OMR SCANNER

By

MEENU SHARMA-031010
SHARMISHTHA GUPTA-031011
(HEADED BY Dr. SARIT PAL)

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

MAY - 2007

Submitted in partial fulfillment of the Degree of Bachelor of Technology

DEPARTMENT OF ELECTRONICS AND COMMUNICATION
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY-WAKNAGHAT

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
(Established by H.P. State Legislative vide Act No. 14 of 2002)
Waknaghat, P.O. Dumehar Bani, Kandaghat, Distt. Solan - 173215 (H.P.) INDIA
Website : www.juit.ac.in
Phone No. (91) 01792-257999 (30 Lines).
Fax: (91) 01792 245362

Dr. Sarit Pal

Assistant Professor

Department of Electronics and Communication Engineering

CERTIFICATE

This is to certify that the work entitled, "low cost OMR scanner" submitted by Meenu Sharma(031010) and Sharmishtha Gupta(031011) in partial fulfillment for the award of degree of Bachelor of Technology in Electronics and Communication of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Sarit Pal

[Sarit Pal]

Supervisor

ACKNOWLEDGEMENT

We are highly thankful to our project head Dr. Sarit Pal who not only helped us develop the concept but also guided us throughout the whole time period of making the project. He has been an immense source of inspiration and an able guide due to whom we have been able to complete the project successfully. We are also thankful to our college lab technicians especially Mr. Pramod and Mr. Manoj Pande and our colleagues who gave us valuable help whenever we needed it.

[illegible] Sharmishtha

Meenu Sharma Sharmishtha Gupta

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

MCQ	Multiple Choice Question
OMR	Optical Mark Reader

ABSTRACT

Optical Mark Reader (OMR) is thoroughly reliable and highly efficient provided that high standards are maintained at both the planning and implementation stages. It is necessary to ensure that OMR forms are designed with due attention to data integrity checks, the best use is made of features built into the OMR, used data integrity is checked before the data is processed and data is validated before it is processed. This project paper describes the design and implementation of a low cost OMR prototype system for marking multiple-choice tests automatically.

Chapter 1
INTRODUCTION

1.1 OBJECTIVE

The main purpose of the project is to read the answers of MCQs marked on an A4 sized sheet and then compute the marks scored based on keyboard input correct answers. The options as answer to the questions are in the form of bubbles marked in front of the question numbers.

The format of the answer sheet is as follows:

	A	B	C	D
1.	O	O	O	O

The correct answer option (as per the person marking the sheet) will have to be darkened, which will be read and compared by the device with the fed data and later the score will be calculated by a computer program.

1.2 CONCEPTS OF OMR

OMR refers to the technique of converting handwritten marks into an ASCII value. The Mark is created by filling a circle or a box on a pre-printed form.

An Optical Mark Reader (OMR) is simply a machine that takes sheet paper and records whether marks appear on the paper in predefined areas. OMR machines are not scanners in the sense that they do not form an image of the sheets that pass through. Instead, the OMR device simply detects whether predefined areas are blank or have been marked.

The fact that OMRs don't form an image differentiates them from desktop scanners that are often used by departments to mark MCQ examinations.

Optical reading of documents is a tried and tested technique which saves you a considerable amount of time and effort. The user had to print documents according to strict norms after which a special file, detailing the structure of the different pages had to be programmed for the optical reader.

Once the documents had been read the data is transferred to a software program especially adapted, in order to make use of the information.

OMR technology detects the absence or presence of a mark, but not the shape of the mark. Forms are scanned through an OMR scanner. The forms contain small circles, referred to as 'bubbles' that are filled in by the respondent. OMR cannot recognize hand-printed or machine-printed characters; the scanner does not create an image of the form.

1.3 OMRs IN THE MARKET

OMR (optical mark read) scanning continues to stand as the most reliable, accurate, and cost-effective way to extract information from a user group, collect and collate it, and deliver it to decision-makers and other stake-holders in a timely, meaningful way.

One of the key components in any OMR data collection system is the software that operates the OMR scanner. The design of document scanning software and how easy it can be used generally is a very good indication on how effective it may be in capturing data using OMR scanners.

There are different types of models available in the market having the following characteristics:-

- Optical character recognition (OCR) products are used to read text from paper and translate it into data that can be manipulated by computers.
- Gray scale optical mark reader.
- Some equipped have two types of reading a mark reader and an optical character reader.

Chapter 2
BASIC MODEL OF THE PROPOSED SYSTEM

2.1 BASIC CONCEPT

The basic model of the mark sensor involves scanning the page and reading the data which is further processed to generate the total marks scored.

The answers have been marked in the form of darkened bubbles. This answer sheet is placed on a glass platform with light source above it, then it is scanned and read by using a sensing device comprising LDRs (light dependent resistors) which change resistance depending on whether sensing a darkened or non-darkened bubble.

The sensing device is being controlled by a rotor which is receiving input from the output port RS232, computer programmed for the same. The data received from the sensing device in the form of voltage is further fed to the computer port RS232 through D25 connector.

The aforesaid data is processed through a computer program which receives input from the port RS232 and compares it with the computer fed input to compute the total marks scored. This result is displayed on the computer screen.

2.2 BLOCK DIAGRAM

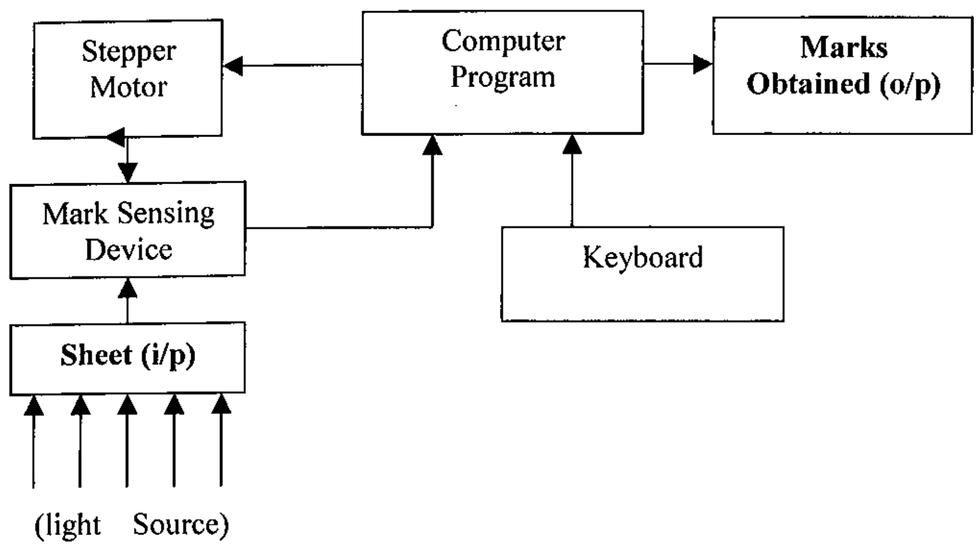

Fig.2-1

In the above block diagram, the sheet used in the input to the mark sensing device is illuminated by light. The stepper motor is controlled by a computer program which further controls the mark sensing device. The output of the mark sensing device is fed to the computer which also receives input from the computer keyboard and ultimately generates output in the form of total marks obtained.

Chapter 3
SENSING THE MARKS

The sensor for marks comprises of 4 LDRs, with one LDR placed below each of the 4 bubbles as answer options of a question. The sheet is placed on a glass platform illuminated by a light source from above. The LDR receives different light from darkened and non-darkened bubble and hence exhibits different resistance.

LIGHT DEPENDENT RESISTOR

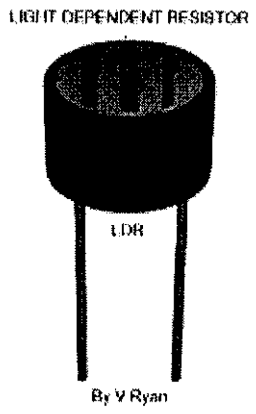

Fig.3-1

In the dark, the resistance of the LDR is very high, typically around 1M ohm. In bright light it is low, typically 1K ohm. An example of the peak spectral response of the LDR (VT936G from EG&G) is 550nm. The continuous power dissipation is 80mW and the maximum voltage, which can be applied to it, is 100V.

Using the circuit in fig. 3-2 we are able to receive different levels from marked and unmarked options.

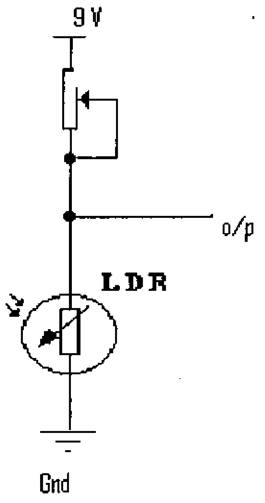

Fig.3-2

As the LDR changes resistance the change in potential is detected by the circuit. The output voltage is fed as input to the computer through RS232.

Chapter 4
MECHANICAL MOTION FOR SCANNING

The mechanical motion required for scanning the page is being done through a motor receiving input from the computer through port RS232. The rotor has a toothed wheel attached to it. These teeth are used to move a toothed strip which has sensing device attached perpendicularly to it.

When the device is OFF the sensing device is at the start of page. As we switch ON the mark sensor, the sensing device is moved just enough to reach the first marked answer. This answer is sensed and then the rotor revolves, just enough so that the sensing device is placed above the second answer and so on. After scanning and reading the entire answer sheet, the rotor revolves in the opposite direction to place the sensing device at its initial position.

The motion of the rotor has been programmed based on the following parameters:

1. Distance between answers.
2. Number of steps required to produce the required lateral motion.

STEPPER MOTOR – Stepper motor is an electromagnetic actuator. It is an incremental drive (digital) actuator and is driven in fixed angular steps. This means that a digital signal is used to drive the motor and every time it receives a digital pulse it rotates a specific number of degrees in rotation.

•Each step of rotation is the response of the motor to an input pulse (or digital command).

•Step-wise rotation of the rotor can be synchronized with pulses in a command-pulse train, assuming that no steps are missed, thereby making the motor respond faithfully to the pulse signal in an open-loop manner.

.

STEPPER MOTOR BASICS

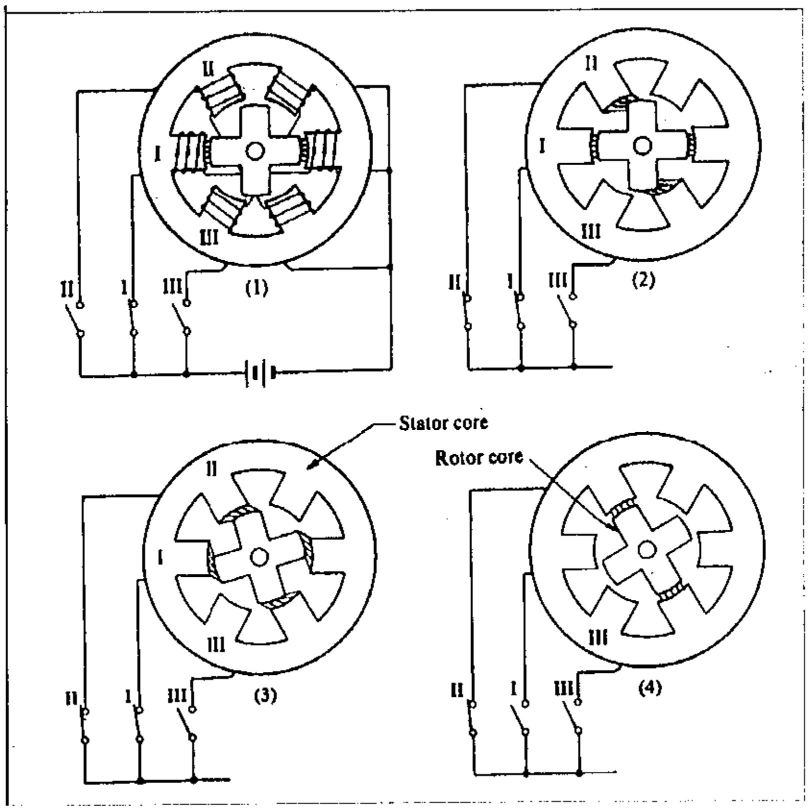

Fig.4-1 STEPPER MOTOR STATES FOR MOTION

The figure in the previous page is the cross-section view of a single-stack variable-reluctance motor. The inner device is called the rotor and has four poles. Both the stator and rotor are made of soft steel. The stator has three sets of windings as shown in the figure. Each set has two coils connected in series. A set of windings is called a "phase". The motor above, using this designation, is a three-phase motor. Current is supplied from the DC power source to the windings via the switches I, II, and, III.

Starting with state (1) in the upper left diagram, note that in state (1), the winding of Phase I is supplied with current through switch I. This is called in technical terms, "phase I is excited". Arrows on the coil windings indicate the magnetic flux, which occurs in the air-gap due to the excitation. In state I, the two stator poles on phase I being excited are in alignment with two of the four rotor teeth. This is an equilibrium state.

Next, switch II is closed to excite phase II in addition to phase I. Magnetic flux is built up at the stator poles of phase II in the manner shown in state (2), the upper right diagram. A counter-clockwise torque is created due to the "tension" in the inclined magnetic flux lines. The rotor will begin to move and achieve state (3).

When switch I is opened to de-energize phase I, the rotor will travel and reach state (4). The angular position of the rotor can thus be controlled in units of the step angle by a switching process. If the switching is carried out in sequence, the rotor will rotate with a stepped motion; the switching process can also control the average speed.

STEP ANGLE

The step angle, the number of degrees a rotor will turn per step, is calculated as follows:

Step angle (Θs)=360°/S

S=mNr

m=number of phases

Nr=number of rotor teeth

For this motor:

m=3

Nr=40

S= mNr =3*40=120

Θs =360°/120=3.0°

360° (one rotational motion) = 2.4 inch (lateral motion)

Distance between two answers = 1 inch

Degree required to move from one answer to another = 360°/2.4 =150°

Hence no. of steps required = 150°/3.0° = 50

BASIC WIRING DIAGRAM

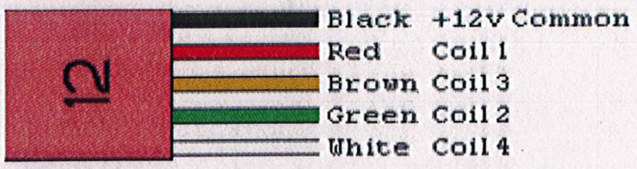

Fig.4-2

Two phase stepper-motor wiring diagram

The motor in the previous figure is a two-phase motor. This is sometimes called UNIPOLAR. The two-phase coils are center-tapped and in this case they the center-taps are connected to ground. The coils are wound so that current is reversed when the drive signal is applied to either coil at a time. The north and south poles of the stator phases reverse depending upon whether the drive signal is applied to coil 1 as opposed to coil 2.

STEP SEQUENCING

There are three modes of operation when using a stepper motor. The mode of operation is determined by the step sequence applied. The three step sequences are:

Full H = HIGH = +V

L=LOW=0V

WAVE STEPPING

The wave stepping sequence is shown below.

STEP	L1	L2	L3	L4
1	H	L	L	L
2	L	H	L	L
3	L	L	H	L
4	L	L	L	H

Wave stepping has less torque then full stepping. It is the least stable at higher speeds and has low power consumption. Clock-wise and counter clockwise rotation is accomplished by reversing the step sequence.

DRIVING CIRCUIT FOR STEPPER MOTOR

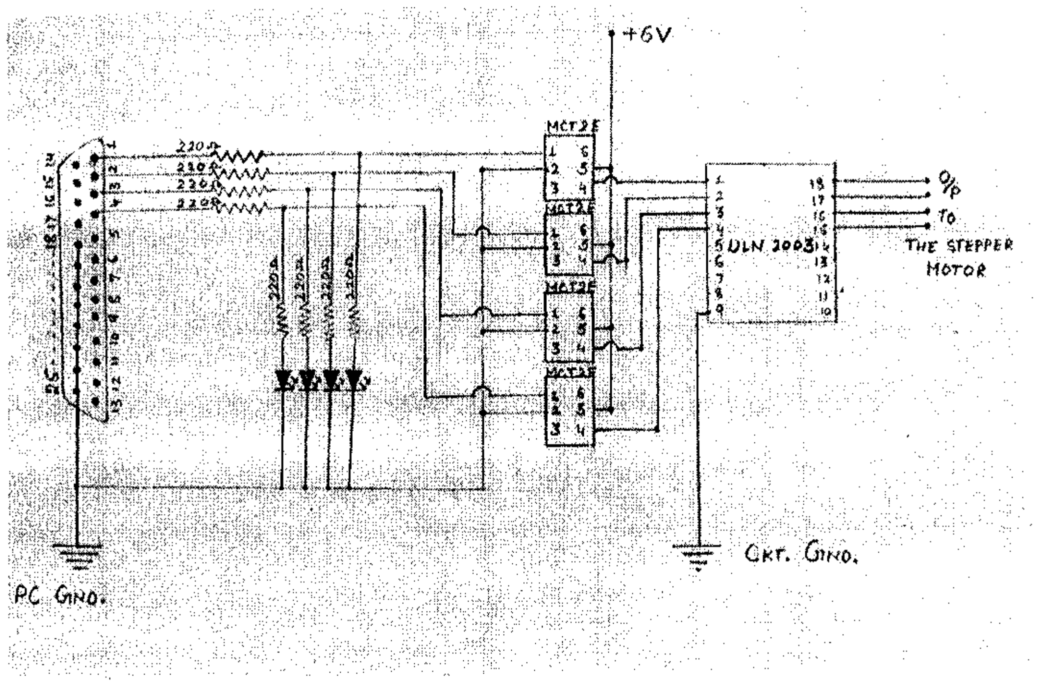

Fig.4-3

Chapter 5

DATA PROCESSING

5.1 INTERFACING WITH COMPUTER

The data received as input from the mark sensor in the form of voltage is fed to the computer through port RS232 through D25 connector.

Interfacing to the Parallel Printer Port

The Parallel Printer Port had a total of 12 digital outputs and 5 digital inputs accessed via 3 consecutive 8-bit ports in the processor's I/O space.

- 8 output pins accessed via the **DATA Port**
- 5 input pins (one inverted) accessed via the **STATUS Port**
- 4 output pins (three inverted) accessed via the **CONTROL Port**
- The remaining 8 pins are grounded

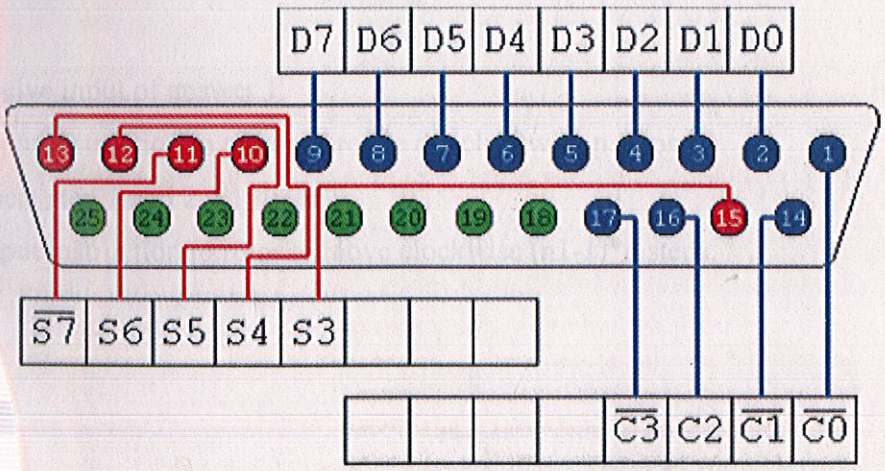

Fig.5-1 25-way Female D-Type Connector

5.2 PROGRAMMING

A computer program is used to run the stepper motor as well as to receive the data and process it. The program outputs instructions to the mark sensor located at position above the first answer to sense the data and takes it as input. It then instructs the stepper motor to move anticlockwise so as that lateral motion of the sensor places it above answer 2. After this answer is sensed the stepper motor is again instructed to rotate, so that the sensor reaches question 3 and so on. After the last question is reached, the rotor moves clockwise till the sensor reaches its initial position above answer 1.

Algorithm:

Let no of questions be n1

Distance between two questions x units

And n be the no. of steps required for the stepper motor to move the sensor laterally x units.

1. Receive input of answer 1.
2. Output instruction to motor to rotate anticlockwise n steps.
3. Repeat step 1 and 2, n1 times.
4. Output instruction to rotor to move clockwise (n1-1)*n steps.

Chapter 6
WORKING

The model comprises of a sensing plate with four separate sensing circuits one below each bubble which are options of an answer. The answer sheet is placed on a glass platform which is illuminated by light from the above side. The sensing device is on the below of the sheet.

The sensing device uses LDRs to sense which option has been darkened. The LDRs exhibits different voltages based on the light intensity projected on it. Since a dark and non darkened bubble pass different intensity of light, the LDR below a dark and a non dark bubble exhibit different resistance. This chance in resistance leads to change voltage. **The different voltages resulting due to darkened and non-darkened bubbles act as output of the sensing device.**

This sensing device is attached to a toothed plate. The sensing plate is perpendicular to the toothed plate. This toothed plate is placed close to a toothed wheel attached to the stepper motor. As the stepper motor rotates the toothed wheel rotates too. The teeth of this wheel fit into the teeth of the toothed plate, thus making this plate move laterally. With the lateral motion of the toothed plate, the sensing plate attached perpendicularly to it also moves. **Thus the rotational motion of the stepper motor creates lateral motion of the sensing plate.**

In off condition the sensing device lays some steps above the first answer option set of the answer sheet. Upon switch on firstly it moves to the first answer option set and then senses the first answer marked. Then the stepper motor rotates, thus moving the sensor to second answer. When this answer is sensed, the rotor moves the sensing plate to third answer.

This procedure is followed till all the answers are read. Now the stepper motor moves in the reverse direction thus placing the sensing device in its initial position. **The motion of the stepper motor, as well as the processing of the output from the sensing device is done through a computer program.**

The output of the sensing device is fed as parallel input to the computer through computer port RS232, using a D25 connector. **The sensing device voltage output is taken as input to the port, while the instructions to the steeper motor are the output through the port.**

The computer program receives the examinees marked answers as input through the port, which is compared to the actual answers input through the keyboard. The instructions to the stepper motor are also given through the program. This is done in a particular order. After sensing the each answer and storing the input, the stepper motor is instructed to move a particular no. of steps. This is done till all answers have been read. Now the stepper motor is instructed to move back to its initial position. Now the result is computed in the form of total marks scored. This is done by comparing the original answer set answered marked answer set.

Chapter 7
CONCLUSION

Optical Mark Reader (OMR) scanning continues to stand as the most reliable, accurate, and cost-effective way to extract information from a user group, collect and collate it, and deliver it to decision-makers and other stake-holders in a timely, meaningful way.

One of the key components in any OMR data collection system is the software that operates the OMR scanner. The design of document scanning software and how easy it can be used generally is a very good indication on how effective it may be in capturing data using OMR scanners. Besides this cost is also a major factor at least in schools and other institutions which are not able to bear the cost of the OMRs available in the market. These institutions do not need very sophisticated OMRs but cheap and simple devices which can ease off the burden of evaluating tests. The device developed by us serves this purpose. It is cheap and easy to use and thus can be of immense use in schools and universities for evaluation of Multiple Choice Questions.

Appendix A:

```
//ScanDlg.cpp:implementation file

#include "stdafx.h"
#include "Scan.h"
#include "ScanDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

int LINES;
int DATAin;
int DATAout;

#define UP 0
#define DOWN 1

#define PORTin 0x379
#define PORTout 0x378

int ERR_Status=0;
int SOL[50], SOL_Scanned[50], RESULT[50];
CString SolStr;
```

```
/* ----Prototypes of Inp and Outp--- */
short _stdcall Inp32(short PortAddress);
void _stdcall Out32(short PortAddress, short data);
/*--------------------------------*/
////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
 public:
        CAboutDlg();   // Dialog Data
        enum { IDD = IDD_ABOUTBOX };

   // ClassWizard generated virtual function overrides
  //{{AFX_VIRTUAL(CAboutDlg)

protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support
  // Implementation
protected:
        //{{AFX_MSG(CAboutDlg)
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
        //{{AFX_DATA_INIT(CAboutDlg)
        //}}AFX_DATA_INIT
}
```

```
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CAboutDlg)
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
        //{{AFX_MSG_MAP(CAboutDlg)
                // No message handlers
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CScanDlg dialog

CScanDlg::CScanDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CScanDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CScanDlg)
        //}}AFX_DATA_INIT
        // Note that LoadIcon does not require a subsequent DestroyIcon in
        Win32
        m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}
void CScanDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CScanDlg)
        DDX_Control(pDX, IDC_SOL, m_SOL);
        //}}AFX_DATA_MAP
}
```

```
BEGIN_MESSAGE_MAP(CScanDlg, CDialog)
        //{{AFX_MSG_MAP(CScanDlg)
        ON_WM_SYSCOMMAND()
        ON_WM_PAINT()
        ON_WM_QUERYDRAGICON()
        ON_BN_CLICKED(IDC_SC, OnScan)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CScanDlg message handlers
BOOL CScanDlg::OnInitDialog()
{
        CDialog::OnInitDialog();
        // Add "About..." menu item to system menu.
        // IDM_ABOUTBOX must be in the system command range.
        ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
        ASSERT(IDM_ABOUTBOX < 0xF000);
        LINES=8;
        SetDlgItemText(IDC_LINES,"8");
        CMenu* pSysMenu = GetSystemMenu(FALSE);
        if (pSysMenu != NULL)
        {
             CString strAboutMenu;
             strAboutMenu.LoadString(IDS_ABOUTBOX);
             if (!strAboutMenu.IsEmpty())
             {
                    pSysMenu->AppendMenu(MF_SEPARATOR);
               pSysMenu->AppendMenu(MF_STRING,IDM_ABOUTBOX,
               strAboutMenu);
             }
        }
        // Set the icon for this dialog.  The framework does this automatically
        //  when the application's main window is not a dialog
        SetIcon(m_hIcon, TRUE);                    // Set big icon
        SetIcon(m_hIcon, FALSE);            // Set small icon
              // TODO: Add extra initialization here
              return TRUE;  // return TRUE  unless you set the focus to a control
}

void CScanDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
        if ((nID & 0xFFF0) == IDM_ABOUTBOX)
        {
              CAboutDlg dlgAbout;
              dlgAbout.DoModal();
        }
        else
        {
              CDialog::OnSysCommand(nID, lParam);
        }
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.
```

```
void CScanDlg::OnPaint()
{
        if (IsIconic())
        {
                CPaintDC dc(this); // device context for painting

                SendMessage(WM_ICONERASEBKGND, (WPARAM)
dc.GetSafeHdc(), 0);

                // Center icon in client rectangle
                int cxIcon = GetSystemMetrics(SM_CXICON);
                int cyIcon = GetSystemMetrics(SM_CYICON);
                CRect rect;
                GetClientRect(&rect);
                int x = (rect.Width() - cxIcon + 1) / 2;
                int y = (rect.Height() - cyIcon + 1) / 2;

                // Draw the icon
                dc.DrawIcon(x, y, m_hIcon);
        }

        else
        {
                CDialog::OnPaint();
        }
}
```

```
// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CScanDlg::OnQueryDragIcon()
{
          return (HCURSOR) m_hIcon;
}

void CScanDlg::OnScan()
{
          // TODO: Add your control notification handler code here
          int i;
          ERR_Status=0;
          UpdateData();
          LINES=GetDlgItemInt(IDC_LINES,NULL,FALSE);
          m_SOL.GetWindowText( SolStr );

 /*  Convert Textual Solution to numbers */

          if(LINES != SolStr.GetLength() )
          {
                AfxMessageBox("ERROR !! \r\n\r\nSolutions provided mismatch
                with total lines",MB_OK,NULL);
                        ERR_Status=1;
          }
          else
          {
                for(i=0;i<LINES;i++)
                {
                        switch(SolStr.GetAt(i))
                        {
```

```
                        case 'a':
                        case 'A':       SOL[i]=8;
                                        break;
                        case 'b':
                        case 'B':       SOL[i]=4;
                                        break;
                        case 'c':
                        case 'C':       SOL[i]=2;
                                        break;
                        case 'd':
                        case 'D':       SOL[i]=1;
                                        break;
                        default:        ERR_Status=2;
                }
        }
}

if(! ERR_Status)
{
        /* Show Original Pattern */
        CString str="\r\n\r\n";
        for(i=0;i<LINES;i++)
        {
                switch(SOL[i])
                {
                        case 8: str+=" 1  0  0  0\r\n";
                                break;
                        case 4: str+=" 0  1  0  0\r\n";
                                break;
                        case 2: str+=" 0  0  1  0\r\n";
                                break;
```

```
                         case 1: str+=" 0  0  0  1\r\n";
                                 break;
                 }
         }

         SetDlgItemText(IDC_PATTERN,str);
         /*  SCAN the sheet */
         //CString msg="\r\n\r\n\r\n\r\nScanning !!";
         //SetDlgItemText(IDC_SCANNED,msg);

         Step(150,UP);
         _sleep(2);

         DATAin=(~Inp32(PORTin));
         DATAin&=120;
         DATAin>>=3;

      SOL_Scanned[0]=DATAin;
      for(i=0;i<=LINES-1;i++)
      {
        Step(50,UP);
        _sleep(4);
           DATAin=(~Inp32(PORTin));
           DATAin&=120;
           DATAin>>=3;

           SOL_Scanned[i]=DATAin;
      }

      Step(150 + (LINES-1)*50,DOWN);
//Take the header back to the original position (ie. at the start)
```

```
/* Show Scanned Pattern */
        str="\r\n\r\n";
        for(i=0;i<LINES;i++)
        {
          switch(SOL_Scanned[i])
          {
            case 8: str+=" 1  0  0  0\r\n";
                    break;
            case 4: str+=" 0  1  0  0\r\n";
                    break;
            case 2: str+=" 0  0  1  0\r\n";
                    break;
            case 1: str+=" 0  0  0  1\r\n";
                    break;
            case 0: str+=" 0  0  0  0\r\n";
                    break;
        }
      }

        SetDlgItemText(IDC_SCANNED,str);
        ShowResult();  //display the result in the result box
  }
        else
        {
          if(ERR_Status==2)
          AfxMessageBox("ERROR !! \r\n\r\nInvalid options provided in the
Solution string",MB_OK,NULL);
        }
}
```

```
//GetDlgItem(IDC_BTN_STOP)->EnableWindow( FALSE );
//SetDlgItemText(IDC_E1,"OFF");
void CScanDlg::Step(int S,int dir)
{
           int i=0;
           int DELAY=20;
           while(1)
           {
              Out32(PORTout,DATAout);
              _sleep(DELAY);

             if(i==S-1) break;
               if(dir==0)       // UP (clockwise direction)
                 DATAout<<=1;
              else    // DOWN (counter-clockwise direction)
                 DATAout>>=1;
             if( (DATAout&15) == 0)
             {
                if(DATAout==16) DATAout=1;
                else DATAout=8;
             }
            i++;
           }}
```

```
void CScanDlg::ShowResult()
{
        int TOTAL=0;

        for(int i=0;i<LINES;i++)
        {
          if(SOL[i]==SOL_Scanned[i])
          {
            TOTAL++;
             RESULT[i]=1;
          }
         else
         {
                    RESULT[i]=0;
            }}

        CString Res;
        Res.Format("%d / %d",TOTAL,LINES);
        SetDlgItemText(IDC_RESULT,Res);
}
```

Appendix B:

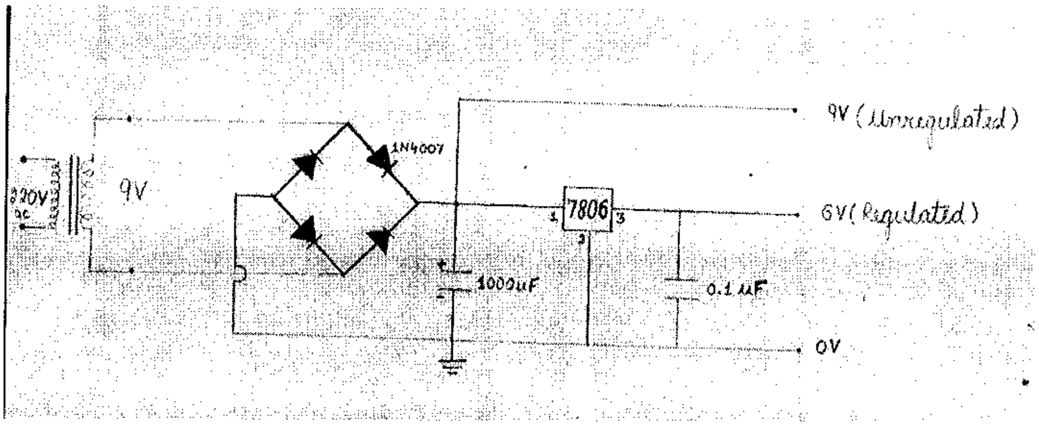

Fig.B-1 Generation of 9V and 6V DC power supply

BIBLIOGRAPHY

- http://www.leadtools.com/home2/VertMkts/LTProdOvrvw.htm
- http://www.veltronics.com/omrprocessor.html#m1
- http://www.datablocks.com/default.html
- http://www.pearsongov.com/news/index.htm?1180258231374
- http://www.imagesco.com/articles/picstepper/02.html
- http://www.electronics-project-design.com/ConformalCoating.html
- http://www.technologystudent.com/index.htm
- http://electronics-diy.com/schematics.php
- www.doctronics.co.uk/ldr_sensors.htm
- http://www.arcelect.com/rs232.htm
- http://www.springerlink.com/content/w578384p3v8u5382/
- http://www.electronic-circuits-diagrams.com/index.shtml
- http://www.allaboutcircuits.com/vol_3/chpt_3/4.html