Parallel Port Based DAS

Contents

3.1 Introduction to Parallel Port ............................................................ 54
3.2 Design Description ........................................................................ 58
3.3 Hardware Design ........................................................................... 59
  3.3.1 PCB Schematic ...................................................................... 59
  3.3.2 PCB Fabrication ...................................................................... 61
3.4 Software Design ............................................................................ 63
  3.4.1 Application Program Development ...................................... 63
  3.4.2 Algorithm .............................................................................. 64
  3.4.3 Flowchart ................................................................................ 66
  3.4.4 Application Program in C ...................................................... 71
  3.4.5 Application Program in Visual Basic ...................................... 78
  3.4.6 Application Program in LabVIEW ......................................... 84
3.5 Experimental Observations............................................................ 86
References..................................................................................................... 91
3.1 Introduction to Parallel Port

A parallel port is a type of interface found on computers for connecting various peripherals. In computing, a parallel port is a parallel communication physical interface. It is also known as a 'printer port' or 'Centronics port'. The IEEE 1284 standard defines the bi-directional version of the port, which allows the transmission and reception of data bits at the same time.

The original PC’s parallel port had eight outputs, five inputs, and four bidirectional lines. These are enough for communicating with many types of peripherals. In early parallel ports, the data lines were unidirectional (data out only) so it was not easily possible to feed data in to the computer. On many newer PCs, it can also serve as inputs, for faster communications with scanners, drives, and other devices that send data to the PC. However, a workaround was possible by using 4 of the 5 status lines. A circuit could be constructed to split each 8-bit byte into two 4-bit nibbles which were fed in sequentially through the status lines. Each pair of nibbles was then re-combined into an 8-bit (1-Byte). Such type of data transfer is called Nibble mode data transfer [1, 2].

Many manufacturers of personal computers and laptops consider parallel to be a legacy port and no longer include the parallel interface. For electronics hobbyists the parallel port is still often the easiest way to connect to an external circuit board. It is faster than the other common legacy port (serial port) and requires no serial-to-parallel converter, and requires far less interface logic and software than a USB target interface.

Port Types

As the design of the PC evolved, several manufacturers introduced improved versions of the parallel port. The new port types are compatible with the original design, but added new abilities, mainly for increasing speed. Speed is an important parameter. As computers and peripherals become faster, the jobs they do become more complicated, and the amount of information they need to exchange has increased. A fast interface
also makes it feasible to use portable, external version of peripheral, otherwise it has to be installed inside the computer.

**Original (SPP)**

The parallel port in the original PC is sometimes called the SPP, for standard parallel port, even though the original port had no written standard beyond the schematic diagrams and documentation for the IBM PC. The port in the original PC was based on an existing Centronics printer interface.

SPPs can transfer eight bits at once to a peripheral, using a protocol similar to that used by the original Centronics interface. The SPP does not have a byte-wide input port, but for PC-to-peripheral transfers, SPPs can use a Nibble mode that transfers each byte 4 bits at a time. Nibble mode is slow, but has become popular as a way to use the parallel port for input.

The following Fig. 3.1 and Fig. 3.2 show the standard parallel port DB25 and Centronics (36 pins) respectively. Table 3.1 shows the pinouts for parallel port connectors.

![Fig. 3.1: Parallel Port Pin Out (DB25)](image)

![Fig. 3.2: Parallel Port Pin Out (Centronics 36 pin)](image)
EPP

The EPP (enhanced parallel port) was originally developed by chip maker Intel, PC manufacturer Zenith, and Xircom, a maker of parallel-port networking products. As on the PS/2-type port, the data lines are bidirectional. An EPP can read or write a byte of data in one cycle of the ISA expansion bus, or about 1 microsecond, including handshaking as compared to four cycles for an SPP port. An EPP can switch directions quickly, so it is very efficient when used with disk and tape drives and other devices that transfer data in both directions. An EPP can also emulate an SPP.

ECP

The ECP (extended capabilities port) was first proposed by Hewlett Packard and Microsoft. Like the EPP, the ECP is bidirectional and can transfer data at ISA-bus speeds. ECPs have buffers and support for DMA (direct memory access) transfers and data compression. ECP transfers are useful for printers, scanners, and other peripherals that transfer large blocks of data. An ECP can also emulate an SPP port, and many ECPs can emulate an EPP as well.

Multi-mode Ports

Many newer ports are multi-mode ports that can emulate some or all of the above types. They often include configuration options that can make all of the port types available, or allow certain modes while locking out the others.

Addressing

The standard parallel port uses three contiguous addresses, usually in one of these ranges:

- 3BCh, 3BDh, 3BEh
- 378h, 379h, 37Ah
- 278h, 279h, 27Ah
The first address in the range is the port’s base address, also called the Data register or just the port address. The second address is the port’s Status register, and the third is the Control register. Table 3.2 shows the bit to pin mapping for the standard parallel port (SPP).

<table>
<thead>
<tr>
<th>Pin No (DB25)</th>
<th>Pin No (36 pin)</th>
<th>Signal name</th>
<th>Direction</th>
<th>Register - bit</th>
<th>Inverted</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Strobe</td>
<td>In/Out</td>
<td>Control-0</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Data0</td>
<td>Out</td>
<td>Data-0</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Data1</td>
<td>Out</td>
<td>Data-1</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Data2</td>
<td>Out</td>
<td>Data-2</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>Data3</td>
<td>Out</td>
<td>Data-3</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>Data4</td>
<td>Out</td>
<td>Data-4</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>Data5</td>
<td>Out</td>
<td>Data-5</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>Data6</td>
<td>Out</td>
<td>Data-6</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>Data7</td>
<td>Out</td>
<td>Data-7</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>Ack</td>
<td>In</td>
<td>Status-6</td>
<td>No</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>Busy</td>
<td>In</td>
<td>Status-7</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>Paper-Out</td>
<td>In</td>
<td>Status-5</td>
<td>No</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>Select</td>
<td>In</td>
<td>Status-4</td>
<td>No</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>Linefeed</td>
<td>In/Out</td>
<td>Control-1</td>
<td>Yes</td>
</tr>
<tr>
<td>15</td>
<td>32</td>
<td>Error</td>
<td>In</td>
<td>Status-3</td>
<td>No</td>
</tr>
<tr>
<td>16</td>
<td>31</td>
<td>Reset</td>
<td>In/Out</td>
<td>Control-2</td>
<td>No</td>
</tr>
<tr>
<td>17</td>
<td>36</td>
<td>Select-Printer</td>
<td>In/Out</td>
<td>Control-3</td>
<td>Yes</td>
</tr>
<tr>
<td>18-25</td>
<td>19-30,33,17,16</td>
<td>Ground</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Tab. 3.1: Pinouts for parallel port connectors

<table>
<thead>
<tr>
<th>Address</th>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base (Data port)</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Base+1 (Status port)</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>Base+2 (Control port)</td>
<td>13</td>
<td>2</td>
</tr>
</tbody>
</table>

~ indicates a hardware inversion of the bit.

Tab. 3.2: Bit to Pin Mapping for the Standard Parallel Port (SPP)
3.2 Design Description

As per the sampling theorem, the signal should be sampled at a frequency greater than twice the maximum frequency component present in the signal. If the signal is undersampled (i.e., sampled at a frequency less than twice the maximum frequency component in the signal), aliasing will occur [3]. Aliasing refers to reflection of high frequency components into low frequencies in the frequency spectrum. Aliasing results in error in frequency spectrum computation. To prevent aliasing, an anti-aliasing filter (a low-pass filter) may be used at the input stage. The input signal is passed through the filter. As the temperature of the room does not change abruptly, sampling rate is set at two samples per second. Sample rate can be changed by slide modification in the program according to our choice.

For single channel DAS, an ADC 0804 is used. ADC 0804 converts the analog signals to digital data. It is an 8-bit successive approximation type ADC and operates with +5 V supply voltage. It has $100\mu s$ conversion time at 640 kHz and operates within 100 kHz to 1460 kHz clock frequency [4]. This ADC is operated in free running mode, by connecting Pin no. 3 (WR) and Pin no. 5 (INTR) together. To trigger ADC conversion start, a switch is put in between INTR and GND. Clock frequency is adjusted by selecting proper values of the resistor and capacitor.

For eight channel DAS, ADC 0808 is used to convert the analog signals to digital data. It is an 8-bit successive approximation type ADC and operates with +5 V supply voltage. It has $100\mu s$ conversion time at 640 kHz and operates within 10 kHz to 1280 kHz clock frequency [5]. Successive approximation type ADC requires the analog input signal to be held constant, while conversion is going on. So, a sample-and-hold circuit is used to perform this operation using LF398 IC [6].

Since ADC0808 does not have internal clock generator, a clock circuit is made using an IC 7414 and one ceramic capacitor and one resistor. The design description for this is given in Chapter 2, Section 2.8, “Clock Signal Generator Circuit” [7].

The schematic diagrams of the circuits for experimental setup is shown in Fig. 3.3

---

Parallel Port Based DAS

3,2 Design Description

As per the sampling theorem, the signal should be sampled at a frequency greater than twice the maximum frequency component present in the signal. If the signal is under sampled (i.e. sampled at a frequency less than twice the maximum frequency component in the signal) aliasing will occur [3]. Aliasing refers to reflection of high frequency components into low frequencies in the frequency spectrum. Aliasing results in error in frequency spectrum computation. To prevent aliasing, an anti-aliasing filter (a low-pass filter) may be used at the input stage. The input signal is passed through the filter. As the temperature of the room does not change abruptly, sampling rate is set at two samples per second. Sample rate can be changed by slide modification in the program according to our choice.

For single channel DAS, an ADC 0804 is used. ADC 0804 converts the analog signals to digital data. It is an 8-bit successive approximation type ADC and operates with +5 V supply voltage. It has $100\mu s$ conversion time at 640 kHz and operates within 100 kHz to 1460 kHz clock frequency [4]. This ADC is operated in free running mode, by connecting Pin no. 3 (WR) and Pin no. 5 (INTR) together. To trigger ADC conversion start, a switch is put in between INTR and GND. Clock frequency is adjusted by selecting proper values of the resistor and capacitor.

For eight channel DAS, ADC 0808 is used to convert the analog signals to digital data. It is an 8-bit successive approximation type ADC and operates with +5 V supply voltage. It has $100\mu s$ conversion time at 640 kHz and operates within 10 kHz to 1280 kHz clock frequency [5]. Successive approximation type ADC requires the analog input signal to be held constant, while conversion is going on. So, a sample-and-hold circuit is used to perform this operation using LF398 IC [6].

Since ADC0808 does not have internal clock generator, a clock circuit is made using an IC 7414 and one ceramic capacitor and one resistor. The design description for this is given in Chapter 2, Section 2.8, “Clock Signal Generator Circuit” [7].

The schematic diagrams of the circuits for experimental setup is shown in Fig. 3.3
and Fig. 3.4 respectively. The detector unit is the temperature sensor IC LM35, which is pre-calibrated in $mV/^\circ C$ [8]. The amplifier unit consists of an OpAmp of low offset and high gain. It amplifies the analog voltage obtained from the detector unit up to a certain desired voltage level (i.e. 0 to 5 Volt, as the max input voltage of the ADC is $5 \pm 5\%$ Volt). If the sensor is kept beyond the specified range, the output at the amplifier will exceed the 5 volt limit and may generate error. This may lead to damage the ADC also. For safety purposes, a shunt Zener diode is connected across the output of the amplifier. It limits the max output voltage to 5.1 volt and thus protects the ADC.

The input analog signal so obtained is converted into 8-bit digital signal by the ADC in 256 steps. Thus the resolution of the ADC is $0.0196 \, V \approx 19.6 \, mV$. The digital data so obtained is fed into the PC through the parallel port. The operation of the ADC and Graphics Display with Data logging is controlled by using a programme written in “C” or ‘Visual Basic’ or ‘LabVIEW’. The display of the output in Graphical form is in real time mode and data is also saved in a file simultaneously for further use and analysis.

### 3.3 Hardware Design

The hardware design consists of the proper selection of the components as per the requirement, drawing of the schematics for the PCB layout. It involves component selection, component description and hardware details of the system design. Then, development of the circuit layout and circuit assembly followed by its testing is to be carried out.

#### 3.3.1 PCB Schematic

The schematic for the PCB layout for single channel parallel port based DAS is shown in Fig. 3.3 and for eight channel in Fig. 3.4. It is drawn by using PCB CAD tool called KiCAD.
Fig. 3.3: Schematic of the Single Channel Parallel Port based DAS

Fig. 3.4: Schematic of the Eight Channel Parallel Port based DAS
3.3.2 PCB Fabrication

PCB layout for the schematic Fig. 3.3 is made using CAD tools, as shown in Fig. 3.5. PCB layout is transferred to the glass epoxy copper clad by screen printing technique. Then, the printed PCB is allowed to etch in ferric chloride solution. After etching is over, holes are made by drilling with appropriate drill-bit sizes. Thus, the corresponding circuit assembly is made accordingly as shown in Fig. 3.6.

![Fig. 3.5: PCB Layout of the Single Channel Parallel Port based DAS](image1)

Like earlier, PCB layout for the schematic of Fig. 3.4 is made as shown in Fig. 3.7 and circuit assembly is made accordingly as shown in Fig. 3.8.

![Fig. 3.6: Single Channel Parallel Port based DAS](image2)
Fig. 3.7: PCB Layout of the Eight Channel Parallel Port based DAS

Fig. 3.8: Eight Channel Parallel Port based DAS
3.4 Software Design

For proper functioning of the designed DAS, we have to develop application program. Application program can be developed in various ways using various programming languages. We have tested our designed system by using application programs developed using ‘C’ or ‘Visual Basic’ or ‘LabVIEW’. For the parallel port based DAS, it does not require any firmware as no microcontroller is used. The following section discusses about the development of application program.

3.4.1 Application Program Development

Application program development involves the development of the logic of the programming for proper functioning of the system. It can be done in two steps (i) writing algorithm of the problem, then (ii) drawing the flowchart for this algorithm. In mathematics and computer science, an algorithm is a step-by-step procedure for calculations. Algorithms are used for calculation, data processing, and automated reasoning. More precisely, an algorithm is an effective method expressed as a finite list of well-defined instructions for calculating a function [9]. A flowchart is a type of diagram that represents an algorithm or process, showing the steps as boxes of various kinds, and their order by connecting these with arrows. This diagrammatic representation can give a step-by-step solution to a given problem. Process operations are represented in these boxes, and arrows connecting them represent flow of control. Data flows are not typically represented in a flowchart, in contrast with data flow diagrams; rather, they are implied by the sequencing of operations. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields [10]. If the development of algorithm and flowchart have been completed, then the corresponding program can be written by using any language. The following sections give the development of algorithm and flowchart for the application program to be written. Programs have been demonstrated by giving appropriate pseudocodes.
3.4.2 Algorithm

Algorithms of the application programs to be developed for Parallel Port based DAS are given below:

Algorithm for Single Channel using C

1. Start
2. Define Data, Status and Control Ports
3. Make the Parallel Port 'Bidirectional'
4. Input a filename and open it
5. Initialize Graphics mode
6. Read ADC data from the Parallel port Register
7. Compute pixel co-ordinates from the ADC value
8. Plot the data point with the pixel and join it with a line to the previous pixel data point
9. save the ADC data to the file
10. 'If Key pressed' go to step 11; else go to 6
11. Close the file
12. Stop
13. End

Algorithm for Eight Channel using C

1. Start
2. Define Data, Status and Control Ports
3. Make the Parallel Port 'Bidirectional'
4. Input a filename and open it
5. Initialize ADC Channel = 0
6. Send ADC Channel select signal
7. Send Start of Conversion (SOC) signal
8. wait 1 mS for EOC
9. Read ADC data from the Parallel port Register and store it to the varible
10. If channel < 8, increase ADC Channel by 1, then go to 6; else go to 11
11. Save the ADC data from the variables to file
12. 'If Key pressed' goto step 14; else goto 13
13. Reset ADC Channel to 0, then goto 6
14. Close the file
15. Stop
16. End

Algorithm for Single Channel using Visual Basic

1. Start
2. Define Data, Status and Control Ports
3. Make the Parallel Port 'Bidirectional'
4. A temporary file is created and open
5. Initialize Graphics mode
6. Read ADC data from the Parallel port Register
7. Compute pixel co-ordinates from the ADC value
8. If checkbox is checked, goto 9; else goto 10
9: Plot the data point with the pixel and join it with a line to the previous pixel data point
10. Save the ADC data to the file
11. If 'Save As' button pressed, goto 12; else goto 13
12. Save the file to a user specified filename and location
13. If 'Exit' button pressed, go to step 14; else go to 6
14. Close the file
15. Stop
16. End

Algorithm for Eight Channel using Visual Basic

1. Start
2. Define Data, Status and Control Ports
3. Make the Parallel Port 'Bidirectional'
4. A temporary file is created and open
5. Initialize ADC Channel = 0
6. Send ADC Channel select signal
7. Send Start of Conversion (SOC) signal
8. wait 1 mS for EOC
9. Read ADC data from the Parallel port Register and store it to the variable
10. If channel < 8, increase ADC Channel by 1, then go to 6; else go to 11
11. Save the ADC data from the variables to file
12. If 'Save As' button pressed, goto 13; else goto 14
13. Save the file to a user specified filename and location
14. If ‘‘Exit’’ button pressed go to step 16; else goto 13
15. Reset ADC Channel to 0, then goto 6
16. Close the file
17. Stop
18. End

Algorithm for Single Channel using LabVIEW

1. Start
2. Make the Parallel Port ‘‘Bidirectional’’
3. Read ADC data from the Parallel port Register
4. Plot the data point using the ‘‘chart vi’’
5. ‘‘STOP’’ Key pressed, go to step 5; else go to 2
6. Stop
7. End

3.4.3 Flowchart

Flowcharts, corresponding to the above algorithms, of the application programs to be developed for Parallel Port based DAS are given as

1. Fig. 3.9 represents the flowchart for the application program in ‘C’ for single channel parallel port based DAS,

2. Fig. 3.10 represents the flowchart for the application program in ‘C’ for eight channel parallel port based DAS,

3. Fig. 3.11 represents the flowchart for the application program in ‘Visual Basic’ for single channel parallel port based DAS,

4. Fig. 3.12 represents the flowchart for the application program in ‘Visual Basic’ for eight channel parallel port based DAS,
Fig. 3.9: Flowchart for the Application Program in C for Single Channel DAS

START

Define Data, Status and Control ports

Data port is set bidirectional

Input a filename and open it

Initialize Graphics

Read ADC data from Parallel Port register

Compute pixel coordinates from data

Plot the data point in the chart and join with previous data with line

Save the data to file

If Key pressed

Yes

Close the opened file

END
Fig. 3.10: Flowchart for the Application Program in C for Eight Channel DAS

START

Define data, status and control ports

Input a filename and open it

Initialize Graphics, Initialize Channel = 0, Set Data port to bidirectional

Select ADC channel, Send SOC signal, Wait for 1mS for EOC

Read ADC data from Parallel Port register and store it in variables

Increase Channel by 1

If channel < 8

Yes

Save the data from the variables to file

If Key pressed

Yes

Close the opened file

END

Reset Channel = 0

No

No
Fig. 3.11: Flowchart for the Application Program in VB for Single Channel DAS
Parallel Port Based DAS

```
START
Define Data, Status and Control ports
A temporary file is created and opened
Initialize Graphics, Initialize Channel = 0, Data port is set to bidirectional
Select ADC channel, Send SOC signal, Wait for 1mS for EOC
Read ADC data from Parallel Port register and store it in the variables
Increase Channel by 1
If Channel < 8
    Yes
    [If checkbox of the channels checked]
    Yes
    Plot the data points in the chart and join with previous data points with line
    No
    Save the data into the file
    If "Save As" button is pressed
    No
    If "EXIT" button is pressed
    Yes
    Close the opened file
STOP
```

Fig. 3.12: Flowchart for the Application Program in VB for Eight Channel DAS

70
3.4.4 Application Program in C

In computer science and numerical computation, pseudocode is an informal high-level description of the operating principle of a computer program or other algorithm. It uses the structural conventions of a programming language, but is intended for human reading rather than machine reading [11].

For single channel DAS, the ‘C’ pseudocode are as follows:

Main function of the application program in C

```c
void main()
{
  int i, time=61, newvolt=0, oldvolt=0, dat=0;
  int num=0, min=0;
  float yold, ynew, data, temp;
  char ch[10];
  FILE *fp;
  struct date d;
  struct time t;
  int gd=DETECT, gm;

  intro();

  printf("Enter a file name with extension (.txt):");
  scanf("%s", ch);
  fp=fopen(ch, "a");

  fprintf(fp, "Real Time Temperature Monitoring and Data Logging System");
  printf(fp, 

  fprintf(fp,"\nDate of Data Logging :
  %d-%d-%d\n", d.da_day, d.da_mon, d.da_year);

  time(&t);

  fprintf(fp,"\nStarting time of the Data Logging :
  %2d:%02d:%02d\n", t.ti_hour, t.ti_min, t.ti_sec);

  fprintf(fp, "\n_____________________________
" );
```

71
While (!kbhit())
{
    outportb(CONT, 0x20);
    outportb(CONT, 0x24);
    delay(200);
    dat = inportb(DATA);
    data = (float)(dat * 50) / 255;
    oldvolt = newvolt;
    newvolt = dat;
    setcolor(12);
    yold = (oldvolt * 199.24) / (127.0);
    ynew = (newvolt * 199.24) / (127.0);
    if (time <= 540)
        line(time, 430 - (int)yold, time + 4, 430 - (int)ynew);
    else
    {
        time = 61;
        clrscr();
        initgraph(&gd, &gm, "C:\\TC\\BGI");
        min++;
        drawchart(min);
    }
    num++;
    fprintf(fp, "\n%d\t\t%.2f\n", num, data);
    delay(300);
    time = time + 4;
}
gettime(&t);
closegraph();
fprintf(fp, "\n____________________________\n");
fprintf(fp, "\n\nEnd of the Reading of the Current Data Logging.\n");
fprintf(fp, "\n\nTime at the End of Data Logging : %2d:%02d:%02d\n", t.ti_hour, t.ti_min, t.ti_sec);
fprintf(fp, "\nProgramme Developed By:- N. Monoranjan Singh\n");
fprintf(fp, "\n\n
Programme Developed By:- N. Monoranjan Singh
");
fclose(fp);
Function for drawing graphical chart

void drawchart(int min)
{
    int x;
    float q;
    char b[120];
    outline();
    grid();
    setbkcolor(15);
    setcolor(4);
    outtextxy(170,465,"(REAL-TIME TEMPERATURE MONITORNG SYSTEM)");
    settextstyle(2,1,4);
    setcolor(1);
    outtextxy(15,150,"Temperature in Degree Celcious ( C )");
    settextstyle(2,0,0);
    outtextxy(225,448,"Time in Minutes (mm:ss)");
    setcolor(1);
    line(60,30,60,430); /* draw the x and y axis */
    line(60,430,540,430);
    for(q=5.0;q>=0.0;q--) /* draw y-axis graduation and calibration */
    {
        line(53,30+(5.0-q)*80,62,30+(5.0-q)*80);
        sprintf(b,"%1f",q*10);
        outtextxy(28,23+(5-q)*80,b);
    }
    for(x=0;x<60;x=x+10) /* draw x-axis graduation and calibration */
    {
        line(60+x*8,428,60+x*8,435);
        sprintf(b,"%d:%d",min,x);
        outtextxy(60+x*7.8,435,b);
    }
}
Function for drawing outline of the graphics

```c
void outline()
{
    setcolor(S);
    line(10,10,10,475);
    line(10,475,630,475);
    line(10,475,630,10);
    line(630,10,630,475);
}
```

Function for drawing gridline

```c
void grid()
{
    int x;
    float q;
    for(q=50;q>=0.0;q--) /* draw y-axis graduation and calibration */
    {
        line(58,30+(5.0-q/10)*80,52,30+(5.0-q/10)*80);
    }
    for(x=1;x<=60;x++) /* draw x-axis graduation and calibration */
    {
        line(60+x*8,428,60+x*8,432);
    }
}
```

For eight channel DAS, the ‘C’ code for the application program is as follows:-

Main function of the application program in C

```c
void main()
{
    int IN0, IN1, IN2, IN3, IN4, IN5, IN6, IN7 ,T=0;
    int J,K,L,M,N,O,P,Q;

    char filename[10];
    FILE*fp;
    clrscr();
    gotoxy(12,10);
```
puts("Enter a file name with extension .TXT and press ENTER key");
gotoxy(20, 12);
gets(filename);

fp=fopen(filename,"w");
clrscr();
if(fp==NULL)
{
    gotoxy(20,15);
    puts("cannot open a file 

");
    gotoxy(20,18);
    printf("press any key to exit...");
    getch();
    exit();
}
else

    fprintf(fp,"*************************\n");
    fprintf(fp,"8 - Channel Data Acquisition System\n");
    fprintf(fp,"*************************\n");
    fprintf(fp,"OUTPUT VOLTAGE IN MILLIVOLT TAKEN WITH 0.5 SECOND DELAY\n");
    fprintf(fp,"*************************\n");
    fprintf(fp,"\n TIME IN0 IN1 IN2 IN3 IN4 IN5 IN6 IN7\n");
    fprintf("\n\t TIME INQ\t INl\t IN2\t IN3\t IN4\t IN5\t
 IN6\t IN7\t\n ");

while(!kbhit())
{
    delay(100);
    outportb(PORT+2,0X2B); /*(1011)BI-DIRECTION AND
                           C0=SOC; C1=A, C2=B, C3=C
                           OF THE CONTROL PORT*/
    delay(1);
    outportb(PORT+2,0X2A); /*(1010)SOC IS KEPT HIGH FOR
                           INQ*/
    delay(1); /* WAIT FOR SOC PULSE WIDTH*/
outportb(PORT+2,0X2B); /*(1011)SOC IS KEPT LOW*/
delay(1); /*WAIT FOR THE EOC PROCESS*/
IN0=inportb(PORT); /*READING CHANNEL IN0*/
//J=(float)IN0*(5.0/255)*1000;/*DATA CONVERSION*/
outportb(PORT+2,0X29); /*(1001)SELECT CANNEL IN1*/
delay(1);
outportb(PORT+2,0X28); /*(1000)SOC IS KEPT HIGH FOR IN1*/
delay(1); /* WAIT FOR SOC PULSE WIDTH*/
outportb(PORT+2,0X29); /*(1001)SOC IS KEPT LOW*/
delay(1); /*WAIT FOR THE EOC PROCESS*/
IN1=inportb(PORT); /*READING CHANNEL IN1*/
//K=(float)IN1*(5.0/255)*1000;/*DATA CONVERSION*/
outportb(PORT+2,0X2F); /*(1111)SELECT CANNEL IN2*/
delay(1);
outportb(PORT+2,0X2E); /*(1110)SOC IS KEPT HIGH FOR IN2*/
delay(1); /* WAIT FOR SOC PULSE WIDTH*/
outportb(PORT+2,0X2F); /*(1111)SOC IS KEPT LOW*/
delay(1); /*WAIT FOR THE EOC PROCESS*/
IN2=inportb(PORT); /*READING CHANNEL IN2*/
//L=(float)IN2*(5.0/255)*1000;/*DATA CONVERSION*/
outportb(PORT+2,0X2D); /*(1101)SELECT CANNEL IN3*/
delay(1);
outportb(PORT+2,0X2C); /*(1100)SOC IS KEPT HIGH FOR IN3*/
delay(1); /* WAIT FOR SOC PULSE WIDTH*/
outportb(PORT+2,0X2D); /*(1101)SOC IS KEPT LOW*/
delay(1); /*WAIT FOR THE EOC PROCESS*/
IN3=inportb(PORT); /*READING CHANNEL IN3*/
//M=(float)IN3*(5.0/255)*1000;/*DATA CONVERSION*/
outportb(PORT+2,0X23); /*(0011)SELECT CANNEL IN4*/
delay(1);
outportb(PORT+2,0X22); /*(0010)SOC IS KEPT HIGH FOR IN4*/
delay(1); /* WAIT FOR SOC PULSE WIDTH*/
outportb(PORT+2,0x23); /*(1011)SOC IS KEPT LOW*/
delay(1); /*WAIT FOR THE EOC PROCESS*/
IN4=inportb(PORT); /*READING CHANNEL IN4*/
//N=(float)IN4*(5.0/255)*1000; /*DATA CONVERSION*/

outportb(PORT+2,0x21); /*(0001)SELECT CHANNEL IN5*/
delay(1); /* WAIT FOR SOC PULSE WIDTH*/
outportb(PORT+2,0x21); /*(1001)SOC IS KEPT LOW*/
delay(1); /*WAIT FOR THE EOC PROCESS*/
IN5=inportb(PORT); /*READING CHANNEL IN5*/
//O=(float)IN5*(5.0/255)*1000; /*DATA CONVERSION*/

outportb(PORT+2,0x27); /*(0011)SELECT CHANNEL IN6*/
delay(1); /* WAIT FOR SOC PULSE WIDTH*/
outportb(PORT+2,0x27); /*(1011)SOC IS KEPT LOW*/
delay(1); /*WAIT FOR THE EOC PROCESS*/
IN6=inportb(PORT); /*READING CHANNEL IN6*/
//P=(float)IN6*(5.0/255)*1000; /*DATA CONVERSION*/

outportb(PORT+2,0x25); /*(0010)SELECT CHANNEL IN7*/
delay(1); /* WAIT FOR SOC PULSE WIDTH*/
outportb(PORT+2,0x25); /*(1010)SOC IS KEPT LOW*/
delay(1); /*WAIT FOR THE EOC PROCESS*/
IN7=inportb(PORT); /*READING CHANNEL IN7*/
//Q=(float)IN7*(5.0/255)*1000; /*DATA CONVERSION*/

T=T+1;
fprintf(fp,"\n\t %d\t %d\t %d\t %d\t %d\t %d\t %d\t %d\t \n\t %d\t %d\t \n\t %d\t %d\t\n
77
3.4.5 Application Program in Visual Basic

For single channel DAS, the 'Visual Basic' pseudo code are as follows:

Function to read from the Parallel Port (Single Channel)

Public Sub PPRead()
    Do
        If stat = 0 Then : Exit Sub
        End If
        chdata = Inp(888)
        Delay(1)
        If acquirestatus = True Then
            If counter = gap Then
                start_ACQ()
                counter = 0
            ElseIf counter < gap Then
                counter = counter + 1
            ElseIf counter > gap Then
                counter = 0
            End If
        End If
    Loop
End Sub

Function to create chart

Private Sub DrawGuidelinesAndChart(ByVal PicBox As PictureBox)
    Dim tchdata As Single

tchdata = chdata * 1024 / 255
Dim bm As New Bitmap(PicBox.Width, PicBox.Height)
Dim gr As Graphics = Graphics.FromImage(bm)

For i As Integer = 48 To 400 Step 48
    gr.DrawLine(Pens.WhiteSmoke, PicBox.Width - Xmove, i,
                PicBox.Width, i)
Next i

NewValue1 = 1 + tchdata * 400 / 1024
gr.TranslateTransform(0, Picgraph.Height)
If CheckBox1.Checked = True Then
    gr.DrawLine(Pens.Black, _Picgraph.Width - 1 - Xmove,
                -OldValue1, _Picgraph.Width - 1, -NewValue1)
    OldValue1 = NewValue1
End If
Picgraph.Image = bm
gr.Dispose()      

End Sub

Function to create horizontal gridlines of the graph

Private Sub DrawGuidelines(ByVal PicBox As PictureBox)
    Dim bmp As New Bitmap(PicBox.Width, PicBox.Height)
    Dim gr As Graphics = Graphics.FromImage(bmp)
    For i As Integer = 48 To 400 Step 48
        gr.DrawLine(Pens.WhiteSmoke, 8, i, PicBox.Width, i)
    Next i
    PicBox.Image = bmp
End Sub

Function to create vertical values of the graph

Private Sub DrawVerticalValues(ByVal PB As PictureBox)
    Dim bmp As New Bitmap(PB.Width, PB.Height)
    Dim gr As Graphics = Graphics.FromImage(bmp)
    For i As Integer = 48 To 400 Step 48
        gr.DrawLine(Pens.WhiteSmoke, i, 8, PicBox.Width, i)
    Next i
    PicBox.Image = bmp
End Sub
Dim gv As Graphics = Graphics.FromImage(bmp)

For i As Integer = 40 To 400 Step 40
    gv.DrawLine(Pens.WhiteSmoke, 0, i, PB.Width, i)
Next i

Dim NextMarker As Double = 256
For i As Integer = 0 To 360 Step 40
    gv.DrawString(CStr(NextMarker), New Font("Verdana", 8, FontStyle.Regular), Brushes.Black, 1, i)
    NextMarker -= 25.6
Next

' Step 4
PB.Image = bmp
' Step 5
gv.Dispose()
End Sub

For eight channel DAS, the ‘Visual Basic’ pseudo code are as follows:-

Function to read from the Parallel Port (Eight Channel)

Public Sub PPRead()
    Dim data(8) As Integer
    Dim cont(8) As Single 'Control Signal
    cont(1) = 43 'OX2B); /*(1011)BI-DIRECTION and
        C0=SOC,C1=A,C2=B,C3=C of The Control Port
    cont(2) = 41 'OX29); /*(1001)SELECT CANNEL IN1
    cont(3) = 47 'OX2F); /*(1111)SELECT CANNEL IN2
    cont(4) = 45 'OX2D); /*(1101)SELECT CANNEL IN3
    cont(5) = 35 'OX23); /*(0011)SELECT CANNEL IN4
    cont(6) = 33 'OX21); /*(0001)SELECT CANNEL IN5
    cont(7) = 39 'OX27); /*(0111)SELECT CANNEL IN6
    cont(8) = 37 'OX25); /*(0101)SELECT CANNEL IN7
    Do
        For i = 1 To 6
            If stat = 0 Then : Exit Sub
        End If
            Out(820, cont(i))
        Next i
    Loop
End Sub
Delay(1)
Out(89®, cont(i) - 1)
Delay(1)
Out(89®, cont(i))
Delay(1)
chdata(i) = Inp(888)
If i = 8 Then
  i = 0
  Delay(1)
End If
If acquirestatus = True Then
  If counter = gap Then 'acquire and reset counter
    start_ACQ()
    counter = 0
  ElseIf counter < gap Then 'increment counter to attain count
    counter = counter + 1
  ElseIf counter > gap Then 'reset counter if selection is less than counter
    counter = 0
  End If
End If
Next

End Sub

Function for drawing graphical chart

Private Sub DrawGuidelinesAndChart(ByVal PicBox As PictureBox)
  Dim tchdata(8) As Single
  For i = 1 To 8
    tchdata(i) = chdata(i) * 1024 / 255
  Next

  Dim bm As New Bitmap(PicBox.Width, PicBox.Height)
  Dim gr As Graphics = Graphics.FromImage(bm)

  For i As Integer = 40 To 400 Step 40
gr.DrawLine(Pens.WhiteSmoke, PicBox.Width - Xmove, 
i, PicBox.Width, i)
Next i

gr.DrawImage(PicBox.Image, -Xmove, 0)

NewValue1 = 1 + tchdata(1) * 400 / 1024
NewValue2 = 1 + tchdata(2) * 400 / 1024
NewValue3 = 1 + tchdata(3) * 400 / 1024
NewValue4 = 1 + tchdata(4) * 400 / 1024
NewValue5 = 1 + tchdata(5) * 400 / 1024
NewValue6 = 1 + tchdata(6) * 400 / 1024
NewValue7 = 1 + tchdata(7) * 400 / 1024
NewValue8 = 1 + tchdata(8) * 400 / 1024

g.TranslateTransform(0, Picgraph.Height)

If CheckBox1.Checked = True Then
    gr.DrawLine(Pens.Black, _Picgraph.Width - 1 
    -Xmove, -OldValue1, _Picgraph.Width - 
    1, -NewValue1)
    OldValue1 = NewValue1
End If
If CheckBox2.Checked = True Then
    gr.DrawLine(Pens.Blue, _Picgraph.Width - 1 
    -Xmove, -OldValue2, _Picgraph.Width - 
    1, -NewValue2)
    OldValue2 = NewValue2
End If
If CheckBox3.Checked = True Then
    gr.DrawLine(Pens.Red, _Picgraph.Width - 1 
    -Xmove, -OldValue3, _Picgraph.Width - 
    1, -NewValue3)
    OldValue3 = NewValue3
End If
If CheckBox4.Checked = True Then
    gr.DrawLine(Pens.Green, _Picgraph.Width - 1 
    -Xmove, -OldValue4, _Picgraph.Width - 
    1, -NewValue4)
    OldValue4 = NewValue4
End If
If CheckBox5.Checked = True Then
    OldValue5 = NewValue5
End If
If CheckBox6.Checked = True Then
    OldValue6 = NewValue6
End If
If CheckBox7.Checked = True Then
    OldValue7 = NewValue7
End If
If CheckBox8.Checked = True Then
    OldValue8 = NewValue8
End If

Picgraph.Image = bm
gr.Dispose()
3.4.6 Application Program in LabVIEW

**Fig. 3.13:** LabVIEW Program for measuring temperature

**Fig. 3.14:** LabVIEW Program for measuring light intensity
LabVIEW is a powerful system design software built specifically for tasks performed by engineers and scientists. LabVIEW stands for Laboratory Virtual Instrumentation Engineering Workbench. Since it is a system design platform and development environment for a visual programming language from National Instruments, the graphical language is named ‘G’. LabVIEW is commonly used for data acquisition, instrument control, and industrial automation on a variety of operating systems including Microsoft Windows.

Fig. 3.13 and Fig. 3.14 are the G-program called ‘vi’ (virtual instrumentation) for measuring temperature and light intensity respectively.

Fig. 3.15 and Fig. 3.16 represent the screen shot of the Visual Basic program for single channel and eight channel DAS respectively.

Fig. 3.15: Screen shot of the Visual Basic Program for Single Channel DAS
3.5 Experimental Observations

Light source for different colours (wavelength) have been designed using array of LEDs, with different colours as shown in Fig. 3.17. These light sources are used to study the characteristics of light intensities for different colours using the designed DAS.
Fig. 3.18: Screen shot of the online temperature monitoring by Single Channel DAS, using C Program

Fig. 3.19: Screen shot of the online temperature monitoring by Single Channel DAS, using LabVIEW Program
Fig. 3.20: Light intensity measurement by Luxmeter vs DAS for Red Light

LED Red

\[ y = 2E^{-08}x^3 - 1E^{-04}x^2 + 0.2137x + 73.271 \]

Fig. 3.21: Light intensity measurement by Luxmeter vs DAS for Yellow Light

LED Yellow

\[ y = 4E^{-06}x^2 - 0.0002x^2 + 0.2631x + 56.227 \]
Parallel Port Based DAS

LED Blue

\[ y = -6 \times 10^{-11}x^3 + 8 \times 10^{-07}x^2 + 0.0106x + 93.357 \]

Fig. 3.23: Light intensity measurement by Luxmeter vs DAS for Blue Light

LED Green

\[ y = 2 \times 10^{-07}x^2 - 0.0006x^2 + 0.5222x + 31.313 \]

Fig. 3.22: Light intensity measurement by Luxmeter vs DAS for Green Light

89
**Results and Conclusion**

Fig. 3.18 and Fig. 3.19 shows the screenshot for recording of online temperature monitoring using ‘C’ program and program developed in LabVIEW respectively. Both the two output shows same result. The temperature is also recorded in the hard disk of the computer simultaneously, for ‘C’ program [12].

Experimental observations are taken for red, yellow, green and blue colour lights. Their characteristic response with intensities are shown in Fig. 3.20, Fig. 3.21, Fig. 3.22, Fig. 3.23 respectively. It shows that, the spectral responses of the light intensities are almost similar in visible region spectrum.
REFERENCES


