

Original Research Article

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

Chukwuedozie N Ezema^{*1}, Okechi Onuoha², Nwanyinnaya Nwogu³, Albert C Agulanna⁴, Helen U Nonyelu⁵

¹Nnamdi Azikiwe University Awka Anambra State, ^{2,3,4}Projects Development Institute (PRODA), Enugu State, Nigeria, ⁵Scientific Equipment Development Institute (SEDI), Enugu State, Nigeria
ecnaxel@gmail.com, onuohaokechi2000@yahoo.com, nnayanwogu@gmail.com, acagulanna@yahoo.com, al4helen@yahoo.com

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.

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 [22, 23]. 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 1000Watts 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 [4].

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.

Practically all computers depend on an accurate internal clock signal to allow synchronized processing [2]. 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. 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 [3].

1.2 The Timer

A Timer is a specialized type of clock. A timer can be used to control the sequence of an event or process [2]. Examples include: Mechanical timers, Electromechanical Timers, Digital Timers and Computer Timers.

1.2.1 Parallel Port Basics

In computers, ports are used mainly for two reasons: Device control and communication. 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 [20]. 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. 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 [1].

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.

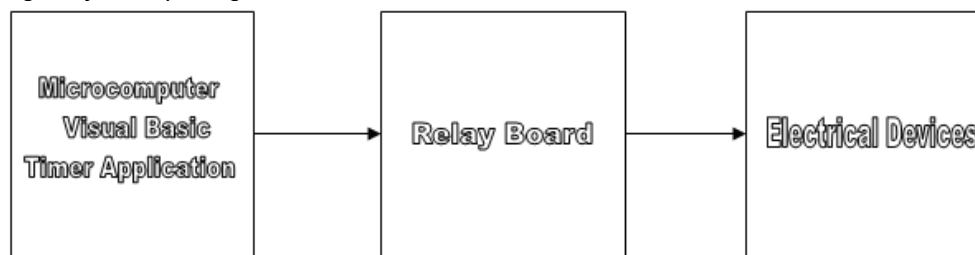


Fig 1: Block Diagram Overview of the Research

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. 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.

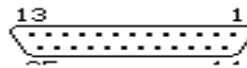
Digital outputs are items such as speakers, lights, buzzers, LEDs and circuits etc. Digital inputs are devices such as micro-switches and relays. They are either 'on' or 'off'.

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 25pin 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.

Parallel Port as seen
from outside the PC.



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

2.4 Reading Data from the Parallel Port

Base Address +2 Pin	Output	Base Address +1 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. 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. 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.

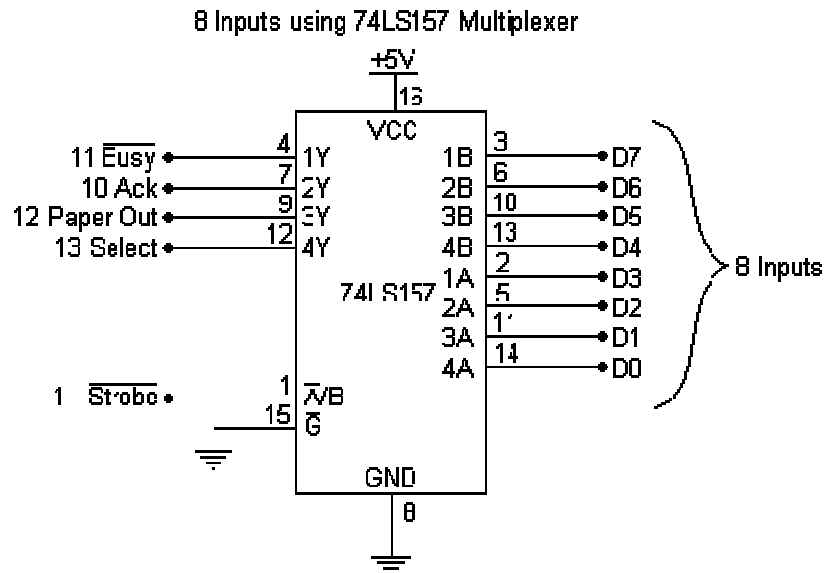


Fig 4: Input Interface

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.

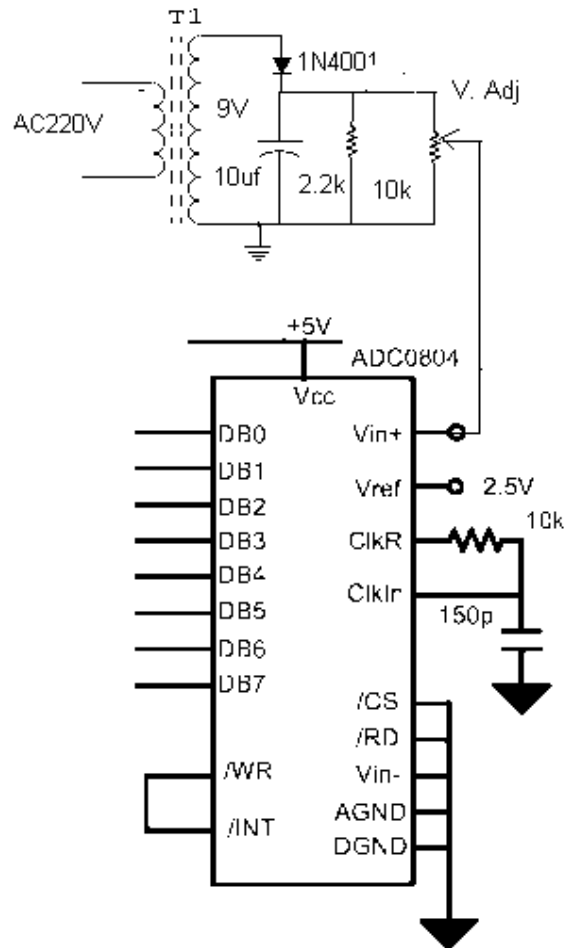


Fig 5: Voltage measurement

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 VR 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.

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. 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)

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

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

187 188 **2.7 Functional Description**

189 The ADC0804 contains a circuit equivalent of the 256R network. Analog switches are
190 sequenced by successive approximation logic to match the analog difference input voltage $[V_{in} (+) -$
191 $V_{in} (-)]$ to a corresponding tap on the R -network. The most significant bit is tested first and after 8
192 comparisons (64 clock cycles) a digital 8-bit binary code (11111111 = full-scale) is transferred to an
193 output latch and then an interrupt is asserted (INTR makes a high-to-low transition). A conversion in
194 process can be interrupted by issuing a second start command. The device may be operated in the
195 free-running mode by connecting INTR to the WR input with CS =0. To ensure start-up under all
196 possible conditions, an external WR pulse is required during the first power-up cycle.

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

201 A functional diagram of the A/D converter is shown in Figure 6. All of the package pin outs are
202 shown and the major logic control paths are drawn in heavier weight lines. The converter is started by
203 having CS and WR simultaneously low. This sets the start flip-flop (F/F) and the resulting "1" level
204 resets the 8-bit shift register, resets the Interrupt (INTR) F/F and inputs a "1" to the D flop, F/F1, which
205 is at the input end of the 8-bit shift register. Internal clock signals then transfer this "1" to the Q output
206 of F/F1. The AND gate, G1, combines this "1" output with a clock signal to provide a reset signal to
207 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
208 the 8-bit shift register then can have the "1" clocked in, which starts the conversion process. If the set
209 signal were to still be present, this reset pulse would have no effect (both outputs of the start F/F
210 would momentarily be at a "1" level) and the 8-bit shift register would continue to be held in the reset
211 mode. This logic therefore allows for wide CS and WR signals and the converter will start after at least
212 one of these signals returns high and the internal clocks again provide a reset signal for the start F/F.

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

218 Note that this SET control of the INTR F/F remains low for 8 of the external clock periods (as
219 the internal clocks run at 1/8 of the frequency of the external clock). If the data output is continuously
220 enabled (CS and RD both held low), the INTR output will still signal the end of conversion (by a high-
221 to-low transition), because the SET input can control the Q output of the INTR F/F even though the
222 RESET input is constantly at a "1" level in this operating mode. This INTR output will therefore stay
223 low for the duration of the SET signal, which is 8 periods of the external clock frequency (assuming
224 the A/D is not started during this interval).

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

231 When data is to be read, the combination of both CS and RD being low will cause the INTR
232 F/F to be reset and the TRI-STATE output latches will be enabled to provide the 8-bit digital outputs.
233

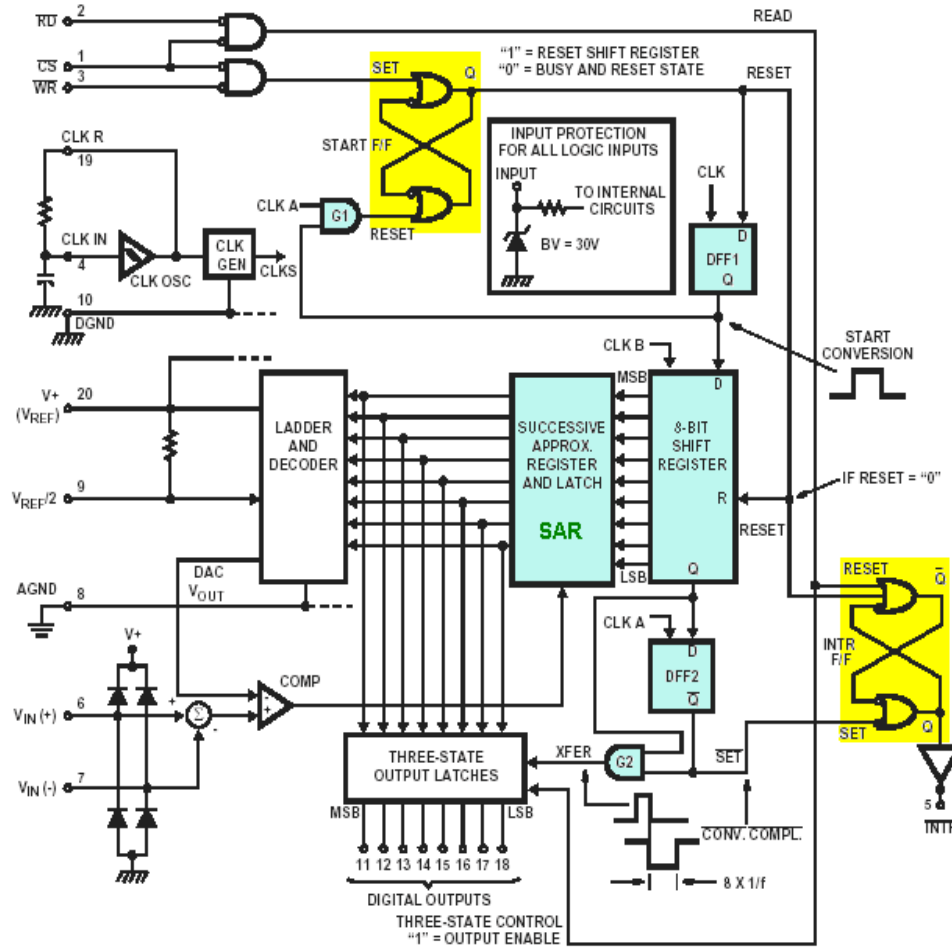


Fig. 6: ADC function diagram

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

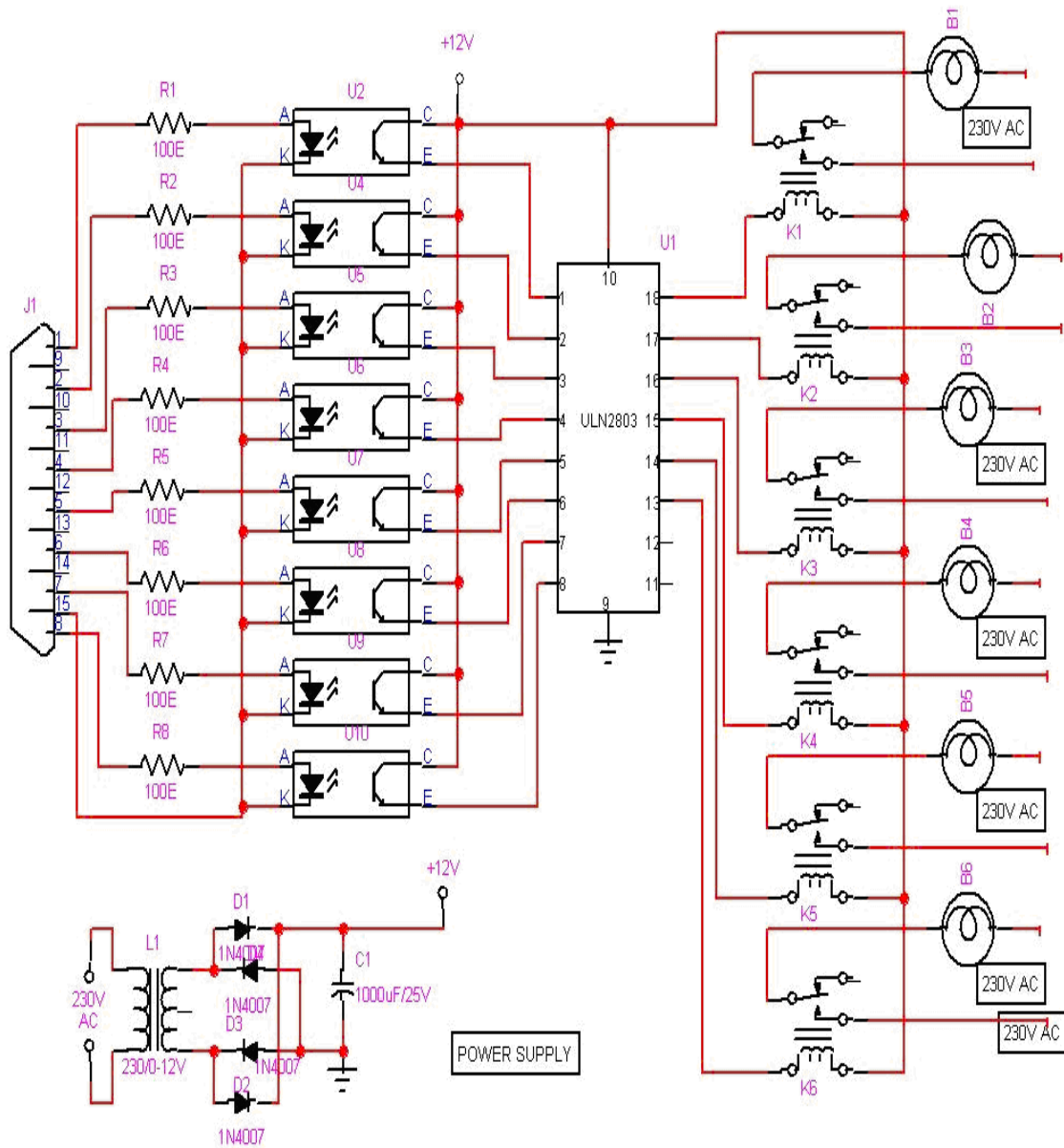


Fig. 7: Circuit Diagram

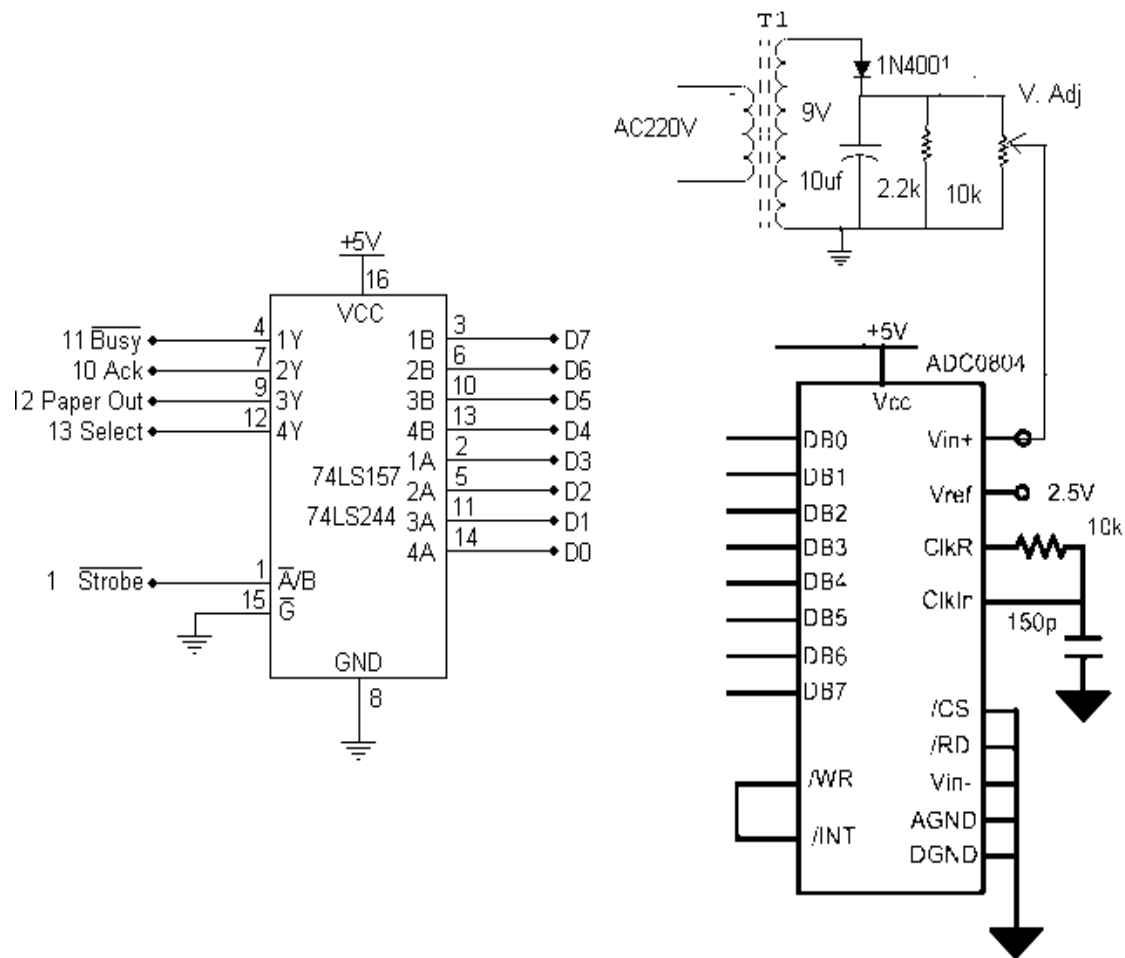


Fig. 8: Visual Basic Timer program

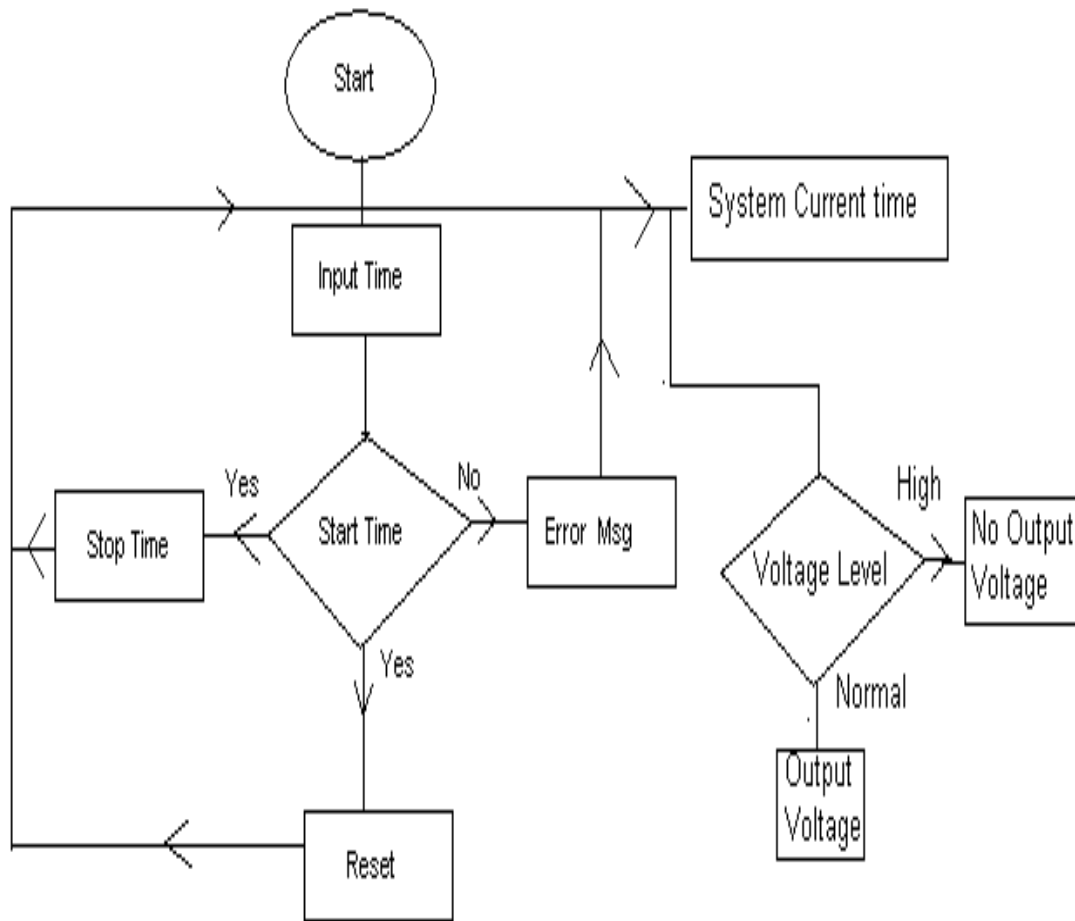


Fig. 9: System Flow Chart

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.

System implementation included four procedures:

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

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.

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.

3.1 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.

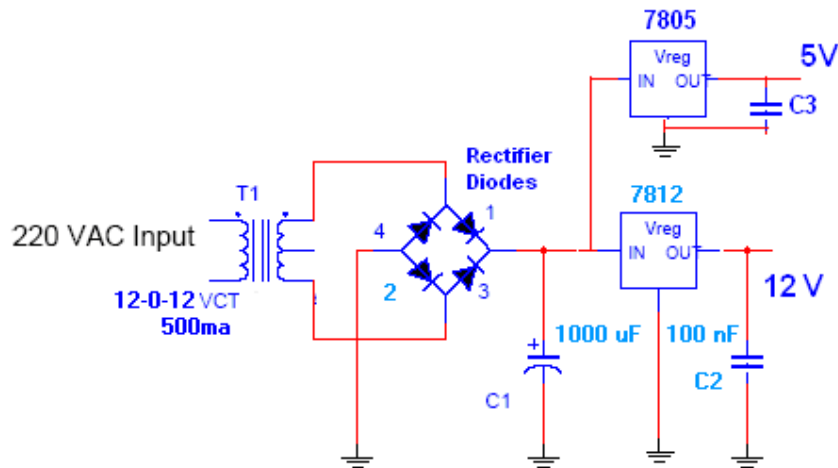


Fig 10: AC/DC Converter Schematic

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:

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

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:

$$C = \frac{I}{2 \cdot f \cdot V_r} = \frac{0.5}{2 \cdot 60 \cdot 2} = 2.1 \cdot 10^{-3} = 2100\mu F$$

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:

$$V_{max} = V_{peak} = V_{rms} \cdot \sqrt{2} = 12 \cdot 1.41 \sim 17V_{olt}$$

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"

3.2 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.

4. SYSTEM TESTING AND INTEGRATION

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.

4.1 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.

4.2 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.

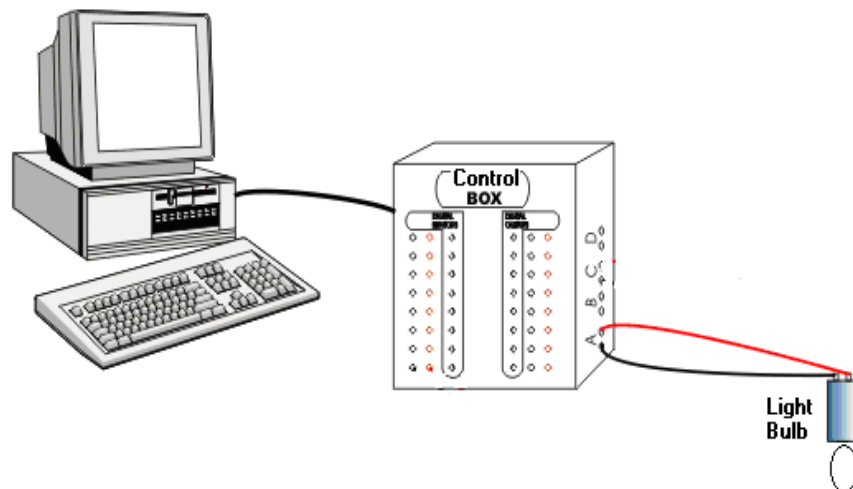


Fig 11: Relay Board Timer Control system setup test

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.

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.

REFERENCES

- [1] Paul H, Winfield H. The Art of Electronics. Cambridge University Press: 1995.
- [2] Okafor EC. Digital Devices and Applications. Immaculate Publications Ltd Enugu, Nigeria; 2003
- [3] Fuller JP. Reference Manual Visual Basic Programming. 1997
- [4] Goggle Search engine. www.google.com: ADC0804, ULN2803, 74LS244, 74LS157 Datasheets.
- [5] 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).
- [6] Theraja B L, Theraja AK. *A Textbook of Electrical Technology*, S. Chand & Company Ltd; 2004.
- [7] Tocci NS, Widmer R, Mos GL. *Digital Systems: Principles and Applications*, 9th ed. Pearson Prentice hall; 2007.
- [8] Jain RP. *Modern Digital Electronics*, 3rd ed. Tata Mcgraw-Hill Publishing Company Limited; 2004.
- [9] Mano MM. *Digital Logic and Computer Design*, Prentice Hall of India Pvt. Ltd: 2008
- [11] Aguinaga J. "Study of Transfer Switches", *MSc Thesis Report*, Helsinki University of Technology, Espoo, Finland; 2008: 102.
- [12] Akparibo RA. "A Solar Radiation Tracker for Solar Energy Optimisation", *BSc Project Report*, University of Mines and Technology, Tarkwa; 2011: 20 - 32.
- [13] Anderson WJ. "Automatic Transfer Switches and Engine Control"; 2003. www.file-ee-patents.com, Accessed: February 10, 2013.
- [1] Anon V. "Low Voltage Automatic Transfer Switch System"; 2010. www.asco.com, Accessed: December 20, 2012.
- [14] Anon V. "Automatic Transfer Switches International"; 2012. www.cumminspower.com. Accessed: December 20, 2012.
- [15] Anon V. *Automatic Transfer Switch-Owner's Manual*, Generac Power Systems Inc., Whitewater, USA; 2012: 1 - 20.
- [16] Anon V. "Automatic Transfer Switch"; 2013. www.wikipedia.org/wiki/Transfer_switch, Accessed: February 10, 2013.
- [17] Anon V. "Proteus PCD Design Package"; 2013 www.labcenter.com/products/pcb_overview.cfm. Accessed: January 2, 2013.
- [18] Anon V. "Electrical and Electronic Components"; 2013 www.getprice.com.au/electronic-components.htm. Accessed: April 1, 2013.
- [19] Anon V. "Electronic Components"; 2013. www.futurlec.com/Components.shtml. Accessed: April 1, 2013.
- [20] 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
- [21] Ezema CN et al. "Energy Efficient Hybrid Digital Weighing Scale"; *International Journal of Engineering Research and Reviews* 2016: 4(1) 74-81.
- [22] Somalraju S, Murali V, Saha G, Vaidehi V. "Robust railway crack detection scheme (RRCDS) using LED- LDR assembly", 2012 International Conference on Recent Trends in Information Technology (ICRTIT), pp. 477-482, 19- 21 April 2012.
- [23] 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).

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
```

```

427 Dim StartMsg As Variant
428 Dim StartTime As Variant
429 Dim nibble1 As Byte, nibble2 As Byte, convert As Integer, flag As Boolean
430 Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
431 Private Sub ChkBit1_Click()
432 If ChkBit1.Value = 1 And Text1 > 0 Then
433     Timer3.Interval = 1000
434     Timer3.Enabled = True
435     OutNum = 1
436     Call Outport
437     Text1.BackColor = vbGreen
438 Else
439     ChkBit1.Value = 0
440     Text1 = 0
441 End If
442 End Sub
443
444 Private Sub ChkBit2_Click()
445 If ChkBit2.Value = 1 And Text2 > 0 Then
446     Timer4.Interval = 1000
447     Timer4.Enabled = True
448     OutNum = 2
449     Call Outport
450     Text2.BackColor = vbGreen
451 Else
452     ChkBit2.Value = 0
453     Text2 = 0
454 End If
455 End Sub
456
457 Private Sub ChkBit3_Click()
458 If ChkBit3.Value = 1 And Text3 > 0 Then
459     Timer5.Interval = 1000
460     Timer5.Enabled = True
461     OutNum = 4
462     Call Outport
463     Text3.BackColor = vbGreen
464 Else
465     ChkBit3.Value = 0
466     Text3 = 0
467 End If
468 End Sub
469
470 Private Sub ChkBit4_Click()
471 If ChkBit4.Value = 1 And Text4 > 0 Then
472     Timer6.Interval = 1000
473     Timer6.Enabled = True
474     OutNum = 8
475     Call Outport
476     Text4.BackColor = vbGreen
477 Else
478     ChkBit4.Value = 0
479     Text4 = 0
480 End If
481 End Sub
482
483 Private Sub ChkBit5_Click()
484 If ChkBit5.Value = 1 And Text5 > 0 Then
485     Timer7.Interval = 1000
486     Timer7.Enabled = True

```

```

487     OutNum = 16
488     Call Output
489     Text5.BackColor = vbGreen
490 Else
491     ChkBit5.Value = 0
492     Text5 = 0
493 End If
494 End Sub
495
496 Private Sub ChkBit6_Click()
497 If ChkBit6.Value = 1 And Text6 > 0 Then
498     Timer8.Interval = 1000
499     Timer8.Enabled = True
500     OutNum = 32
501     Call Output
502     Text6.BackColor = vbGreen
503 Else
504     ChkBit6.Value = 0
505     Text6 = 0
506 End If
507 End Sub
508
509 Private Sub ChkBit7_Click()
510 If ChkBit7.Value = 1 And Text7 > 0 Then
511     Timer9.Interval = 1000
512     Timer9.Enabled = True
513     OutNum = 64
514     Call Output
515     Text7.BackColor = vbGreen
516 Else
517     ChkBit7.Value = 0
518     Text7 = 0
519 End If
520 End Sub
521
522 Private Sub ChkBit8_Click()
523 If ChkBit8.Value = 1 And Text8 > 0 Then
524     Timer10.Interval = 1000
525     Timer10.Enabled = True
526     OutNum = 128
527     Call Output
528     Text8.BackColor = vbGreen
529 Else
530     ChkBit8.Value = 0
531     Text8 = 0
532 End If
533 End Sub
534
535 Private Sub cmdExit_Click()
536 Call PortOut(888, 0)
537 Call PortOut(890, 0)
538 Unload Me
539 End
540 End Sub
541
542 Private Sub Command1_Click()
543
544     If Text9 > 0 Then
545         Call PortOut(888, &HFF)
546         Timer2.Interval = 1000

```

```

547     Timer2.Enabled = True
548     Text9.BackColor = vbGreen
549     flag = True
550     Image1.Visible = True
551 Else
552     Call PortOut(888, &H0)
553     Timer2.Enabled = False
554     Check1 = MsgBox("Please Enter Countdown Time in Seconds in All Device Timer Box")
555     flag = False
556
557 End If
558 End Sub
559
560 Private Sub Command2_Click()
561     Call PortOut(888, &H0)
562     Timer2.Enabled = False
563     Text9.BackColor = vbRed
564     Text9 = 0
565     flag = False
566     Image1.Visible = False
567
568 End Sub
569
570 Private Sub Form_Load()
571     Call PortOut(888, 0)
572     Call PortOut(890, 0)
573     Form1.Show
574     StartMsg = MsgBox("Please,First Enter the Countdown Time in Seconds in device box then  Select
575 Device!", vbCritical)
576     Text1 = 0
577     Text2 = 0
578     Text3 = 0
579     Text4 = 0
580     Text5 = 0
581     Text6 = 0
582     Text7 = 0
583     Text8 = 0
584     Text9 = 0
585     flag = False
586 End Sub
587 Private Sub Text1_KeyPress(KeyAscii As Integer)
588     '
589     ' Invalidate keystroke if not a digit, or backspace.
590     '
591     If (Not IsNumeric(Chr$(KeyAscii)) And Chr$(KeyAscii) <> vbBack) Then
592         KeyAscii = 0
593         Check1 = MsgBox("Please You Must Enter Time in Seconds", vbCritical)
594     End If
595 End Sub
596
597 Private Sub Text2_KeyPress(KeyAscii As Integer)
598     If (Not IsNumeric(Chr$(KeyAscii)) And Chr$(KeyAscii) <> vbBack) Then
599         KeyAscii = 0
600         Check1 = MsgBox("Please You Must Enter Time in Seconds", vbCritical)
601     End If
602 End Sub
603
604 Private Sub Text3_KeyPress(KeyAscii As Integer)
605     If (Not IsNumeric(Chr$(KeyAscii)) And Chr$(KeyAscii) <> vbBack) Then
606         KeyAscii = 0

```

```

607 Check1 = MsgBox("Please You Must Enter Time in Seconds", vbCritical)
608 End If
609 End Sub
610
611 Private Sub Text4_KeyPress(KeyAscii As Integer)
612 If (Not IsNumeric(Chr$(KeyAscii)) And Chr$(KeyAscii) <> vbBack) Then
613 KeyAscii = 0
614 Check1 = MsgBox("Please You Must Enter Time in Seconds", vbCritical)
615 End If
616 End Sub
617
618 Private Sub Text5_KeyPress(KeyAscii As Integer)
619 If (Not IsNumeric(Chr$(KeyAscii)) And Chr$(KeyAscii) <> vbBack) Then
620 KeyAscii = 0
621 Check1 = MsgBox("Please You Must Enter Time in Seconds", vbCritical)
622 End If
623 End Sub
624 Private Sub Text6_KeyPress(KeyAscii As Integer)
625 If (Not IsNumeric(Chr$(KeyAscii)) And Chr$(KeyAscii) <> vbBack) Then
626 KeyAscii = 0
627 Check1 = MsgBox("Please You Must Enter Time in Seconds", vbCritical)
628 End If
629 End Sub
630
631 Private Sub Text7_KeyPress(KeyAscii As Integer)
632 If (Not IsNumeric(Chr$(KeyAscii)) And Chr$(KeyAscii) <> vbBack) Then
633 KeyAscii = 0
634 Check1 = MsgBox("Please You Must Enter Time in Seconds", vbCritical)
635 End If
636 End Sub
637
638 Private Sub Text8_KeyPress(KeyAscii As Integer)
639 If (Not IsNumeric(Chr$(KeyAscii)) And Chr$(KeyAscii) <> vbBack) Then
640 KeyAscii = 0
641 Check1 = MsgBox("Please You Must Enter Time in Seconds", vbCritical)
642 End If
643 End Sub
644
645 Private Sub Text9_KeyPress(KeyAscii As Integer)
646 If (Not IsNumeric(Chr$(KeyAscii)) And Chr$(KeyAscii) <> vbBack) Then
647 KeyAscii = 0
648 Check1 = MsgBox("Please You Must Enter Time in Seconds", vbCritical)
649 End If
650 End Sub
651
652 Public Sub Timer1_Timer()
653 Dim Value As Byte
654 Dim value1 As Byte
655 Dim Volt As Integer
656 lblTime.Caption = Format(Time, "hh:mm:ss ampm")
657 'voltage level
658 Call PortOut(890, 1)
659 Sleep (100)
660 get_val ((PortIn(889) Or &H80) And &HF8)
661 nibble1 = convert
662
663 Call PortOut(890, 2)
664 Sleep (100)
665 get_val ((PortIn(889) Or &H80) And &HF8)
666 nibble2 = convert

```

```

667
668     Volt = (((Val("&H" & Hex(nibble2) & Hex(nibble1))) * 19.60784) / 1000)
669     Select Case Volt
670         Case 0: Label3.Caption = (210)
671         Case 1: Label3.Caption = (220)
672         Case 2: Label3.Caption = (230)
673         Case 3: Label3.Caption = (240)
674         Case 4: Label3.Caption = (250)
675         Case 5: Label3.Caption = (260)
676     End Select
677
678     If (Label3.Caption) > 240 Then
679         Call PortOut(888, 0)
680     Else
681     End If
682 End Sub
683
684 Public Sub Outport()
685 If OutNum = 0 Then
686     Call PortOut(888, 0)
687 Else
688     PortState = PortIn(888)
689     PortNum = PortState + OutNum
690     Call PortOut(888, PortNum)
691 End If
692 End Sub
693 Private Sub Timer2_Timer()
694 If Text9 = 0 Then
695     Timer2.Enabled = False
696     OutNum = 0
697     Call Outport
698     Text9.BackColor = vbRed
699     flag = False
700     Image1.Visible = False
701     Exit Sub
702 Else
703     Text9 = Text9 - 1
704 End If
705 End Sub
706
707 Private Sub Timer3_Timer()
708
709 If Text1 = 0 Then
710     Timer3.Enabled = False
711     OutNum = 0
712     Call Outport
713     Text1.BackColor = vbRed
714     ChkBit1.Value = 0
715     Exit Sub
716 Else
717     Text1 = Text1 - 1
718 End If
719 End Sub
720
721 Private Sub Timer4_Timer()
722 If Text2 = 0 Then
723     Timer4.Enabled = False
724     OutNum = 0
725     Call Outport
726     Text2.BackColor = vbRed

```

```

727         ChkBit2.Value = 0
728     Exit Sub
729 Else
730     Text2 = Text2 - 1
731 End If
732 End Sub
733
734 Private Sub Timer5_Timer()
735 If Text3 = 0 Then
736     Timer5.Enabled = False
737     OutNum = 0
738     Call Outport
739     Text3.BackColor = vbRed
740     ChkBit3.Value = 0
741     Exit Sub
742 Else
743     Text3 = Text3 - 1
744 End If
745 End Sub
746
747 Private Sub Timer6_Timer()
748 If Text4 = 0 Then
749     Timer6.Enabled = False
750     OutNum = 0
751     Call Outport
752     Text4.BackColor = vbRed
753     ChkBit4.Value = 0
754     Exit Sub
755 Else
756     Text4 = Text4 - 1
757 End If
758 End Sub
759
760
761 Private Sub Timer7_Timer()
762 If Text5 = 0 Then
763     Timer7.Enabled = False
764     OutNum = 0
765     Call Outport
766     Text5.BackColor = vbRed
767     ChkBit5.Value = 0
768     Exit Sub
769 Else
770     Text5 = Text5 - 1
771 End If
772
773 End Sub
774
775 Private Sub Timer8_Timer()
776 If Text6 = 0 Then
777     Timer8.Enabled = False
778     OutNum = 0
779     Call Outport
780     Text6.BackColor = vbRed
781     ChkBit6.Value = 0
782     Exit Sub
783 Else
784     Text6 = Text6 - 1
785 End If
786

```

```

787 End Sub
788
789 Private Sub Timer9_Timer()
790 If Text7 = 0 Then
791     Timer9.Enabled = False
792     OutNum = 0
793     Call Outport
794     Text7.BackColor = vbRed
795     ChkBit7.Value = 0
796     Exit Sub
797 Else
798     Text7 = Text7 - 1
799 End If
800
801 End Sub
802 Private Sub Timer10_Timer()
803 If Text8 = 0 Then
804     Timer10.Enabled = False
805     OutNum = 0
806     Call Outport
807     Text8.BackColor = vbRed
808     ChkBit8.Value = 0
809     Exit Sub
810 Else
811     Text8 = Text8 - 1
812 End If
813
814 End Sub
815 Private Sub get_val(Value As Integer)
816     Select Case Value
817     Case &H80:
818         convert = 0
819     Case &H88:
820         convert = 1
821
822     Case &H90:
823         convert = 2
824     Case &H98:
825         convert = 3
826     Case &HA0:
827         convert = 4
828     Case &HA8:
829         convert = 5
830
831     Case &HB0:
832         convert = 6
833
834     Case &HB8:
835         convert = 7
836
837     Case &HC0:
838         convert = 8
839
840     Case &HC8:
841         convert = 9
842
843     Case &HD0:
844         convert = 10
845
846     Case &HD8:

```

```
847         convert = 11
848
849     Case &HE0:
850         convert = 12
851     Case &HE8:
852         convert = 13
853
854     Case &HF0:
855         convert = 14
856     Case &HF8:
857         convert = 15
858     Case Else:
859         convert = &HFF
860
861     End Select
862 End Sub
```