File Transfer between SD Memory Card and USB Flash Drive through UART PIC Interfacing with Mass Storage Device Controller

by

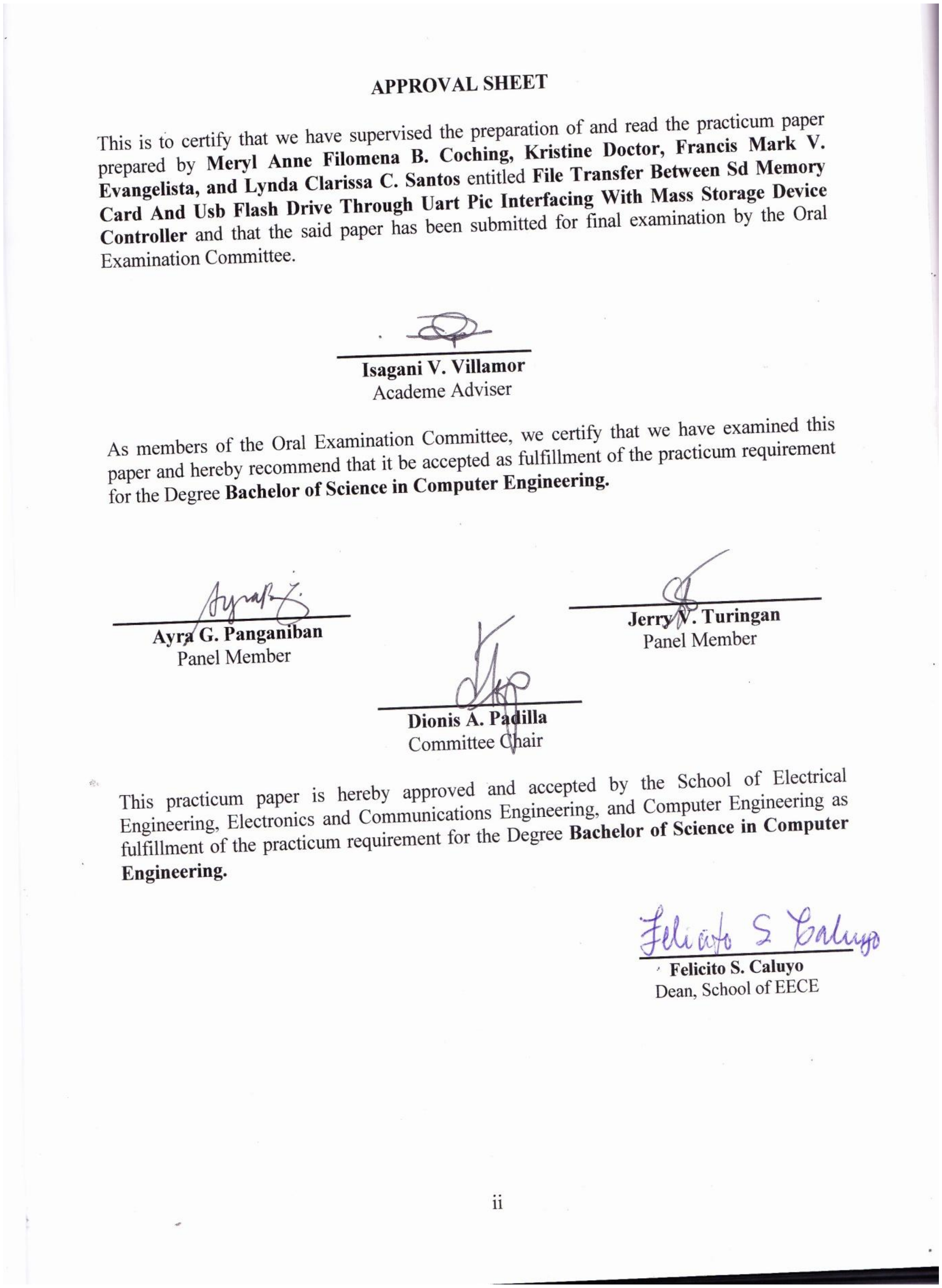
Meryl Anne Filomena B. Coching
Kristine Doctor
Francis Mark V. Evangelista
Lynda Clarissa C. Santos

A Thesis Report Submitted to the School of Electrical Engineering, Electronics Engineering, and Computer Engineering in Partial Fulfillment of the requirements for the degree

Bachelor of Science in Computer Engineering

Mapua Institute of Technology

December 2011



APPROVAL SHEET

This is to certify that we have supervised the preparation of and read the practicum paper prepared by **Meryl Anne Filomena B. Coching, Kristine Doctor, Francis Mark V. Evangelista, and Lynda Clarissa C. Santos** entitled **File Transfer Between Sd Memory Card And Usb Flash Drive Through Uart Pic Interfacing With Mass Storage Device Controller** and that the said paper has been submitted for final examination by the Oral Examination Committee.

[illegible]

Isagani V. Villamor
Academe Adviser

As members of the Oral Examination Committee, we certify that we have examined this paper and hereby recommend that it be accepted as fulfillment of the practicum requirement for the Degree **Bachelor of Science in Computer Engineering.**

[illegible]

Ayra G. Panganiban
Panel Member

[illegible]

Jerry V. Turingan
Panel Member

[illegible]

Dionis A. Padilla
Committee Chair

This practicum paper is hereby approved and accepted by the School of Electrical Engineering, Electronics and Communications Engineering, and Computer Engineering as fulfillment of the practicum requirement for the Degree **Bachelor of Science in Computer Engineering.**

Felicito S Caluyo

Felicito S. Caluyo
Dean, School of EECE

ii

ACKNOWLEDGEMENT

This study would not have been possible without the people who supported and helped us. First of all, we would like to thank Engr. IsaganiVillamor, our thesis adviser, for helping, guiding and giving us valuable advices. He imparted us his knowledge and expertise for the completion of this study. We give our deepest gratitude to our parents for continuously supporting and inspiring us to our long journey. To all of our closest friends and love ones for giving us emotional care and camaraderie that helped us get through difficult times. We would also like to thank our panel members and instructor Engr. Jerry Turingan, Engr. Dionis Padilla and Engr. AyraPanganiban for giving us recommendations and corrections to improve this study. Lastly, to the Almighty God for providing us the strength, knowledge and willpower to finish this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Flash memories like SD Memory card and USB Flash drive are widely used nowadays. However, transferring files within those storage devices cannot be done without the aid of desktop computers, laptop, etc. This paper presents the research and development of a device as an alternative to computers in transferring files between a USB Flash Drive and SD Memory Card. A mass storage device controller together with a microcontroller was used to initiate the operation of the device with a maximum capacity of 4 GB in the USB Flash Drive and 4 GB Memory Card. The study began by constructing the main device down to its testing part in which the integrity and accuracy of files were considered. T-test was used as a statistical method to determine whether the difference between two proportions; that is the file accuracy of transferring file from the device and from the device of the previous research is significant in the Implementation of a USB Slave to Slave File Transfer Device Using Microcontrollers. Though some errors were produced in the testing process, the final result stated that there was no significant difference between the file transfer accuracy of the device and the previous study.

Keywords: USB Flash Drive, SD Memory Card, USB Embedded Host Controller, UART, File Transfer.

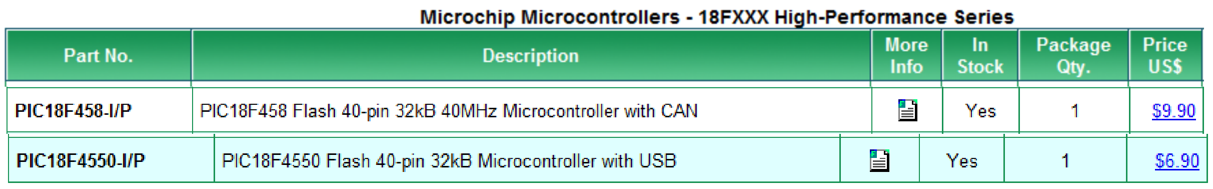
Chapter 1

INTRODUCTION

Portability and mobility of storage devices like flash drives and memory cards are becoming so popular to carry and transport data on the go. Unlike CD/DVD, hard drive, floppy disk or tape drive, a flash memory has no moving parts, so it guarantees fast read/write speeds and more durable compared to other forms of storage media. Flash memory does not follow a standard form factor, so it can be designed or shaped depending on what the customer wants. Using a flash drive/memory card to transfer a file typically uses a computer (e.g. laptop, pc, netbook or tablet) to initialize mass storage communication which is not portable. Although a memory card is used in mobile phones withtheir diminutive sizes, it limits another memory card type of larger size to host them. Additionally, mobile phones cannot host a flash drive and memory card at the same time to transfer data. The proposed system bridges the gap of flash drive to memory card data transfer when a computer is not at hand. Moreover, booting a computer just to transfer a file/s brings annoyance.

In an earlier research, an embedded system is used wherein a slave to slave flash drive transfer makes it possible to transfer file/s using a programmable microcontroller with USB multi-role embedded host/peripheral controller and navigation control. This makes it possible to transfer data from a flash drive to another flash drive with 100% data integrity using MD5 checksum algorithm. That research also benchmarks the speed of data transfer depending on the size of the file and the depth of the file location in a folder. The device consumes a small amount of power only, has low memory footprint and less resources are utilized.

The research project being proposed in this study addresses the problem of flash drive to flash drive transfer without the use of a computer. The transfer of data between flash drive and memory card is not included. This study focuses on a design that is inexpensive and can readily be implemented without the use of a computer. Figure 1.1 shows the comparison of prices for the microcontrollers used by the previous research and the proposed system.


Microchip Microcontrollers - 18FXXX High-Performance Series

Part No.	Description	More Info	In Stock	Package Qty.	Price US$
PIC18F458-I/P	PIC18F458 Flash 40-pin 32kB 40MHz Microcontroller with CAN	📄	Yes	1	$9.90
PIC18F4550-I/P	PIC18F4550 Flash 40-pin 32kB Microcontroller with USB	📄	Yes	1	$6.90

Figure 1.1 Comparison of Prices of Microcontrollers used

It can be shown that PIC18F458, the one that was used in the previous research costs more than the microcontroller that will be used in this research project, which is PIC18F4550. The previous researchers used 20x4 Liquid Crystal Display (LCD) screen which is larger than the size of the screen that will be used in the device and therefore it will not be as expensive as the device used in the previous research.

It is important to note that the speed of data transfer is slow compared to a computer due to the host controller used which is USB 2.0 full speed. Theoretically, if the host microcontroller used follows the USB 2.0 high speed standard, the speed of data transfer should be much faster than the computer since there is no associated overhead in the bus compared to the computer.

The main objective of this study is to develop a device that can transfer files from flash drive to memory card and vice versa. Specific objectives include the following: to create an interface that will list all the file/s or folder/s in the current directory,to make sure that the user has the ability to navigate through it using the buttons; to have a successful file transfer between flash drive and memory card; and to determine whether there is a significant difference between the file accuracy of transferring file from USB Flash Drive to SD Memory Card or vice versa used in the present study and that of the previous study.Since there is only one display, the user can choose if he/she wants to see the contents of the flash drive or the memory card only. The compatibility between the flash drive and the memory card will depend on the capacity, standard used and additional features added by the manufacturer such as creating additional encrypted partition. The design is an extension of the previous research but of different implementation. The device should handle all file formats since it will not deal with the content of the file but rather its existence on the physical medium. However, it is important to mention that the study will concentrate on the integrity of the data copied. The user’s ability to transfer data between flash drive and memory card in a small devicecan reduce the utilization of computer which translates into saving power and resources. Additionally, it uses less amount of carbon footprint because of low heat emission and only uses portable electricity.The device emits low heat in such a way that it does not require that high amount of electricity, which produces heat whenever it functions. Also, compared to the computers used in transferring files between a USB flash drive and a memory card, the device has lesser components than the first. The device is said to use portable electricity in such a way that the device only uses 5V source of power which is low compared to other computers such as laptops which uses 15V to 20V. The device can also be battery operated. A portable embedded system capable of transferring data can save unexpected circumstances such as no electricity or no computer availability. It can be also interfaced with other designs provided that they have a common port to transmit data. The key importance of this design is to show how powerful an embedded system can be in specific applications wherein it keeps the cost at minimum due to its use of limited resources.

The device is capable of transferring data to and from the Secure Digital card and Universal Serial Bus disk. It supports only File Allocation Table (FAT/FAT32) file system to be able to access data. Since the memory card slot is Secure Digital compliant, it can mount other memory card/s that can be plugged via SD adapter. However, this is not covered by the study. Some SD type memory cards such as Secure Digital High Capacity (SDHC) and Secure Digital Extended Capacity (SDXC) are newer standards which are not supported by the device. Only flash drive/s with total space of 4.0 Gigabyte can be plugged on the USB port that conforms to USB 2.0 standard while external hard drives are not supported due to their high power requirements. The device will have LCD and directional buttons for file/folder navigation, as well as another button to show the available options. Copy, paste, delete and auto-replace of file only are supported due to time constraints in the development and the file name follows the 8.3 convention.

Chapter 2

REVIEW OF RELATED LITERATURE

Implementation of a USB Slave To Slave File Transfer Device Using Microcontrollers

A thesis study entitled “Implementation of a USB Slave to Slave File Transfer Device” by Mark Alvin U. Chua, Charles D. Jorge, Ana Marian M. Pedro, Brian Emmanuel G. Tam, and Gregory G. Cu is about developing a device that copies and transfers file 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. Figure 2.1 shows the system setup while Figure 2.2 shows the general block diagram of the system. Contents of the flash drives are displayed in their 8.3 filename format through a twenty character by four lines - sized dot-matrix character liquid crystal display (LCD). The system is also able to check for identical file/folder/directory names and requests for a user confirmation to either proceed and overwrite a file/folder/directory or not. In addition, the system is able to check if there is a 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 system uses the Cypress CY7C67300, a programmable microcontroller and USB multi-role embedded host/peripheral controller, which has its own Basic Input/Output System and Framework program.

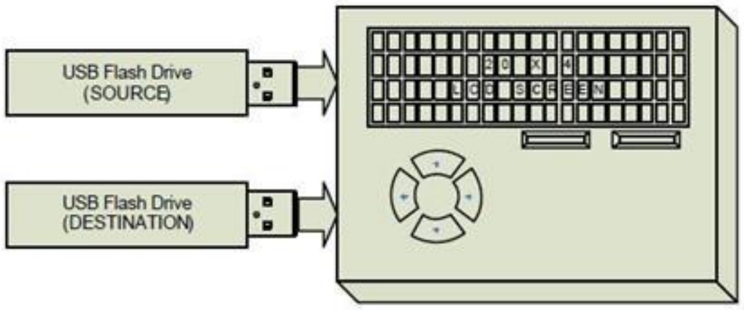


Fig. 2.1 System Setup

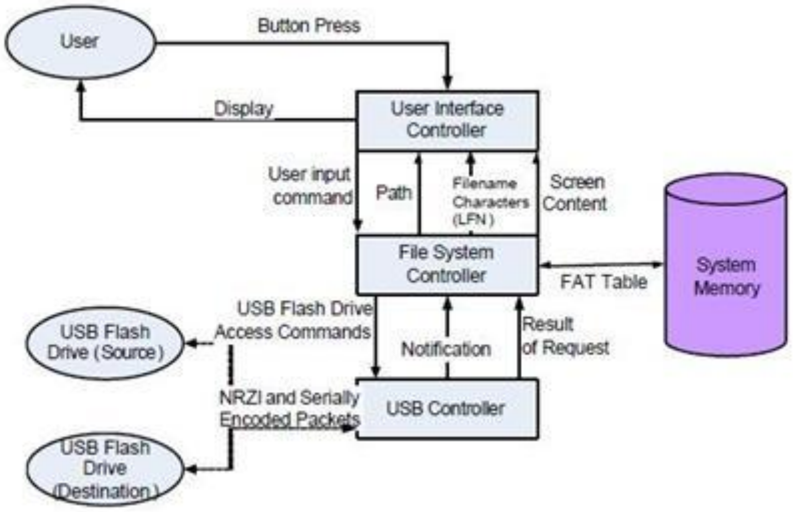


Fig. 2.2 General Block Diagram

Interfacing To an MMC or SD Card via SPI

Cyan Technology wrote an article entitled “Interfacing to an MMC or SD Card via SPI”. The article discusses that MMC card and the SD card are flash memory storage based devices. Both card types support proprietary data transfer protocols using four data bits, and are compatible though they have different initialisation. The major difference is that the SD card is designed to provide optional security by allowing encryption of the device contents. The MMC card supports additional bus widths (up to 8 bits). The SD card also supports several modes that are not present in the MMC card, including SDIO (secure digital input/output) that can be used as an external communications interface using the standard SD card format.

The application is based on the use of an MMC/SD card interface daughter board, connected to the eCOG1k evaluation board. The daughter board contains a card socket, as well as the necessary connections for monitoring and control. Figure 2.3 shows the schematic of the daughter board, including eCOG1k specific connections.

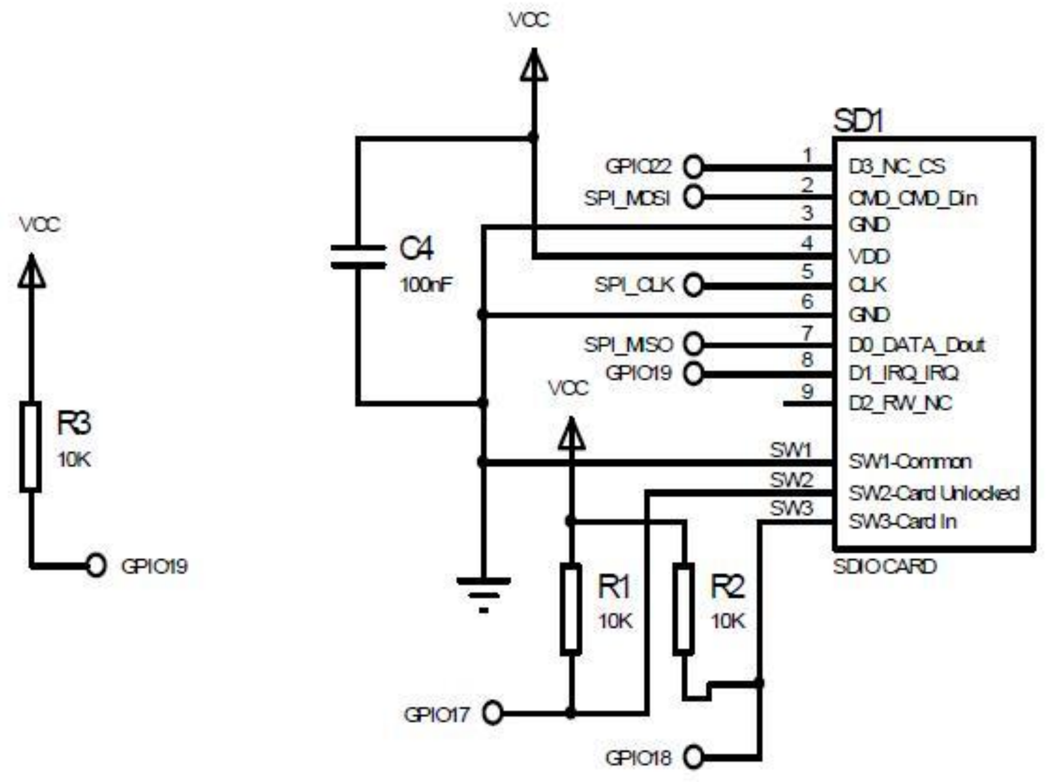


Figure 2.3 Connection Schematic for the MMC/SD card socket

Elements of a Transfer

According to Jan Axelson, who wrote the book entitled “USB COMPLETE Second Edition”, the elements of transferring data consist of the following (Please refer to Figure 2.4). All bus traffic travels to or from a device endpoint. The endpoint is a buffer that stores multiple bytes. Typically, the endpoint is a block of data memory or a register in the controller chip. The data stored at an endpoint may be received data or data waiting to transmit. The host also has buffers that hold received data and data waiting to transmit, but the host does not have endpoints. Instead, the host serves as the start and finish for communications with device endpoints.

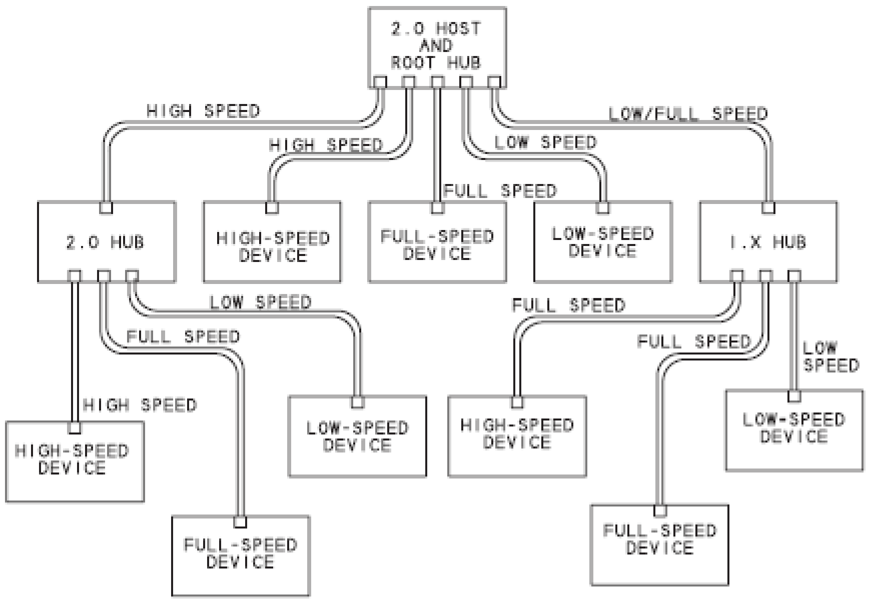


Figure 2.4 A USB 2.0 hub uses high speed whenever possible, switching to low and full speeds when necessary

Embedded Universal Serial Bus Host

A USB host controller is needed to access such USB storage devices. An embedded system that functions as a USB host for flash drives requires the following hardware (Figure

2.5). A microcontroller or intelligent hardware is required to manage the embedded system's operation. A USB host controller, which can be embedded in a microcontroller chip or on a separate chip that interfaces to the CPU, microcontroller, or other intelligent hardware, is needed. A flash drive is connected to a USB port on the host (Axelson, 2006).

The host manages the flow of data on the bus. Multiple peripherals may want to transfer data at the same time. The host controller divides the available time into segments called frames (on a full-speed host) or microframes (on a high-speed host). The host gives each transmission a portion of a frame or microframe. A frame is 1 millisecond; a microframe is 125 microseconds.

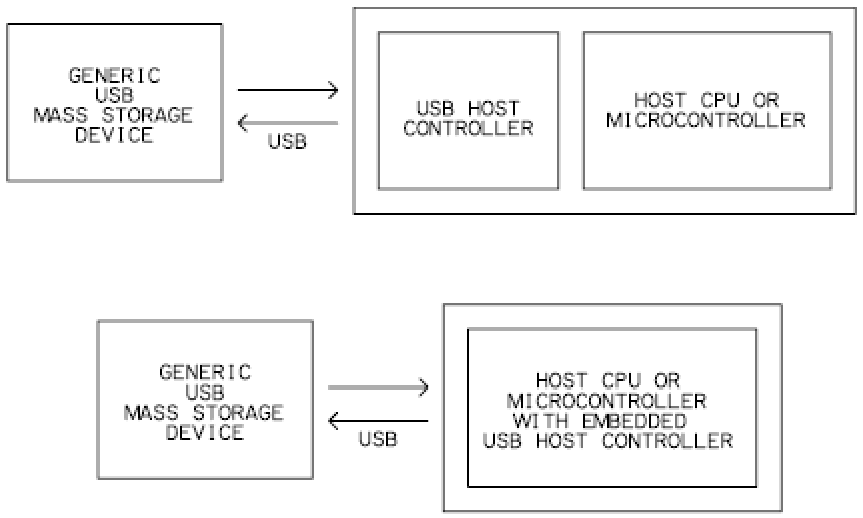

GENERIC USB MASS STORAGE DEVICE
USB
USB HOST CONTROLLER
HOST CPU OR MICROCONTROLLER
GENERIC USB MASS STORAGE DEVICE
USB
HOST CPU OR MICROCONTROLLER WITH EMBEDDED USB HOST CONTROLLER

Figure 2.5 To access generic USB mass-storage devices, an embedded system must contain a USB host controller, which can be on a separate chip or embedded in a microcontroller.

Bus Speeds

The USB 2.0 specification defines three bus speeds: high speed at 480 Mbps, full speed at 12 Mbps and low speed at 1.5 Mbps. In addition to data, the bus carries status, control and error-checking signals. All peripherals share the bus, so the data throughput is always less than the bit rate on the bus. In theory, on an otherwise idle bus, a full-speed device can transfer just over 1.2 MBps and a high-speed device can transfer more than 53 MBps. The actual rate of data transfer varies depending on the efficiency of the host's and device's programming, how busy the bus is, and hardware capabilities of the host and drive.

A review of the literature shows that there had been studies on host controller devices, storage controllers, and file transfer using embedded systems. However, these studies show that there is no existing system similar to this study. Existing studies only transfer files of both USB flash drive and no study existsin managing file between different devices like SD memory card and USB flash drive.

Chapter 3

FILE TRANSFER BETWEEN SD MEMORY CARD AND USB FLASH DRIVE THROUGH UART PIC INTERFACING WITH MASS STORAGE DEVICE CONTROLLER

Abstract

Flash memories like SD Memory card and USB Flash drive are widely used nowadays. However, transferring files within those storage devices cannot be done without the aid of desktop computers, laptop, etc. This paper presents the research and development of a device as an alternative to computers in transferring files between a USB Flash Drive and SD Memory Card. A mass storage device controller together with a microcontroller was used to initiate the operation of the device with a maximum capacity of 4 GB in the USB Flash Drive and 4 GB Memory Card. The study beganby constructing the main device down to its testing part in which the integrity and accuracy of files were considered. T-test was used as a statistical method to determine whether the difference between two proportions; that is the file accuracy of transferring file from the device and from the device of the previous research is significant in the Implementation of a USB Slave to Slave File Transfer Device Using Microcontrollers. Though some errors were produced in the testing process, the final result stated that there was no significant difference between the file transfer accuracy of the device and the previous study.

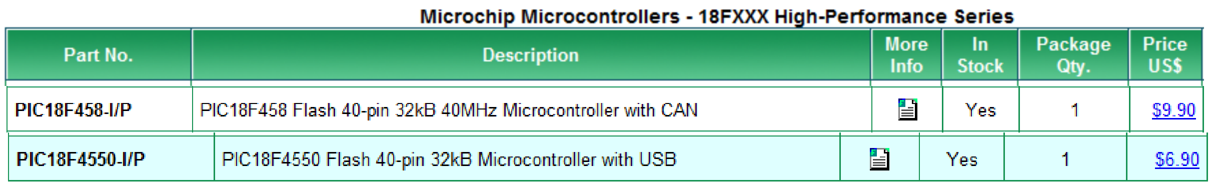
Keywords: USB Flash Drive, SD Memory Card, USB Embedded Host Controller, UART, File Transfer.

Introduction

Portability and mobility of storage devices like flash drives and memory cards are becoming so popular to carry and transport data on the go. Unlike CD/DVD, hard drive, floppy disk or tape drive, a flash memory has no moving parts, so it guarantees fast read/write speeds and more durable compared to other forms of storage media. Flash memory does not follow a standard form factor, so it can be designed or shaped depending on what the customer wants. Using a flash drive/memory card to transfer a file typically uses a computer (e.g. laptop, pc, netbook or tablet) to initialize mass storage communication which is not portable. Although a memory card is used in mobile phones withtheir diminutive sizes, it limits another memory card type of larger size to host them. Additionally, mobile phones cannot host a flash drive and memory card at the same time to transfer data. The proposed system bridges the gap of flash drive to memory card data transfer when a computer is not at hand. Moreover, booting a computer just to transfer a file/s brings annoyance.

In an earlier research, an embedded system is used wherein a slave to slave flash drive transfer makes it possible to transfer file/s using a programmable microcontroller with USB multi-role embedded host/peripheral controller and navigation control. This makes it possible to transfer data from a flash drive to another flash drive with 100% data integrity using MD5 checksum algorithm. That research also benchmarks the speed of data transfer depending on the size of the file and the depth of the file location in a folder. The device consumes a small amount of power only, has low memory footprint and less resources are utilized.

The research project being proposed in this study addresses the problem of flash drive to flash drive transfer without the use of a computer. The transfer of data between flash drive and memory card is not included. This study focuses on a design that is inexpensive and can readily be implemented without the use of a computer. Figure 1.1 shows the comparison of prices for the microcontrollers used by the previous research and the proposed system.



Microchip Microcontrollers - 18FXXX High-Performance Series

Part No.	Description	More Info	In Stock	Package Qty.	Price US$
PIC18F458-I/P	PIC18F458 Flash 40-pin 32kB 40MHz Microcontroller with CAN	📄	Yes	1	$9.90
PIC18F4550-I/P	PIC18F4550 Flash 40-pin 32kB Microcontroller with USB	📄	Yes	1	$6.90

Figure 1.1 Comparison of Prices of Microcontrollers used

It can be shown that PIC18F458, the one that was used in the previous research costs more than the microcontroller that will be used in this research project, which is PIC18F4550. The previous researchers used 20x4 Liquid Crystal Display (LCD) screen which is larger than the size of the screen that will be used in the device and therefore it will not be as expensive as the device used in the previous research.

It is important to note that the speed of data transfer is slow compared to a computer due to the host controller used which is USB 2.0 full speed. Theoretically, if the host microcontroller used follows the USB 2.0 high speed standard, the speed of data transfer should be much faster than the computer since there is no associated overhead in the bus compared to the computer.

The main objective of this study is to develop a device that can transfer files from flash drive to memory card and vice versa. Specific objectives include the following: to create an interface that will list all the file/s or folder/s in the current directory, to make sure that the user has the ability to navigate through it using the buttons; to have a successful file transfer between flash drive and memory card; and to determine whether there is a significant difference between the file accuracy of transferring file from USB Flash Drive to SD Memory Card or vice versa used in the present study and that of the previous study.Since there is only one display, the user can choose if he/she wants to see the contents of the flash drive or the memory card only. The compatibility between the flash drive and the memory card will depend on the capacity, standard used and additional features added by the manufacturer such as creating additional encrypted partition. The design is an extension of the previous research but of different implementation. The device should handle all file formats since it will not deal with the content of the file but rather its existence on the physical medium. However, it is important to mention that the study will concentrate on the integrity of the data copied. The user’s ability to transfer data between flash drive and memory card in a small devicecan reduce the utilization of computer which translates into saving power and resources. Additionally, it uses less amount of carbon footprint because of low heat emission and only uses portable electricity.The device emits low heat in such a way that it does not require that high amount of electricity, which produces heat whenever it functions. Also, compared to the computers used in transferring files between a USB flash drive and a memory card, the device has lesser components than the first. The device is said to use portable electricity in such a way that the device only uses 5V source of power which is low compared to other computers such as laptops which uses 15V to 20V. The device can also be battery operated. A portable embedded system capable of transferring data can save unexpected circumstances such as no electricity or no computer availability. It can be also interfaced with other designs provided that they have a common port to transmit data. The key importance of this design is to show how powerful an embedded system can be in specific applications wherein it keeps the cost at minimum due to its use of limited resources.

The device is capable of transferring data to and from the Secure Digital card and Universal Serial Bus disk. It supports only File Allocation Table (FAT/FAT32) file system to be able to access data. Since the memory card slot is Secure Digital compliant, it can mount other memory card/s that can be plugged via SD adapter. However, this is not covered by the study. Some SD type memory cards such as Secure Digital High Capacity (SDHC) and Secure Digital Extended Capacity (SDXC) are newer standards which are not supported by the device. Only flash drive/s with total space of 4.0 Gigabyte can be plugged on the USB port that conforms to USB 2.0 standard while external hard drives are not supported due to their high power requirements. The device will have LCD and directional buttons for file/folder navigation, as well as another button to show the available options. Copy, paste, delete and auto-replace of file only are supported due to time constraints in the development and the file name follows the 8.3 convention.

Methodology

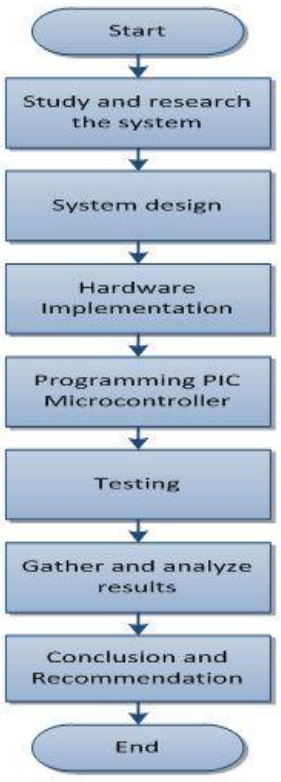

Start
Study and research the system
System design
Hardware Implementation
Programming PIC Microcontroller
Testing
Gather and analyze results
Conclusion and Recommendation
End

Figure 3.1 Methodology Flow Chart

The methodology in this study is summarized through a flow chart shown in Figure 3.1 and is followed with a detailed discussion for each method in order to meet the desired objectives.

A conceptual diagram is created by the researchers in order to visualize the main device to be made and also its functions. A study is conducted for the suitable materials needed to build the system. The conceptual diagram shown in Figure 3.2 includes the system design necessary in the hardware and software integration.

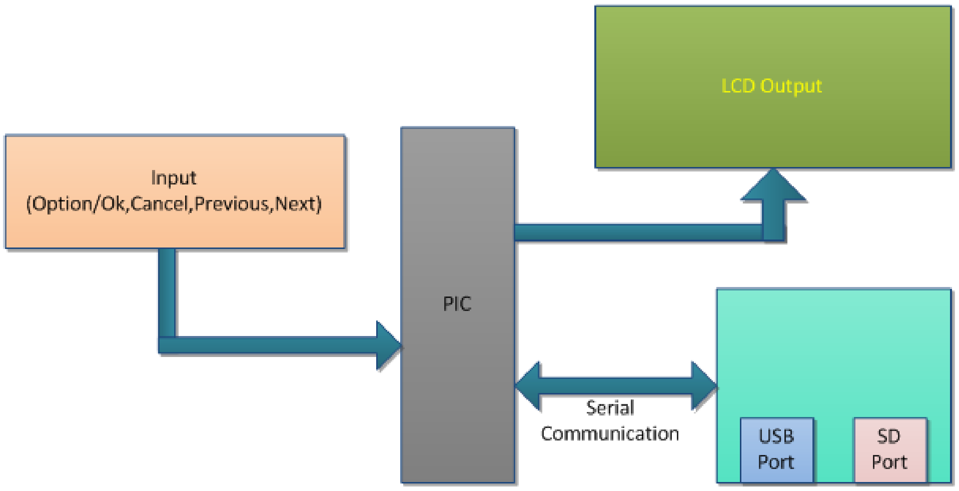


Figure 3.2 Conceptual Diagram

To be able to transfer file/s between a USB Flash Drive and an SD Memory Card, a device will be developed using a Mass Storage Device Controller and a microcontroller. The USB Device or Host Controller has built-in commands and file system support, particularly a FAT file system, which will be utilized to aid the file transfer. The microcontroller will be programmed and will be responsible for reading and writing files in a flash memory or flash drive. Specifically, the PIC18F4550 microcontroller will access the said host hardware controller via an asynchronous serial (UART) port. In UART mode, UART_TX pin is used to send data/responses to the microcontroller and UART_RX pin to receive commands/data from the microcontroller. The said two pins will be connected to the TX and RX UART pins of PIC18F4550 respectively. The command set provided by the mass storage host controller will be parsed by the microcontroller using C language. To make the device portable, a 9V battery supply to be regulated into 6V will be used. A schematic diagram shown in Figure 3.3 is simulated to come up with the desired output.

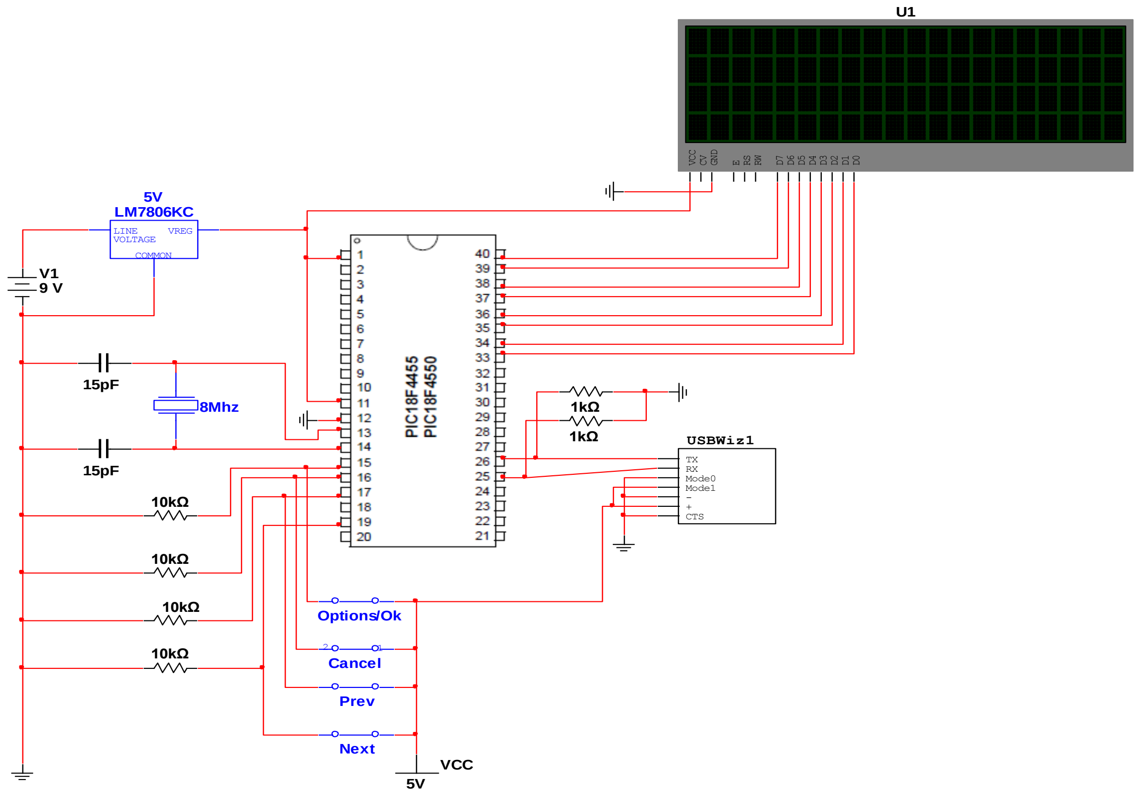


Figure 3.3 Schematic Diagram of the main device

The user interface will be connected to the host controller and PIC microcontroller, and will serve as the main component in which the user can view the file/s inside the flash drive and memory card. It will consist of a 4x20 Liquid Crystal Display (LCD), and buttons for selection. The PIC will accept inputs from the user through buttons and the PIC will determine what button is pressed to perform functions and processes that are needed. When a file has been successfully copied or transferred from one storage device to another, an indication will be displayed on the screen which confirms that the file is done copying.

The program flow chart for different subroutines for the PIC microcontroller in the following figures below shows the process in responding to the user activity, specifically in the file management between the SD Memory Card and USB Flash Drive.

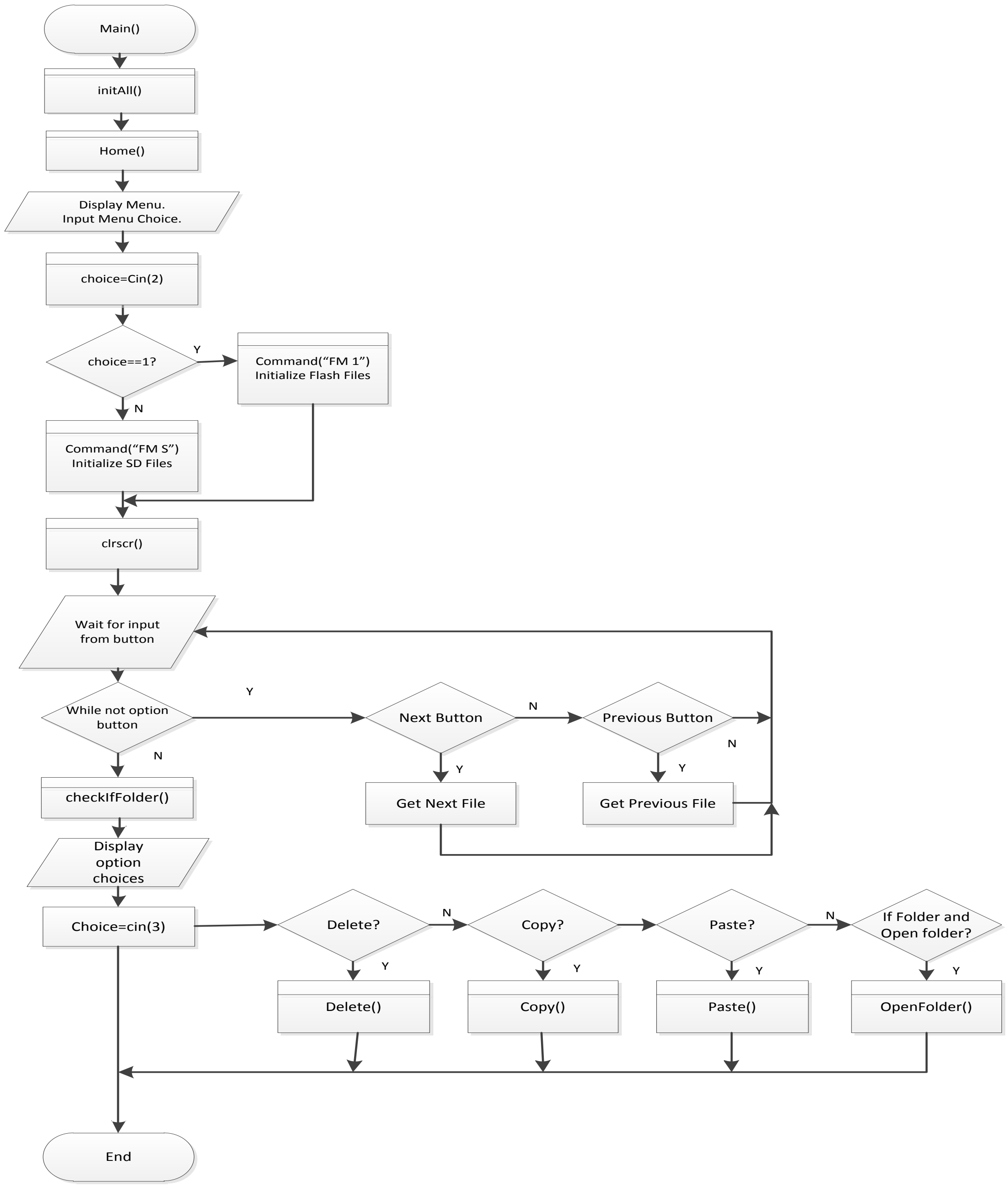


Figure 3.4 Program Flow Chart – Main()

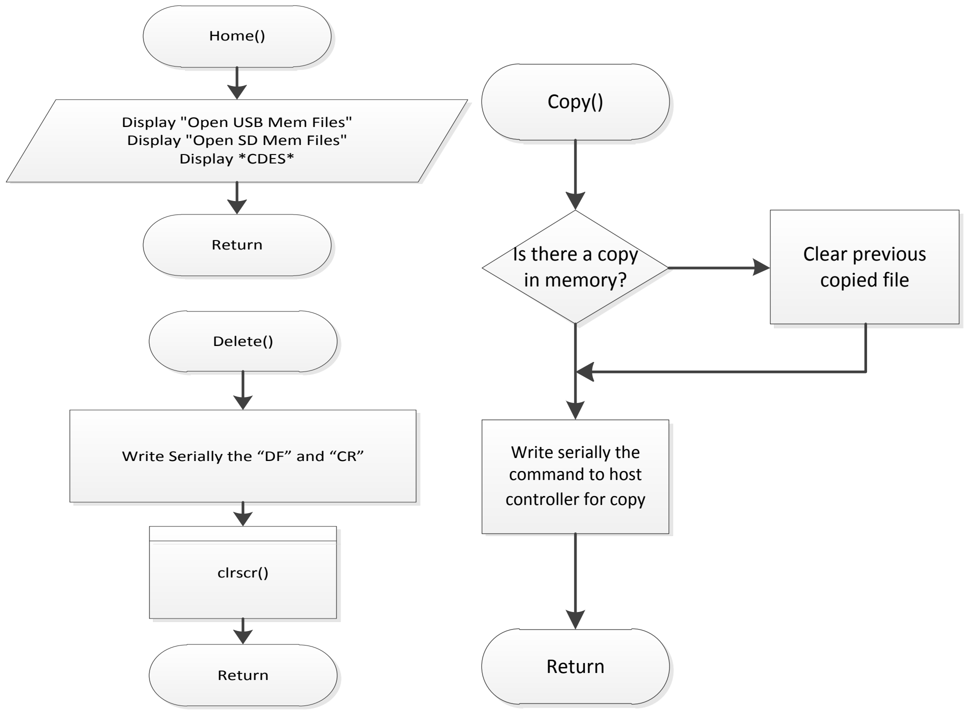

Home()
Display "Open USB Mem Files"
Display "Open SD Mem Files"
Display *CDES*
Return
Delete()
Write Serially the “DF” and “CR”
clrscr()
Return
Copy()
Is there a copy in memory?
Clear previous copied file
Write serially the command to host controller for copy
Return

Figure 3.5 Program Flow Chart – Home() and Copy()

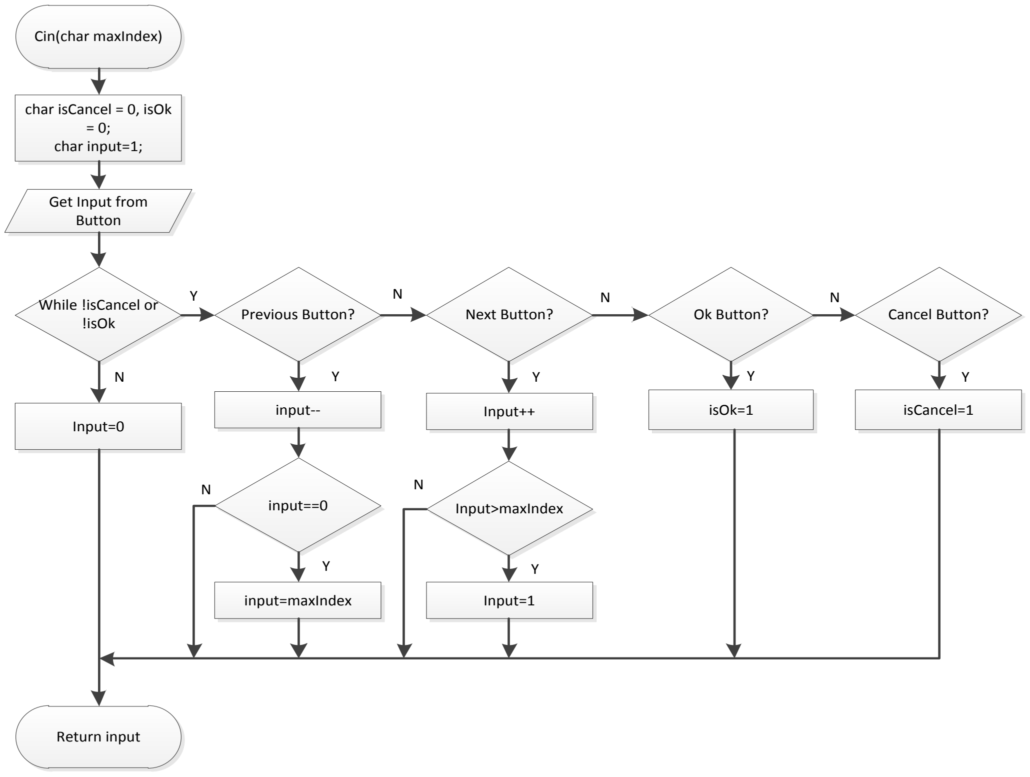


Figure 3.6 Program Flow Chart – Cin()

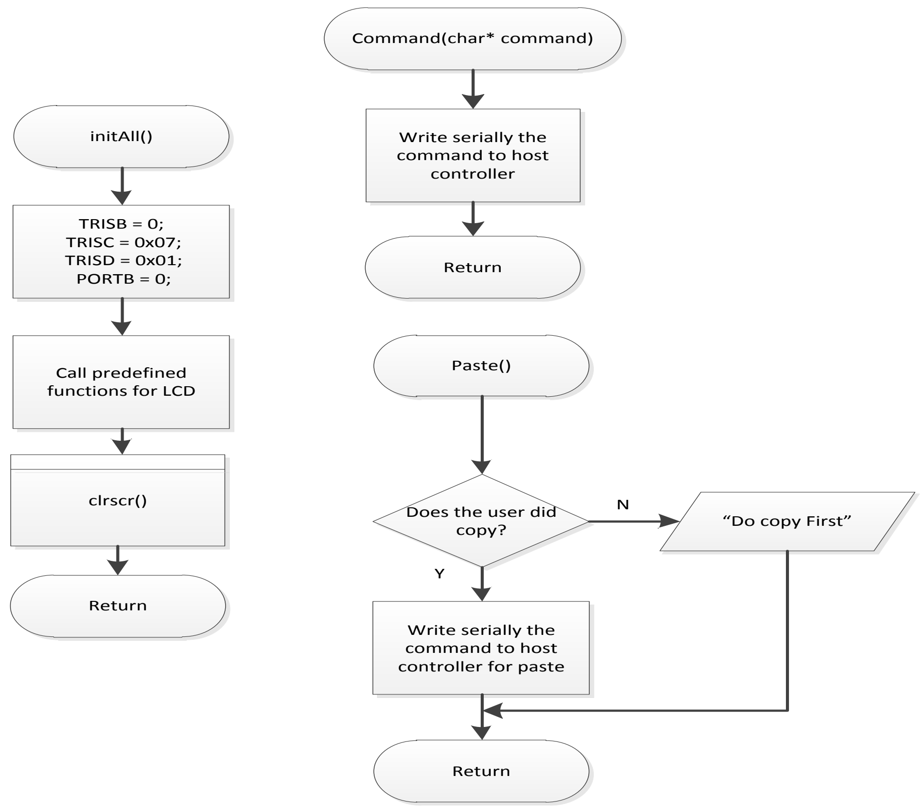


Figure 3.6 Program Flow Chart – initAll() and Command()

A transfer transaction is considered successful if, for a single file copy, the file has been copied completely. For folder copy, all the contents within the folder/directory concerned are copied completely; and this includes sub-folders and files within the folders to be copied. Testing the accuracy of a file or folder transfer is a means of testing the integrity of the written data in a USB flash drive or SD memory card. An experiment should be conducted in order to determine the file accuracy percentage by taking the ratio of the size of the successfully transferred file and the original file size in bytes. The Message Digest Algorithm-5 (MD5) will also be usedto verify data integrity through the creation of a 128-bit message digest from data input (which may be a message of any length) that is claimed to be as unique to that specific data as a fingerprint is to the specific individual.

Before the accuracy test, the information about the sample files to be copied must be gathered first. For the single file copy test, Table 3.1 summarizes the different information such as the individual file name, file name extension and the file size in bytes to be copied. Note that 15 sample files will be tested and copied from a 4 GB USB Flash Drive to a 4 GB SD Memory Card and 15 samples also from 4 GB SD Memory Card to a 4 GB USB Flash Drive, for a total of 30 samples.

Table 3.1 Single File Copy Test

File Name	File Extension	File Size in Bytes
1. File1	pka	85KB
2. File2	txt	1KB
3. File3	mp3	5,546KB
4. File4	exe	1,386KB
5. File5	xls	46KB
6. File6	docx	1,255KB
7. File7	bmp	2,903KB
8. File8	cache	8KB
9. File9	rar	4,365KB
10. File10	jpg	5,652KB
11. File11	exe	7,925KB
12. File12	pptx	302KB
13. File13	exe	3,783KB
14. File14	pdf	2,045KB
15. File15	avi	6,384KB

After the information has been gathered, the file accuracy test is next. Table 3.2 summarizes the file accuracy test results when files are being copied from a 4 GB SD Memory Card to a 4 GB USB Flash Drive.

Table 3.2 File Accuracy (from SD Memory Card to USB Flash Drive)

File Name	File Extension	File Size in Kbytes	Transfer Speed	Accuracy
1. File1	pka	85KB		
2. File2	txt	1KB		

3. File3	mp3	5,546KB		
4. File4	exe	1,386KB		
5. File5	xls	46KB		
6. File6	docx	1,255KB		
7. File7	bmp	2,903KB		
8. File8	cache	8KB		
9. File9	rar	4,365KB		
10. File10	jpg	5,652KB		
11. File11	exe	7,925KB		
12. File12	pptx	302KB		
13. File13	exe	3,783KB		
14. File14	pdf	2,045KB		
15. File15	avi	6,384KB		

On the other hand, Table 3.3 summarizes the file accuracy test results when files are being copied from a 4 GB Flash Drive to a 4 GB SD Memory Card. Note that the same files will be copied and for the purpose of testing, those files inside the Flash Drive that were copied during the previous test will be erased first to avoid overwriting of files and to give way to the next test.

Table 3.3 File Accuracy (from USB Flash Drive to SD Memory Card)

File Name	File Extension	File Size in Bytes	Transfer Speed	Accuracy
1. File1	pka	85KB		
2. File2	txt	1KB		
3. File3	mp3	5,546KB		
4. File4	exe	1,386KB		
5. File5	xls	46KB		
6. File6	docx	1,255KB		
7. File7	bmp	2,903KB		
8. File8	cache	8KB		
9. File9	rar	4,365KB		
10. File10	jpg	5,652KB		
11. File11	exe	7,925KB		
12. File12	pptx	302KB		
13. File13	exe	3,783KB		
14. File14	pdf	2,045KB		
15. File15	avi	6,384KB		

However, it is worthwhile to note that the accuracy of transferring files between a SD Memory Card and a USB Flash Drive cannot be assured to be a hundred percent accurate due to several factors.

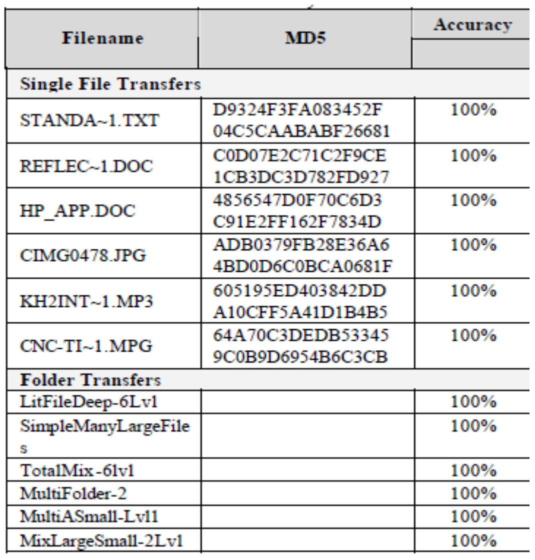
The accuracy results of Tables 3.2 and Tables 3.3 will then be compared individually with the past research in which a USB Slave to Slave File Transfer Device shown in Figure 3.5 is used and tested its file accuracy. The statistical treatment to be used is the two-proportion or the so called two-sample t-test. This test procedure will be made to determine whether the difference between two proportions; that is the file accuracy of transferring file from the device and from the previous research is significant. It would also verify if same actual files are copied and transferred directly from whichever source and destination, be it USB Flash Drive of SD Memory Card. Therefore, two kinds of statistically treated tests are expected at the end of this chapter; the first one will compare if the file accuracy of transferring file/s from SD Memory card to USB Flash Drive and from the previous research is significant. The second test will compare if the file accuracy of transferring file/s from USB Flash Drive to SD Memory card, used in the present study and from the previous research is significant

The two-sample t-test begins by stating the null and alternative hypothesis. In this case, the null hypothesis states that there is no difference between the two population proportions. Therefore, the null and alternative hypotheses will be stated in the following form.

$$H_0: P_1 = P_2$$

$$H_a: P_1 \neq P_2$$

The next step will be stating the rejection criteria, starting with the degrees of freedom (df) wherein the formula is stated to be the sum of the samples in both groups minus 2 (df) = (n1+n2)-2. In this case, df is defined to be (15+12)-2=25. Note that the sample size n1 is the sample size from Table 3.2 wherein the sizes are tested from the accuracy results from SD Memory Card to USB Flash Drive. On the other hand, n2 will be obtained from the accuracy results of the previous research wherein its sample size is equal to 12.



Filename	MD5	Accuracy
Single File Transfers		
STANDA~1.TXT	D9324F3FA083452F 04C5CAABABF26681	100%
REFLEC~1.DOC	C0D07E2C71C2F9CE 1CB3DC3D782FD927	100%
HP_APP.DOC	4856547D0F70C6D3 C91E2FF162F7834D	100%
CIMG0478.JPG	ADB0379FB28E36A6 4BD0D6C0BCA0681F	100%
KH2INT~1.MP3	605195ED403842DD A10CFF5A41D1B4B5	100%
CNC-TI~1.MPG	64A70C3DEDB53345 9C0B9D6954B6C3CB	100%
Folder Transfers		
LitFileDeep-6Lv1		100%
SimpleManyLargeFile s		100%
TotalMix-6lv1		100%
MultiFolder-2		100%
MultiASmall-Lvl1		100%
MixLargeSmall-2Lv1		100%

Figure 3.7 File Accuracy Test Results from USB Slave to Slave File Transfer Device

The next step is to determine the level of confidence – alpha. The t-distribution table is used to determine the critical value in order to get the level of confidence for a two-tailed t-test. In this case, the level of confidence for a degree of freedom equal to 25 is Alpha.05, with a critical value tcv= 2.060 (See Appendix).

The next step is to compute for the standard error (SE) of the sampling distribution difference between two proportions.

$$S_{\overline{x}_1-\overline{x}_2} = \sqrt{\frac{(n_1-1)s_1^2+(n_2-1)s_2^2}{n_1+n_2-2}}\sqrt{\frac{n_1+n_2}{n_1 n_2}}$$

where n_1 is the size of Table 3.2, and n_2 is the size of Figure 3.5. s_1^2 is the variance computed in sample 1 and s_2^2 is the variance computed in sample 2. The variance is computed using the $s^2 = \frac{\sum(X-M)^2}{N-1}$ formula,

where X is the accuracy percentage of each file, M is the mean or the average accuracy percentage of each sample and N is the number of scores in each sample.

Lastly, the t-ratio is computed using the formula, Write the t-ratio formula, t = (mean1 - mean2) / sqrt((variance1 / size of sample1) + (variance2 / size of sample2)) or

$$t = \frac{\overline{X_1} - \overline{X_2}}{S_{\overline{x}_1-\overline{x}_2}}$$

If the t-value is greater than the critical value 2.060, the Null Hypothesis will be rejected and say that the two samples have the difference. Otherwise, a significant difference is not found and fails to reject the Null Hypothesis.

Results and Discussion

In performing the MD-5 checksum algorithm for both file transfer between USB Flash Drive and SD Memory Card, a table is created to gather the data such as the file name and extension of the sample files, and the source and destination's (from USB Flash Drive to

SD Memory Card or vice versa) actual checksums. Another column is added to compute for the accuracy percentage per file transfer. In this case, the computation is based on the ratio of the 32-hexadecimal numbers of the destination from the source.

Table 3.4 summarizes the data gathered in testing the file accuracy from SD Memory Card to USB Flash Drive. Based from the results obtained, some file transfers were not successfully established. Specifically, the destination checksum of file1.pka is different from its source by 3 hexadecimal numbers. Same is through with file4.exe and file14.pdf lastly, for file15, in which an error occurred during the file transfer and therefore, no destination checksum was made. The file accuracy percentage for each field is 100% if the source is equal to the destination. In the case of the three files that yield to different results other than that, the computation using the file accuracy percentage formula is done.

Table 3.4 File Accuracy Percentage using MD-5 Checksum (from SD Memory Card to USB Flash Drive)

File Name	**Source (MD5)**	**Destination (MD5)**	**Accuracy**
File1.pka	b7ca1cc671e70c0cd350833e 2b22e1f3	b7ca1cc671e10a2cd350833e 2b22e1f3	90.625%
File2.txt	88fcc62dbc58d1f4b3fa8dedb b6aaf38	88fcc62dbc58d1f4b3fa8ded bb6aaf38	100%
File3.mp3	641846409979 7641232d7dd c9a96b6ee	641846409979 7641232d7dd c9a96b6ee	100%
File4.exe	c7722c4ec4fc3ac818658ef00 a49e2b4	c7722c4eca3dbad818658ef0 0a49e2b4	81.25%
File5.xls	13ae3a6a1db81cfcef570d0a2 4ea09e2	13ae3a6a1db81cfcef570d0a 24ea09e2	100%
File6.docx	f59cb7d6c15ffe465a76c1b01 66f772c	f59cb7d6c15ffe465a76c1b0 166f772c	100%
File7.bmp	033d47aab079aa403f96f033c 78eb5ec	033d47aab079aa403f96f033 c78eb5ec	100%
File8.cache	c8221439a4d15092bc29f0f0 7909fef7	c8221439a4d15092bc29f0f0 7909fef7	100%
File9.rar	5d84e04b9c11f600e9daaae7 4dec0b9d	5d84e04b9c11f600e9daaae7 4dec0b9d	100%

File10.jpg	f417634ec8dbfa3fb6c57c36a 614398d	f417634ec8dbfa3fb6c57c36a 614398d	100%
File11.exe	30be90d08f03a4a749f661b2 efaf01ec	30be90d08f03a4a749f661b2 efaf01ec	100%
File12.pptx	02e3ac32137dd06b10df59b4 46885fec	02e3ac32137dd06b10df59b4 46885fec	100%
File13.exe	3a9500a528286d773309bc1d 21d91469	3a9500a528286d773309bc1 d21d91469	100%
File14.pdf	c7722c4ec4fc3ac818658ef00 a49e2b4	c7722c4ec4fc3ac818658efaa a83f7c4	71.875%
File15.avi	a115b7bb3d272fb148e1fb75 85ffd806	----	0%

On the other hand, Table 3.6 summarizes the file accuracy percentage from USB Flash Drive to SD Memory Card. Based from the results in the table; the checksum of the source is equal to that of the destination for all file transfer operations. Therefore, the accuracy percentage for all files being transferred from USB Flash Drive to SD Memory Card is 100%.

Table 3.5 File Accuracy Percentage using MD-5 Checksum (from USB Flash Drive to SD Memory Card)

File Name	**Source (MD5)**	**Destination (MD5)**	**Accuracy**
File1.pka	b7ca1cc671e70c0cd350833e 2b22e1f3	b7ca1cc671e70c0cd350833e 2b22e1f3	100%
File2.txt	88fcc62dbc58d1f4b3fa8dedb b6aaf38	88fcc62dbc58d1f4b3fa8ded bb6aaf38	100%
File3.mp3	6418464099797641232d7dd c9a96b6ee	6418464099797641232d7dd c9a96b6ee	100%
File4.exe	c7722c4ec4fc3ac818658ef00 a49e2b4	c7722c4ec4fc3ac818658ef0 0a49e2b4	100%
File5.xls	13ae3a6a1db81cfcef570d0a2 4ea09e2	13ae3a6a1db81cfcef570d0a 24ea09e2	100%
File6.docx	f59cb7d6c15ffe465a76c1b01 66f772c	f59cb7d6c15ffe465a76c1b0 166f772c	100%
File7.bmp	033d47aab079aa403f96f033c 78eb5ec	033d47aab079aa403f96f033 c78eb5ec	100%
File8.cache	c8221439a4d15092bc29f0f0 7909fef7	c8221439a4d15092bc29f0f0 7909fef7	100%
File9.rar	5d84e04b9c11f600e9daaae7 4dec0b9d	5d84e04b9c11f600e9daaae7 4dec0b9d	100%
File10.jpg	f417634ec8dbfa3fb6c57c36a 614398d	f417634ec8dbfa3fb6c57c36a 614398d	100%
File11.exe	30be90d08f03a4a749f661b2 efaf01ec	30be90d08f03a4a749f661b2 efaf01ec	100%

File12.pptx	02e3ac32137dd06b10df59b446885fec	02e3ac32137dd06b10df59b446885fec	100%
File13.exe	3a9500a528286d773309bc1d21d91469	3a9500a528286d773309bc1d21d91469	100%
File14.pdf	c7722c4ec4fc3ac818658ef00a49e2b4	c7722c4ec4fc3ac818658ef00a49e2b4	100%
File15.avi	a115b7bb3d272fb148e1fb7585ffd806	a115b7bb3d272fb148e1fb7585ffd806	100%

The file accuracy results obtained in Tables 3.4 and 3.5 are transferred to the last columns of Tables 3.2 and 3.3 so as to complete the File Accuracy tables of both scenarios.

Table 3.2 File Accuracy (from SD Memory Card to USB Flash Drive)

File Name	File Extension	File Size in Kbytes	Transfer Speed	Accuracy
1. File1	pka	85KB	0:1	90.625%
2. File2	txt	1KB	0:1	100%
3. File3	mp3	5,546KB	1:50	100%
4. File4	exe	1,386KB	0:29	81.25%
5. File5	xls	46KB	0:1	100%
6. File6	docx	1,255KB	0:25	100%
7. File7	bmp	2,903KB	1:00	100%
8. File8	cache	8KB	0:1	100%
9. File9	rar	4,365KB	1:32	100%
10. File10	jpg	5,652KB	1:58	100%
11. File11	exe	7,925KB	2:39	100%
12. File12	pptx	302KB	0:3	100%
13. File13	exe	3,783KB	1:16	100%
14. File14	pdf	2,045KB	0:53	71.875%
15. File15	avi	6,384KB	0:2	0%

Table 3.3 File Accuracy (from USB Flash Drive to SD Memory Card)

File Name	File Extension	File Size in Bytes	Transfer Speed	Accuracy
1. File1	pka	85KB	0:3	100%
2. File2	txt	1KB	0:1	100%
3. File3	mp3	5,546KB	2:31	100%
4. File4	exe	1,386KB	0:22	100%
5. File5	xls	46KB	0:2	100%
6. File6	docx	1,255KB	0:27	100%
7. File7	bmp	2,903KB	1:17	100%
8. File8	cache	8KB	0:1	100%
9. File9	rar	4,365KB	2.17	100%
10. File10	jpg	5,652KB	2:48	100%
11. File11	exe	7,925KB	3:01	100%
12. File12	pptx	302KB	0:7	100%
13. File13	exe	3,783KB	1:34	100%

14. File14	pdf	2,045KB	0:52	100%
15. File15	avi	6,384KB	2:38	100%

Based from the results gathered, differences can be seen during the file transfer of the device between the SD Memory Card and USB Flash Drive especially on the transfer speed of the device. It can be observed that in terms of the transfer speed, transferring files from USB Flash Drive to SD Memory Card requires more time than transferring files from SD Memory Card to USB Flash Drive when using the device. However, in terms of the file accuracy percentage, Table 3.2 has obtained file accuracy less than 100% which means that errors were produced during the testing process and the transfer of a single file was unsuccessful for some.

The accuracy, M, for the sample files n_1 transferred as shown in Table 3.2 is (90.625+100+100+81.25+100+100+100+100+100+100+100+100+100+71.875+0) / 15 = 89.5833.

The variance s_1^2 for sample n_1 was computed to be $((90.625-89.5833)^2 + (100-89.5833)^2 + (100-89.5833)^2 + (81.25-89.5833)^2 + (100-89.5833)^2 + (100-89.5833)^2 + (100-89.5833)^2 + (100-89.5833)^2 + (100-89.5833)^2 + (100-89.5833)^2 + (100-89.5833)^2 + (100-89.5833)^2 + (100-89.5833)^2 + (71.875-89.5833)^2 + (0-89.5833)^2) / 14 = 685.9189$

Since all the 12 files are 100% accurate for sample n_2 as shown in Figure 3.5, which is the data from the previous study, the mean M is equal to 100 and the variance s_1^2 is equal to 0.

The standard error (SE) of the sampling distribution difference between two proportions was obtained by $\sqrt{\frac{14(685.9189)+11(0)}{25}}\sqrt{\frac{15+12}{15(12)}} = 7.5901$.

Finally, the t-ratio, t, was obtained to be $\frac{89.5833-100}{7.5901}$ = -1.3724. Since -1.3724<2.060, the null hypothesis is not rejected. It then implies that there is no significant difference between the accuracy of the transferring files from SD Memory Card to USB Flash Drive and the previous study which is USB Slave to Slave File Transfer Device.

For the second part of the statistical treatment, Table 3.3 is then compared to Figure 3.5. However, since the file accuracy percentage in Table 3.3 is also the same as the latter, it is automatically concluded that $P_1 = P_2$ and the null hypothesis is not rejected, implying that there is no significant difference between the file accuracy of the transferring files from USB Flash Drive to SD Memory Card and the previous study which is USB Slave to Slave File Transfer Device.

Conclusion

The development of the main device for this study requires both the hardware and software to be properly coordinated with each other to meet the objectives of this study.The researchers were able to develop the device using Mass Storage Device Controller and microcontroller as the major components of the hardware.

The user interface of the system provides the capability to browse the contents of the specified file or folder. To differentiate a file from a folder, a folder has a slash symbol (/) at the end of the name. The user navigates through the Options/OK button, cancel button and to scroll the contents using the previous and next button of the hardware.

The design has successfully transferred data between the flash drive and SD (Secure Digital) memory card. The file management features of the device are configured correctly to respondto the user activities. It is also able to host memory cards that can fit in SD card adapter. The accuracy of the file transferred is verified using MD5 checksum which checks for file integrity.

During the testing process, the file transfer from SD Memory Card to USB Flash Drive had a higher transfer speed compared to the other way around even when same exact files are being tested. Some errors were produced especially in the part of the first scenario wherein the accuracy of transferring files from SD Memory Card to USB Flash Drive is less than a hundred percent. Factors affecting such may include both external and internal device errors. External factors include the noise and interferences obtained by the physical device. Internal device errors include the program inside the USB host controller. Nevertheless, after the statistical treatment, it is concluded that there is no significant difference between the file transfer accuracy of the device and device used in the previous study.

References

[1] Mark Alvin U. Chua Charles D. Jorge Ana Marian M. Pedro Brian Emmanuel G. Tam Gregory G. Cu, “Implementation of a USB Slave to Slave File Transfer Device Using Microcontrollers”

[2] Jan Axelson, “USB COMPLETE Second Edition,” Madison, WI: Lakeview Research LLC, 2004

[3] Jan Axelson, “Serial Port Complete Second EditionCOM Ports, USB Virtual COM Ports, and Ports for Embedded Systems”,Dec. 2007,

[4] Jan Axelson, USB Mass Storage: Designing and Programming Devices and Embedded Hosts

[5] GHI Electronics, “USBWiz User Manual”, Rev. 2.27 April 2009

[6] GHI Electronics, “USBWiz OEM Manual”, 2006

[7] Dogan Ibrahim, Advanced PIC Microcontroller Projects in C: From USB to RTOS with the PIC 18F Series

[8] MikroElektronika, C Compiler for Microchip PIC Microcontrollers User’s Manual

[9] John Morton, The PIC Microcontroller: Your Personal Introductory Course, Third Edition

[10] Cyan Technology, “Interfacing to an MMC or SD Card via SPI”, 2008

Chapter 4

CONCLUSION

The development of the main device for this study requires both the hardware and software to be properly in accordance with each other and to meet the objectives of this study.The researchers were able to develop the device using Mass Storage Device Controller and microcontroller as the major components of the hardware.

The user interface of the system provides the capability to browse the contents of the specified file or folder. To differentiate a file from a folder, a folder has a slash symbol (/) at the end of the name. The user navigates through the Options/OK button, cancel button and to scroll the contents using the previous and next button of the hardware.

The design has successfully transferred data between the flash drive and SD (Secure Digital) memory card. The file management features of the device are configured correctly to respond for the user activities. It is also able to host memory cards that can fit in SD card adapter. The accuracy of the file transferred is verified using MD5 checksum which checks for file integrity.

During the testing process, the file transfer from SD Memory Card to USB Flash Drive has a higher transfer speed compared to the other way around even when same exact files are being tested. Some errors were produced when the accuracy of transferring files from SD Memory Card to USB Flash Drive was tested and is less than a hundred percent. Factors affecting such may include both external and internal device errors. External factors include the noise and interferences obtained by the physical device. Internal device errors include the program inside the USB host controller. Nevertheless, after the statistical treatment, it is concluded that there is no significant difference between the file transfer accuracy of the device and the device used in previous study.

Chapter 5

RECOMMENDATION

There are still several functions that need to be improved in the prototype of this study. As designed, the directories cannot be transferred between SD Memory card and USB Flash Drive. Therefore, it is recommended that the system be reprogrammed to transfer folders between the two memories. Searching files can also be difficult especially when traversing many files and folders. It may also be time consuming for the user especially when there are a lot of files in the memory. To make the system more efficient, a search or find function should be integrated to the system.

The LCD is limited to 4 x 20 characters only. As a result, only one file can be viewed at a time. Simultaneous viewing for both drives can help the user look for files and switch between two drives faster. Thus, instead of using a normal 4 x 20 Liquid Crystal Display, future researchers can use a larger size of it. LiquidGraphics Crystal Display can also be used for a high quality of visual experience; however it can be a trade-off with the low power advantage of the device.

Since the device is capable of transferring data to and from the Secure Digital card and Universal Serial Bus disk only, it is recommended to have additional memory slots of some memory cards such as Memory Stick Pro and the like. This will enable the microcontrollerto be used to handle numerous input/output ports for serial communication.

The filenames to be displayed in the device are limited to only 8 characters long and the file extension is limited to 3 characters only because of the FAT/FAT32 implications. If ever future researchers advise the support of New Technology File System (NTFS) on the device, the USB host controller should be replaced with a host controller to support it.The need to implement the system using larger memory size is also recommended.

Copy, paste, delete and auto-replace of file are the only features of the current device. Additional file management features should be added to make the device more useful and beneficial since it can be easily implemented inside the program code of the device as long as the right algorithm and structure is acquired.

REFERENCES

[1] Mark Alvin U. Chua Charles D. Jorge Ana Marian M. Pedro Brian Emmanuel G. Tam Gregory G. Cu, “Implementation of a USB Slave to Slave File Transfer Device Using Microcontrollers”

[2] Jan Axelson, “USB COMPLETE Second Edition,” Madison, WI: Lakeview Research LLC, 2004

[3] Jan Axelson, “Serial Port Complete Second EditionCOM Ports, USB Virtual COM Ports, and Ports for Embedded Systems”,Dec. 2007,

[4] Jan Axelson, USB Mass Storage: Designing and Programming Devices and Embedded Hosts

[5] GHI Electronics, “USBWiz User Manual”, Rev. 2.27 April 2009

[6] GHI Electronics, “USBWiz OEM Manual”, 2006

[7] Dogan Ibrahim, Advanced PIC Microcontroller Projects in C: From USB to RTOS with the PIC 18F Series

[8] MikroElektronika, C Compiler for Microchip PIC Microcontrollers User’s Manual

[9] John Morton, The PIC Microcontroller: Your Personal Introductory Course, Third Edition

[10] Cyan Technology, “Interfacing to an MMC or SD Card via SPI”, 2008

APPENDICES

Appendix A

Program Listing

```
//LCD declarations
sbit LCD_RS at RD2_bit;sbit LCD_EN at RD3_bit;
sbit LCD_D7 at RB7_bit;sbit LCD_D6 at RB6_bit;sbit LCD_D5 at RB5_bit;sbit LCD_D4 at
RB4_bit;sbit LCD_D3 at RB3_bit;sbit LCD_D2 at RB2_bit;sbit LCD_D1 at RB1_bit;sbit
LCD_D0 at RB0_bit;

sbitLCD_RS_Direction at TRISD2_bit;sbitLCD_EN_Direction at TRISD3_bit; sbit
LCD_D7_Direction at TRISB7_bit;sbit LCD_D6_Direction at TRISB6_bit;sbit
LCD_D5_Direction at TRISB5_bit;sbit LCD_D4_Direction at TRISB4_bit;sbit
LCD_D3_Direction at TRISB3_bit;sbit LCD_D2_Direction at TRISB2_bit;sbit
LCD_D1_Direction at TRISB1_bit;sbit LCD_D0_Direction at TRISB0_bit;

//GLOBAL VARIABLES
char CR = 0x0D;        //CARRIAGE RETURN
char BS = 0x08;        //BACKSPACE

char* devOn;

charfName[12],fAttrib[2],fSize[8];
char isEof = 0;

int nextMinus1 = 0;
intprevCtr = 0;

charcopyBufferName[12];
char copyBufferSize[8];
char isCopy = 0;

voidclrscr(){
 Lcd_Cmd(_LCD_CLEAR);
}

void Command(char* command){
UART1_Write_Text(command);
```

```
UART1_Write(CR);
}


voidCout(char lineNum,char* fileLineString){ Lcd_Out(lineNum,1,fileLineString);
}

voidgotoxy(char row,char col){ LCD_Out(row,col,"");
}

void Home(){
Cout(1,"Open USB Mem Files"); Cout(2,"Open SD Mem Files");
Lcd_Out(4,8,"*CDES*");
}



charcheckIfFolder(char fAttrib[]){ if((fAttrib[0] == '1' &&fAttrib[1] == '0')
    || (fAttrib[0] == '3' &&fAttrib[1] == '2')
    || (fAttrib[0] == '1' &&fAttrib[1] == '2')){ return 1;
   }else{ return 0;
   }
}

voidWritePerCh(char size,char array[],char type){ char ctrSize = 0;
 if(type == 0){                              // 0 if write in LCD
  while(ctrSize< size){
   if(array[ctrSize]!=' '){
    Lcd_Chr_CP(array[ctrSize]);
   }
   ctrSize++;
  }
 }else if(type == 1){                    //if 1 write in UART
  while(ctrSize< size){
   if(array[ctrSize]!=' '){
    UART1_Write(array[ctrSize]);
   }
   ctrSize++;
  }
 }
}

void FilesDisplay(){ clrscr(); Lcd_Out(1,5,devOn); gotoxy(3,1);
   WritePerCh(8,fSize,0); gotoxy(2,1); WritePerCh(12,fName,0);
   if(checkIfFolder(fAttrib)){

    LCD_Cmd(_LCD_MOVE_CURSOR_LEFT); LCD_Chr_Cp('/');
   }
}


chardummy_char; void Dummy_Read(){ int timeout = 0;
while(timeout<30000){
 if(UART1_Data_Ready()){ dummy_char = UART1_Read();
 }
 timeout++;
}
}

voidinitAll(){ TRISB = 0; TRISC = 0x07; TRISD = 0x01;
 PORTB = 0; LCD_Init();
 Lcd_Cmd(_LCD_UNDERLINE_ON); Lcd_Cmd(_LCD_BLINK_CURSOR_ON);
 UART1_Init(9615);
 Dummy_Read();
 Cout(1,"Initializing..."); Delay_ms(1000); clrscr();
}

charCin(char maxIndex){ char isCancel = 0, isOk = 0; char input=1;

  while(!isCancel&& !isOk){ //buttons
   if(PORTC.F2){ //previous Delay_ms(500);
    input--; if(input == 0){
     input = maxIndex;
    }
   }
   if(PORTD.F0){ //next Delay_ms(500); input++;
    if(input >maxIndex){ input = 1;
    }
   }
   if(PORTC.F0 ){ // OK is pressed Delay_ms(500);
    isOk = 1;
   }
   if(PORTC.F1 ){ // Cancel is pressed Delay_ms(500);
    isCancel = 1; input = 0;
   }
   //end of buttons determination

    if(input == 1){ gotoxy(1,1);
    }else if(input == 2){ gotoxy(2,1);
    }else if(input == 3){ gotoxy(3,1);
    }else if(input == 4){ gotoxy(1,15);
    }else if(input == 5){ gotoxy(2,15);
    }else if(input == 6){ gotoxy(3,15);
    }

  } //end of while

  return input;
}   //end of Cin function

charHandleErrorCodes(){ // function HandleErrorCodes char errorCode[10] = "";
charerrorCodeCtr = 0; char noError = 0;

  while(errorCodeCtr<4){ if(UART1_Data_Ready()){ errorCode[errorCodeCtr] =
   UART1_Read(); //Lcd_Chr_CP(errorCode[errorCodeCtr]); Delay_ms(1);
    errorCodeCtr++;
   }
  }

//start of identifying the errors if(strstr(errorCode,"00")){
noError = 1;
}
returnnoError;
}
voidgetFileString(){   //GETFILESTRING

charexactFullFileStr[25];   char   fileLineStr[40]   =   "";   char*
tempPtr;



charfileLineCtr = 0; int timeout = 0;


//getFileStringAgain: Command("NF"); while(timeout<10000){

if(UART1_Data_Ready()){ fileLineStr[fileLineCtr++] = UART1_Read();
```

```
}

timeout++; }//end of while

if(strstr(fileLineStr,"!4D")== 0){

tempPtr = strchr(fileLineStr,CR);      //point to the first CR
tempPtr++;      //inc to alis the CR, point to next

strncpy(exactFullFileStr,tempPtr,24);    //copy the first 24 char so the exess are remove

//get the fName only

strncpy(fName,exactFullFileStr,12);      //copy the first 12 char to fName

//get the attrib

fAttrib[0] = exactFullFileStr[13]; fAttrib[1] = exactFullFileStr[14];

//get the size

tempPtr = strrchr(fileLineStr,' ');        //point to the last ' '

tempPtr++;      //then inc to point to the next char

strncpy(fSize,tempPtr,8);       //then copy 8 chars

}else{

isEof = 1;

}

} //end of function getFileString()



void Copy(){

}

void Paste(){
UART1_Write_Text("OF 1W>");

WritePerCh(12,copyBufferName,1);
```

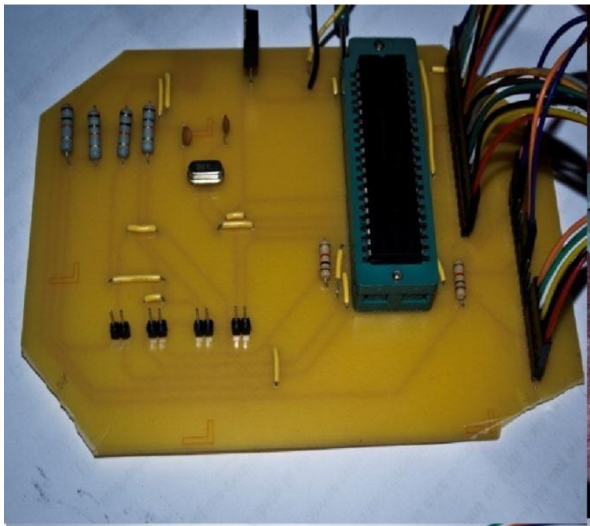
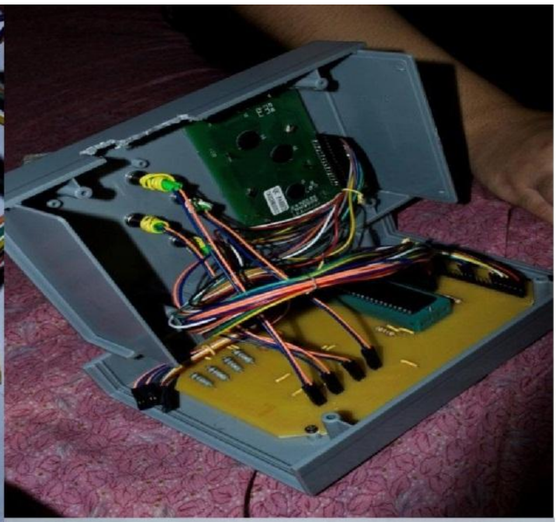
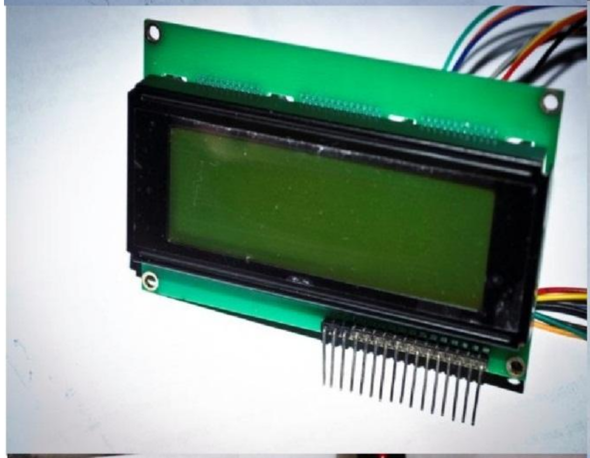
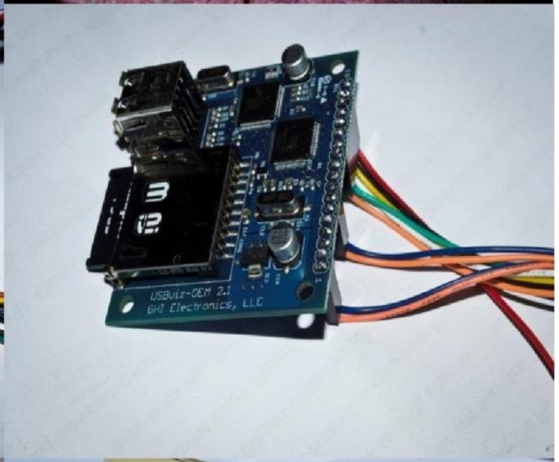
```
UART1_Write(CR);

Dummy_Read();


UART1_Write_Text("RW 0 1>"); WritePerCh(8,copyBufferSize,1); UART1_Write(CR);

clrscr(); Cout(2,"Copying");

while(!UART1_Data_Ready()){

}

if(HandleErrorCodes()){ clrscr(); Cout(2,"Success");

//UART1_Write_Text("CF 0"); UART1_Write_Text("CF 1"); Delay_ms(1000);

}else{ Cout(2,"Error");

}

}



//MAIN FUNCTION!!!!! //variables used in main function char choice;

void main(){    // Main function

initAll();

Start:


clrscr(); Home(); inputAgain:

choice = Cin(2); //the the user will wait for the user input if(choice == 0){

gotoinputAgain; }else if(choice == 1){


    Command("FM 1");
    devOn = "Flash Files";
  }else{
   Command("FM S");
   devOn = "SD Mem Files";
  }

if(!HandleErrorCodes()){
 Dummy_Read();
 Cout(3,"Error");
 gotoinputAgain;
}else{
  // view files in storage
  JustOpen:
  clrscr();

  Command("IL");
  Dummy_Read();
  Cout(1,"Loading...");
  Delay_us(3000);
  nextMinus1 = 0;

  Current:     //if cancel is pressed
  clrscr();
  Cout(2,"Press Next");
  do{
    if(PORTD.F0){       //this is the next button     (->)pressNext:
     Delay_ms(500);
     nextMinus1++;
     getFileString();
     if(isEof){
      nextMinus1--;
      isEof = 0;
      Cout(4,"EOF, Press Previous");
     }else{

     FilesDisplay();

     }
    } // end of next button


    if(PORTC.F2){ //this is the previous button (<-)
     Delay_ms(500);


      prevCtr = 1;
      nextMinus1--;
      if(nextMinus1 <= 0){
      nextMinus1++; }else{
        Command("IL");
        Dummy_Read();
        while(prevCtr<=nextMinus1){
        getFileString();
         prevCtr++;
        } //end of while

       FilesDisplay();

      } //end of else
    } //end of if previous button
  }while(!PORTC.F0); //option
  //go here if options is pressed

  if(PORTC.F0){        //ok so now if we just pressed options in the while code above, it will
automatically go here
    clrscr();
    Delay_ms(500);
    if(checkIfFolder(fAttrib)){
     Cout(1,"Home");
     Cout(2,"Open Folder");
     Cout(3,"Paste");
     FolOpInputAgain:
      choice = Cin(3);
     if(choice == 0){
      //cancel is pressed
      goto Current; //just back
     }else if(choice == 1){
       goto Start;
     }else if(choice == 2){          //for open folder
       UART1_Write_Text("CD ");
```

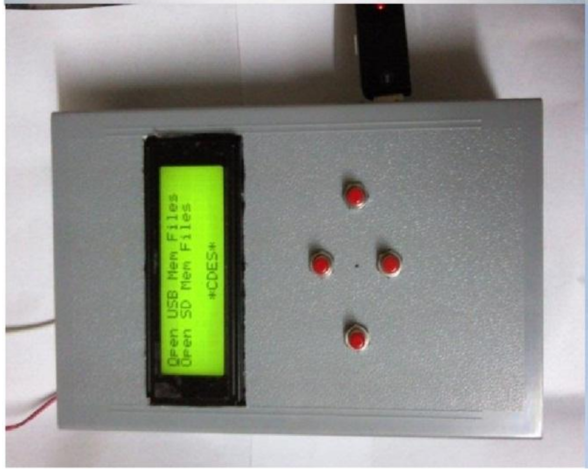
```
        WritePerCh(12,fName,1);
        UART1_Write(BS);      //ok kasipag folder dba '.' lngung extension xemprepag "CD
folder.", backspace kaparafoldernamelng
        UART1_Write(CR);
        gotoJustOpen;             //IL again
       }else if(choice == 3){ //PASTE nmannasa folder nakatapat...
         if(isCopy){
         Paste(); }else{
         clrscr();
          Cout(2,"Do copy first!");
          Delay_ms(500);
         }

         gotoJustOpen;
       }
     }else{
      Cout(1,"Home         | Delete");
      Cout(2,"Copy");
      Cout(3,"Paste");
      FileOpInputAgain:
       choice = Cin(4);
      if(choice == 0){
       //cancel is pressed, need to back to current
       goto Current;      //just back
      }else if(choice == 1){
       goto Start;
      }else if(choice == 2){      //this is if copy is chosen
        if(isCopy){
         Command("CF 0");
        }
        UART1_Write_Text("OF 0R>");
        WritePerCh(12,fName,1);
        UART1_Write(CR);
        strncpy(copyBufferName,fName,12);
        strncpy(copyBufferSize,fSize,8);
        isCopy = 1;
        //goto Current;
        gotoJustOpen;
      }else if(choice == 3){        //paste is chosen
```

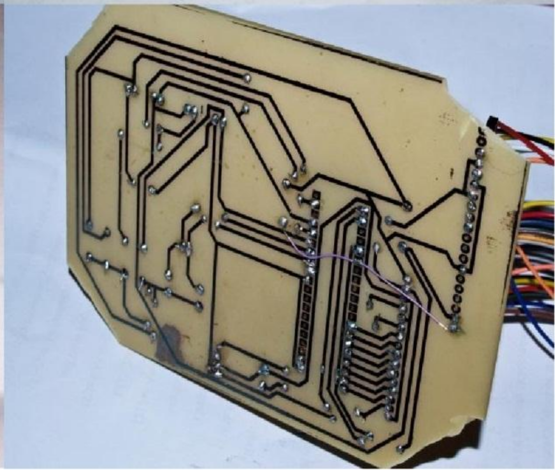
```
       if(isCopy){ Paste();
       }else{ clrscr();
         Cout(2,"Do copy first!");
         Delay_ms(500);
       }

       gotoJustOpen;

     }else if(choice == 4){      //delete is chosen
       UART1_Write_Text("DF ");
       WritePerCh(12,fName,1);
       UART1_Write(CR);
       clrscr();
       Cout(2,"Deleted");
       Delay_ms(500);

       gotoJustOpen; } //end of paste //goto Current;
    }
   } //end of PORTC.F0 if option is pressed
}
}//end of main
```

Appendix B

Pictures of Prototype

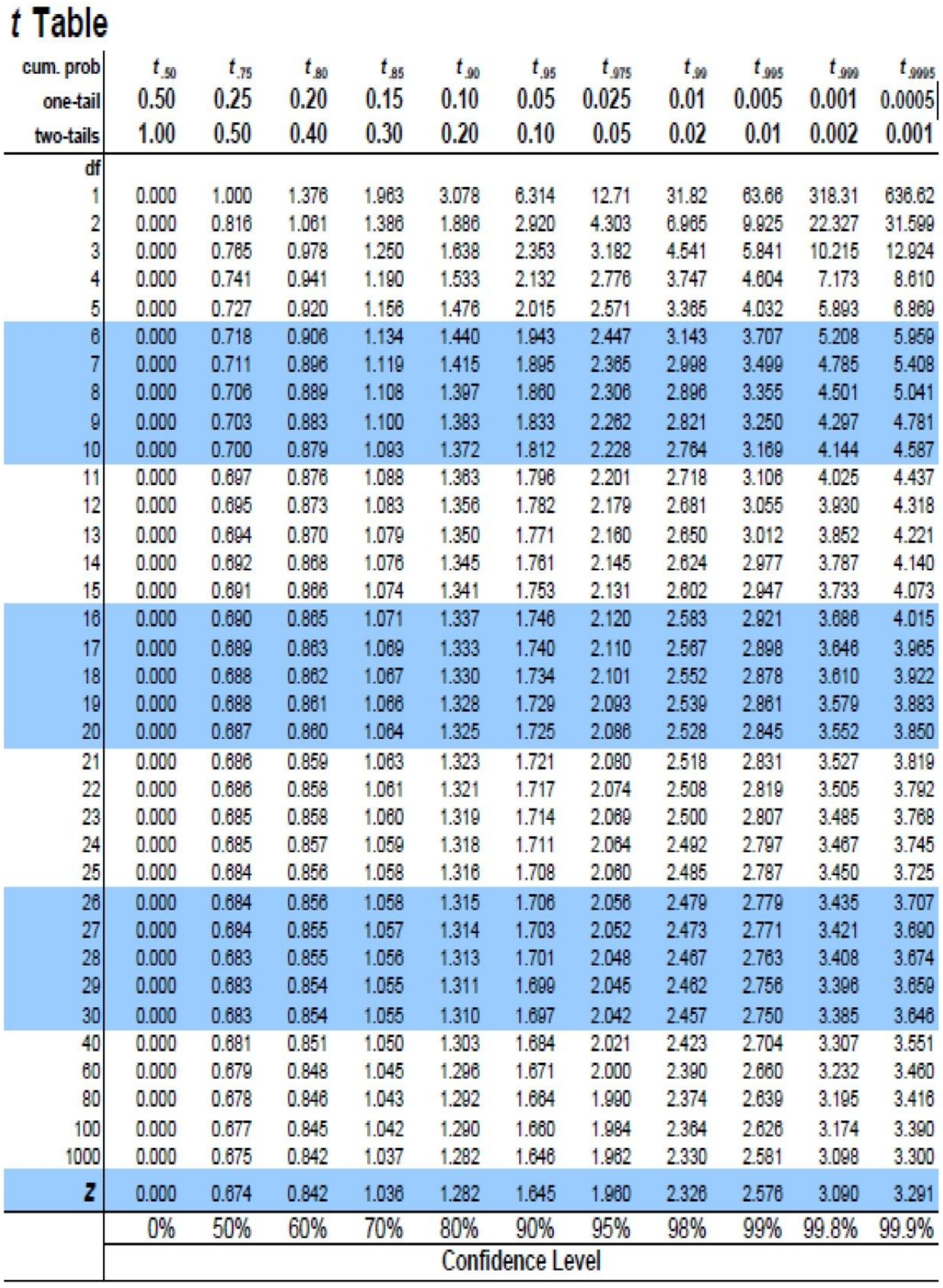









Appendix C

T-Table



t Table

cum. prob	$t_{.50}$	$t_{.75}$	$t_{.80}$	$t_{.85}$	$t_{.90}$	$t_{.95}$	$t_{.975}$	$t_{.99}$	$t_{.995}$	$t_{.999}$	$t_{.9995}$
one-tail	0.50	0.25	0.20	0.15	0.10	0.05	0.025	0.01	0.005	0.001	0.0005
two-tails	1.00	0.50	0.40	0.30	0.20	0.10	0.05	0.02	0.01	0.002	0.001
df											
1	0.000	1.000	1.376	1.963	3.078	6.314	12.71	31.82	63.66	318.31	636.62
2	0.000	0.816	1.061	1.386	1.886	2.920	4.303	6.965	9.925	22.327	31.599
3	0.000	0.765	0.978	1.250	1.638	2.353	3.182	4.541	5.841	10.215	12.924
4	0.000	0.741	0.941	1.190	1.533	2.132	2.776	3.747	4.604	7.173	8.610
5	0.000	0.727	0.920	1.156	1.476	2.015	2.571	3.365	4.032	5.893	6.869
6	0.000	0.718	0.906	1.134	1.440	1.943	2.447	3.143	3.707	5.208	5.959
7	0.000	0.711	0.896	1.119	1.415	1.895	2.365	2.998	3.499	4.785	5.408
8	0.000	0.706	0.889	1.108	1.397	1.860	2.306	2.896	3.355	4.501	5.041
9	0.000	0.703	0.883	1.100	1.383	1.833	2.262	2.821	3.250	4.297	4.781
10	0.000	0.700	0.879	1.093	1.372	1.812	2.228	2.764	3.169	4.144	4.587
11	0.000	0.697	0.876	1.088	1.363	1.796	2.201	2.718	3.106	4.025	4.437
12	0.000	0.695	0.873	1.083	1.356	1.782	2.179	2.681	3.055	3.930	4.318
13	0.000	0.694	0.870	1.079	1.350	1.771	2.160	2.650	3.012	3.852	4.221
14	0.000	0.692	0.868	1.076	1.345	1.761	2.145	2.624	2.977	3.787	4.140
15	0.000	0.691	0.866	1.074	1.341	1.753	2.131	2.602	2.947	3.733	4.073
16	0.000	0.690	0.865	1.071	1.337	1.746	2.120	2.583	2.921	3.686	4.015
17	0.000	0.689	0.863	1.069	1.333	1.740	2.110	2.567	2.898	3.646	3.965
18	0.000	0.688	0.862	1.067	1.330	1.734	2.101	2.552	2.878	3.610	3.922
19	0.000	0.688	0.861	1.066	1.328	1.729	2.093	2.539	2.861	3.579	3.883
20	0.000	0.687	0.860	1.064	1.325	1.725	2.086	2.528	2.845	3.552	3.850
21	0.000	0.686	0.859	1.063	1.323	1.721	2.080	2.518	2.831	3.527	3.819
22	0.000	0.686	0.858	1.061	1.321	1.717	2.074	2.508	2.819	3.505	3.792
23	0.000	0.685	0.858	1.060	1.319	1.714	2.069	2.500	2.807	3.485	3.768
24	0.000	0.685	0.857	1.059	1.318	1.711	2.064	2.492	2.797	3.467	3.745
25	0.000	0.684	0.856	1.058	1.316	1.708	2.060	2.485	2.787	3.450	3.725
26	0.000	0.684	0.856	1.058	1.315	1.706	2.056	2.479	2.779	3.435	3.707
27	0.000	0.684	0.855	1.057	1.314	1.703	2.052	2.473	2.771	3.421	3.690
28	0.000	0.683	0.855	1.056	1.313	1.701	2.048	2.467	2.763	3.408	3.674
29	0.000	0.683	0.854	1.055	1.311	1.699	2.045	2.462	2.756	3.396	3.659
30	0.000	0.683	0.854	1.055	1.310	1.697	2.042	2.457	2.750	3.385	3.646
40	0.000	0.681	0.851	1.050	1.303	1.684	2.021	2.423	2.704	3.307	3.551
60	0.000	0.679	0.848	1.045	1.296	1.671	2.000	2.390	2.660	3.232	3.460
80	0.000	0.678	0.846	1.043	1.292	1.664	1.990	2.374	2.639	3.195	3.416
100	0.000	0.677	0.845	1.042	1.290	1.660	1.984	2.364	2.626	3.174	3.390
1000	0.000	0.675	0.842	1.037	1.282	1.646	1.962	2.330	2.581	3.098	3.300
z	0.000	0.674	0.842	1.036	1.282	1.645	1.960	2.326	2.576	3.090	3.291
	0%	50%	60%	70%	80%	90%	95%	98%	99%	99.8%	99.9%
	Confidence Level										