British Journal of Applied Science & Technology
17(2): 1-19, 2016, Article no.BJAST.25046
ISSN: 2231-0843, NLM ID: 101664541

SCIENCEDOMAIN *international*
www.sciencedomain.org

Microcomputer Based Multipoint Time Operated Power Switching System (With Overload Protection)

Chukwuedozie N. Ezema[1*], Okechi Onuoha[2], Nwanyinnaya Nwogu[2], Albert C. Agulanna[2] and Helen U. Nonyelu[3]

[1]Nnamdi Azikiwe University, Awka, Anambra State, Nigeria.
[2]Projects Development Institute (PRODA), Enugu State, Nigeria.
[3]Scientific Equipment Development Institute (SEDI), Enugu State, Nigeria.

Authors' contributions

The authors acknowledged that the manuscript submitted is their own original work. All authors participated in the work in a substantive way and are prepared to take public responsibility for the work. The submitted manuscript is reviewed and approved by all authors.

Article Information

DOI: 10.9734/BJAST/2016/25046
Editor(s):
(1) Samir Kumar Bandyopadhyay, Department of Computer Science and Engineering, University of Calcutta, India.
Reviewers:
(1) Kexin Zhao, University of Florida, Gainesville, USA.
(2) Rajen Pudur, National Institute of Technology, Arunachal Pradesh, India.
Complete Peer review History: http://www.sciencedomain.org/review-history/15792

Original Research Article

Received 15th February 2016
Accepted 15th April 2016
Published 14th August 2016

ABSTRACT

The objective of this research is to design a "Microcomputer Based Multipoint Time Operated Power Switching System" (with overload protection). The system is made up of eight relays that can be controlled independently through the parallel port of a computer using visual basic software. It is a time operate electrical appliance controlling system and is a reliable circuit that takes over the task of switching the electrical devices ON/OFF with respect to time. Once is set and device activated, the device remain on till the set time elapses. With the aid of software devices on/off indicate the set time perimeter (hr. Min: sec) and the corresponding real time clock are displayed at the front of the panel. It can be time event, intervals of time and can definitely control AC appliances, it will trigger a relay when it has timed down. The system makes use of a real time check built around the computer system clock and keeps track of the time. When this time is equals to the programmed off time of a device, then the corresponding relay for the devices is switched off. The switch time can be edited using the peripherals devices like the keyboard and mouse.

**Corresponding author: E-mail: ecnaxel@gmail.com;*

Keywords: Microcomputer; multipoint time; power switching system; overload protection.

1. INTRODUCTION

The subject of time and the traditions surrounding it is sometimes taken for granted, especially in Africa where we do things and say "African Time". But have we ever considered an industrial production process where events done in time, were neglected for once or to heat a delicate substance for a specified period of time was over looked; losses may amount. Even our day to day activities is guided by time, time to sleep, time to wake, time to eat etcetera. Hardly anything is done without time. Time can be set from a few seconds to several hours converted in seconds as one chooses. Microcomputer Based Multipoint Time Operated Power Switching System" (with overload protection) can time events, intervals of time and can definitely control A.C. appliances; it will trigger a relay when it has timed down [1,2]. It may be used for dark room or PC board exposure timer, exit room timer. It can be used in laboratories, kitchens, washers, dishwashers, driers, and for competitions in educational institutes, and can power any electrical appliance rated below 1000 Watts or more. Also other parameters or features implemented in the research are the overload protection, voltage level display which is measured using a voltage monitor and an Analog to digital converter (ADC0804). The research can find uses in several applications [3].

This research is justified on the basis of the purpose and importance of Timed Control and Power regulation in various aspects. Timers are in homes, Laboratories, Industries, Sports arena, Gadgets or devices and Schools, and even in Offices. The main purpose of a Timer is not always only to keep interval of preset time but can also be used to control devices according the preset time, e.g. an alarm clock, a VCR, an egg timer or a time bomb [4].

Practically all computers depend on an accurate internal clock signal to allow synchronized processing [5]. A few researches are developing CPUs based on asynchronous circuits. Some computers also maintain time and date for all manner of operations whether they are for alarms, event initiation or just to display the time of day.

This simple and flexible timer is accurate like the real-time clock of the computer used for the purpose [6]. The timer is software based; the program written in Visual Basic is self-explanatory. In this research, "Microcomputer Based Multipoint Time Operated Power Switching System", exposition will be done on how application software can be used to implement Timer and Control functions on a Microcomputer [7].

1.1 The Timer

A Timer is a specialized type of clock. A timer can be used to control the sequence of an event or process [5,8]. Examples include: Mechanical timers, Electromechanical Timers, Digital Timers and Computer Timers.

1.1.1 Parallel port basics

In computers, ports are used mainly for two reasons: Device control and communication [9, 10]. We can program PC's parallel ports for both. Parallel ports are mainly meant for connecting the printer to the PC. But we can program this port for many more applications beyond that.

Parallel ports are easy to program and faster compared to the serial ports. But main disadvantage is it needs more number of transmission lines. Because of this reason parallel ports are not used in long distance communications [11]. Let us know the basic difference between working of parallel port and serial port. In serial ports, there will be two data lines: One transmission and one receive line. To send a data in serial port, it has to be sent one bit after another with some extra bits like start bit, stop bit and parity bit to detect errors [12]. But in parallel port, all the 8 bits of a byte will be sent to the port at a time and an indication will be sent in another line. There will be some data lines, some control and some handshaking lines in parallel port [13].

2. SYSTEM ANALYSIS

2.1 Design Approach

Each individual module was designed and constructed separately. After successful simulation and testing, they were put together to create the finalized version.

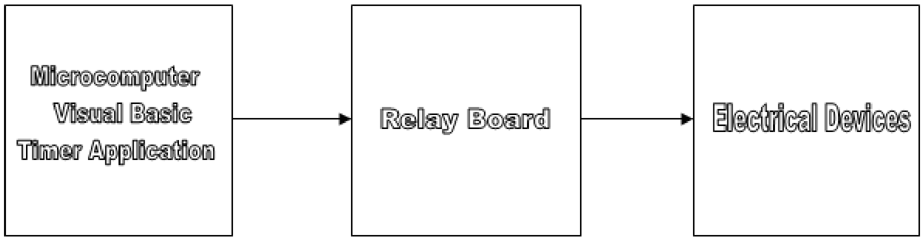
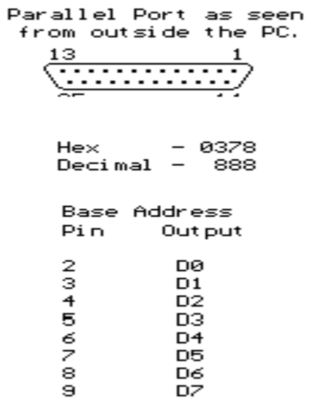


Fig. 1. Block diagram overview of the research

2.2 Software Development

There are different types of interfaces depending on the application, but they are all used to convert a program so that an electrical or a robotic device can be controlled. Interfaces can be connected to any computer because they all do the same job [14,15]. A typical example is a relay board. A 'Relay Board is a Smart Box', a common interface and they can be connected to most types of computer with parallel port. It is possible to connect up to eight devices such as motors temperature sensors, movement sensors; light sensors and data acquisition etc.



```
Hex      - 0378
Decimal  -  888

Base Address
Pin     Output

2        D0
3        D1
4        D2
5        D3
6        D4
7        D5
8        D6
9        D7
```

Fig. 2. Typical parallel port interface

Digital outputs are items such as speakers, lights, buzzers, LEDs and circuits etc. Digital inputs are devices such as micro-switches and relays. They are either 'on' or 'off'.

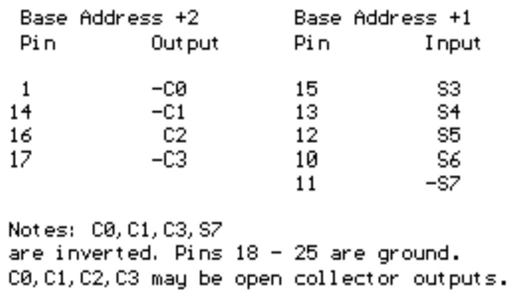
2.3 Out Putting Data to the Parallel Port: Relay Interface

To output data to a parallel port is simple and straight forward. All that is needed is a 25 pin D connector, a relay board that comprises of Resistor to limit current, Transistors or relay driver modules like (ULN2803) and 5V power supply. Then locate the parallel port data lines (D0-D7) which are pins (2-9) and the port ground pins (18-25); then attach the board.

Using a typical Visual basic code - Port out (port address, value), example Port Out (888, H01); this will energize relay at D0 (Pin2) particularly. This relay will have a Logic 1or High (5V) data written to the data line. Subsequently a logic zero (0) written to the data line well de-energize the relay.

2.4 Reading Data from the Parallel Port

Parallel port status and control lines for reading data from the parallel port is explained in the figure below.



```
 Base Address +2          Base Address +1
 Pin        Output        Pin        Input

  1          -C0          15           S3
 14          -C1          13           S4
 16           C2          12           S5
 17          -C3          10           S6
                          11          -S7

Notes: C0,C1,C3,S7
are inverted. Pins 18 - 25 are ground.
C0,C1,C2,C3 may be open collector outputs.
```

Fig. 3. Parallel port status and control lines

A standard parallel port has only 5 input lines, while some support bi-directional mode [16-18]. To do this you can use the 5 input lines of the Status Port and the 3 inputs (open collector) lines of the Control Port. The inputs to the Parallel Port have been chosen as such, to make life easier for us. Busy just happens to be the MSB (Bit 7) of the Status Port, then in ascending order comes, Ack, Paper Out and Select, making up the most significant nibble. The Control port is used to read the least significant nibble.

However to avoid conflicts and non-compatibility, the Nibble mode is the preferred way of reading 8 bits of data without placing the port in reverse mode and using the data lines [19,20]. Nibble mode uses octal buffers and line drivers with 3-state outputs (74ls244) or a Quad 2 line to 1 line multiplexer (74LS157) to read a nibble of data at a time. Then it "switches" to the other nibble and reads it. Software can then be used to construct the two nibbles into a byte. The only disadvantage of this technique is that it is slower. It now requires a few I/O instructions to read the one byte, and it requires the use of an external IC.

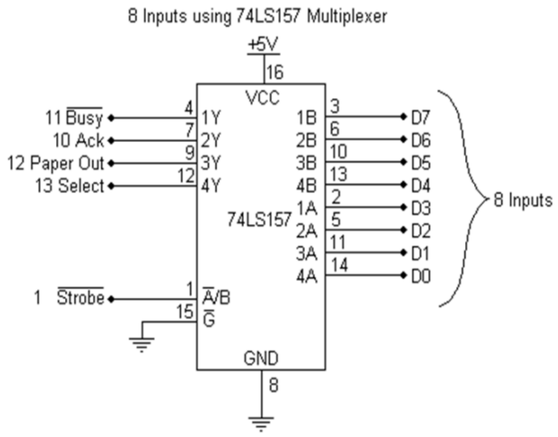


Fig. 4. Input interface

The diagram above shows 4 input lines of the parallel port. The operation of the 74LS157, Quad 2 line to 1 line multiplexer is quite simple. It simply acts as four switches. When the A/B input is low, the A inputs are selected. E.g. 1A passes through to 1Y; 2A passes through to 2Y etc. When the A/B is high, the B inputs are selected. The Y outputs are connected up to the Parallel Port's status port, in such a manner that it represents the MSnibble of the status register. While this is not necessary, it makes the software easier.

To use this circuit, first we must initialize the multiplexer to switch either inputs A or B. We will read the LSnibble first, thus we must place A/B low. The strobe is hardware inverted, thus we must set Bit 0 of the control port to get a low on Pin 1.

```
PortOut (Address, Value); /* Select Low Nibble */
```

Once the low nibble is selected, we can read the LSnibble from the Status Port. Take note that the Busy Line is inverted; however we won't tackle it just yet. We are only interested in the MSnibble of the result, thus we add the result with 0xF0, to clear the LSnibble.

```
a = (PortIn (Address) and &HF0); /* Read Low Nibble */
```

Now it's time to shift the nibble we have just read to the LSnibble of variable a,

```
a = a >> 4; /* Shift Right 4 Bits */
```

But Visual Basic does not have the above function but we found another way to maneuver this shift instruction. Don’t forget the busy line is always toggled. We are now half way there. It's time to get the MSnibble, thus we must switch the multiplexer to select inputs B. Then we can read the MSnibble and put the two nibbles together to make a byte,

```
PortOut (address, Value And & HFE); /* Select High Nibble (B)*/

a = PortIn (address,) and &HF0); /* Read High Nibble */
```

Note the LPT1 input port address decimal ‘889’ and the Control lines address decimal ‘890’. The state of the 4 lines is received as a single 8 bit number between 0-255 which is stored as the value of (V). Each switch input represents a decimal value of 16,32,64 and 128 which correspond to pins 13,12,10 and 11. The last 4 bits (1, 2, 4 and 8) are not used and should return a high level.

2.5 Voltage Measurement and Calibration

This is done using the ADC0804 input channels (D 0-D1). Here to set the voltage, the potentiometer VR. Adj is connected to +9V half wave rectifier through the 5V zener diode. For the purpose of testing, you can vary VR1 to adjust the voltage from 0 to 5V.

For this, transformer half wave rectified supply provides a proportional voltage to the ADC chip. The VR. Adj. point gives a voltage that varies with mains voltage. At exactly 220V mains, the 9V transformer gives a peak voltage of approximately, 9V - Diode drop (0.7V) = 8.3V. At point V_R Adj the value is 2.3 V. It increases approximately to 5V when the mains voltage rises to 259V and drops to zero when mains voltage drops to 172V in effect, giving 0 to 5V over this range. The Value of VR1 is in such a manner that the output Voltage is never greater than 5.0V.

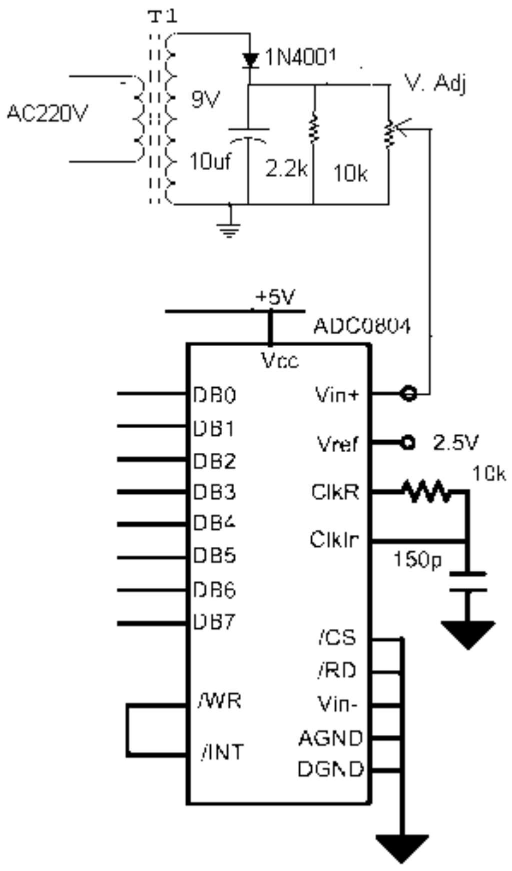


Fig. 5. Voltage measurement

With this arrangement the high voltage and low voltage protection can be implemented. Likewise the overload protection function is also implemented and it arises when there is low voltage the component being power will tend to draw more current causing an overload [21,22]. However this is prevented, by detecting the low voltage and high voltage situations. Also when a short circuit happens, the current increases drastically and voltage drops drastically this situation is also detected by low voltage function.

2.6 Voltage Calibration

The input voltage from normal AC to worst case conditions 259V can be adjusted from 0 to 5V because the ADC can only measure voltages between 0 - 5V and represent the values measured as a 8 bit binary number from 0 - 255.

At AC (259V) = Vmax dc (15V) = V input max ADC (5V)

At AC (220V) = Vmax dc (9V) = V input ADC (2.3V)

In order to determine the voltage increments which can be measured one has to divide the scaled input voltage by 255 and that equals: 15V/255 = 58.8 mV.approx 59 mV.

This means that at every count of ADC voltage increases by 59mV. With this idea the AC voltage can easily be measured.

2.7 Functional Description

The ADC0804 contains a circuit equivalent of the 256R network [23]. Analog switches are sequenced by successive approximation logic to match the analog difference input voltage [V in (+) − V in (−)] to a corresponding tap on the R - network. The most significant bit is tested first and after 8 comparisons (64 clock cycles) a digital 8-bit binary code (11111111 = full-scale) is transferred to an output latch and then an interrupt is asserted (INTR makes a high-to-low transition). A conversion in process can be interrupted by issuing a second start command [16]. The device may be operated in the free-running mode by connecting INTR to the WR input with CS =0. To ensure start-up under all possible conditions, an external WR pulse is required during the first power-up cycle.

On the high-to-low transition of the WR input the internal SAR (Successive Approximation Register) latches and the shift register stages are reset. As long as the CS input and WR input remain low, the A/D will remain in a reset state. Conversion will start from 1 to 8 clock periods after at least one of these inputs makes a low-to-high transition.

A functional diagram of the A/D converter is shown in Fig. 6. All of the package pin outs are shown and the major logic control paths are drawn in heavier weight lines. The converter is started by having CS and WR simultaneously low. This sets the start flip-flop (F/F) and the resulting “1” level resets the 8-bit shift register, resets the Interrupt (INTR) F/F and inputs a “1” to the D flop, F/F1, which is at the input end of the 8-bit shift register. Internal clock signals then transfer this “1” to the Q output of F/F1. The AND gate, G1, combines this “1” output with a clock signal to provide a reset signal to the start F/F. If the set signal is no longer present (either WR or CS is a “1”) the start F/F is reset and the 8-bit shift register then can have the “1” clocked in, which starts the conversion process. If the set signal were to still be present, this reset pulse would have no effect (both outputs of the start F/F would momentarily be at a “1” level) and the 8-bit shift register would continue to be held in the reset mode. This logic therefore allows for wide CS and WR signals and the converter will start after at least one of these signals returns high and the internal clocks again provide a reset signal for the start F/F.

After the “1” is clocked through the 8-bit shift register (which completes the SAR search) it appears as the input to the D-type latch, LATCH 1. As soon as this “1” is output from the shift register, the AND gate, G2, causes the new digital word to transfer to the TRI-STATE output latches. When LATCH 1 is subsequently enabled, the Q output makes a high-to-low transition which causes the INTR F/F to set. An inverting buffer then supplies the INTR input signal.

Note that this SET control of the INTR F/F remains low for 8 of the external clock periods (as the internal clocks run at 1⁄8 of the frequency of the external clock). If the data output is continuously enabled (CS and RD both held low), the INTR output will still signal the end of conversion (by a high-to-low transition), because the SET input can control the Q output of the

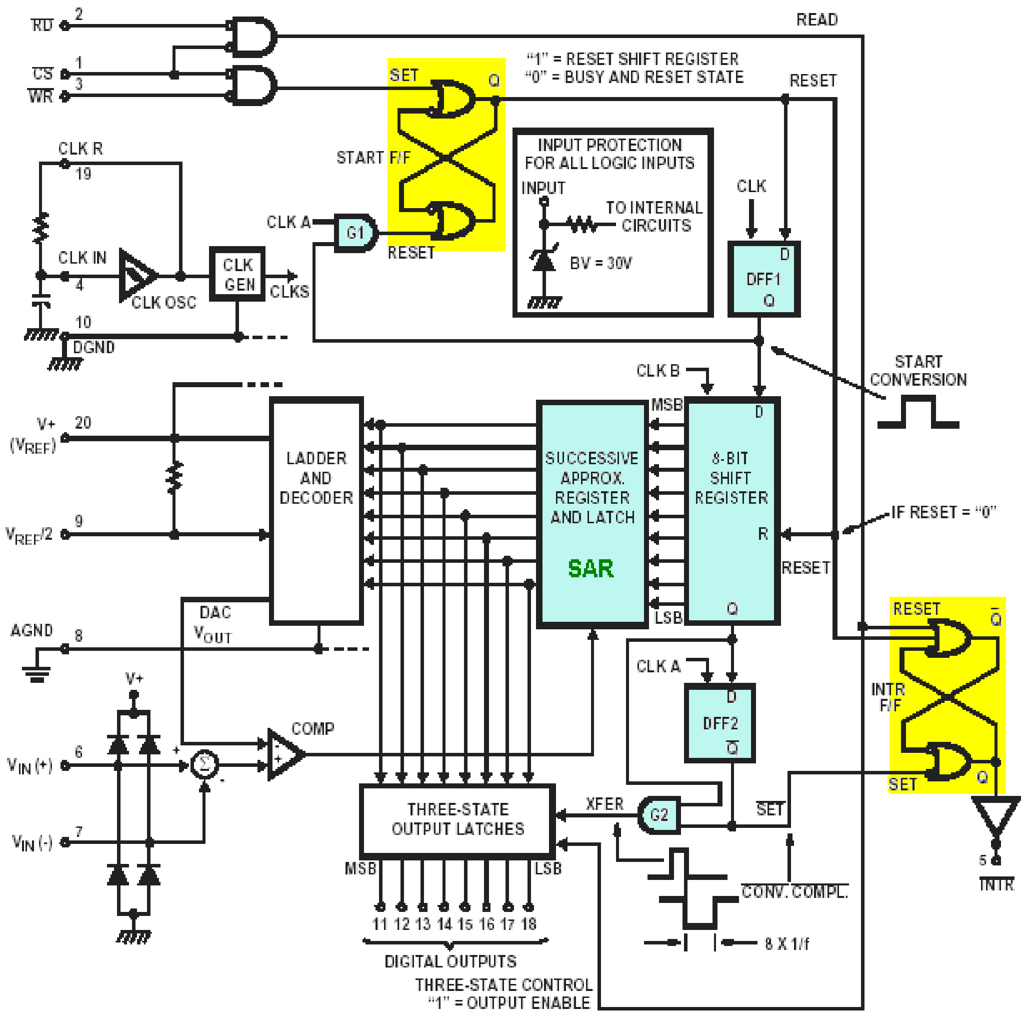


Fig. 6. ADC function diagram

INTR F/F even though the RESET input is constantly at a “1” level in this operating mode. This INTR output will therefore stay low for the duration of the SET signal, which is 8 periods of the external clock frequency (assuming the A/D is not started during this interval).

When operating in the free-running or continuous conversion mode (INTR pin tied to WR and CS wired low), the START F/F is SET by the high-to-low transition of the INTR signal. This resets the SHIFT REGISTER which causes the input to the D-type latch, LATCH 1, to go low. As the latch enable input is still present, the Q output will go high, which then allows the INTR F/F to be RESET. This reduces the width of the resulting INTR output pulse to only a few propagation delays (approximately 300 ns).

When data is to be read, the combination of both CS and RD being low will cause the INTR F/F to be reset and the TRI-STATE output latches will be enabled to provide the 8-bit digital outputs.

The complete circuit diagram, Visual Basic Timer program and the Visual Basic Timer program are presented below.

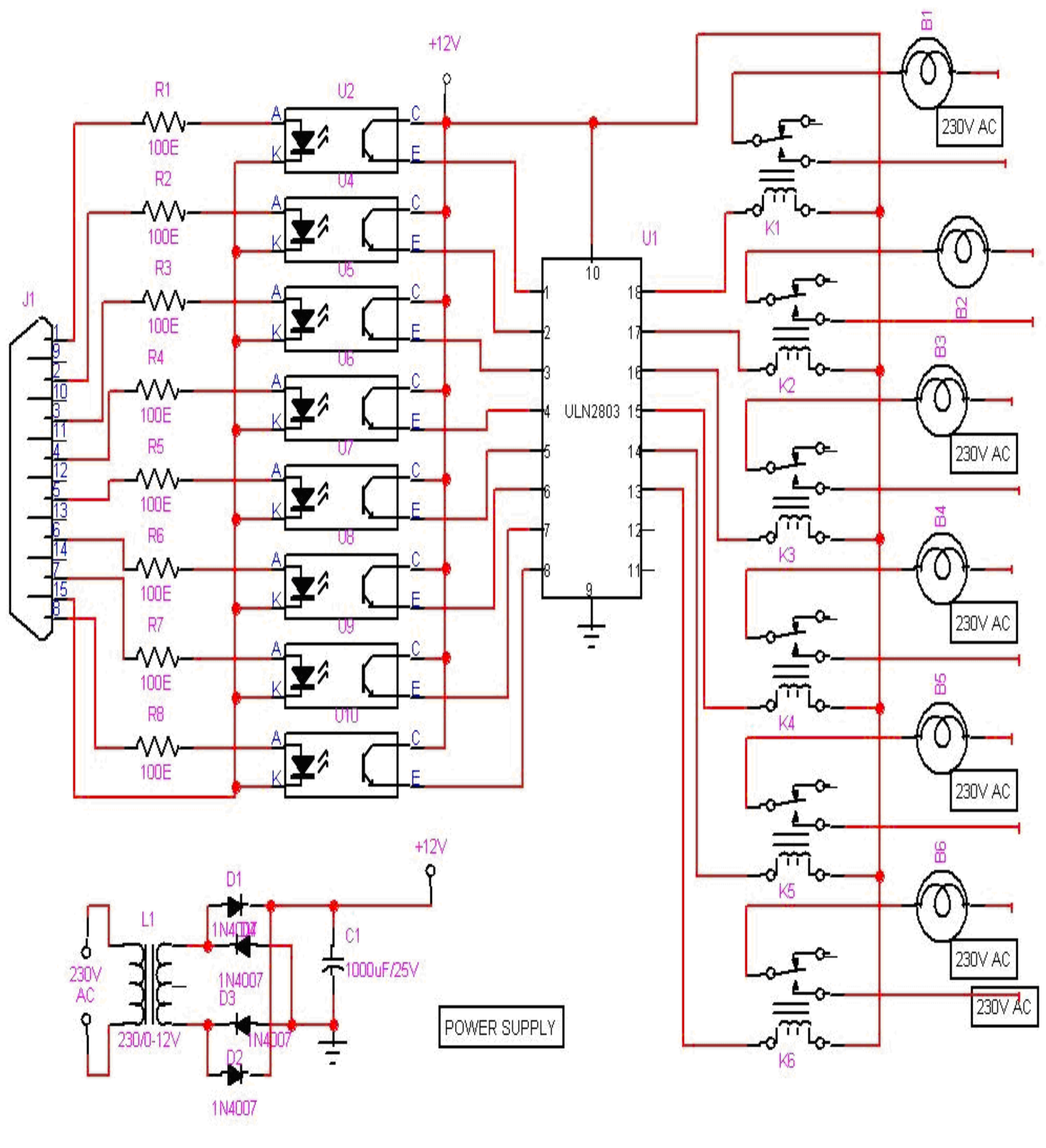


Fig. 7. Circuit diagram

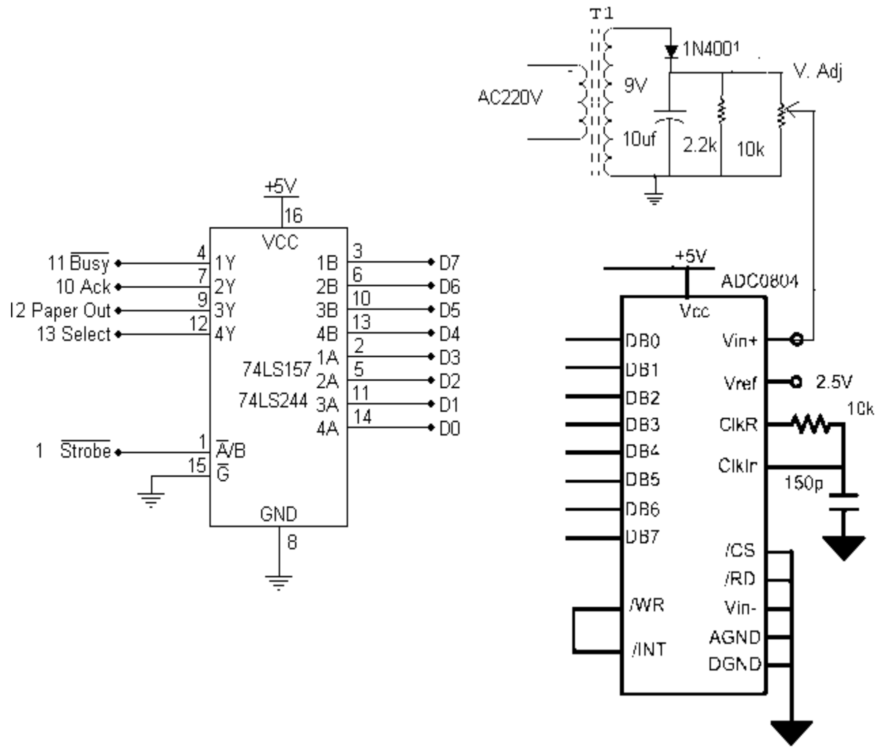


Fig. 8. Visual basic timer program

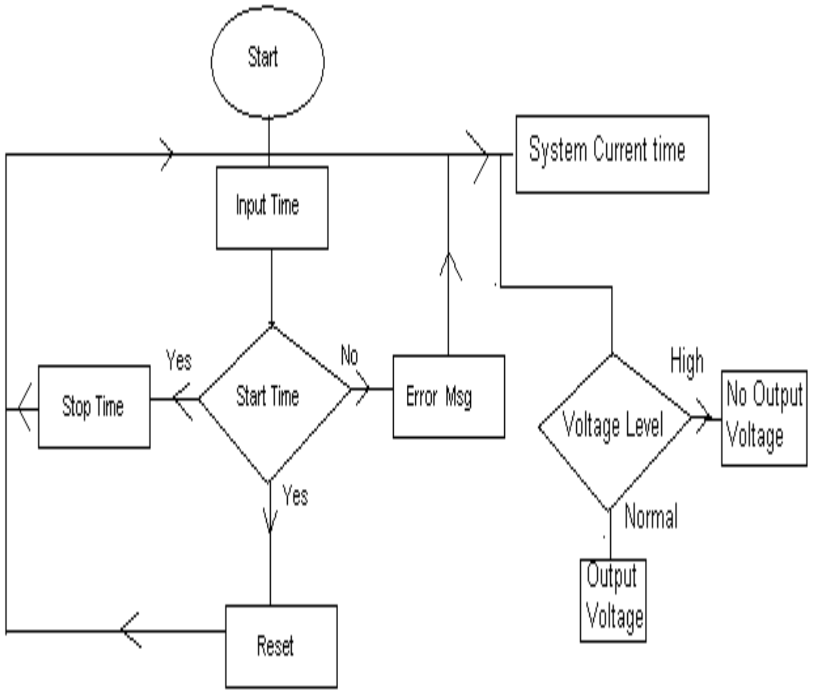


Fig. 9. System flow chart

3. SYSTEM IMPLEMENTATION

Analysis of the theories in design methodology and specifications was made to govern the construction. After successful simulation and testing, the research was put together to create the finalized version.

3.1 System Implementation Included Four Procedures

1 Components Sourcing.
2 Bread Board Implementation.
3 Strip/Vero Board Construction.
4 Software Coding

3.2 Component Sourcing

Gathering of components from the specification in the circuit design precedes assembly or construction of the project. We started with listing the component according to types and values.

3.3 Strip/Vero Board Construction

This involves the actual construction, which is the hard wiring of the circuit already prototyped. This consists of thin copper strips on one side and plain insulator board on the other side punched with hole at 0.1- inch matrix interval. Components are mounted on the plain side and soldered on the copper side. The copper strip runs from left to right and all components soldered on the strip are automatically joined together; where that is not required the strip is cut at appropriate point with drill or sharp object. All the components were mounted and soldered taking care that the transistors, electrolytic capacitors, diodes and the regulator were mounted the correct way round. While soldering the solder blobs should not touch the adjacent tracks otherwise there will be short circuit. Finally the IC should be mounted on the IC socket and tracks under the socket were cut to separate the pins.

3.4 Regulated Power Supply Design

This module will consist of a center tapped Transformer that will step down the AC voltages from 220V to 12 Vrms. Then this will pass through a Full Wave Bridge Rectifier and then a filtering capacitor to achieve close to DC voltage level.

The transformer, 220 to 12 volts. The secondary rms voltage is 12 Volt; we can calculate the V peak after the diodes, using the formula:

$$V_{DC(peak)} = 12\sqrt{2} - 1.4 = 17 - 1.4 \approx 15.64V$$

This is the maximum value, for knowing the minimum we need to subtract the ripple:

$$V_r = \frac{I}{2\bullet f\bullet C}$$

Let's assume that the circuit we are going to feed requires that V ripple is not more than 2 Volt.

$$V_{min} = 15.6 - 2 = 13.6V$$

Choosing the Right Capacitor:

$$C = \frac{I}{2\bullet f\bullet V_r} = \frac{0.5}{2\bullet 60\bullet 2} = 2.1\bullet 10^{-3} = 2100uF$$

The nearest available size is 2200uF. Now we need to know what is the maximum voltage that the capacitor is going to be exposed to. We must consider the worst case; this is the V rectified peak in no-load condition, which means without deducting the voltage drop on diodes:

$$V_{max} = V_{peak} = V_{rms}\bullet\sqrt{2} = 12\bullet 1.41 \simeq 17Volt$$

Available commercial values are 16V, 25V, 35V, 50V, 63V. Well, 25V is ok for us. So, our capacitor is completely defined as "Electrolytic capacitor 2200uF x 25V".

3.5 The VB Programming Process

1. Start Visual Basic.
2. Create a new application or load an existing application. Insert the necessary object and adjust their properties and code where necessary. Test your application with the debugging tools Visual Basic supplies. The debugging tools help you locate and eliminate program errors (or *bugs*) that can appear despite your best efforts to keep them out. A *bug* is a program error that you must correct (debug) before your program will execute properly.
3. However VB has quick compiler that once you are typing your program, it is compiling as you move to the next line. Then finally run your finished application project.
4. Connect your computer interface and run the program, observe what happens.
5. Quit Visual Basic.

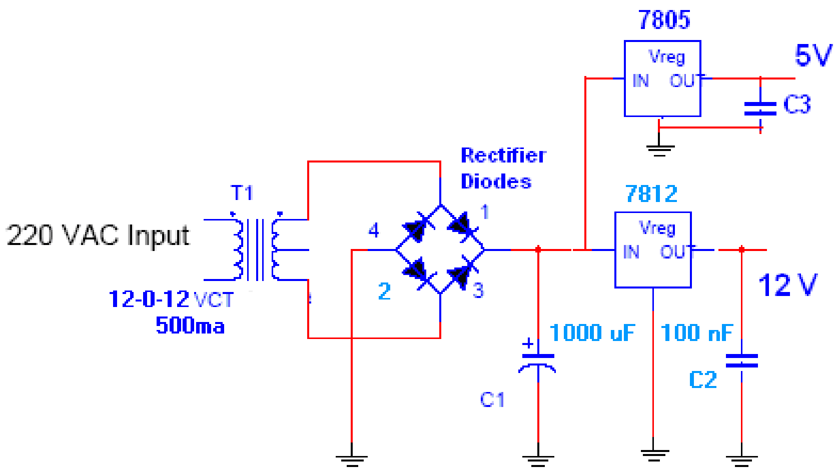


Fig. 10. AC/DC converter schematic

4. SYSTEM TESTING AND INTEGRATION

4.1 Inspection and Resistance Tests

Inspection and resistance tests has to be performed before connecting power to the constructed circuit, to avoid system blowing up and hours of efforts will be frustrated. By inspection, it was ensured that all components had been well soldered and no bridges across conducting path. Using the ohmmeter at lowest range, it was ensured that there is no short circuit between supply live and neutral, and DC +ve and –ve lines. Then the system was connected to power.

4.2 D.C Voltage (Power Supply) Tests

The output voltage of the power supply was measured to be 12V and 5V respectively, which is the output of the voltage regulator in line with the specification.

4.3 Functional Test

Here the system is checked for performance as expected by connecting the system as show in the diagram below. The computer system was powered up, and then the Visual Basic Application program (Relay Board Timer) started. Next the control box was powered. An ON is set anytime it is intended to leave a device powered in the timer application software, finally the timer is started.

From observations, after a thorough testing of the program and the system in general, the system performed as expected, switching on the device and then switching it off immediately the preset time has elapsed. The timer program is interesting and shows features like elapsed time, ON (Green) and OFF (Red) Indicator colors, timer input error checks and voltage isolation features of the control box and so on.

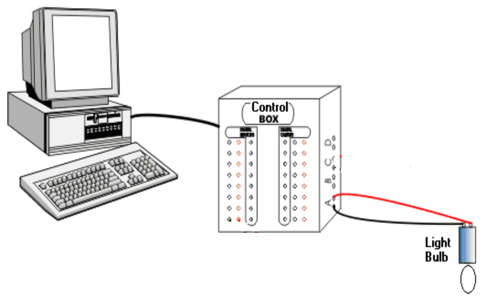


Fig. 11. Relay board timer control system setup test

5. CONCLUSION

The research, “Microcomputer Based Time Operated Power Switching System (with overload protection)”, proved to be a very interesting research to embark on, however whenever there is a task to perform, there could be problems likely to come up. In the course of this research different designs were tried in other to come up with easier and simplified approach to the “Timer Program”. Creating an error check in the program proved to be a challenging experience that demands a lot of thinking and trials.

The areas of continued work could involve improving the timer program and the control box to control more than one device simultaneously. Finally, any improvement on the design would depend on an individual's choice of design.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Somalraju S, Murali V, Saha G, Vaidehi V. Robust railway crack detection scheme (RRCDS) using LED- LDR assembly. International Conference on Recent Trends in Information Technology (ICRTIT). 2012;477-482.
2. Noor MH, Razak MF, Saaid MS. Design and development of "smart basket" system for resource optimization 2015. IEEE Control and System Graduate Research Colloquium (ICSGRC); 2015.
3. Goggle Search engine. Available:www.google.com ADC0804, ULN2803, 74LS244, 74LS157 Datasheets.
4. Anon V. Automatic transfer switch; 2013. Available:www.wikipedia.org/wiki/Transfer_switch (Accessed: February 10, 2013)
5. Okafor EC. Digital devices and applications. Immaculate Publications Ltd Enugu, Nigeria; 2003.
6. Ezema CN, et al. Energy efficient hybrid digital weighing scale. International Journal of Engineering Research and Reviews. 2016;4(1):74-81.
7. Fuller JP. Reference Manual Visual Basic Programming; 1997.
8. Tocci NS, Widmer R, Mos GL. Digital systems: Principles and applications, 9th ed. Pearson Prentice hall; 2007.
9. Das M, Sanaullah HM, Sarower MM, Hassan CK. Development of a cell phone based remote control system: An effective switching system for controlling home and office appliances. International Journal of Electrical and Computer Sciences IJECS-IJENS. 2009;9(10).
10. Anon V. Electrical and electronic components; 2013. Available:www.getprice.com.au/electronic-components.htm (Accessed: April 1, 2013)
11. Ezema CN, et al. Design and construction of enhanced microcontroller based electronic voting machine with digital display. International Journal of Electrical and Electronics Research. 2016;4(1):93-103.
12. Theraja BL, Theraja AK. A textbook of electrical technology. S. Chand & Company Ltd; 2004.
13. Paul H, Winfield H. The art of electronics. Cambridge University Press; 1995.
14. Jain RP. Modern digital electronics, 3rd ed. Tata Mcgraw-Hill Publishing Company Limited; 2004.
15. Anderson WJ. Automatic transfer switches and engine control; 2003. Available:www.file-ee-patents.com Accessed: February 10, 2013.
16. Mano MM. Digital logic and computer design. Prentice Hall of India Pvt. Ltd; 2008.
17. Aguinaga J. Study of transfer switches. MSc Thesis Report, Helsinki University of Technology, Espoo, Finland. 2008;102.
18. Anon V. Automatic transfer switch-owner's manual, generac power systems Inc., Whitewater, USA. 2012;1-20.
19. Akparibo RA. A solar radiation tracker for solar energy optimization. BSc Project Report, University of Mines and Technology, Tarkwa. 2011;20-32.
20. Anon V. Electronic components; 2013. Available:www.futurlec.com/Components.shtml (Accessed: April 1, 2013)
21. Anon V. Low voltage automatic transfer switch system; 2010. (Available:www.asco.com Accessed: December 20, 2012)
22. Anon V. Proteus PCD design package; 2013. Available:www.labcenter.com/products/pcboverview.cfm (Accessed: January 2, 2013)
23. Anon V. Automatic transfer switches international; 2012. Available:www.cumminspower.com (Accessed: December 20, 2012)

APPENDIX

SOURCE CODE

```
Option Explicit
'Dim Msg1 As Variant
Dim Check1 As Variant
Dim KeyAscii As Integer
Dim OutNum As Integer
Dim PortState As Integer
Dim PortNum As Integer
Dim StartMsg As Variant
Dim StartTime As Variant
Dim nibble1 As Byte, nibble2 As Byte, convert As Integer, flag As Boolean
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
Private Sub ChkBit1_Click()
If ChkBit1.Value = 1 And Text1 > 0 Then
     Timer3.Interval = 1000
     Timer3.Enabled = True
     OutNum = 1
    Call Outport
     Text1.BackColor = vbGreen
Else
  ChkBit1.Value = 0
  Text1 = 0
   End If
End Sub

Private Sub ChkBit2_Click()
If ChkBit2.Value = 1 And Text2 > 0 Then
     Timer4.Interval = 1000
     Timer4.Enabled = True
     OutNum = 2
     Call Outport
     Text2.BackColor = vbGreen
 Else
  ChkBit2.Value = 0
  Text2 = 0
End If
End Sub

Private Sub ChkBit3_Click()
If ChkBit3.Value = 1 And Text3 > 0 Then
     Timer5.Interval = 1000
     Timer5.Enabled = True
     OutNum = 4
     Call Outport
     Text3.BackColor = vbGreen
 Else
  ChkBit3.Value = 0
  Text3 = 0
End If
End Sub

Private Sub ChkBit4_Click()
If ChkBit4.Value = 1 And Text4 > 0 Then
     Timer6.Interval = 1000
     Timer6.Enabled = True
     OutNum = 8
     Call Outport
     Text4.BackColor = vbGreen
 Else
  ChkBit4.Value = 0
  Text4 = 0
End If
End Sub

Private Sub ChkBit5_Click()
If ChkBit5.Value = 1 And Text5 > 0 Then
     Timer7.Interval = 1000
     Timer7.Enabled = True
     OutNum = 16
     Call Outport
     Text5.BackColor = vbGreen
 Else
  ChkBit5.Value = 0
  Text5 = 0
End If
End Sub

Private Sub ChkBit6_Click()
If ChkBit6.Value = 1 And Text6 > 0 Then
     Timer8.Interval = 1000
     Timer8.Enabled = True
     OutNum = 32
     Call Outport
     Text6.BackColor = vbGreen
 Else
  ChkBit6.Value = 0
  Text6 = 0
End If
End Sub

Private Sub ChkBit7_Click()
If ChkBit7.Value = 1 And Text7 > 0 Then
     Timer9.Interval = 1000
     Timer9.Enabled = True
     OutNum = 64
     Call Outport
     Text7.BackColor = vbGreen
 Else
  ChkBit7.Value = 0
  Text7 = 0
End If
End Sub

Private Sub ChkBit8_Click()
If ChkBit8.Value = 1 And Text8 > 0 Then
     Timer10.Interval = 1000
     Timer10.Enabled = True
     OutNum = 128
     Call Outport
     Text8.BackColor = vbGreen
 Else
```

```
  ChkBit8.Value = 0
  Text8 = 0
End If
End Sub

Private Sub cmdExit_Click()
Call PortOut(888, 0)
Call PortOut(890, 0)
Unload Me
End
End Sub

Private Sub Command1_Click()

     If Text9 > 0 Then
         Call PortOut(888, &HFF)
     Timer2.Interval = 1000
      Timer2.Enabled = True
      Text9.BackColor = vbGreen
        flag = True
        Image1.Visible = True
 Else
          Call PortOut(888, &H0)
             Timer2.Enabled = False
   Check1 = MsgBox("Please Enter Countdown Time in Seconds in All Device Timer Box")
    flag = False

   End If
End Sub

Private Sub Command2_Click()
        Call PortOut(888, &H0)
             Timer2.Enabled = False
       Text9.BackColor = vbRed
       Text9 = 0
       flag = False
       Image1.Visible = False

End Sub

Private Sub Form_Load()
   Call PortOut(888, 0)
     Call PortOut(890, 0)
    Form1.Show
    StartMsg = MsgBox("Please,First Enter the Countdown Time in Seconds in device box then  Select
Device!", vbCritical)
    Text1 = 0
    Text2 = 0
    Text3 = 0
    Text4 = 0
    Text5 = 0
    Text6 = 0
    Text7 = 0
    Text8 = 0
    Text9 = 0
    flag = False
End Sub
```

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
'
' Invalidate keystroke if not a digit, or backspace.
'
If (Not IsNumeric(Chr$(KeyAscii)) And Chr$(KeyAscii) <> vbBack) Then
KeyAscii = 0
Check1 = MsgBox("Please You Must Enter Time in Seconds", vbCritical)
End If
End Sub

Private Sub Text2_KeyPress(KeyAscii As Integer)
If (Not IsNumeric(Chr$(KeyAscii)) And Chr$(KeyAscii) <> vbBack) Then
KeyAscii = 0
Check1 = MsgBox("Please You Must Enter Time in Seconds", vbCritical)
End If
End Sub

Private Sub Text3_KeyPress(KeyAscii As Integer)
If (Not IsNumeric(Chr$(KeyAscii)) And Chr$(KeyAscii) <> vbBack) Then
KeyAscii = 0
Check1 = MsgBox("Please You Must Enter Time in Seconds", vbCritical)
End If
End Sub

Private Sub Text4_KeyPress(KeyAscii As Integer)
If (Not IsNumeric(Chr$(KeyAscii)) And Chr$(KeyAscii) <> vbBack) Then
KeyAscii = 0
Check1 = MsgBox("Please You Must Enter Time in Seconds", vbCritical)
End If
End Sub

Private Sub Text5_KeyPress(KeyAscii As Integer)
If (Not IsNumeric(Chr$(KeyAscii)) And Chr$(KeyAscii) <> vbBack) Then
KeyAscii = 0
Check1 = MsgBox("Please You Must Enter Time in Seconds", vbCritical)
End If
End Sub
Private Sub Text6_KeyPress(KeyAscii As Integer)
If (Not IsNumeric(Chr$(KeyAscii)) And Chr$(KeyAscii) <> vbBack) Then
KeyAscii = 0
Check1 = MsgBox("Please You Must Enter Time in Seconds", vbCritical)
End If
End Sub

Private Sub Text7_KeyPress(KeyAscii As Integer)
If (Not IsNumeric(Chr$(KeyAscii)) And Chr$(KeyAscii) <> vbBack) Then
KeyAscii = 0
Check1 = MsgBox("Please You Must Enter Time in Seconds", vbCritical)
End If
End Sub

Private Sub Text8_KeyPress(KeyAscii As Integer)
If (Not IsNumeric(Chr$(KeyAscii)) And Chr$(KeyAscii) <> vbBack) Then
KeyAscii = 0
Check1 = MsgBox("Please You Must Enter Time in Seconds", vbCritical)
End If
End Sub
```

```
Private Sub Text9_KeyPress(KeyAscii As Integer)
If (Not IsNumeric(Chr$(KeyAscii)) And Chr$(KeyAscii) <> vbBack) Then
KeyAscii = 0
Check1 = MsgBox("Please You Must Enter Time in Seconds", vbCritical)
End If
End Sub

Public Sub Timer1_Timer()
Dim Value As Byte
Dim value1 As Byte
Dim Volt As Integer
    lblTime.Caption = Format(Time, "hh:mm:ss ampm")
    'voltage level
Call PortOut(890, 1)
        Sleep (100)
        get_val ((PortIn(889) Or &H80) And &HF8)
        nibble1 = convert

    Call PortOut(890, 2)
       Sleep (100)
        get_val ((PortIn(889) Or &H80) And &HF8)
        nibble2 = convert

      Volt = (((Val("&H" & Hex(nibble2) & Hex(nibble1))) * 19.60784) / 1000)
         Select Case Volt
         Case 0: Label3.Caption = (210)
         Case 1: Label3.Caption = (220)
        Case 2: Label3.Caption = (230)
        Case 3: Label3.Caption = (240)
        Case 4: Label3.Caption = (250)
        Case 5: Label3.Caption = (260)
        End Select

       If (Label3.Caption) > 240 Then
         Call PortOut(888, 0)
      Else
      End If
  End Sub

Public Sub Outport()
If OutNum = 0 Then
 Call PortOut(888, 0)
   Else
   PortState = PortIn(888)
   PortNum = PortState + OutNum
 Call PortOut(888, PortNum)
 End If
End Sub
Private Sub Timer2_Timer()
If Text9 = 0 Then
          Timer2.Enabled = False
          OutNum = 0
      Call Outport
         Text9.BackColor = vbRed
          flag = False
          Image1.Visible = False
          Exit Sub
```

```
     Else
          Text9 = Text9 - 1
End If
End Sub

Private Sub Timer3_Timer()

 If Text1 = 0 Then
          Timer3.Enabled = False
          OutNum = 0
      Call Outport
         Text1.BackColor = vbRed
          ChkBit1.Value = 0
          Exit Sub
     Else
          Text1 = Text1 - 1
End If
End Sub

Private Sub Timer4_Timer()
If Text2 = 0 Then
          Timer4.Enabled = False
          OutNum = 0
      Call Outport
         Text2.BackColor = vbRed
          ChkBit2.Value = 0
          Exit Sub
     Else
          Text2 = Text2 - 1
End If
End Sub

Private Sub Timer5_Timer()
If Text3 = 0 Then
          Timer5.Enabled = False
          OutNum = 0
      Call Outport
         Text3.BackColor = vbRed
          ChkBit3.Value = 0
          Exit Sub
     Else
          Text3 = Text3 - 1
End If
End Sub

Private Sub Timer6_Timer()
If Text4 = 0 Then
          Timer6.Enabled = False
          OutNum = 0
      Call Outport
         Text4.BackColor = vbRed
          ChkBit4.Value = 0
          Exit Sub
     Else
          Text4 = Text4 - 1
End If
End Sub
```

```
Private Sub Timer7_Timer()
If Text5 = 0 Then
          Timer7.Enabled = False
          OutNum = 0
      Call Outport
         Text5.BackColor = vbRed
          ChkBit5.Value = 0
          Exit Sub
     Else
          Text5 = Text5 - 1
End If

End Sub

Private Sub Timer8_Timer()
If Text6 = 0 Then
          Timer8.Enabled = False
          OutNum = 0
      Call Outport
         Text6.BackColor = vbRed
          ChkBit6.Value = 0
          Exit Sub
     Else
          Text6 = Text6 - 1
End If

End Sub

Private Sub Timer9_Timer()
If Text7 = 0 Then
          Timer9.Enabled = False
          OutNum = 0
          Call Outport
         Text7.BackColor = vbRed
          ChkBit7.Value = 0
          Exit Sub
     Else
          Text7 = Text7 - 1
End If

End Sub
Private Sub Timer10_Timer()
If Text8 = 0 Then
          Timer10.Enabled = False
          OutNum = 0
      Call Outport
         Text8.BackColor = vbRed
          ChkBit8.Value = 0
          Exit Sub
     Else
          Text8 = Text8 - 1
End If

End Sub
Private Sub get_val(Value As Integer)
       Select Case Value
       Case &H80:
        convert = 0
      Case &H88:
         convert = 1

      Case &H90:
        convert = 2
     Case &H98:
        convert = 3
      Case &HA0:
        convert = 4
      Case &HA8
        convert = 5

      Case &HB0:
        convert = 6

      Case &HB8:
        convert = 7

      Case &HC0:
        convert = 8

      Case &HC8:
        convert = 9

      Case &HD0:
        convert = 10

      Case &HD8:
        convert = 11

      Case &HE0:
        convert = 12
      Case &HE8:
        convert = 13

      Case &HF0:
        convert = 14
      Case &HF8:
        convert = 15
      Case Else:
        convert = &HFF

     End Select
 End Sub
```