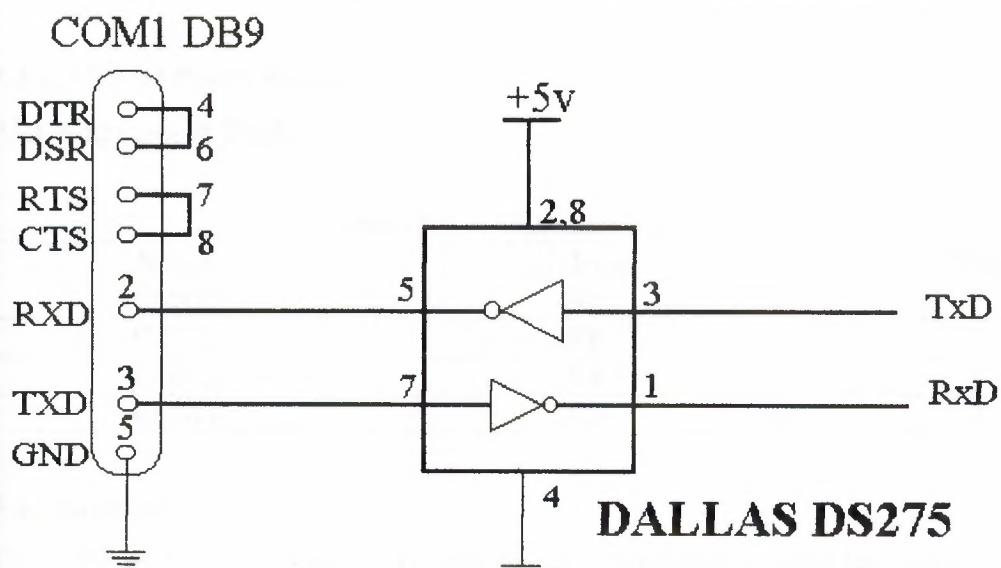
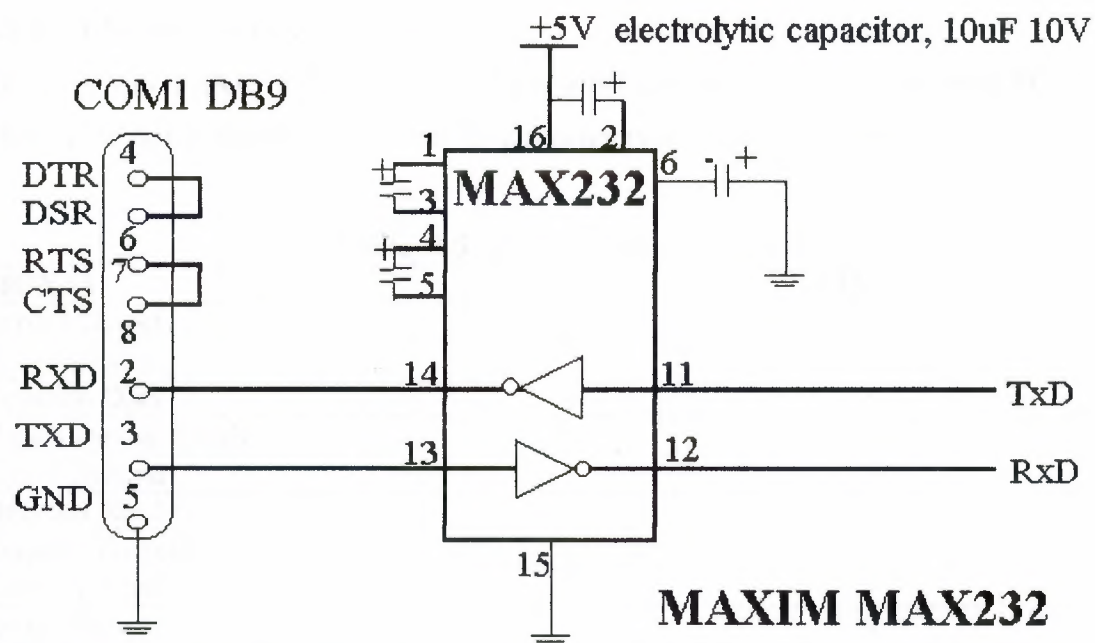


Fig 3.9 Converter Chips

3.3.4.2 RS232D (9 pin Connector)

The following table[24] illustrates the 9 pin serial connector as found on most PC's today. This has all but replaced the previous 25 pin connector found on earlier PC's.

Table 3.5: Pin Descriptions of DB9

SIGNAL	PIN No.
Carrier Detect	1
Receive Data	2
Transmit Data	3
Data Terminal Ready	4
Signal Ground	5
Data Set Ready	6
Request To Send	7
Clear To Send	8
Ring Indicator	9

3.3.4.3 Serial Port's Registers (PC's)

Port Addresses & IRQ's

Table 3.6 : Standard Port Addresses

Name	Address	IRQ
COM1	3F8	4
COM2	2F8	3
COM3	3E8	4
COM4	2E8	3

3.4. Summary

The goal of this project was to design and implement a complete data logger system, including the following components: the data logger, a graphical user interface that runs on a personal computer for viewing the logged data, and an intermediate communication station, to regulate communication between the microcontroller and the PC.

This chapter has describes the design of the data logger hardware.

4. DEVELOPMENT OF THE MICROCONTROLLER C PROGRAM USING PICC COMPILER

4.1 Overview

The PICC Lite [26] is a C compiler developed for the PIC series of microcontrollers. The compiler was designed primarily for the use of students and hobbyists who needed an inexpensive PIC compiler, for educational and small projects. The compiler itself is limited to only the 16C84, 16F84, 16F84A, 16F627, 12F629, 12F675, 16F877 and the 16F877A Microchip processors, it does not come with any library source code, and there is a limitation of two ram banks for general purpose ram (applicable to the 16F627, 16F877 and 16F877A). There is also a ROM size limitation of 2k with the 16F877 and 16F877A. You can perform 24 or 32 bit float calculations, though there is no support for printing (printf) long or float numbers.

This chapter covers HTL, the **HI-TECH C Programmer's development Lite** environment. At the end of the chapter the source listing and the details of the microcontroller C program developed by the author (PIC_PROGRAM.C) are given. The chapter also described the steps necessary to download the compiled program into the memory of a microcontroller.

4.2 Starting HTLPIC

To start HTLPIC, simply type `htlpic` at the MS-DOS prompt. After a brief period of disk activity you will be presented with a screen similar to the one shown in Figure 4.1.

The initial HTLPIC screen is broken into three windows. The top window contains the menu bar, the middle window the HTLPIC text editor, and the bottom window is the message window. Other windows may appear when certain menu items are selected. The editor window is what you will use most of the time.

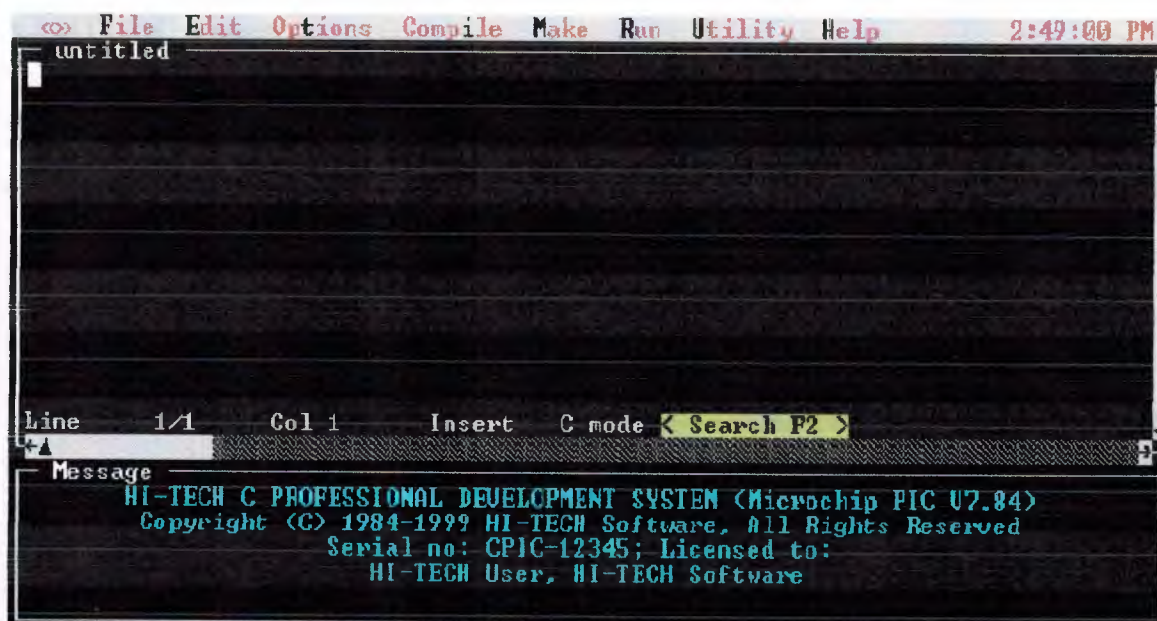


Figure 4.1 HTLPIC Startup Screen

4.3 Using HTLPIC

HTLPIC uses the HI-TECH Windows user interface to provide a text screen-based user interface. This has multiple overlapping windows and pull-down menus. The user interface features which are common to all HI-TECH Windows applications are described later in this chapter.

Alternatively, HTLPIC can use a single command line argument. This is either the name of a text file, or the name of a project file. If the argument has an extension .prj, HTLPIC will attempt to load a project file of that name. File names with any other extension will be treated as text files and loaded by the editor.

If an argument without an extension is given, HTLPIC will first attempt to load a .prj file, then a .c file.

For example, if the current directory contains a file called x.c and HTLPIC is invoked with the command: *htlpic x*

it will first attempt to load x.prj and when that fails, will load x.c into the editor. If no source file is loaded into the editor, an empty file with name untitled will be started.

4.4 HTLPIC menus

This section presents [26] an item-by-item description of each of the HTLPIC menus. The description of each menu includes a screen print showing the appearance of the menu within a typical HTLPIC screen.

4.4.1 <>> menu

The <>> (system) menu is present in all HI-TECH Windows based applications. It contains handy system configuration utilities and desk accessories which we consider worth making a standard part of the desktop.

About HTLPIC

The About HTLPIC dialog displays information on the version number of the compiler and the licence details.

Setup

This menu item selects the standard mouse firmware configuration menu, and is present in all HI-TECH Windows based applications. The "mouse setup" dialog allows you to adjust the horizontal and vertical sensitivity of the mouse, the ballistic threshold2 of the mouse and the mouse button auto-repeat rate.

This dialog will also display information about what kind of video card and monitor you have, what DOS version is used and free DOS memory available.

4.4.2 File menu

The File menu contains file handling commands, the HTLPIC Quit command and the pick list:

Open ... Alt-O

This command loads a file into the editor. You will be prompted for the file name and if a wildcard (e.g. "*.C") is entered, you will be presented with a file selector dialog. If the previous edit file has been modified but not saved, you will be given an opportunity to save it or abort the Open command.

New ... Alt-N

The New command clears the editor and creates a new edit file with default name "untitled". If the previous edit file has been modified but not saved, you will be given a chance to save it or abort the New command.

Save... Alt-S

This command saves the current edit file. If the file is "untitled", you will be prompted for a new name, otherwise the current file name (displayed in the edit window's frame) will be used.

Save as ... Alt-A

This command is similar to Save, except that a new file name is always requested.

Autosave

This item will invoke a dialog box allowing you to enter a time interval in minutes for auto saving of the edit file. If the value is non-zero, then the current edit file will automatically be saved to a temporary file at intervals. Should HTLPIC not exit normally, e.g. if your computer suffers a power failure, the next time you run HTLPIC, it will automatically restore the saved version of the file.

Quit... Alt-Q

The Quit command is used to exit from HTLPIC to the operating system. If the current edit file has been modified but not saved, you will be given an opportunity to save it or abort the Quit command.

Clear pick list

This clears the list of recently-opened files which appear below this option.

Pick list... ctrl-F4

The pick list contains a list of the most recently-opened files. A file may be loaded from the pick list by selecting that file. The last file that was open may be retrieved by using the short-cut ctrl-F4.

4.4.3 Edit menu

The Edit menu contains items relating to the text editor and clipboard. The edit menu is shown in Figure 4.2.

Cut ... Alt-X

The Cut option copies the current selection to the clipboard and then deletes the selection. This operation may be undone using the Paste operation. The previous contents of the clipboard are lost.

Copy... Alt-C

The Copy option copies the current selection to the clipboard without altering or deleting the selection. The previous contents of the clipboard are lost.

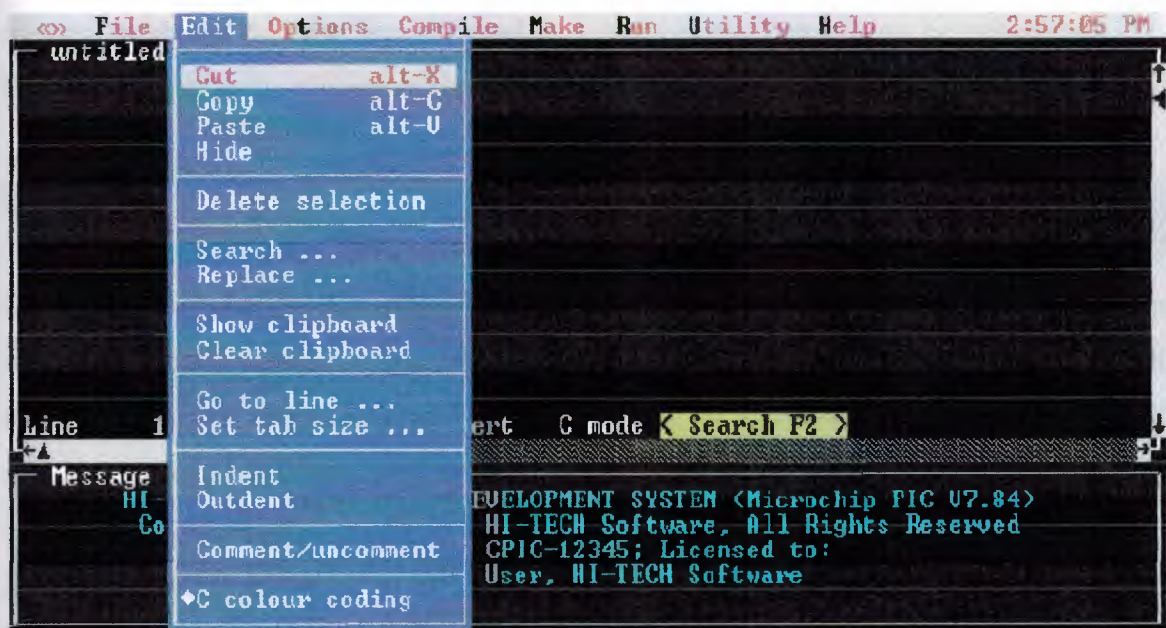


Figure 4.2 HTLPIC Edit Menu

Paste... Alt-V

The Paste option inserts the contents of the clipboard into the editor before the current line. The contents of the clipboard are not altered.

Hide

The Hide option toggles the current selection between the hidden and displayed state. This option is equivalent to the WordStar ctrl-K H command.

Delete selection

This menu option deletes the current selection without copying it to the clipboard. Delete selection should not be confused with Cut as it cannot be reversed and no copy of the deleted text is kept. Use this option if you wish to delete a block of text without altering the contents of the clipboard.

Search

This option produces a dialog to allow you to enter a string for a search. You can select to search forwards or backwards by selecting the appropriate button. You can also decide if the search should be case sensitive and if a replacement string is to be substituted. You make these choices by clicking in the appropriate brackets.

Replace

This option is almost the same as the search option. It is used where you are sure you want to search and replace in the one operation. You can choose between two options. You can search and then decide whether to replace each time the search string is found. Alternatively, you can search and replace globally. If the global option is chosen, you should be careful in defining the search string as the replace can not be undone.

Show clipboard

This menu options hides or displays the clipboard editor window. If the clipboard window is already visible, it will be hidden. If the clipboard window is currently hidden it will be displayed and selected as the current window. The clipboard window behaves like a normal

editor window in most respects except that no block operations may be used. This option has no key equivalent.

Clear clipboard

This option clears the contents of the clipboard, and cannot be undone. If a large selection is placed in the clipboard, you should use this option to make extra memory available to the editor after you have completed your clipboard operations.

Go to line

The Go to line command allows you to go directly to any line within the current edit file. You will be presented with a dialog prompting you for the line number. The title of the dialog will tell you the allowable range of line numbers in your source file.

Set tab size

This command is used to set the size of tab stops within the editor. The default tab size is 8, values from 1 to 16 may be used. For normal C source code 4 is also a good value. The tab size will be stored as part of your project if you are using the Make facility.

Indent

Selecting this item will indent by one tab stop the currently highlighted block, or the current line if there is no block selected.

Outdent

This is the reverse operation to Indent. It removes one tab from the beginning of each line in the currently selected block, or current line if there is no block.

Comment/Uncomment

This item will insert or remove C++ style comment leaders (//) from the beginning of each line in the current block, or the current line. This has the effect of commenting out those lines of code so that they will not be compiled. If a line is already commented in this manner, the comment leader will be removed.

C colour coding

This option toggles the colour coding of text in the editor window. It turns on and off the colours for the various types of text. A mark appears before this item when it is active.

4.4.4 Options menu

The Options menu contains commands which allow selection of compiler options, and target processor.

Selections made in this menu will be stored in a project file, if one is being used. The Options menu is shown in Figure 4.3.

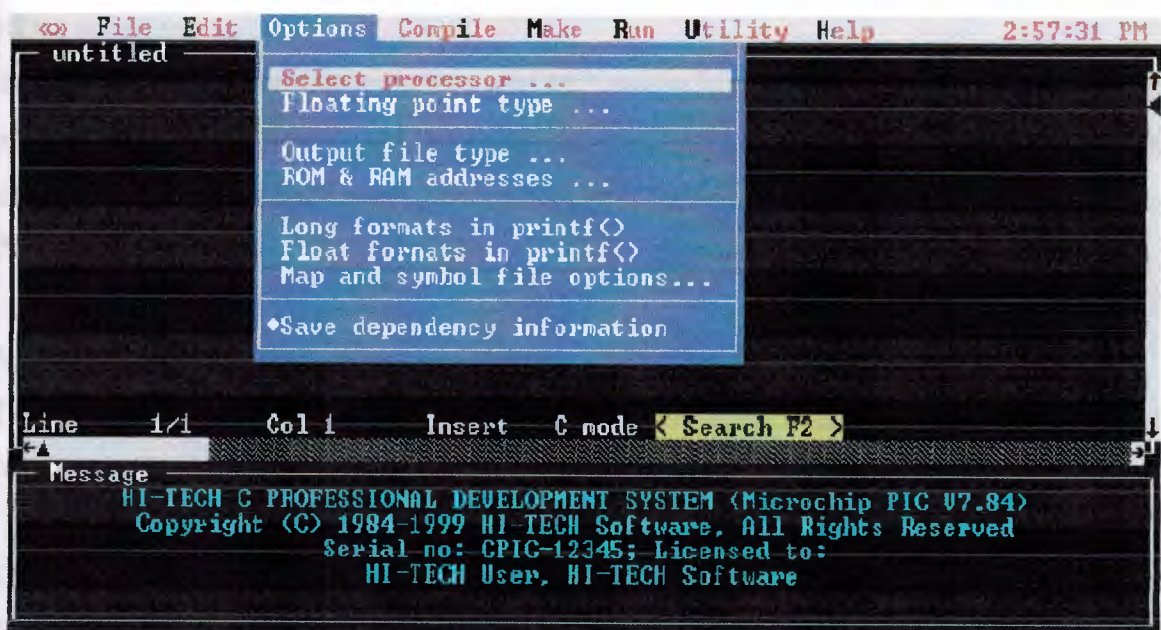


Figure 4.3 Options Menu

Select processor

This option activates a dialog box which allows you to select the processor type you wish to use. The help string at the bottom of the dialogue indicates the amount of ROM space, in words, that each device contains. In addition the string indicates the number of RAM banks the device contains and the total amount of general-purpose RAM bytes available. This is the amount of RAM other than that used by the special function registers. If the help string displays "including common" then some of the RAM space is taken up by common memory which is used by the compiler for internal use.

Floating point type

This selects the format used for floating point doubles. The default format is the 24-bit truncated IEEE 754 format, but you may choose to use the 32-bit IEEE 754 format.

Output file type

The default output file type is Bytecraft COD. The other choices are: Motorola S-Record HEX, Intel HEX, Binary Image, UBROF, Tektronix HEX, American Automation symbolic HEX and Intel OMF-51. This option will also allow you to specify that you want to create a library. A library can only be created from a project file.

ROM addresses

This menu is highlighted when a high-end PIC device is selected from the Select processor menu. This dialog allows you to specify the address ranges covered by external ROM devices. The addresses are used to store code and const data. (This option is disabled under PIC LITE)

Map and symbol file options

This dialog box allows you to set various options pertaining to debug information, the map file and the symbol file.

Source level debug info

This menu item is used to enable or disable source level debug information in the current symbol file. If you are using MPLAB, you should enable this option

Sort map by address

By default, the symbol table in the in the link map will be sorted by name. This option will cause it to be sorted numerically, based on the value of the symbol.

Suppress local symbols

Prevents the inclusion of all local symbols in the symbol file. Even if this option is not active, the linker will filter irrelevant compiler generated symbols from the symbol file.

Fake local symbols

This modifies the debug information produced to allow MPLAB to examine most local variables. It also adjusts source-level single stepping information to be that required by MPLAB.

MPLAB-ICD support

This button automatically adjusts the linker settings so that the output code is suitable for the MPLAB In-Circuit Debugger. This menu item is only highlighted if the selected processor has ICD capability.

Save dependency information

With this checked (which is the default), dependency information is saved in the project file. This means that restarting the HTL is much faster for a large project.

4.4.5 Compile menu

The Compile menu, [26] shown in Figure 4.4, contains the various forms of the compile command along with several machine independent compiler configuration options.

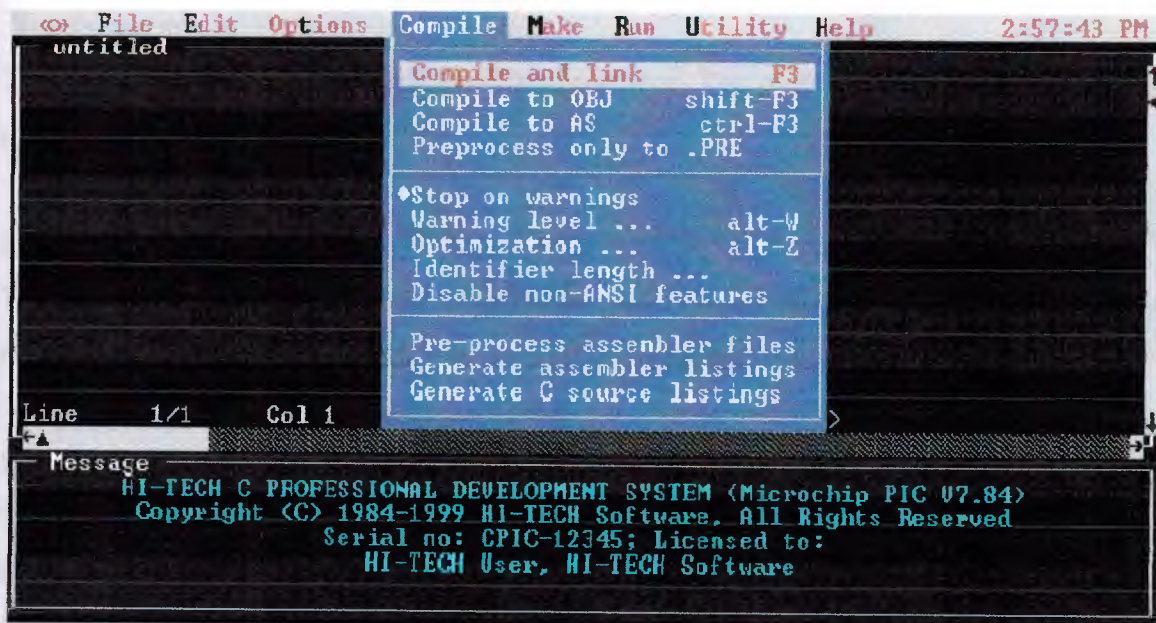


Figure 4.4 HTLPIC Compile Menu

Compile and link ... F3

This command will compile a single source file and then invoke the linker and other utilities to produce an executable file. If the source file is an .as file, it will be passed directly to the assembler. The output file will have the same base name as the source file, but a different extension. For example program.c would be compiled to program.cod.

Compile to .OBJ... shift-F3

Compiles a single source file to a .obj file only. The linker and objtohex are not invoked. The .as files will be passed directly to the assembler. The object file produced will have the same base name as the source file and the extension .obj.

Compile to .AS ... ctrl-F3

This menu item compiles a single source file to assembly language, producing an assembler file with the same base name as the source file and the extension .as. This option is handy if you want to examine or modify the code generated by the compiler. If the current source file is an .as file, nothing will happen.

Preprocess only to .PRE

This runs the source file through the C preprocessor. The output of this action is a .pre file of the same name as the source file.

Stop on Warnings

This toggle determines whether compilation will be halted when non-fatal errors are detected. A mark appears against this item when it is active.

Warning level ... alt-W

This command calls up a dialog which allows you set the compiler warning level, i.e. it determines how selective the compiler is about legal but dubious code. The range of currently implemented warning levels is -9 to 9, where lower warning levels are stricter. At level 9 all warnings (but not errors) are

suppressed. Level 1 suppresses the `func()` declared implicit `int` message which is common when compiling UNIX-derived code. Level 3 is suggested for compiling code written with less strict (and K&R) compilers. Level 0 is the default. This command is equivalent to the `W` option of the `PICL` command.

Optimisation ... Alt-Z

Selecting this item will open a dialog allowing you to select different kinds and levels of optimisation. The default is no optimization. Selections made in this dialog will be saved in the project file if one is being used.

Identifier length

By default C identifiers are considered significant only to 31 characters. This command will allow setting the number of significant characters to be used, between 31 and 255.

Disable non-ANSI features

This option is used to enable strict ANSI conformance of all special keywords. `HI-TECH C` supports the special keywords such as `persistent` and `interrupt` which are used to prevent variables being cleared on startup, and to handle interrupts using C code. If this option is used, these keywords, for example, are changed to `__persistent` and `__interrupt` respectively so as to strictly conform to the ANSI standard. This is the same as the `STRICT` option when compiling from the command line.

Pre-process assembler files

Selecting this item will make `HTLPIC` pass assembler files through the pre-processor before assembling. This makes it possible to use C pre-processor macros and conditionals in assembler files. A mark appears before the item when it is selected.

Generate assembler listing

This menu option tells the assembler to generate a listing file for each C or assembler source file which is compiled. The name of the list file is determined from the name of the symbol file, for example `program.c` will produce a listing file called `program.lst`.

Generate C source listing

Selecting this option will cause a C source listing for each C file compiled. The listing file will be named in the same way as an assembler file (described above) but will contain the C source code with line numbers, and with tabs expanded. The tab expansion setting is derived from the editor tab stop setting.

4.4.6 Make menu

The Make menu (Figure 4.5)[26] contains all of the commands required to use the HTLPIC project facility. The project facility allows creation of complex multiple-source file applications with ease, as well as a high degree of control of some internal compiler functions and utilities.

To use the project facility, it is necessary to follow several steps.

- Create a new project file using the New project ... command. After selecting the project file name, HTLPIC will present several dialogs to allow you to set up options including the processor type and optimisation level.
- Enter the list of source file names using the Source file list ... command.
- Set up any special libraries, pre-defined pre-processor symbols, object files or linker options using the other items in the Make menu.
- Save the project file using the Save project command.
- Compile your project using the Make or Re-Make command.

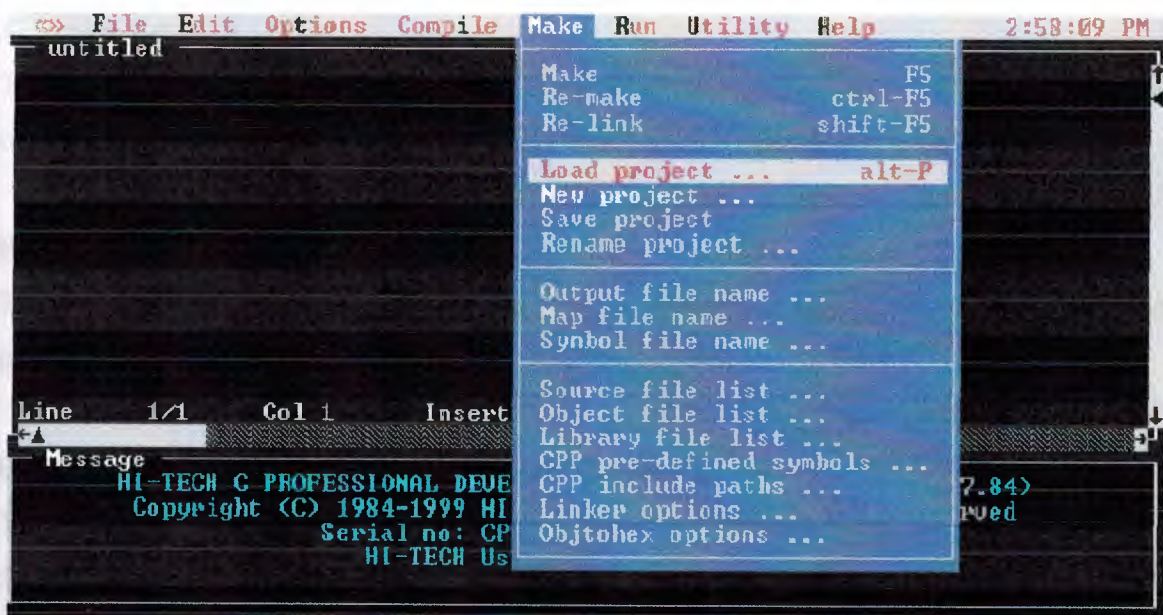


Figure 4.5 HTLPIC Make Menu

Make... F5

The Make command re-compiles the current project. When Make is selected, HTLPIC re-compiles any source files which have been modified since the last Make command was issued. HTLPIC determines whether a source file should be recompiled by testing the modification time and date on the source file and corresponding object file. If the modification time and date on the source file is more recent than that of the object file, it will be re-compiled. If all object (.obj) files are current but the output file cannot be found, HTLPIC will re-link using the object files already present. If all object files are current and the output file is present and up to date, HTLPIC will print a message in the message window indicating that nothing was done.

HTLPIC will also automatically check dependencies, i.e. it will scan source files to determine what files are included, and will include those files in the test to determine if a file needs to be recompiled. In other words, if you modify a header file, any source files including that header file will be recompiled.

If you forget to use the source file list to select the files to be included, HTLPIC will produce a dialog warning that no files have been selected. You will then have to select the DONE button or press escape. This takes you back to the editor window.

Re-make... ctrl-F5

The Re-make command forces recompilation of all source files in the current project. This command is equivalent to deleting all object files and then selecting Make.

Re-link... shift-F5

The Re-link command relinks the current project. Any object files which are missing or not up to date will be regenerated.

Load project ... Alt-P

This command loads a pre-defined project file. You are presented with a file selection dialog allowing a .prj file to be selected and loaded. If this command is selected when the current project has been modified but not saved, you will be given a chance to save the project or abort the Load project command. After loading a project file, the message window title will be changed to display the project file name.

New project

This command allows the user to start a new project. All current project information is cleared and all items in the Make menu are enabled. The user will be given a chance to save any current project and will then be prompted for the new project's name.

Following entry of the new name HTLPIC will present several dialogs to allow you to configure the project. These dialogs will allow you to select: processor type; float type; output file type; optimization settings; and map and symbol file options. You will be asked to enter source file names via the Source file list.

Save project

This item saves the current project to a file.

Rename project

This will allow you to specify a new name for the project. The next time the project is saved it will be saved to the new file name. The existing project file will not be affected if it has already been saved.

Output file name

This command allows the user to select the name of the compiler output file. This name is automatically setup when a project is created. For example if a project called prog1 is created and a COD file is being generated, the output file name will be automatically set to prog1.cod.

Map file name

This command allows the user to enable generation of a symbol map for the current project, and specify the name of the map. The default name of the map file is generated from the project name, e.g. prog1.map.

Symbol file name

This command allows you to select generation of a symbol file, and specification of the symbol file name. The default name of the symbol file will be generated from the project name, e.g. prog1.sym. The symbol file produced is suitable for use with MPLAB.

Source file list

This option displays a dialog which allows a list of source files to be edited. The source files for the project should be entered into the list, one per line. When finished, the source file list can be exited by pressing escape, clicking the mouse on the DONE button, or clicking the mouse in the menu bar.

The source file list can contain any mix of C and assembly language source files. C source files should have the suffix .c and assembly language files the suffix .as, so that HTLPIC can determine where the files should be passed.

Object file list

This option allows any extra .obj files to be added to the project. Only enter one .obj file per line. Operation of this dialog is the same as the source file list dialog.

This list will normally only contain one object file: the run-time start off module for the current processor. For example, if a project is generating code for the PIC16C84, by default this list will contain a runtime startoff module called picrt400.obj. Object files

corresponding to files in the source file list SHOULD NOT be entered here as .obj files generated from source files are automatically used. This list should only be used for extra .obj files for which no source code is available, such as run-time startoff code or utility functions brought in from an outside source.

If a large number of .obj files need to be linked in, they should be condensed into a single .lib file using the *LIBR* utility and then accessed using the *Library file list ...* command.

Library file list

This command allows any extra object code libraries to be searched when the project is linked. This list normally only contains the default library for the processor being used. For example, if the current project is for a PIC16C84, this list will contain the library pic400-c.lib. If an extra library, brought in from an external source, is required, it should be entered here.

It is a good practice to enter any non-standard libraries before the standard C libraries, in case they reference extra standard library routines. The normal order of libraries should be user libraries then the standard C library. Sometimes it is necessary to scan a user library more than once. In this case you should enter the name of the library more than once.

CPP pre-defined symbols

This command allows any special pre-defined symbols to be defined. Each line in this list is equivalent to a D option to the command line compiler PCC. For example, if a CPP macro called DEBUG with value 1, needs to be defined, add the line DEBUG=1 to this list. Some standard symbols will be predefined in this list, these should not be deleted as some of the standard header files rely on their presence.

CPP include paths

This option allows extra directories to be searched by the C pre-processor when looking for header files. When a header file enclosed in angle brackets is included, for example <stdio.h>, the compiler will search each directory in this list until it finds the file.

Linker options

This command allows the options passed to the linker by HTLPIC to be modified. The default contents of the linker command line are generated by the compiler from information selected in the Options menu.

Objtohex options

This command allows the options passed to objtohex by HTLPIC to be modified. Normally you will not need to change these options as the generation of output files can be chosen in the Options menu. However, if you want to generate one of the unusual output formats which objtohex can produce, like COFF files, you will need to change the options using this command.

4.4.7 Run menu

The Run menu [26] shown in Figure 4.6, contains options allowing MS-DOS commands and user programs to be executed.

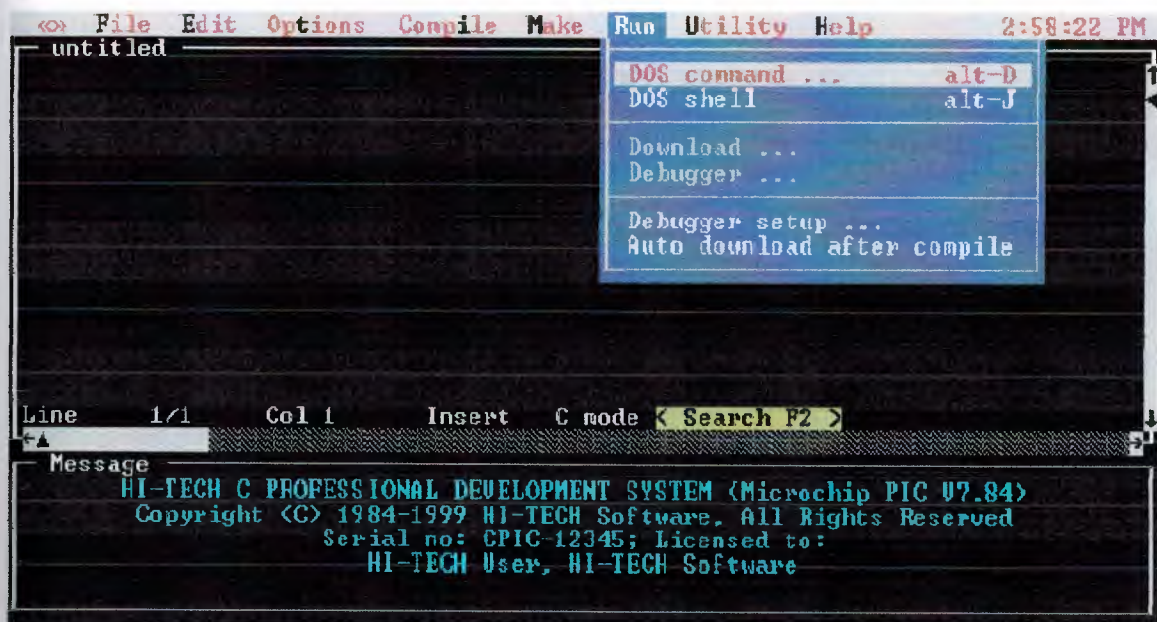


Figure 4.6 HTLPIC Run Menu

DOS command ... Alt-d

This option allows an MS-DOS command to be executed exactly like it had been entered at the command.com prompt. This command could be an internal MS-DOS command like dir, or the name of a program to be executed. If you want to escape to the MS-DOS command processor, use the DOS Shell command below.

DOS Shell... Alt-J

This item will invoke an MS-DOS command.com shell, i.e. you will be immediately presented with a MS-DOS prompt, unlike the DOS command item which prompts for a command. To return to HTLPIC, type exit at the MS-DOS prompt.

Other Options

All other options in this menu are for future enhancements to the compiler.

4.4.8 Utility menu

The Utility menu (Figure 4.7) [26] contains any useful utilities which have been included in HTLPIC.

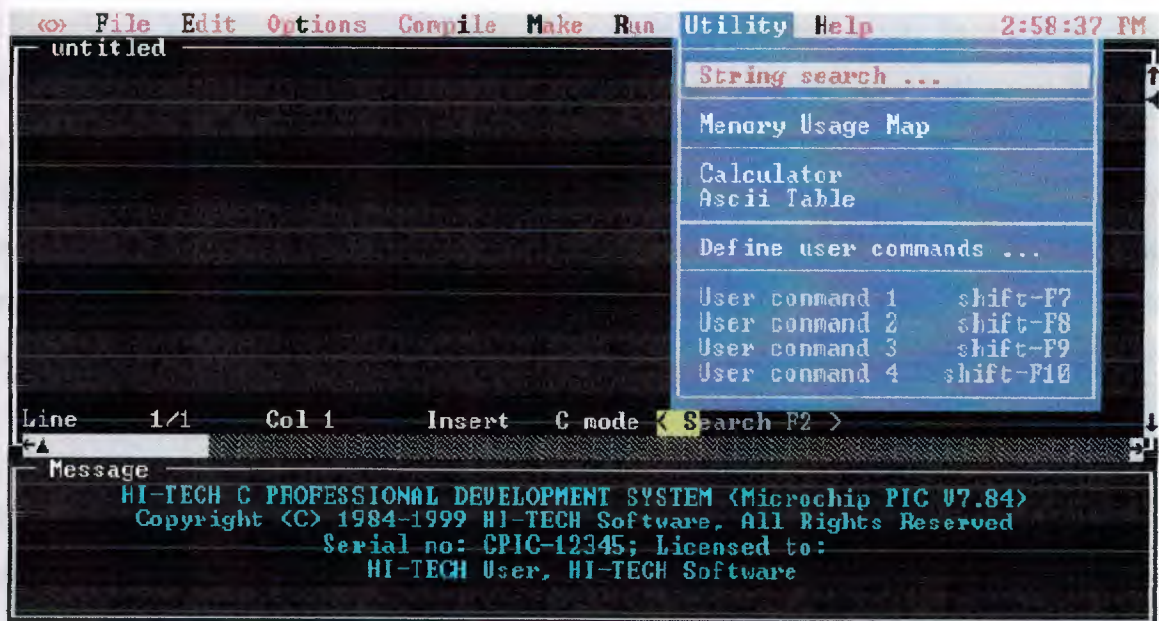


Figure 4.7 HTLPIC Utility Menu

String search

This option allows you to conduct a string search in a list of files. The option produces a dialog which enables you to type in the string you are seeking and then select a list of files to search. You can also select case sensitivity. It is possible to limit the search to a source file list or just the current project.

Memory usage map

This option displays a window which contains a detailed memory usage map of the last program which was compiled. The memory usage map window may be closed by clicking the mouse on the close box in the top left corner of the frame, or by pressing esc while the memory map is the front most window.

Calculator

This command selects the HI-TECH Software programmer's calculator. This is a multi-display integer calculator capable of performing calculations in bases 2 (binary), 8 (octal), 10 (decimal) and 16 (hexadecimal). The results of each calculation are displayed in all four bases simultaneously.

Operation is just like a "real" calculator - just press the buttons! If you have a mouse you can click on the buttons on screen, or just use the keyboard. The large buttons to the right of the display allow you to select which radix is used for numeric entry.

The calculator window can be moved at will, and thus can be left on screen while the editor is in use. The calculator window may be closed by clicking the OFF button in the bottom right corner, by clicking the close box in the top left corner of the frame, or by pressing esc while the calculator is the front most window.

Ascii Table

This option selects a window which contains an ASCII look-up table. The ASCII table window contains four buttons which allow you to close the window or select display of the table in octal, decimal or hexadecimal.

The ASCII table window may be closed by clicking the CLOSE button in the bottom left corner, by clicking the close box in the top left corner of the frame, or by pressing esc while the ASCII table is the front most window.

Define user commands

In the Utility menu are four user-definable commands. This item will invoke a dialog box which will allow you to define those commands. By default the commands are dimmed (not selectable) but will be enabled when a command is defined. Each command is in the form of a DOS command, with macro substitutions available. The macros available are listed in Table 4.1 .

Table 4.1 Macros usable in user commands

Macro name	Meaning
\$(LIB)	Expands to the name of the system library file directory; eg c:\ht-pic\lib\
\$(CWD)	The current working directory
\$(INC)	The name of the system include directory
\$(EDIT)	The name of the file currently loaded into the editor. If the current file has been modified, this will be replaced by the name of the auto saved temporary file. On return this will be reloaded if it has changed.
\$(OUTFILE)	The name of the current output file, i.e. the executable file.
\$(PROJ)	The base name of the current project, eg if the current project file is audio.prj, this macro will expand to audio with no dot or file type.

Each user-defined command has a hot key associated. They are shift F7 through shift F10, for commands 1 to 4. When a user command is executed, the current edit file, if changed, will be saved to a temporary file, and the \$(EDIT) macro will reflect the saved temp file name, rather than the original name. On return, if the temp file has changed it will be reloaded into the editor. This allows an external editor to be readily integrated into HTLPIC.

4.4.9 Help menu

The Help menu (Figure 4.8) contains items allowing you to obtain help about any topics listed. On startup, HTLPIC searches the current directory and the help directory for tbl files, which are added to the Help menu. The path of the help directory can be specified by

the environment variable HT_PICL_HLP. If this is not set, it will be derived from the full path name used when HTLPIC was invoked. If the help directory cannot be located, none of the standard help entries will be available.

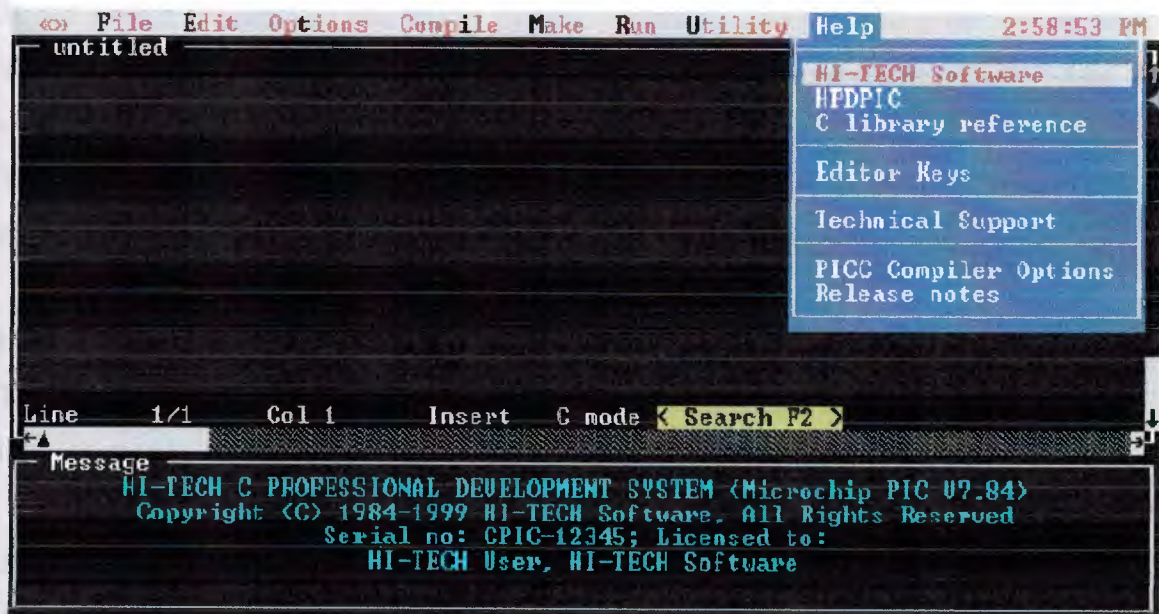


Figure 4.8 HTLPIC Help Menu

HI-TECH Software

This includes information on contacting HI-TECH Software and the licence agreement.

HTLPIC

This option produces a window showing all the topics for which help is available. Topics include Chip types, Compiler optimizations, Editor Searching, Floating-point sizes, String search and User defined commands.

C Library Reference

This command selects an on-line manual for the standard ANSI C library. You will be presented with a window containing the index for the manual. Topics can be selected by double clicking the mouse on them, or by moving the cursor with the arrow keys and pressing return.

Once a topic has been selected, the contents of the window will change to an entry for that topic in a separate window. You can move around within the reference using the keypad cursor keys and the index can be re-entered using the INDEX button at the bottom of the window.

If you have a mouse, you can follow hypertext links by double clicking the mouse on any word. For example, if you are in the tan entry and double click on the reference to asin, you will be taken to the entry for asin.

This window can be re-sized and moved at will, and thus can be left on screen while the editor is in use.

Editor Keys

This option displays a list editor commands and the corresponding keys used to activate that command.

Technical Support

This option displays a list of dealers and their phone numbers for you to use should you require technical support.

PICL Compiler Options

This option displays a window showing all the PICL compiler options. They are displayed in a table showing the option and its meaning. You can scroll through the table using the normal scroll keys or the mouse.

Release notes

This option displays the release notes for your program. You can scroll through the window using the normal scrolling keys or the mouse.



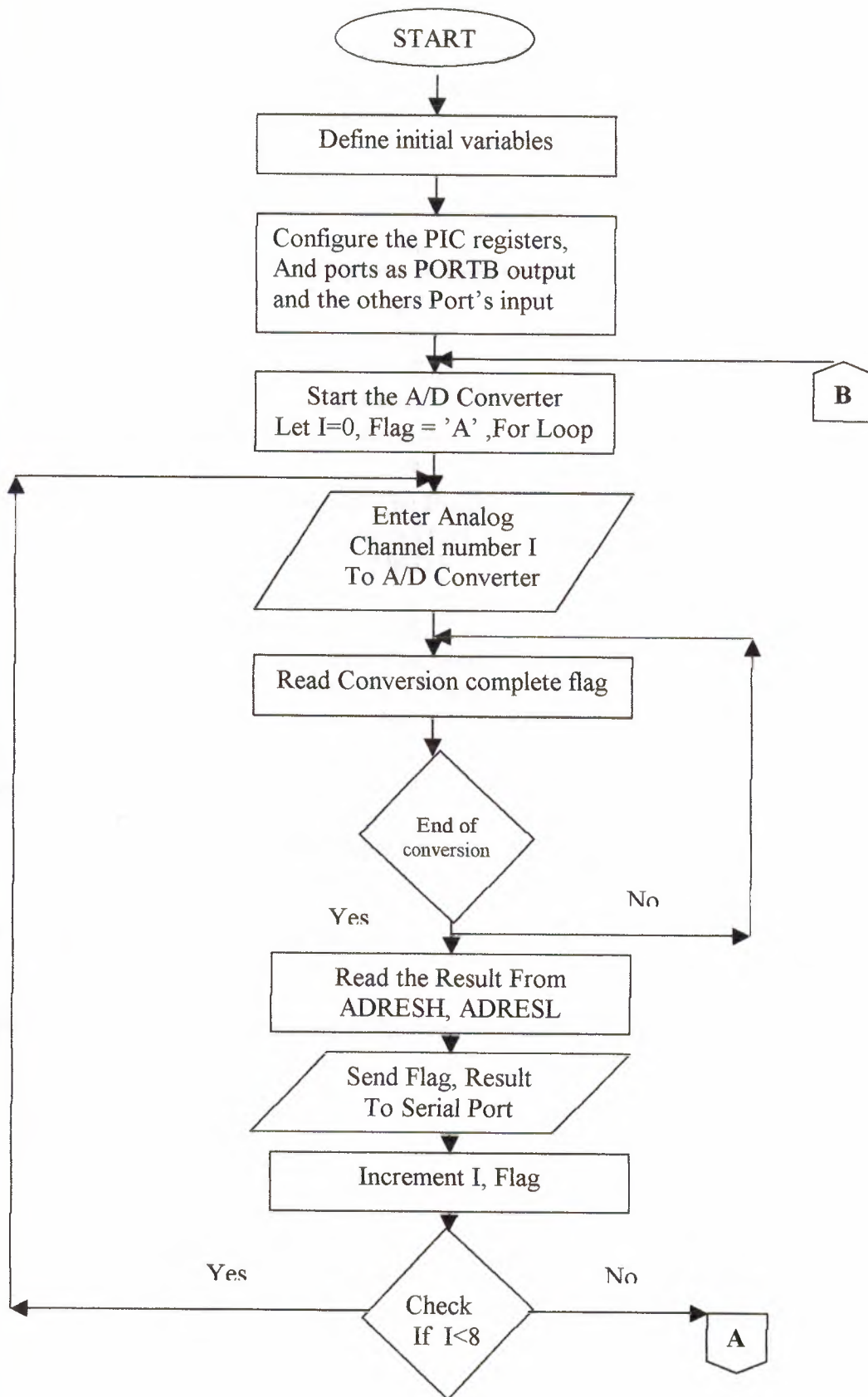
4.5 Creating and Compiling the Data Logger Program

This section describes the microcontroller C program developed by the author. The complete source code listing of this program is given in Appendix A. The program source code is named PIC_PROGRAM.C and the details of the program are described in the following sub-sections in detail.

4.5.1 Flow Chart Of PIC_PROGRAM.C

The flow chart of the microcontroller program is shown in Figure 4.9. The operation of the program is basically as follows:

At the beginning of the program various variables used in the program are declared and the I/O ports are configured. A loop is then formed. Inside this loop the A/D converter is started and analog data is read from the sensors connected to the data logger. The microcontroller incorporates 8 A/D channels and the analog data is read from all of the channels. After reading the analog data, the digital data is read from PORT C and PORT D of the microcontroller. At the end of the loop the analog and the digital data are sent to a PC using a serial communications line.



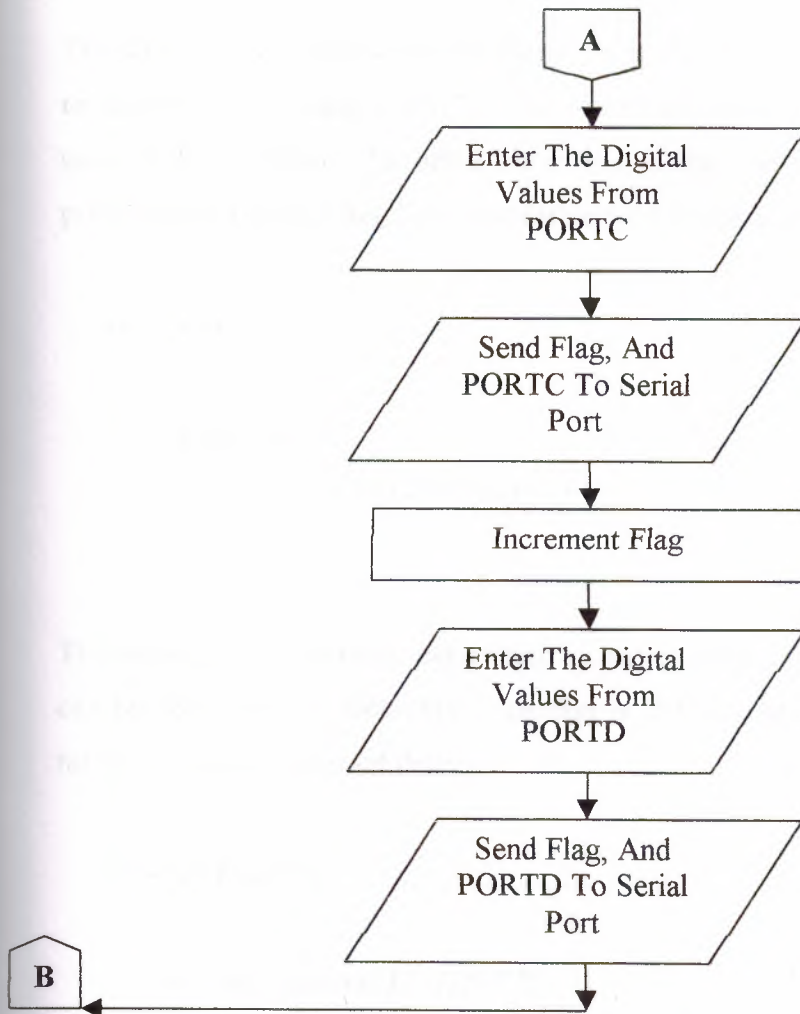


Figure 4.9 Flow Chart of PIC_PROGRAM.C

4.5.2 PIC_PROGRAM Discussion

The following header files are used at the beginning of the program:

```
#include <pic.h>
#include <delay.c>
#include <stdio.h>
#include <serial.c>
```

The source code of these header files exist in the PICC compiler library. Pic.h file is used to select the processor 16F877. The second file delay.c is included so that delays can be used in the program. The third file stdio.h is the standard I/O library which contains the printf() and other I/O function responsible for I/O operations.

```
void wait_1_sec()
{
    unsigned int i;
    for(i=0; i<4; i++) DelayMs(250);
}
```

The above is a function to make delay of approximately 1 second. The maximum delay that can be set using the DelayMs() function is 250 ms and thus this function is called four times to create a 1 second delay.

```
void send_data()
{
    unsigned char rs232_1[]=" "; //1
    unsigned char rs232_2[]="000"; //2
    sprintf(rs232_1,"%c",flag); //3
    printf("%s",rs232_1); //4
    if(temp<100)temp = (temp+800); //5
    sprintf(rs232_2,"%d",temp); //6
    printf("%s",rs232_2); //7
    wait_1_sec(); //8
}
```

The above function send_data () is used to transmit data from the PIC to the serial port. Lines 1 and 2 declare two arrays of type char, the size of them are 1,3 respectively. Line 3 is to put the flag value in the 1st array. Line 4 is to transmit the first array through the serial transmitter. Line 5 is a conditional statement and this line to sure that, the value of the

reading sensor will fill the three bytes of the `rs232_2[]` array. For example, if the reading value is 37 it will occupy 2 positions so I add to it 800 to give 837(3 position). Lines 6 and 7 are the same as lines 3 and 4. Line 8 calls to function `wait_1_sec()` in order to create a 1 second delay.

The `main()` function of the program is explained below:

```
unsigned int i;  
const int lsb = 5000/1024;  
float mV;  
unsigned int thigh;
```

These lines define the various variables and constants used in the program.

```
TRISB = 0;  
TRISA = 1;  
TRISE = 0x03;  
TRISD = 1;  
TRISC = 1;
```

TRIS is a special PIC microcontroller register used to set the direction of an I/O port. If a bit of a TRIS register is cleared then the corresponding port bit is set as an output port. Similarly, if a bit of a TRIS register is set to 1, the corresponding port bit is set as an input port. In the above code, PORT B lines are defined as outputs, PORT A, PORT C and PORT D port lines are defined as inputs, and bits 0 and 1 of PORT E are set as input ports, the other PORT E bits set as outputs.

```
ADCON1 = 0x80;
```

The above line configures the A/D converter register `ADCON1` with binary value (1000 0000) which sets all of PORT A bits as analog inputs with a reference voltage of, $V_{ref} = VDD$.


```
ADCON0 = 0x41;
```

```
DelayMs(250);
```

The above line configures the A/D register ADCON0 with binary value (0100 0001) which sets the A/D converter channel 0 ready to receive analog data.

An infinite continuous loop is then formed by the for statement:

```
for(;;)
```

```
flag = 'A';
```

```
ADCON0 = 0x45;
```

The following C code was used to read the analog data from each channel and then send the data in serial format using the send_data() function. The A/D channel selection was performed by adding an offset to the ADCON0 register inside the for loop. The A/D channels are each 10 bits wide. Register ADRESH holds the upper 2 bits of the converted data. Similarly, register ADRESL holds the lower 8 bits of the converted data.

```
for(i=0;i<8;i++) //1
{
    while((ADCON0 & 4) != 0); //2
    thigh = ADRESH; //3
    thigh = 256*thigh + ADRESL; //4
    mV = thigh*lsb; //5
    temp = (int) (mV/10.0); //6
    send_data(); //7
    flag++; //8
    if(i<7) ADCON0 = 0x45+(i+1)*8; //9
}
```

The digital inputs were read by simply assigning the port values to program variables. The digital data was also sent over the serial communications channel using the `send_data()` function:

```
temp = PORTD;
send_data();
temp = PORTC;
flag++;
send_data();
```

4.5.3 Compiling The Data Logger Program Using PICC Compiler

- 1- Starting HTLPIC: From the Start menu, Programs, HI_TECH Software, PICC lite, HTLPIC.

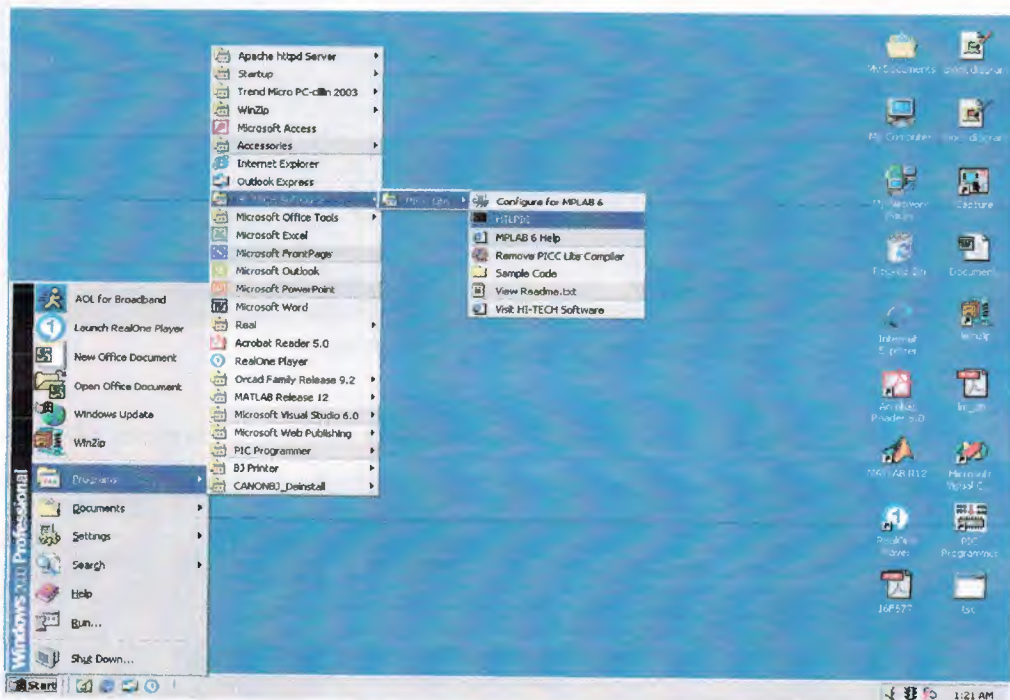


Figure 4.10 Starting HTLPIC

- 2- Open the program :From the main menu, File, Open

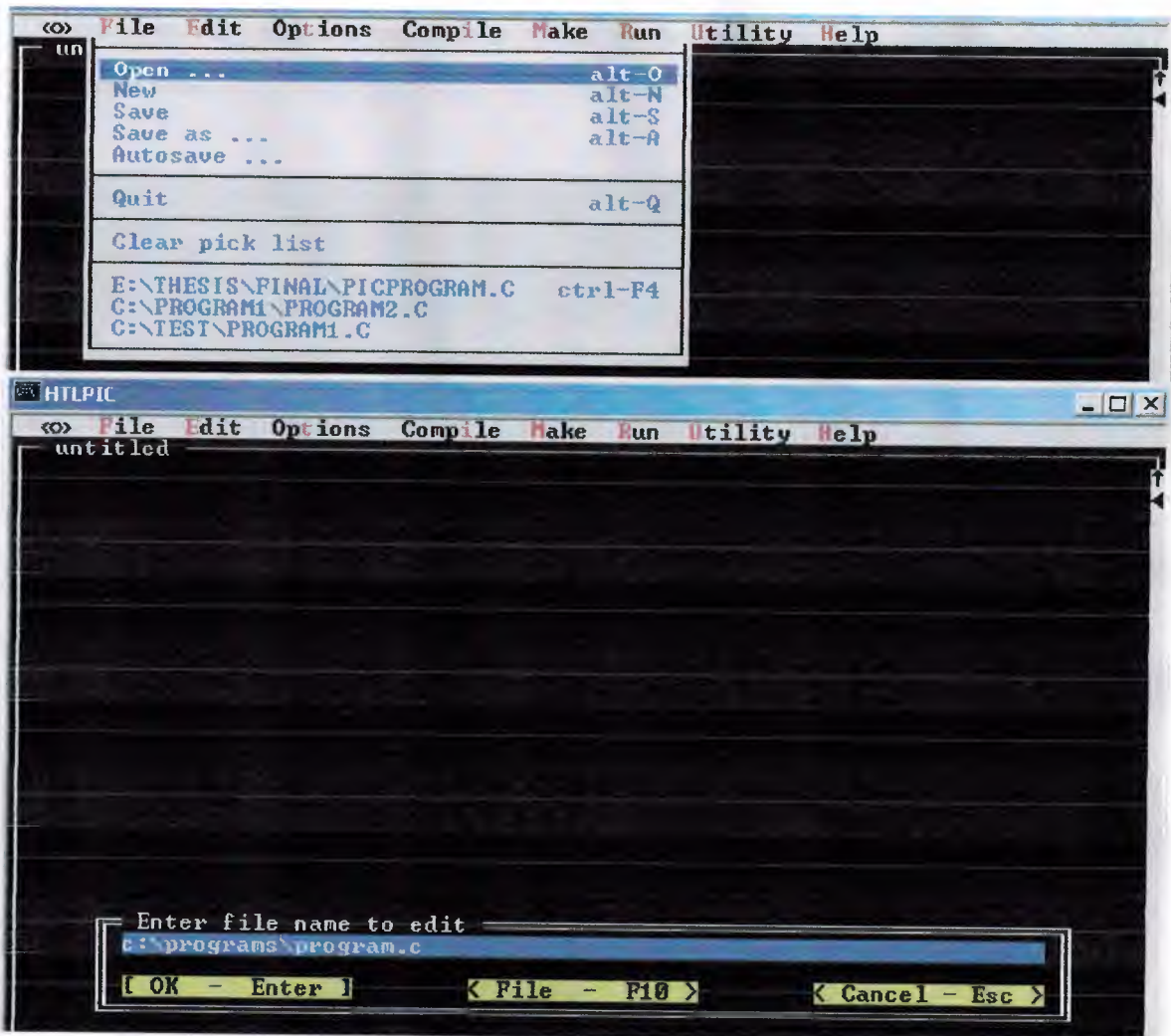


Figure 4.11 Open the program.c

3-Compile the program: From the main menu, Compile, Compile and link

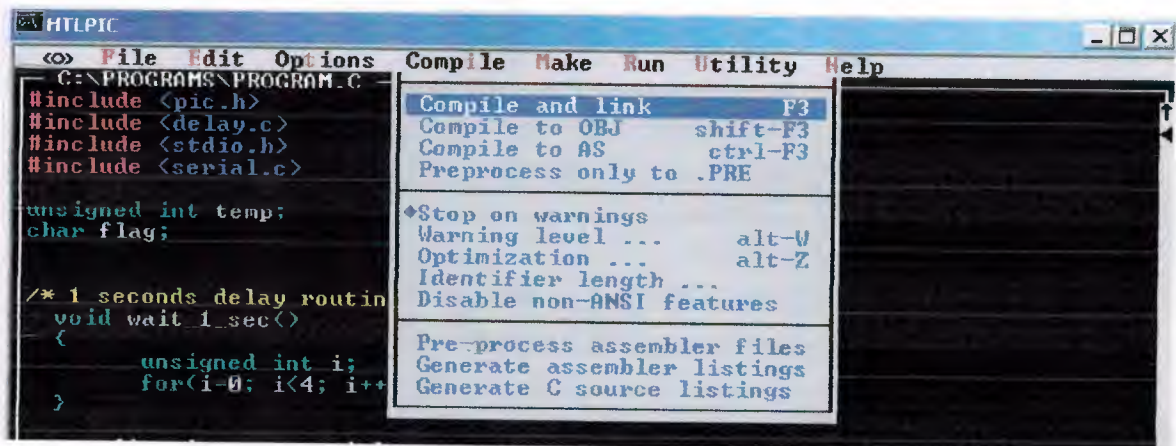


Figure 4.12 Compile the program

The compiler now expects the user to select the processor type. Here, we select 16F877, then press OK



Figure 4.13 Processor Selection

Then the floating point type is asked and we can select either 24 bit or 32 bit floating point operations.

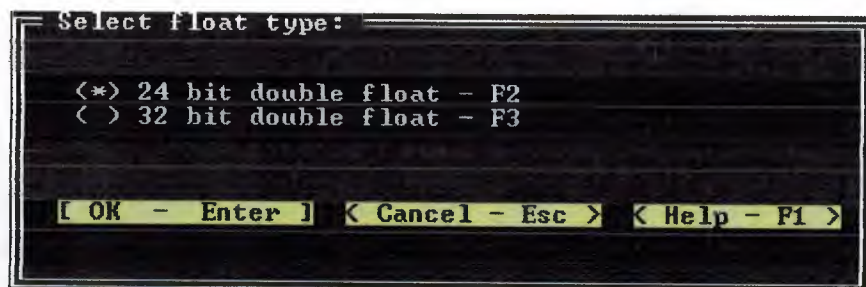


Figure 4.14 Floating Point Type Selection

Then the level of optimization is asked. It is best to select the full optimization so that the code generated is fully optimized.

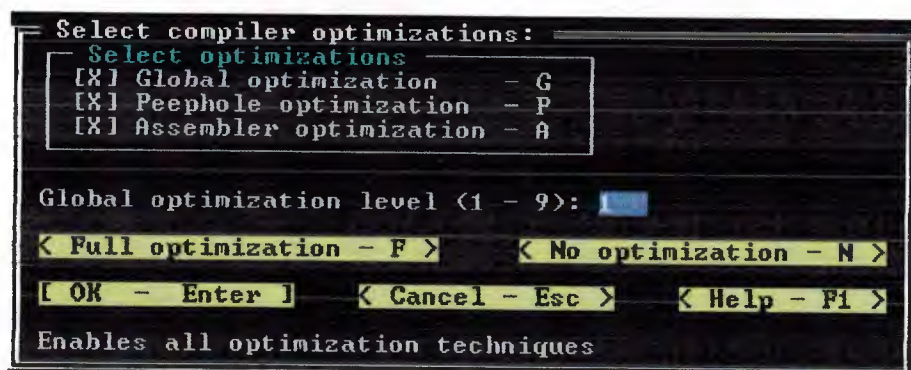


Figure 4.15 Optimization Selection

The output file format is the next choice. This depends upon the type of chip we have and the Intel Hex format is usually selected here

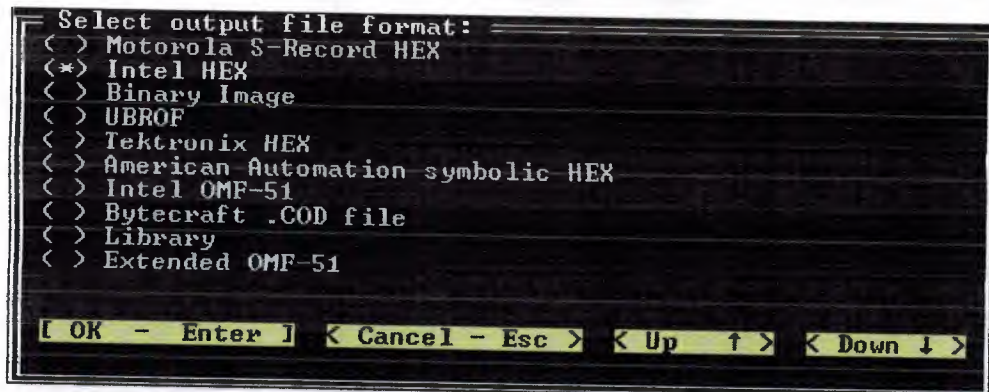


Figure 4.16 Output file format Selection

The final choice is whether or not we need to generate map and symbol files of the compiled code. This step is not normally required and one should just press the OK button

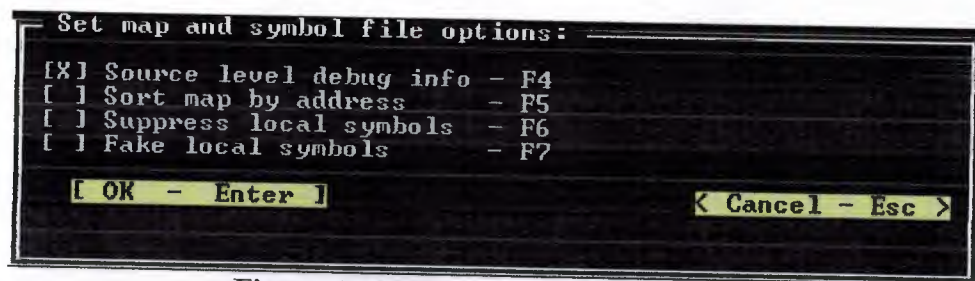


Figure 4.17 Map and symbol file options

Then it will compile the program, if there is any error in the program it will display the errors in the bottom window of the compiler page.



Figure 4.18 Compiling Process

If there are no errors, the compiler will display in the bottom window (message window) the following figure (Fig 4.19). Also, a new file with the extension .HEX will be created in the same directory as the original source file. In this example, the file created will be named as PIC_PROGRAM.HEX.


```
Microchip PIC 08.02
Bank 0 RAM $0020 - $0044 $0025 <37> bytes
Bank 0 RAM $0070 - $007B $000C <12> bytes
$0031 <49> bytes total Bank 0 RAM
```

Figure 4.19 Bottom window(message window)

4.5.4 Download PIC_PROGRAM.HEX To PIC16F877 Chip

- 1- Starting PIC Programmer: From the Start menu, Programs, PIC Programmer, PIC Programmer

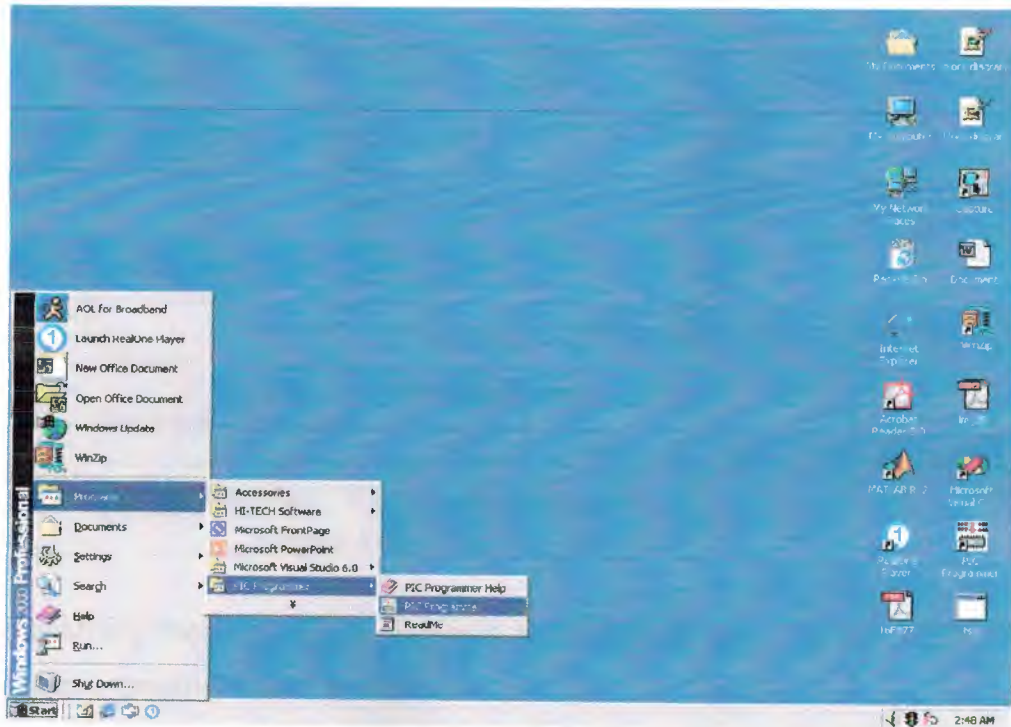


Figure 4.20 Starting PIC Programmer

If the connection between the programmer and serial port is correct then the following message will be displayed on the user terminal:

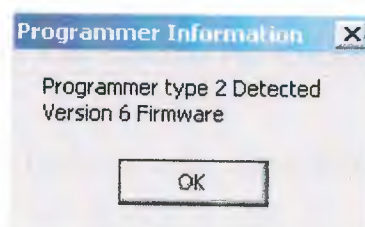


Figure 4.21 Programmer ready

- 2- Open the program: From the main menu, File, Open. Then go to the location where is the PROGRAM.HEX is stored.

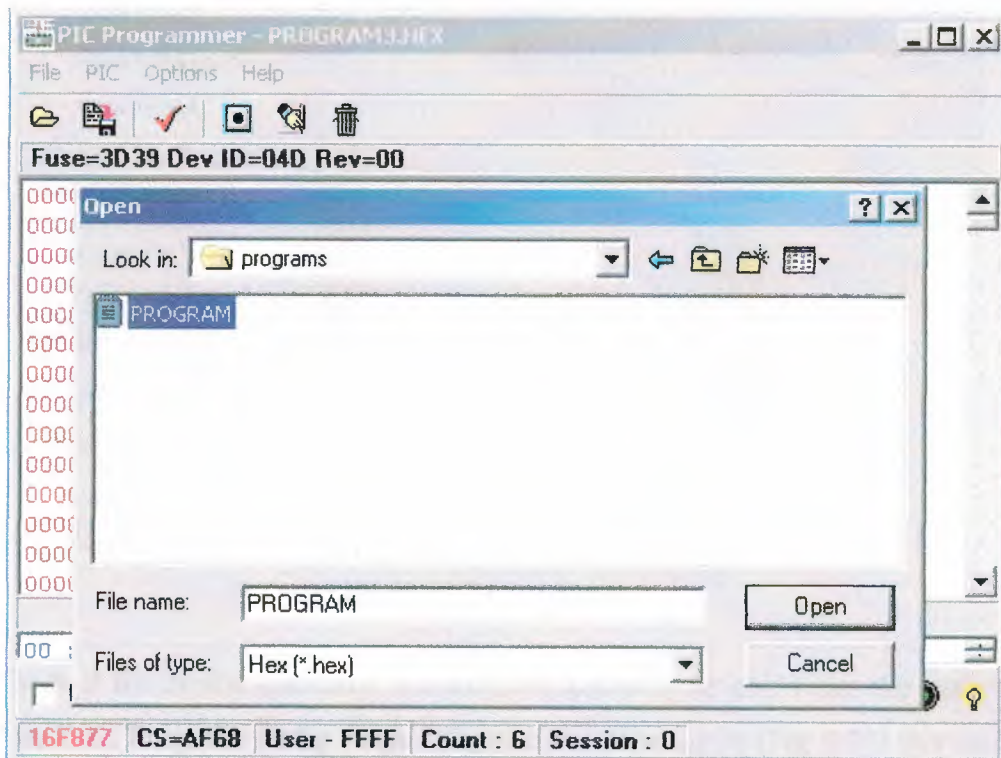


Figure 4.22 Open the program.hex

Then the following message will be displayed to inform the user that the .HEX file has been loaded into the programmer software.

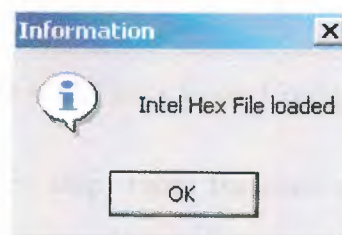


Figure 4.23 Loading PROGRAM.HEX

- 3- Select crystal Oscillator: From the main menu, File, Fuses. And check XT box, and remove the others.

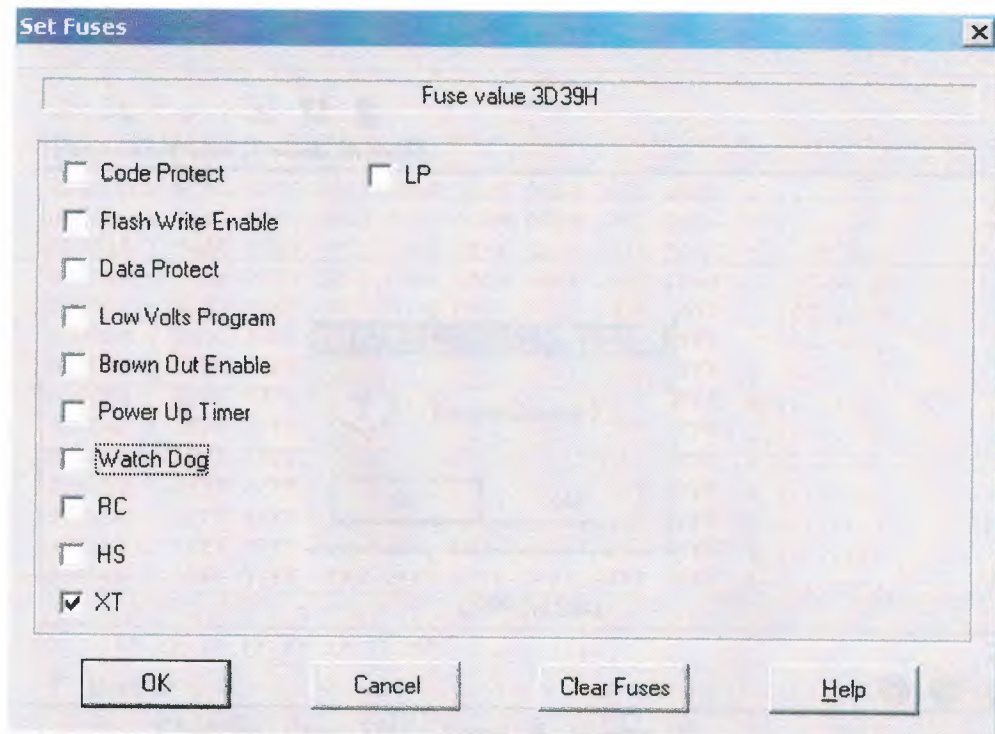


Figure 4.24 Set Fuses

- 4- Check if the crystal oscillator is configured correctly or not: From the main menu, File, PIC, Program config fuses. If appear the message in (Fig 4.25) this mean the oscillator configuration is ok.

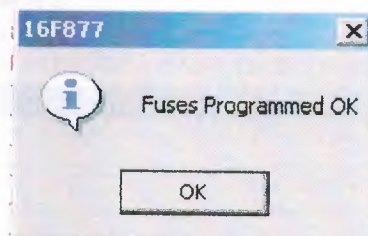


Figure 4.25 Check Fuses Configuration

- 5- Install the program to the chip: From the main menu, File, PIC, Program Entire Device. It will display a message to ask if you sure to install it to the chip (yes).

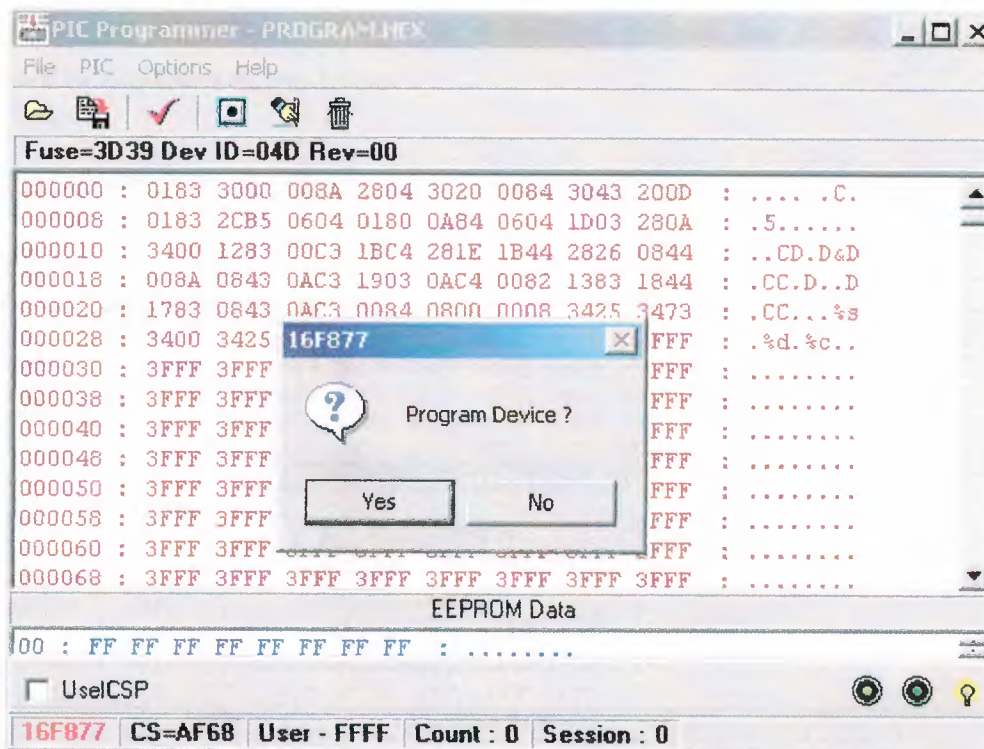


Figure 4.26 Install Program.hex to the 16F877 chip

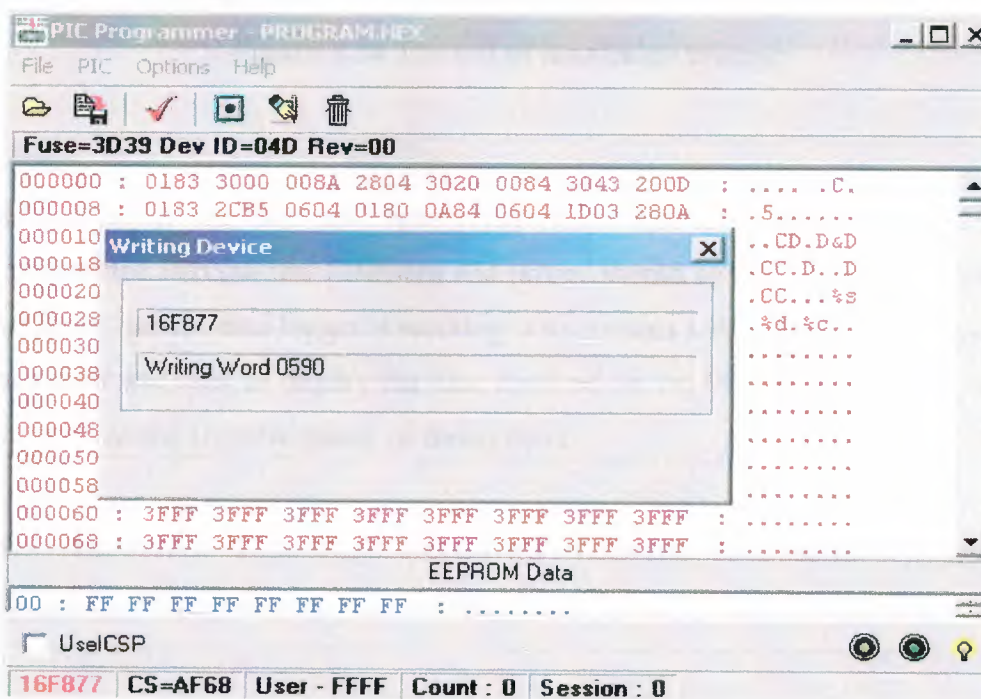


Figure 4.27 Writing on 16F877 chip

When the process of writing to the chip is finished without any errors, the following message will be displayed on the screen:

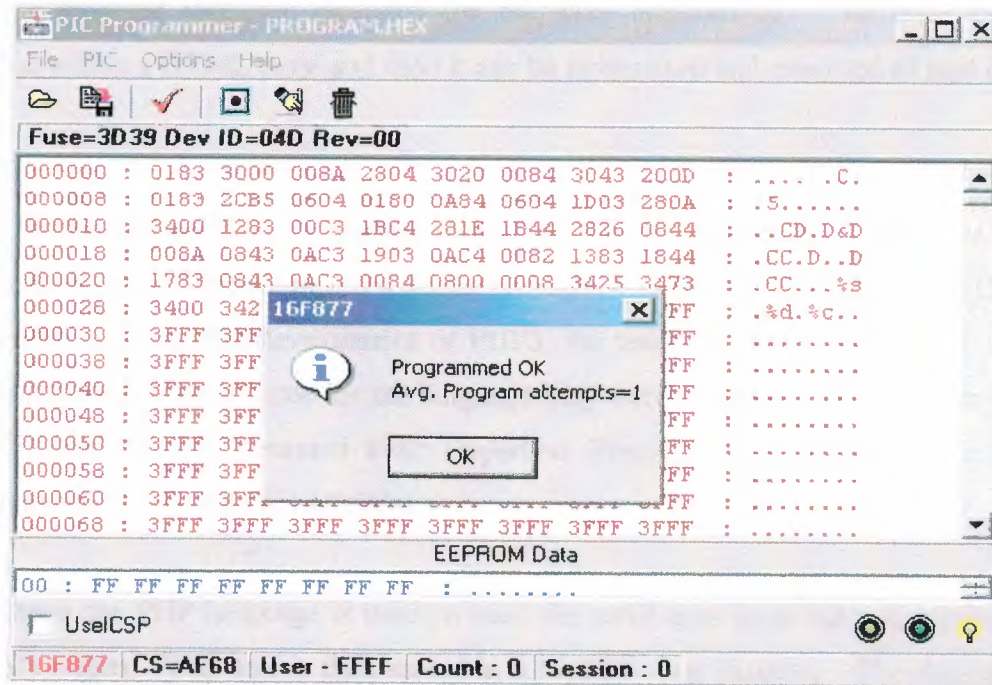


Figure 4.28 The end of installation process

4.6 Testing

The microcontroller chip has now been programmed and is ready for testing. The chip should be inserted into the test hardware and power should be applied. The easiest way to test whether or not the data logger is working is to connect a PC to the serial output port of the data logger and then to display the data received on the PC using a terminal emulation software, such as the Hyperterminal, or SmartTerm.

The author had some initial hardware and software problems. But after these problems were corrected the data logger system was working correctly.

5. PHP SCRIPTING LANGUAGE

5.1 Overview

PHP is a scripting language designed for the Web applications. The code can be embedded within a HTML page and then it can be interpreted and executed as part of the HTML.

PHP/F1, which stood [27] for Personal Home Pages/Form Interpreter, was updated and rewritten by a team of programmers working around the globe and was released as PHP3. At some point during the development of PHP3, the team decided that Personal Home Pages wasn't such a good name for the language they were creating, so a vote was called and PHP was officially renamed PHP Hypertext Preprocessor, which is a recursive acronym like GNU (GNU's not UNIX).

In this thesis the PHP language is used to read the serial data from the microcontroller based data logger. The data is then stored in a MySQL type database. The database is queried using the PHP language. Thus, the user can collect the sensor data automatically from the data logger and this data is then automatically stored in a database on the PC. The user can then display the collected data on his or her terminal.

This chapter gives the basic principles of the PHP scripting language. The actual PHP code developed by the author is described completely in Chapter 7.

5.2 Some of PHP's Strengths

Some of PHP's main competitors are Perl, Microsoft Active Server Pages (ASP), java server pages (JSP), in comparison to these products, PHP has many strengths [29] including the following:

- High performance
- Interfaces to many different database systems.
- Built-in libraries for many common web tasks
- Low cost

- Ease of learning and use
- Portability
- Availability of source code

A more detail discussion of these strengths follows.

Performance

PHP is very efficient. Using a single inexpensive server, you can serve millions of hits per day.

Database integration

PHP has native connections available to many database systems. In addition to MySQL. One can directly connect to PostgreSQL, mSQL, Oracle, dbm, filePro, Hyperwave. Infomix. And Sybase database

Using the Open Database Connectivity Standard (ODBC), you can connect to any database that provides an ODBC driver. This includes Microsoft products, and many others.

Built-in libraries

Because PHP was designed for use on the Web, it has many built-in functions for performing many useful Web-related tasks. One can generate GIF images on-the-fly, connect to other network services, send email, work with cookies, and generate PDF document.

Cost

PHP is free. One can download the latest version at any time from <http://www.PHP.net> for no charge.

Learning PHP

The syntax of PHP is based on other programming languages, primarily C and Perl. If you know C or Perl, or a C-like language such as C++ or Java, you will be productive using PHP almost immediately.

Portability

PHP is available for many different operating systems. One can write PHP code on the free Unix like operating systems such as Linux and FreeBSD, commercial Unix versions such as Solaris and IRIX, or on different versions of Microsoft Windows.

Your code will usually work without modification on a different system running PHP.

Source Code

The source code of PHP is easily available. Unlike commercial, closed-source products, if there is something you want modified or added to the language, you are free to do this.

One does not need to wait for the manufacturer to release patches. Also, one does not need to worry about the manufacturer going out of business or deciding to stop supporting a product.

5.3 Conventions in PHP

There are a few simple conventions that must be followed when programming in PHP. These conventions are [30] standard rules that one needs to apply to the script so that the Web server can correctly interpret the code. If these basic rules are not followed, one may encounter the ubiquitous parse error. They are annoying, but usually easy to fix, especially once the basic rules are known.

PHP Start and End Tags

The most common convention that should be followed is to enclose all of the PHP parts of your script within the PHP start and end tags. These tags are `<?PHP` and `?>` respectively. These tags tell the Web server that anything contained within the tags is PHP code and should be interpreted as such. If the server is set up to use short tags, one can use `<? and ?>`.

Semicolons

The other fundamental rule that should be followed is to place semicolons at the end of each line of PHP code. Note that this doesn't mean after each physical line, but after each command that you write.

```
print("This is a valid line of PHP code.\n");
```

The previous line command equals one valid line of code. The semicolon is placed at the end of the second physical line, indicating the end of the command.

There are some exceptions when you won't need to place a semicolon at the end of the line. These include

- If the line ends in a colon (:) .
- If the line ends in an open ({) or close (}) curly brace .
- If the line ends with the open (<?) or close (?>) PHP tag .

5.4 Embedding PHP into Your HTML Pages

Creating a site [29] that uses PHP is almost the same as creating a site that uses plain old HTML. One can use one of many WYSIWYG HTML editors, or you can use a text editor. The basic principles for coding PHP scripts are the same as for writing HTML; write your HTML and PHP markup, then test it in a browser.

When writing your PHP scripts, you must specify to the Web server which parts of the page the Web server needs to process using PHP and which parts of the page should be processed as normal HTML. When a Web server sends plain HTML information to a browser, it just spits out the file without even really parsing (reading in) the content of the file. The web server must parse PHP (read every line and process it), so you always need to specify when you are breaking out of HTML and going into PHP. Tagging the PHP parts of your page with PHP tags tells the Web server to start reading the files and parse them according to the PHP language. Anything between these tags is evaluated by the PHP part of the Web server.

The most basic example of this is:

1. `<html>`
2. `<head><title>The First Rule</title></head>`
3. `<body>`

```

4. <?PHP
5. print("PHP stands for <b>PHP Hypertext Preprocessor\n</b>");
6. ?<
7. </body>
8. </html>

```

For the most part, this script looks like normal HTML markup, except for lines 4 through 6. If you were to view the script in a browser, you see only the following line, PHP stands for PHP Hypertext Preprocessor. And if you were to look at the source code from within the browser you would simply see:

```

<html>
<head><title>The First Rule</title></head>
<body>
PHP stands for <b>PHP Hypertext Preprocessor</b>
</body>
</html>

```

When the page is requested, the Web server processes the page as normal HTML until it encounters the <?PHP tag (in line 4 of the example). Once the Web server reaches line 4, it interprets the rest of the page as PHP, until it gets to a ?> tag (in line 6); at that point, the Web server returns to processing the page as normal HTML until it either gets to another <?PHP tag or the page ends.

This illustrates the concept of server-side scripting, the PHP has been interpreted and executed on the web server, as distinct from JavaScript and other client-side technologies that are interpreted and executed within a web browser on a user's machine.

The code that we consists of four things:

- PHP tags
- PHP statement
- Whitespace
- Comments

PHP tags

The PHP code with `< ?` and ended with `?>` , these symbols are called PHP tags that tell the Web server where the PHP code starts and finishes. Any text between the tags will be interpreted as PHP. Any text out side these tags will be treated as normal HTML

PHP statement

We tell the PHP interpreter what to do by having PHP statements between our `< ?` and `?>` tags.

```
echo "Mohammad klaib " ;
```

semicolon(;) is used to separate PHP statement

White space

Spacing charcters such as new lines, spaces and tabs known as white space (white space) is ignored in PHP and HTML.

There is no need to have any white space between PHP statement but White space use as an aid to humans-to enhance readability

```
echo "Mohammad" ;
```

```
echo "klaib" ;
```

```
and
```

```
echo "Mohammad" ; echo "klaib" ;
```

are equivalent, but first version is easier to read.

Comments

The PHP interpreter will ignore any text in a comment. Essentialy the PHP parse skips over the comments that are equivalent to whitespace.

PHP supports C, C++, and shell script style comments.

Multiline comments should begin with `/*` and end with `*/` . as in C.

```
/* This is a master thesis */
```

Single line comments `//`, `i = 0;` `// initialization`

5.5 Operators

Operators are [31] symbols that you can use to manipulate values and variables by performing an operation on them. We'll need to use some of these operators to work out the totals.

- Arithmetic Operators
- String Operators
- Assignment Operators
- Combination Assignment Operators
- Pre- and Post-increment and Decrement
- Comparison Operators
- Logical operators

5.5.1 Arithmetic Operators

Arithmetic operators [32] are very straight forward they are just the normal mathematical operators. The arithmetic operators are shown in Table 5. 1.

Table 5.1 PHP's Arithmetic Operators

Operator	Name	Example	Example
+	Addition	$\$a + \b	Sum of \$a and \$b.
-	Subtraction	$\$a - \b	Difference of \$a and \$b.
*	Multiplication	$\$a * \b	Product of \$a and \$b.
/	Division	$\$a / \b	Quotient of \$a and \$b
%	Modulus	$\$a \% \b	Remainder of \$a divided by \$b

With each of these operators, we can store the result of the operation. For example

$\$result = \$a + \$b;$

Addition and subtraction work as you would expect. The result of these operators is to add or subtract, respectively, the values stored in the \$a and \$b variables.

Multiplication and division also work much as you would expect. Note the use of the asterisk as the multiplication operator, rather than the regular multiplication symbol, and the slash as the division operator, rather than the regular division symbol.

The modulus operator returns the remainder of dividing the \$a variable by the \$b variable. Consider this code fragment:

```
$a = 27;  
$b = 10;  
$result = $a % $b;
```

The value stored in the \$result variable is the remainder when we divide 27 by 10: that is, 7.

One should note that arithmetic operators are usually applied to integers or doubles. If you apply them to strings, PHP will try and convert the string to a number.

5.5.2 String Operators

The string concatenation operator to add two strings and to generate and store a result much as you would use the addition operator to add two numbers.

```
$a = "Mohammed";  
$b = "Klaib";  
$result = $a . $b;
```

The \$result variable will now contain the string "Mohammad Klaib".

5.5.3 Assignment Operators

We've already seen =, the basic assignment operator. Always refer to this as the assignment operator, and read it as "is set to." For example

```
$read_number = ' ';
```

This should be read as "\$read_number is set to null".

Combination Assignment Operators

There is a set of combined assignment operators. Each of these is a shorthand way of doing another operation on a variable and assigning the result back to that variable. For example $\$a += 5$;

This is equivalent to writing $\$a = \$a + 5$;

The first statement is faster.

Table 5.2 PHP's Combined Assignment Operators

Operator	Example	Equivalent to
$+=$	$\$a += \b	$\$a = \$a + \$b$
$-=$	$\$a -= \b	$\$a = \$a - \$b$
$*=$	$\$a *= \b	$\$a = \$a * \$b$
$/=$	$\$a /= \b	$\$a = \$a / \$b$
$\% =$	$\$a \% = \b	$\$a = \$a \% \$b$
$.=$	$\$a . = \b	$\$a = \$a . \$b$

5.5.4 Pre- and Post-increment and Decrement

The pre- and post-increment ($++$) and decrement ($--$) operators are similar to the $+=$ and $-=$ operators, but with a couple of twists.

Table 5.3 PHP's Increment/decrement Operators

Example	Name	Effect
$++\$a$	Pre-increment	Increments $\$a$ by one, then returns $\$a$
$\$a++$	Post-increment	Returns $\$a$, then increments $\$a$ by one
$--\$a$	Pre-decrement	Decrements $\$a$ by one, then returns $\$a$
$\$a--$	Post-decrement	Returns $\$a$, then decrements $\$a$ by one

All the increment operators have two effects—they increment and assign a value. Consider the following

```
$a=4 ;
```

```
echo ++$4 ;
```

\$a is incremented to 5 and then the value 5 is returned and printed the value of this whole expression is 5 .

However, if the ++ is after the \$a, we are using the post-increment operator. This has a different effect. Consider the following

```
$a=4;
```

```
echo $a++;
```

That is, first, the value of \$a is returned and printed, and second, it is incremented. The value of this whole expression is 4. This is the value that will be printed. However, the value of \$a after this statement is executed is 5.

5.5.5 Comparison Operators

The comparison [29] operators are used to compare two values. Expressions using these operators return either of the logical values true or false depending on the result of the comparison.

PHP also supports a number of comparison operators. A summary of all comparison operators is shown in Table 5.4.

Table 5.4 PHP's Comparison Operators

Operator	Name	Example	Example
==	Equal	\$a == \$b	True if \$a is equal to \$b..
===	Identical	\$a === \$b	True if \$a is equal to \$b, and they are of the same type. (PHP 4 only)
!=	Not identical	\$a != \$b	True if \$a is not equal to \$b, or they are not of the same type. (PHP 4 only)

!=	Not equal	$\$a \neq \b	True if \$a is not equal to \$b.
<>	Not equal	$\$a <> \b	True if \$a is not equal to \$b
<	Less than	$\$a < \b	True if \$a is strictly less than \$b.
>	Greater than	$\$a > \b	True if \$a is strictly greater than \$b.
<=	Less than or equal to \$a	$\leq \$b$	True if \$a is less than or equal to \$b.
>=	Greater than or equal to \$	$a \geq \$b$	True if \$a is greater than or equal to \$b.

5.5.6 Logical operators

The logical operators [29] are used to combine the results of logical conditions

Table 5.5 PHP's logical Operators

Operator	Name	Example	result
!	Not	$! \$a$	True if \$a is not true.
&&	And	$\$a \&\& \b	True if both \$a and \$b are true.
 	Or	$\$a \b	True if either \$a or \$b is true.
And	And	$\$a \text{ and } \b	True if both \$a and \$b are true.
or	Or	$\$a \text{ or } \b	True if either \$a or \$b is true.

5.6 Control structure

Control structure is the structure within a language that allows us to control the flow of execution through a program or script. One can group them into conditionals structures, and repetition structures, or loops, We will consider the specific implementation of each of these in PHP.

5.6.1 Making Decisions with Conditionals

To sensibly respond to our user's input, our code needs to be able to make decisions.

The constructs that tell our program to make decisions are called conditionals.

- if statement .
- else statement .
- else if statement .
- switch statement .
-

if statement

The if construct is one of the most important features of many languages, PHP included. It allows for conditional execution of code fragments. PHP features an if structure that is similar to that of C:

The following example would display a is bigger than b if \$a is bigger than \$b:

```
if ($a > $b) print "a is bigger than b";
```

else Statements

Often you'd want to execute a statement if a certain condition is met, and a different statement if the condition is not met. This is what else is for. else extends an if statement to execute a statement in case the expression in the if statement evaluates to FALSE. For example, the following code would display a is bigger than b if \$a is bigger than \$b, and a is NOT bigger than b otherwise:

```
if ($a > $b) { print "a is bigger than b"; }  
else { print "a is NOT bigger than b"; }
```

The else statement is only executed if the if expression evaluated to FALSE,

elseif statement

elseif, as its name suggests, is a combination of if and else. Like else, it extends an if statement to execute a different statement in case the original if expression evaluates to

FALSE. However, unlike else, it will execute that alternative expression only if the elseif conditional expression evaluates to TRUE. For example, the following code would display a is bigger than b, a equal to b or a is smaller than b:

```
if ($a > $b) { print "a is bigger than b"; }  
elseif ($a == $b) { print "a is equal to b"; }  
else { print "a is smaller than b";}
```

There may be several elseifs within the same if statement. The first elseif expression (if any) that evaluates to true would be executed. In PHP, you can also write 'else if' (in two words) and the behavior would be identical to the one of 'elseif' (in a single word).

The elseif statement is only executed if the preceding if expression and any preceding elseif expressions evaluated to FALSE, and the current elseif expression evaluated to TRUE.

switch Statements

The switch statement is similar to a series of IF statements on the same expression. In many occasions, you may want to compare the same variable (or expression) with many different values, and execute a different piece of code depending on which value it equals to. This is exactly what the switch statement is for.

```
switch ($i) {  
case 0: print "i equals 0";  
        break;  
case 1: print "i equals 1";  
        break;  
case 2: print "i equals 2";  
        break;  
}
```

It is important to understand how the switch statement is executed in order to avoid mistakes. The switch statement executes line by line (actually, statement by statement). In the beginning, no code is executed. Only when a case statement is found with a value that matches the value of the switch expression does PHP begin to execute the statements. PHP continues to execute the statements until the end of the switch block, or the first time it sees a break statement. If you don't write a break statement at the end of a case's statement list, PHP will go on executing the statements of the following case. For example:

```
switch ($i) {  
    case 0: print "i equals 0";  
    case 1: print "i equals 1";  
    case 2: print "i equals 2";  
}
```

In a switch statement, the condition is evaluated only once and the result is compared to each case statement. In an elseif statement, the condition is evaluated again. If your condition is more complicated than a simple compare and/or is in a tight loop, a switch may be faster.

A special case is the default case. This case matches anything that wasn't matched by the other cases.

5.6.2 Iteration: Repeating Actions

One thing that computers have always been very good at is automating repetitive tasks [29]. If there is something that you need done the same way a number of times, you can use a loop to repeat some parts of your program.

Loop statements tell PHP to execute a statement or block repeatedly.

while Loops

The simplest kind of loop in PHP is the while loop. Like an if statement, it relies on a condition. The difference between a while loop and an if statement is that an if statement

execute, the following block of code once if the condition is true. A while loop executes the block repeatedly for as long as the condition is true.

The while loop is generally used when we don't know how many iterations will be required to make the condition true. If a fixed number of iterations is required, consider using a for loop.

The basic structure of a while loop is

```
while( condition ) expression;
```

The following while loop will display the numbers from 1 to 5.

```
$num ;  
while ($num <= 5)  
{  
echo $num. " <br>" ;  
$num ++ ;  
}
```

At the beginning of each iteration, the condition is tested. If the condition is false, the block will not be executed and the loop will end. The next statement after the loop will then be executed.

for Loops

The way that we used the while loops previously is very common. We set a counter to begin with. Before each iteration, we tested the counter in a condition. At the end of each iteration, we modified the counter.

The basic structure of a for loop is

```
for( expression1; condition; expression2) expression3;
```

We can rewrite the while loop example as a for loop. The PHP code will become

```
for($num =0; $num <= 5; $num ++ )  
echo $num. " <br> ";
```

Both the while version and the for version are functionally identical.

Both these loop types are equivalent-neither is better or worse than the other. In a given situation, you can use whichever you find more intuitive.

do...while

do..while loops are very similar to while loops, except the truth expression is checked at the end of each iteration instead of in the beginning. The main difference from regular while loops is that the first iteration of a do...while loop is guaranteed to run (the truth expression is only checked at the end of the iteration), whereas it's may not necessarily run with a regular while loop (the truth expression is checked at the beginning of each iteration, if it evaluates to FALSE right from the beginning, the loop execution would end immediately).

```
$i = 0;  
do {  
    print $i;  
} while ($i>0);
```

The above loop would run one time exactly, since after the first iteration, when truth expression is checked, it evaluates to FALSE (\$i is not bigger than 0) and the loop execution ends.

Break

break ends execution of the current for, while, or switch structure.

break accepts an optional numeric argument which tells it how many nested enclosing structures are to be broken out of.

```
/* Using the optional argument. */
```

```

$i = 0;
while (++$i) {
    switch ($i) {
        case 5: echo "At 5<br>\n";
                break 1; /* Exit only the switch. */
        case 10: echo "At 10; quitting<br>\n";
                break 2; /* Exit the switch and the while. */
        default: break;
    }
}

```

continue

continue is used within looping structures to skip the rest of the current loop iteration and continue execution at the beginning of the next iteration.

continue accepts an optional numeric argument which tells it how many levels of enclosing loops it should skip to the end of.

```

$i = 0;
while ($i++ < 5) {
    echo "Outer<br>\n";
    while (1) {
        echo " Middle<br>\n";
        while (1) {
            echo " Inner<br>\n";
            continue 3;
        }
        echo "This never gets output.<br>\n";
    }
    echo "Neither does this.<br>\n";
}

```


5.7 Summary

PHP is a server side scripting language designed specifically for the Web. Within an HTML page, it can embedded PHP code that will be executed each time the page is visited. PHP has much strength, and it is based on other programming languages, primarily C, so it is easy to learn, as described in this chapter.

This chapter gave the basic principles of the PHP scripting language. The actual PHP code developed by the author is described completely in Chapter 7.

6. DESIGNING DATABASE FOR DATA LOGGER USING MySQL

6.1. Overview

MySQL (pronounced My-Ess-Que-Ell) is [34] a very fast, robust, relational database management system (RDBMS). A database enables one to efficiently store, search, sort, and retrieve data. The MySQL server controls access to the data to ensure that multiple users can work with it concurrently, to provide fast access to it, and ensure that only authorized users can obtain access. Hence, MySQL is a multi-user, multi-threaded server. It uses SQL (Structured Query Language). The standard database query language worldwide. MySQL has been publicly available since 1996, but has development history going back to 1979. It has now won the Linux Journal Readers' Choice Award three years running.

SQL stands [35] for Structured Query Language. It's the most standard language for accessing relational database management system (RDBMS). SQL is used to store and retrieve data to data and from database. It is used in database systems such as MySQL, oracle, and Microsoft SQL Server among others.

Data Definition Languages (DDL), used for defining databases, and Data Manipulation Languages (DML), used for querying databases, SQL covers both of these bases.

The DDL is used when you're initially setting up a database, You use (DML) of SQL far more frequently because these are the parts that we use to store and retrieve real data in a database.

The data generated by the data logger is sent to the PC over the RS232 serial communications link. A PHP script program running on the PC captures this data and stores it in a MySQL type database. The data can then be analyzed and sent to the user terminal whenever required.

This chapter described the details of the MySQL database used in the thesis to store the captured data.

6.2 Some of MySQL Strengths

Some of MySQL's main competitors are Perl, Microsoft SQL Server, and Oracle. MySQL has much strength [29] including the following:

- High performance
- Low cost
- Ease of learning and configure
- Portability
- Availability of source code

A more detail discussion of these strengths follows.

Performance

MySQL is undeniably fast.

Low Cost

MySQL is available at no cost.

Ease of use

Most modern databases use SQL, MySQL is also easier to set up than many similar product.

Portability

MySQL can be used in many different Unix as well as under Microsoft windows

Source code

As with PHP, you can obtain and modify the source code of MySQL

6.3 Users and Privileges

A MySQL system can have many users. The root user should generally be used for administration purpose only, for security reasons. For each user who needs to use the system, I need to set up an account and password.

6.4 Designing Database for data logger

First you need to create a database to store your data. Make certain you have your MySQL daemon running.

6.4.1 Create database

The general form [36] to create database :

```
Mysql> create database dbname;
```

At the MySQL prompt type :

```
Mysql> create database thesis;
```

6.4.2 Open database

The general form [39] of open database

```
Mysql> use dbname;
```

At the MySQL prompt, enter the following command

```
Mysql> use thesis;
```

The MySQL server responds with:

```
Database changed
```

6.4.3 Creating database tables

Now you can create [36] the table you'll be using for the upcoming projects. Enter the following commands at the MySQL prompt. Hit Enter after each line. MySQL doesn't try to interpret the commands until it sees a semicolon (;), so the command itself isn't really executed on the server until you enter the last line.

The general form of a CREATE TABLE statement is

Create table tablename(columns) ;

For example to create table for channel 1 values:

```
mysql> create table ch1(
```

```
-> reading_number int not null auto_increment,
```

```
-> day int not null,
```

```
-> month int not null,
```

```
-> year int not null,
```

```
-> time time not null,  
-> value int not null),  
-> primary key(reading_number));
```

Another example to create table for users:

```
mysql> create table sys_admin(  
-> user varchar(20) not null,  
-> password varchar(20) not null);
```

If the server gives you an ERROR and spits out a bunch of garbage at you, just try again from the top. You need to enter each line in the sequence from the beginning, exactly as shown.

What Keywords Mean

NOT NULL means [33] that all the rows in the table must have a value in this attribute. If it isn't specified, the field can be blank (NULL).

AUTO-INCREMENT is a special MySQL feature you can use on integer columns. It means if we leave that field blank when inserting rows into the table, MySQL will automatically generate a unique identifier value. The value will be one greater than the maximum value in the column already. You can only have one of these in each table. Columns that specify AUTO_INCREMENT must be indexed.

PRIMARY KEY after a column name specifies that this column is the primary key for the table. Entries in this column have to be unique. MySQL will automatically index this column. Notice that where I used it above with reading_number in the ch1 table, I used it with AUTO-INCREMENT. The automatic index on the primary key takes care of the index required by AUTO INCREMENT.

Specifying PRIMARY KEY after a column name can only be used for single column primary keys.

6.4.4 Looking at the Database with SHOW and DESCRIBE

The tables can be viewed [39] in the database by typing

```
mysql> show tables;
```

MySQL will display a list of all the tables in the database:

```
mysql> show tables;
+-----+
| Tables_in_thesis |
+-----+
| ch1               |
| ch2               |
| ch3               |
| ch4               |
| ch5               |
| ch6               |
| ch7               |
| ch8               |
| sys_admin         |
| values1           |
+-----+
```

Figure 6.1 Show command

You can view the database by typing

```
mysql> show databases;
```

MySQL will display a list of all the tables in the database:

You can see a structure of particular table, by command describe, or desc ,General form

```
mysql> desc tablename;
```

```
mysql> desc ch1;
```

And the server display the description of the selected table

```
mysql> desc ch1;
+-----+-----+-----+-----+-----+-----+
| Field          | Type   | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| reading_number | int(11) |      | PRI | NULL    | auto_increment |
| day            | int(11) |      |     | 0        |                |
| mounth         | int(11) |      |     | 0        |                |
| year           | int(11) |      |     | 0        |                |
| time           | time   |      |     | 00:00:00 |                |
| value          | int(11) |      |     | 0        |                |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.02 sec)
```

Figure 6.2 Describe command

6.5 SQL Queries Used In Data Logger Database

6.5.1 Inserting Data

We can use the insert statement to put rows of data into the database, The usual form [41] of an insert statement is

Insert into tablename values ('value1', 'value2', ..., 'valuen');

For example :

```
mysql> insert into chl values('','10','10','2003','12:12:12','25');
```

The values specified would be used to fill in the table column in order, to see how it looks in the table by issuing a select statement.

Type the following at the MySQL prompt:

```
mysql> select * from chl;
```

The server responds with:

```
mysql> select * from chl;
+-----+-----+-----+-----+-----+-----+
| reading_number | day | month | year | time   | value |
+-----+-----+-----+-----+-----+-----+
| 1             | 10  | 10    | 2003 | 12:12:19 | 35    |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.08 sec)
```

Figure 6.3 MySQL After Insert command

6.5.2 Modifying Data

If we wish to modify something [36] incorrectly entered, or want to modify some data, to correct the error, or modify the data, we can use the update command.

The general form of an update statement:

Update tablename set column1=expression1, column2=expression2 [where condition];

An example is given below:

```
mysql> select * from sys_admin;
```

```
mysql> select * from sys_admin;
+-----+-----+
| user      | password |
+-----+-----+
| mohammed  | 1979     |
| dr_dogan  | 12345    |
+-----+-----+
2 rows in set (0.00 sec)
```

Figure 6.4 MySQL before update command

Type the following at the MySQL prompt:

```
mysql> update sys_admin set password='NEU' where user = 'mohammed' ;
```

To make sure everything is correct, let us issue the select command again. Type the following at the MySQL, prompt:

```
mysql> select * from sys_admin;
```

The server responds:

```
mysql> SELECT * FROM SYS_ADMIN;
+-----+-----+
| user      | password |
+-----+-----+
| mohammed  | NEU      |
| dr_dogan  | 12345    |
+-----+-----+
2 rows in set (0.00 sec)
```

Figure 6.5 MySQL After update command

And as can be seen from the output, the password is changed correctly.

6.5.3 Deleting Data

To delete a row [37] from specific table, the general form of a delete statement is:

```
mysql> delete from tablename [where condition]
```

For example:

```
mysql> select * from ch1;
```

reading_number	day	mounth	year	time	value
2	10	10	2003	11:01:00	22
1	10	10	2003	11:00:00	20
3	1	1	2004	12:12:18	55
4	1	1	2004	12:12:19	30
5	1	1	2004	12:12:20	30

```
5 rows in set (0.01 sec)
```

Figure 6.6 MySQL before delete command

Type this command at the MySQL prompt:

```
mysql> delete from ch1 where year = 2003;
```

Now do a select statement and see what the table looks like.

```
mysql> select * from ch1;
```

The server responds with:

```
mysql> delete from ch1 where year=2003;
Query OK, 2 rows affected (0.03 sec)

mysql> select * from ch1;
```

reading_number	day	mounth	year	time	value
3	1	1	2004	12:12:18	55
4	1	1	2004	12:12:19	30
5	1	1	2004	12:12:20	30

```
3 rows in set (0.04 sec)
```

Figure 6.7 MySQL After delete command

The delete command simply deletes all rows that stored in 2003,

6.5.4 Retrieving Data from the Database

The SELECT statement. It's used to retrieve data from a database by selecting rows that match specified criteria from a table. There are a lot of options and different ways to use the SELECT statement.

The basic form of a SELECT is

SELECT items FROM tables WHERE condition

This query lists the contents of the time and value columns from the ch1 table where the reading is less than 40:

```
mysql> select time,value from ch1 where value<40 ;
```


This query, has the following output

```
mysql> select time,value from ch1 where value<40;
+-----+-----+
| time      | value |
+-----+-----+
| 12:12:19  | 30    |
| 12:12:20  | 30    |
+-----+-----+
```

Figure 6.8 Select some columns command

To retrieve all columns and all rows from the ch1 table, we would use

```
mysql> select * from ch1;
```

which will give the following output:

```
mysql> select * from ch1;
+-----+-----+-----+-----+-----+-----+
| reading_number | day | month | year | time      | value |
+-----+-----+-----+-----+-----+-----+
| 4              | 1   | 1     | 2004 | 12:12:18  | 55    |
| 5              | 1   | 1     | 2004 | 12:12:19  | 30    |
| 7              | 1   | 1     | 2004 | 12:12:20  | 30    |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

Figure 6.9 Select all columns command

6.5.4.1 Retrieving Data with Specific Criteria

To access a subset of the rows in a table, we need to specify some selection criteria.

One can do this with a WHERE clause. For example:

```
mysql> select * from ch1 where value='30';
```

Will select all the columns from the ch1 table, but only the rows with reading value equal 30. Here's the output:

```
mysql> select * from ch1 where value='30';
+-----+-----+-----+-----+-----+-----+
| reading_number | day | month | year | time      | value |
+-----+-----+-----+-----+-----+-----+
| 5              | 1   | 1     | 2004 | 12:12:19  | 30    |
| 7              | 1   | 1     | 2004 | 12:12:20  | 30    |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.36 sec)
```

Figure 6.10 Select specific criteria command

The WHERE clause specifies the criteria used to select particular rows. In this case, I have selected rows with a value equal 30.

6.5.5 Altering Tables After Creation

If we want to alter the structure of the tables within our database, we can use the flexible ALTER TABLE statement. The basic form of this statement :

ALTER TABLE tablename alteration [,alteration ...]

Let's look at a few of the more common uses of ALTER TABLE.

One thing that comes up frequently is the realization that you haven't made a particular column "big enough" for the data it has to hold. For example, in sys_admin table, I have allowed names to be 20 characters long. After I start getting some data, I might notice that some of the names are too long and are being truncated. I can fix this by changing the data type of the column so that it is 40 characters long instead:

```
mysql> alter table sys_admin modify user varchar(40) not null;
```

```
mysql> desc sys_admin;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| user  | varchar(20) |      |      |          |       |
| password | varchar(20) |      |      |          |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)

mysql> alter table sys_admin modify user varchar(40) not null;
Query OK, 2 rows affected (0.06 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> desc sys_admin;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| user  | varchar(40) |      |      |          |       |
| password | varchar(20) |      |      |          |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Figure 6.11 Altering the sys_admin table

Another common occurrence is the need to add a column. If the column email does not exist, I can add email column to the sys_admin table as follows:

```
mysql> alter table sys_admin add email varchar(40) after password ;
```

I can delete the column that I just added it as follows:

```
mysql> alter table sys_admin drop email ;
```

```
mysql> desc sys_admin;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| user  | varchar(40) |      |      |          |       |
| password | varchar(20) |      |      |          |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> alter table sys_admin add email varchar(40) after password;
Query OK, 2 rows affected (0.59 sec)
Records: 2 Duplicates: 0 Warnings: 0

mysql> desc sys_admin;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| user  | varchar(40) |      |      |          |       |
| password | varchar(20) |      |      |          |       |
| email | varchar(40) | YES  |      | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.42 sec)

mysql> alter table sys_admin drop email;
Query OK, 2 rows affected (0.04 sec)
Records: 2 Duplicates: 0 Warnings: 0

mysql> desc sys_admin;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| user  | varchar(40) |      |      |          |       |
| password | varchar(20) |      |      |          |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Figure 6.12 Add and Drop column commands

6.5.6 Dropping Tables

At times we may want to get rid of an entire table. We can do this with the DROP TABLE statement. This is very simple, and it looks like this:

```
DROP TABLE table_name;
```

This will delete all the rows in the table and the table itself.

6.5.7 Dropping a Whole Database

We can go even further and eliminate an entire database with the `DROP DATABASE` statement, which looks like this:

```
DROP DATABASE database_name;
```

This will delete all the rows, all the tables, all the indexes, and the database itself .

6.6 Summary

MySQL is very fast, robust, enables store, search, sort, and retrieve data. The MySQL server controls access to the data to ensure that multiple users can work with it concurrently, it is a multi-user, multi-threaded server. It is a suitable storage to contain the data generated by the data logger that is sent to the PC over the RS232 serial communications link.

7. USING PHP AND MYSQL TO COLLECT AND ANALYZE THE DATA

7.1 Overview

The data created by the data logger is sent to the PC using the standard RS232 communications line. The PHP script running on the PC opens the serial communications channel and reads the serial data. This data is then stored in the MySQL database for future analysis. MySQL is a small and easily manageable database and this is one of the reasons why it has been selected in this thesis.

This chapter describes the programs developed by the author in order to capture and display the data received from the data logger.

7.2 Web Database Architectures

The following subsections of the thesis will describe:

- The internal architecture of the database designed by the author
- The details of the PHP scripting programs developed by the author
- The details of the MySQL database developed by the author

Architecture

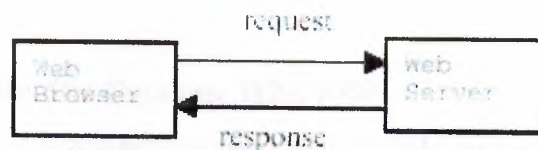


Figure7.1 The basic operation of a web

This system consists of two objects:

A web browser and a web server [28], a communication link is required between them. A web browser make a request of the server, the server send back the response. This

architecture suits a server delivering static pages well. The architecture that delivers a database backed web site is a little more complex.

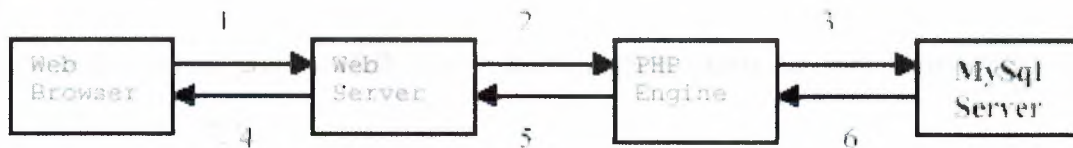


Figure 7.2 The complex operation of a web

A typical web database transaction consists of the following stages:

- 1- A user's Web browser issues an HTTP request for a particular Web page. For example, the user might have requested a search for all information about ch1 reading, using an HTML form. The search results page is called results.
- 2- The Web server receives the request for results, retrieves the file. And passes it to the engine for processing.
- 3- The engine begins parsing the script. Inside the script is a command to connect to the database and execute a query (perform the search for books). PHP opens a connection to the MySQL server and sends on the appropriate query.
- 4- MYSQL server receives the database query, processes it, and sends the results back, to the PHP engine.
- 5- The PHP engine finishes running the script that will usually involve formatting the query result nicely in HTML. It then returns the resulting HTML to the Web server.
- 6- The web server passes the HTML back to the browser, where the user can see the information requested.

7.3 Putting Content Into The Database With PHP

Using PHP for the task is less cumbersome, more flexible, and it can easily be done using a Web browser.

The logic behind PHP and database interaction is simple.

- Connect to the database server and login.
- Choose the database to use.
- Send SQL queries to the server to add, delete, and modify data.

3.1 Setting Up a Connection

To connect to the MySQL server [42] one has to issue the command:

```
$db = mysql_connect($db_server_address, $db_username, $db_password)
```

In this thesis the author has used the `mysql_connect()` function to connect to the database. The user has to pass to it the name of the host on which the MySQL server is running, the username to log in as, and the password of that user. All of these are optional, and if they are not specified, the function uses some sensible defaults: localhost for the host, the username that the PHP process runs as, and a blank password.

The function returns a link identifier to the MySQL database on success (which you ought to store for further use) or false on failure. The result is worth checking as none of the rest of code will work without a valid database connection. The following code was used by the author:

```
$db = mysql_connect($db_server_address, $db_username, $db_password)
    or die("\nError : ". mysql_error(). "\n");
echo("\nConnected successfully to database server.\n");
```

This is the same as

```
if( !$db){
    echo "Error: Could not connect to database. Please try again later.";
    exit ;}
else {echo("\nConnected successfully to database server.\n");}
```

An alternative function that does almost the same thing as `mysql_connect()` is `mysql_pconnect()`. The difference is that `mysql_pconnect` returns a persistent connection to the database.

A normal connection to the database will be closed when a script finishes execution, or when the script calls the `mysql_close ()` function. a persistent connection remains open after the script finishes execution cannot be closed with the `mysql_close ()` function.

When `mysql_pconnect ()` is called, before it tries to connect to the database, it will automatically check if there is a Persistent connection already open. If so, it will use this one rather than opening a new one. This saves time and server overhead

7.3.2 Choosing a Database to Use

When using MySQL from a command line interface, we need to tell it which database we plan to use with a command such as:

```
use thesis;
```

The same rule applies when connecting from the Web. I perform this from PHP with calling the `mysql_select_db($db_name, $db)` function, as follows:

```
mysql_select_db("thesis",$db);
```

The above example tries to use the database called *thesis*. We can also optionally include the database link you would like to perform this operation on (in this case `$db`), but if we don't specify it, the last opened link will be used. If you don't have a link open, the default one will be opened as if you had called `mysql_connect ()`.

7.3.3 Querying the Database

To actually perform the query, we can use the `mysql_query()` function.

Before doing this, it's a good idea to set up the query you want to run:

```
$sql = "insert into ch1 (", '$dd', '$mm', '$yy', '$current_time', '$n_data')";
```

In this case, insert to *ch1* table these values (reading_number, day, month, year, time, value);

We can now run the query:

```
mysql_query($sql);
```

We pass it the query we want to run, and optionally, the database link (again, in this case \$db). If not specified, the function will use the last opened link. If there isn't one, the function will open the default one as if you had called `mysql_connect()`.

7.4 Using PHP To Read Data From The Database

7.4.1 Querying the Database

To actually perform the query, we can use the `mysql_query()` function. Before doing this, it's a good idea to set up the query you want to run:

```
$sql = "select * from ch1 where day='$D1' && month='$D2' && year = '$D3'";
```

In this case, I'm searching for all values that stored in the specific date.

We can now run the query:

```
$result = mysql_query($sql);
```

We can use the `mysql_db_query()` function instead `mysql_query()`. It's very similar but allows you to specify which database you would like to run the query on. It is like a combination of the `mysql_select_db()` and `mysql_query()` functions.

Both of these functions return a result identifier (that allows you to retrieve the query results) on success and false on failure. We should store this (as we have in this case in \$result) so that you can do something useful with it.

7.4.2.Retrieving the Query Results

A variety of functions are available to out the results out of the result identifier in different ways. The result identifier is the key to accessing the zero, one, or more rows returned by the query.

In this thesis the author used `mysql_fetch_array()`. This function gives the rows one by one that returned by the query and one should pass it the result identifier. It's useful to know the number of rows if we plan to process or display the results, we should know how many rows to loop them:

```
<? while($row_count = mysql_fetch_array($result)) {?>
    <tr>
    <td
width="34%"align="center"><?printf("%s",$row_count["time"]);?>&nbsp;  </td>
    <td
width="34%"align="center"><?printf("%s",$row_count["value"]);?>&nbsp;  </td>
    >
    </tr><? }?>
```

In each iteration of this loop, we are calling `mysql_fetch_array()`. The loop will not execute if no rows are returned. This is a function that takes each row from the result set and returns the row as an associative array, with each key an attribute name and each value the corresponding in the array.

```
$row_count= mysql_fetch_array($result);
```

Given the associative array `$row_count`, we can go through each field and display them appropriately,.

7.5 Other PHP Database Interfaces

PHP supports libraries for connecting to a large number of databases including Oracle, Microsoft SQL Server, mSQL, and PostgreSQL. In general, the principles of connecting to and querying any of these databases are the same. The individual function names vary, and different databases have slightly different functionality. but connecting to MySQL, easily adapt your knowledge to any of the others.

If we want to use a database that doesn't have a specific library available in PHP we can use the generic ODBC functions. ODBC stands for Open Database Connectivity . It is a standard to connect the databases. It has the most limited functionality of any of the function sets,

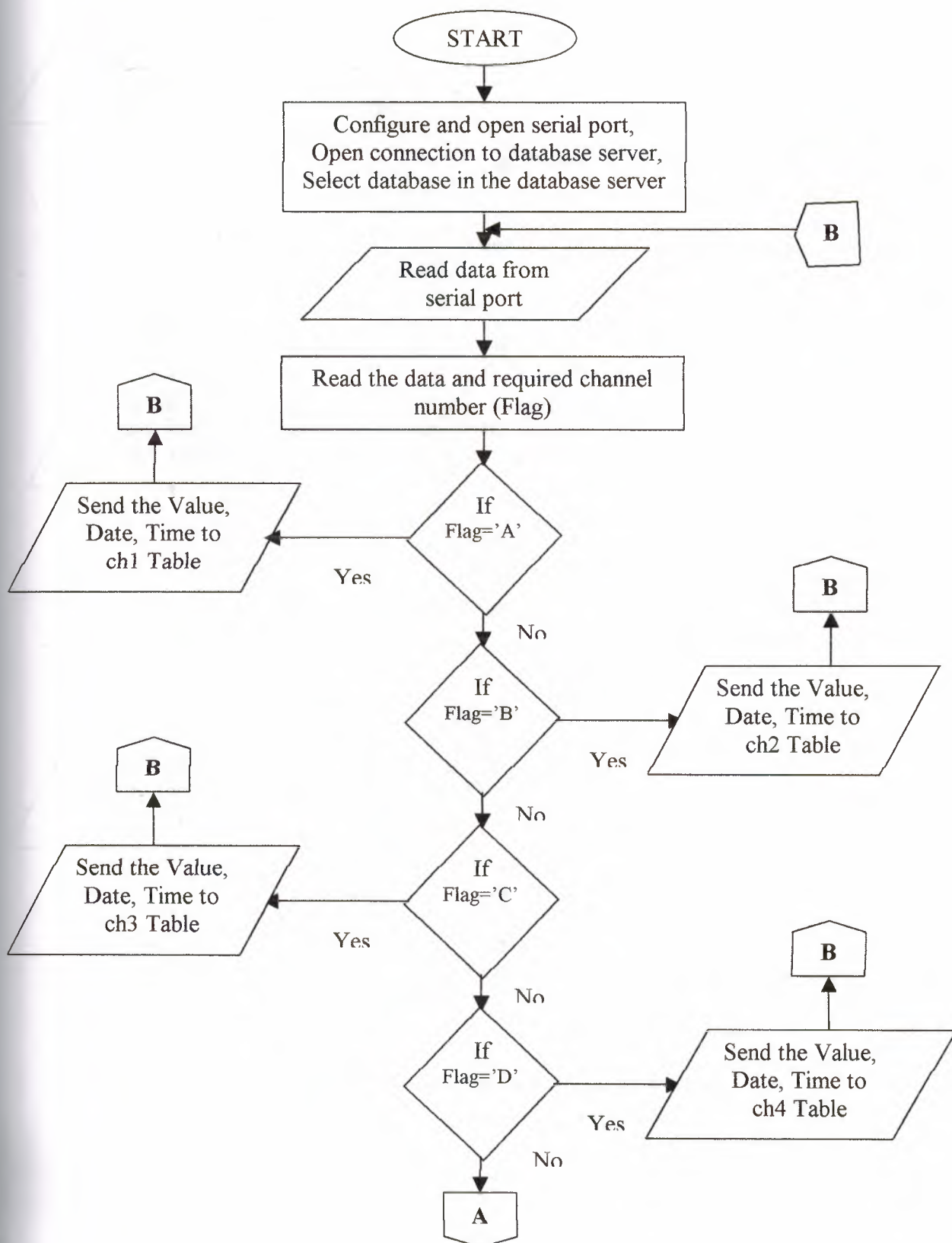
7.6 PHP Script Files

This section describes the PHP script files developed by the author.

7.6.1 READ.PHP Script File

This program reads the data from the serial port (Com1) and stores the data in the MySQL database.

7.6.1.1 Flow Chart of READ.PHP Script



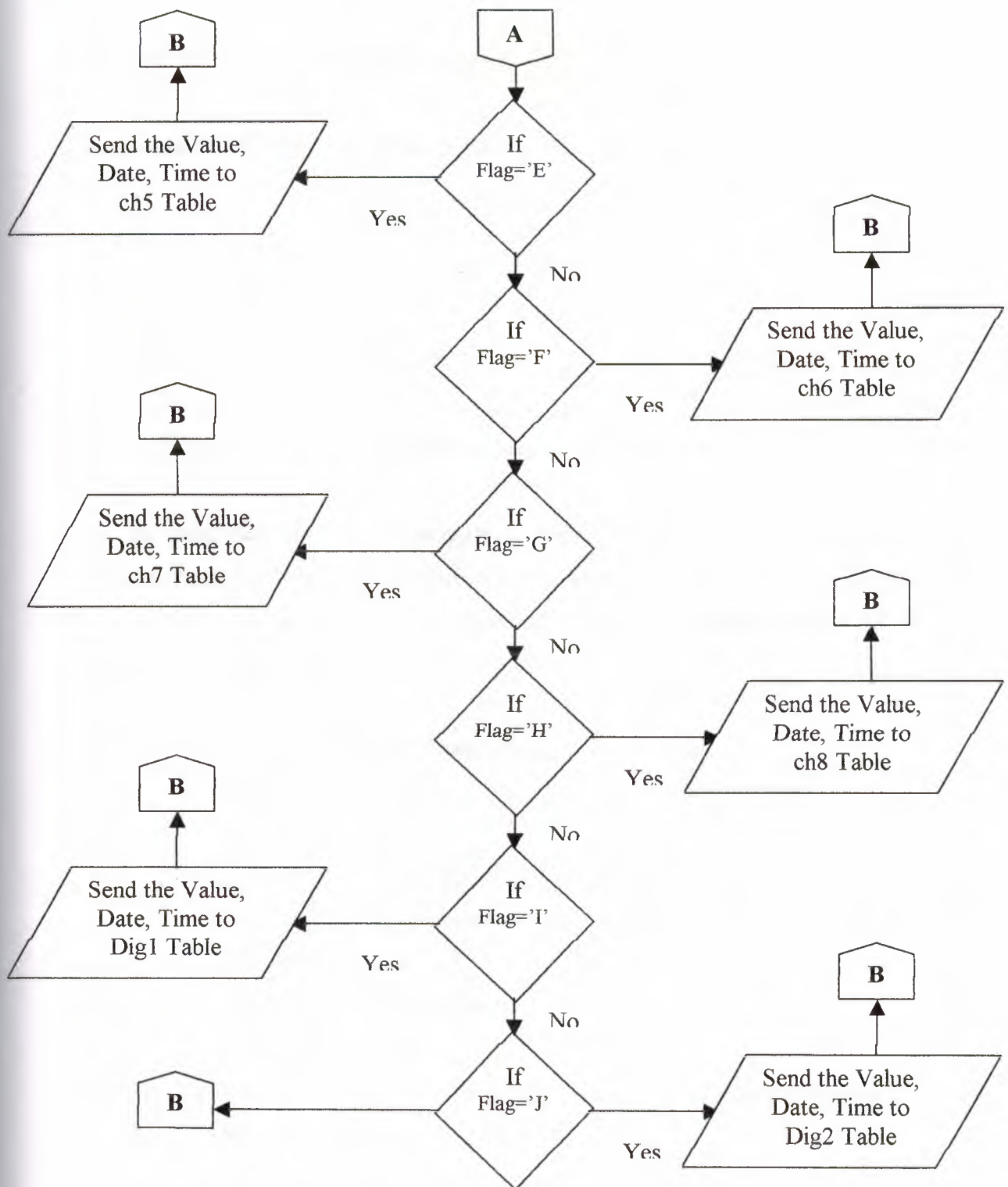


Figure 7.3 Flow Chart of READ.PHP Script

7.6.1.2 READ.PHP Script File Discussion

The operation of the READ.PHP script file is explained below.

```
$com_no = 1;  
$baud = 9600;  
$parity = "N";  
$data_bits = 8;  
$stop_bits = 1;  
set_com($com_no, $baud, $parity, $data_bits, $stop_bits);
```

First, the serial port parameters are defined, where port COM1 was chosen as the serial communications port, the speed of transmission was chosen to be 9600 bits/sec , the size of the data 8 bits with no parity, and there is one stop bit. Then the function `set_com` was called with these parameters :

```
function set_com($com_no, $baud, $parity, $data, $stop) {  
    $port_name = "COM1:";  
    $parity = "N";  
    $cmd_str = "MODE $port_name BAUD=$baud PARITY=$parity  
    DATA=$data STOP=$stop TO=ON OCTS=ON ODSR=OFF IDSR=OFF  
    RTS=HS DTR=ON";  
    exec($cmd_str, $output, $result);  
    if($result!=0){  
        echo("\nError while trying to set COM port parameters - exiting.\n");  
        exit;    }
```

This function is used to set DOS com port parameters, and then calls the DOS MODE function to set the parameters. The output from the MODE command is echoed to the local console. If the MODE command is not successful the script exits.

```

$db_server_address = "localhost";
$db_name = "THEISIS";
$db_username = "mohammad";
$db_password = "1979";
$db = mysql_connect($db_server_address, $db_username, $db_password)
    or die("\nError : " . mysql_error(). "\n");
echo("\nConnected successfully to database server.\n");
if (!mysql_select_db($db_name, $db)) {
    echo("\nData base not found in data base server\n");
}
else {echo("\nSuccessfully selected database.\n");}

```

In this part of the script the database server (DB) connection parameters are defined. The server address is the localhost which has IP address 127.0.0.2, and to permit to enter to DB server should use the shown username and password. On the DB server, a database called thesis has been created.

```

$sk=0;
while($sk<10){
    $current_time = date ("H:i:s");
    $dd = date("d");
    $mm = date("m");
    $yy = date("y")+2000
    $data = NULL;
    $data = fgets($serial_port,5);
    $data[1]=(int)$data[1];
    $data[2]=(int)$data[2];
    $data[3]=(int)$data[3];
    $n_data=((($data[1]*100)+($data[2]*10)+($data[3])));
    if($n_data>799)$n_data=($n_data-800);
}

```


The above code is responsible for reading the serial port. Function `fgets()` is used to read 5 bytes of data as a string from the serial port with time stamping. The data is stored in the following array elements: `data[0]` stores a flag which identifies the channel number of the data logger. `data[1]`, `data[2]`, and `data[3]` contain the actual measured analog data in string format. These values are converted into integer.

```
switch($data[0]){
    case 'A':
        $sql = "insert into ch1 values ('', '$dd', '$mm', '$yy', '$current_time',
            '$n_data')";
        if (!mysql_query($sql, $db)) {echo("Error : ".mysql_error()."\n");}
        break;
    case 'B':.....
    .
    .
    .
    case 'J':.....
    default :
        printf("\n error channel selection\n ");
        break;
}
```

The above code uses a switch statement where the channel number is the choice. Then, depending upon the channel number the measured values are stored in the MySQL database tables.

7.6.2 HOME.PHP Script File

This script is an HTML code. It represent the first page that will appear to the user and it shows information about the thesis name and at the end of this page there is a link (go) which when pressed will open the second page (ADMIN.PHP).

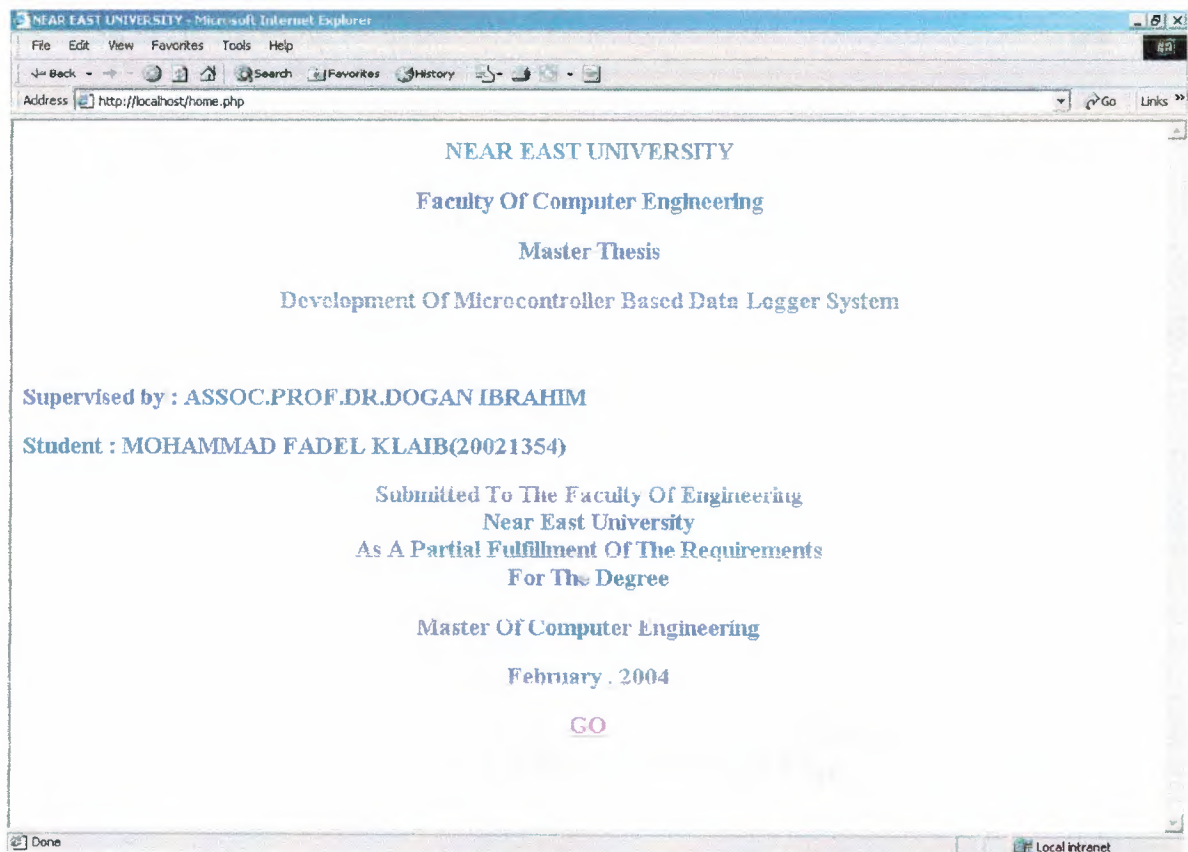


Figure 7.4 HOME.PHP Script Page

7.6.3 ADMIN.PHP Script File

This script file is HTML, and PHP code. It is used for security and it asks the user to enter the username and password. as in (Fig. 7.4). It then compares the information with the information that exist in the database server, and if the user enters incorrect user name or password it will reject him from entering to the main page as in Fig 7.5 and Fig 7.6 respectively. If the user enters the correct information it will open the main page which is handled by the script file CHANNELS.PHP.

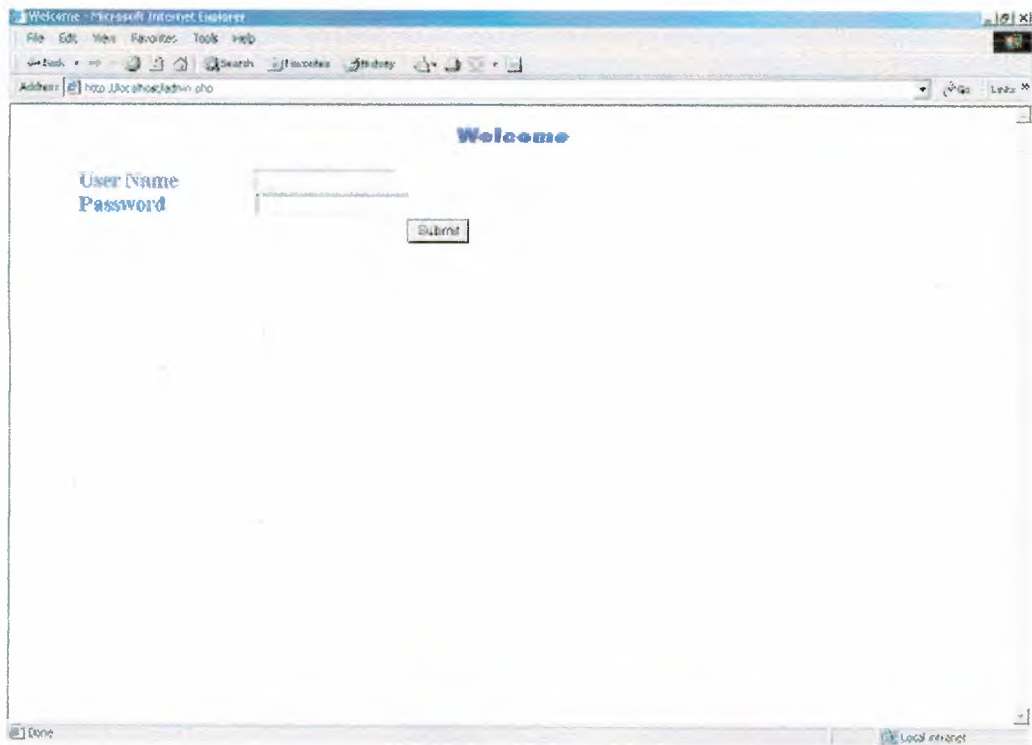


Figure 7.5 ADMIN.PHP Script Page

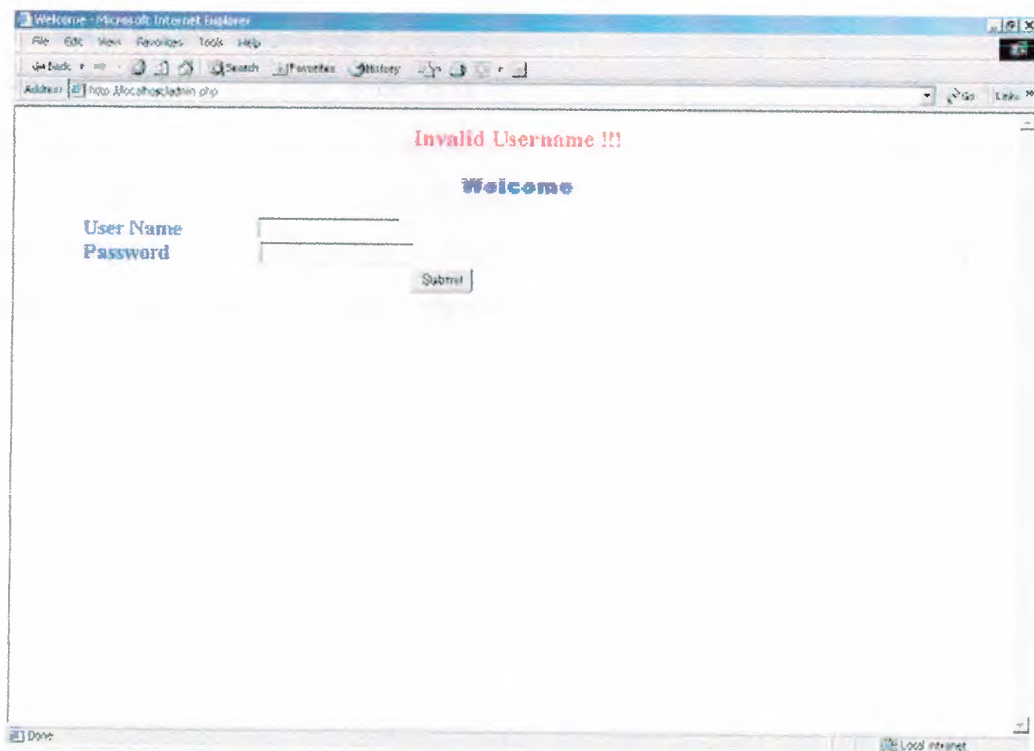


Figure 7.6 Invalid Username Message.

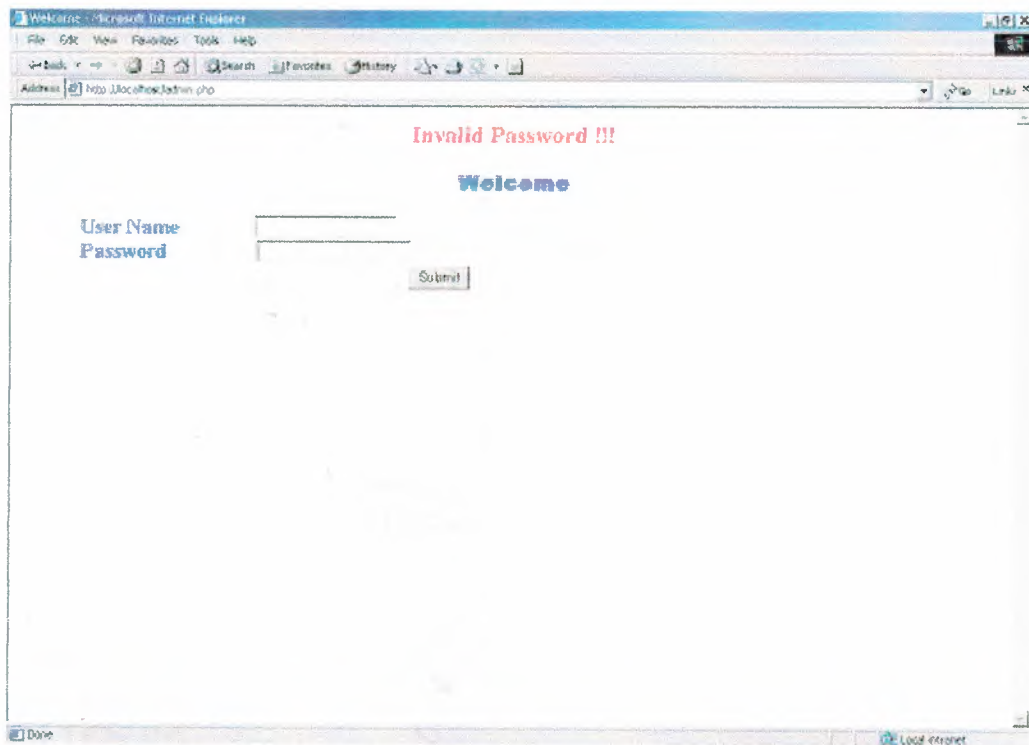
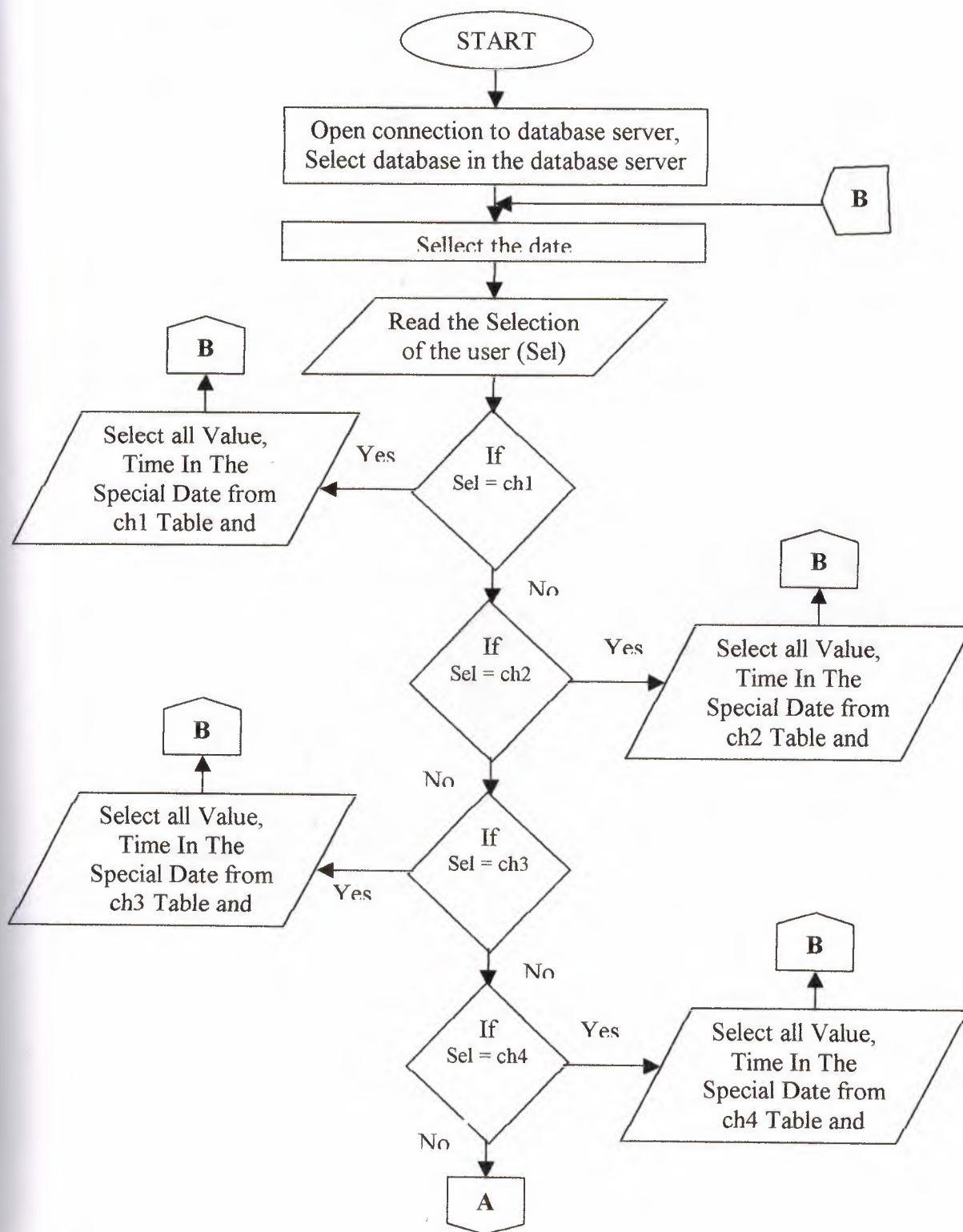


Figure 7.7 Invalid Password Message.

7.6.4 DISPLAY.PHP Script File

This script is *HTML*, and *PHP* code, it is represented in flow chart in (fig 7.8). And in (fig 7.9) the page that lets the user to select which channel data to display with the specified date. This page will show to the user, 8 Analog channels (the output of the analog sensors), and the 16 digital sensor, as shown in (Fig 7.9). First of all the user should select the date and then the required channel number should be selected.

7.6.4.1 Flow Chart of DISPLAY.PHP Script



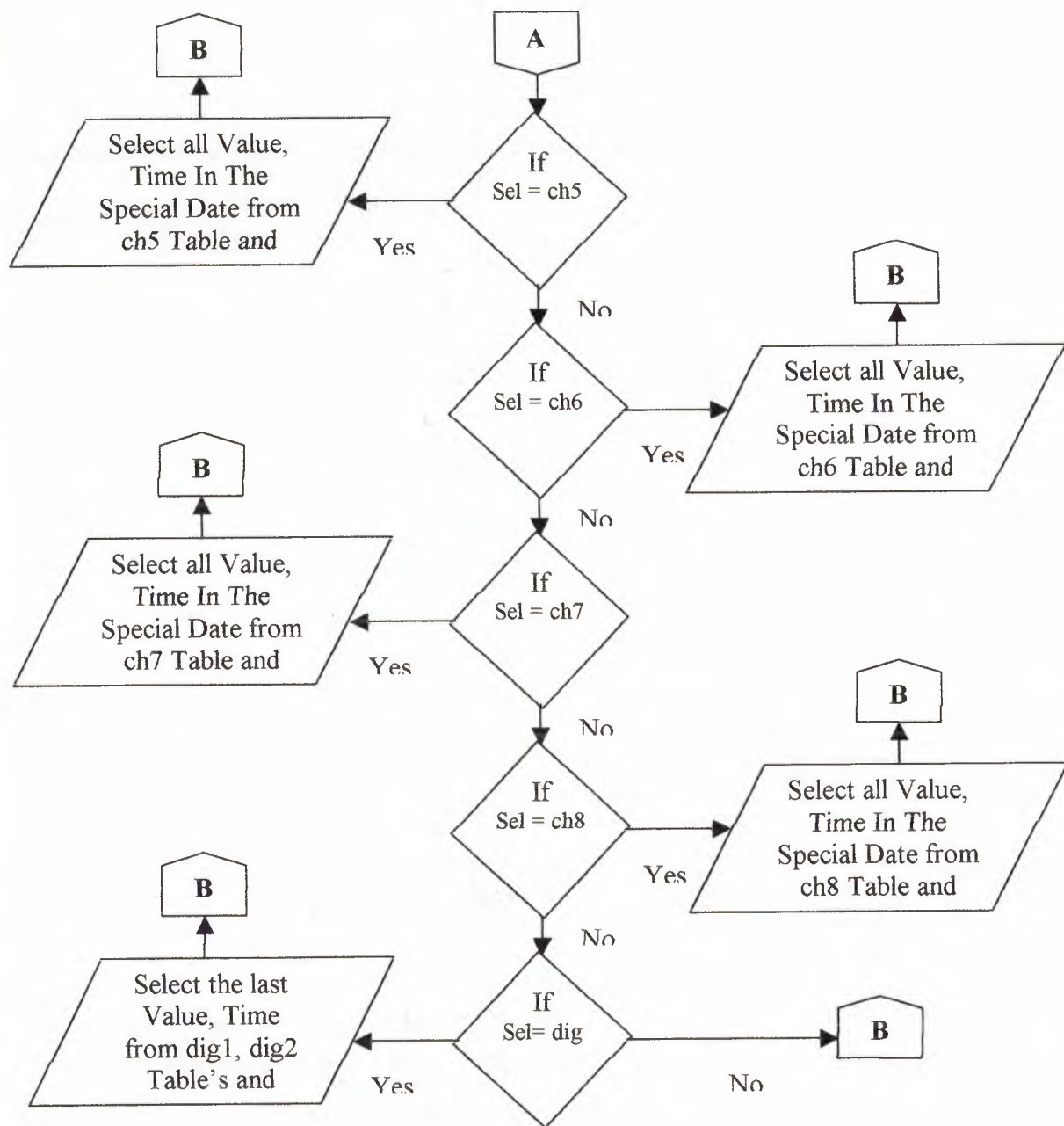


Figure 7.8 Flow Chart of DISPLAY.PHP Script

7.6.4.2 DISPLAY.PHP Script Screens

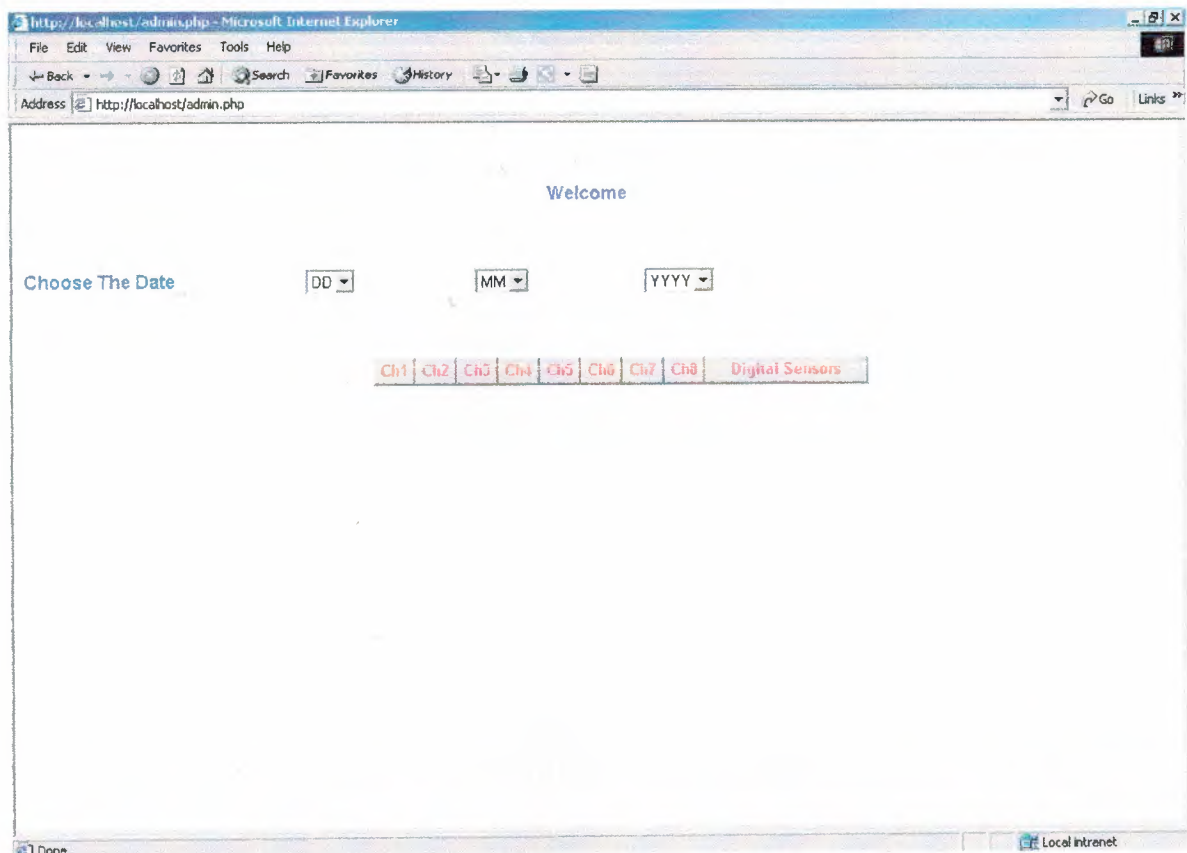


Figure 7.9 DISPLAY.PHP Script Page

For example if the user selects the date as 10-02-2004 and then selects channel 2 (ch2), the web server will bring from the DB server all the values which match the required date and the required channel number, as shown in Fig 7.10.

http://localhost/admin.php - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Search Favorites History

Address http://localhost/admin.php Go Links

The Requested Data Is

Date: 10-02-2004

Time	Value
01:35:35	409
01:35:46	406
01:35:56	409
01:38:09	406
01:38:20	406
01:38:41	409
01:38:50	408
01:39:01	406
01:41:04	404
01:41:14	409
01:41:24	409
01:42:16	407
01:42:26	407
01:42:36	394
01:43:48	395
01:43:58	407
01:48:25	404
01:48:35	404
01:48:46	407
01:54:04	0
01:54:14	0
01:54:24	0

Done Local intranet

Figure 7.10 Ch2 Output Values

Another example, if the user selects the digital sensor data, the web server will display the last reading of all digital sensors that stored on the DB server, where 0 means OFF and 1 means ON state, as shown in Fig. 7.11.

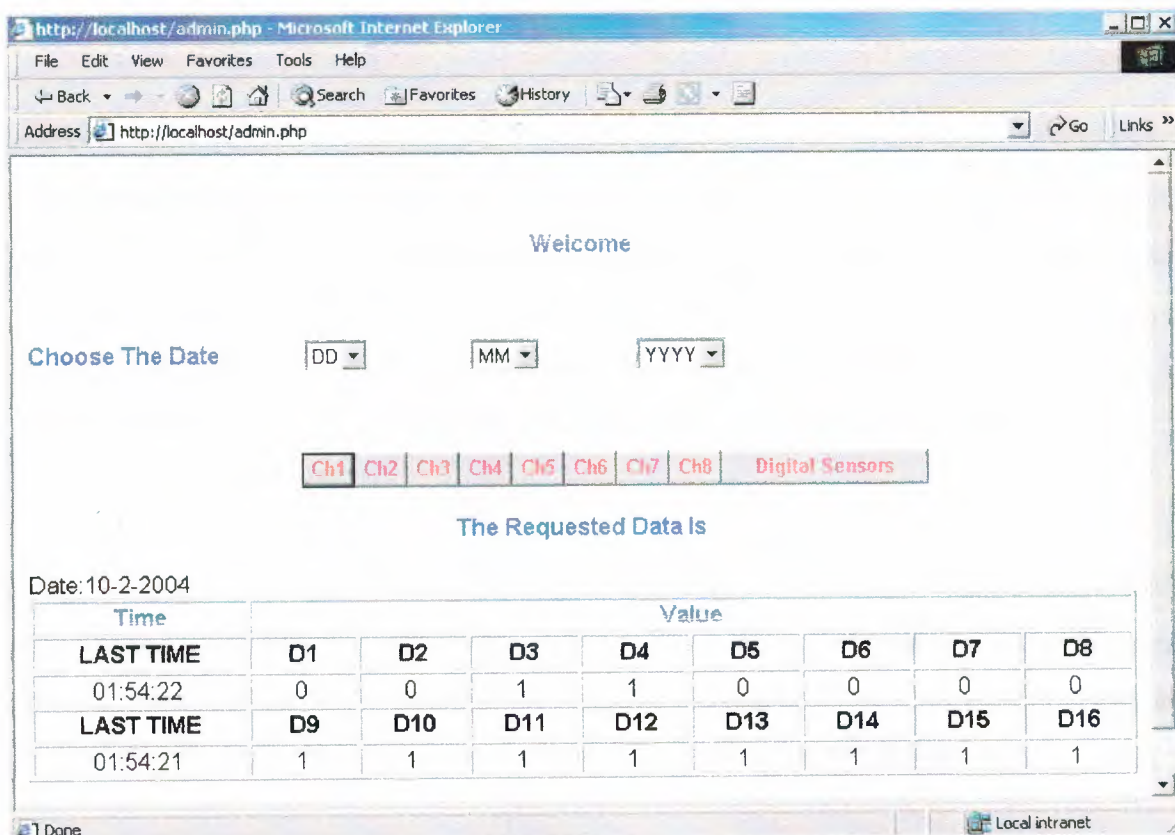


Figure 7.11 Digital Sensors Output

7.7 Summary

When setting out to build an internet based application there are many choices open to the developer. Some of the choices depend on the cost of the products, some depend on the operating system being used, and some depend upon the complexity of the application. In this thesis, the PHP scripting language and the MySQL database are used. The main reasons of using these products are because they are freely available on the Internet and also they are flexible and easy to use.

CONCLUSION

This thesis is about the design and development of a microcontroller based data logging and analysis system. The system receives analog and digital data from various sensors connected to the microcontroller external ports. This data is then sent to a PC for long term logging and analysis purposes. In this thesis, the PHP scripting language is used on the PC to collect and analyse the sensor data. The data read from the microcontroller is stored in a MySQL database for analysis.

The data logger system hardware and the related software has been designed, developed, and tested by the author successfully. A PIC16F877 microcontroller is used in this thesis but any other type of microcontroller can be used as long as it has a serial port. Also, any kind of sensor can be used as long as it can be interfaced to the microcontroller.

The system developed in this thesis has the advantage that it is PHP script based, making it widely available on the web, and easily accessible by a compatible HTML reader, such as the Microsoft Internet Explorer.

The system developed by the author can be enhanced further by including a graphical output. For example, the collected data can be analyzed statistically and the variation of the sensor data can be plotted in the form of a line-chart or a bar-chart. The data logger system can also be made more usable by including a large memory so that the collected data can be stored locally before it is sent to the PC. This will have the advantage that the data logger and the PC can be remote from each other during the data collection. It may also be possible to include a wireless communication link between the data logger and the PC so that the two can operate totally remotely from each other.

REFERENCES

- [1] Campbell Scientific, "Data Acquisition Systems For Science, Industry and Research", November 25, 2003.
http://www.campbellsci.co.uk/data_loggers.htm
- [2] Gregg M. Daly, "The use of low-cost data loggers in monitoring building systems performance", 1996-2003
http://www.onsetcomp.com/Newsletters/412_WEEC_papers.html
- [3] Mukaro, R., Carelse, X.F., 1997. "A serial communication program for accessing a microcontroller-based data-acquisition system". Computers and Geosciences 23 (9) 1027-1032.
- [4] D. Ibrahim, "Z80 Programming", Bilesim Yayincilik, 2003, Ankara, Turkey
- [5] D. Ibrahim, "Microcontroller programming in C for the 8051", Butterworth-Heinemann, 2000, London.
- [6] Optimum Energy Products Ltd, "Temperature & Relative Humidity Loggers".
<http://www.electricitymetering.com/category/category.scd-pscdfa-220277-selectProductLine-idzq312zq1true>
- [7] Quasar Electronics Ltd, "Four Channel Pressure Data Logger", 18 December 2003.
<http://www.electronic-kits-and-projects.com/>
- [8] Veriteq Instruments Inc., "Validatable Temperature Logger", 2004.
<http://www.veriteq.com/html/vrtq2800VL1spec.htm>
- [9] Veriteq Instruments Inc., "Precision Temperature Datalogger", 2004.
<http://www.veriteq.com/html/vrtq2101.htm>
- [10] Pico Technology Ltd., "ADC-16 High resolution data logger", 1991-2004.
<http://www.picotech.com/high-resolution.html>
- [11] Pico Technology Ltd., "PT-104 Platinum Resistance Thermometer Data Logger", 1991-2004.
<http://www.picotech.com/pt100.html>
- [12] Pico Technology Ltd., "Multi Channel Data Acquisition", 1991-2004.
<http://www.picotech.com/data-acquisition.html>

- [13] Pico Technology Ltd., "ADC-10 & ADC-12 -single channel, low cost data loggers", 1991-2004.
<http://www.picotech.com/low-cost-data-loggers.html>
- [14] Curtis D. Johnson, Process Control Instrumentation Technology, 5th ed., Prentice-Hall International Inc., University of Houston, 1997.
- [15] "Sensors".
<http://www.rec.ri.cmu.edu/education/multimedia/sensors.shtml>
- [16] Institute of Physics and IOP Publishing Limited, "magnetometer sensor", 2004.
<http://www.iop.org/EJ/abstract/0957-0233/13/12/339>
- [17] Perng, J. K., Fisher, B., Hollar, S., Pister, K.S.J., "Acceleration Sensing Glove." ISWC International Symposium on Wearable Computers, San Francisco, October 18-19th, 1999.
- [18] "Light Sensor".
<http://www.rec.ri.cmu.edu/education/multimedia/lightsensor.shtml>
- [19] T.D. McGee, Principles and Methods of Temperature Measurement, A Wiley-Interscience Publication, John Wiley & Sons, New York, NY (1988)
- [20] Temperatures.com, "Noncontact Temperature Sensors", 2003 – 2004.
<http://www.temperatures.com/ncsensors.html>
- [21] Pump world, "Atmospheric Pressure", 2003.
<http://www.pumpworld.com/atmos.htm>
- [22] Honeywell International Inc, "Humidity and Moisture Sensors", 2003.
<http://content.honeywell.com/sensing/prodinfo/humiditymoisture/>
- [23] Colin K McCordy, "The PIC16F877 Microcontroller", 2002.
<http://www.mccord.plus.com/FYP/4.htm>
- [24] ARC Electronics, "RS232 Data Interface".
<http://www.arcelect.com/rs232.htm>
- [25] Wichit Sirichote, "RS232C Level Converter", 2002.
<http://chaokhun.kmitl.ac.th/~kswichit/MAX232/MAX232.htm>
- [26] HI-TECH Software, "PICC Lite C Manual", Fourth Printing, November 2002.
<http://www.cs.ucr.edu/~eblock/db/downloads/PICmanual2.pdf>
- [27] Zend Technologies Ltd, "About PHP", 1999– 2000.
<http://www.zend.com/zend/aboutphp.php>

- [28] Christopher Cosentino, Essential PHP for Web Professionals, D0.50x9.00x 6.03, Publisher: Prentice Hall PTR, ISBN: 0130889032; 1st edition (October 2000).
- [29] PHP Documentation Group, " PHP Manual", 1997-2004.
<http://us3.php.net/manual/en/>
- [30] Emmie Lewis, " Beginning PHP Tutorials", 2003-2004.
<http://perl.about.com/cs/beginningphp/>
- [31] W3Schools , "PHP Tutorial".
<http://www.w3schools.com/php/default.asp>
- [32] Tap Internet, "Tap Internet PHP Training Courses".
<http://www.tapinternet.com/index.php/html/main/php-training.html>
- [33] Kevin Yank, "Build Your Own Database Driven Website Using PHP & MySQL", 2nd Edition, Sept 2001, 295 Pages ISBN: 0-9579218-1-0.
- [34] David Gowans, "PHP/MySQL Tutorial", 1999-2001.
<http://www.freewebmasterhelp.com/tutorials/phpmysql>
- [35] W3Schools, "SQL Tutorial", 1999-2004.
<http://www.w3schools.com/sql/default.asp>
- [36] Spoono LLC, "MySQL Tutorial", 2000-2004.
<http://www.tutorialized.com/browse.php?cat=17>
- [37] Spoono LLC, "Delete a Row From mySQL", 2000-2004.
<http://www.tutorialized.com/tutorial.php?id=2998>
- [38] Spoono LLC, "Display X Number of Columns Per Row", 2000-2004.
<http://www.tutorialized.com/tutorial.php?id=3000>
- [39] Spoono LLC, "Displaying a Database", 2000-2004.
<http://www.tutorialized.com/tutorial.php?id=2999>
- [40] Spoono LLC, " Edit a Row In mySQL", 2000-2004.
<http://www.tutorialized.com/tutorial.php?id=3001>
- [41] Spoono LLC, "Add a Row to mySQL", 2000-2004.
<http://www.tutorialized.com/tutorial.php?id=2997>
- [42] Graeme Merrall, "PHP/MySQL Tutorial", 2003.
<http://hotwired.lycos.com/webmonkey/programming/php/tutorials/tutorial4.htm>

APPENDIX A PROGRAM LISTINGS

1- PIC_PROGRAM.C

The listing of the microcontroller C program developed by the author is given below:

```
. *****
;
;*   Project Idea : Data Logger Program USING 16F877          *
;*   Designed by : Eng. Mohammad Klaib                       *
;*   Start Date   : October 23, 2003                         *
;*   Last Update  : February 05, 2004                         *
.*****
;

#include <pic.h>
#include <delay.c>
#include <stdio.h>
#include <serial.c>

// Global variables
unsigned int temp;
char flag;

/* Function To Make Delay 1 Second */
void wait_1_sec()
{
    unsigned int i;
    for(i=0; i<4; i++) DelayMs(250);
}
```

```
/* Function To Sending Data To Serial Port */
```

```
void send_data()
```

```
{
```

```
    unsigned char rs232_1[]=" ";
```

```
    unsigned char rs232_2[]="000";
```

```
    sprintf(rs232_1,"%c",flag);
```

```
    printf("%s",rs232_1);
```

```
    if(temp<100)temp = (temp+800);
```

```
    sprintf(rs232_2,"%d",temp);
```

```
    printf("%s",rs232_2);
```

```
    wait_1_sec();
```

```
}
```

```
/* The Main Function */
```

```
main()
```

```
{
```

```
    unsigned int i;
```

```
    const int lsb = 5000/1024;
```

```
    float mV;
```

```
    unsigned int thigh;
```

```
    /* configuration the register */
```

```
    TRISB = 0;    /* port B output */
```

```
    TRISA = 1;    /* port A input analog 0-4 */
```

```
    TRISE = 0x03; /* port E input analog 5-7 */
```

```
    TRISD = 1;    /* port D input Digital 0-7 */
```

```
    TRISC = 1;    /* port C input Digital 8-15*/
```

```
    ADCON1 = 0x80;
```

```
    /* All PORTA AND PORTE Are ANALOG AND Vref=VDD */
```

```
    ADCON0 = 0x41; /* select PA0 as input to A/D converter and A/D on */
```

```
    DelayMs(250);
```



```

for(;;)          /* Continuous loop */
{
    flag = 'A'; /* initial flag */
    ADCON0 = 0x45; /* start read PA0 */

    /* send Analog Data */
    for(i=0;i<8;i++)
    {
        while((ADCON0 & 4) != 0);
        thigh = ADRESH;
        thigh = 256*thigh + ADRESL;
        mV = thigh*lsb;
        temp =(int) (mV/10.0);
        send_data();
        flag++;
        if(i<7) ADCON0 = 0x45+(i+1)*8; /* read next sensor */
    }

    /* send Digital data PORTD */
    temp = PORTD;
    send_data();

    /* send Digital data PORTC */
    temp = PORTC;
    flag++;
    send_data();
}
}

```

2- READ.PHP SCRIPT

The listing of the READ.PHP script developed by the author is given below:

```
// *****
//   Project Idea: Reading Data From Serial Port And Store it on MySQL DB   *
//   Designed by : Eng. Mohammad Klaib                                     *
//   Start Date   : January 1, 2004                                         *
//   Last Update  : February 09, 2004                                       *
// *****

function set_com($com_no, $baud, $parity, $data, $stop) {
    $port_name = "COM1:";
    $parity = "N";

    // use DOS mode command to set com port parameters
    $cmd_str = "MODE $port_name BAUD=$baud PARITY=$parity DATA=$data
    STOP=$stop TO=ON OCTS=ON ODSR=OFF IDSR=OFF RTS=HS DTR=ON";
    exec($cmd_str, $output, $result);

    // check exit status from MODE command
    if($result!=0){
        echo("\nError while trying to set COM port parameters - exiting.\n");
        exit;
    }

    }//end of the function set_com

//-----
// main function.
//-----

// Set com port parameters
```

```

$com_no = 1;
$baud = 9600;
$parity = "N";
$data_bits = 8;
$stop_bits = 1;

// Open connection to database server
$db_server_address = "localhost";
$db_name = "THESIS";
$db_username = "mohammad";
$db_password = "1979";

// Configure and open serial port
set_com($com_no, $baud, $parity, $data_bits, $stop_bits);
echo("\nTrying to open serial port COM1 ... \n");
$serial_port = fopen("COM$com_no", "w+b");
if ($serial_port) {
    echo("\nSuccessfully opened serial port COM$com_no.\n");
}
else {
    echo("\nError while trying to open serial port COM$com_no - exiting.\n");
    exit;
}

// Open connection to database server
$db = mysql_connect($db_server_address, $db_username, $db_password)
    or die("\nError : " . mysql_error(). "\n");
echo("\nConnected successfully to database server.\n");

// Select database in the database server
echo("\nTrying to select database $db_name ... \n");

```



```

if (!mysql_select_db($db_name, $db)) {
    echo("\nData base not found in data base server\n");
}

else {echo("\nSuccessfully selected database.\n");}

// Read data from serial Port and insert into database
echo("\nStarting data aquisition ... \n\n");
$k=0;
while($k<10){
    $current_time =date ("H:i:s");
    $dd = date("d");
    $mm = date("m");
    $yy = date("y")+2000;

    echo("\nTrying to read from serial port COM$com_no... \n");
    $data = NULL;
    $data = fgets($serial_port,5);
    printf("\n data= %s ",$data);
    $data[1]=(int)$data[1];
    $data[2]=(int)$data[2];
    $data[3]=(int)$data[3];
    $n_data=((($data[1]*100)+($data[2]*10)+($data[3]));
    if($n_data>799)$n_data=($n_data-800);
    printf(" need_data= %d ",$n_data);
    switch($data[0]){
        case 'A':
            $sql = "insert into ch1 values ('', '$dd', '$mm', '$yy', '$current_time',
            '$n_data')";
            if (!mysql_query($sql, $db)) {echo("Error : " . mysql_error()."\n");}
            break;
    }
}

```

```

case 'B':
    $sql = "insert into ch2 values ('', '$dd', '$mm', '$yy', '$current_time',
        '$n_data')";
    if (!mysql_query($sql, $db)) {echo("Error : " . mysql_error()."\n");}
    break;

case 'C':
    $sql = "insert into ch3 values ('', '$dd', '$mm', '$yy', '$current_time',
        '$n_data')";
    if (!mysql_query($sql, $db)) {echo("\nError : " . mysql_error()."\n");}
    break;

case 'D':
    $sql = "insert into ch4 values ('', '$dd', '$mm', '$yy', '$current_time',
        '$n_data')";
    if (!mysql_query($sql, $db)) {echo("\nError : " . mysql_error()."\n");}
    break;

case 'E':
    $sql = "insert into ch5 values ('', '$dd', '$mm', '$yy', '$current_time',
        '$n_data')";
    if (!mysql_query($sql, $db)) {echo("\nError : " . mysql_error()."\n");}
    break;

case 'F':
    $sql = "insert into ch6 values ('', '$dd', '$mm', '$yy', '$current_time',
        '$n_data')";
    if (!mysql_query($sql, $db)) {echo("\nError : " . mysql_error()."\n");}
    break;

case 'G':
    $sql = "insert into ch7 values ('', '$dd', '$mm', '$yy', '$current_time',
        '$n_data')";
    if (!mysql_query($sql, $db)) {echo("\nError : " . mysql_error()."\n");}
    break;

```

```

case 'H':
    $sql = "insert into ch8 values ('', '$dd', '$mm', '$yy', '$current_time',
        '$n_data')";
    if (!mysql_query($sql, $db)) {echo("\nError : " . mysql_error()."\n");}
    break;
case 'I':
    $sql = "insert into dig1 values ('', '$dd', '$mm', '$yy', '$current_time',
        '$n_data')";
    if (!mysql_query($sql, $db)) {echo("\nError : " . mysql_error()."\n");}
    break;
case 'J':
    $sql = "insert into dig2 values ('', '$dd', '$mm', '$yy', '$current_time',
        '$n_data')";
    if (!mysql_query($sql, $db)) {echo("\nError : " . mysql_error()."\n");}
    break;
default :
    echo("\n Error channel selection ");
    break;
} //end of switch

$k++;
} //end of while loop
echo("\nCompleted data aquisition.\n");
?>

```


3- HOME.PHP SCRIPT

The listing of the HOME.PHP script developed by the author is given below:

```
<html>
<head>
<meta http-equiv="Content-Language" content="en-us">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<title>NEAR EAST UNIVERSITY</title>
</head>
<body>
<p align="center"><font color="#000080"><b>NEAR EAST UNIVERSITY</b>
</font></p>
<p align="center"><font color="#000080"><b>Faculty Of Computer Engineering
</b></font></p>
<p align="center"><font color="#000080"><b>Master Thesis</b></font></p>
<p align="center"><font color="#000080"><b>Development Of Microcontroller Based
Data Logger System</b></font></p>
<p align="left">&nbsp;</p>
<p align="left"><font color="#000080"><b>Supervised by :
ASSOC.PROF.DR.DOGAN
IBRAHIM</b></font></p>
<p align="left"><font color="#000080"><b>Student : MOHAMMAD FADEL
KLAIB(20021354)</b></font></p>
<p align="center"><font color="#000080"><b>Submitted To The Faculty Of
Engineering <br>
Near East University<br>
As A Partial Fulfillment Of The Requirements<br>
For The Degree</b></font></p>
```

```
<p align="center"><font color="#000080"><b>Master Of Computer Engineering
</b></font></p>
<p align="center"><font color="#000080"><b>February , 2004</b></font></p>
<p align="center"><font color="#000080"><b><a href="admin.php">GO</a></b>
</font></p>
</body>
</html>
```

4- ADMIN.PHP SCRIPT

The listing of the ADMIN.PHP script developed by the author is given below:

```
<?php
if(isset($B1)):
    $db = mysql_connect("localhost","root");
    mysql_select_db("mom");
    $sql = "select * from sys_admin where user = '$T1'";
    $result = mysql_query($sql);
    $row_count = mysql_fetch_array($result);
    if($T1!=$row_count["user"]):
        ?>
        <p align="center"><b><font color="#FF0000" size="4">Invalid Username !!!</font>
        </b> </p>
        <? elseif ($T2!= $row_count["password"]):?>
        <p align="center"><b><font color="#FF0000" size="4">Invalid Password !!!</font>
        </b></p>
        <? elseif($row_count==0):?>
        <p align="center"><b><font color="#FF0000" size="4">Oh..... where are you going
        !!!</font></b></p>
        <? else:?>
        <frameset rows="1000,*">
        <frameset cols="100%,*">
        <frame name="main" src="http://localhost/ch1.php" target="_self">
        </frameset>
        <? endif,
        endif, ?>
        <form action="admin.php" method="post">
        <html>
```



```

<head>
<meta http-equiv="Content-Language" content="en-us">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<title>Welcome</title>
</head>
<body bgcolor="#FFFFFF">
<p align="center"><b><font face="Arial Black" color="#000080">Welcome</font>
</b></p>
<table border="0" width="100%" cellpadding="0" cellspacing="0">
<tr>
<td width="6%"></td>
<td width="73%"><font color="#000080"><b>User Name</b>
<span style="background-color: #000080"> <input type="text" name="T1"
size="20"> </span> </font></td>
<td width="21%"></td>
</tr>
<tr>
<td width="6%"></td>
<td width="73%"><font color="#000080"><b>Password</b><b>
</b><input type="password" name="T2" size="22"></font></td>
<td width="21%"></td>
</tr>
<tr>
<td width="6%"></td>
<td width="73%"></td>
<td width="21%"></td>
</tr>
<tr>
<td width="6%"></td>

```

```

<td width="73%"></td>
<td width="21%"></td>
</tr>
<tr>
<td width="6%"></td>
<td width="73%">
  <p align="center"><font color="#000080"><input type="submit" value="Submit"
name="B1"></font></td>
  <td width="21%"></td>
</tr></table>
<p><font color="#000080"></font></p>
</body>
</html>
</form>

```

5- CHANNELS.PHP SCRIPT

The listing of the CHANNELS.PHP script developed by the author is given below:

```
<?function select_ch($table_name,$D1,$D2,$D3) {  
    $db = mysql_connect("localhost","root");  
    mysql_select_db("thesis");  
    $sql = "select * from $table_name where day = '$D1' && mounth = '$D2' &&  
    year = '$D3'";  
    $result = mysql_query($sql);  
    ?>  
    <p align="center"><font color="#000080"><b>The Requested Data Is</b></font>  
    </font>  
    </p>  
    <?&br/>    print("Date:");  
    print $D1 ;  
    print("-");  
    print $D2 ;  
    print("-");  
    print $D3 ;  
    ?>  
    <table border="1" width="100%">  
        <tr>  
            <td width="33%" align="center"><font color="#000080"><b>Time</b></font>  
        </td>  
            <td width="34%" align="center"><font color="#000080"><b>Value</b></font>  
        </td>  
    </tr>  
    <?&br/>
```



```

while($row_count = mysql_fetch_array($result)){
    ?>
    <tr>
<td width="34%"align="center"><?printf("%s",$row_count["time"]);?>&nbsp;</td>
<td width="34%"align="center"><?printf("%s",$row_count["value"]);?>&nbsp;</td>
    </tr>
    <?
    }
    }//end of the function

?>
<base target="_self">
<form action="ch1.php" method="post">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1256">
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<title>Welcome</title>
</head>
<body>
<p>&nbsp;</p>
<p align="center"><font color="#000080"><b>Welcome</b></font></p>
<p align="center">&nbsp;</p>
<table border="0" width="100%" cellpadding="0" cellspacing="0">
    <tr>
<td width="25%"><font color="#000080"><b>Choose The Date</b></font></td>
        <td width="15%"><select size="1" name="D1">
            <option selected>DD</option>
            <option>1</option><option>2</option><option>3</option>
            <option>4</option><option>5</option><option>6</option>
            <option>7</option><option>8</option><option>9</option>

```

```

<option>10</option><option>11</option><option>12</option>
<option>13</option><option>14</option><option>15</option>
<option>16</option><option>17</option><option>18</option>
<option>19</option><option>20</option><option>21</option>
<option>22</option><option>23</option><option>24</option>
<option>25</option><option>26</option><option>27</option>
<option>28</option><option>29</option><option>30</option>
<option>31</option></select></td>
<td width="15%"><select size="1" name="D2">
<option selected>MM</option>
<option>01</option><option>02</option><option>03</option>
<option>04</option><option>05</option><option>06</option>
<option>07</option><option>08</option><option>09</option>
<option>10</option><option>11</option><option>12</option>
</select></td>
<td width="45%"><select size="1" name="D3">
<option selected>YYYY</option>
<option>2003</option><option>2004</option><option>2005</option>
<option>2006</option><option>2007</option><option>2008</option>
<option>2009</option><option>2010</option><option>2011</option>
<option>2012</option><option>2013</option><option>2014</option>
</select></td>
</tr>
</table>
<p>&nbsp;</p>
<table border="0" width="100%" cellspacing="0" cellpadding="0">
<tr>
<td width="20%"></td>
<td width="26%"></td>
<td width="14%"><input type="submit" value="Ch1" name="B1" style="color:
#FF0000; font-weight: bold"><input type="submit" value="Ch2" name="B2"

```

```

style="color: #FF0000; font-weight: bold"><input type="submit" value="Ch3"
name="B3" style="color: #FF0000; font-weight: bold"><input type="submit"
value="Ch4" name="B4" style="color: #FF0000; font-weight: bold"><input
type="submit" value="Ch5" name="B5" style="color: #FF0000; font-weight:
bold"><input type="submit" value="Ch6" name="B6" style="color: #FF0000; font-
weight: bold"><input type="submit" value="Ch7" name="B7" style="color: #FF0000;
font-weight: bold"><input type="submit" value="Ch8" name="B8" style="color:
#FF0000; font-weight: bold"><input type="submit" value="Digital Sensors"
name="B9" style="color: #FF0000; font-weight: bold"></td>

```

```

<td width="20%"></td><td width="20%"></td><td width="20%"></td>

```

```

<td width="20%"></td><td width="20%"></td>

```

```

</tr></table>

```

```

<?

```

```

if(isset($B1)):select_ch('ch1',$D1,$D2,$D3);endif;

```

```

if(isset($B2)):select_ch('ch2',$D1,$D2,$D3);endif;

```

```

if(isset($B3)):select_ch('ch3',$D1,$D2,$D3);endif;

```

```

if(isset($B4)):select_ch('ch4',$D1,$D2,$D3);endif;

```

```

if(isset($B5)):select_ch('ch5',$D1,$D2,$D3);endif;

```

```

if(isset($B6)):select_ch('ch6',$D1,$D2,$D3);endif;

```

```

if(isset($B7)):select_ch('ch7',$D1,$D2,$D3);endif;

```

```

if(isset($B8)):select_ch('ch8',$D1,$D2,$D3);endif;

```

```

if(isset($B9)):

```

```

$db = mysql_connect("localhost","root");

```

```

mysql_select_db("thesis");

```

```

$sql2 = "select * from dig1";

```

```

$result2 = mysql_query($sql2);

```

```

while(($row_count2 = mysql_fetch_array($result2))){$needed_row=$row_count2;}

```

```

?>

```

```

<p align="center"><font color="#000080"><b>The Requested Data Is</b></font>

```

```

</p>

```

```

<?

```



```

<td width="10%" align="center"><?printf("%d",$bin%2);$bin=$bin/2;?>&nbsp;</td>
<td width="10%" align="center"><?printf("%d",$bin%2);$bin=$bin/2;?>&nbsp;</td>
<td width="10%" align="center"><?printf("%d",$bin%2);$bin=$bin/2;?>&nbsp;</td>
<td width="10%" align="center"><?printf("%d",$bin%2);$bin=$bin/2;?>&nbsp;</td>
</tr>
<?
$sql3 = "select * from dig2";
$result3 = mysql_query($sql3);
while(($row_count3 = mysql_fetch_array($result3))){$needed_row=$row_count3;}
?>
<tr>
<td width="20%">
<p align="center"><b>LAST TIME</b></td><td width="10%">
<p align="center"><b>D9</b></td><td width="10%">
<p align="center"><b>D10</b></td><td width="10%">
<p align="center"><b>D11</b></td><td width="10%">
<p align="center"><b>D12</b></td><td width="10%">
<p align="center"><b>D13</b></td><td width="10%">
<p align="center"><b>D14</b></td><td width="10%">
<p align="center"><b>D15</b></td><td width="10%">
<p align="center"><b>D16</b></td></tr>
<tr>
<td width="20%" align="center"><?printf("%s",$needed_row["time"]);?>&nbsp;</td>
<? $bin=(int)$needed_row["value"];?>
<td width="10%" align="center"><?printf("%d",$bin%2);$bin=$bin/2;?>&nbsp;</td>
<td width="10%" align="center"><?printf("%d",$bin%2);$bin=$bin/2;?>&nbsp;</td>
<td width="10%" align="center"><?printf("%d",$bin%2);$bin=$bin/2;?>&nbsp;</td>
<td width="10%" align="center"><?printf("%d",$bin%2);$bin=$bin/2;?>&nbsp;</td>
<td width="10%" align="center"><?printf("%d",$bin%2);$bin=$bin/2;?>&nbsp;</td>
<td width="10%" align="center"><?printf("%d",$bin%2);$bin=$bin/2;?>&nbsp;</td>
<td width="10%" align="center"><?printf("%d",$bin%2);$bin=$bin/2;?>&nbsp;</td>

```

```
<td width="10%" align="center"><?printf("%d",$bin%2);$bin=$bin/2;?>&nbsp;</td>
</tr>
<?
endif;
?>
</table>
<p>&nbsp;</p>
</body>
</html>
</form>
```


APPENDIX B

TABLES USED IN THE PROGRAMS

The details of the tables used in the programs are given below:

1- User's Table

```
mysql> create table sys_admin(  
-> user varchar(20) not null,  
-> password varchar(20) not null);
```

2- Channel 1 Table

```
mysql> create table ch1(  
-> reading_number int not null auto_increment,  
-> day int not null,  
-> month int not null,  
-> year int not null,  
-> time time not null,  
-> value int not null),  
-> primary key(reading_number));
```

3- Channel 2 Table

```
mysql> create table ch2(  
-> reading_number int not null auto_increment,  
-> day int not null,  
-> month int not null,  
-> year int not null,  
-> time time not null,  
-> value int not null),  
-> primary key(reading_number));
```

4- Channel 3 Table

```
mysql> create table ch3(  
-> reading_number int not null auto_increment,  
-> day int not null,  
-> month int not null,  
-> year int not null,  
-> time time not null,  
-> value int not null),  
-> primary key(reading_number));
```

5- Channel 4 Table

```
mysql> create table ch4(  
-> reading_number int not null auto_increment,  
-> day int not null,  
-> month int not null,  
-> year int not null,  
-> time time not null,  
-> value int not null),  
-> primary key(reading_number));
```

6- Channel 5 Table

```
mysql> create table ch5(  
-> reading_number int not null auto_increment,  
-> day int not null,  
-> month int not null,  
-> year int not null,  
-> time time not null,  
-> value int not null),  
-> primary key(reading_number));
```

7- Channel 6 Table

```
mysql> create table ch6(  
-> reading_number int not null auto_increment,  
-> day int not null,  
-> month int not null,  
-> year int not null,  
-> time time not null,  
-> value int not null),  
-> primary key(reading_number));
```

8- Channel 7 Table

```
mysql> create table ch7(  
-> reading_number int not null auto_increment,  
-> day int not null,  
-> month int not null,  
-> year int not null,  
-> time time not null,  
-> value int not null),  
-> primary key(reading_number));
```

9- Channel 8 Table

```
mysql> create table ch8(  
-> reading_number int not null auto_increment,  
-> day int not null,  
-> month int not null,  
-> year int not null,  
-> time time not null,  
-> value int not null),  
-> primary key(reading_number));
```


10- Digital Channel's (1-8) Table

```
mysql> create table dig1(  
-> reading_number int not null auto_increment,  
-> day int not null,  
-> month int not null,  
-> year int not null,  
-> time time not null,  
-> value int not null),  
-> primary key(reading_number));
```

11- Digital Channel's (9-16) Table

```
mysql> create table dig2(  
-> reading_number int not null auto_increment,  
-> day int not null,  
-> month int not null,  
-> year int not null,  
-> time time not null,  
-> value int not null),  
-> primary key(reading_number));
```