INNOVATIVE STANDALONE EMBEDDED FILE COPIER FROM CAMERA/PENDRIVE

A PROJECT REPORT

Submitted by

BINCY P ANDREWS

NEETHU NICLAVOSE

RESHMA VARGHESE

TINU MARY C D

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY (B.TECH)

in

COMPUTER SCIENCE & ENGINEERING

of

UNIVERSITY OF CALICUT

Under the guidance of

MS.ASHWATHI VISWANATHAN

JUNE 2011

Department of Computer Science & Engineering

JYOTHI ENGINEERING COLLEGE, CHERUTHURUTHY

THRISSUR 679 531

INNOVATIVE STANDALONE EMBEDDED FILE COPIER FROM CAMERA/PENDRIVE

A PROJECT REPORT

Submitted by

BINCY P ANDREWS

NEETHU NICLAVOSE

RESHMA VARGHESE

TINU MARY C D

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY (B.TECH)

in

COMPUTER SCIENCE & ENGINEERING

of

UNIVERSITY OF CALICUT

Under the guidance of

MS.ASHWATHI VISWANATHAN

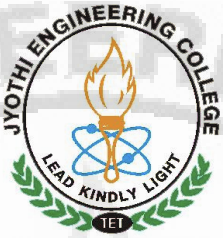
JUNE 2011

Department of Computer Science & Engineering

JYOTHI ENGINEERING COLLEGE, CHERUTHURUTHY

THRISSUR 679 531

Department of Computer Science & Engineering
JYOTHI ENGINEERING COLLEGE, CHERUTHURUTHY
THRISSUR 679 531



JUNE 2011

BONAFIDE CERTIFICATE

Certified that this project report **" INNOVATIVE STANDALONE EMBEDDED FILE COPIER "** being submitted in partial fulfillment of the requirements for the award of degree of **Bachelor of Technology** of **University of Calicut** is the bonafide work of **" BINCY P ANDREWS, NEETHU NICLAVOSE,RESHMA VARGHESE ,TINU MARY C D "**, who carried out the project work under our supervision.

MURALEE KRISHNAN C
HOD
Dept. of CSE

ASHWATHI VISWANATHAN
Lecturer
Dept. of CSE

CONTENTS

ACKNOWLEDGEMENT

We take this opportunity to express our heartfelt gratitude to all respected personalities who had guided, inspired and helped us in the successful completion of this project.

First and foremost, we express our thanks to **The Lord Almighty** for guiding us in this endeavour and making it a success.

We are thankful to our Principal **Dr. U Lazar John** and the Management for providing us with excellent lab and infrastructure facilities.

Our sincere thanks to the Head of the Department of Computer Science & Engineering, Prof. **Muralee Krishnan C** for his valuable guidance and suggestions.

We would like to express our deepest gratitude to **Ms. Ashwathi Viswanathan** for his/her valuable contributions and guidance.

Last but not least, we thank all our teaching and non teaching staffs of Department of Computer Science & Engineering, and also our friends for their immense support and help in all the stages for the development of the project.

ABSTRACT

The USB (Universal Serial Bus) is a fast and flexible interface for connecting devices to computers/microcontrollers. This project aims to develop an embedded device (act as host) for detecting and transferring data from one USB pen drive/camera to another one. The project aims to implement USB protocol in a microcontroller using the embedded technology.

The development of a USB based transfer system seeks to create a bridge between two USB devices for memory transfer when a computer is unavailable.When the pen drive/camera is connected to the port of the controller the device detects it and sends a signal back. This has to be acknowledged by the pen drive/camera and the communication between the systems is established using the USB protocol.

This device is very helpful to avoid the use of a computer to detect and transfer data from the pen drive/digital camera.The device will also be lighter because it contains fewer components than a PC, making it more convenient to carry around.

List of Figures

List of Tables

List of Abbreviations

USB	: *Universal Serial Bus*
LCD	: *Liquid Crystal Display*
PCB	: *Printed Circuit Board*
IC	: *Integrated Circuit*
PC	: *Personal Computer*
CD	: *Compact Disc*
FAT	: *File Allocation Table*
API	: *Application Programming Interface*

CHAPTER 1
Introduction

1.1 Overview

As the development of USB enabled peripherals increases, the Universal Serial Bus (USB) has rapidly become a standard in communication with the Personal Computer (PC) and has lead to new technologies for interfacing memory devices. These memory/storage devices connect to the USB ports and appear as removable storage device in personal computers, the most popular of which is the USB Flash Drive. The disadvantage of using USB Flash Drives is that it requires a PC to initiate file transfers between one another. As a solution to the USB Flash Drive disadvantage, the research project aims to develop a device that allows file transfers between two USB memory devices without the need for a Personal Computer (PC). This paper discusses the current implementation of all three modules as well as the testing process and partial results from file transfer speed tests between USB Flash Drives of different brands and capacities.

1.2 Motivation and Technical Relevance

The device is very helpful to avoid the use of a computer to detect and transfer data from the pen drive/digital camera. One of the future applications of this project is that it can be used to develop portable mp3 player. We can store songs to common pen drives. This host device accesses data from pen drive and to store or play this. The working of this is similar to above one.

Universal Serial Bus (USB) is a specification to establish communication between devices and a host controller (usually a personal computer), developed and invented by Ajay Bhatt, while working for Intel. USB has effectively replaced a variety of interfaces such as serial and parallel ports.USB can connect computer peripherals such as mice, keyboards, digital cameras, printers, personal media players, flash drives, Network Adapters, and external hard drives. For many of those devices, USB has become the standard connection method.

USB was designed for personal computers, but it has become commonplace on other

devices such as smartphones, PDAs and video game consoles, and as a power cord. As of 2008, there are about 2 billion USB devices sold per year, and approximately 6 billion total sold to date.Unlike older connection standards such as RS-232 or Parallel port, USB connectors also supply electric power, so many devices connected by USB do not need a power source of their own.

A floppy disk may not be the best way to move or save your backup files because the backup files can be larger if u have a larger collection. Saving them to a CD takes several steps and you have to remember to change the properties from ”read only” when you copy them back from a CD.If your computer is fairly new, then it most likely has one or more USB ports. This means you can use a USB Flash Drive.These little devices can be smaller than a pack of gum but can hold large amounts of data. They are readily available at almost any place that sells computer supplies .Ranging in price from 15 to over 100 in sizes from 128 MB to 2 GB they offer a great way for moving and storing your backup data. Make sure you get one with as much memory as you can easily afford to be sure that you have enough room for your files.

Once they are plugged into a USB port (either on the back of your PC or conveniently located on the front of some newer PCs) they show up like another hard drive on your computer. You can copy files to and from the the USB Flash drive just as you would to a floppy, zip, or hard drive.

1.3 Project Plan

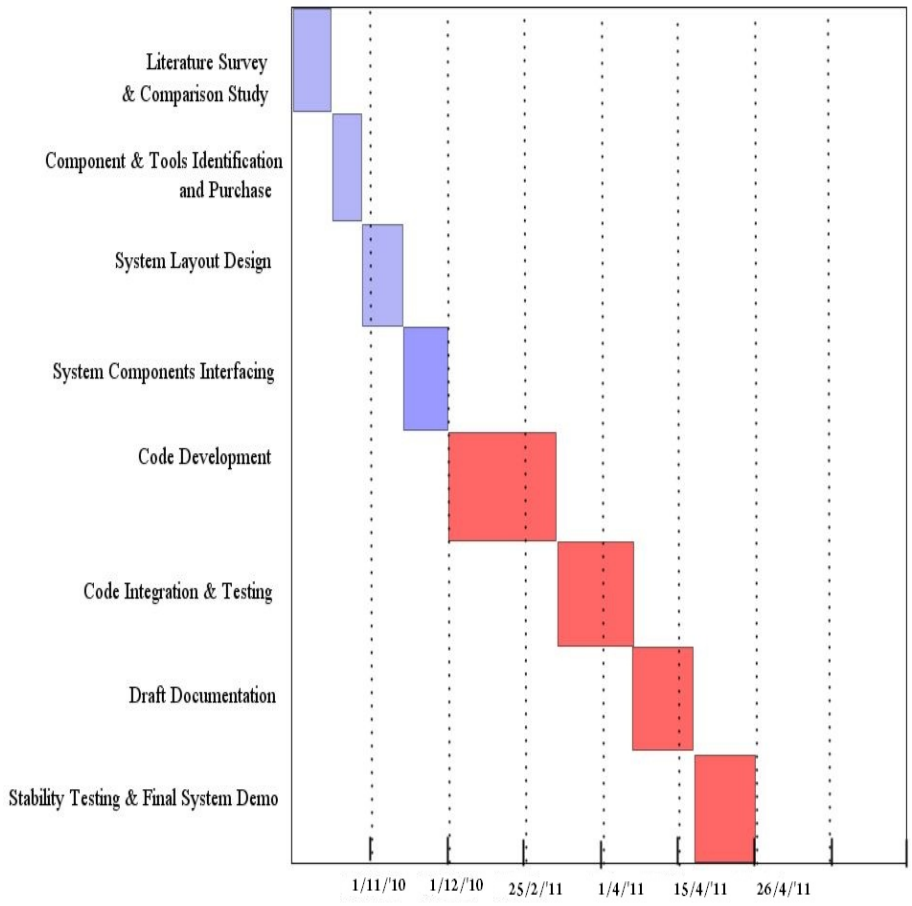


Fig. 1.1: Project Plan

1.4 Synopsis

The USB (Universal Serial Bus) is a fast and flexible interface for connecting devices to computers/microcontrollers. This project aims to develop an embedded device (act as host) for detecting and transferring data from one USB pen drive/camera to another one. The project aims to implement USB protocol in a microcontroller using the embedded technology.

When the pen drive/camera is connected to the port of the controller the device detects it and sends a signal back. This has to be acknowledged by the pen drive/camera and the communication between the systems is established using the USB protocol.

When the USB device is connected to the USB port of the controller, the device driver detects the presence of new device and loads its driver, if it is previously installed. Once proper driver is loaded, the communication between the systems is established using the USB driver protocol.

The firmware written in microcontroller read the content of the USB device and its ID using USB cables. When a pen drive is connected display the system display all file names in LCD. The data from one pen drive/camera can be read and stored in the other one according to the user command. The device takes read, write or other work according to the user commands.

After the host has exchanged enumeration information with the device and a device driver has been assigned and loaded, the application communication can be fairly straightforward. At the host, application can use slandered API function to read and write to the device. At the device , transferring data typically requires placing data to send in the USB controllers transmit buffer, reading received data from the receive buffer, and on completing a transfer, ensuring that the device is ready for the next transfer .

This device is very helpful to avoid the use of a computer to detect and transfer data from the pen drive/digital camera. One of the future applications of this project is that it can be used to develop portable mp3 player. We can store songs to common pen drives. This host device accesses data from pen drive and to store or play this. The working of this is similar to above one. [**?**]

1.5 Member roles and responsibilities

The entire Project work was divided between all members of the group.And each has been assigned various roles for achieving the goals of the project.And the following table shows the team roles and responsibilities.

1.1: Team Organization

Name	**Role/Responsibility**
Reshma	Leader
Neethu	LCD Interfacing
Bincy	Keyboard Interfacing
Tinu	Pic interfacing

CHAPTER 2

Literature Survey

The project idea is originated from the paper Implementation of a USB Slave to Slave File Transfer Device Using Microcontrollers by Mark Alvin U. Chua Charles D. Jorge Ana Marian M. Pedro Brian Emmanuel G. Tam Gregory G. Cu ,College of Computer Studies ,De La Salle University Manila.

As the development of USB enabled peripherals increases, the Universal Serial Bus (USB) has rapidly become a de facto standard in communication with the Personal Computer (PC) and has lead to new technologies for interfacing memory devices. These memory/storage devices connect to the USB ports and appear as removable storage device in personal computers, the most popular of which is the USB Flash Drive. The disadvantage of using USB Flash Drives is that it requires a PC to initiate file transfers between one another.[1] This paper discusses the current implementation of all three modules as well as the testing process and partial results from file transfer speed tests between USB Flash Drives of different brands and capacities. The USB Slave to Slave File Transfer Device is a device that facilitates file and folder/directory transfers from one flash drive to another flash drive using the USB 1.1 interface without the need for a Personal Computer (PC) to act as mediator. The system allows the user to select files or folders/directories for copying from a source flash drive to a user selectable directory in the destination flash drive. In addition, the system is able to check if there is sufficient memory space for the file/folder/directory to be copied onto the destination flash drive; if not, the system requests the user to delete some files or folder/directories to free some memory in the destination flash drive. The user may or may not opt to delete anything.[2]

This paper discusses the current implementation of all three modules. User Interface Controller Module is responsible for obtaining user input and displaying user requested information thru an LCD. User input is received thru this module. All information to be displayed is received from the File System Controller module.The LCD Controller submodule is the hardware that interfaces the LCD to the main hardware system. It is composed of a 20 character by 4 lines LCD and a microcontroller solely for controlling the LCD and receiving user input. The Navigation submodule is the input handling hardware of the system. This submodule includes six input buttons and circuitry for filtering the input signals. The Navigation submodule sends a corresponding signal for each button press to the microcontroller that controls the LCD so that it could update the screen. The File System Controller is responsible for all file management processes. The File Manager Sub module is responsible for all file man-

agement functions/features that are available in the system. These features include copying a file or folder, delete. This module accepts the commands to either read or write onto a USB flash drive as requested by the File System Controller Module. Input from the USB flash drives is the result of an earlier request for data. The module also governs the data that needs to be transmitted to the USB Flash Drives, as well as process and send the results of the transaction to the File System Controller Module. In addition, the module also sends notifications to alert the user of the status of the system or the operation. The user interface system is designed to allow the user to browse and choose specific files or folders to transfer between the source and destination devices. The screen consists of only the important data to avoid cluttering the screen.[3]

CHAPTER 3

Proposed system

3.1 Introduction

Architecture of a system describes the overall organization and structure of the system in terms of its major constituents and their interactions. In designing the architecture the designer must consider many functional requirements such s cost and reliability. While the architecture of the system is of course influenced by these requirements there are also some structuring principles that govern the design of the architecture. In particular depending on the system requirements some specific decomposition of the system into components is required.

First knowing about architectures that have already been tried and tested in previous system allows the architect to start on a design quickly and with confidence .Associated with each architecture are the design decisions that must be addressed.architecture establishes the modes of computation among the components of the system.thus an architecture serves as an integration platform for interconnecting the different subsystems of the system.our project is based on event based architecture .in an event based architecture components respond to the occurrence of an event.an event might be detection of USB or removal USB or copying of files. Components are designs to create events or start their operations when they receive an event. Events such as detection of USB are performed by the USB controller and various modules like LCD module,keyboard module, and usb module respond to the occurrence of the events.

3.2 Member effort

Project Work was divided between the members of the group and each were assigned tasks to complete within the scheduled time.The following table presents each members effort in the our team.

3.1: Module Allocation

#	Task	Estimated Effort	Start Date	End Date	Person
1	Task 1 LCD interfacing	(2 weeks)	(7/9/2010)	(21/9/2010)	Neethu
2	Task 2 Keyboard interfacing	(3 weeks)	(5/12/2010)	(26/12/2010)	Bincy
3	Task 3 USB interfacing	(3 weeks)	(5/12/2010)	(26/12/2010)	Reshma
4	Task 4 PIC interfacing	(3 months)	(7/9/2010)	(12/4/2011)	Tinu

CHAPTER 4

System requirements specification

4.1 Software Requirements

- Programming language used embedded C
- PCB design tool: protel software

4.2 Hardware Requirements

- PIC16F877

☐ 40-Pin Flash-Based, 8-Bit Microcontroller

☐ Operating frequency DC 20 MHz

☐ 8k FLASH program memory

☐ 368 bytes of data memory

☐ I/O ports A,B,C,D,E

☐ Serial communication USART

☐ 35 Instruction set

- USB host interfacing IC
- LCD panel(2 by 16)
- Control Buttons

CHAPTER 5
Design and analysis

5.1 Module breakup

Following table shows the various modules in our project.

5.1: Module Description

Module	Description
Module1 USB interfacing module	USB devies are detected and installed by the controller
Module2 Keyboard interfacing module	User inputs are obtained by this module.
Module3 Lcd interfacing module	The LCD module is used to display the files in the USBs
Module4 PIC microcontroller module	Burn the Code using ICD3 to the PIC microcontroller.

USB interfacing module:

When the USB device is connected to the USB port of the controller, the device driver detects the presence of new device and loads its driver, if it is previously installed. Once proper driver is loaded, the communication between the systems is established using the USB driver protocol.

Keyboard interfacing module:

A 4 by 4 keypad is also interfaced.It consists of buttons left,right,reset etc.Also renaming of folders/files can be done.

Lcd interfacing module:

This module is responsible for obtaining user input and displaying user requested information through an LCD. Once the firmware written in microcontroller read the content of the USB device and its ID using USB cables, all the file names in USB are displyed in LCD. Then data from one USB can be read and stored in the other one according to the user command.The device takes read, write or other work according to the user commands.

PIC Microcontoller module:

PIC microcontroller consists of small number of fixed length instructions.Separate code and data spaces needed for the devices.It is interfaced with the lcd module.

CHAPTER 6

Implementation

6.1 Introduction

When the pen drive is connected to the port of the controller the device detects it and sends a signal back. This has to be acknowledged by the pen drive/camera and the communication between the systems is established using the USB protocol.When the USB device is connected to the USB port of the controller, the device driver detects the presence of new device and loads its driver, if it is previously installed. Once proper driver is loaded, the communication between the systems is established using the USB driver protocol.

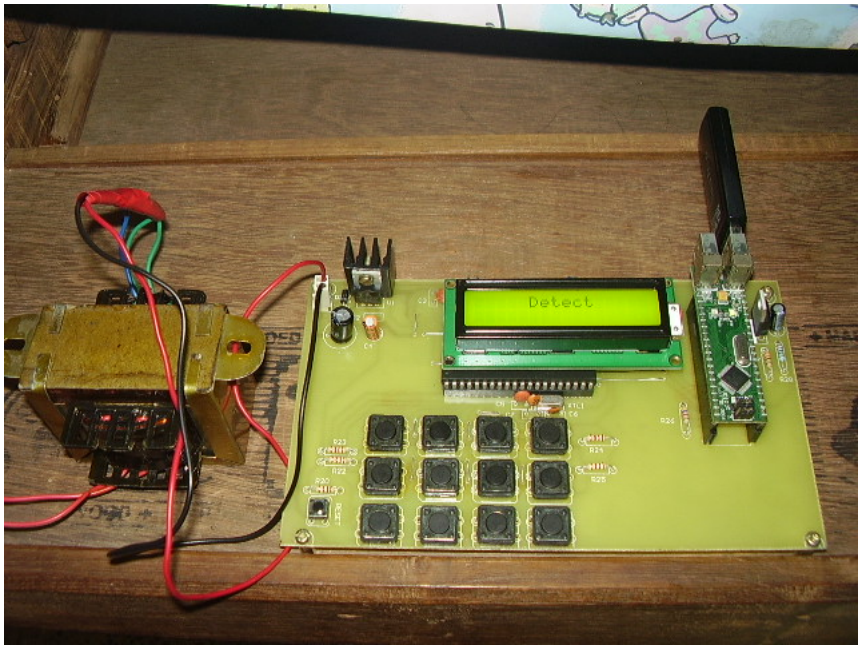


Fig. 6.1: Detection

The firmware written in microcontroller read the content of the USB device and its ID using USB cables. When a pen drive is connected display the system display all file names in LCD. The data from one pen drive/camera can be read and stored in the other one according to the user command. The device takes read, write or other work according to the user commands.

Fig. 6.2: File names Diplayed on LCD

At the device , transferring data typically requires placing data to send in the USB controllers transmit buffer, reading received data from the receive buffer, and on completing a transfer, ensuring that the device is ready for the next transfer .

6.2 Algorithm / Pseudo codes

Codes for our project include the following:

```
#include <pic.h>
#include "DELAY.c"
#define RS RC5
#define RW RC0
#define EN RC1
bit DIR_Flg;
bit V_flg;
bit E_flg;
bit R_flg;
```

```
bit fst_flg;
bit detect;
bit File_DIR;
bit Find_Size;
bit read_ready;
unsigned char Dol_Count,Size_Count,Size_Arr[8],rem;
bit Dol_Flg;
bit Size_Comp;
unsigned int File_Size,File_Size1;
bank2 unsigned char USB_DAT,i,Disp[60];
unsigned char coun=0,no=0,File_Count1=0,File_Count=0,no1=0,i1;
bit A_flg;
bit P_flg;
bit F_flg;
bit read_comp;
bit Read_Flg;
bank1 unsigned char File_Name[5][13];
void UartInit();
void LcdInitial();
void subcom(int );
void subdata(char);
void Delay3Milli();
void sendusb(char);
void Basic_Initial();
void key_row1();
void key_board();
static void Send2Lcd(const char *CPtr);
static void Send2USB(const char *CPtr);

void main()
{
        Basic_Initial();
    UartInit();
        LcdInitial();
        Send2Lcd(" EMBEDDED FILE  "); // INITIAL DISPLAY
        subcom(0xc0);
        Delay3Milli();
```

```
Send2Lcd("      COPIER      ");      // INITIAL DISPLAY
DelayMs(1000);
subcom(0x01);

DelayMs(2000);
fst_flg=1;
Send2USB("FWV\r"); // SEND COMMAND
                                        FOR IDENTIFY THE VERSION
DelayMs(1000);
while(!detect)   // IF 0X0D NOT RCV
                                    MEANS THE MODULE IS DETECTED
{
        subcom(0x80);   // FIRST LINE ADRESS FOR LCD
        Delay3Milli();
        Send2Lcd("Not Detect"); //DISPLAY STATUS IN LCD
        fst_flg=1;
        Send2USB("FWV\r"); // AGAIN SEND FIRM
                                WIRE COMMAND
        DelayMs(100);
   i=0;
}



while(1)       // INFINITE LOOP START HERE
{
        if(detect)//IF OXOD WILL RCV MEANS
                THE MODULE IS DETECTED
        {
    detect=0;
                fst_flg=0;
                subcom(0x80);
                Delay3Milli();
                Send2Lcd("      Detect    ");// DISPLAY
                                STATUS IN LCD
                sendusb(0X0D);
                DelayMs(10);(100);
```

```
                Send2USB("ECS\r");
                Delay3Milli();
                Send2USB("A:\r");
                DelayMs(10);(100);
                sendusb('\r');
        }

key_board();  //KEBOARD CONFIGARATION

if(Size_Comp)     // SIZE WILL BE RCVD
        {
                subcom(0xc0);
                File_Size=0;
                Size_Comp=0;
                for(i=4;i>0;i--)
                {
              File_Size=File_Size*10;
                File_Size=Size_Arr[i]+File_Size;
                }
                File_Size1=File_Size;

                for(i=4;i>0;i--)//FUNCTION FOR
                                DISPLAY THE FILE SIZE
                {
               rem=File_Size1%10;
                subdata(rem+'0');
                File_Size1=File_Size1/10;

                }


                sendusb('\r');
                Send2USB("OPR ");// COMMAND FOR
                                 OPEN FOR READ
              for(coun=0;coun<12;coun++)
             sendusb(File_Name[File_Count][coun]);
             sendusb('\r');
```

```
            DelayMs(10);(100);
            DelayMs(10);(100);
            DelayMs(10);(100);
        Send2Lcd("Size Rxve");
            read_ready=1;
            Read_Flg=1;
            i1=0;
            Send2USB("RDF ");

            sendusb(File_Size);
            sendusb(0x0d);
            DelayMs(10);(100);
}

if(read_comp)
{
            Read_Flg=0;
            subcom(0xc0);
        Delay3Milli();
        Send2Lcd("readcomp");
            Send2USB("CLF ");

        Delay3Milli();
        Send2Lcd("Close file");
            DelayMs(10);(100);
            Send2USB("B:\r");
            DelayMs(10);(100);
            Send2USB("OPW ");
            for(coun=0;coun<12;coun++)
            sendusb(File_Name[File_Count][coun]);
            sendusb('\r');
            DelayMs(10);(100);
            subcom(0xc0);
        Delay3Milli();
        Send2Lcd("Copiying");
            Send2USB("WRF ");
            sendusb(0x00);
```

```
sendusb(0x00);
sendusb(0x00);
sendusb(10);
sendusb('\r');
DelayMs(10);(100);
for(i=0;i<10;i++)
{
        sendusb(Disp[i]);
        DelayMs(10);(100);
}
Send2USB("WRF ");
sendusb(0x00);
sendusb(0x00);
sendusb(0x00);
sendusb(10);
sendusb('\r');
        for(i=10;i<20;i++)
{
        sendusb(Disp[i]);
        DelayMs(10);(100);
}
Send2USB("WRF ");
sendusb(0x00);
sendusb(0x00);
sendusb(0x00);
sendusb(10);
sendusb('\r');
        for(i=20;i<30;i++)
{
        sendusb(Disp[i]);
        DelayMs(10);(100);
}
Send2USB("WRF ");
sendusb(0x00);
sendusb(0x00);
sendusb(0x00);
sendusb(10);
```

```
                        sendusb('\r');
                                for(i=30;i<File_Size;i++)
                        {
                                sendusb(Disp[i]);
                                DelayMs(10);(100);
                        }
                        i=0;
                        File_Size=0;
                        Read_Flg=0;

                        Send2USB("CLF ");
                        for(coun=0;coun<12;coun++)
                        sendusb(File_Name[File_Count][coun]);
                        sendusb('\r');
                        DelayMs(10);(100);
                        sendusb('\r');
                        subcom(0xc0);
                    Delay3Milli();
                   Send2Lcd("Complete");
                   Delay3Milli();
                   Send2USB("A:\r");
                }
        }
}

void interrupt rece()
{
        RCIF=0;
        USB_DAT=RCREG;

         if((Read_Flg)&&(i1<File_Size))
        {
                Disp[i1]=USB_DAT;
                i1++;
        }
        else if((Read_Flg)&&(i1>=File_Size))
        {
```

```
Disp[i1]='\0';
        read_comp=1;
        }
        if((USB_DAT=='V')&&(fst_flg))
        {
                V_flg=1;

        }
        else if((USB_DAT=='D')&&(V_flg))
        {
                V_flg=0;
                E_flg=1;
        }
        else if((USB_DAT=='F')&&(E_flg))
        {
                E_flg=0;
                R_flg=1;
        }
        else if((USB_DAT=='C')&&(R_flg))
        {
                R_flg=0;
                A_flg=1;
        }
        else if((USB_DAT==0x0d)&&(A_flg))
        {
                A_flg=0;
                fst_flg=0;
                detect=1;
        }

                if(DIR_Flg)
        {

                if(USB_DAT!=0x0d)
                {
                        File_Name[no][coun]=USB_DAT;
                        coun++;
```

```
                    }
                    else  if(USB_DAT==0x0d)
                    {

                              no1++;
                              if(no1>File_Count1)
                              {
                              no++;
                              coun=0;
                              if(no>=5)
                              {
                                        DIR_Flg=0;
                                        File_DIR=1;
                                        File_Count=0;

                              }
                              }
                    }
          }
if(Find_Size)
          {

           if(USB_DAT==' ')
            {
             Dol_Flg=1;
             Size_Count=1;
            }
else  if((Dol_Flg)&&(Size_Count<5)&&(USB_DAT!=' '))
          {
           Size_Arr[Size_Count]=USB_DAT;
           Size_Count++;
          }
          if(Size_Count>=5)
            {
                    Find_Size=0;
                    Size_Comp=1;
                    Dol_Flg=0;
```

```
            }
                    }
}

void Basic_Initial()
{
        V_flg=0;
        E_flg=0;
        R_flg=0;
        fst_flg=1;
        detect=0;
        Find_Size=0;
        TRISB=0X1F;
        RBPU=0;
}
void UartInit()
{
        TRISD=0x00;         //SET PORTD AS OUTPUT
        TRISC=0x80;         //SET PORTC AS OUTPUT
        RCSTA    = 0;
        TXSTA    = 0;
        RCREG    = 0;
        BRGH     = 1;
        SPBRG    = 0x81; //High Speed - 115200
        SYNC     = 0;
        SPEN     = 1;
        TXEN     = 1;     /* Transmit Enabled */
        CREN=1;
        GIE=1;
        PEIE=1;
        RCIE=1;
}

void LcdInitial()
{

        subcom(0x30);       //8 BIT LCD INTERFACE
```

```
        subcom(0x0c);       //CURSOR OFF
        subcom(0x01);       //CLEAR DISPLAY
        subcom(0x06);       //ADDRESS INCRIMENT
        subcom(0x80);       //ADDRESS OF 1ST POSITION
        subcom(0x38);       //2ROW DISPLAY
}

void subcom(int command)
{
        RS=0;           //SELECT COMMAND REGISTER
        RW=0;             //WRITE TO LCD
        EN=1;            //ENABLE THE MODULE
        PORTD=command;     //COMMAND SEND THROUGH PORTD
        DelayMs(10);(3); //MAKE 3MS DELAY
        EN=0;                       //DISABLE THE MODULE
}

void subdata(char lcddata)
{
        RS=1;                       //SELECT DATA REGISTER
        RW=0;                       //WRITE TO LCD
        EN=1;                       //ENABLE THE MODULE
        PORTD=lcddata;   //COMMAND SEND THROUGH PORTD
        DelayMs(10);(10); //MAKE 10MS DELAY
        EN=0;                       //DISABLE THE MODULE
}

static void Send2Lcd(const char *CPtr)
{
        while(*CPtr != '\0')
        {
                subdata(*CPtr);
            DelayMs(10);(0x04);
                CPtr++;
        }
}
```

```
static void Send2USB(const char *CPtr)
{
        while(*CPtr != '\0')
        {
                sendusb(*CPtr);
            CPtr++;
        }
}

void sendusb(char usbdata)
{
        TXREG=usbdata;
        while(!(TRMT))
        {}
}

void Delay3Milli()
{
        DelayMs(10);(1);
        DelayMs(10);(1);
        DelayMs(10);(1);
}

void key_board()
{
        RB5=0;
        RB6=1;
        RB7=1;
        key_row1();
        RB5=1;
        RB6=0;
        RB7=1;
        key_row2();

}

void debounce()
```

```
{
        char key3;
        key3=PORTB&0x0f;
        while(key3!=0x0f)
        {
                key3=PORTB&0x0f;
                //DelayMs(10);(10);
        }
}
void key_row1()
{
 char key1,key2,key3;
 key1=PORTB&0x1e;
  if(key1!=0x1e)
  {
   DelayMs(10);(10);
    key2=PORTB&0x1e;
        if(key2!=0x0f)
        {
         if(key1==key2)
         {
          if(key1==0x0e)
           {
          subcom(0x80);
          Size_Comp=0;
          Size_Count=0;
          Find_Size=1;
          Send2USB("DIR ");
          for(coun=0;coun<12;coun++)
          sendusb(File_Name[File_Count][coun]);
          sendusb(0x0d);
          }
          else if(key1==0x16)
           {
            subcom(0x80);
            subdata('2');
         if((File_DIR)&&(File_Count>0))
```

```
            {
             File_Count1--;
             subcom(0x80);
              File_Count--;
              for(coun=0;coun<12;coun++);
            subdata(File_Name[File_Count][coun]);

            }
           if((File_DIR)&&(File_Count==0))
            {

              File_DIR=0;
              Send2USB("DIR\r");
              DelayMs(10);(100);
DIR_Flg=1;
            }
            }
          else  if(key1==0x1a)
            {
             subcom(0x80);
             subdata('3');
          if((File_DIR)&&(File_Count<5))
            {
             File_Count1++;
             subcom(0x80);
                        File_Count++;
                  for(coun=0;coun<12;coun++);
          subdata(File_Name[File_Count][coun]);

             }
          if((File_DIR)&&(File_Count>=4))
            {
              File_DIR=0;
              Send2USB("DIR\r");
DelayMs(10);(100);
              DIR_Flg=1;
            }
```

```
            }
        else  if(key1==0x1c)
        {       subcom(0x80);
                    subdata('4');
                     DIR_Flg=1;
                    Send2USB("DIR\r");
                    DelayMs(10);(100);

         }
                            debounce();
                            }
                    }
        }


}
```

CHAPTER 7
Testing & Maintenance

7.1 Tests

As the part of the project, we did the testing of each module separately. The programs for each module is written and tested using an IDE. The hardware part is configured on bread boards and veri?ed the outputs.

7.1: Unit Test Chart

No	Unit Name	Test Status
1	LCD Module	Complete
2	KEYBOARD Module	Complete
3	USB Module	Complete
4	PIC Module	Complete

7.1.1 Integration Testing

When the unit tests are satisfactorily conducted the system as a complete entity must be tested. When the coding of individual modules was over we needed to integrate each application as individual tasks and develop an standalone embedded system that can perform functions such as copy,delete the contents between two USB drives.Contents of the flash disks are displayed through the LCD from which we specifically copy and delete files.We ensured that all tasks worked correctly.Threee tests are used to asses the performance of the system,speed of the file transfers, accuracy of the copied files, and USB Device recognition. We performed test on files that work properly.

7.1.2 Blackbox Testing

Black-box testing is a method of software testing that tests the functionality of an application as opposed to its internal structures or workings (see white-box testing). Specific knowledge of the applications code/internal structure and programming knowledge in generals not

required. Test cases are built around speci?cations and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specification , requirements, and designs to derive test cases. These tests can be functional or non-functional,though usually functional. The test designer selects valid and invalid inputs and determines the correct output. There is no knowledge of the test objects internal structure.

7.1.3 White Testing

White-box or glass-box testing relies on analysing the code itself and the internal logic of the software. White-box testing is often, but not always, the purview of programmers. It uses techniques which range from highly technical or technology specific testing through to things like code inspections. Although white-box techniques can be used at any stage in a software products life cycle they tend to be found in Unit testing activities. In our project we found white box testing was more useful. Since this testing involves input and a corresponding output, since it is a method of testing software that tests internal structures or workings of an application. Since the coding of individual application has been completed, we tested each of the modules and ensured that the logic is correct and output obtained is correct. We also used white box testing in the integration testing stage for ensuring that the final implementation of our project is as per our plan.

7.2 Maintenance

The main objective of this project is to create a FAT based file system device, which can transfer data from one USB Flash device to another USB Flash disk device without the use of a computer. The device has two main processes, to copy and delete a certain file.There are several devices that is use to transfer data but were restricted to the given operating system of Windows.One of the main advantages of this system is that USB Flash disk is very much popular; therefore, the general public can participate and anticipate the quick and available data using this device.Embedded device has been designed in such a way that future modifications can be carried out smoothly. There are four modules in this project and integrating these modules should be done with care so that there should not be erroneous interfacing issues. So the modules are clearly defined such that changes to one module can be applied without affecting the other module. It is designed to tolerate shock, vibration, humidity and temperature extremes. As new needs and applications are arising day by day, we have maintained a level of easiness and simplicity in adding new changes to the system.

CHAPTER 8
Conclusion

8.1 Introduction

The USB (Universal Serial Bus) is a fast and flexible interface for connecting devices to computers/microcontrollers. This project aims to develop an embedded device (act as host)for detecting and transferring data from one USB pen drive/camera to another one. The project aims to implement USB protocol in a microcontroller using the embedded technology. As a solution to the USB Flash Drive disadvantage, our project aims to develop a device that allows file transfers between two USB need for a Personal Computer. The following can be assured.

- Portability
- Avoids the use of computer
- Low power consumption
- simple user interface
- Reduced start up time
- Light weight

8.2 Future Scope

One of the future applications of this project is that it can be used to develop portable mp3 player. We can store songs to common pen drives. This host device accesses data from pen drive and to store or play this. The working of this is similar.More over blue tooth module can be added to it which extends its scope. More over the device may be converted into touch screen which will make the device more attractive.

REFERENCES

[1] M. A. U. C. Charles, "Implementation of a usb slave to slave file transfer device using microcontrollers," *IEEE Transactions on Information Theory*, vol. 22, pp. 644–654, November 2010.

[2] USBFlashDrive.org[site], "Usb flash drives, usb memory and portable computer hard drive information," July 1 2005. [Online]. Available: http://www.usbflashdrive.org/usbfd_overview.html

[3] J. Axelson, *USB COMPLETE Second Edition*, 2nd ed. Madison, 2004, vol. 2.