International Journal of Engineering and Applied Sciences (IJEAS)
ISSN: 2394-3661, Volume-4, Issue-4, April 2017 (Approved by University Grants Commission, India)

Design of a Power Failure Monitoring Data-Logger System

Sadiq M.O., Adejumobi O.K., Akindele O.A., Olayiwola S.O.

***Abstract*— Power problems such as power surge, high voltage spikes and power outages have caused a lot of damages in homes, offices etc. The problem of observing correctly power supply activities in electrical systems prompted the development of this device.**

This paper therefore presents the 'Design of a 'Power Failure Monitoring Data-Logger System' that will monitor, record, alert and display all power failure activities on the Liquid Crystal display (LCD) unit. This data Logger (or data Recorder) is a microcontroller-based device that records dates and times of data acquisition in order to produce a sequence of events.

In essence, the data is normally collected in non-volatile memory for later download to a computer system. This makes it ideally suited for applications requiring portability.

The design is divided into the Power Supply, Data Logger, User Interface, LCD and the Microcontroller Units. The power supply activities in any electrical system are processed by the Microcontroller and stored in the Logger (Recorder). The Analysis is then displayed the LCD. The Microcontroller which is the major component is programmed in Micro-C language.

It is however recommended that this Data Logger be improved upon to have access to wireless communication systems for automatic alarming, reporting and remote control of events as they are monitored. An automatic transfer Switch should be incorporated to change to any available back-up power source.

In conclusion, proper use of Power failure monitor data Logger will greatly reduce the stress of personnel in monitoring power failure activities.

***Index Terms*— Data, Logger, Microcontroller, Power Failure, Liquid Crystal Display.**

I. INTRODUCTION

A data logger is any device that can be used to store data (Duffy, 2011). This includes many data acquisition devices such as plug-in boards or serial communication systems which use a computer as a real time data recording system. However, most instrument manufacturers consider a data logger a standalone device that can read various types of electrical signals and store the data in internal memory for later download to a computer (Kipp, 1998).
Increasingly, but not entirely, they are based on a digital processor (or computer). They are generally small, battery powered, portable, and equipped with a microprocessor, internal memory for data storage, and sensors. Some data loggers interface with a personal computer and utilize software to activate the data logger and view and analyze the collected data, while others have a local interface device (keypad, LCD) and can be used as a stand-alone device.

Sadiq, M.O., Electrical Engrg Dept. Polytechnic, Ib
Adejumobi, O.K. Computer Engrg, Polytechnic, Ib
Akindele, O.A. Electrical Engrg Dept. Polytechnic, Ib
Olayiwola, S.O. Electrical E Engrg Dept. Polytechnic, Ib

Data loggers vary between general purpose types for a range of measurement applications to very specific devices for measuring in one environment or application type only (Todd, 2009). It is common for general purpose types to be programmable; however, many remain as static machines with only a limited number or no changeable parameters. Electronic data loggers have replaced chart recorders in many applications. One of the primary benefits of using data loggers is the ability to automatically collect data on a 24-hour basis. Upon activation, data loggers are typically deployed and left unattended to measure and record information for the duration of the monitoring period. This allows for a

II. METHODOLOGY

Hardware Section:
In Figures 1 and 2, the design is divided into six (6) Modules namely; Power Supply, User Interface Buttons, Data Logger, Microcontroller, LCD Display and the Alarm.

Software Section:
The Design Algorithm
1. Start

2. Micro-controller configuration
3. Start an endless Loop
4. If power failure is detected, Then the data logger should log ON, Alarm ON, Logger remains ON until the power supply is ON.
5. End Loop End
6.

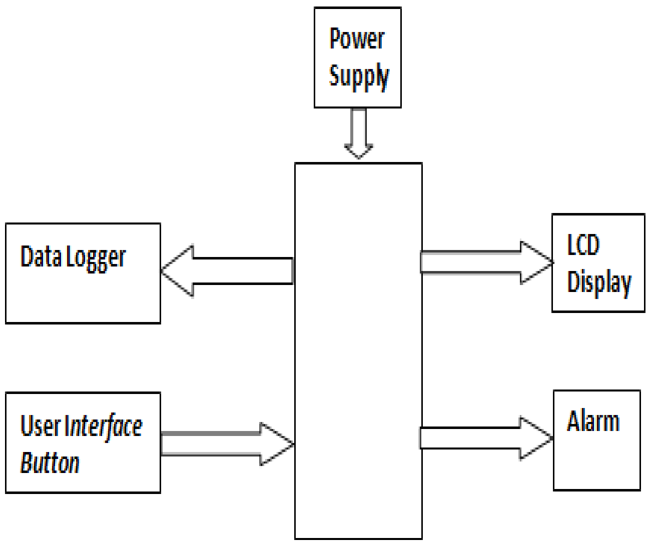

Figure 1: Block Diagram of Power Failure Monitoring Data Logger System

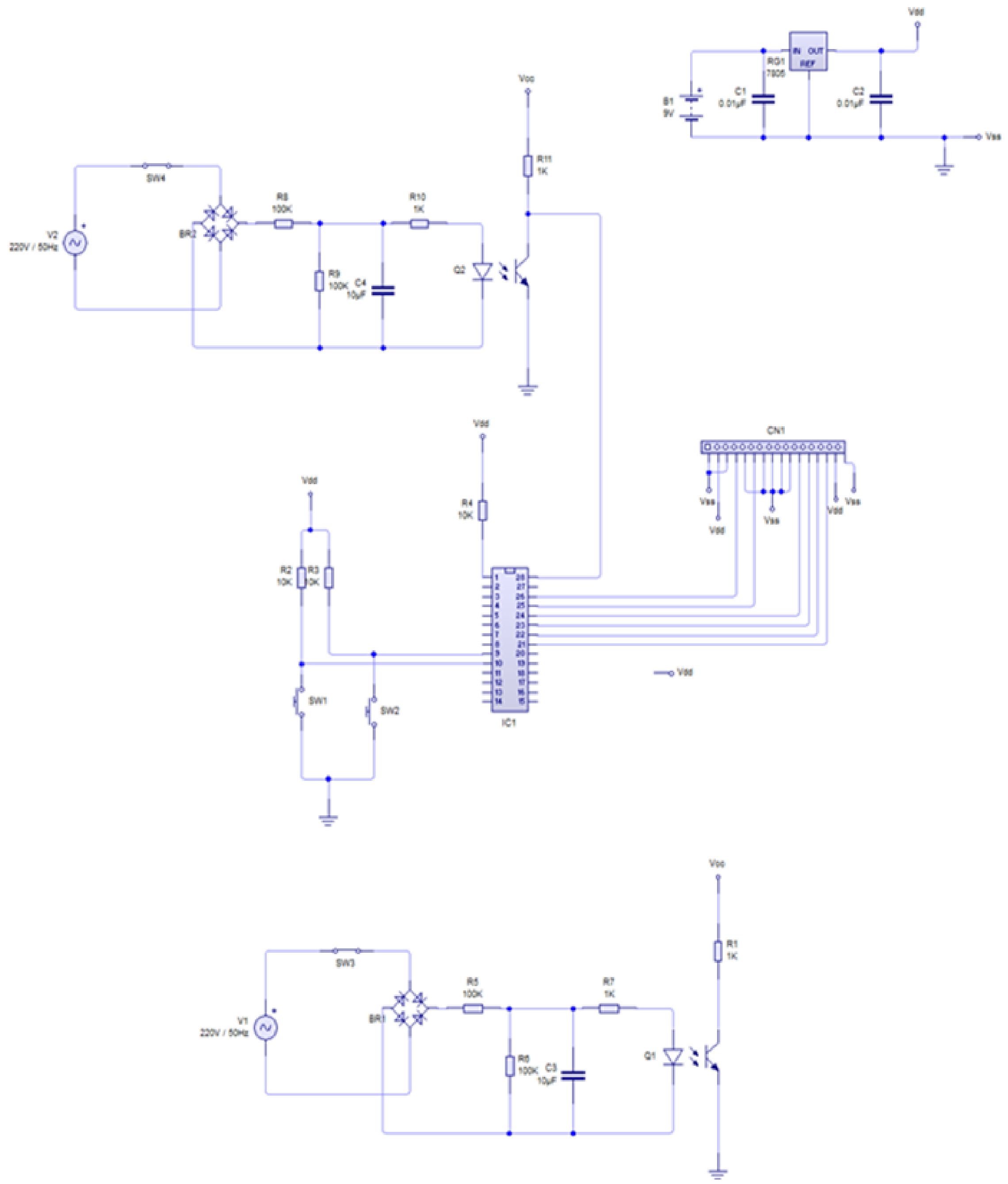

Figure 2: The Circuit Analysis of Power Failure Monitoring Data Logger System

Program Code

```
Design: Power  Failure Monitoring Data Logger System
#define mode RA7_bit
#define incf RA6_bit
#define min 2
#define max 3
#define Power_failure  RC1_bit
#define wait delay_ms(400)
/*/ LCD module connections
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D4 at RB0_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D7 at RB3_bit;
sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// End LCD module connections
   */
// LCD module connections
sbit LCD_RS at RB5_bit;
sbit LCD_EN at RB4_bit;
sbit LCD_D4 at RB3_bit;
sbit LCD_D5 at RB2_bit;
sbit LCD_D6 at RB1_bit;
sbit LCD_D7 at RB0_bit;
sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// End LCD module connections
sbit Data at RA0_bit;
sbit DataDir at TRISA0_bit;
bit oldstate;
char message1[] = "T = 00 C/H = 00%  ";
//char message2[] = "RH   = 00.0 %";
char message3[16]= "Time:          ";
char saved_[16] = "";
char saved_time_off[12][16] = "  POWER ON    ";
char saved_time_on[12][16] =  "  POWER ON    ";
char time_[16] =     "  PWR ON       ";
char time_off[16] =  "  PWR OFF      ";
char time_on;
void power_off();
void power_on();
//int  s;
unsigned short  secs , mins, hrs, alarm_mins, alarm_hrs;
//declare time variable
unsigned short  current_month, day;
unsigned  int leap_year, current_year, ref_year;
unsigned long time_cnt;
unsigned short cnt = 0;
unsigned short i;
char flag;
void conversion();
void StartSignal(){
  DataDir = 0;     // Data port is output
  Data    = 0;
  Delay_ms(25);
  Data    = 1;
  Delay_us(30);
  DataDir = 1;     // Data port is input
}
void InitTimer0(){
  T0CON         = 0x82;
  TMR0H          = 0x0B;
  TMR0L         = 0xDC;
  GIE_bit        = 1;
  TMR0IE_bit          = 1;
}
void interrupt(){
  if(PIR1.TMR2IF){

   T2CON.TMR2ON = 0; // stop timer
   PIR1.TMR2IF  = 0; // Clear TMR0 interrupt flag
  }
  else if (TMR0IF)
  {
     TMR0IF_bit = 0;
     TMR0H          = 0x0B;
     TMR0L         = 0xDC;
     time_cnt++;
       if(time_cnt== 5)
       {  time_cnt = 0;
           secs++;
           if (secs==60)
             {
               secs = 0;            //reset second
                mins++;              //increment minute time by 1
             }
           if(mins == 60)
             {
              mins = 0;          //reset min
              hrs++;             //increment hour time by 1
              }
            if (hrs ==24)
              {
                 hrs = 0;             //reset
                 Day++;
              }                //    hours
       }
       }
}
void main() {
  unsigned short check;
  char s = 0, flag=0;
  char lcd_[20];
  ANCON0 = 255;                          // Configure AN pins
as digital
  ANCON1 = 255;
  TRISA7_bit = 1;
  TRISA6_bit = 1;
  TRISC1_bit = 1;
  OSCCON = 0xF7;

  INTCON.GIE = 1;    //Enable global interrupt
  INTCON.PEIE = 1;   //Enable peripheral interrupt
  //Configure Timer0 module
  InitTimer0();
  // Configure Timer2 module
  PIE1.TMR2IE = 1;  // Enable Timer2 interrupt
  T2CON = 0;        // Prescaler 1:1, and Timer2 is off initially
  PIR1.TMR2IF =0;   // Clear TMR INT Flag bit
  TMR2 = 0;
  secs = 12;
  hrs = 11;
  time_cnt = 0;
  mins = 30;
 Lcd_Init();
 Lcd_Cmd(_Lcd_Clear);
 Lcd_Cmd(_LCD_CURSOR_OFF);
// PORTC = 255;
   time_on = 0;
   cnt = 1;
   flag = 0;
do { conversion();
   lcd_out(1,1,message3);
  // lcd_out(2,1,saved_time_on[cnt]) ;
    for(i=0;i<20*4; i++)
    {
         if(RC1_bit && !flag)
         {
          flag=1;
         }
         if(!RC1_bit && flag)
         {
           s++;
           flag= 0;
         }
         delay_ms(1);
    }
        if(s>2)
        { s=0;
          lcd_out(2,1,saved_time_on[cnt]);
         power_on();
         lcd_out(2,1,saved_time_on[cnt]);
        }
       else
       {  s=0;

         power_off();
         lcd_out(2,1,saved_time_off[cnt])  ;
        }
         if(!incf)
         { i = cnt-1;
          while(incf);
          do{
          if(!mode){
          lcd_cmd(_lcd_clear);
          lcd_out(1,1,saved_time_off[i]);
          lcd_out(2,1,saved_time_on[i]);
           // while(mode);
          delay_ms(1000);
          i--; }
          }
          while(i>0)  ;
           lcd_cmd(_lcd_clear);
           delay_ms(500);
         }
         else if(!mode)
          { lcd_cmd(_lcd_clear);
           while(!mode);
           wait;
           conversion();
           lcd_out(2,1,message3);
           do{  while(!incf)
                {
                 while(!incf);
                 wait;
                 mins++;
                 if (mins == 60) mins = 0;
                 conversion();
             //   lcd_cmd(_lcd_underline_on) ;
                 lcd_out(2,1,message3);
                }
             }while(mode);
             while(!mode);
             wait;
            do{  while(!incf)
                {
                 while(!incf);
                 wait;
                 hrs++;
                 if (hrs == 24) hrs = 0;
                 conversion();
                 lcd_out(2,1,message3);
                }
             }while(mode);
              wait;
             lcd_cmd(_lcd_clear);
             delay_ms(500);
  }while(1);
void conversion()
{
     message3[6] = Hrs/10 + 48;
     message3[7]=  Hrs%10 +48;
     message3[8]= ':';
     message3[9]=  Mins/10 +48;
     message3[10]=  Mins%10 +48;
     message3[11]= ':';
     message3[12]=  secs/10 +48;
     message3[13]=  secs%10 +48;

     time_[10] = Hrs/10 + 48;
     time_[11]=  Hrs%10 +48;
     time_[12]= ':';
     time_[13]=  Mins/10 +48;
     time_[14]=  Mins%10 +48;

     time_off[10] = Hrs/10 + 48;
     time_off[11]=  Hrs%10 +48;
     time_off[12]= ':';
     time_off[13]=  Mins/10 +48;
     time_off[14]=  Mins%10 +48;
}
void power_off()
{
          //  lcd_out(2,1,saved_time_off[cnt]) ;

            if(!time_on)
            {  for(i = 1; i < 15; i++)
               saved_time_off[cnt][i] = time_off[i];
            }
            saved_time_off[cnt][0] = cnt+48;

            time_on = 1;
            flag = 1;
}
void power_on()
{
  // lcd_out(2,1,saved_time_on[cnt]) ;
            if(time_on)
            {
             for(i = 1; i < 15; i++)
                {
                 saved_time_on[cnt][i] = time_[i];
             saved_time_on[cnt][0] = cnt+48;
             time_on = 0;
             if(flag)
             {
              cnt++;
              if(cnt == 10) cnt = 0;
              flag =0;
             }
```

III. Testing

Verification Strategy

As part of the design, the fully functional unit was tested to ensure its working ability and was finally modified to meet the desire specification.

Software Tests

The design was simulated by the stimulation tools provided by MikroC and Virtual breadboard; the tools are used to debug the software code written for the microcontroller to ensure the microcontroller works strictly according to the specifications.

IV. Conclusion and Recommendations

Conclusiom

In recent years, power failure problems like power surge, high voltage spikes have cause a lot of damages and loss of properties in homes and companies which tends to be the main reason why most companies rely on using generators. The proper implementation and use of power failure monitor data logger device will monitor and record power failure activities for proper management and planning.

Recommendations

This device should be adopted in homes and offices to monitor power failure activities for effective planning. However, a Control system (i.e an automatic Transfer Switch) should be incorporated to change to other available sources of power supply as back-up.

References

[1] Duffy, A (2011). ‘Meaning and Definition of Data logger’ *Ottawa Citizen*, retrieved 16 Mar 2012

[2] Todd, B; Schltz, Hawkins, Jensen (2009). "Low Cost RFID Threshold Shock Sensors". *IEEE Sensors Journal***9** (4): 464–469. doi:10.1109/jsen.2009.2014410. Retrieved 8 Mar 2012.

[3] Kipp, W (1998), "Understanding Data Logger Devices" (PDF), *ISA 44th International Instrumentation Symposium*, ISA, retrieved 8 Mar 2012

[4] *Shipping Monitor*, NASA, retrieved 8 March 2012

[5] Sheehan, R; Singh (August 1997), *Analysis Techniques for Package Distribution Environment Data*, Test Engineering &Management, pp. 18–20