A Personal Computer- Based Programmer For the 8051/8052 UV and Flash ROM Based microcontroller Family.

Monaf Sabri Tapou

Received on: 28/12/2005
Accepted on: 24 /4 /2008

Abstract

The 8051/8052 microcontroller family is very popular and is found in a huge number of applications in professional systems and amateur projects.

This paper describes the design of a programming system for the 8051/8052 UV and Flash ROM based microcontrollers that work in conjunction with an IBM style personal computer. The programmer uses the standard parallel printer port operating in bidirectional mode to interface with the host personal computer. An integral power supply and external step down power transformer provides the programmer with all required operating voltages.

The programmer is controlled by software written in Visual Basic running on the host personal computer. Two versions of the software were written; the first will run on personal computers that operate under Windows 95, windows 98, and windows millennium. The other will run under windows NT/2000, and windows XP. This was done because unlike Windows 95 & 98, Windows NT/2000/XP will cause an exception (Privileged Instruction) if an attempt is made to access a port that an application program is privileged to talk to and program execution is halted. This problem was solved by modifying the I/O permission bitmap in system Kernel Mode Driver to allow a particular task, access to certain I/O ports.

Hardware and software were designed to support most types of 8051/8052 microcontrollers made by known chip manufactures like Intel, AMD, Philips, Atmel, ST microelectronics, and Cygnal. Upon detecting chip type by reading signature byte, the proper programming voltage, timing, and algorithm suitable for a given chip are chosen automatically by software running on host. The programmer supports all memory microcontroller functions, including signature read, code read, code write, code verification, chip erase for microcontrollers equipped with flash memory, and lock bit write.

الخلاصة

ان عائلة المسيطر الدقيق 8051/8052 تعتبر من المسيطرات الأكثر شيوعا و استخداما في النطبيقات المحترفة و تطبيقات الهواة. هذا البحث يتناول تصميم منظومة برمجة للمسيطرات الدقيقة من عائلة 8051/8052 و التي تشغل من قبل حاسوب شخصي من طراز IBM. ان منظومة المبرمج تستخدم المنفذ التشعبي للطابعة (Parallel) Printer Port العامل بالنمط المزدوج الإتجاه (Bidirectional, PS/2, or EPP) للتحكم بعمل ممنظومة المبرمج . تم بناء منظومة المبرمج على لوح مطبوع مع دائرة تجهيز القدرة و يتم تغذيته من محول خارجي. يوفر مجهز القدرة كل فولتيات العمل المطلوبة لتشغيل المبرمج.المبرمج يتم التحكم به بواسطة برمجيات كتبت بلغة Visual Basic. تم كتابة برنامجين، الأول يعمل تحتانظمة التشغيل Windows 95, Windows 98, Windows Millennium و الثاني يعمل تحت انظمة التشغيل Windows NT/2000, Windows XP لأن انظمة التشغيل هذه لا تعطي لبرمجيات المستخدم حرية الوصول الى منافذ الإدخال و الإخراج و يجب تحوير جداول السماح في سجلات نظام التشغيل لكي تتيح تنفيذ اوامر الإدخال و الإخراج على منفذ معيّن.ان الكيان المادي و البرمجي للمنظومة يدعم استخدام معظم المسيطرات المنتجة من قبل الشركات المعروفة مثل Intel, AMD, Atmel, Philips و غيرها . و تدعم منظومة المبرمج كل فعّاليات الكتابة الى ذاكرة المسيطر الدقيق بضمنها قراءة التوقيع الخاص الذي يدل على نوع المسيطر، قراءة البرامج الداخلية و توكيدها و مسح ذاكرة المسيطر الذي يحتوي على ذاكرة من نوع فلاش و كذلك كتابة البايت التي تقفل ذاكرة المسيطر و لا تتيح قراءتها.

*Depts. Control and system Eng.Univ.of Tech.

General

Designing a programmer requires developing the knowledge of how each of the several types of microcontrollers the programmer will operate upon. Each type requires special signaling, timing, and programming voltages. The required information is gathered from data sheets supplied by chip manufacturers and vendors. And extensive survey was carried out to gather information about all the 8051 / 8052 family chip manufacturers. The search included text books, semiconductor data libraries, and manufacturer's web sites. The next step in designing the programmer is to investigate the interface system that is going to be used to control the programmer as it was decided to use the IBM compatible personal computer to control the programmer operation.

The parallel printer port found as a standard port on all personal computers was chosen to interface the programming system to the personal computer. Although most of modern printers have abandoned the parallel port and now are using the Universal Serial Bus "USB", the port still finds many useful applications and has several powerful features that make it an attractive choice for equipment designers. The choice of this port to interface to PC has a major influence on hardware and software design of the microcontroller programmer. Figure (1) shows the block diagram of the microcontroller programming system.

The hardware of the programmer was designed while keeping in mind minimizing component count, and utilizing the parallel port to the best of its capabilities. Circuit design underwent several refinements in order to improve performance and reduce component count. The Parallel printer port operating in bidirectional mode is chosen for the hardware realization. Its data port (8 bit wide) can operate as an input port or as an output port under program control. There are five input lines from the status port and another 4 output lines from the control port. These are the available I/O lines for the design. The programming of the internal programmable ROM of the 8051/8052 microcontroller family requires 8 I/O lines for data, 14 O/P lines for address, 4 O/P lines for programming mode select, and another 3 lines for programming voltage select, data routing, and programming strobe.

The data port of the parallel printer port is used to furnish data, address, and control for the programmer. Therefore four latches are used to accommodate for the data, address, and control signal generation, one octal tristate buffer for data verification and an 8 out of 3 decoder to generate data steering signals and other control signal required for programmer operation.

Figure (2) shows the interface circuit to host computer through the parallel printer port. The main components of the circuit are:

* Four 8bit Latches "74LS373"; these are used to hold the address, data, and control signals to the Micro controller that is to be programmed.
* One 8bit tristate buffer "74LS244", this is used to read the contents of the latest word written to the microcontroller internal EPROM in the verification cycle of the programming algorithm or during code read operation.
* One 8 out of 3 decoder "74LS138" in addition to one hex inverter "74LS04" which are used to generate the control signals that will direct data from parallel printer port to the four latches and enable the 8 bit tristate buffer during data verification cycle and microcontroller memory content read.

Figure (3) shows the power supply circuit of the programmer, it consists of three linear voltage regulators based on

the 78** series used to generate 13.4V, 5.7V "Vpp" and 5V supply to the entire programmer circuit the last was mounted on a proper heat sink with effective area of 10 cm^2 which will dissipate the generated heat and allows system operation in elevated environment temperatures. Transistor Q_3 is driven by Q_2 and is used to switch the power supply on and off under program control.

Figure (4) shows the programmer prototype that was assembled on a 160mm*100mm standard Euro card prototyping board that has accommodated all the electronics of the programmer except the power transformer and filter. A 40 pin ZIF socket is used to accommodate the microcontroller that is required to be programmed. The hardware connects to the host parallel printer port through a standard parallel printer cable.

The Software: **[2, 3, 6, 7]**

The software written is intended to provide user interface and perform the various tasks assigned to the microcontroller programming system.

The software was written using Visual Basic 6.0 environment. It was chosen due to familiarity and the popularity of this programming language, a second reason behind this choice is the flexibility of this language to create simple and efficient graphical user interfaces. But Visual Basic language does not include instructions to perform data input or data output to and from a specific port address. This feature is found in other programming languages like Delphi and Visual C. There are two ways to include this feature into the Visual Basic environment, the first is to install the MSDN package which allows Visual Basic to acquire the necessary program components to access I/O ports, the second is to write a program to access I/O ports using Delphi or Visual C and store it in DLL format then this DLL file is declared as a function in the Visual basic program [3]. This function handles all the communication between the host computer and the microcontroller programmer hardware. Below is the source code for this program written in Delphi.

```
Library inpout32;
uses SysUtils;
Procedure Out32
(PortAddress:smallint;Value:smallint);stdc
all;export;
var
  ByteValue:Byte;
begin
    ByteValue:=Byte(Value);
    asm
      push dx
      mov dx,PortAddress
      mov al, ByteValue
      out dx,al
      pop dx
    end;
end;

function
Inp32(PortAddress:smallint):smallint;stdc
all;export;
var
  ByteValue:byte;
begin
  asm
      push dx
      mov dx, PortAddress
      in al,dx
      mov ByteValue,al
      pop dx
  end;
  Inp32:=smallint(ByteValue) and $00FF;
end;
Exports
    Inp32,
    Out32;
begin
end.
```

The main program running on personal computer performs all the functions of the microcontroller programming system.

When the software is executed, it starts by checking the operating mode of the parallel printer port of the computer on which it is running, if the mode is other than the bidirectional mode, a screen is displayed to ask program user to change its mode to the desired one then rerun the programmer software, otherwise it starts by displaying the main window of the software which has the form shown in figure (5).

The programmer software accepts files created by popular assemblers for the 8051 / 8052 microcontroller family in standard Intel Hex format. This object file format is supported by many cross assemblers, utilities and most EPROM programming systems.

The software can be divided into two sections;

A: User Interface Section

The user interface section deals with designing the graphical user interface and the operations in response to user input. This main window contains four sub menus (file, edit, processing, and help menus).

1-File menu: this menu contains ***new, open, save***, and ***save as*** tools for opened file management, and contains ***print*** to print opened file, which will appear in the Edit area after executing the opening operation like any program working under windows environment. This menu appears in figure (6).

2-Edit menu : this menu contains ***cut ,copy , paste , find , find record , increase font size*** ,and ***decrease font size*** , for editing files written in Intel hex format as shown in figure (7) ,as well as the ***analysis function*** which analyzes the opened file and displays the result in a special window as shown in figure (8) .

3-Processing menu: this menu contains the main functions of the software, which are ***Burning in,*** *(writing data to microcontrollers ROM),* ***chip Erase,*** *Read,* and the rest of the programmer functions as shown in figure (9).

4-Help menu: this menu provides software user with explanations on the programming system functions and how to use the system to program a microcontroller. It also includes System features and the range of microcontrollers that can be used.

B: Programmer Operation control Section

This section of the program performs the actual tasks and functions of the microcontroller programming system. For example writing data to microcontroller ROM starts by opening the file the user wants to burn its contents into microcontrollers ROM it and then send the address location to which data are written to address latches (latch1 & latch2) by executing these instructions:

Out &H37A , 11-------- -→ Latch1 Enable
Out &H378 , low address→ fill latch 1 with low Address
Out &H37A ,10 ----------→ Latch2 Enable
Out &H378, high address→ fill Latch 2 with high Address

After executing these instructions the desired program data to be sent to microcontroller is put in the data latch (LATCH 3) by executing these instructions.

Out &H37A , 9 ----------→ Latch3 Enable
Out &H378, Data to be sent→ fill Latch 3 with Data byte

Then the appropriate control and program signals are sent to the microcontroller to set it up for data writing process.

This is done by writing the proper control word into the control latch (LATCH4). The following instructions execute the writing operation.

Out 37A, 3----------→ Latch4 Enable
Out 378, Control signals-→ fill Latch 4 with control word

The choice of timing of the programming operation is dependent on Chip type for example for Intel type microcontrollers the programming operation is not self timed and the process of programming should be timed using software timing in accordance with the chip timing and repeat programming requirement. In Atmel 89C51 the programming operation is self timed, and it should be noted that the (ALE/-PROG) signal is removed upon receiving a logic high signal from the progress indicator bit found at P3.4 which is set low when ALE is pulled high during programming.

The above mentioned steps are repeated until all the data in the program file are written to the microcontroller flash ROM.

Operating the Programmer under Windows 2000 or Windows Xp

The programmer system software was originally written to operate under Windows 98, Windows 98 SE, and Windows me. And there was no evident problem in software execution under these operating systems, but when software was executed under Windows NT/2000 and Windows XP it caused an application error and the program stopped executing the moment an I/O operation is executed. In Windows NT the application error message indicates the exception privileged instruction. In Windows XP it only indicates problem encountered.

The reason behind this error is that Windows NT/2000 and Windows XP have strict control over I/O ports. Unlike Windows 98& me, Windows NT/2000/XP will cause an exception error (Privileged Instruction) if an attempt is made to access a port that user programs are not privileged to talk to and will display the message box shown in figure (10) for Windows NT, and figure (11) for Windows XP.

Accessing I/O Ports in protected mode is governed by two events, The I/O privilege level (IOPL) in the EFLAGS register and the I/O permission bit map of a Task State Segment (TSS) [2].

Under Windows NT, there are only two I/O privilege levels used, level 0 & level 3. User mode programs will run in privilege level 3, while device drivers and the kernel will run in privilege level 0, commonly referred to as ring 0. This allows the operating system and drivers running in kernel mode to access the ports, while preventing user mode processes from manipulating the I/O ports and causing conflicts. All user mode programs should talk to a device driver which arbitrates access.

The I/O permission bitmap can be modified to allow user mode programs the ability to access the I/O ports. When an I/O instruction is executed, the processor will first check if the task is privileged to access the ports. Should this be the case, the I/O instruction will be executed. However if the task is not allowed to do I/O, the processor will then check the I/O permission bitmap.

The I/O permission bitmap, as the name suggests uses a single bit to represent each I/O address. If the bit corresponding to a port is set, then the instruction will generate an exception. However, if the bit is clear then the I/O operation will proceed. This gives a means to allow certain processes to access certain ports. There is one I/O permission bitmap per task. There are two solutions to solving the problem of I/O access under Windows NT. The first solution is to write a device driver which runs in ring 0 (I/O privilege level 0) to access the required I/O ports, then data can be passed to and from user mode program to the device driver via IOCTL calls. The driver can then execute I/O instructions.

Another possible alternative is to modify the I/O permission bitmap to allow a particular task, access to certain I/O ports. This grants user mode program running in I/O privilege level 3 to do unrestricted I/O

operations on selected ports, per the I/O permission bitmap. So for the programmer software to operate under Windows XP or NT a modified version is presented where it included a special executable that was written to alter I/O permission map and allow the microcontroller programming software to operate under Windows XP or NT.

Software was written with full remarks that will aid the future developers to enhance the program by including additional functions.

Conclusions and Results

The designed system was built on a prototyping board because several changes to the early design were made during the course of testing and function verifying process which took several tests followed by software/hardware modifications to overcome the encountered malfunctions until the final form discussed earlier has been reached. The system in its final form has been tested using ***Atmel 89C52, Atmel 89C51, Intel 87C51,*** and ***AMD*** 87C51 microcontrollers alternately, then over more than 200 different programming / verification operations were carried out successfully and no data error cases were encountered.

The programming system was used to program microcontrollers implemented in several accomplished projects that have successfully functioned.

When compared with the Atmel proposed programmer, it can be seen that the designed system had additional operational features and design enhancements.

The designed system can program a larger variety of microcontrollers other than those provided by Atmel.

An additional safety feature of automatically powering the programmer on/off while changing the microcontroller that is required to be programmed which will protect CMOS microcontrollers from damaging effects of transients, current surges, and static discharges encountered when inserting semiconductor devices while circuit is powered.

The chosen semiconductor devices are cheaper and more popular that those found in the design proposed by Atmel.

Finally the programming software provided by Atmel works in DOS mode only and its user interface is not friendly.

The data port of the parallel printer port is connected directly to control latches, while only the verify tri-state buffer is connect through 8 ballast resistors to protect the parallel printer port from damage through incorrect direction control. This approach provides higher signal levels at the latches inputs which enables error free operation of the programming system and makes it possible to program microcontrollers at a higher rate (when possible depending on Microcontroller type) resulting in shorter programming time.

The designed system has plenty of room for future enhancements and modifications to accommodate chip formats other than the popular DIP40 by preparing adapters that fit in the ZIF40 socket to accept other chip carrier formats. It can also be further modified to accept programmable microcontrollers other than those from 8051 / 8052 family.

References

1- Author: C.J.Savant, Jr.
Martin S.roden
Gordan L.Carpenter
Book Title: Electronic Circuit Design
Publisher: The Benjamin/Cummings Publishing Company Inc.
Publication year: 1991
ISBN: 0-8053-0292-1

2- Author: Ed Bott, Carl Siechert, and Craig Stinson
Book Title: Microsoft Windows XP Inside Out
Publisher: Microsoft Press
Publication year: 2001

ISBN: 0-7356-1382-6

3- Author: Jan Axelson
Book Title: Parallel Port Complete
Publisher: Penram International Publishing (India) Pvt. Ltd.
Publication year: 1996
ISBN: 81-87972-02-5

4- Author: Steven J. Faulkenberry
Book Title: Introduction to Operational Amplifiers with linear circuit applications
Publisher: John-Wiley Inc.
Publication year: 1986
ISBN: 0-09-32346-1

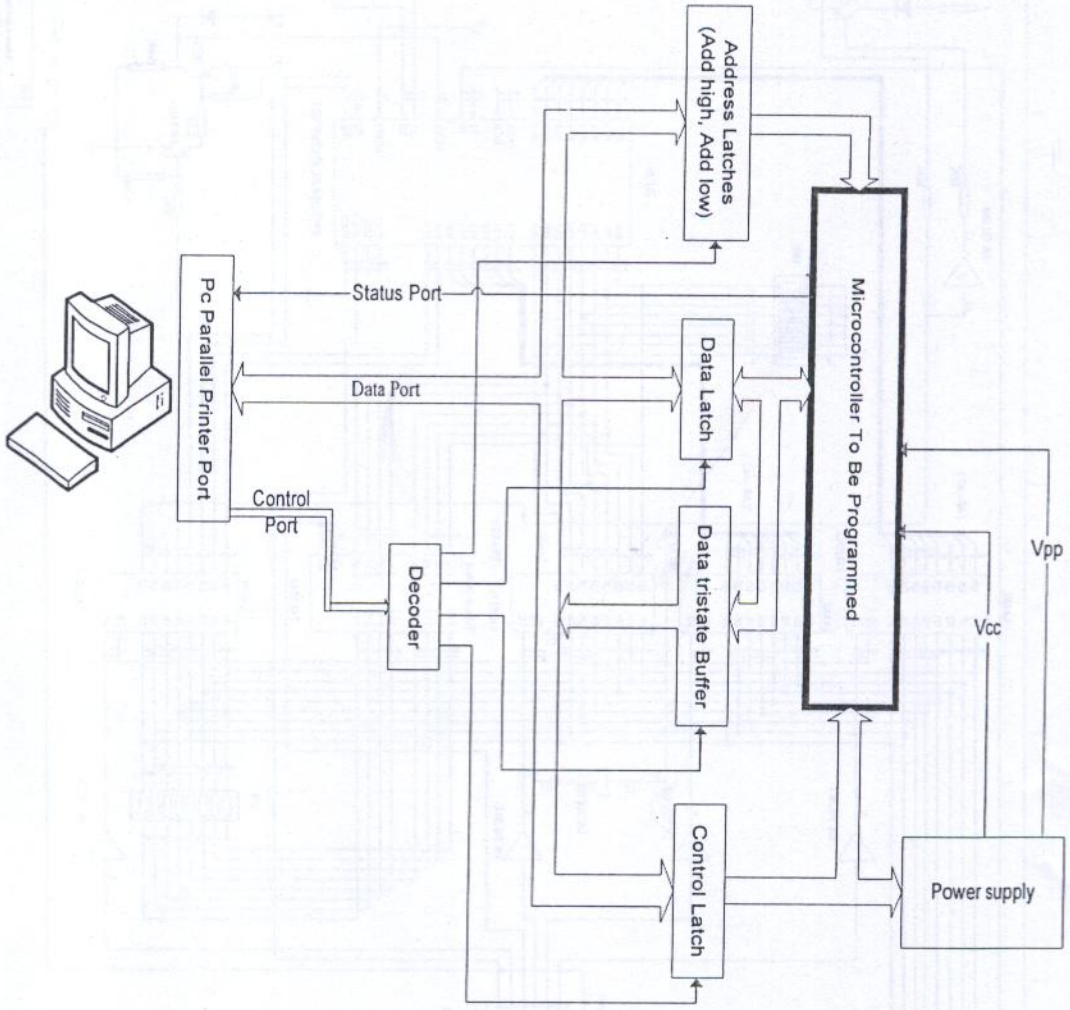

Figure (1) Block Diagram of the Microcontorller Programming System

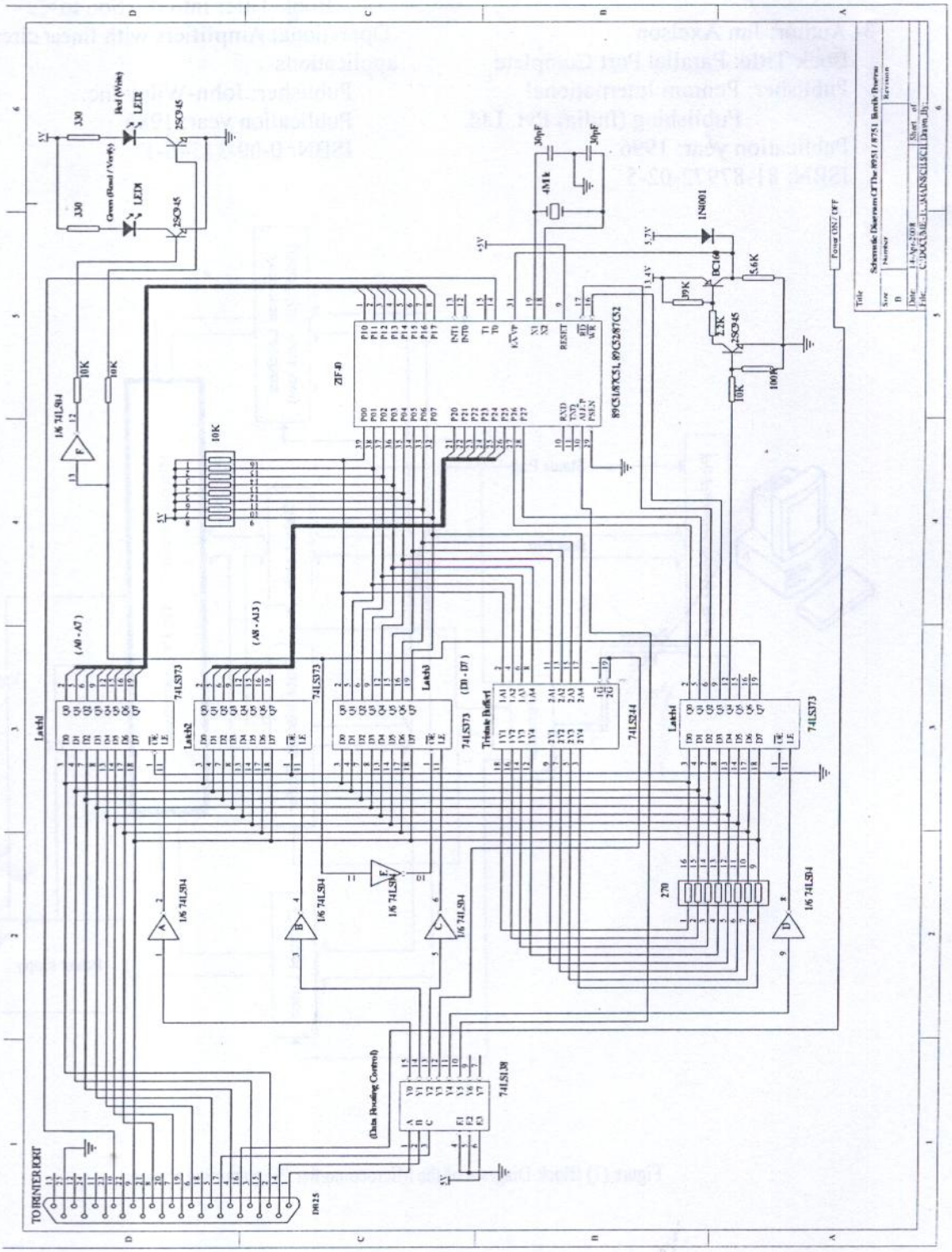

Figure (2) The Main Circuit Diagram of the Microcontroller Programming System

Figure (2) The Main Circuit Diagram of the Microcontroller Programming System

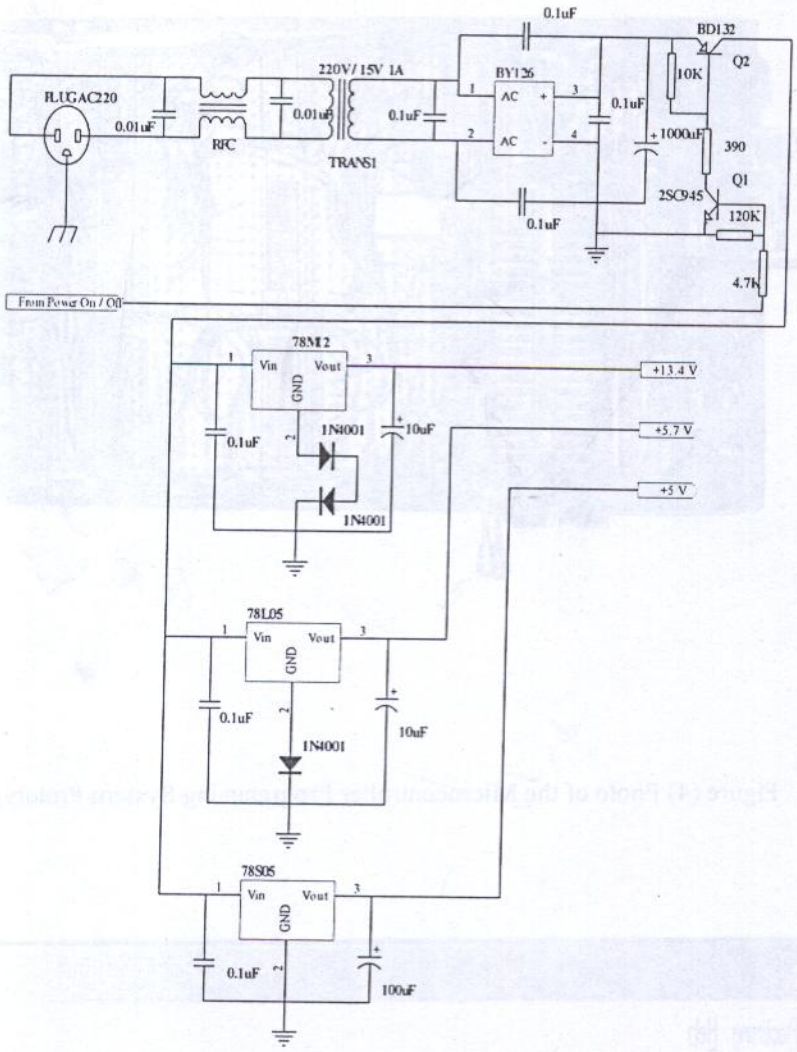

Figure (3) Schematic Diagram of the Power Supply Section of the Microcontroller Programming System

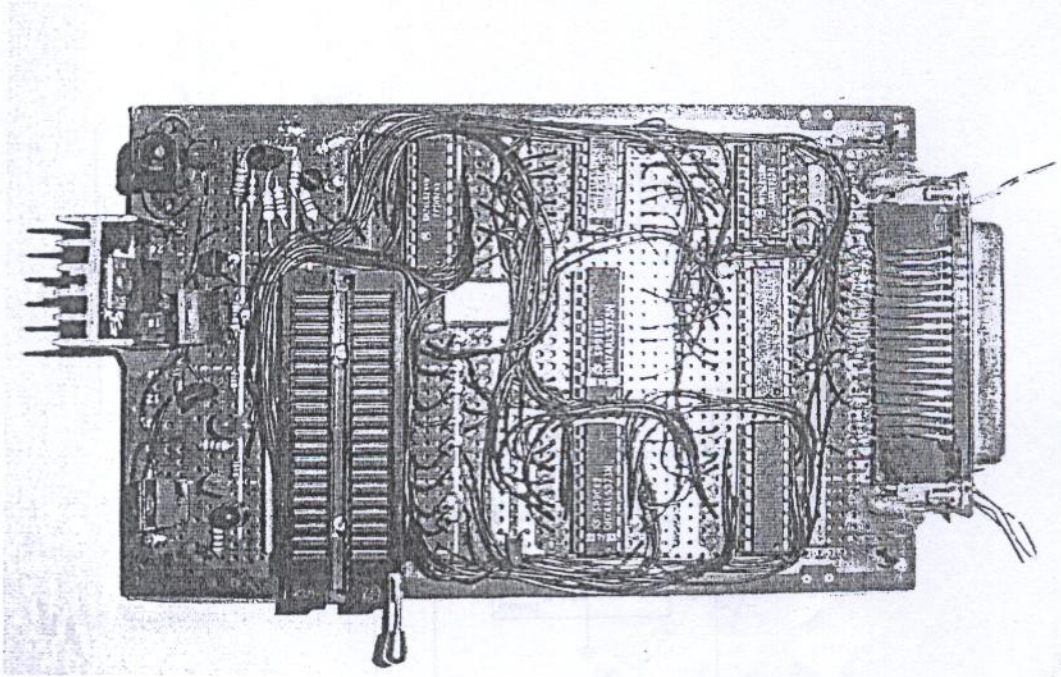

Figure (4) Photo of the Microcontroller Programming System Prototype

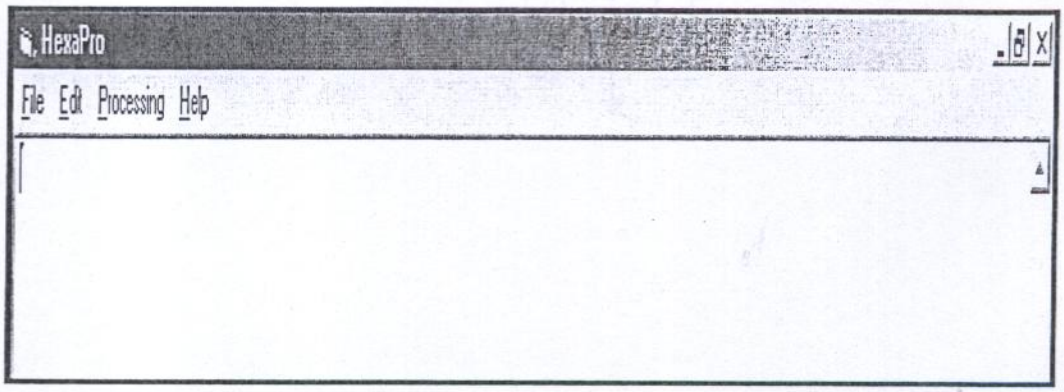

Figure (5) The Main Window

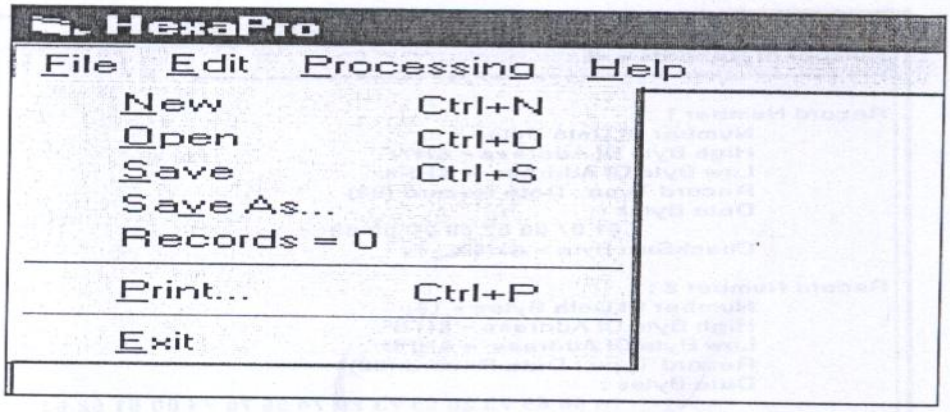

Figure (6) The File Menu Contents.

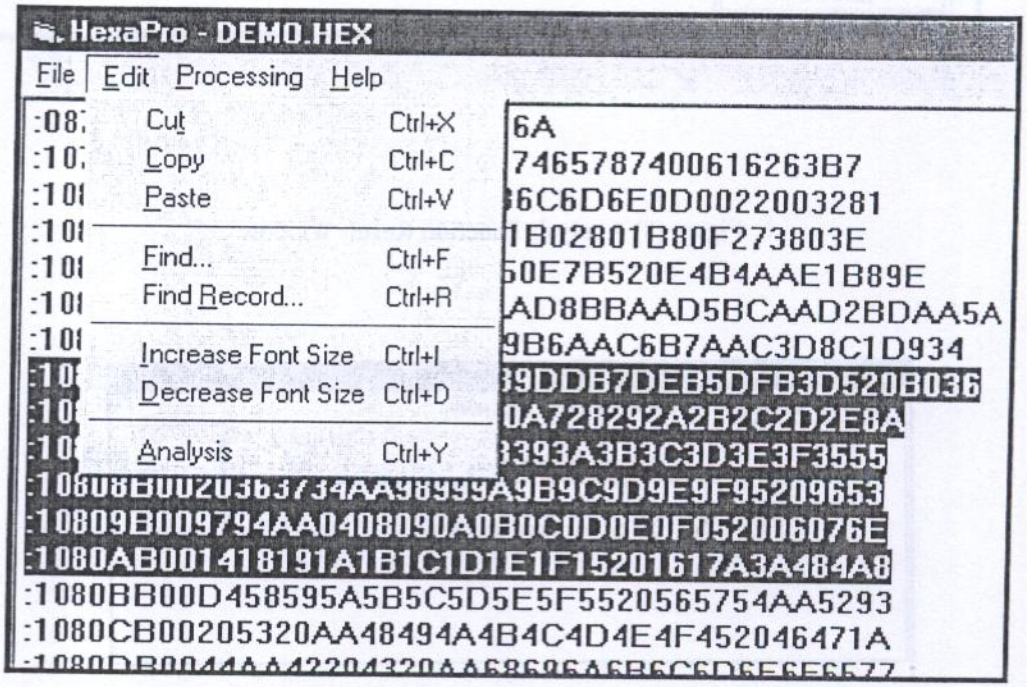

Figure (7) Edit Menu Contents.

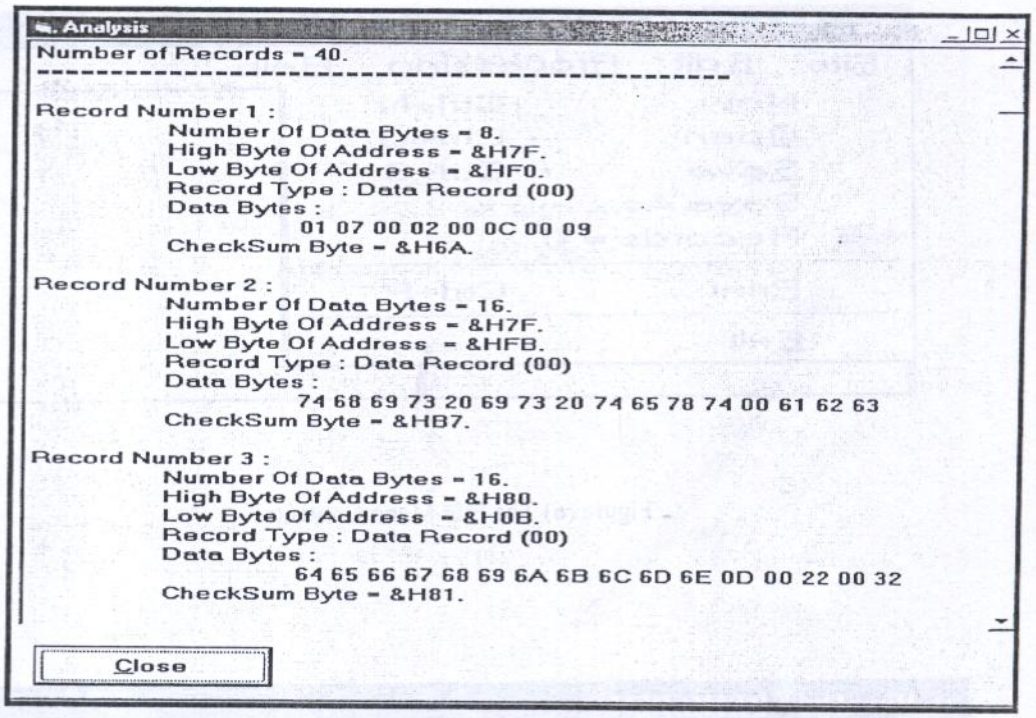

Figure (8) Analysis Function Result Window

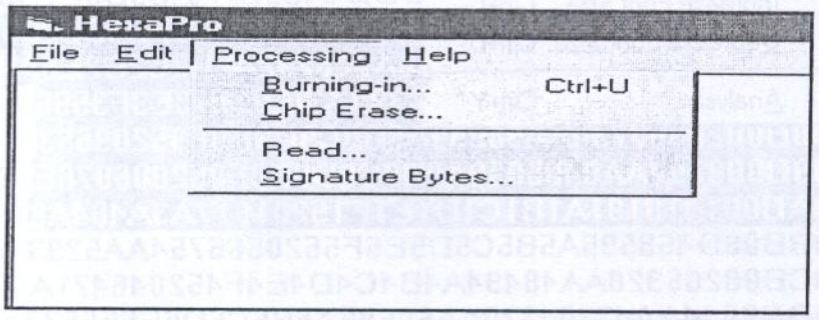

Figure (9a) Processing Menu.

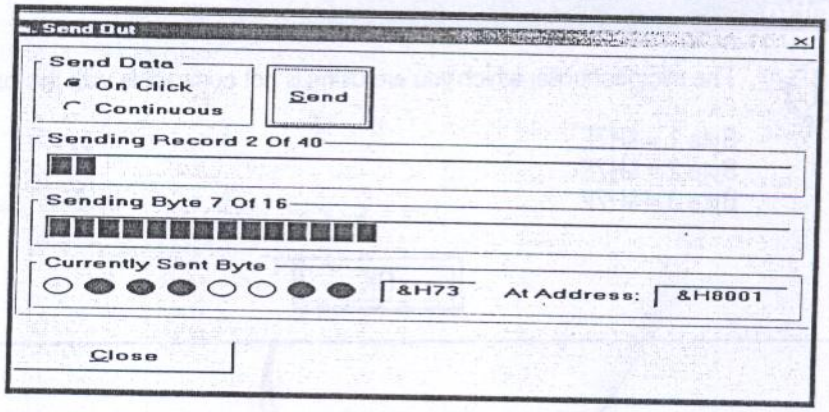

Figure (9b) Send Data And Programming Progress Monitoring .

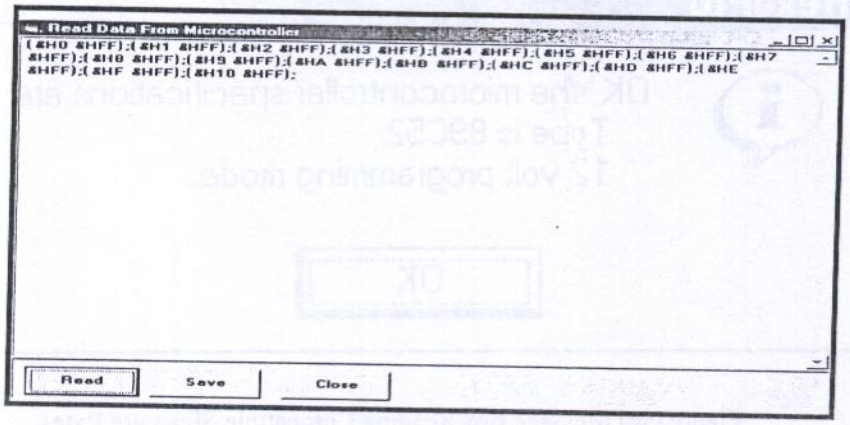

Figure (9c) Read Data From Chip.

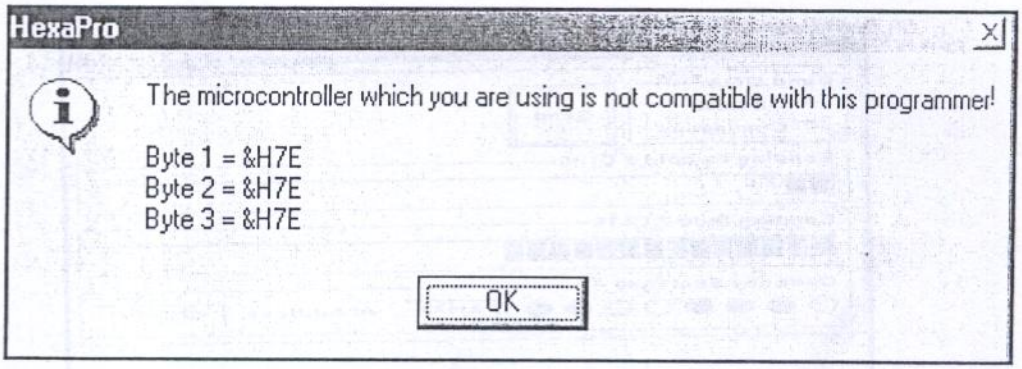

Figure (9d) Message Box Reading A Wrong Signature Bytes.

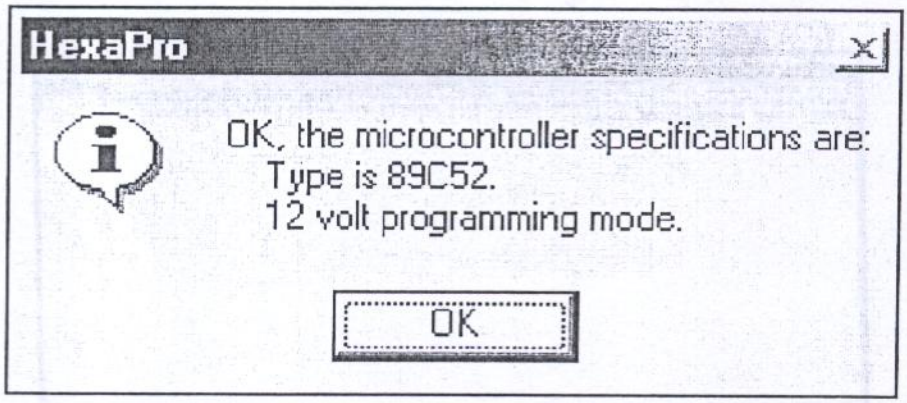

Figure (9e) Message Box Reading Compatible Signature Bytes.

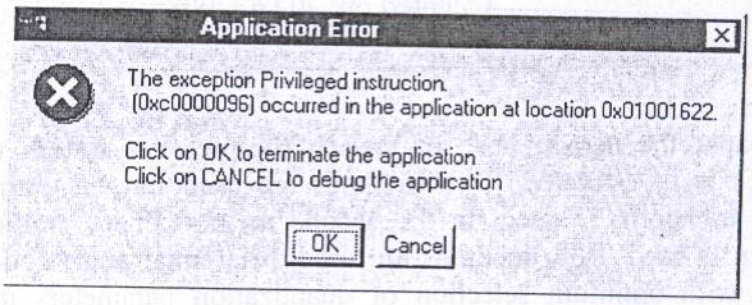

Figure (10) Message Box Displayed After the Execution of a Privileged Instruction on Windows NT.

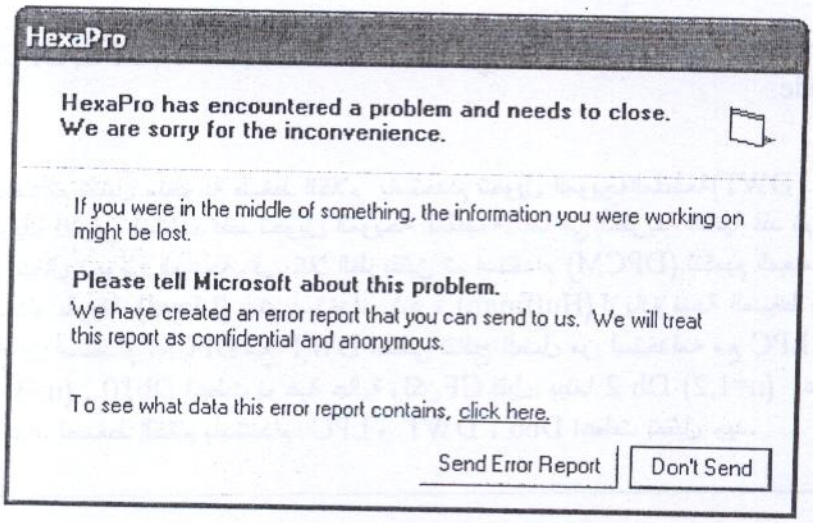

Figure (11) Message Box Displayed After the Execution of a Privileged Instruction on Windows XP.