

เครื่องเล่น MP3 จาก SD CARD
MP3 PLAYER FROM SD CARD



เลขหมู่.....
เลขทะเบียน 104154
วัน,เดือน,ปี 30 ต.ค. 2552

b.....
i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาอิเล็กทรอนิกส์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2551

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เครื่องเล่น MP3 จาก SD CARD

MP3 PLAYER FROM SD CARD

โดย

นายอนุพงศ์ วิบูลย์ศิริชัย 48011056

น.ส.ธันยรัตน์ สุภาโกมลนันท์ 48011064

น.ส.อุษา ตันตระกูล 48011130

อาจารย์ที่ปรึกษา

รศ.ดร.สมศักดิ์ ชุมช่วย

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2551

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2551

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องเล่น MP3 จาก SD CARD (MP3 PLAYER FROM SD CARD)

ผู้จัดทำ

- 1.นายอนุพงศ์ วิบูลย์ศิริชัย รหัสนักศึกษา 48011056
- 2.น.ส.ธันยนันท์ ศุภาโกมลนันท์ รหัสนักศึกษา 48011064
- 3.น.ส.อุษา ตันตระกูล รหัสนักศึกษา 48011130


..... อาจารย์ที่ปรึกษา
()

เครื่องเล่น MP3 แบบใช้ SD Card

นายอนุพงศ์ วิบูลย์ศิริชัย รหัส 48011056

นางสาวธันยนันท์ สุภาโกมลนันท์ รหัส 48011064

นางสาวอุษา ตันตระกูล รหัส 48011130

รศ.ดร.สมศักดิ์ ชุมช่วย

ปีการศึกษาที่ 2551

บทคัดย่อ

โครงการนี้เป็นการออกแบบและสร้าง เครื่องเล่น MP3 ซึ่งจะแบ่งออกเป็น 4 ส่วน คือ ส่วนSD Card, ส่วนMP3 Decoder, ส่วนLCD และ ส่วนไมโครคอนโทรลเลอร์ ซึ่งส่วนที่หนึ่ง คือ การเก็บข้อมูล เพลงจะถูกเก็บไว้ใน SD Card โดยมีการเก็บข้อมูลในรูปแบบ FAT 32 ส่วนที่สองคือ การถอดรหัส ข้อมูลโดยใช้ MP3 decoder เป็นการนำข้อมูลดิจิทัลมาถอดรหัสเป็นอนาล็อก แล้วส่งข้อมูลอนาล็อก ไปยังชุดหูฟัง ส่วนที่สาม คือ การแสดงผล เป็นการนำชื่อเพลงออกมาแสดงผ่านทางหน้าจอ LCD และส่วนสุดท้ายคือ การควบคุม ส่วนนี้เป็นส่วนที่ควบคุมการทำงานทั้งหมดโดยใช้ ไมโครคอนโทรลเลอร์ตระกูล PIC เครื่องเล่น MP 3 ส่วนนี้สามารถเล่นไฟล์ MP3 และ ไฟล์ WAV ได้ และยังสามารถควบคุม การเล่นเพลง หยุดเล่นเพลง เลือกเพลง และการปรับระดับเสียง

MP3 Player with SD Card Interface

Mr. Anupong Wiboonsirichai ID.48011056

Miss Thanyanan Supagomnonn ID.48011064

Miss Usa Tontrakoon ID.48011130

Ph.D. Prof. Somsak Choomchuay

Educational Year 2008

Abstract

This MP3 player with built-in SD card interface includes four parts which are a SD card reader, a MP3 decoder, a LCD display and a microcontroller. The audio data is stored in SD card using FAT32 format. A PIC microcontroller is served as a controller unit. It transfers the audio data from SD card to MP3 decoder. The decoder then drives analog audio directly to the head-set speaker. This Portable MP3 can play MP3 file and WAV file. It has user interface buttons for control basic function such as play, stop and volume control.

สารบัญ

เรื่อง	หน้า
บทที่ 1 บทนำ	1
บทที่ 2 ทฤษฎี	3
2.1 MP3	3
2.2 wave files กับ mp3	3
2.3 ข้อดีของการเก็บเพิ่มในรูปแบบ mp3	4
2.4 ข้อเสียของการเก็บเพิ่มในรูปแบบ mp3	4
2.5 ไมโครคอนโทรลเลอร์	5
2.5.1 โครงสร้างโดยทั่วไปของไมโครคอนโทรลเลอร์	5
2.5.2 PIC	7
2.5.3 PIC ชนิดต่าง	11
2.5.4 PIC เบอร์ต่างๆ	13
2.5.5 Types of Oscillator	15
2.6 flash memory	16
2.7 มาตรฐาน flash memory รูปแบบใหม่	17
2.8 SD card	18
2.8.1 รูปแบบพื้นฐาน	19
2.8.2 อัตราการรับส่งข้อมูล	19

สารบัญ (ต่อ)

เรื่อง	หน้า
2.8.3 อุปกรณ์ที่รองรับ	20
2.8.4 SD card ทำงานอย่างไร	20
2.9 ระบบไฟล์ flash 16 และ flash32 ทำงานอย่างไร	20
2.9.1 FAT	21
2.10 FAT 32	26
2.10.1 ข้อควรคำนึงเกี่ยวกับ FAT32	30
2.11 SPI mode	31
2.11.1 Command and response	31
2.11.2 SPI command set	32
2.11.3 SPI response	33
2.12 ขั้นตอนการเช็คค่าเริ่มต้นสำหรับ SPI mode	33
2.12.1 Power on	34
2.12.2 Software reset	34
2.12.3 Initialization	34
2.12.4 การโอนย้ายข้อมูล	34
2.12.5 การอ่านแบบ single block	35
2.12.6 การอ่านแบบ multiple block	35

สารบัญ (ต่อ)

เรื่อง	หน้า
2.13 การอ่านค่า รีจิสเตอร์ CSD และ CID	36
บทที่ 3 การออกแบบ	37
- ส่วนด้าน Hardware	37
3.1 SD card	38
3.2 Microcontroller PIC 18F8722	39
3.2.1 การติดต่อระหว่าง Microcontroller, SD card และ	40
ตัวถอดรหัส MP3	
3.3 MP3 decoder VS1002D	41
3.4 LCD	43
3.5 วงจรภาคจ่ายไฟ 5 V. และ 3.3 V.	44
- ส่วนด้าน Software	47
3.6 Flow chart	47
3.6.1 ภาค SD Card	47
3.6.2 ภาค LCD	48
3.6.3 ภาค SWITCH ควมคุมทั้ง 6 ขา	49
บทที่ 4 ผลการทดลอง	50
4.1 ส่วนผลการทดลองของ Software	50

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

เรื่อง	หน้า
4.1.1 การทดลองเพื่อทดสอบว่า microcontroller สามารถ ติดต่อกับ SD Card ได้หรือไม่	50
4.1.2 การทดลองเพื่อทดสอบว่า SW ควบคุมสามารถ ใช้งานได้หรือไม่	51
4.2 ส่วนผลการทดลองของ Hardware	53
บทที่ 5 สรุปผลการทดลอง	55
5.1 สรุปและวิจารณ์	55
5.2 ประโยชน์ที่ได้รับ	55
ภาคผนวก	
กิตติกรรมประกาศ	
เอกสารอ้างอิง	

สารบัญรูป

เรื่อง	หน้า
รูปที่ 2.1 โครงสร้างภายในไมโครคอนโทรลเลอร์	7
รูปที่ 2.2 รูปแบบการเขียนโปรแกรมด้วยภาษาAssembly แบบไฟล์เดี่ยว	8
รูปที่ 2.3รูปแบบการเขียนโปรแกรมด้วยภาษาAssembly แบบหลายไฟล์	9
รูปที่ 2.4รูปแบบการเขียนโปรแกรมด้วยภาษาขั้นสูง	9
รูปที่ 2.5 บล็อกแสดงการเบิร์นเพื่อนำไมโครคอนโทรลเลอร์ไปใช้	10
รูปที่ 2.6ตัวอย่างไมโครคอนโทรลเลอร์แบบ OTP	11
รูปที่ 2.7ตัวอย่างไมโครคอนโทรลเลอร์แบบ EPROM	12
รูปที่ 2.8ตัวอย่างไมโครคอนโทรลเลอร์แบบ EEPROM	12
รูปที่ 2.9ลักษณะ oscillatorแบบต่างๆ	16
รูปที่ 2.10 SD card	18
รูปที่ 2.11 รูปแบบของฮาร์ดดิสก์ออกเป็นร่อง	21
รูปที่ 2.12 การจับเก็บข้อมูลของคัลสเตอร์	24
รูปที่ 2.13 Dialog scan hard disk	24
รูปที่ 2.14 กราฟเปรียบเทียบ Highend disk winmark ของ FAT16 และ FAT32	29
รูปที่ 2.15กราฟเปรียบเทียบ disk performance / bus ของ FAT16 และ FAT32	29
รูปที่ 2.16กราฟเปรียบเทียบ disk performance / bus ของ FAT16 และ FAT32	30

สารบัญรูป (ต่อ)

เรื่อง	หน้า
รูปที่ 2.17 แสดงไพบ์ของ R1 responses และ R3 response	33
รูปที่ 2.18 แสดงการอ่านแบบ single block	35
รูปที่ 2.19 แสดงการอ่านแบบ multiple block	35
รูปที่ 3.1 Block Diagram แสดงการทำงานของเครื่องเล่น MP3	37
รูปที่ 3.2 โครงสร้าง SD card	38
รูปที่ 3.3 การเชื่อมต่อระหว่าง microcontroller กับ SD Card	38
รูปที่ 3.4 การเชื่อมต่อระหว่าง microcontroller กับ SW ควบคุมการเล่นเพลง	40
รูปที่ 3.5 การติดต่อของ Microcontroller กับ SD card และ VS1002D โดยใช้ SPI 2 ports	40
รูปที่ 3.6 วงจร MP3 decoder โดย VS1002D	41
รูปที่ 3.7 โครงสร้าง VS1002D	42
รูปที่ 3.8 การเชื่อมต่อระหว่าง microcontroller กับ MP3 Decoder VS 1002D	42
รูปที่ 3.9 การเชื่อมต่อระหว่าง microcontroller กับ LCD	44
รูปที่ 3.10 แสดงวงจรภาคจ่ายไฟ	44
รูปที่ 3.11 ภาพ Model เครื่องเล่น MP3 แบบ SD Card	46
รูปที่ 3.12 Flow Chart การทำงานของภาค SD Card	47
รูปที่ 3.13 Flow Chart การทำงานของภาค LCD	48

สารบัญรูป (ต่อ)

เรื่อง	หน้า
รูปที่ 3.14 Flow Chart การทำงานของภาค SWITCH ควบคุมทั้ง 6 ขา	49
รูปที่ 4.1 แสดงผลการเบิร์นโค้ดลงไมโครคอนโทรลเลอร์ ว่าโปรแกรมสามารถ ทำให้ไมโครคอนโทรลเลอร์เชื่อมต่อกับ SD Card ได้	50
รูปที่ 4.2 แสดงผลการเบิร์นโค้ดลงไมโครคอนโทรลเลอร์ ว่าสามารถเลื่อนเพลง เป็นเพลงถัดไปและวนรูปเพลงได้	51
รูปที่ 4.3 แสดงผลการเบิร์นโค้ดลงไมโครคอนโทรลเลอร์ ว่าสามารถเลื่อนเพลง เป็นเพลงถัดไปและเพลงก่อนหน้าได้	52
รูปที่ 4.4 ภาพการแสดงชื่อเพลงบนจอ LCD ของค่า current file ที่ 1	53
รูปที่ 4.5 ภาพการแสดงชื่อเพลงบนจอ LCD ของค่า current file ที่ 2	53
รูปที่ 4.6 ภาพการแสดงชื่อเพลงบนจอ LCD ของค่า current file ที่ 3	54

สารบัญตาราง

เรื่อง	หน้า
ตารางที่ 2.1 ตารางเปรียบเทียบความจุของฮาร์ดดิส FAT 16 กับ FAT32	23
ตารางที่ 2.2 ตารางเปรียบเทียบขนาดของคัสเตอร์เมื่อกำหนดขนาด ของพาร์ติชันต่างๆใน FAT16	27
ตารางที่ 2.3 ตารางเปรียบเทียบขนาดของคัสเตอร์เมื่อกำหนดขนาด ของพาร์ติชันต่างๆใน FAT32	28
ตารางที่ 2.4 ตารางแสดง command ที่ใช้ทั่วไปในการอ่านเขียน และกำหนด ค่าเริ่มต้นของการ์ด	32

บทที่ 1

บทนำ

1.1 ความเป็นมาของโครงการ

ปัจจุบันจะสังเกตได้ว่าเครื่องเล่นทั่ว ๆ ไป จะมีความสามารถในการเล่นไฟล์ข้อมูลได้หลายประเภท เช่น DVD, VCD, CD และที่ขาดไม่ได้คือ MP3 ซึ่งเป็นอีกหนึ่งมาตรฐานในการบีบอัดไฟล์เสียง และกำลังได้รับความนิยมอยู่ในขณะนี้

เนื่องจากผู้จัดทำได้มีโอกาสอ่านบทความทางอิเล็กทรอนิกส์ ตลอดจนข้อมูลข่าวสารทางอินเทอร์เน็ต ทำให้ทราบว่าปัจจุบันมีไอซีที่สามารถถอดรหัสไฟล์ MP3 ได้ และมีอยู่หลายบริษัทที่ผลิตออกมาให้ทดลองใช้กัน เช่น STA013 ของ ST Technology, AT89C51SDN2 ของ ATMEL, MICRONAS MAS3570 และ VS10xx ของ VLSI เป็นต้น สำหรับการใช้งานก็มีความยากง่ายต่างกัน เช่น การต่อวงจรรวมภายนอกเทคนิคในการติดต่อสื่อสาร สำหรับผู้จัดทำเลือกใช้ VS1002D เหตุผลคือ เป็นไอซีที่ใช้อุปกรณ์ภายนอกน้อยชิ้น การติดต่อเพื่อควบคุมการทำงานจะง่ายกว่า และมีราคาถูกกว่าไอซีเบอร์อื่น ๆ ที่ได้กล่าวมาข้างต้น

เทคโนโลยีเกี่ยวกับการจัดหน่วยความจำ (Memory Card) ก็ได้มีการพัฒนาออกสู่ตลาดมากมายหลายรูปแบบ เช่น Memory Stick, Smart Media, Compact Flash, SD/MMc Card เป็นต้น ซึ่งการใช้งานก็มีหลายลักษณะเช่นกัน ในส่วนของผู้จัดทำได้เลือกใช้ SD Card เนื่องจากเห็นว่ามีความเหมาะสมในเรื่องของราคาที่ไม่ค่อยสูงมากและเป็นการดีที่สามารถหาซื้อได้ง่าย

สุดท้ายในส่วนของไมโครคอนโทรลเลอร์ เนื่องจากผู้จัดทำต้องการไมโครคอนโทรลเลอร์ที่ต้องการกำลังไฟฟ้าค่อนข้างต่ำ มีการประมวลผลที่เร็ว ผู้จัดทำจึงเลือกใช้ไมโครคอนโทรลเลอร์ตระกูล PIC (PIC18F8722) ในการควบคุมการทำงานทั้งหมด และเลือกใช้ภาษาซีในการเขียนโค้ดโปรแกรม

1.2 วัตถุประสงค์

เพื่อศึกษาการเขียน/อ่านข้อมูลการ์ดหน่วยความจำแบบ SD Card ศึกษาการบีบอัดไฟล์แบบ MP3, ศึกษาการติดต่อสื่อสารข้อมูลในลักษณะ SPI BUS, เรียนรู้และฝึกใช้งานแอลซีดี ตลอดจนเพิ่มทักษะในการใช้งานไมโครคอนโทรลเลอร์ตระกูล PIC

1.3 ขอบเขตของโครงการ

โครงการเครื่องเล่น MP3 ชนิดพกพา มี SD Card นี้ ได้ออกแบบและสร้างวงจรในการเขียน/อ่านการ์ดหน่วยความจำแบบ SD Card, การถอดรหัสไฟล์ MP3 โดยใช้ไอซี VS1002D ซึ่งการติดต่อกับอุปกรณ์ทั้งสองจะเป็นการติดต่อในลักษณะ SPI BUS, มีการแสดงผลโดยใช้จอแอลซีดี และใช้ไมโครคอนโทรลเลอร์ (PIC18F4539) ในการควบคุม

1.4 โครงสร้างของรายงาน

รายงานฉบับนี้ได้อธิบายขั้นตอน วิธีในการออกแบบ รวมทั้งวงจร และผลการทดลองคุณสมบัติต่าง ๆ ของเครื่องเล่น MP3 ชนิดพกพา มี SD Card โดยมีเนื้อหาแบ่งเป็นบทต่าง ๆ ดังนี้

บทที่ 2 ทฤษฎี จะกล่าวถึงทฤษฎี และหลักการพื้นฐานต่าง ๆ ที่เกี่ยวข้องกับการออกแบบและสร้างเครื่องเล่น MP3 ชนิดพกพา มี SD Card

บทที่ 3 การออกแบบ จะกล่าวถึงขั้นตอนในการออกแบบและโค้ดโปรแกรม ในการติดต่อกับการ์ดหน่วยความจำแบบ SD Card, การติดต่อกับไอซีถอดรหัสไฟล์ MP3, และการติดต่อกับแอลซีดีแสดงผล โดยใช้ไมโครคอนโทรลเลอร์ PIC

บทที่ 4 การทดลอง และผลการทดลอง จะกล่าวถึงการทดลอง และผลการทดลอง เมื่อทำการทดสอบสั่งงานเครื่องเล่น MP3 ชนิดพกพา มี SD Card นี้ ด้วยอินพุตต่าง ๆ

บทที่ 5 สรุป และวิจารณ์

บทที่ 2

ทฤษฎี

2.1 MP3

MP3 มาจากคำว่า MPEG1- Audio Layer 3 สำหรับคำว่า MPEG นั้นเป็นคำที่ย่อมาเพื่อใช้เรียกเป็นมาตรฐาน โดยเรียกย่อมาจากตามองค์กรซึ่งเป็นผู้ที่ได้ริเริ่มมาตรฐานการบีบอัดภาพเคลื่อนไหวและเสียง ซึ่งชื่อเต็มขององค์กรนี้ก็คือ Motion Picture Experts Groups (อักษรตัวใหญ่รวมได้เป็นคำว่า MPEG นั้นเอง) โดยเริ่มจัดตั้งมาเมื่อช่วงประมาณปลายปี 1997

MP3 คือฟอร์แมตของไฟล์ชนิดหนึ่งซึ่งบรรจุข้อมูลออดิโอ (ข้อมูลเสียง) ที่ใช้เล่นในคอมพิวเตอร์ ไฟล์ชนิดนี้ใช้เทคโนโลยีบีบอัดข้อมูลให้มีขนาดเล็กลงมาก ๆ โดยแปลงจากไฟล์เพลงนามสกุล WAV ในอัตราส่วน 1:12 ขอยกตัวอย่างขนาดของไฟล์ MP3 เพลงหนึ่งความยาว 3 นาที หากเป็นไฟล์ WAV จะมีขนาด 40 MB (เท่ากับแผ่นฟลอปปี้ดิสก์ จำนวน 27 - 28 แผ่น จึงจะบรรจุได้ 1 เพลง) แต่ถ้าแปลงเป็นไฟล์ MP3 ขนาดจะลดลงเหลือ 3 MB (เท่ากับความจุฟลอปปี้ดิสก์ 2 แผ่นกว่า) ขนาดเล็กลงถึง 12 เท่า MP3 เป็นเทคโนโลยีบีบอัดข้อมูลที่พัฒนามาจาก MPEG รุ่นก่อน

วัตถุประสงค์ของการจัดตั้งมาตรฐานของ MPEG ก็เพื่อกำหนดมาตรฐานของการบีบอัด (Compress) ไฟล์ข้อมูลเกี่ยวกับภาพเคลื่อนไหวและเสียง เพื่อให้การบีบอัดนั้นนำไปสู่ความสามารถบีบอัดให้ได้ขนาดไฟล์ที่เล็กลงโดยยังคงคุณภาพ (Quality) ของไฟล์ต้นฉบับ (Raw) ไว้ให้ได้มากที่สุด เช่น การตัดช่วงของเสียงต่างๆที่เป็นข้อมูลที่ไม่จำเป็นออกจากไฟล์ต้นฉบับเพื่อลดขนาดของไฟล์ที่จะบันทึกให้เล็กลง เป็นต้น

2.2 Wave Files กับ MP3

แต่เดิมการบันทึกเสียงลงในเครื่องคอมพิวเตอร์นั้นจะทำโดยการบันทึกเป็นรูปแบบ (Format) ที่เราเรียกว่า เวฟไฟล์ (Wave Files) ซึ่งมีส่วนขยาย (หรือบางคนเรียกชื่อนามสกุล) เป็น .wav ซึ่งเป็นการบันทึกที่ได้คุณภาพเสียงสูงที่สุด โดยไม่มีการลดทอนคุณภาพของเสียงต้นฉบับ ซึ่งจะแตกต่างจากเสียงต้นฉบับหรือไม่ นั่นขึ้นอยู่กับองค์ประกอบภายนอกอื่น ๆ ด้วย เช่น คุณภาพของอุปกรณ์และเครื่องมือที่

ใช้ในการบันทึก (ยกตัวอย่าง : คุณภาพของสาย, ไมโครโฟน หรือ เสียงและสัญญาณรบกวนต่าง เป็นต้น)เรียกได้ว่ามาเท่าไรบันทึกเท่านั้นทำให้ในการบันทึกไฟล์เป็นเวฟไฟล์นั้น เราจะได้ขนาดไฟล์ประมาณ 10 MBytes ต่อเวลาที่ใช้นับที่เสียง 1 นาที หมายความว่าถ้าเราบันทึกเสียงหรือเพลงทั่วไป ที่มี ความยาวเฉลี่ยประมาณ 3-5 นาทีต่อเพลงแล้วละก็ เราจะต้องใช้พื้นที่ในการบันทึกประมาณ 30-50 MBytes ต่อเพลงเลยทีเดียวหลังจากได้มีการนำเอารูปแบบการบันทึกเสียงในแบบ MP3 เข้ามาใช้ กัน ทำให้อายุที่เราจะต้องสูญเสียพื้นที่ในการจัดเก็บไฟล์เสียงนั้นน้อยลงกว่าเดิมราวๆ 10 เท่า ซึ่งก็คือ ไฟล์เสียง 1 นาที สามารถบันทึกได้ด้วยพื้นที่เพียงราว 1 MBytes เท่านั้นดังนั้นขนาดของเพลงทั่วไปที่ยาว ราว 3-5 นาที ก็จะใช้พื้นที่จัดเก็บเพียง 3-5 MBytes เท่านั้น โดยที่คุณภาพของเสียงที่เราได้ยินเมื่อเทียบกับ เสียงที่เราได้ฟังจากแผ่น CD นั้น ลดทอนลงไปเพียง 10% จากคุณภาพเสียงของ CD เพลง คิดจากอัตรา การบีบอัด(bitrate) 125 Kbps ที่ความถี่ (frequency) 44 KHz จึงทำให้ได้รับความนิยมอย่างมาก ในเวลา อันรวดเร็ว แต่ก็นิยมสำหรับในหมู่นักฟังเพลงทั่วไปเท่านั้น แต่สำหรับตัวผู้ผลิตนั้นในขั้นตอนการ บันทึกเพื่อการผลิตให้คงคุณภาพเสียงสูงที่สุด ก็ยังคงจะนิยมบันทึกเสียงให้คงอยู่ในรูปแบบ "wave file" ต่อไป

2.3 ข้อดีของการเก็บแฟ้มในรูปแบบ MP3

เนื่องจากแฟ้ม MP3 เป็นแฟ้มที่มีขนาดเล็ก ดังนั้นจึงใช้พื้นที่ในการเก็บน้อยโปรแกรมที่ใช้เล่น แฟ้ม MP3 ก็เช่น โปรแกรม Windows Media Player และโปรแกรมนี้ก็จะมามีมากับเครื่องคอมพิวเตอร์อยู่แล้ว ไม่ต้องไปหา Download ใหม่ (แต่ต้องเป็น windows 98 ขึ้นไป) รวมไปถึง Winamp ซึ่งหา download ได้ ทั่วไป

2.4 ข้อเสียของการเก็บแฟ้มในรูปแบบ MP3

ได้มีการลักลอบก๊อปปี้เพลงจาก CD มาแจกจ่ายทั่วเน็ต โดยไม่ได้รับอนุญาต จากเจ้าของ ลิขสิทธิ์ และเมื่อคนโหลดเพลง MP3 ไปฟังแล้ว ก็อาจจะไม่ซื้ออัลบั้มอีก ถือได้ว่า ส่งผลเสียต่อ ผลประโยชน์ ของค่ายเพลง และตัวศิลปินเอง

2.5 ไมโครคอนโทรลเลอร์ (Microcontroller)

ไมโครคอนโทรลเลอร์ (Microcontroller) มาจากคำ 2 คำ คำหนึ่งคือ ไมโคร (Micro) หมายถึงขนาดเล็ก และคำว่า คอนโทรลเลอร์ (controller) หมายถึงตัวควบคุมหรืออุปกรณ์ควบคุม ดังนั้น ไมโครคอนโทรลเลอร์ จึงหมายถึงอุปกรณ์ควบคุมขนาดเล็ก แต่ในตัวอุปกรณ์ควบคุมขนาดเล็กนี้ ได้บรรจุความสามารถที่คล้ายคลึงกับระบบคอมพิวเตอร์ ที่คนโดยส่วนใหญ่คุ้นเคย กล่าวคือภายใน ไมโครคอนโทรลเลอร์ ได้รวมเอาซีพียู, หน่วยความจำ และพอร์ต ซึ่งเป็นส่วนประกอบหลักสำคัญของระบบคอมพิวเตอร์เข้าไว้ด้วยกัน โดยทำการบรรจุเข้าไว้ในตัวเดียวกัน

2.5.1 โครงสร้างโดยทั่วไปของไมโครคอนโทรลเลอร์

โครงสร้างโดยทั่วไป ของไมโครคอนโทรลเลอร์นั้น สามารถแบ่งออกมาได้เป็น 5 ส่วนใหญ่ๆ ดังต่อไปนี้

1. หน่วยประมวลผลกลางหรือซีพียู (CPU : Central Processing Unit)
2. หน่วยความจำ (Memory) สามารถแบ่งออกเป็น 2 ส่วน คือ หน่วยความจำที่มีไว้สำหรับเก็บโปรแกรมหลัก (Program Memory) เปรียบเสมือนฮาร์ดดิสก์ของเครื่องคอมพิวเตอร์ตั้งโต๊ะ คือข้อมูลใดๆ ที่ถูกเก็บไว้ในนี้จะไม่สูญหายไปแม้ไม่มีไฟเลี้ยง อีกส่วนหนึ่งคือหน่วยความจำข้อมูล (Data Memory) ใช้เป็นเหมือนกระดานทดในการคำนวณของซีพียู และเป็นที่พักข้อมูลชั่วคราวขณะทำงาน แต่หากไม่มีไฟเลี้ยง ข้อมูลก็จะหายไปคล้ายกับหน่วยความจำแบบ (RAM) ในเครื่องคอมพิวเตอร์ทั่วไป
แต่สำหรับไมโครคอนโทรลเลอร์สมัยใหม่ หน่วยความจำข้อมูลจะมีทั้งที่เป็นหน่วยความจำแบบ ซึ่งข้อมูลจะหายไปเมื่อไม่มีไฟเลี้ยง และเป็นอีอีพรอม (EEPROM : Erasable Electrically Read-Only Memory) ซึ่งสามารถเก็บข้อมูลได้แม้ไม่มีไฟเลี้ยง
3. ส่วนติดต่อกับอุปกรณ์ภายนอก หรือพอร์ต (Port) มี 2 ลักษณะคือ พอร์ตอินพุต (Input Port) และพอร์ตส่งสัญญาณหรือพอร์ตเอาต์พุต (Output Port) ส่วนนี้จะใช้ในการเชื่อมต่อกับอุปกรณ์ภายนอก ถือว่าเป็นส่วนที่สำคัญมาก ใช้ร่วมกันระหว่างพอร์ตอินพุต เพื่อรับสัญญาณ อาจจะด้วยการกดสวิตช์ เพื่อนำไปประมวลผลและส่งไปพอร์ตเอาต์พุต เพื่อแสดงผลเช่น การติดสว่างของหลอดไฟ เป็นต้น

4. ช่องทางเดินของสัญญาณ หรือบัส (BUS) คือเส้นทางการแลกเปลี่ยนสัญญาณข้อมูลระหว่าง ซีพียู หน่วยความจำและพอร์ต เป็นลักษณะของสายสัญญาณ จำนวนมากอยู่ภายในตัวไมโครคอนโทรลเลอร์ โดยแบ่งเป็นบัสข้อมูล (Data Bus) , บัสแอดเดรส (Address Bus) และบัสควบคุม Control Bus)

บัสข้อมูลเป็นสายสัญญาณที่บรรจุข้อมูล เพื่อการประมวลผลทั้งหมด ขนาดของบัสจะขึ้นอยู่กับความสามารถการประมวลผลของซีพียู สำหรับในงานทั่วไป ขนาดของบัสข้อมูลจะเป็น 8 บิต และในปัจจุบันได้มีการพัฒนาขึ้นมาจนถึง 16,32 และ 64 บิต

บัสแอดเดรสเป็นสายสัญญาณที่บรรจุค่าตำแหน่งของหน่วยความจำ โดยการติดต่อกับหน่วยความจำนั้น ซีพียู ต้องกำหนดตำแหน่งที่ต้องการอ่านหรือเขียนก่อน ดังนั้นจำนวนสายสัญญาณของแอดเดรสจึงต้องมีจำนวนมาก ยิ่งมากเท่าไร ก็จะเป็นการแสดง ขนาดของหน่วยความจำที่ไมโครคอนโทรลเลอร์สามารถติดต่อได้ โดยสามารถคำนวณได้จาก

จำนวนแอดเดรสของหน่วยความจำ = 2 ยกกำลัง n (n คือจำนวนของเส้นทาง)

ยกตัวอย่าง ไมโครคอนโทรลเลอร์ตัวหนึ่งมีสายแอดเดรส 10 เส้น ดังนั้น

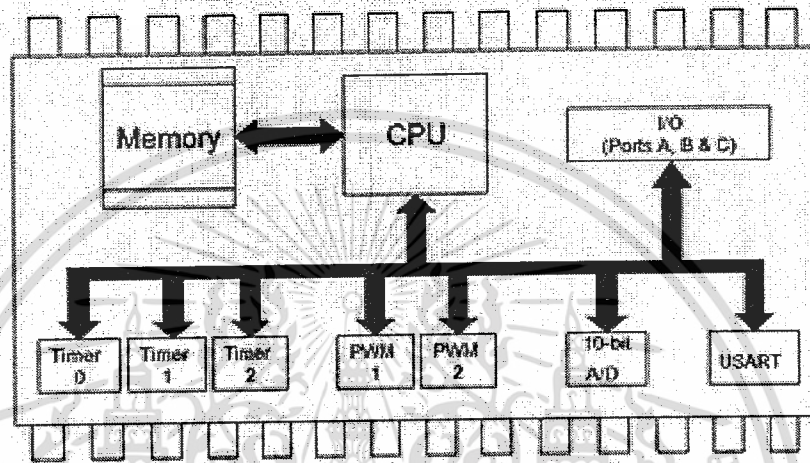
ไมโครคอนโทรลเลอร์ตัวนี้ สามารถติดต่อกับหน่วยความจำได้ 2 ยกกำลัง $10 = 1,024$ ตำแหน่ง

หากต้องการทราบความจุของหน่วยความจำจริงๆ จะต้องทราบถึงขนาดของบัสข้อมูลก่อนว่าเป็นเท่าใด หากเป็น 8 บิต ความจุของหน่วยความจำที่มีสายแอดเดรส 10 เส้น จะเท่ากับ $8 \times 1024 = 8,192$ บิต และ 1 กิโลไบต์ เท่ากับ 1,024 ไบต์ ดังนั้น ไมโครคอนโทรลเลอร์ดังกล่าว จึงมีความจุของหน่วยความจำเท่ากับ 8,192 บิต หรือ 1,024 ไบต์ หรือ 1 กิโลไบต์

บัสควบคุมเป็นกลุ่มของสายสัญญาณควบคุมการติดต่อทั้งหมดของซีพียูกับหน่วยความจำและพอร์ต สำหรับสายสัญญาณเลือกควบคุมหลักได้แก่ สายสัญญาณเลือก-อ่าน-เขียนหน่วยความจำ สายสัญญาณเลือกเลือก อ่าน-เขียน ข้อมูล กับพอร์ต

5. วงจร กำเนิดสัญญาณนาฬิกา นับเป็นส่วนประกอบที่สำคัญมากอีกส่วนหนึ่ง เนื่องจากการทำงานที่เกิดขึ้นในตัวไมโครคอนโทรลเลอร์ จะขึ้นอยู่กับกำหนัดจังหวะ หาก

สัญญาณนาฬิกาที่มีความถี่สูง จึงหะการทำงานก็จะสามารถทำได้ขึ้นส่งผลให้ไมโครคอนโทรลเลอร์ตัวนั้น มีความเร็วในการประมวลผลสูงตามไปด้วย



รูปที่ 2.1 โครงสร้างภายในของไมโครคอนโทรลเลอร์

2.5.2 PIC

PIC คือ microcontroller อีกตระกูลหนึ่ง ย่อมาจากคำว่า Peripheral Interface Controller ซึ่ง concept ของเจ้า microcontroller ตระกูลนี้ก็คือ พยายามรวมเอาทุกอย่างเอาไว้ในตัวของมันไม่ว่าจะเป็น PROGRAM MEMORY, RAM, EEPROM, SERIAL, I2C, PWM, A/D ฯลฯ โดยไม่จำเป็นต้องต่ออุปกรณ์เสริมจากภายนอก ในตัวของ PIC จะมีฟังก์ชันที่ใช้ในการประมวลผล รวมทั้งหน่วยความจำ ซึ่งทำให้มันเหมือนกับ CPU ตัวหนึ่งเลยทีเดียว

ตอนที่ 1 การเขียนโปรแกรม Microcontroller

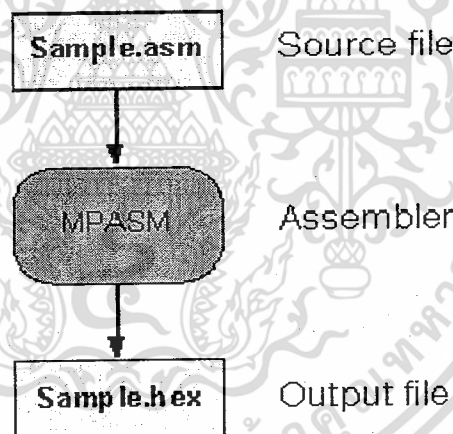
ภาษาที่ใช้สำหรับการเขียนโปรแกรมบน Microcontroller

ภาษาที่ใช้สำหรับการเขียนโปรแกรมบน Microcontroller (ต่อไปจะขอเรียกสั้นๆว่า MCU) แบ่งได้เช่นเดียวกับการเขียนโปรแกรมบนคอมพิวเตอร์คือ ภาษาระดับสูง และภาษาระดับต่ำ

- ภาษาระดับสูงเช่น C, Basic ข้อดีคือเขียนง่าย, แก้ไขเปลี่ยนแปลง หรือเพิ่มเติมได้ง่าย ส่วนข้อเสียก็คือการทำงานจะช้า ขนาดโปรแกรมที่เขียนมีขนาดใหญ่
- ภาษาระดับต่ำ ซึ่งก็คือ ภาษา Assembly ข้อดีคือ ตัว compiler แจกฟรี ขนาดโปรแกรมหลังจาก compiled แล้วมีขนาดเล็ก โปรแกรมมีความเร็ว แต่ข้อเสียก็คือเขียนยาก เพราะลักษณะภาษาไม่ค่อยสื่อความหมาย แก้ไขเปลี่ยนแปลงยาก

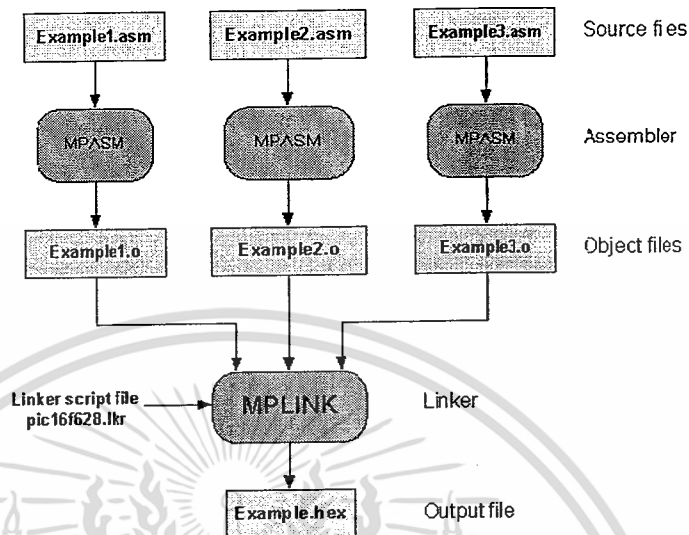
รูปแบบการเขียนโปรแกรม Microcontroller สามารถแบ่งได้ดังนี้

1. เขียนด้วยภาษา Assembly แบบ ไฟล์เดียว หลังจากนั้นก็ทำการ Compile ด้วย Assembler ของ MCU ตัวนั้น ซึ่งส่วนในผู้ผลิต Chip MCU จะแจกจ่ายให้ฟรี สำหรับ Assembler ของ Microchip ก็คือ MPASM โดยไฟล์ที่ได้มามีได้หลายชนิดแต่ส่วนใหญ่จะอยู่ในรูปของ Hex file



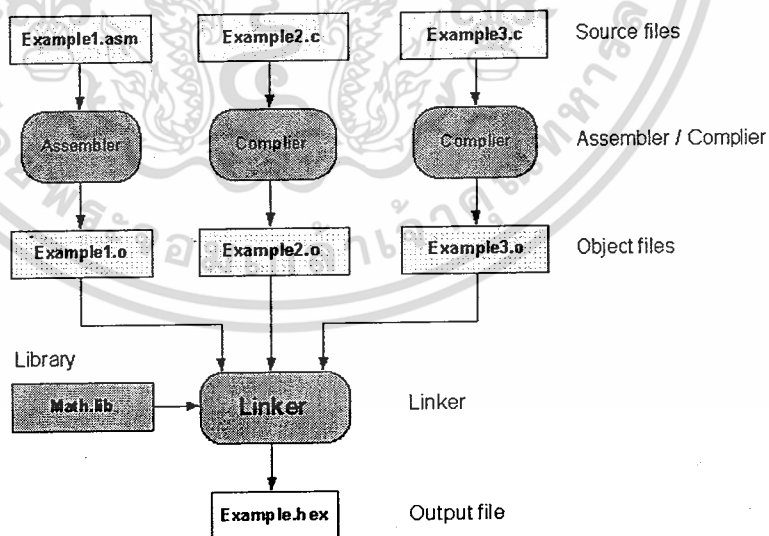
รูปที่ 2.2 รูปแบบการเขียนโปรแกรมด้วยภาษา Assembly แบบไฟล์เดียว

2. ใช้ภาษา Assembly เช่นกัน แต่แบ่งเป็นหลายๆ ไฟล์ หลังจากนั้นก็จะ Compile แต่ละไฟล์ให้ออกมาเป็น Object files และทำการรวมกันด้วย Linker ในขณะที่ทำการ link ก็จะมี script file ของ MCU เบอร์นั้นๆ ประกอบ หลังจากทำการ Link แล้วก็จะได้ Hex file ออกมา



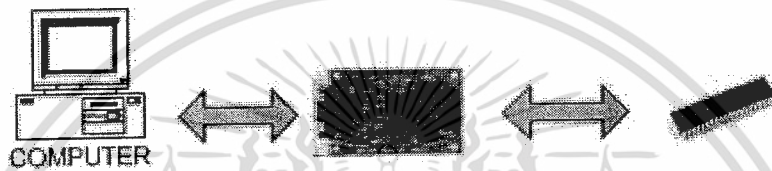
รูปที่ 2.3 รูปแบบการเขียนโปรแกรมด้วยภาษา Assembly แบบหลายไฟล์

3. การเขียนด้วยภาษาสูง อาจจะเป็น C, Basic ฯลฯ ซึ่งอาจจะเขียนร่วมกับ ภาษา assembly โดยไฟล์ที่เขียนจะถูกทำให้กลายเป็น Object files โดย Assembler สำหรับภาษา Assembly และ Compiled โดย Compiler สำหรับภาษาสูง ทำการ Link เข้าด้วยกันด้วย Linker ซึ่งขณะทำการ Link ก็จะมีการรวมเอา Library ที่ถูกเรียกใช้ในโปรแกรมเข้าไปรวมด้วยกัน สุดท้ายก็จะออกมาเป็น Hex file



รูปที่ 2.4 รูปแบบการเขียนโปรแกรมด้วยภาษาชั้นสูง

หลังจากได้ Hex file มาแล้ว เราก็จะทำการอัด โปรแกรมเข้าสู่ chip ด้วยตัวโปรแกรมเมอร์ ส่วนใหญ่จะมีรูปแบบคือ มี Software บนคอมพิวเตอร์ สำหรับใช้ในการควบคุมการอ่าน เขียน หรือ ลบ โดยส่วนใหญ่จะเชื่อมต่อไปยัง programmer ด้วย serial, parallel มีราคาให้เลือกตั้งแต่หลักร้อยไปจนถึงหลักหมื่น เมื่ออัดโปรแกรมเข้า chip ได้แล้วเราก็พร้อมจะนำไปทดสอบการทำงานต่อไป



รูปที่ 2.5 บล็อกแสดงการเบิร์นเพื่อนำไมโครคอนโทรลเลอร์ไปใช้

ตอนที่ 2 ภาษาที่ใช้สำหรับเขียนโปรแกรม PIC

MPASM

เป็น Assembler ที่ทางบริษัท MICROCHIP แจกจ่ายให้ใช้กันได้ฟรีๆ เป็นที่นิยมที่สุดสำหรับผู้ที่ใช้งานที่ชอบเขียนด้วยภาษา Assembly สามารถหา download ได้ที่ web site ของทาง MICROCHIP โดยจะติดมากับ MPLAB-IDE

MPLAB C17 และ MPLAB C18

เป็น C compiler ของทางค่าย MICROCHIP เอง support เฉพาะเบอร์ PIC17XXX และ PIC18XXX ทำให้ไม่ค่อยได้รับความนิยมนัก อีกทั้งตัว compiler ก็สู้ของค่ายอื่นไม่ค่อยได้ด้วย

HITECH-PIC C compiler

เป็น C compiler ของ HITECH มีประสิทธิภาพการ compiler ให้เป็นภาษา assembly สูง (คือแปลงออกมาเป็น assembly แล้ว code ไม่ใหญ่มาก) แต่ราคา software ค่อนข้างแพงทีเดียว

CCS C compiler

เป็น compiler ที่ประสิทธิภาพเกินราคา คนทั่วไปนิยมใช้เยอะเพราะราคาค่อนข้างถูก

แต่เรื่องขนาด file และประสิทธิภาพของ code ที่ compiled ออกมาได้ยังเป็นที่ยกเถียง กันอยู่ระหว่าง HI-TECH กับ CCS อันไหนแน่นกว่ากัน

PBASIC Compiler

มีโครงสร้างเป็นภาษา BASIC เป็นภาษาสคริปต์ เพราะเขียนง่าย เข้าใจง่าย เขียนสนุก ตัวอย่างเพียบ แต่ขนาดไฟล์ที่ compiled ออกมาได้มีขนาดใหญ่มาาก !! ใส่ลง chip ไม่ค่อยลง ที่สุดแล้ว คงเหมาะสำหรับเขียนแบบสนุกๆ หรือเป็น program ที่ไม่ใหญ่และไม่ซับซ้อน

2.5.3 PIC ชนิดต่างๆ

MCU ในตระกูล PIC ถิ่นแบ่งออกตามชนิดของ PROGRAM MEMORY แบ่งได้เป็น 3 แบบคือ

1. OTP (one time programmable)
2. EPROM (erasable programmable ROM)
3. EEPROM / Flash (electronically erasable programmable ROM)

OTP เป็น chip ที่มีราคาถูกที่สุดในสามประเภท สาเหตุก็มาจากราคา chip แบบ OTP จะสามารถทำการโปรแกรมได้แค่ครั้งเดียวเท่านั้น หลังจาก chip ได้ถูกโปรแกรมไปแล้วจะไม่สามารถโปรแกรมเข้าไปใหม่ได้อีก ดังนั้น chip ประเภทนี้จะนิยมใช้หลังจากได้พัฒนาโปรแกรมจนกระทั่งจุดบกพร่องต่างๆ ในโปรแกรม ไม่มีอีกแล้ว เพราะจะมีต้นทุนต่ำเมื่อเทียบตัว memory ประเภทอื่น จะมีตัวอักษร C แสดงบนตัว chip เช่น 16C84, 16C74



40-LEAD PDIP
"P" OR "PL"

รูปที่ 2.6 ตัวอย่างไมโครคอนโทรลเลอร์แบบ OTP

EPROM เป็น chip ที่มี program memory ที่เมื่อเขียนโปรแกรมเข้าไปแล้วสามารถโปรแกรมใหม่ด้วยการลบโปรแกรมเดิมโดยให้แสง uV (ultra Violet) ส่องผ่านเข้าไปยัง chip ประมาณ 5-10 นาที ดังนั้นที่ด้านบนของ chip จะมีกรอบกระจกเพื่อให้เห็นแสง uV สามารถส่องผ่านเข้าไปในตัว chip ได้ แต่ก็จะมีจำนวนครั้งในการลบโปรแกรมเช่นกัน เมื่อลบโปรแกรมด้วยแสง uV มากๆ เข้าก็จะเกิดการฉีกขาดของโปรแกรมไม่เข้าตัวเอง จะมีตัวอักษร JW หรือว่าดูเอาว่ามีกรอบกระจกอยู่บน chip หรือไม่



รูปที่ 2.7 ตัวอย่างไมโครคอนโทรลเลอร์แบบ EPROM

EEPROM / Flash เป็น chip ที่ออกมาไม่กี่ปีนี้เอง ส่วนของ program memory สามารถอ่านหรือเขียนด้วยสัญญาณทางไฟฟ้า ใช้เวลาในการลบข้อมูลไม่กี่วินาที และสามารถลบ และเขียนใหม่ได้หลายพันครั้ง ทำให้เป็นที่นิยมที่สุดใน 3 ประเภท มีตัวอักษร F เป็นตัวออก เช่น 16F84, 16F877



รูปที่ 2.8 ตัวอย่างไมโครคอนโทรลเลอร์แบบ EEPROM

2.5.4 PIC เบอร์ต่างๆ

ปัจจุบัน MCU ของ PIC มีหลากหลายเบอร์ จนแทบจะเลือกใช้ไม่ถูก แต่ในความเป็นจริงแล้ว การออกเบอร์ใหม่ๆ ออกมาได้มีการกำหนดทิศทางที่แน่นอน ซึ่งเราสามารถ จะแบ่งออกเป็น 6 กลุ่ม ได้แก่

1. PIC12CXXX, PIC12FXXX (FLASH MCUs)
2. PIC16C5X
3. PIC16CXXX
4. PIC17CXXX
5. PIC16FXXX (FLASH MCUs)
6. PIC18CXXX, PIC18FXXX (FLASH MCUs)

PIC16C5X

- มีคำสั่งในภาษา assembly 33 คำสั่ง
- มี I/O, Timer, Watch dog
- มี feature ค่อนข้างน้อย จะไม่มี I2C หรือ SERIAL เพราะเป็น chip ที่ออกมาในยุคแรกๆ ของ microchip

เนื่องจากเป็นเบอร์ที่ออกมานานมากแล้ว จึงมีแต่แบบ OTP และ EPROM feature ก็มีน้อย I2C หรือ USART นี้ไม่ต้องพูดถึงต้องเขียนเองล้วนๆ Microchip คงเห็นจุดด้อยตรงนี้เลยเซ็นตระกูลใหม่เข้ามาคือ PIC16CXXX

PIC16CXXX

- มีคำสั่งในภาษา assembly 35 คำสั่ง
- มี I/O, มี Timer มากกว่า 1 ตัว, Watch dog, I2C, USART, SPI, PWM
- เริ่มมี feature ใส่เข้าไปมากขึ้น ทำให้การ interface กับอุปกรณ์ภายนอกทำได้ง่ายขึ้น

เห็นว่า feature มีเพียบ ทั้ง I2C, USART, PWM ช่วยในการเขียนโปรแกรมได้มากทีเดียว หลังจากนั้นก็ได้ออกตระกูล PIC12CXXX ออกมา ซึ่งเป็น pic ที่มีเพียง 8 ขา ดูแล้วเหมาะกับงานเล็กๆ

จุดเด่นคือมี OSC ขนาด 4 MHz ภายใน และมี DATA memory ส่วนหนึ่งเป็น EEPROM อยู่ภายในด้วย แต่ในส่วนของ PROGRAM memory ยังเป็น OTP และ EPROM อยู่ หลังจากนั้น PIC12FXXX ได้ออกตามหลังมาหลังจาก PIC ได้ออก PIC16FXXX ไปแล้ว

PIC12CXXX และ PIC12FXXX

- มี 8 ขา
- มีคำสั่งในภาษา assembly ทั้งหมด 33 หรือ 35 คำสั่ง
- มี I/O, Timer, Watch dog
- บางเบอร์มี internal EEPROM
- จะมี 4MHz oscillator อยู่ภายในตัว chip เลย ทำให้เราไม่ต้องมี oscillator ภายนอก (แต่หากต้องการความแม่นยำของเวลาก็จำเป็นต้องใช้ Crystal oscillator ภายนอก)

PIC16CXXX และ PIC17CXXX จะออกมาพร้อมๆ กัน แต่ต่างกันที่ PIC17CXXX ถือว่าเป็นเบอร์ที่ใหญ่กว่า มีคำสั่งใน assembly มากกว่า รวมทั้งขนาดของ PROGRAM memory จะมีขนาดมากกว่าในตระกูล PIC16CXXX และสามารถต่อ memory ภายนอกก็ได้

PIC17CXXX

- มีคำสั่งในภาษา assembly 58 คำสั่ง มีคำสั่งการ คูณหาร
- สามารถต่อกับ program memory ภายนอก ได้อย่างได้ถึง 64K x 16 Program memory
- มี I/O, มี Timer มากกว่า 1 ตัว, Watch dog, I2C, USART, SPI, PWM

PIC16F8XXX เป็น PIC ที่ได้รับความนิยมมาก เป็นที่ชื้อหาเพราะเป็น chip ยุคแรกๆ ที่มี PROGRAM MEMORY เป็นแบบ FLASH และมี DATA MEMORY ที่เป็น EEPROM เพิ่มเติมเข้ามา รวมทั้งตัว PROGRAM CHIP สามารถสร้างได้ง่าย อีกทั้งยังมี In Circuit Debugger อยู่ภายในทำให้ไม่จำเป็นต้องพึ่ง Emulator ที่มีราคาแพงอีกต่อไป

PIC16F8XXX (FLASH MCUs)

- มีคำสั่งในภาษา assembly 35 คำสั่ง
- มี I/O, มี Timer มากกว่า 1 ตัว, Watch dog, I2C, USART, SPI, PWM
- มี A/D ขนาด 10 bits

- มี Program memory แบบ Flash สามารถโปรแกรมใหม่ได้หลายครั้ง
- มี EEPROM ภายใน
- ในตระกูล 16F87X สนับสนุน In Circuit Debugger (ICD)

จุดด้อยคือ program memory มีลักษณะเป็น page จึงได้พัฒนาออกมาเป็น PIC18CXXX ซึ่ง PROGRAM MEMORY ไม่ถูกแบ่งเป็น page อีกต่อไป และเพิ่มคำสั่ง ASSEMBLY เป็น 77 คำสั่ง รวมทั้งออกแบบให้ PROGRAM MEMORY มีขนาดใหญ่ขึ้นเพื่อรองรับ การเขียนโปรแกรมภาษา C หลังจากนั้นจึงได้ออก PIC18FXXX ตามมา พร้อมกับ ICD2

PIC18CXXX, PIC18FXXX (FLASH MCUs)

- มีคำสั่งในภาษา assembly 77 คำสั่ง
- มี I/O, มี Timer มากกว่า 1 ตัว, Watch dog, I2C, USART, SPI, PWM, CAN
- มี A/D ขนาด 10 bits
- สามารถต่อกับ program memory ภายนอก โดยอ้างได้ถึง 64K x 16
- มี Program memory เป็นแบบ Flash
- ในตระกูล 18FXXX มี EEPROM ภายใน
- ในตระกูล 18FXXX สนับสนุน In Circuit Debugging (ICD)

2.5.5 TYPES OF OSCILLATOR

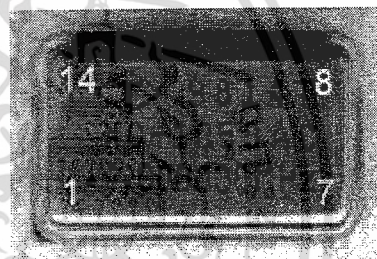
MCU ในตระกูล PIC สามารถเลือก oscillator ได้ว่าจะใช้แบบภายในหรือภายนอก

สำหรับ oscillator ภายใน PIC จะเป็นประเภท RC oscillator ที่ความถี่คงที่ 4 MHz ที่ $V_{dd}=5\text{ V}$ ที่อุณหภูมิ 25 องศาเซลเซียส ที่ต้องระบุอุณหภูมิ เพราะว่า oscillator ประเภท RC ความถี่จะเปลี่ยนแปลงตามอุณหภูมิ

1. CERAMIC RESONATOR นิยมใช้ในกรณีที่ความถี่สูงไม่มาก ขอมรับ ERROR ของ ความถี่ได้พอสมควร เพราะว่ามีราคาถูกเมื่อเทียบกับ OSCILLATOR ที่ต่อภายนอกประเภทอื่น ไม่ต้องมี C ต่อภายนอกด้วย โดยขากลางจะเป็น GND ส่วนอีกสองขาต่อกับ CLKIN และ CLKOUT.

2. QUARTZ CRYSTAL OSCILLATOR จะไม่มีวงจรขยายสัญญาณภายใน มีแต่ crystal ที่กำเนิดสัญญาณความถี่ กำลังต่ำออกมา จะต้องใช้ capacitor ต่อลง GND ของทั้งสองขา ตามรูป และทั้งสองขาต่อเข้ากับ CLKIN และ CLKOUT จะมีราคาแพงกว่า CERAMIC RESONATOR แต่จะให้ความเที่ยงตรงของความถี่ได้ดีกว่า

3. TTL CRYSTAL SQUAR-WAVE OSCILLATOR คือ oscillator ที่มี crystal อยู่ภายใน พร้อมทั้งวงจรอยู่ภายในตัว ให้ความถี่ที่ออกมามีเสถียรภาพมาก แต่จะมีราคาแพง ลักษณะการต่อคือ ขา 14 ต่อ 5V, ขา 7 ต่อ GND, ขา 8 ต่อกับ CLK IN ส่วนขา หนึ่งไม่ใช้, ขา CLKOUT ของ PIC ก็ไม่ต้องต่อ



รูปที่ 2.9 ลักษณะ OSCILLATOR แบบต่างๆ

2.6 Flash Memory

"Flash" เป็นชื่อที่ได้มาจากวิธีการจัดการของ chip ที่นำมาใช้ ซึ่งสามารถลบข้อมูลที่บรรจุภายใน cell ได้ด้วยการกระทำเพียงครั้งเดียว โดยข้อมูลที่บันทึกใน Flash Memory สามารถลบ หรือแก้ไขข้อมูลที่จัดเก็บไว้ในหน่วยความจำที่เรียกว่า "block"

หน่วยความจำแฟลช (Flash Memory) คือ หน่วยความจำขนาดเล็กประเภท non-volatile ที่สามารถบันทึกข้อมูลลงไปได้โดยไม่ต้องใช้แบตเตอรี่ ข้อมูลไม่มีการสูญหายเมื่อปิดสวิตช์ ซึ่งมีส่วนที่ใช้บันทึกข้อมูล เรียกว่า solid state chips ซึ่งใช้กระบวนการทางไฟฟ้าในการบันทึกข้อมูลและมีตัวควบคุมการอ่านและเขียนใน ตัวเอง หน่วยความจำสำรองแบบ solid state เป็นหน่วยความจำแบบ non-volatile ที่เป็นที่นิยมใช้กันอย่างแพร่หลายในอุปกรณ์ Digital เช่น กล้องถ่ายภาพแบบดิจิทัล ที่

สามารถบันทึกข้อมูลได้มากและรวดเร็ว หรือในอุปกรณ์ที่สามารถบันทึกและเคลื่อนย้ายข้อมูลได้สะดวกรวดเร็ว

Flash Memory มักถูกใช้สำหรับการเก็บ control code เช่น BIOS เป็นต้น เนื่องจากง่ายต่อการ update ข้อมูล ซึ่ง BIOS คือ โปรแกรมที่ใช้เพื่อเริ่มการทำงานของระบบคอมพิวเตอร์หลังเปิดสวิตช์ โดย BIOS จะคอยควบคุม หรือ จัดการกับกระแสข้อมูลที่อยู่ระหว่างระบบปฏิบัติการ (Operating System : OS) กับอุปกรณ์ต่าง ๆ ที่ติดตั้งอยู่ เช่น ฮาร์ดดิสก์ กราฟิกการ์ด คีย์บอร์ด เมาส์ หรือ ปริ้นเตอร์ อย่างไรก็ตาม Flash Memory ยังไม่สามารถนำมาใช้งานแทนแรมได้ เพราะแรมต้องการการระบุตำแหน่งในระดับไบต์ ซึ่งละเอียดกว่าไม่ใช่ว่าระดับ block ที่ใช้ใน Flash Memory

ในปัจจุบัน Flash Memory จะถูกนำมาใช้กับอุปกรณ์อิเล็กทรอนิกส์แบบพกพาขนาดเล็ก เช่น กล้องดิจิทัล เครื่องบันทึกเสียง โทรศัพท์มือถือ หรือเครื่องบันทึกข้อมูลมือถือแบบ PDA PC card สำหรับเครื่องคอมพิวเตอร์ Notebooke ที่ต้องการหน่วยความจำที่ออกแบบให้มีขนาดเล็กมากในการจัดเก็บข้อมูลต่าง ๆ

- ข้อดี:
1. สามารถเก็บข้อมูลได้ดีกว่า magnetic disk เนื่องจากการเก็บข้อมูลโดยใช้ chip
 2. ในขณะที่ทำการบันทึกจะไม่มีการหมุนของอุปกรณ์ภายใน flash memory card ทำให้มีความทนต่อการกระทบกระเทือนได้มากกว่า magnetic disk
 3. มีขนาดเล็ก และ สามารถถอดออกได้ง่าย

- ข้อเสีย:
1. มีข้อจำกัดด้านความจุ ซึ่งมีความจุน้อยกว่าฮาร์ดดิสก์
 2. ยังไม่สามารถนำมาใช้แทนหน่วยความจำ RAM ได้

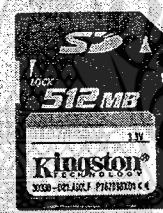
2.7 มาตรฐาน Flash Memory รูปแบบใหม่

ในโลกของเทคโนโลยี หน่วยความจำ Flash Memory กำลังได้รับความนิยมเป็นอย่างสูงในการจัดเก็บข้อมูลภาพ เสียง วิดีโอ บนเครื่องเล่นเพลง ภาพยนตร์แบบพกพา โทรศัพท์มือถือ ทำให้มีผู้คนมีความต้องการในการจัดเก็บข้อมูลไว้ในอุปกรณ์ที่มีขนาดความจุในการเก็บข้อมูลสูงขึ้น จึงได้มีการสร้างมาตรฐานใหม่ของหน่วยความจำ Secure Digital ซึ่งได้รับการพัฒนาขึ้นในรูปแบบ SDHC (Secure Digital High Capacity) หรือเรียกอีกอย่างหนึ่งว่า SD 2.0 นั่นเอง เป็นรูปแบบการจัดเก็บข้อมูลสำหรับการ์ด

หน่วยความจำชนิด SD แต่รองรับความจุที่สูงถึง 4 – 32 GB

มาตรฐานสื่อบันทึกข้อมูล SDHC ใช้ในการจัดเก็บไฟล์ที่มีขนาดใหญ่กว่า 2 GB จึงใช้ระบบไฟล์แบบ FAT32 เมื่อเทียบกับรูปแบบ SD Card แบบเดิม ซึ่งใช้ระบบไฟล์ในรูปแบบ FAT16 โดยมีขนาดเท่ากับ SD Card แบบเดิม แต่ก็ควรจะตรวจสอบก่อนว่า อุปกรณ์ที่เราใช้ รองรับ SDHC หรือไม่ และควรพิจารณาไปถึง Card Reader ด้วย

ในปัจจุบัน กล้องดิจิทัล เครื่องเล่นเพลง โทรศัพท์มือถือรุ่นใหม่ ๆ ต่างก็รองรับมาตรฐานใหม่ ออกผลิตภัณฑ์ที่รองรับ SDHC มากขึ้น จะเห็นได้จากมาตรฐาน MiniSDHC และ MicroSDHC ดังนั้น เรื่องความเข้ากันได้ของมาตรฐานใหม่ ก็ยังคงต้องรอเวลาสักนิด ให้ผลิตภัณฑ์ใหม่ๆ ออกมารองรับ SDHC ให้แพร่หลายเสียก่อน เช่นเดียวกับมาตรฐานการเชื่อมต่อ USB ซึ่งปัจจุบันเป็นมาตรฐาน 2.0 เรียบร้อยแล้ว



รูปที่ 2.10 SD Card

2.8 SD card (Secure Digital)

คำว่า SD card หรือ SD เลขๆ ที่เรียกกันจนติดปากนั้น มันย่อมาจาก Secure Digital เป็นสื่อเป็นข้อมูลที่ได้รับความนิยมมาก ถึงแม้ว่ามันจะไม่ค่อยจะ secure แบบชื่อก็ตาม สำหรับ SD Card นั้นมันก็คือ nand flash memory

NAND Flash ถูกพัฒนาขึ้นมาเพื่อเป็นหน่วยเก็บข้อมูลขนาดกลาง สำหรับกล้องถ่ายรูปดิจิทัล, โทรศัพท์มือถือ และ PDA คุณสมบัติของ NAND Flash ROM คือ มีราคาถูก เก็บข้อมูลได้มาก และมีอายุการใช้งานได้นานกว่า NOR Flash ROM แบบเดิมถึง 10 เท่า

ในส่วนของคุณสมบัติของ NAND Flash ROM คือ มีความเร็วในการจัดการข้อมูลช้ากว่า เพราะเป็นโครงสร้างการจัดการแบบ execute in place (XIP) กล่าวคือ กระบวนการทำงานของเครื่องที่ใช้ NAND

Flash ROM จำเป็นจะต้องอ่านข้อมูลจาก NAND Flash ROM ก่อน แล้วค่อยเรียกขึ้นไปทำงาน คล้ายกับการทำงานของฮาร์ดดิสก์ในเครื่องพีซีเดสก์ท็อป และ main memory

ดังนั้น ส่วนของ main memory ที่ใช้ NAND Flash ROM จำเป็นจะต้องมีขนาดค่อนข้างใหญ่เพื่อใช้สำรองระบบการจัดการข้อมูลแบบนี้ จึงทำให้พื้นที่ของส่วนความจำเพื่อใช้เก็บโปรแกรมและข้อมูลอื่นๆ จะเหลือน้อยลง

สำหรับข้อดีของ SD Card ก็คือ ไม่มีส่วนของการเคลื่อนไหวของกลไกต่างๆ ใน card จึงสามารถทำตกได้โดยไม่เสียหาย และไม่ต้องอาศัยพลังงานใดๆ ไปหล่อเลี้ยงเอาไว้ ไม่ว่าจะเป็น SD Card, Mini SD หรือ Micro SD มันก็คือหลักการเดียวกัน การทำงานเหมือนกันแต่จะต่างกันที่ขนาดรูปร่าง เพื่อให้สอดคล้องกับพื้นที่ของอุปกรณ์ที่จะใช้งานร่วม

2.8.1 รูปแบบพื้นฐาน

เอสดีการ์ดเป็นหน่วยความจำแบบแฟลชที่มีต้นกำเนิดจากการ์ดความจำรุ่นก่อนหน้าคือ มัลติมีเดียการ์ด (Multimedia card-MMC) แต่มีขนาดที่หนากว่า มีอัตราการรับส่งข้อมูลสูงกว่า มักมีปุ่มป้องกันการเขียนข้อมูลทับ และมีคุณสมบัติควบคุมสิทธิการใช้แบบดิจิทัล (Digital rights management) แต่ไม่มีการใช้คุณสมบัตินี้อย่างแพร่หลายมากนัก ขนาดมาตรฐานของเอสดีการ์ดคือ 32 มิลลิเมตร × 24 มิลลิเมตร × 2.1 มิลลิเมตร แต่สามารถลดความหนาให้เหลือ 1.4 มิลลิเมตร เท่ากับมัลติมีเดียการ์ดได้

2.8.2 อัตราการรับส่งข้อมูล

อัตราการรับส่งข้อมูลของเอสดีการ์ดใช้หน่วยวัดเดียวกับอัตราการรับส่งข้อมูลของซีดีรอม โดยความเร็ว 1 เท่า มีค่าเท่ากับ 150 กิโลไบต์/วินาที เอสดีการ์ดจะมีความเร็วเริ่มต้นที่ 6 เท่า (900 กิโลไบต์/วินาที) การ์ดความเร็วสูงจะมีอัตราการรับส่งข้อมูลที่ 66 เท่า (10 เมกะไบต์/วินาที) และการ์ดระดับบนสุดจะมีความเร็วในการรับส่งข้อมูลอยู่ที่ 150 เท่าหรือสูงกว่า กล้องถ่ายภาพดิจิทัลบางรุ่นต้องการการ์ดความเร็วสูงเพื่อให้สามารถถ่ายภาพต่อเนื่องได้อย่างรวดเร็วและบันทึกภาพเคลื่อนไหวได้อย่างนุ่มนวล ในช่วงปลายปีค.ศ. 2005 (พ.ศ. 2548) อุปกรณ์ส่วนใหญ่รองรับมาตรฐานเอสดีการ์ดรุ่น 1.01 ซึ่งสามารถรับความเร็วสูงสุดได้ที่ 66 เท่า ส่วนมาตรฐาน 1.1 สามารถรับความเร็วสูงสุดได้ถึง 133 เท่า

2.8.3 อุปกรณ์ที่รองรับ

อุปกรณ์ที่สามารถอ่านเอสดีการ์ดได้จะสามารถอ่านมัลติมีเดียการ์ดได้เช่นกัน แต่เอสดีการ์ดไม่สามารถใส่ในช่องใส่มัลติมีเดียการ์ดที่มีขนาดบางกว่าได้ เอสดีการ์ดสามารถใส่ในช่องใส่คอมแพคแฟลช (CompactFlash) หรือ พีซีการ์ด (PC Card-PCMCIA) โดยการใช้ตัวแปลง ส่วน มินิเอสดี (miniSD) และ ไมโครเอสดี (microSD) สามารถใช้ตัวแปลงโดยตรงใส่ในช่องใส่เอสดีการ์ดได้ นอกจากนี้เอสดีการ์ดยังสามารถอ่านผ่านเครื่องอ่านที่จะใช้กับพอร์ตยูเอสบี, ไฟร์ไวร์ รวมถึงพอร์ตแบบขนาน และยังสามารถใช้ผ่านฟลอปปีดิสก์ ผ่านตัวแปลงได้เช่นกัน

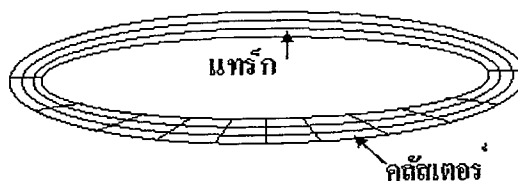
2.8.4 SD Card ทำงานอย่างไร

ภายใน SD card จะมี microcontroller ขนาดเล็กคล้ายกับการทำงานของ CPU ทำหน้าที่คอยดูแลและติดต่อกับอุปกรณ์ต่างๆ ที่ใช้ SD Card ในขณะนั้น เช่น PDA Phone หรือ โทรศัพท์มือถือ โดย microcontroller จะทำหน้าที่จัดการเรื่องข้อมูลต่างๆ ที่ถูกอ่านและถูกเขียนลงใน Flash โดย CPU หรือ microcontroller จะทำงานเฉพาะในช่วงที่มีการใช้งานข้อมูลใน card เท่านั้น

การทำงานของ SD Card จะเร็วหรือช้าขึ้นก็ขึ้นอยู่กับความเร็วของ CPU ในเครื่อง PDA Phone หรือ อุปกรณ์ต่างๆ ด้วย เช่น เมื่อมีการใช้งาน SD Card บน โทรศัพท์มือถือ โทรศัพท์จะส่งข้อมูลแจ้งไปยัง SD card ว่า microcontroller นั้น สามารถรองรับความเร็วสูงสุดที่เท่าไร ก็ให้ใช้ความเร็วนั้น เพราะฉะนั้นหาก clock ของโทรศัพท์ยังเร็ว การทำงานของ card ก็จะเร็วตามไปด้วย โดยข้อมูลจะถูกส่งในแต่ละวินาทีได้รวดเร็วขึ้น

2.9 ระบบไฟล์ FAT16 และ FAT32 ทำงานอย่างไร

ฮาร์ดดิสก์เมื่อเราได้ฟอร์แมตแล้วจะจัดรูปแบบของฮาร์ดดิสก์ออกเป็นร่องเรียกว่า แทร็ก (Track) และในแต่ละแทร็กก็จะแบ่งเป็นคลัสเตอร์ (Cluster) และในแต่ละคลัสเตอร์จะถูกแบ่งออกเป็นเซ็กเตอร์ (ดังรูป)



รูปที่ 2.11 รูปแบบของฮาร์ดดิสก์ออกเป็นร่อง

โดยปกติแล้วไฟล์หนึ่งๆ จะถูกแบ่งออกเป็นท่อนๆ กระจายอยู่ในเซ็กเตอร์ต่างๆ ในฮาร์ดดิสก์ไม่ได้ต่อเนื่องกันเป็นชิ้นเดียวกันจนจบไฟล์ ขณะเดียวกันระบบปฏิบัติการก็จะมีวิธีการรวบรวมข้อมูลที่กระจายอยู่เนี่ยมารวมกันเป็นไฟล์ การที่มันสามารถรวบรวมข้อมูลที่กระจายอยู่ขึ้นมาได้ ก็เพราะมันเก็บชื่อและตำแหน่งของข้อมูลแต่ละท่อนเอาไว้ที่ FAT (File Allocation Table) ดังนั้นถ้าหาก FAT ตัวนี้หายไป ฮาร์ดดิสก์ก็จะเรียกข้อมูลขึ้นมาไม่ได้

และโปรแกรม Scandisk ที่พวกเราอีกหลายคนมักจะตั้งความหวังกับมันไว้สูง ซึ่งที่จริงแล้วการสแกนดิสก์นั้น ไม่ได้อ่านข้อมูลทั้งหมดบนดิสก์ และไม่ได้ซ่อมแซมดิสก์แบบทางกายภาพอะไรเลย สแกนดิสก์เพียงแต่อ่านข้อมูลใน FAT และไปอ่านข้อมูลในช่วงต้นจากเซ็กเตอร์ต่างๆ ของไฟล์เท่านั้น มันจะไล่อ่านข้อมูลช่วงต้นๆ นี้ไปเรื่อยๆ จนหมดทั้งตัวฮาร์ดดิสก์ หากสแกนไปพบไฟล์ที่อยู่ในแบดเซ็กเตอร์มันก็จะช่วยย้ายไฟล์ไปอยู่ที่อื่น แล้วจึงค่อยไปเขียนตำแหน่งบอจุดที่ย้ายไปให้แก่ FAT อีกทีหนึ่ง ทำให้อีกหลายๆ คนที่ตั้งความหวังเอาไว้เมื่อฮาร์ดดิสก์เกิดแบดเซ็กเตอร์ขึ้นแล้วสแกนดิสก์จะช่วยได้นั้นเป็นการเข้าใจผิด และการฟอร์แมตฮาร์ดดิสก์ใหม่ก็ไม่ได้ช่วยแก้ไขอะไรเหมือนกัน การแก้ไขแบดเซ็กเตอร์นั้นจะต้องใช้โปรแกรมประเภทจัดการพาร์ติชันช่วยแก้ไข เช่น Partition Commander หรือใช้ Fdisk ลบพาร์ติชันเก่าออกแล้วสร้างพาร์ติชันขึ้นใหม่แต่วิธีนี้ข้อมูลบนพาร์ติชันนั้นจะหายไป

2.9.1 FAT (File Allocation Table)

พื้นที่ในฮาร์ดดิสก์ที่ใช้เก็บข้อมูลนั้นจะถูกแบ่งออกเป็นเซ็กเตอร์ และแต่ละเซ็กเตอร์จะมีขนาด 512 ไบต์ ขณะที่คอมพิวเตอร์ติดต่อกับฮาร์ดดิสก์นั้น มันไม่ได้ติดต่อกันแบบเซ็กเตอร์ต่อเซ็กเตอร์ แต่มันติดต่อโดยส่งข้อมูลกันเป็นคลัสเตอร์ ซึ่งในคลัสเตอร์หนึ่งๆ ก็จะประกอบไปด้วยเซ็กเตอร์จำนวนหนึ่ง (อาจจะเป็น 8,16,32,64 ขึ้นอยู่กับขนาดของพาร์ติชัน) และภายในแต่ละเซ็กเตอร์นั้นก็ประกอบไปด้วยส่วนหนึ่งของไฟล์หรืออาจจะได้เต็มไฟล์ หากไฟล์นั้นมีขนาดไม่เกิน 512 ไบต์

เมื่อฮาร์ดดิสก์จะส่งข้อมูลออกไปแต่ละครั้ง มันก็จะต้องไปอ่านข้อมูลตามเซกเตอร์ต่างๆ ของไฟล์ตามอินเด็กซ์ที่บอกตำแหน่งไว้ เมื่ออ่านแล้วคอนโทรลเลอร์ก็จะนำมาประกอบกันเป็นไฟล์ ถ้าหากไฟล์ที่ต้องการนั้นมีขนาดใหญ่กว่าหนึ่งคลัสเตอร์ฮาร์ดดิสก์ก็ต้องไปอ่านข้อมูลที่เหลือของไฟล์นั้นออกมาแล้วส่งออกไปอีกทีหนึ่ง

FAT16 และ FAT32 ซึ่งไมโครซอฟท์เป็นผู้ออกแบบขึ้นใช้สำหรับวินโดวส์ ความแตกต่างของ FAT ทั้งสองตัวนี้ที่เด่นชัดที่สุดก็คือการรองรับความจุสูงสุด และประสิทธิภาพในการจัดการกับข้อมูล FAT32 นั้นสามารถรับความจุสูงสุดได้ที่ 2 เทอราไบต์ (Terabyte หรือ 2,199 GB) มากกว่า FAT16 ที่รับความจุสูงสุดได้แค่ 2 กิกะไบต์ และ FAT32 ยังใช้ขนาดคลัสเตอร์ที่เล็กกว่า มีประสิทธิภาพมากกว่า FAT16

Hard disk Partition Size	Cluster (Allocation Unit) Size	FAT Type
0 - 127M	4 Sectors or 2,048 (2K) bytes	16-bit
128 - 255M	8 Sectors or 4,096 (4K) bytes	16-bit
256 - 511M	16 Sectors or 8,192 (8K) bytes	16-bit
512 - 1,023M	32 Sectors or 16,384 (16K) bytes	16-bit
1,024 - 2,047M	64 Sectors or 32,768 (32K) bytes	16-bit
0 - 259M	1 Sectors or 512 bytes	32-bit
260 - 511M	2 Sectors or 1,024 (1K) bytes	32-bit

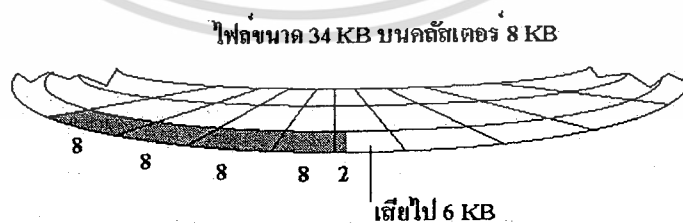
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

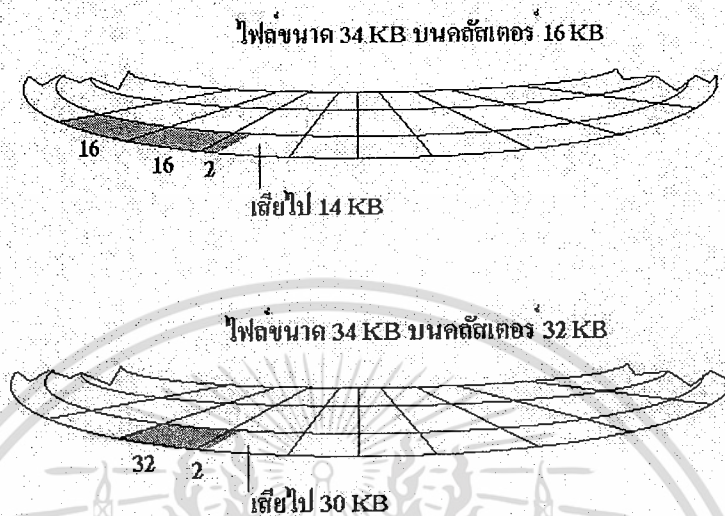
512 - 8,191M	8 Sectors or 4,096 (4K) bytes	32-bit
8,192 - 16,383M	16 Sectors or 8,192 (8K) bytes	32-bit
16,384 - 32,767M	32 Sectors or 16,384 (16K) bytes	32-bit
32,768M - 2,048G	64 Sectors or 32,768 (32K) bytes	32-bit

ตารางที่ 2.1 ตารางเปรียบเทียบความจุของฮาร์ดดิสก์ของ FAT 16 และ FAT 32

ระบบไฟล์ FAT16 ใช้การนับคลัสเตอร์แบบ 16 บิต หมายความว่า FAT16 ใช้เลขไบนารีนับอยู่ 16 หลัก เมื่อคิดตามจำนวนแล้วก็จะได้เท่ากับ $2^{16}-1$ (ลบ 1 เพราะมีการนับเลขศูนย์เข้าไปด้วย) หรือได้สูงสุดเท่ากับ 65,535 คลัสเตอร์ และ 65,535 นี้คือจำนวนคลัสเตอร์สูงสุดของ FAT16 โดยเอา $65,535 \times 64 \times 512$ ก็จะได้ 2,147,450,880 ไบต์ หรือ 2 กิกะไบต์ คือความจุสูงสุด

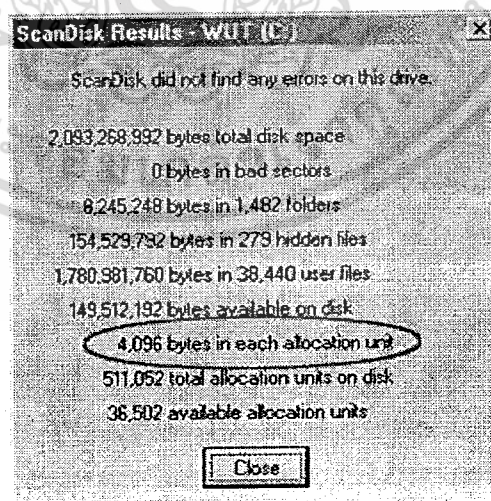
ระบบไฟล์ FAT32 นั้นใช้เลขไบนารีนับถึง 32 หลัก ทำให้สามารถนับจำนวนคลัสเตอร์ได้มากกว่า FAT16 และขนาดคลัสเตอร์ก็เล็กกว่าด้วย ที่กล่าวว่า FAT32 สามารถรองรับฮาร์ดดิสก์ที่มีความจุสูงสุดได้ถึง 2 เทราไบต์ ก็คือ $2^{32}-1$ จะได้เท่ากับ 4,294,967,295 เซ็กเตอร์ คูณด้วย 512 ไบต์ ก็จะได้เท่ากับ 2,199,023,255,040 ไบต์ หรือเท่ากับ 2 เทราไบต์นั่นเอง





รูปที่ 2.12 การจัดเก็บข้อมูลของคลัสเตอร์

จากภาพที่แสดงจึงสรุปได้ว่ายิ่งมีขนาดคลัสเตอร์ใหญ่มากขึ้นแค่ไหน ก็ยังมีพื้นที่สูญเปล่ามากขึ้นเท่านั้นมาถึงตรงนี้หลายคนคงเกิดคำถามว่าแล้วฮาร์ดดิสก์ของฉันมีขนาดคลัสเตอร์เป็นเท่าไร? ง่าย ๆ ก็คือใช้สแกนดิสก์รันแล้วดูที่ค่า bytes in each allocation unit ก็คือขนาดที่เป็นไบต์ของคลัสเตอร์ และ ถ้านำ 512 ไปหารก็จะได้จำนวนเซกเตอร์ในหนึ่งคลัสเตอร์



รูปที่ 2.13 Dialog Scan Hard Disk

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มาถึงตรงนี้ถ้าฮาร์ดดิสก์มีขนาดใหญ่ตั้งแต่ 32 GB ขึ้นไปขนาดของคลัสเตอร์จะใหญ่ถึง 32 KB แล้วเวลานี้ฮาร์ดดิสก์ที่นิยมติดตั้งบนคอมพิวเตอร์ที่ซื้อกันในปัจจุบัน (มีข.45) ก็มีขนาดใหญ่ถึง 40 GB เข้าไปแล้ว อย่างที่กล่าวคลัสเตอร์ยังมีขนาดใหญ่พื้นที่สูญเสียยิ่งมาก ถ้าจะแบ่งเป็น 2 พาร์ทิชัน คลัสเตอร์ก็จะมีขนาด 16 KB ก็ยังมีขนาดใหญ่อยู่ จะแบ่งพาร์ทิชันมากกว่านี้ก็จะมากเกินไปจนสิ้นเวลาจัดเก็บไฟล์

แต่ก็มีวิธีที่จะฟอร์แมตฮาร์ดดิสก์ที่มีขนาดใหญ่ในปัจจุบันให้มีขนาดคลัสเตอร์เล็กกว่า 32 KB ได้โดยระบุพารามิเตอร์ของโปรแกรม Format ดังต่อไปนี้

FORMAT [drive:] /Z:n

ค่า n ก็คือจำนวนเซกเตอร์ต่อหนึ่งคลัสเตอร์

ตัวอย่างถ้าเราจะฟอร์แมตฮาร์ดดิสก์ขนาด 40 GB ให้มีขนาดคลัสเตอร์ 4 KB (4,096) จะใช้คำสั่งดังนี้

A:\>Format C: /Z:8 (ดูค่า n ได้จากตารางข้างต้น)

แต่ที่มิได้ก็คือ เมื่อคลัสเตอร์มีขนาดเล็กลงจำนวนคลัสเตอร์ก็จะมากขึ้น และคอมพิวเตอร์ก็ติดต่อกับฮาร์ดดิสก์เป็นคลัสเตอร์ จึงทำให้เวลาในการเข้าถึงข้อมูลมากขึ้น (Access Time) และถ้าขนาดคลัสเตอร์ใหญ่ขึ้นจำนวนคลัสเตอร์ก็น้อยลง เวลาในการเข้าถึงข้อมูลก็จะน้อยลงไปด้วย (เวลา ยิ่งน้อยยิ่งดี) และบทความนี้ก็ไม่แนะนำให้ฟอร์แมตฮาร์ดดิสก์โดยกำหนดขนาดคลัสเตอร์เอง ควรใช้ค่าดีฟอลต์ดีกว่าคือฟอร์แมตไปตามปกติ

กลับมาที่ FAT32 กันต่อ FAT ก็คือระบบฐานข้อมูลที่เก็บเรคคอร์ดข้อมูลของไฟล์แต่ละตัวไว้ โดยอยู่ในรูปของตาราง แต่ละเรคคอร์ดใน FAT จะมีขนาด 32 บิต สมมติว่าในฮาร์ดดิสก์มีไฟล์ 100 ไฟล์ FAT ก็จะมีขนาด 3,200 บิต (100 ไฟล์ x 32 บิต) มาดูกันต่อว่า 32 บิตในหนึ่งเรคคอร์ดมีอะไรบ้าง

บิตที่ 1 - 8 : ชื่อไฟล์ (File Name)

บิตที่ 9 - 11: นามสกุล (Extension)

ไบต์ที่ 12 : แอตทริบิวต์ (Attribute)

ไบต์ที่ 13 - 22 : สงวนไว้เพื่อเป็นฟีเจอร์ (Reserved bites for latter features)

ไบต์ที่ 23 - 24 : เวลาที่บันทึก (Time Written)

ไบต์ที่ 25 - 26 : คลัสเตอร์เริ่มต้น (Starting Cluster)

ไบต์ที่ 29 - 32 : ขนาดของไฟล์ (File Size)

ฟิลด์ที่น่าสนใจในนี้มี "คลัสเตอร์เริ่มต้น" คลัสเตอร์เริ่มต้นนี้ใช้เป็นจุดเริ่มต้นในการอ่านข้อมูล และถ้าเป็นข้อมูลไฟล์เดียวกัน ในแต่ละเซ็กเตอร์นั้นก็จะมีข้อมูลส่วนหนึ่งที่ตรงกันคือหมายเลขของคลัสเตอร์เริ่มต้น ซึ่งเมื่อระบบต้องการข้อมูลก็จะอ่าน FAT เป็นอันดับแรก จากนั้นจะอ่านคลัสเตอร์เริ่มต้น เมื่อรู้ตำแหน่งเริ่มต้นแล้วก็จะไปอ่านข้อมูลในคลัสเตอร์ และภายในคลัสเตอร์แต่ละตัวจะเก็บตำแหน่งเซ็กเตอร์ที่ต้องอ่านเป็นลำดับต่อไป ซึ่งข้อมูลในคลัสเตอร์ที่คอยบอกตำแหน่งเซ็กเตอร์เรียกว่า Pointer เมื่อระบบไล่อ่านข้อมูลในคลัสเตอร์นั้นออกมาหมดแล้วแต่ยังไม่ครบก็จะอ่านในคลัสเตอร์ต่อไป ซึ่งก็จะอ่านไปเรื่อยๆจนกว่าจะไม่พบ Pointer

ฟิลด์ต่อมาที่น่าสนใจก็คือ "ชื่อไฟล์" ชื่อไฟล์นี้คือจะใช้ในการยกเลิกการลบไฟล์โดยใช้ Undelete คือเมื่อเราลบไฟล์ใดไฟล์หนึ่งไปแล้วมันไม่ได้ถูกลบไปจริงๆ ระบบเพียงแค่เปลี่ยนอักขระตัวแรกของชื่อไฟล์เป็นอักขระพิเศษเท่านั้น และก็จะคงอยู่อย่างนั้นจนกว่าจะถูกเขียนทับ ส่วนการกู้ไฟล์นั้นก็กลับมาก็คือการเปลี่ยนอักขระดังกล่าวกลับเป็นตัวเดิมหรือเป็นตัวอื่น ก็เหมือนกับการกู้ไฟล์ด้วย UnErase ของ Norton Utilities

อีกฟิลด์ที่น่าสนใจในคือ ไบต์สงวน ทั้ง FAT32 และ NTFS ใช้การนับแบบ 32 บิต ซึ่งมากพอที่จะใส่ออฟชั่นอื่นๆของไฟล์ลงไปได้ โดยเฉพาะ NTFS ที่สามารถใส่ข้อมูล เช่น การรักษาความปลอดภัย การบีบอัดข้อมูล เป็นต้น

2.10 พาร์ติชันแบบ 32 บิต (FAT32)

ปัจจุบันฮาร์ดดิสก์ขนาด 2.1 GB เป็นมาตรฐานไปแล้ว แต่พาร์ติชันแบบ 16 บิต (FAT16) ไม่สามารถจัดการฮาร์ดดิสก์ได้อย่างมีประสิทธิภาพ FAT32 จึงถูกพัฒนาเพื่อแก้ปัญหา โดย FAT32

สามารถกำหนดขนาดต่อพาร์ติชันได้ถึง 2 เทราไบต์ (2048 GB) และหากขนาดพาร์ติชันไม่เกิน 8 GB ขนาดของคลัสเตอร์จะเท่ากับ 4 กิโลไบต์ ซึ่งช่วยเพิ่มประสิทธิภาพการใช้พื้นที่ฮาร์ดดิสก์เก็บข้อมูลมากขึ้น (ดูตารางเปรียบเทียบ FAT16 และ FAT32 ข้างล่าง)

อีกประการหนึ่งก็คือ ปัจจุบัน Controller EIDE บนเมนบอร์ดออกแบบสำหรับฮาร์ดดิสก์แบบ LBA Mode ทั้งหมดแล้ว รวมทั้งฮาร์ดดิสก์เองก็เช่นเดียวกัน แต่ยังไม่มีการใช้ประโยชน์จากฮาร์ดดิสก์ และเมนบอร์ดดังกล่าวได้อย่างเต็มประสิทธิภาพ FAT32 จึงเป็นทางแก้ปัญหานั้น

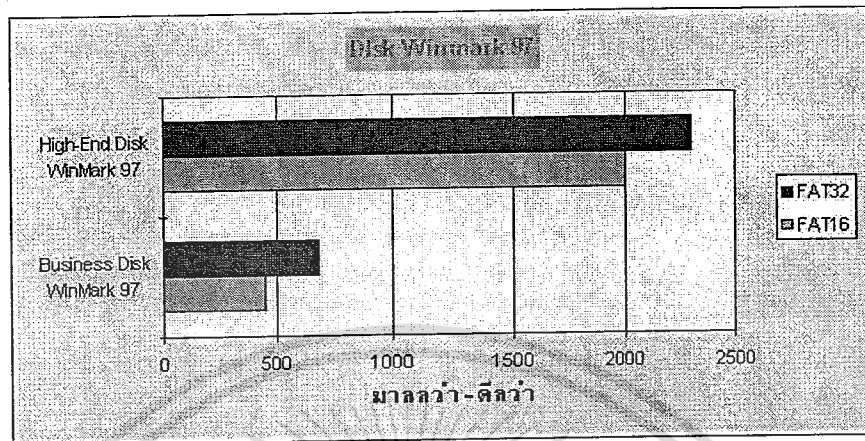
Cluster size	Partition Size	FAT Type	Note
4K (4096 bytes)	16 MB	FAT12	-
2K (2048 bytes)	32 MB	FAT16	(DOS versions < 4.0
2K (2048 bytes)	128 MB	FAT16	(DOS versions >= 4.0
4K (4096 bytes)	256 MB	FAT16	-
8K (8192 bytes)	512 MB	FAT16	-
16K (16384 bytes)	1024 MB (1GB)	FAT16	-
32 K (32768 bytes)	2 GB	FAT16	-

ตาราง 2.2 ตารางเปรียบเทียบขนาดของคลัสเตอร์เมื่อกำหนดขนาดของพาร์ติชันต่างๆ กันใน FAT16

Partition size	Cluster size
less than 260 MB	512 bytes
260 MB - 8 GB	4 kilobytes (KB)
8 GB - 16 GB	8 KB
16 GB - 32 GB	16 KB
greater than 32 GB	32 KB

ตารางที่ 2.3 ตารางเปรียบเทียบขนาดของคลัสเตอร์เมื่อกำหนดขนาดของพาร์ติชันต่างๆ กันใน FAT32

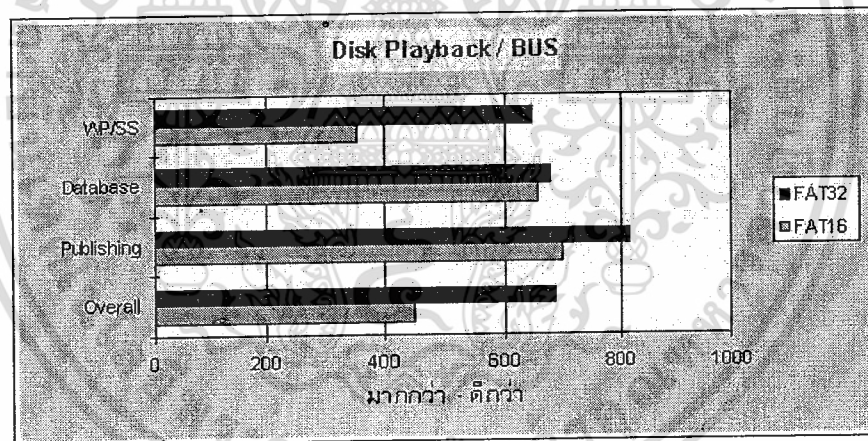
จากตาราง จะเห็นว่าหากต้องการเก็บไฟล์ขนาด 1 กิโลไบต์บนฮาร์ดดิสก์ขนาดต่อพาร์ติชัน FAT 16 ขนาด 1.2 GB จะเสียพื้นที่ถึง 32 กิโลไบต์ แต่หากเปลี่ยนเป็น FAT32 จะใช้พื้นที่เพียง 4 กิโลไบต์ ดังนั้นประโยชน์ของ FAT 32 ที่เห็นได้อย่างชัดเจนคือ ประหยัดพื้นที่ฮาร์ดดิสก์ ถึงแม้ว่า ในทางทฤษฎีแล้วหากจำนวนคลัสเตอร์เพิ่มจะส่งผลให้ระยะเวลาเข้าถึงข้อมูล (Access Time) เพิ่มขึ้น แต่จากการทดสอบวัดประสิทธิภาพโดย Winbench 97 พบว่าประสิทธิภาพของฮาร์ดดิสก์ที่ใช้พาร์ติชันแบบ FAT32 เพิ่มขึ้น ซึ่งอาจแยกพิจารณาได้สองกรณี กรณีแรกคือ ประสิทธิภาพของฮาร์ดดิสก์เพิ่มขึ้น เนื่องจาก Windows 95 OSR2 ปรับปรุง BUS Mastering ส่วนของ ESDI Controller ทำให้การส่งข้อมูลดีขึ้น หรืออาจพิจารณาอีกด้านหนึ่งว่าประสิทธิภาพของฮาร์ดดิสก์ไม่ได้เพิ่มขึ้น แต่เนื่องจาก Winbench ไม่สามารถวัดประสิทธิภาพของฮาร์ดดิสก์พาร์ติชัน 32 บิต ได้อย่างถูกต้องแม่นยำ ซึ่งเป็นไปได้ค่อนข้างน้อย เพราะทดสอบมากกว่า 10 ครั้ง ให้ผลเหมือนกัน



รูปที่ 2.14 กราฟเปรียบเทียบ Highend Disk Winmark ของ FAT16 และ FAT32

จากกราฟเปรียบเทียบ Highend Disk Winmark เห็นได้ว่าเมื่อใช้ FAT32 จะได้ประสิทธิภาพ

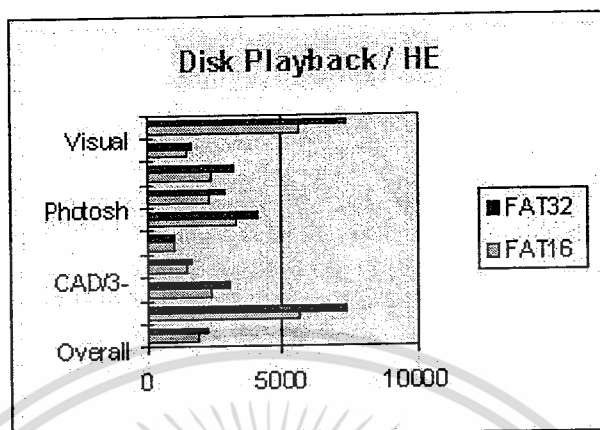
ดีกว่า



รูปที่ 2.15 กราฟเปรียบเทียบ Disk Performance / BUS ของ FAT16 และ FAT32

จากกราฟเปรียบเทียบ Disk Performance / BUS เห็นได้ว่า Windows 95 OSR2 นั้น

ได้รับการพัฒนาส่วนของ BUS Mastering ดีกว่า



รูปที่ 2.16 กราฟเปรียบเทียบ Disk Performance / BUS ของ FAT16 และ FAT32

กราฟเปรียบเทียบ Disk Performance / BUS พบว่าประสิทธิภาพของฮาร์ดดิสก์ที่ใช้พาร์ติชัน FAT 32 ให้ประสิทธิภาพการถ่ายเทข้อมูลดีกว่า

ข้อเสียของ FAT32 คือ หากกำหนดขนาดต่อพาร์ติชันมากกว่า 1 GB เวลาต้องการเรียงลำดับข้อมูล (Defragmentation) จะใช้เวลานานมาก นอกจากนี้ความไม่เข้ากันกับ OS ตัวอื่นก็เป็นปัญหาเช่นกัน อย่างไรก็ตามเป็นที่น่าสังเกตว่า FAT 32 เป็นแนวทางของไมโครซอฟต์ในการนำไปสู่ Windows 98 หรือที่เรียกว่าเมมphis (Memphis) ซึ่งเป็น OS ตัวต่อไปที่จะใช้ FAT 32 และไมโครซอฟต์มีแผนการนำออกขายในไตรมาสที่สองของปี 2541

2.10.1 ข้อควรคำนึงเกี่ยวกับ FAT32

1. หากคอมพิวเตอร์เครื่องหนึ่งแบ่งฮาร์ดดิสก์เป็นหลายพาร์ติชันและติดตั้ง OS หลายๆตัว และประเภทของพาร์ติชันแตกต่างกัน เมื่อบูตด้วย OS FAT16 หรือ NTFS จะมองไม่เห็นพาร์ติชันที่ฟอร์แมตด้วย FAT32 อย่างไรก็ตามหากเข้าถึงข้อมูลในพาร์ติชัน FAT32 ด้วยเครือข่ายจะสามารถเข้าถึงข้อมูลได้

2. ยูทิลิตี้ส์เกี่ยวกับการจัดการฮาร์ดดิสก์ที่ออกแบบมาสำหรับ FAT16 เช่น Norton 8.0 หรือ Norton Utilities 9.0 สำหรับ วินโดวส์ 95 จะเอาไปใช้กับฮาร์ดดิสก์ที่ฟอร์แมตแบบ FAT32 ไม่ได้ (ถ้าฝืนใช้จนได้ ก็แสดงว่าความโชคร้ายมาเยือนแล้ว

3. BIOS เก้าที่ออกแบบมาสำหรับเครื่องตระกูล 486 ไม่สามารถใช้งาน FAT32 ได้ เนื่องจาก FAT32 ต้องใช้กับฮาร์ดดิสก์แบบ LBA เท่านั้น

4. เกี่ยวกับความไม่เข้ากันของ FAT32 Application หลัก ๆ ใช้งานได้ตามปกติ รวมถึงเกมส์ด้วย ยกเว้น Application ประเภททูลิตี้ ดังที่กล่าวในข้อ 2 ที่ไม่สามารถใช้งานได้

5. หากต้องการใช้ FAT 32 เมนบอร์ดต้องสนับสนุนฮาร์ดดิสก์แบบ EIDE และ LBA Mode เนื่องจากหากฟอร์แมตแบบ FAT32 ขนาดของพาร์ติชันต่ำสุดจะเท่ากับ 512 เมกะไบต์

2.11 SPI Mode

เป็นทางเลือกหนึ่งในการติดต่อกับ host ภายนอก ซึ่งหน่วยความจำจะสามารถติดต่อโดยช่องทาง SPI Port หรือ GPIO Port ที่มีไมโครคอนโทรลเลอร์ทั่วไปเพราะฉะนั้น SPI Mode

จะเหมาะสมกับการประยุกต์ใช้งานในโครงการนี้โดยจะทำการเลือกโหมดการทำงานเป็น โหมด 0

2.11.1 Command and response

ใน SPI Mode ทิศทางข้อมูลจะอยู่จะอยู่บน Signal Line ซึ่งการโอนย้ายอนุกรมข้อมูลจะกำหนดเป็นไบต์ ส่วนเฟรมของ command เป็นการส่งผ่านจาก host ไปยังการ์ดซึ่งมีความยาวของขนาดเท่ากับ 6 ไบต์ ต่อ 1 คำสั่ง command เมื่อเฟรมของ command เป็นการส่งผ่านไปยังการ์ดแล้วค่า Response (R1, R2, R3) ของ command นั้นจะถูกส่งกลับมายัง host ซึ่ง host นั้นจะต้องอ่านค่าตลอดจนกว่าจะส่งมาอย่างถูกต้องของ Response ที่ command นั้นเพราะการส่งผ่านข้อมูลของการ์ดมายัง host นั้นจะถูกจะขับจากสัญญาณนาฬิกาที่ host กำหนดมานั้นเอง Command response time นั้นจะมีโครงสร้างจาก 0 ถึง 8 ไบต์ โดยสัญญาณ CS นี้จะต้องควบคุมให้เป็นไบต์ 0 ในระหว่างการติดต่อ Command, Response, Data transfer ยังคงกระทำอยู่ ส่วนไบต์ CRC จะเป็นส่วนประกอบของชุดเฟรมของ Command ซึ่งเป็นทางเลือกใน SPI Mode สำหรับตรวจสอบความถูกต้องของข้อมูลที่ส่งออกไป

2.11.2 SPI Command Set

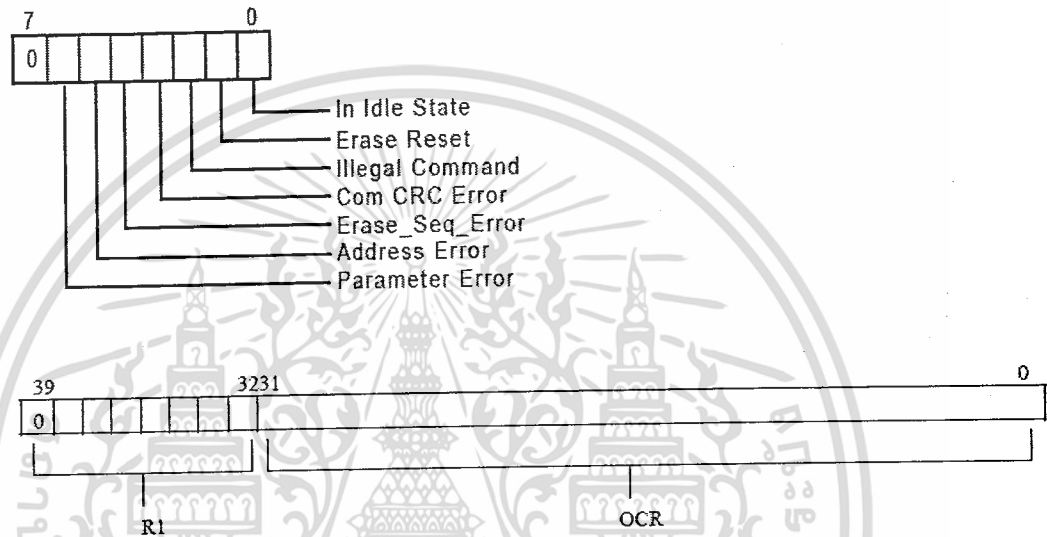
แต่ละ command เป็นการแสดงเครื่องหมายย่อ GO_IDLE_STATE หรือ CMD<n>, โดยที่<n> นั้นเป็นตัวเลขของ command index และค่าของ n เป็นได้ตั้งแต่ 0-63 ซึ่งรูปแบบของตาราง command จะใช้อธิบาย command ที่ใช้ทั่วไป ในการอ่านเขียน และการกำหนดค่าเริ่มต้นของการ์ด

CMD INDEX	SPI Mode	Argument	Resp	Abbreviation	Command Description
CMD26	No				
CMD27	Yes	None	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.
CMD28*	Yes	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).
CMD29*	Yes	[31:0] data address	R1b	CLR_WRITE_PROT	If the card has write protection features, this command clears the write protection bit of the addressed group.
CMD30	Yes	[31:0] write protect data address	R1	SEND_WRITE_PROT	If the card has write protection features, this command asks the card to send the status of the write protection bits. ²
CMD31	Reserved				
CMD32	Yes	[31:0] data address	R1	ERASE_WR_BLK_START_ADDR	Sets the address of the first write block to be erased.
CMD33	Yes	[31:0] data address	R1	ERASE_WR_BLK_END_ADDR	Sets the address of the last write block in a continuous range to be erased.
CMD34 CMD37	Reserved				
CMD38	Yes	[31:0] don't care*	R1b	ERASE	Erases all previously selected write blocks.
CMD39	No				
CMD40	No				
CMD41 ... CMD54	Reserved				
CMD55	Yes	[31:0] stuff bits	R1	APP_CMD	Notifies the card that the next command is an application specific command rather than a standard command.
CMD56	Yes	[31:0] stuff bits [0]: RDAWR ³	R1	GEN_CMD	Used either to transfer a Data Block to the card or to get a Data Block from the card for general purpose/application specific commands. The size of the Data Block is defined with SET_BLOCK_LEN command.
CMD57	Reserved				
CMD58	Yes	None	R3	READ_OCR	Reads the OCR register of a card.
CMD59	Yes	[31:1] don't care* [0:0] CRC option	R1	CRC_ON_OFF	Turns the CRC option on or off. A '1' in the CRC option bit will turn the option on, a '0' will turn it off.
CMD60-63	No				

ตารางที่ 2.4 ตารางแสดง Command ที่ใช้ทั่วไปในการอ่านเขียนและกำหนดค่าเริ่มต้นของการ์ด

2.11.3 SPI Response

รูปแบบของ Response นั้นจะมีอยู่ 3 รูปแบบ คือ R1, R2, R3 ซึ่งจะขึ้นอยู่กับแต่ละ command ที่ใช้นั้นๆ โดยไบต์ของ Response R1 จะเป็นค่ากลับมาของ command ส่วนใหญ่



รูปที่ 2.17 แสดงไบต์ของ R1Response และ R3 Response

ซึ่งแต่ละบิตสำหรับ R1 Response ที่แสดงนั้นถ้าค่าของไบต์ R1 = 0x00 จะหมายความว่าติดต่อกับหน่วยความจำ SD Card สำเร็จแล้ว เมื่อไหร่ก็ตามที่เกิดข้อผิดพลาดขึ้น R1 นั้นจะเซตขึ้นดังที่สอดคล้องกับรูป ค่าของ R3 Response นั้นจะสำหรับ CMD58 เท่านั้น มันจะมีไบต์แรกเหมือนกับ R1 และจะต่อท้ายเป็นขบวนที่บรรจุของ OCR

บาง command จะใช้เวลานานมากกว่าจำนวนของ NCR ที่กำหนด ดังนั้น host คอนโทรลเลอร์นั้นจะรอค่า Response กลับมาด้วยโดยการส่งค่า 0xff ตลอดจนกว่าจะรับ Response กลับมาได้แล้ว

2.12 ขั้นตอนการเซตค่าเริ่มต้นสำหรับ SPI Mode

หลังจาก Power on reset หน่วยความจำ SD นั้นจะเข้าสู่โหมดทำงาน native mode ซึ่งมีรูปแบบการทำงานดังนี้

2.12.1 Power on

หลังจากจ่ายมาถึงค่าแรงดัน 2.2 โวลต์ เราจะต้องเซตค่าของ DI และ CS ให้เป็น 1 ให้มากกว่า 74 พัลส์ ของสัญญาณนาฬิกา ซึ่งจะทำให้การ์ดนั้นสามารถรับ command ต่างๆ ได้

2.12.2 Software Reset

ส่ง CMD0 ร่วมกับ CS เป็น 0 เพื่อที่จะไปรีเซ็ตการ์ด การ์ดนั้นจะไปตรวจสอบสัญญาณ CS เมื่อ CMD ตรวจสอบ ถ้า CS เป็น 0 นั้นการ์ด จะเข้าสู่ SPI Mode ดังนั้น CMD0 จะต้องส่ง native command ด้วยและในส่วนของ CRC นั้นจะต้องมีค่า CRC ที่ถูกต้องด้วย พื้นที่ที่การ์ดเข้าสู่ SPI Mode การตรวจสอบค่า CRC จะไม่พิจารณาอีก ดังนั้นการส่ง CM0 จะกำหนดค่าของ CRC ให้เท่ากับ 0x95 ได้เลยเมื่อ CMD นั้นถูกยอมรับเสร็จการ์ดนั้นจะเข้าสู่สถานะว่าง (idle state) และผลของค่า response R1 จะมีค่าเท่ากับ 0x01 นั่นคือบิตของ in idle state จะถูกเซตขึ้นซึ่งแสดงว่า SD Card นั้นอยู่ในสถานะว่าง

2.12.3 Initialization

ในสถานะว่างเอสดีการ์ดนั้นจะยอมรับเฉพาะ CMD0 , CMD1 , CMD58 เท่านั้น

command อื่นๆ จะถูกปฏิเสธ เมื่อการ์ดนั้นตรวจสอบ CMD1 การ์ดนั้นจะเริ่มพื้มจะทำงาน ซึ่ง host นั้นจะต้องส่ง CMD1 เข้าเรื่อยๆ และต้องคอยตรวจสอบ response พวกนี้จนกว่าค่าของ response R1 จะตอบกลับมาเป็น 0x00 นั่นคือ CMD1 นั้นคือการเคลียร์ค่าของ R1 นั้นเองเพื่อจะทำการส่งค่า Command ต่อไปในช่วงการ initial การ์ดจะใช้เวลาหลายมิลลิวินาทีขึ้นอยู่กับขนาดของการ์ดนั้นๆ ดังนั้นเราจำพิจารณาค่านี้ได้จาก Data Sheet ที่ผู้ผลิตให้มา ซึ่งหลังจากการทำ initial เสร็จก็จะสามารถอ่านและเขียนข้อมูลทั่วไปได้แล้ว ในช่วงเวลานี้จะสามารถอ่านค่ารีจิสเตอร์ OCR และ CID ได้เพื่อที่จะหาย่านของแรงดันที่กระทำและขนาดของการ์ดหรือคุณสมบัติอื่นๆ ที่เราต้องการ

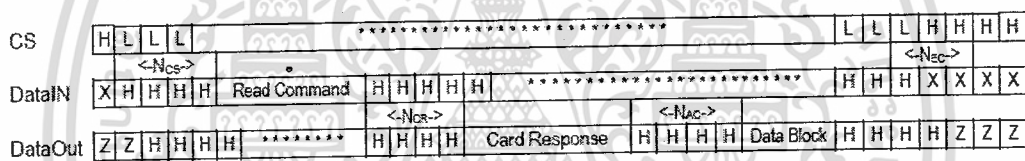
2.12.4 การโอนย้ายข้อมูล

Data packet and Data Response

ในการติดต่อเกี่ยวกับการโอนย้ายข้อมูล 1 block หรือมากกว่า 1 block นั้นจะต้องทำการส่งหรือรับจาก command response ซึ่งการโอนย้ายข้อมูลนั้นจะประกอบไปด้วยกลุ่มของข้อมูลคือ Data Token 1 ไบต์และ CRC อีก 2 ไบต์

2.12.5 การอ่านแบบ single block

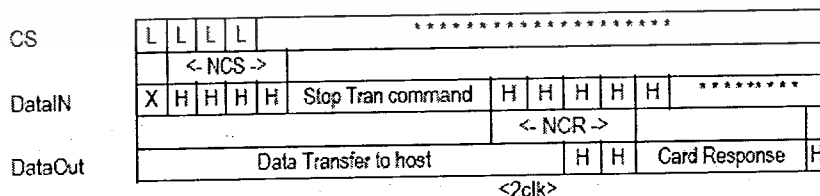
Argument นั้นจะเป็นการระบุตำแหน่งที่จะเริ่มอ่านในหน่วยไบต์เมื่อ CMD17 ถูกยอมรับแล้ว การกระทำของข้อมูล data block จะส่งไป ยัง host หลังจาก Data token ถูกตรวจพบ สิ่งที่ host จะได้รับตามมาก็คือ data block และ CRC 2 ไบต์ โดยปกติขนาดของบล็อกจะเท่ากับ 512 ไบต์ซึ่งเราสามารถเปลี่ยนแปลงได้โดย CMD16 ถ้าเกิดข้อผิดพลาดในระหว่างการอ่านข้อมูล Error token จะถูกส่งมาแทน



รูปที่ 2.18 แสดงการอ่านแบบ Single block

2.12.6 การอ่านแบบ Multiple Block

การอ่านข้อมูลแบบ multiple block เป็นลำดับจากการระบุที่อยู่ เมื่อจำนวนของการโอนย้าย Block จะไม่เจาะจงก่อน command นี้ การติดต่อจะเริ่มตั้งแต่การเปิดและจบที่การอ่าน Multi block การอ่านนั้นจะต่อเนื่องจนกระทั่งหยุดโดย CMD12 ไบต์ที่ได้รับโดยทันทีทันใดนั้นจะตามหลัง CMD12 มันจะถูกตัดทิ้งก่อนได้รับ response ของ CMD12



รูปที่ 2.19 แสดงการอ่านแบบ Multiple Block

2.13 การอ่านคำรีจิสเตอร์ CSD และ CID

การอ่านคำรีจิสเตอร์ทั้ง 2 ตัวนี้จะเหมือนกับการอ่านแบบ single block ยกเว้นสำหรับ ขนาดความยาวของ block CID และ CSD จะถูกส่งไปยัง host เท่ากับ 16 ไบต์ของ data block



บทที่ 3

การออกแบบ

หลักการทำงานของเครื่องเล่น MP3 แบ่งออกเป็น 2 ส่วนใหญ่ๆ คือ

- ส่วน Hardware
- ส่วน Software

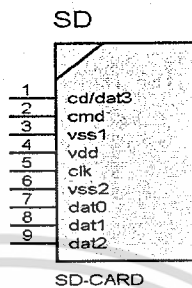
ส่วนการทำงานด้าน Hardware แบ่งได้ 5 ส่วนหลักคือ

1. SD Card
2. Microcontroller PIC18F8722
3. MP3 Decoder VS1002D
4. LCD
5. วงจรภาคจ่ายไฟ

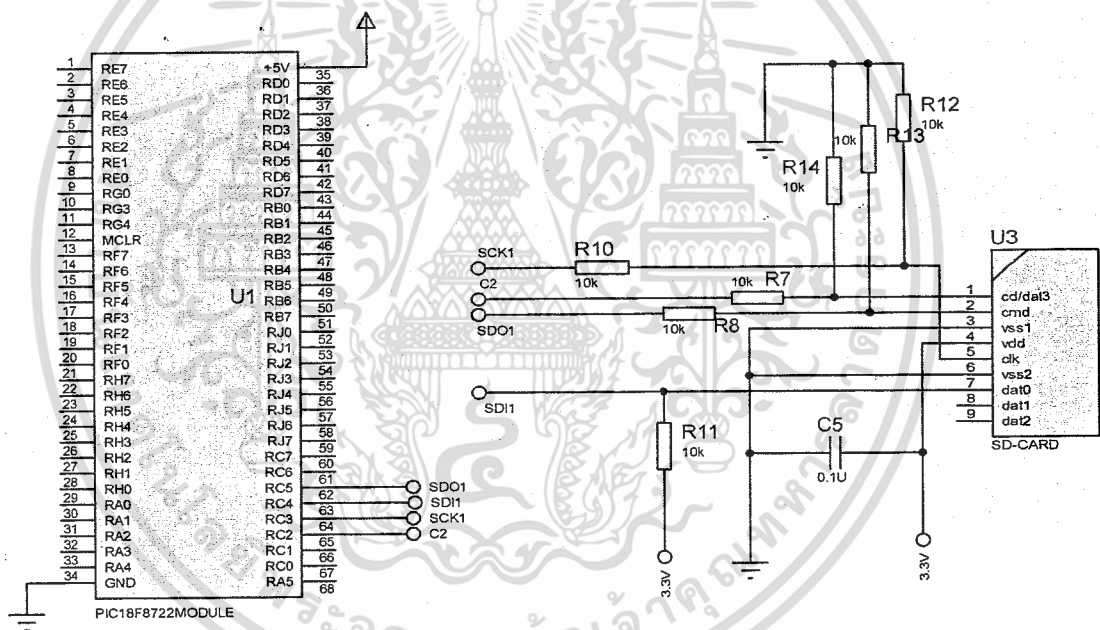


รูปที่ 3.1 บล็อกไดอะแกรมแสดงการทำงานของเครื่องเล่น MP3

3.1 SD Card



รูปที่ 3.2 โครงสร้าง SD Card



รูปที่ 3.3 การเชื่อมต่อระหว่าง microcontroller กับ SD Card

SD card เป็นการ์ดหน่วยความจำชนิดหนึ่งที่ต้องการไฟเลี้ยง 3.3 โวลต์ ซึ่งใน
 โครงการนี้ได้ใช้การติดต่อกับ SD card ผ่านอินเตอร์เฟสแบบ SPI ซึ่งในโหมดนี้ SD card จะมีขาใช้ในการ
 การติดต่อกับไมโครคอนโทรลเลอร์ทั้งหมด 4 ขา คือ CS, MISO, MOSI, CLK แต่ละขามีหน้าที่
 ดังนี้

1. CS (Chip Select) เป็นขาที่ใช้สำหรับเลือกการ์ดให้อยู่ในสถานะพร้อมใช้งานโดยจะเป็นการเลือกการ์ดนั้น ๆ หากมีสถานะเป็น “ Low ” หากเป็น “ Hi ” การ์ดจะไม่ถูกเลือกและการ์ดจะไม่ทำงาน

2. MISO (Master In Slave Out) เป็นขาที่ไม่โครคอนโทรลเลอร์ใช้ให้การรับข้อมูลที่มาจากการ์ด โดยการส่งข้อมูลจะถูกควบคุมด้วย Clock

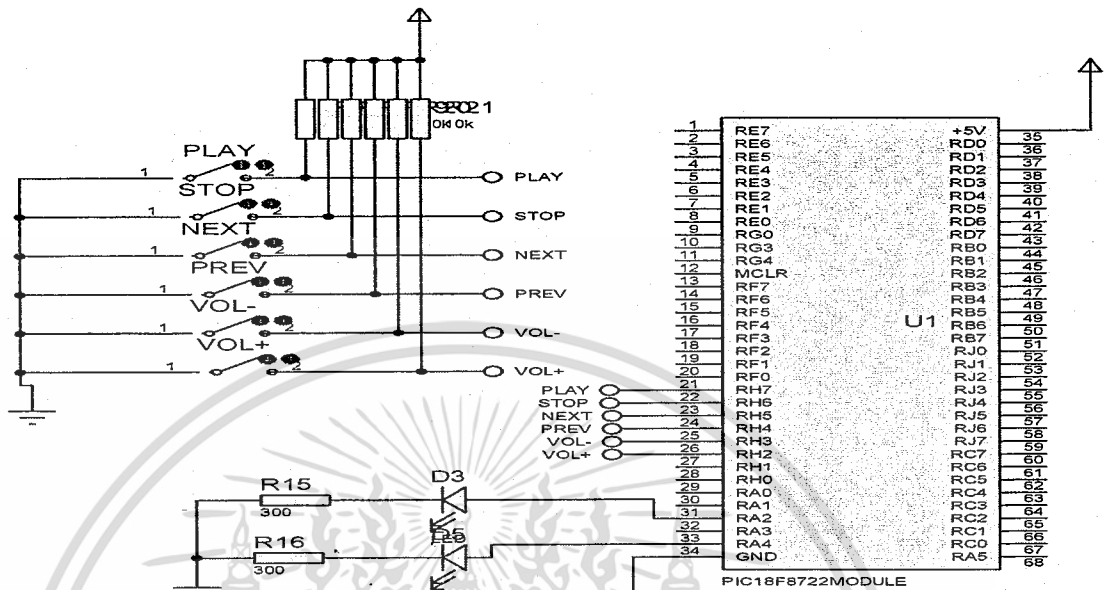
3. MOSI (Master Out Slave In) เป็นขาที่ไม่โครคอนโทรลเลอร์ใช้ในการส่งข้อมูลและเป็นขาที่การ์ดใช้ในการรับข้อมูล โดยการส่งข้อมูลจะถูกควบคุมด้วย Clock

4. CLK (Clock) เป็นขาที่ใช้ควบคุมการรับส่งข้อมูล

เลือกใช้ SD Card สำหรับเป็น Memory ในการเก็บข้อมูลเนื่องจากมีความเหมาะสมทางเรื่องราคา เป็น Memory Card ที่หาซื้อได้ง่าย รวมทั้งในปัจจุบันนี้มีเครื่องใช้อิเล็กทรอนิกส์หลายชนิดใช้หน่วยความจำเป็น SD Card ทำให้สะดวกในการใช้และพกพา

3.2 Microcontroller PIC18F8722

เลือกใช้ไมโครคอนโทรลเลอร์ที่มีการประมวลผลที่รวดเร็ว รองรับการใช้พอร์ตติดต่อในรูปแบบ SPI BUS ที่นำมาใช้ในการติดต่อกับ SD Card และ Decoder VS1002D มีขนาด RAM ที่เพียงพอ และมีจำนวนพอร์ต Input และ Output ที่เพียงพอ จึงเลือกใช้ PIC 18F8722

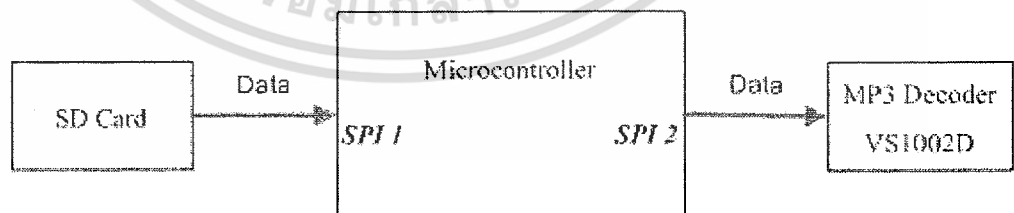


รูปที่ 3.4 การเชื่อมต่อระหว่าง microcontroller กับ SW ควบคุมการเล่นเพลง

ส่วนของ SW เพื่อใช้ในการควบคุมการเล่นเพลง มีทั้งหมด 6 อินพุต คือ PLAY , STOP , NEXT , PREVIOUS , VOLUME_UP และ VOLUME_DOWN โดยจะมีการใช้ port RH7 ถึง RH2 ตามลำดับ ซึ่งจากการต่อจะเห็นได้ว่า SW control ทั้งหมด จะมีการทำงานแบบ Active low เนื่องจากต่อลง GND

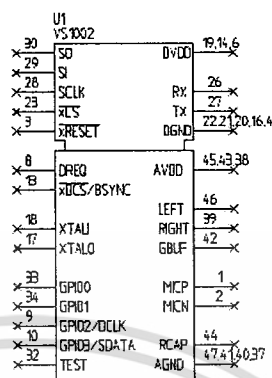
3.2.1 การติดต่อระหว่างไมโครคอนโทรลเลอร์, SD Card และ ตัวถอดรหัส MP3

การติดต่อโดยใช้ SPI 2 พอร์ต



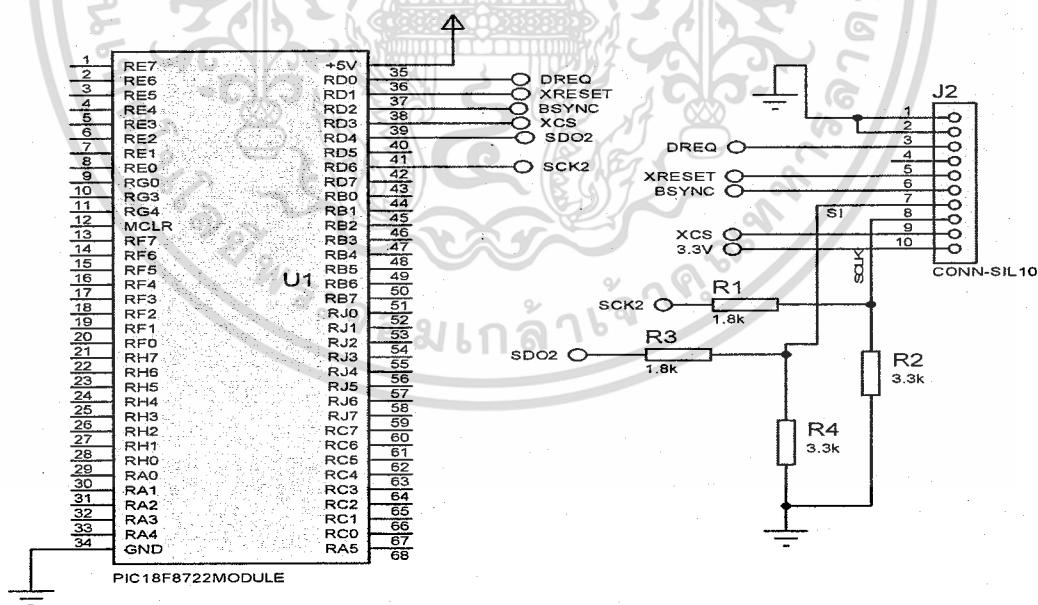
รูปที่ 3.5 การติดต่อกันของ Microcontroller กับ SD Card และ VS1002D โดยใช้รูปแบบ SPI 2 Port

รูปที่ 3.6 วงจร MP3 Decoder โดย VS1002D



รูปที่ 3.7 โครงสร้าง VS1002D

เนื่องจากไอซีถอดรหัสไฟล์ MP3 ที่มีคุณภาพดี และใช้งานง่ายมาก โดย VS1002D สามารถถอดรหัสไฟล์ MP3 และ WAVE ได้ทันที พร้อมทั้งมีภาค Output แบบ Analog เป็นสัญญาณเสียงแบบ Stereo และเนื่องจากมีวงจรขับลำโพงอยู่ในตัวจึงสามารถนำไปต่อกับชุดหูฟัง หรือชุดขยายเสียงได้ทันที แล้วสำหรับการเชื่อมต่อกับไมโครคอนโทรลเลอร์นั้นก็สามารทำได้โดยง่าย โดยการเชื่อมต่อกับพอร์ตอนุกรมแบบ SPI มาตรฐานได้ทันที



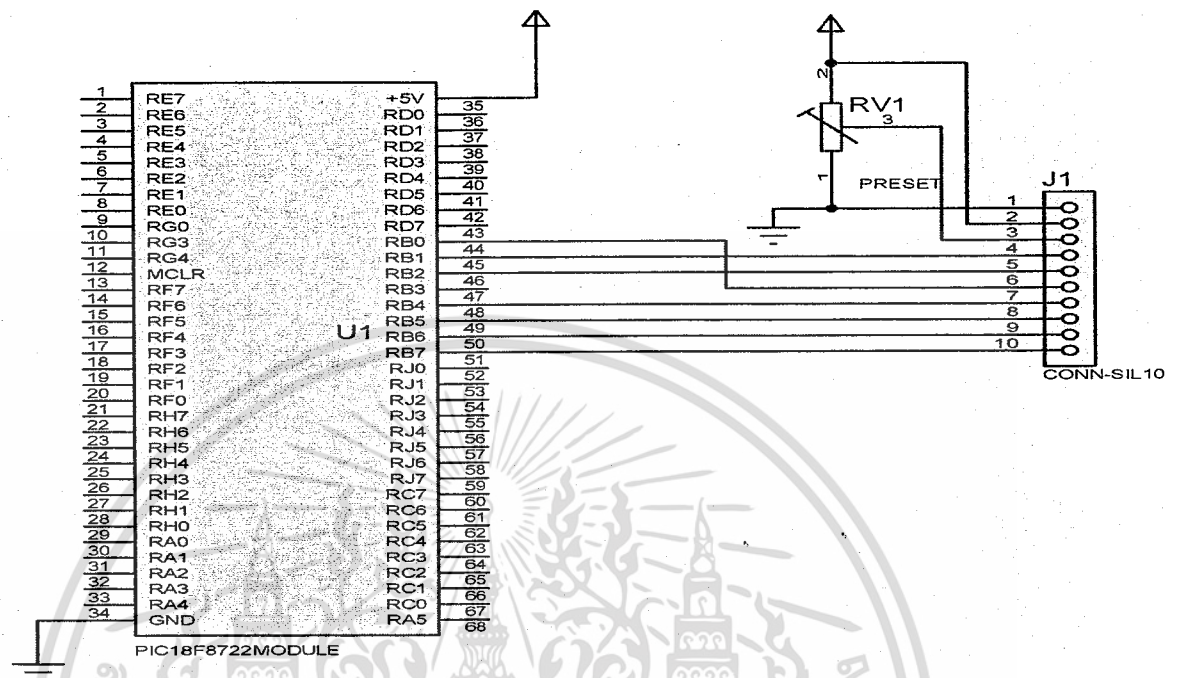
รูปที่ 3.8 การเชื่อมต่อระหว่าง microcontroller กับ MP3 Decoder VS 1002D

ส่วนการเชื่อมต่อกับ MP3 Decoder VS1002D ใช้ทั้งหมด 6 ขา คือ XRESET , XCS , BSYNC , DREQ , SCK2 และ SDO2 โดยจะมีการใช้ port RD0 , RD1 , RD2 , RD3 , RD4 และ RD6 ตามลำดับ ซึ่ง RD4 และ RD6 เป็นพอร์ต SPI2 ของ microcontroller

1. ขา DREQ เป็นขา แสดงการร้องขอข้อมูลของ VS1002D ถ้า เป็น “1” จะมีการเรียกข้อมูลเกิดขึ้น
2. ขา XCS เป็นให้สัญญาณว่าจะอ่านหรือ เขียนข้อมูลและจะทำงานที่ SCI โดยหากให้สัญญาณ XCS เป็น “0” แสดงให้ SCI ทำงานอ่านหรือเขียน และเมื่อค่าเป็น “1” เพื่อจบกระบวนการทำงาน
3. ขา XRESET เป็นขาที่ใช้สำหรับการรีเซ็ตตัว MP3 Decoder เมื่อป้อน “0”
4. ขา SCK2 หรือ SCLK เป็นสัญญาณนาฬิกาของข้อมูลอนุกรม จะถูกใช้เหมือนเป็นสัญญาณนาฬิกาหลักสำหรับอุปกรณ์อินเตอร์เฟส
5. ขา SDO2 หรือ SI เป็นขาของข้อมูลอินพุตแบบอนุกรม ซึ่งในกรณีที่ XCS เป็น “0” SI จะส่งเอาข้อมูลที่ขอมาขึ้นของสัญญาณ SCK
6. ขา BSYNC เมื่อขอมาขึ้นของ BSYNC ทำให้ SDI มีการส่งข้อมูล

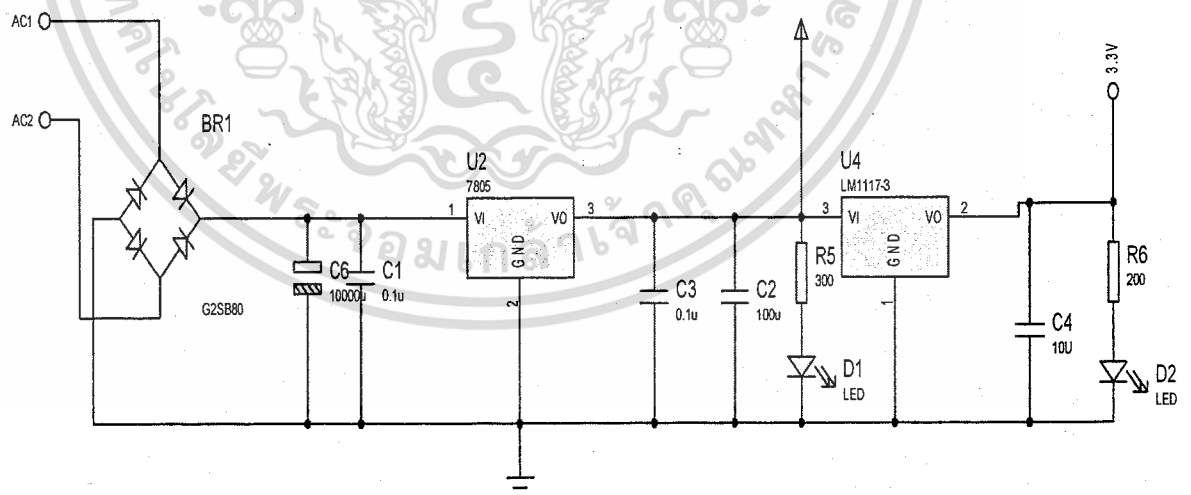
3.4 LCD

มีการนำ จอแสดงผล LCD 16 x 2 คือ จอแสดงผลที่สามารถแสดงตัวอักษรได้ 16 ตัวอักษร 2 บรรทัด เพื่อไว้ใช้ในการแสดงชื่อเพลงที่กำลังเล่นอยู่ การเชื่อมต่อระหว่าง microcontroller กับ LCD จะใช้พอร์ต RB0 ถึง RB7



รูปที่ 3.9 การเชื่อมต่อระหว่าง microcontroller กับ LCD

3.5 วงจรภาคจ่ายไฟ 5 v. และ 3.3 v.



รูปที่ 3.10 แสดงวงจรภาคจ่ายไฟ

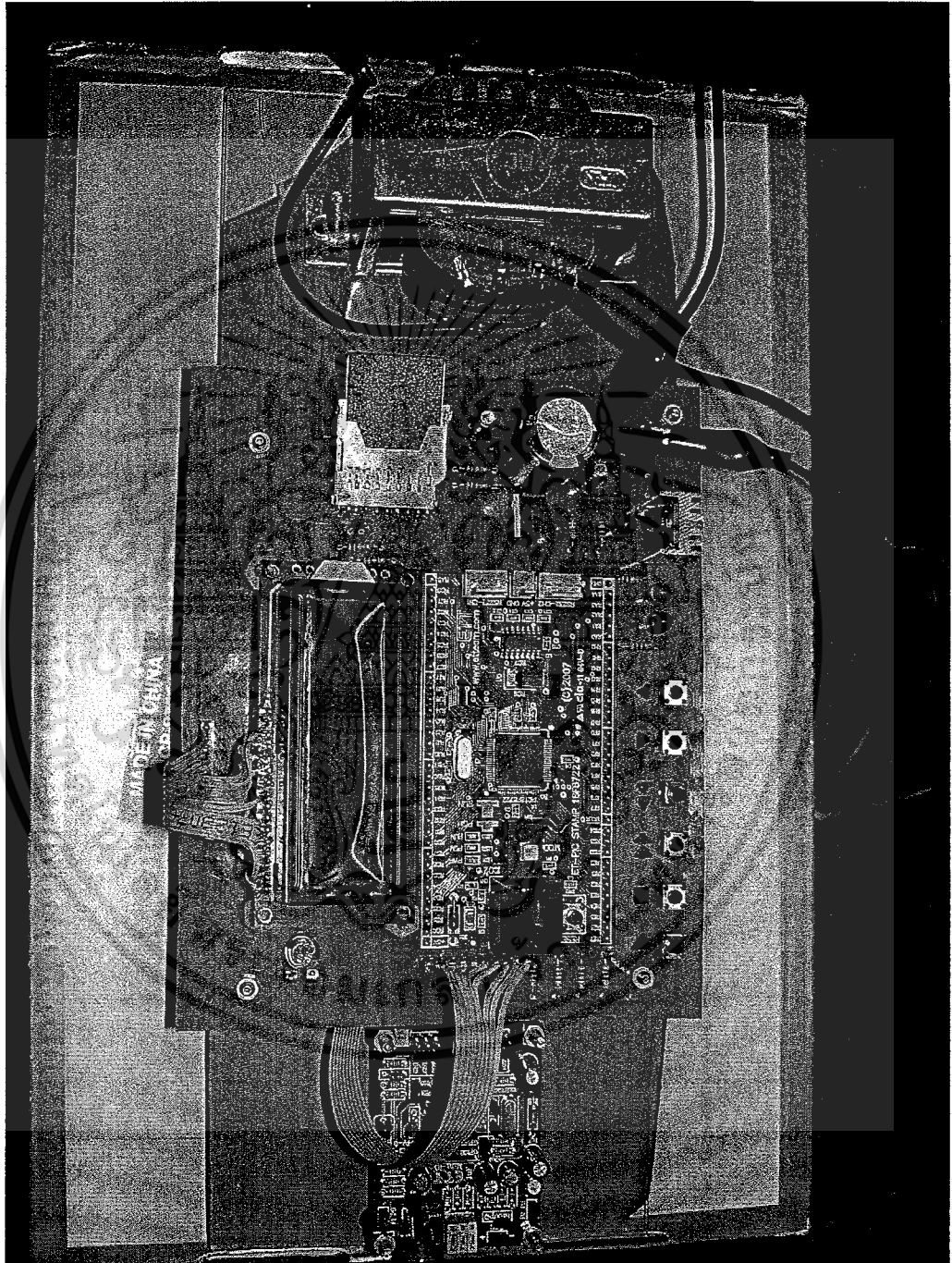
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทำงานของวงจรเริ่มจากแรงดันไฟฟ้ากระแสตรงขนาด 9-12 Volt

- BR1 เป็นวงจรบริดจ์สำหรับเรียงกระแสจากหม้อแปลง
- 7805 จะเป็นตัวลดระดับแรงดันให้เหลือ 5 Volt เพราะเนื่องจากอุปกรณ์ที่ใช้ในวงจรทำงานแรงดันดังกล่าว เช่น LCD และ PIC18f8722
- LM1117T3.3 จะทำหน้าที่ลดระดับแรงดันจาก 5 Volt ให้เหลือ 3.3 Volt เพื่อใช้กับ SD/MMC card และ Decoder VS1002D



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



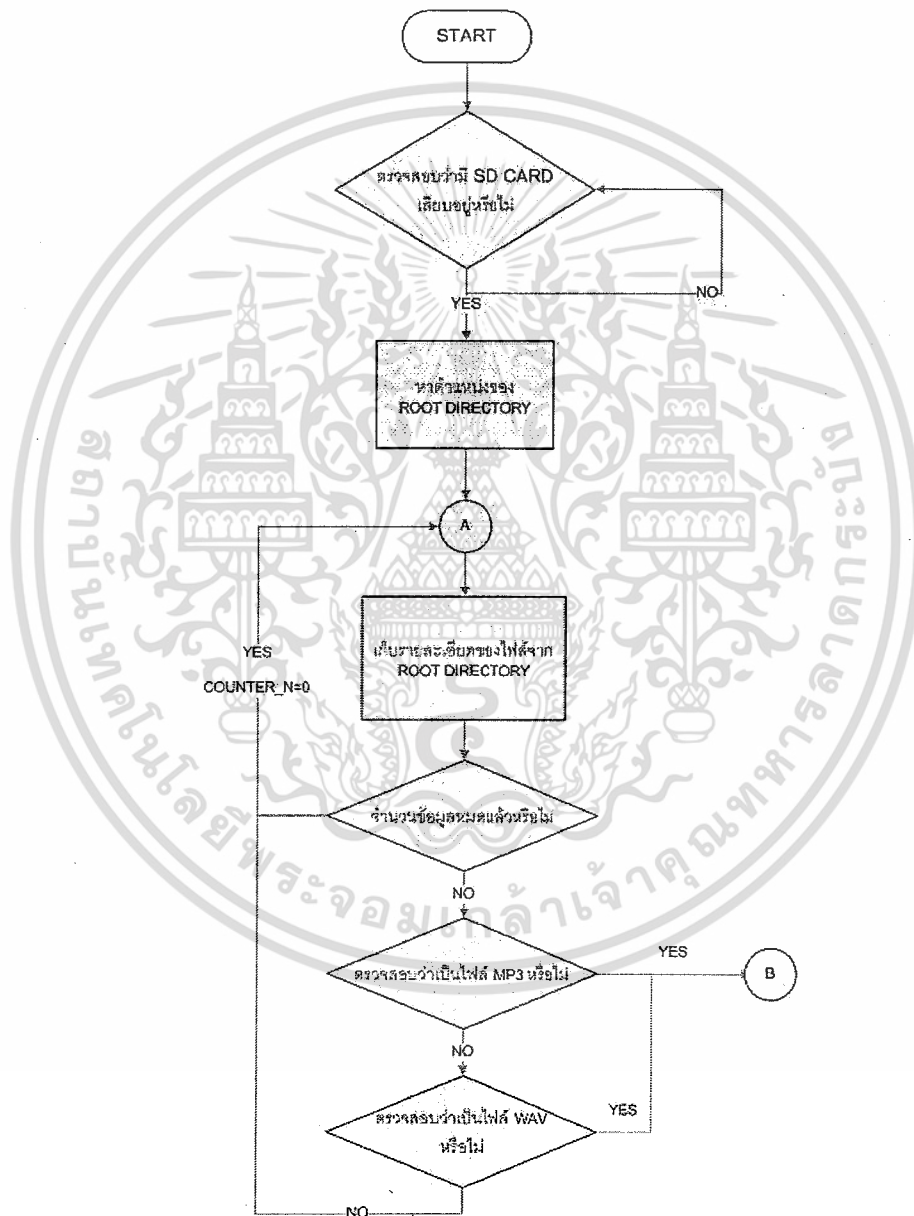
รูปที่ 3.11 ภาพ Model เครื่องเล่น MP3 แบบ SD Card

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ส่วนหลักการทำงานด้าน Software จะแสดงเป็น Flow chart ดังรูปด้านล่าง

3.6 Flow chart

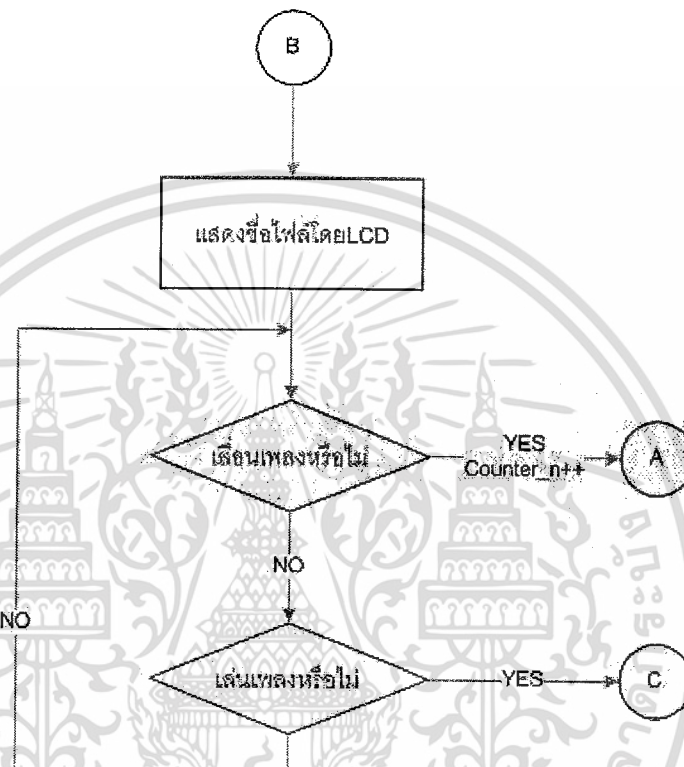
3.6.1 ภาค SD Card



รูปที่ 3.12 Flow Chart การทำงานของภาค SD Card

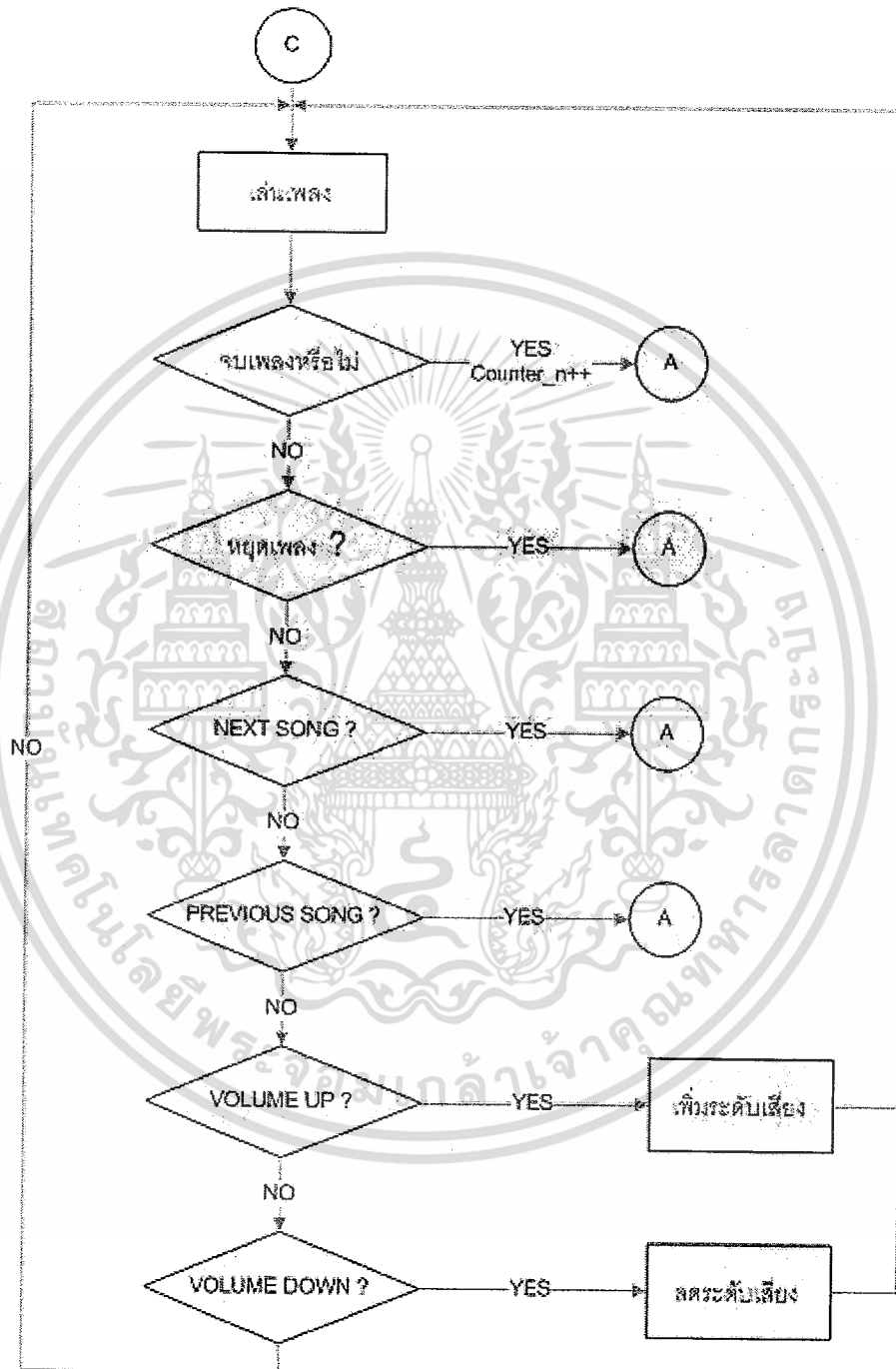
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.6.2 ภาค LCD



รูปที่ 3.13 Flow Chart การทำงานของภาค LCD

3.6.3 ภาค SWITCH ความคุมทั้ง 6 ขา



รูปที่ 3.14 Flow Chart การทำงานของภาค SWITCH ความคุมทั้ง 6 ขา

บทที่ 4

ผลการทดลอง

การทดลองแบ่งออกเป็น 2 ส่วน คือ

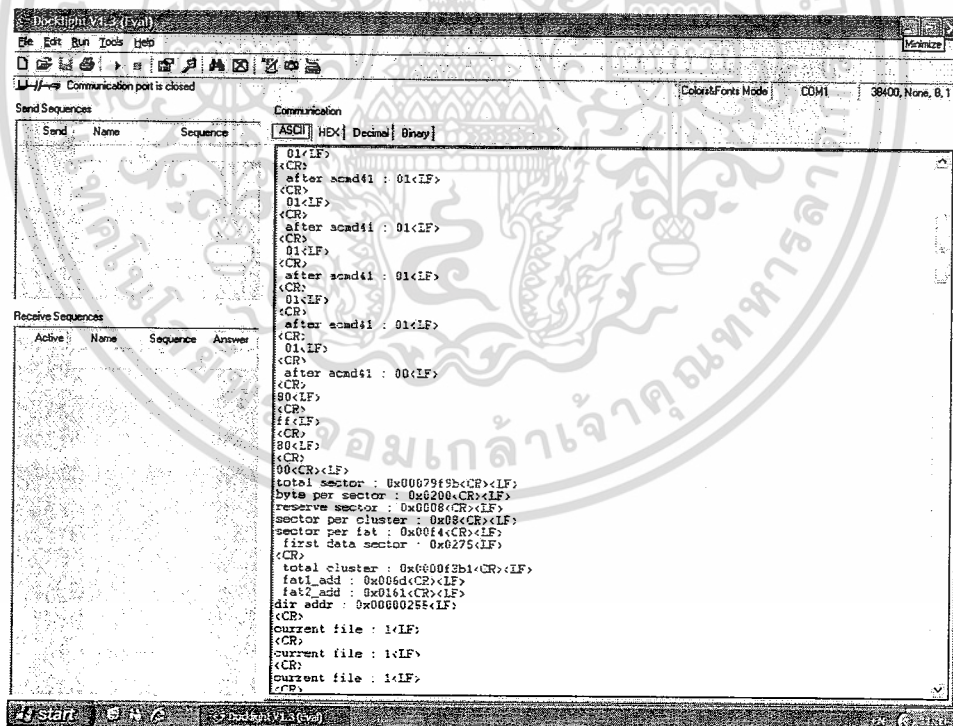
1. ส่วน Software
2. ส่วน Hardware

****หมายเหตุ current file คือ file ที่ microcontroller อ่านค่ามาจาก SD Card ขณะนั้นๆ****

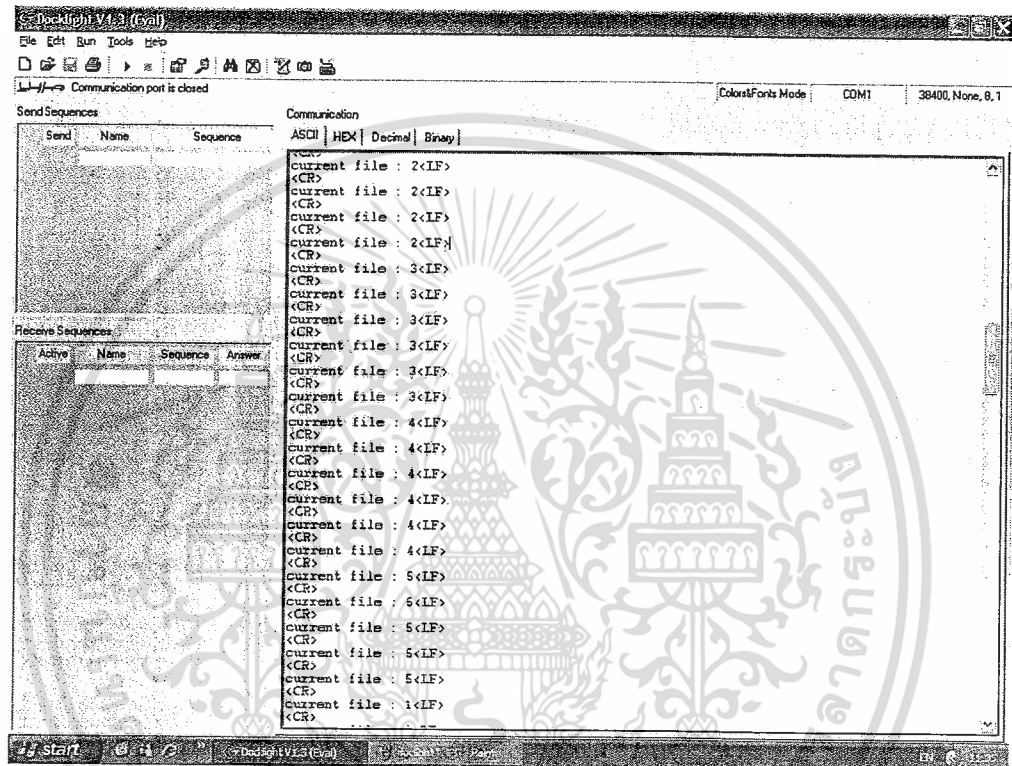
4.1 ส่วนผลการทดลองของ Software

ส่วนการทดลองของ Software จะมีการใช้โปรแกรม Docklight ในการตรวจสอบการทำงานของไมโครคอนโทรลเลอร์ซึ่งจะใช้พอร์ท RS232 ในการติดต่อกับคอมพิวเตอร์

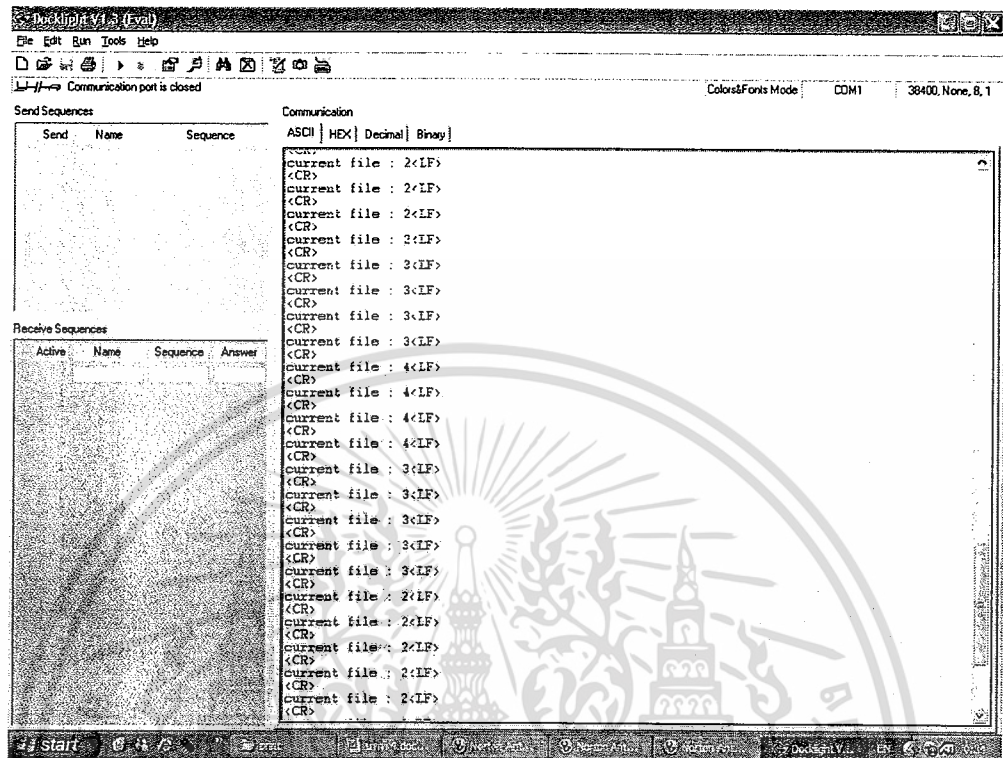
4.1.1 การทดลองเพื่อทดสอบว่า microcontroller สามารถติดต่อกับ SD Card ได้หรือไม่



รูปที่ 4.1 แสดงผลการเบิร์นโค้ดลงไมโครคอนโทรลเลอร์ ว่าโปรแกรมสามารถทำให้ไมโครคอนโทรลเลอร์เชื่อมต่อกับ SD Card ได้



รูปที่ 4.2 แสดงผลการเบิร์นโค้ดลงไมโครคอนโทรลเลอร์ว่าสามารถเลื่อนเพลงเป็นเพลงถัดไปและวนลูปเพลงได้



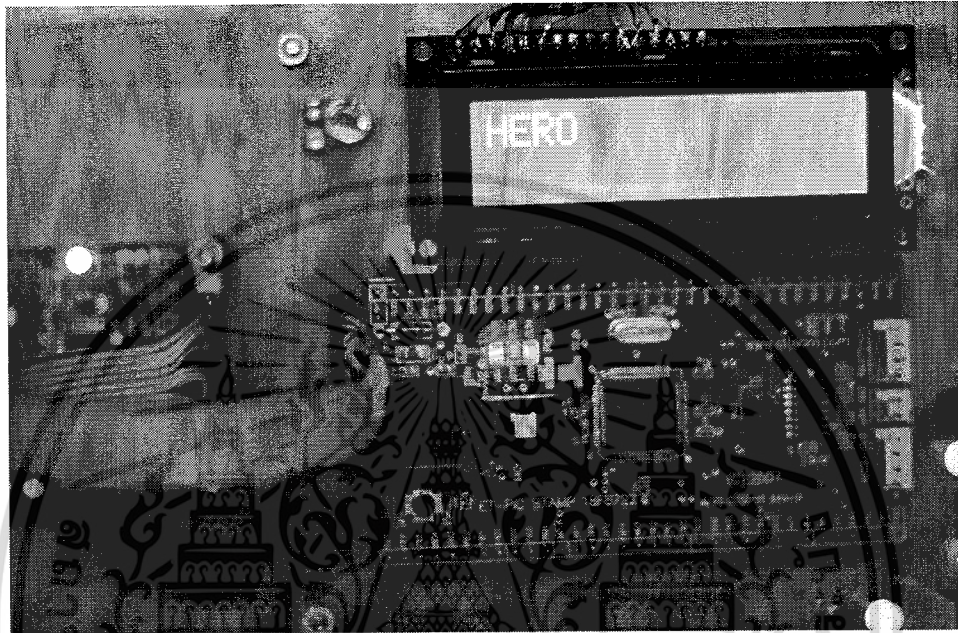
รูปที่ 4.3 แสดงผลการเบิร์น โค้ดลงไมโครคอนโทรลเลอร์ ว่าสามารถเลื่อนเพลงเป็นเพลงถัดไป และเพลงก่อนหน้าได้

จากรูปที่ 4.2 และ รูปที่ 4.3 แสดงให้เห็นว่าการเขียน โปรแกรมให้กับไมโครคอนโทรลเลอร์ใช้ Switchควบคุมการเลื่อนเพลงให้เล่นเพลงถัดไปและเพลงก่อนหน้าได้รวมทั้งสามารถให้เล่นแบบวนลูบเพลงได้ด้วย ซึ่งจะสังเกตได้จาก current file ในรูปจะมีค่าเปลี่ยนไปนั่นคือค่าเมื่อกดปุ่ม NEXT ค่า current file ก็จะเพิ่ม เช่น จาก 3 เปลี่ยนเป็น 4 เป็นต้น ส่วนการเลื่อนเพลงก่อนหน้าเมื่อกดปุ่ม PREV ค่า current file ก็จะลดลง เช่น จาก 4 เปลี่ยนเป็น 3 เป็นต้น

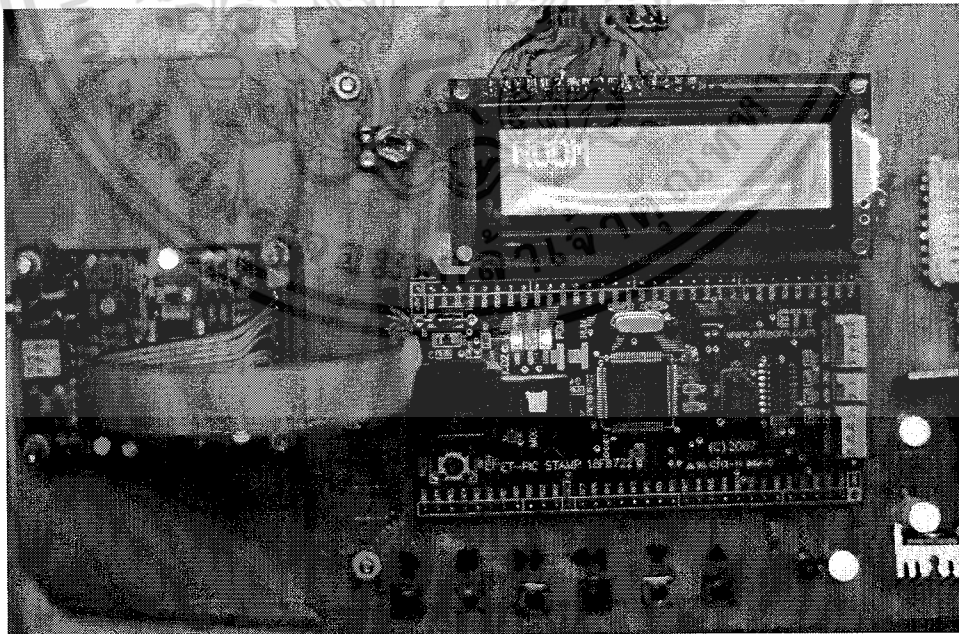
การวนลูบเพลงเห็นได้จาก current file มีการเปลี่ยนจาก file ที่ 5 เป็น 1 ซึ่งในการทดลองนี้ผู้ทำการทดลองได้ใส่เพลงไว้ทั้งหมด 5 เพลง

สำหรับการทดลองใช้ SW ควบคุมนี้เป็นการอ่านชื่อไฟล์เพลงใน SD Card มาเพื่อให้เห็นผลลงบนหน้าจอ LCD ซึ่งสามารถดูผลการทดลองได้ในส่วนของ Hardware ต่อไป

4.2 ส่วนผลการทดลองของ Hardware

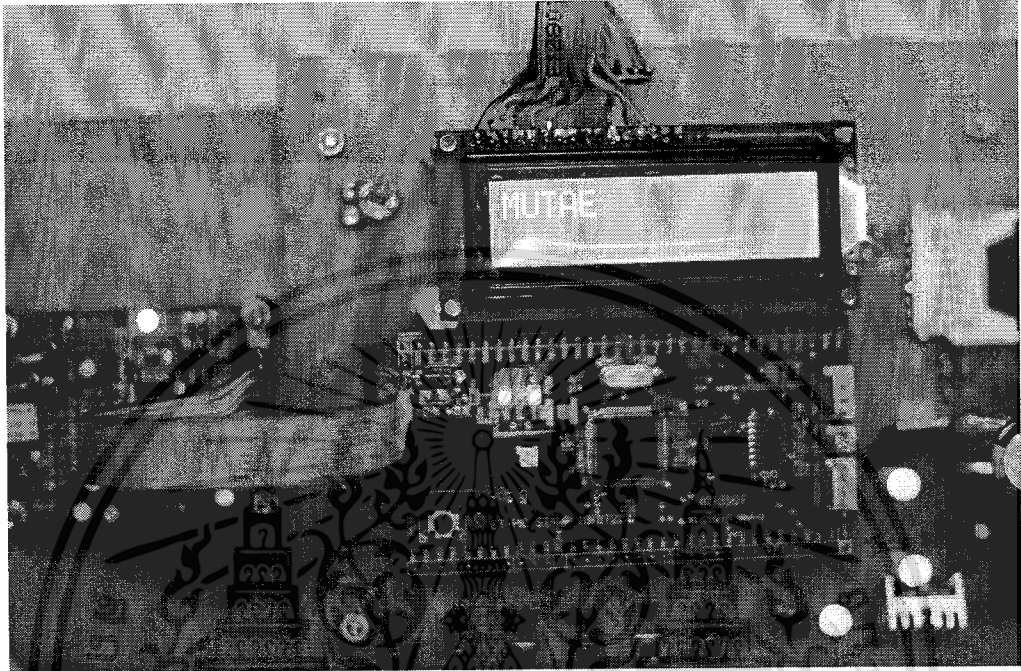


รูปที่ 4.4 ภาพการแสดงผลชื่อเพลงบนจอ LCD ของค่า current file ที่ 1



รูปที่ 4.5 ภาพการแสดงผลชื่อเพลงบนจอ LCD ของค่า current file ที่ 2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.6 ภาพการแสดงชื่อเพลงบนจอ LCD ของค่า current file ที่ 3

จากรูปที่ 4.4 ถึง 4.6 แสดงภาพการแสดงชื่อเพลงบนจอ LCD ของค่า current file ที่ค่าต่างๆ โดยการเลื่อนเพลงใช้วิธีกดปุ่ม NEXT และ PREV เห็นได้ว่าชื่อเพลงก็เปลี่ยนไปตามค่า current file ตามลำดับ

บทที่ 5

สรุปผลการทดลอง

5.1สรุปและวิจารณ์

ในปริญญานิพนธ์ฉบับนี้ ได้นำเสนอรายละเอียดการสร้างเครื่องเล่นไฟล์เสียง MP3 โดยใช้ SD Card เป็นอุปกรณ์ที่ใช้เก็บข้อมูลไฟล์เสียง โดยได้ศึกษารูปแบบมาตรฐานการบันทึกข้อมูลของ SD Card จากนั้นศึกษาการนำไมโครคอนโทรลเลอร์ตระกูล PIC (Module PIC18F8722) มาใช้อ่านไฟล์เสียง MP3 ตามมาตรฐานจาก SD Card เพื่อส่งไปถอดรหัสให้เป็นเสียงที่รับฟังได้โดยตัว Decoder (Module VS1002D) ซึ่งมีระบบขยายสัญญาณในตัว สามารถต่อออกไปยังเครื่องขยายเสียงหรือหูฟังได้ทันที รวมทั้งได้แสดงรูปวงจร และ โปรแกรมไว้ที่ภาคผนวกของปริญญานิพนธ์ฉบับนี้

ตามที่ได้ออกแบบไว้เครื่องเล่น MP3 จาก SD Card มีฟังก์ชันดังนี้

- 1.สามารถกดเลื่อนเพลงได้ คือ NEXT และ PREVIOUS
- 2.สามารถกดเล่นและหยุดได้ คือ PLAY และ STOP
- 3.สามารถปรับระดับความดังของเสียงได้ คือ VOL+ และ VOL-

แต่ทางกลุ่มของข้าพเจ้าทำได้ถึงเพียงข้อที่1คือสามารถใช้ฟังก์ชัน NEXT และ PREVIOUS ได้

เพราะเวลาที่มีจำกัด เนื่องจากกลุ่มข้าพเจ้าได้ตัดสินใจเปลี่ยนโครงงานเข้าไปซึ่งเหลือเวลาเพียง1 เดือน และความไม่คุ้นเคยในการใช้ตัว Decoder (Module VS1002D) ทำให้ไม่สามารถบรรลุจุดประสงค์ได้ ซึ่งจะได้มีการพัฒนาโครงงานนี้ต่อไปในอนาคต ให้สามารถใช้ฟังก์ชันได้อย่างสมบูรณ์

5.2ประโยชน์ที่ได้รับ

- 1.เข้าใจการทำงานของไมโครคอนโทรลเลอร์ตระกูล PIC
- 2.เข้าใจการติดต่อกับSD Card ผ่านโหมดต่างๆ

3.ทราบทฤษฎีต่างๆเช่น MP3 FAT

4.ได้ทบทวนความรู้การสร้าง Power Supply

5.ที่สำคัญที่สุดคือได้ฝึกการทำงานเป็นกลุ่มและได้เห็นความมีน้ำใจของเพื่อนๆกลุ่ม

อื่น



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Code Main (สามารถใช้งานได้ตามที่ผลการทดลองออกใช้กับ mmc.c) คือ สามารถ
เชื่อมต่อกับ SD Card ได้ แสดงชื่อเพลงลงบนจอ LCD และสามารถกดปุ่มเลื่อนเพลงได้

```
#include "C:\Documents and Settings\Administrator\Desktop\k project\sd.h"
//#include "SD_CARD.c"
#include "C:\Documents and Settings\Administrator\Desktop\new2\mmc.c"
#define use_portb_lcd
#include "LCD.c"
//#include <18f8722>
#include "Board.c"
//#include "VS1002.c"

#define PLAY PIN_H7
#define STOP PIN_H6
#define NEXT PIN_H5
#define PREV PIN_H4
#define VOL_DOWN PIN_H3
#define VOL_UP PIN_H2

int32 total_sector,total_cluster,dir_addr;
int16 byte_sector,sector_fat,reserve,first_sector,fat1_addr,fat2_addr,first_data;
int8 sector_cluster;

//////////////////// GET MBR INFO //////////////////////

#separate

int get_mbr (void)
{
    int8 data[512];
    int32 i;
    i=0;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

while(1){
    read_block(i*512,data);
    //fprintf(com,"\n\r %x %x %x",i data[0] data[2]);
    if((data[0]==0xeb)&&(data[2]==0x90)){
        //fprintf(com,"\n\raddress of MBR is %lx",i);
        first_sector=i;
        break;
    }
    i+=1;
    if(i>5000) return(0);
}
if((data[0x13]+data[0x14])==0)
    total_sector=(make32(data[0x23],data[0x22],data[0x21],data[0x20]));
if((data[0x13]+data[0x14])!=0)
    total_sector=(int32)make16(data[0x14],data[0x13]);
printf("\r\ntotal sector : 0x%lx",total_sector);

byte_sector=make16(data[0x0c],data[0x0b]);
printf("\r\nbyte per sector : 0x%lx",byte_sector);

reserve=make16(data[0x0f],data[0x0e]);
printf("\r\nreserve sector : 0x%lx",reserve);

sector_cluster=data[0x0d];
printf("\r\nsector per cluster : 0x%lx",sector_cluster);

sector_fat=make16(data[0x17],data[0x16]);
printf("\r\nsector per fat : 0x%lx",sector_fat);

first_data=first_sector+sector_fat*2+reserve+32;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

printf("\r\n first data sector : 0x%lx",first_data);

total_cluster=(total_sector-(first_data-first_sector))/sector_cluster;

printf("\r\n total cluster : 0x%lx",total_cluster);

fat1_addr=first_sector+reserve;

printf("\r\n fat1_add : 0x%lx",fat1_addr);

fat2_addr=fat1_addr+sector_fat;

printf("\r\n fat2_add : 0x%lx",fat2_addr);

dir_addr=fat2_addr+sector_fat;

printf("\r\n dir addr : 0x%lx",dir_addr);

return(1);
}

////////////////////write_block////////////////////////////////////
#separate
int write_block(int32 address,char *buff)
{
    long i;
    int8 r1;
    r1=mmc_cmd(0x58,address,16);
    //printf("\n\r%x",r1);
    if (r1==0x00) goto send_token ; // we can send data payload
    if (r1==0x40) goto invalid;
    return(false);

invalid:  printf("\n\r write block err %2X\n\r",r1);
    return(false);
    //printf("\n\r command right");

send_token:    SPI_READ(0xFE);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

for (i=0;i < 512;i++){
    SPI_READ(buff[i]); /// send payload
}

SPI_READ(0xFF); // send dummy checksum
SPI_READ(0xFF);
r1=SPI_READ(0xFF);
for( i=0;i<0xffff;i++)
{
    r1=SPI_READ(0xFF);// digest prior operation
    //printf(" %x",r1);
    //if(i%32==0) printf("\n\r");
    if(r1!=0x00) break;
}
//printf("\n\r write complete");
}

#define msg0 "***MP3**"

void main(){

unsigned char volume_values[] = {254, 76, 72, 68, 64, 60, 56, 52, 48, 44, 40, 36, 32, 28, 24,
20, 16, 12, 8, 4, 0};

int32 sound_pointer, root_addr, start_addr, file_size, root_start;
int16 i, sec_total, sec_num, rsv_size, fat_size, byte_per_sec,current_file=0,current_dir=0;
int counter_p, counter_n, file_ad[4],vol_pos, fat, root_ad[4];

int8 buff[512];

int1 play_en,direction=1;
unsigned char data[512];

char str[8], Detail[512]; //file name byte 0-7, extension byte 8-10

//start cluster high 2 bytes : byte 21,20

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

//start cluster low 2 bytes : byte 27,26

//file size : byte 28-31

#bit CKE = 0xFC7.6

setup_adc_ports(NO_ANALOGS|VSS_VDD);

setup_adc(ADC_OFF|ADC_TAD_MUL_0);

setup_psp(PSP_DISABLED);

setup_spi(SPI_MASTER|SPI_H_TO_L|SPI_XMIT_L_TO_H|SPI_CLK_DIV_4);

setup_spi2(SPI_MASTER|SPI_H_TO_L|SPI_CLK_DIV_16);

setup_wdt(WDT_OFF);

setup_timer_0(RTCC_INTERNAL);

setup_timer_1(T1_DISABLED);

setup_timer_2(T2_DISABLED,0,1);

setup_timer_4(T4_DISABLED,0,1);

setup_comparator(NC_NC_NC_NC);

setup_vref(FALSE);

printf("crazy tok");

output_high(PIN_A2);

delay_ms(200);

output_low(PIN_A2);

delay_ms(200);

output_high(PIN_A2);

delay_ms(200);

output_low(PIN_A2);

delay_ms(200);

output_high(PIN_A4);

delay_ms(200);

output_low(PIN_A4);

delay_ms(200);

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
output_high(PIN_A4);  
delay_ms(200);  
output_low(PIN_A4);  
delay_ms(200);
```

```
lcd_init();  
output_high(PIN_A4);  
delay_ms(500);  
LCD_Command(0x01); //clear LCD  
lcd_putc("\f");  
delay_ms(100);  
lcd_putc("k");  
delay_ms(100);  
lcd_putc("a");  
delay_ms(100);  
lcd_putc("r");  
delay_ms(100);  
lcd_putc("a");  
delay_ms(100);  
lcd_putc("o");  
delay_ms(100);  
lcd_putc("k");  
delay_ms(100);  
lcd_putc("e");  
delay_ms(100);  
lcd_putc("n");  
delay_ms(100);
```

```
init(255);
```

```
get_mbr();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    read_block((dir_addr)*512,buff);
while(1)
{
    if(!input(NEXT))
    {
        current_file++;
        direction=1;
        while(!input(NEXT));
    }
    else if(!input(PREV))
    {
        if(current_file!=0)
        {
            current_file--;
            direction=0;
        }
        while(!input(PREV));
    }
    if(((buff[(current_file%16)*32]!=0xE5)||((buff[(current_file%16)*32]!=0x00))&&
        (((buff[(current_file%16)*32+ 0x08]=='M')&&(buff[(current_file%16)*32+
0x09]=='P'))&&
        (buff[(current_file%16)*32+ 0x0A]=='3'))))
    {
        //printf("\n\ryooohoo");
        lcd_command(0x01);
        for(i=0;i<8;i++)
        {
            lcd_putc(buff[(current_file%16)*32+i]);
        }

        printf("\n\rcurrent file : %lu",current_file);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        lcd_putc(".MP3");
        delay_ms(500);
    }
    else if(direction=1)
    {
        current_file++;
        if(current_file==512)
        {
            current_file=0;
        }
        if(current_file%16==0)
        {
            read_block((dir_addr+current_file/16)*512,buff);
        }
    }
    else
    {
        if(current_file!=0)
        {
            current_file--;
        }
        else if(current_file%16==0)
        {
            current_file--;
            read_block((dir_addr+current_file/16)*512,buff);
        }
    }
}
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

VS1002D.c Code ของ Decoder VS1002D

```
#ifndef _VS1011B_
#define _VS1011B_
#include <avr/delay.h>
#define delay(clock=20000000)

#define VS_WRITE_COMMAND 0x02
#define SCI_MODE          0x0
#define SCI_STATUS        0x1
#define SCI_BASS          0x2
#define SCI_CLOCKF        0x3
#define SCI_DECODE_TIME   0x4
#define SCI_AUDATA        0x5
#define SCI_WRAM          0x6
#define SCI_WRAMADDR      0x7
#define SCI_HDAT0         0x8
#define SCI_HDAT1         0x9
#define SCI_AIADDR        0xa
#define SCI_VOL           0xb
#define SCI_AICTRL0       0xc
#define SCI_AICTRL1       0xd
#define SCI_AICTRL2       0xe
#define SCI_AICTRL3       0xf
```

```
/******
```

Function : MP3WriteRegister

Description : send VS1011B's SCI command

Parameters : addressbyte, highbyte, lowbyte

Returned : nothing

```
*****/
```

```
Void MP3WriteRegister(addressbyte, highbyte, lowbyte){
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

MP3Selectcontrol();
spi2_read(VS_WRITE_COMMAND);
spi2_read(addressbyte);
spi2_read(highbyte);
spi2_read(lowbyte);
delay_us(100);    // After sending SCI command, it's not allowed to
                  // send SCI or SDI data for 5 microseconds
MP3DeselectControl();
}

/*****
Function : MP3SendZero
Description : send 2048 zeros to VS1011B
*****/
Void MP3SendZero(){
Int16 I;
MP3SelectData();
while(c<2048){
while( !(input(MP3_DREQ)) );
for (i=0;i<8;i++){
spi2_read(0x00);
c++;
}
}
MP3DeselectData();
}

/*****
Function : MP3SoftReset
Description : VS101B Soft Reset
*****/
Void MP3SoftReset(){

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

MP3WriteRegister(SCI_MODE,0x00,0x04); // SCI_MODE have 14 bits
delay_ms(1);          //0004 = 00 0000 0000 0100b

//1 : set SM_SoftReset

While( !(input(MP3_DREQ)) ); //wait for start up(after software reset

//DreQ stay low '0' at least 250us

MP3WriteRegister(SCI_CLOCKF,0x80,0x00); //double clock speed

/* Send nul bytes to data interface */
MP3SelectData();
spi2_read(0);
spi2_read(0);
spi2_read(0);
spi2_read(0);
Delay_ms(1);
MP3DeselectData();
MP3SendZero();
}

/*****
Function : MP3SetVolume
Description : Set MP3 volume
Parameters : leftchannel, rightchannel
*****/

Void MP3SetVolume(leftchannel, rightchannel)
{
    MP3WriteRegister(SCI_VOL,leftchannel,rightchannel);
}

/*****
Function : MP3Resest
Description : VS101B Reset
*****/

Void MP3Reset(){

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/***** Reset MP3 Decoder *****/

MP3PutInReset();

delay_ms(100);

/***** Send dummy Spi byte to initialize SPI *****/

spi2_read(0xff);

/***** Un-reset MP3 chip *****/

MP3DeselectControl();

MP3DeselectData();

MP3ReleaseFromReset();

While( !(input(MP3_DREQ))); //wait for DREQ

/***** Slow sample rate for slow analog part startup *****/

MP3WriteRegister(SCI_AUDATA,0x00,0x0a); // Set sample rate 10Hz

delay_ms(100);

/***** Switch on the analog parts *****/

MP3SetVolume(0xfe,0xfe); //set volume attenuate 254*(-0.5dB)= -127dB

MP3WriteRegister(SCI_AUDATA,31,64); // Set sample rate 8Hz

MP3SetVolume(0,0); //set volume maximum

MP3SoftReset();

}

/*****

Function : MP3SineTest

Description : test sine sound

*****/

Void MP3SineTest(){

/***** Send SCI command for Sine test *****/

MP3WriteRegister(SCI_MODE, 0x00, 0x20); // SCI_MODE have 14 bits

//002h=00 0000 0010 0000

//1 : SM_TESTS

delay_us(100);

MP3DeselectControl();

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้


```

while ( !(input(MP3_DREQ))) ;

/***** Send a Sine Test Header to Data port *****/

MP3SelectData();

spi2_read(0x53);    //Sine test initialize with the 8-byte sequence
spi2_read(0xef);
spi2_read(0x6e);
spi2_read(0x02);
spi2_read(0x00);
spi2_read(0x00);
spi2_read(0x00);
spi2_read(0x00);
spi2_read(0x00);
delay_us(100);
MP3DeselectData();
delay_ms(3000);

/****Stop the sine test sound****/

MP3SelectData();
spi2_read(0x45);
spi2_read(0x78);
spi2_read(0x69);
spi2_read(0x74);
spi2_read(0x00);
spi2_read(0x00);
spi2_read(0x00);
spi2_read(0x00);
spi2_read(0x00);
delay_us(100);
MP3DeselectData();
delay_ms(500);
}

#endif

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Board.c เป็น การประกาศการใช้งานพอร์ตต่างๆของ PIC 18f8722

```
//#ifndef _BOARD_
#define _BOARD_

/*****MP3 Decoder Connection*****/

#define MP3_XRESET PIN_D1

Void MP3PutInReset(){output_low(MP3_XRESET);}
Void MP3ReleaseFromReset(){output_high(MP3_XRESET);}

#define MP3_XCS PIN_D3
Void MP3SelectControl(){output_low(MP3_XCS);}
Void MP3DeselectControl(){output_high(MP3_XCS);}

#define MP3_BSYCN PIN_D2
Void MP3SelectData(){output_low(MP3_BSYCN);}
Void MP3DeselectData(){output_high(MP3_BSYCN);}

#define MP3_DREQ PIN_D0

/*****Manual SPI *****/

#define SCK PIN_A0
#define SDO PIN_A1
```

LCD.c เป็นโค้ดเพื่อเรียกใช้ LCD

```
/////////////////////////////////////////////////////////////////
///          LCDD.C          ///
///      Driver for common LCD modules      ///
///
///          ///
/// lcd_init() Must be called before any other function.    ///
///
///          ///
/// lcd_putc(c) Will display c on the next position of the LCD.  ///
///
///      The following have special meaning:      ///
///      \f Clear display      ///
///      \n Go to start of second line      ///
///      \b Move back one position      ///
///
///          ///
/// lcd_gotoxy(x,y) Set write position on LCD (upper left is 1,1)  ///
///
///          ///
/// lcd_getc(x,y) Returns character at position x,y on LCD      ///
///
///          ///
/////////////////////////////////////////////////////////////////
///      (C) Copyright 1996,2007 Custom Computer Services      ///
/// This source code may only be used by licensed users of the CCS C ///
/// compiler. This source code may only be distributed to other ///
/// licensed users of the CCS C compiler. No other use, reproduction ///
/// or distribution is permitted without written permission.      ///
/// Derivative programs created using this software in object code ///
/// form are not restricted in any way.      ///
/////////////////////////////////////////////////////////////////
// As defined in the following structure the pin connection is as follows:
// D0 enable
// D1 rs
// D2 rw
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

// D4 D4
// D5 D5
// D6 D6
// D7 D7
//

// LCD pins D0-D3 are not used and PIC D3 is not used.

// Un-comment the following define to use port B

// #define use_portb_lcd TRUE

struct lcd_pin_map { // This structure is overlayed
    BOOLEAN enable; // on to an I/O port to gain
    BOOLEAN rs; // access to the LCD pins.
    BOOLEAN rw; // The bits are allocated from
    BOOLEAN unused; // low order up. ENABLE will
    int data : 4; // be pin B0.
} lcd;

#if defined use_portb_lcd
    // #locate lcd = getenv("sfr:PORTB") // This puts the entire structure over the port
    #ifdef __pch__
        #locate lcd = 0xf81
    #else
        #locate lcd = 6
    #endif

    #define set_tris_lcd(x) set_tris_b(x)
#else
    // #locate lcd = getenv("sfr:PORTD") // This puts the entire structure over the port
    #ifdef __pch__
        #locate lcd = 0xf83
    #else
        #locate lcd = 8
    #endif

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้


```

#endif

#define set_tris_lcd(x) set_tris_d(x)

#endif

#ifndef lcd_type
#define lcd_type 2 // 0=5x7, 1=5x10, 2=2 lines
#endif

#define lcd_line_two 0x40 // LCD RAM address for the second line

BYTE const LCD_INIT_STRING[4] = {0x20 | (lcd_type << 2), 0xc, 1, 6};
// These bytes need to be sent to the LCD
// to start it up.
// The following are used for setting
// the I/O port direction register.

struct lcd_pin_map const LCD_WRITE = {0,0,0,0,0}; // For write mode all pins are out
struct lcd_pin_map const LCD_READ = {0,0,0,0,15}; // For read mode data pins are in

BYTE lcd_read_byte() {
    BYTE low,high;
    set_tris_lcd(LCD_READ);

    lcd.rw = 1;
    delay_cycles(1);
    lcd.enable = 1;
    delay_cycles(1);
    high = lcd.data;
    lcd.enable = 0;
    delay_cycles(1);
    lcd.enable = 1;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    delay_us(1);
    low = lcd.data;
    lcd.enable = 0;
    set_tris_lcd(LCD_WRITE);
    return( (high<<4) | low);
}

```

```

void lcd_send_nibble( BYTE n ) {
    lcd.data = n;
    delay_cycles(1);
    lcd.enable = 1;
    delay_us(2);
    lcd.enable = 0;
}

```

```

void lcd_send_byte( BYTE address, BYTE n ) {
    lcd.rs = 0;
    while ( bit_test(lcd_read_byte(),7) );
    lcd.rs = address;
    delay_cycles(1);
    lcd.rw = 0;
    delay_cycles(1);
    lcd.enable = 0;
    lcd_send_nibble(n >> 4);
    lcd_send_nibble(n & 0xf);
}

```

```

void lcd_init() {
    BYTE i;
    set_tris_lcd(LCD_WRITE);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

lcd.rs = 0;

lcd.rw = 0;

lcd.enable = 0;

delay_ms(15);

for(i=1;i<=3;++i) {

    lcd_send_nibble(3);

    delay_ms(5);

}

lcd_send_nibble(2);

for(i=0;i<=3;++i)

    lcd_send_byte(0,LCD_INIT_STRING[i]);

}

void lcd_gotoxy( BYTE x, BYTE y) {

    BYTE address;

    if(y!=1)

        address=lcd_line_two;

    else

        address=0;

    address+=x-1;

    lcd_send_byte(0,0x80|address);

}

void lcd_putc( char c) {

    switch (c) {

        case '\f' : lcd_send_byte(0,1);

                    delay_ms(2);

                    break;

        case '\n' : lcd_gotoxy(1,2);    break;

        case '\b' : lcd_send_byte(0,0x10); break;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        default : lcd_send_byte(1,c); break;
    }
}

char lcd_getc( BYTE x, BYTE y) {
    char value;
    lcd_gotoxy(x,y);
    while ( bit_test(lcd_read_byte(),7) ); // wait until busy flag is low
    lcd.rs=1;
    value = lcd_read_byte();
    lcd.rs=0;
    return(value);
}

void LCD_Command(int cm){
    lcd_send_byte(0,cm);
}

void LCD_ShiftLeft(){
    lcd_send_byte(0,0x18);
}

void LCD_ShiftRight(){
    lcd_send_byte(0,0x1c);
}

```

```

void LCD_MoveLeft(char p){
    char I;
    for(i=1;i<=p;++i) {
        LCD_ShiftLeft();
        delay_ms(500);
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้


```

    }
}

void LCD_MoveRight(char p){

```

```

    char I;
    for(i=0;i<=p;++i) {
        LCD_ShiftRight();
        delay_ms(500);
    }
}

```

```

void strcpy(char *s1, char *s2){
    while(*s1++= *s2++);
}

```

```

void LCD_String(char *s, int dly){
    while(*s!=0){
        lcd_putc(*s++);
        delay_ms(dly);
    }
}

```

mmc.c เป็นโค้ดเพื่อเรียกใช้งาน SD Card แบบ FAT16

```
#define cs PIN_C2

////////// mmc_cmd//////////

#separate

int mmc_cmd(int8 cmd,int32 address,int8 tries){

    int i,r1;

    for( i=0;i<16;i++) SPI_READ(0xFF);// digest prior operation

    SPI_READ(cmd);

    SPI_READ(MAKE8(address,3));

    SPI_READ(MAKE8(address,2));

    SPI_READ(MAKE8(address,1));

    SPI_READ(MAKE8(address,0));

    SPI_READ(0x95);

    for(i=0;i< tries;i++){

        r1=SPI_READ(0xFF);

        //printf("\r\n r1=%x",r1);

        if ((r1&0x80)==0) break;

    }

    return(r1);

}

//////////Init//////////

#separate

int8 Init (int8 max_tries)

{

    int16 i;

    int8 c,tries=0;

    spi_read(0xff);

    output_high(cs);                //Releasing CS line to "H"

    delay_ms(20);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

for(i=1;i<=20;i++) spi_write(0xff);    //Send min 80 clock

output_low(cs);                        //Pull CS line to "L"

delay_ms(20);

cmd0:  c=mmc_cmd(0x40,0x00000000,128);    //send cmd0

if(c==0x01){

    fprintf(com,"\n\r card idle");

    goto excmd55;

}

if(tries++<max_tries) goto cmd0;

else return(0x01);

excmd55: tries=0;

cmd55:  c=mmc_cmd(0x77,0x00000000,128);    //send cmd55

    fprintf(com,"\n\r %x",c);    // print R1 of CMD55

acmd41: c=mmc_cmd(0x41,0x00000000,128);    //send ACMD41

    fprintf(com,"\n\r after acmd41 : %x",c);    //print R1 of ACMD41

if(c==0x00)

{

    c=mmc_cmd(0x7A,0x00000000,128);    //send cmd58 to read OCR

    if(c==0x00)

    {

        fprintf(com,"\n\r%x",spi_read(0xff));

        fprintf(com,"\n\r%x",spi_read(0xff));

        fprintf(com,"\n\r%x",spi_read(0xff));

        fprintf(com,"\n\r%x",spi_read(0xff));

    }

    else return(0x58);    // if CMD58 is fail return 0x58

    return (0x11);

}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

else if(c==0x04)
{
    // If R1 of ACMD41 is 0x04 indicate that card is mmc
    fprintf(com,"\n\r after acmd41 : %x",c); // print R1 of ACMD41
    tries=0; // start initial mmc not sdc
cmd1:    c=mmc_cmd(0x41,0x00000000,128); //send cmd1
    if(c==0){
        return(0x11);
    }
    if(tries++<max_tries) goto cmd1;
    else return(0x01);
}
else
{
    // R1 of ACMD is 01
    if(tries++<max_tries) goto cmd55; // less than max_tries goto cmd55
    else return (0x01);
}
}

////////// read_BLOCK //////////
#separate
int read_BLOCK( int32 address, int8 *buff)
{
    int r1;
    long i,iw; // allows large gt 255 buff size addressing
    r1=mmc_cmd(0x51,address,16);
    if (r1==0x00) goto get_token; // we can read data payload
    if (r1==0x40) goto invalid;
invalid: return(0);
get_token: for(iw=0;iw<1024;iw++){
    r1=SPI_READ(0xFF);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้


```

        if (r1==0xFE) goto read_data; // read token $FE
    }

    return(0);

read_data: for (i=0;i<512;i++){
    buff[i]=SPI_READ(0xFF);
}

    SPI_READ(0xFF); // read crc
    SPI_READ(0xFF);

return(1);
}

/*#separate
void find_config (int8 *Resolution int8 *Baudrate)
{
    char buffer[24];
    write_block(((nxt-2)*sector_cluster+first_data+current_block)*512,buff); //current block
*/

```

Main.c (ครอบคลุมการทำงานของ MP3 ทั้งหมดของขอบเขตการทำงาน แต่ไม่สามารถใช้งานได้)
ใช้งานคู่กับ SD_Card.c

```
#include "C:\Documents and Settings\Administrator\Desktop\k project\sd.h"

#include "SD_CARD.c"

//#include "C:\Documents and Settings\Administrator\Desktop\new2\mmc.c"

#define use_portb_lcd

#include "LCD.c"

//#include <18f8722>

#include "Board.c"

#include "VS1002.c"

#define PLAY PIN_H7
#define STOP PIN_H6
#define NEXT PIN_H5
#define PREV PIN_H4
#define VOL_DOWN PIN_H3
#define VOL_UP PIN_H2

-

//void main()
//{

// setup_adc_ports(NO_ANALOGS|VSS_VDD);
// setup_adc(ADC_OFF|ADC_TAD_MUL_0);
// setup_psp(PSP_DISABLED);
// setup_spi(SPI_MASTER|SPI_H_TO_L|SPI_XMIT_L_TO_H|SPI_CLK_DIV_4);
// setup_spi2(SPI_MASTER|SPI_H_TO_L|SPI_XMIT_L_TO_H|SPI_CLK_DIV_4);
// setup_wdt(WDT_OFF);
// setup_timer_0(RTCC_INTERNAL);
// setup_timer_1(T1_DISABLED);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

// setup_timer_2(T2_DISABLED,0,1);
// setup_timer_4(T4_DISABLED,0,1);
// setup_comparator(NC_NC_NC_NC);
// setup_vref(FALSE);

//get_mbr();

//}

#fuses HS,NOWDT           //High speed crystal, No watchdog timer
#fuses NOLVP              //No low voltage protection
#use delay(clock=20000000) //use built-in function delay with clock speed 20MHz
#define msg0 "***MP3***"

void main(){

int32 total_sector,total_cluster,dir_addr;
int16 byte_sector,sector_fat,reserve,first_sector,fat1_addr,fat2_addr,first_data;
int8 sector_cluster;
unsigned char volume_values[] = {254, 76, 72, 68, 64, 60, 56, 52, 48, 44, 40, 36, 32, 28, 24,
20, 16, 12, 8, 4, 0};
int32 sound_pointer, root_addr, start_addr, file_size, root_start;
int16 i, sec_total, sec_num, rsv_size, fat_size, byte_per_sec;
int counter_p, counter_n, file_ad[4],vol_pos, fat, root_ad[4];
int1 play_en;
unsigned char data[512];
char str[8], Detail[512]; //file name byte 0-7, extension byte 8-10

//start cluster high 2 bytes : byte 21,20
//start cluster low 2 bytes : byte 27,26
//file size : byte 28-31

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

#bit CKE = 0xFC7.6

```
setup_adc_ports(NO_ANALOGS|VSS_VDD);
setup_adc(ADC_OFF|ADC_TAD_MUL_0);
setup_psp(PSP_DISABLED);
setup_spi(SPI_MASTER|SPI_H_TO_L|SPI_XMIT_L_TO_H|SPI_CLK_DIV_4);
setup_spi2(SPI_MASTER|SPI_H_TO_L|SPI_XMIT_L_TO_H|SPI_CLK_DIV_4);
setup_wdt(WDT_OFF);
setup_timer_0(RTCC_INTERNAL);
setup_timer_1(T1_DISABLED);
setup_timer_2(T2_DISABLED,0,1);
setup_timer_4(T4_DISABLED,0,1);
setup_comparator(NC_NC_NC_NC);
setup_vref(FALSE);

/***** Set Tristate *****/
set_tris_c(0xd0); //c7 , c6, c4 = input , c5, c3, c2, c1, c0 = output
set_tris_d(0x1f); //d4, d3, d2, d1, d0 = input , d7, d6, d5 = output
set_tris_e(0x1f);

/***** Set up SPI *****/
setup_spi(spi_master|spi_1_to_h|spi_clk_div_4); //idle state is low, bit rate = ext_clk/4 = 5Mbps
CKE=1; //detect data at clock rising edge
Delay_ms(2000);

/***** Reset Device *****/
lcd_init();
LCD_Command(0x01); //clear LCD
LCD_Command(0x80); //start 1 st line
Strcpy( str,msg0);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้


```

LCD_String(str,0);

While(SDReboot()); //Reboot SD Card

MP3Reset(); //Reset MP3 Decoder

Counter_n = 0;

Counter_p = 0;

```

```

/*Find start address of root directory */

```

```

while (SDCommand(0x51,0x00,0x00,0x00,0x00));
SDWaitForData();
for(i=0;i<512;i++){
    Detail[i] = spi_read(0xff);
}
rsv_size = Detail[15];
rsv_size = rsv_size<<8;
rsv_size += Detail[14];
fat = Detail[16];
fat_size = Detail[37];
fat_size = fat_size<<8;
fat_size += Detail[36];
root_start = rsv_size + (fat*fat_size);
root_start = root_start*0x200;
root_ad[0] = (root_start>>24) & 0xFF;
root_ad[1] = (root_start>>16) & 0xFF;
root_ad[2] = (root_start>>8) & 0xFF;
root_ad[3] = root_start & 0xFF;

```

```

/***** Find byte per sector *****/

```

```

byte_per_sec = Detail[12];
byte_per_sec = byte_per_sec<<8;
byte_per_sec += Detail[11];

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/* Get File Detail in root directory (start address, file name, file size)*/
while (SDCommand(0x51 ,root_ad[0],root_ad[1],root_ad[2],root_ad[3]));

SDWaitForData();
for(i=0;i<512;i++)
{
    Detail[i] = spi_read(0xff);
}

```

GetFileDetail :

```

start_addr = Detail[21+(counter_n*32)];
start_addr = start_addr<<8;
start_addr += Detail[20+(counter_n*32)];
start_addr = start_addr<<8;
start_addr += Detail[27+(counter_n*32)];
start_addr = start_addr<<8;
start_addr += Detail[26+(counter_n*32)];
start_addr = start_addr * 0x1000;

```

/***** cal file size *****/

```

file_size = Detail[31+(counter_n*32)];
file_size = file_size<<8;
file_size = Detail[30+(counter_n*32)];
file_size = file_size<<8;
file_size = Detail[29+(counter_n*32)];
file_size = file_size<<8;
file_size = Detail[28+(counter_n*32)];

```

/***** cal how many sector per file *****/

```

sec_total = file_size/byte_per_sec;

```

```

/***** Check MP3 file or not? *****/

if ( (Detail[8+(counter_n*32)] == 0x4D) && (Detail[9+(counter_n*32)] == 0x50) &&
    (Detail[10+(counter_n*32)] == 0x33) ) //0x4d=M, 0x50=P, 0x33=3
{
    /***** Show file name on LCD *****/

    LCD_Command(0x01);
    LCD_Command(0x80);
    for(i=0;i<8;i++)
    {
        lcd_putc(Detail[i+(counter_n*32)]);
    }
    check_play:
    while (true)
    {
        delay_ms(200);
        if ( ( ! (input(PLAY)) ) || (play_en == 1) ) //check key input
        {
            goto PlaySong;
        }
        if ( ! (input(NEXT)) )
        {
            counter_n++;
            goto GetFileDetail;
        }
    }
}

/*****Check W A V file or not? *****/

else if( (Detail[8+(counter_n*32)] == 0x57) && (Detail[9+(counter_n*32)] == 0x41) &&
    (Detail[10+(counter_n*32)] == 0x56) ) //0x57=W , 0x41 = A, 0x56=V
{

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/***** Show file name on LCD *****/
LCD_Command(0x01);
LCD_Command(0x08);
for(i=0;i<8;i++)
{
    lcd_putc(Detail[i+(counter_n*32)]);
}
while (true)
{
    if ( (!input(PLAY)) || (play_en == 1) )
        goto PlaySong;
    if (!input(NEXT))
    {
        counter_n++;
        goto GetFileDetail;
    }
}

/* if play until end of last song -> play again at 1st song */
else if (Detail[0+(counter_n*32)] == 0x00)
{
    counter_n = 0;
    goto GetFileDetail;
}

/* if file is not MP3&W A V get new file detail */
else
{
    counter_n++;
    goto GetFileDetail;
}
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

PlaySong:

```
play_en = 1;
sec_num = 0;
vol_pos = 16;
output_high(PIN_A2);
delay_ms(500);
output_low(PIN_A2);
MP3SelectData(); //enable MP3 Decoder to receive data
While(sec_num < sec_total)
{
    /***** fine address of each sector *****/
    start_addr = start_addr + (sec_num*512);
    file_ad[0] = (start_addr>>24) & 0xFF;
    file_ad[1] = (start_addr>>16) & 0xFF;
    file_ad[2] = (start_addr>>8) & 0xFF;
    file_ad[3] = start_addr & 0xFF;
    while (SDCommand(0x51,file_ad[0],file_ad[1],file_ad[2],file_ad[3]));
    SDWaitForData();
    /***** receive data from SD Card *****/
    for(i=0;i<512;i++)
    {
        data[i] = SPIGetChar();
    }
    sound_pointer=0;
    while(sound_pointer < 512)
    {
        while( !(input(MP3_DREQ)) );
        if (!input(VOL_UP))
        {
            //check key input volume control

            MP3DeselectData();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

If(vol_pos == 21)
{
    Vol_pos = 21;
}
else
{
    vol_pos++;
}
MP3SetVolume(volume_values[vol_pos],volume_values[vol_pos]);
MP3SelectData();
}
if (!input(VOL_DOWN))
{
    MP3DeselectData();
    If(vol_pos == 0)
    {
        Vol_pos = 0;
    }
    else
    {
        vol_pos--;
    }
    MP3SetVolume(volume_values[vol_pos],volume_values[vol_pos]);
    MP3SelectData();
}

/***** transmit data to MP3 Decoder *****/

for(i=0;i<8;i++)
{
    SPIPut(data[i]);

    Sound_pointer++;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    }
    if (!input(NEXT))
    {
        //check key input choose song
        counter_p = counter_n;
        counter_n++;
        goto GetFileDetail;
    }
    if (!input(PREV))
    {
        counter_n = counter_p;
        goto GetFileDetail;
    }
    if (!input(STOP))
    {
        play_en = 0;
        goto check_play;
    }
}
sec_num++;
}
counter_p = counter_n;
counter_n++;
goto GetFileDetail;
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

SD_Card.c

```
#ifndef _SD_Card_
#define _SD_Card_
#define SD_CS PIN_A3

void SDSelect(){output_low(SD_CS);}
void SDDEselect(){output_high(SD_CS);}

/*****

Function : SPI8Clocks
Description : send 8 clock cycles
Parameters : nclocks
Returned : nothing
*****/
void SPI8Clocks(unsigned char nClocks){
    while(nclocks--){
        spi_write(0xff);
    }
}

/*****
```

```
Function : SPIGetChar
Description : get response from SD Card
Parameters : nothing
Returned : data on data_out line
```

```
*****/
unsigned char SPIGetChar(){
    spi_write(0xff);
    return spi_read();
}

/*****
```

Function : SDCommand

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Description : send command to SD Card

Parameters : c1, c2, c3, c4, c5

Returned : Response from SD Card (c2)

*****/

```
unsigned char SDCommand(unsigned char c1,unsigned char c2,unsigned char c3,unsigned
char c4, unsigned char c5){
```

```
//c1 = command byte, c2 = command byte 1, c3 = command byte 2,
```

```
// c4 = command byte 3, c5 = command byte 4
```

```
unsigned char c,d,dat_out;
```

```
SDDeselect();
```

```
/* Provide clock edge before and after asserting SD CS */
```

```
SPI8Clocks(8); // NES = 8 Clock cycles
```

```
SDSelect();
```

```
SPI8Clocks(8); // NCS = 8 Clock Cycles
```

```
c=0;
```

```
/* If card still seems to be busy, give it some time... */
```

```
while( ((dat_out=SPIGetChar()) !=0xff) && (c<100) ){
```

```
    c++;
```

```
}
```

```
/* The bus should be stable high now */
```

```
if (dat_out != 0xff){
```

```
    SDDeselect();
```

```
    Return 0x81;
```

```
}
```

```
/* Send SD Command */
```

```
spi_write(c1);
```

```
spi_write(c2);
```

```
spi_write(c3);
```

```
spi_write(c4);
```

```
spi_write(c5);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

spi_write(0x95); //Valid CRC for init, then don't care
/* Now ok to use c1..c5 as temporaries */
/* Wait for R1 stay response (bit 7 low) from SD Card */
c1 = 100; //try max 100 times max NCR(time from end of command
        //to response)= 8 Clocks Cycles = 64 clocks
while( ((c2=SPIGetChar()) & 0x80) && (c1--));
return c2;
}
/*****
Function : SDWaitForData
Description : wait for start block of data (0xfe)
Parameters : nothing
Returned : result of waiting. 0 : Ok, 5 : fail to receive start block of data token
*****/
unsigned char SDWaitForData(){
    unsigned char result;
    // Wait until something else than 0xff is read from the bus
    do{
        result=SPIGetChar();
    }while(result==0xff);
    // Something was received from the bus
    if (result!=0xfe){ //Read fail, so I should flush any data that might be pending from SD Chard
        SPI8Clocks(200); // Flush SD by sending lots of FF's to it
        SPI8Clocks(200);
        SPI8Clocks(200);
        SDDeselect();
        return 5;
    }
    return 0;
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

/******

Function : SDReboot

Description : reboot SD Card (software reset)

Parameters : nothing

Returned : result of reboot. 0 : Ok, 2 : error

*****/

```
unsigned char SDReboot(){
    unsigned char c;
    SDSelect();
    for (c=0;c<200;c++){
        spi_write(0xff);
    }
    while (SDCommand(0x40,0,0,0,0) != 0x01 );
    SDCommand(0x7b,0,0,0,0);
    C=255;
    While( (c--) && (SDCommand(0x41,0,0,0,0)) ){ //max.255times SD-wake up call.
        //If not Idle SD return 0x00
    }
    delay_ms(10);
    if(c==1) //Time-out
    return 2; //Not able to power up SD Card
    }
    return 0; //Reboot Ok
}

#endif
```



MICROCHIP

PIC18F8722 Family Data Sheet

64/80-Pin, 1-Mbit,
Enhanced Flash Microcontrollers with
10-bit A/D and nanoWatt Technology



PIC18F8722 FAMILY

64/80-Pin, 1-Mbit, Enhanced Flash Microcontrollers with 10-Bit A/D and nanoWatt Technology

Peripheral Highlights:

- Two Master Synchronous Serial Port (MSSP) modules supporting 2/3/4-wire SPI™ (all 4 modes) and I²C™ Master and Slave modes
- Two Capture/Compare/PWM (CCP) modules
- Three Enhanced Capture/Compare/PWM (ECCP) modules:
 - One, two or four PWM outputs
 - Selectable polarity
 - Programmable dead time
 - Auto-Shutdown and Auto-Restart
- Two Enhanced Addressable USART modules:
 - Supports RS-485, RS-232 and LIN 1.2
 - Auto-Wake-up on Start bit
 - Auto-Baud Detect
- 10-bit, up to 16-channel Analog-to-Digital Converter module (A/D)
 - Auto-acquisition capability
 - Conversion available during Sleep
- Dual analog comparators with input multiplexing
- High-current sink/source 25 mA/25 mA
- Four programmable external interrupts
- Four input change interrupts

External Memory Interface (PIC18F8527/8622/8627/8722 only):

- Address capability of up to 2 Mbytes
- 8-bit or 16-bit interface
- 8, 12, 16 and 20-bit Address modes

Power-Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 15 μ A typical
- Sleep current down to 0.2 μ A typical
- Timer1 Oscillator: 1.8 μ A, 32 kHz, 2V
- Watchdog Timer: 2.1 μ A

Special Microcontroller Features:

- C compiler optimized architecture:
 - Optional extended instruction set designed to optimize re-entrant code
- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Flash/Data EEPROM Retention: 100 years typical
- Self-programmable under software control
- Priority levels for interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
 - Programmable period from 4 ms to 131s
- Single-Supply In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Wide operating voltage range: 2.0V to 5.5V
- Fail-Safe Clock Monitor
- Two-Speed Oscillator Start-up
- nanoWatt Technology

Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	CCP/ ECCP (PWM)	MSSP			EUSART	Comparators	Timers 8/16-bit	External Bus
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)				SPI™	Master I²C™					
PIC18F6527	48K	24576	3936	1024	54	12	2/3	2	Y	Y	2	2	2/3	N
PIC18F6622	64K	32768	3936	1024	54	12	2/3	2	Y	Y	2	2	2/3	N
PIC18F6627	96K	49152	3936	1024	54	12	2/3	2	Y	Y	2	2	2/3	N
PIC18F6722	128K	65536	3936	1024	54	12	2/3	2	Y	Y	2	2	2/3	N
PIC18F8527	48K	24576	3936	1024	70	16	2/3	2	Y	Y	2	2	2/3	Y
PIC18F8622	64K	32768	3936	1024	70	16	2/3	2	Y	Y	2	2	2/3	Y
PIC18F8627	96K	49152	3936	1024	70	16	2/3	2	Y	Y	2	2	2/3	Y
PIC18F8722	128K	65536	3936	1024	70	16	2/3	2	Y	Y	2	2	2/3	Y

PIC18F8722 FAMILY

1.3 Details on Individual Family Members

Devices in the PIC18F8722 family are available in 64-pin and 80-pin packages. Block diagrams for the two groups are shown in Figure 1-1 and Figure 1-2.

The devices are differentiated from each other in five ways:

1. Flash program memory (48 Kbytes for PIC18F6527/8527 devices, 64 Kbytes for PIC18F6622/8622 devices, 96 Kbytes for PIC18F6627/8627 devices and 128 Kbytes for PIC18F6722/8722).
2. A/D channels (12 for 64-pin devices, 16 for 80-pin devices).
3. I/O ports (7 bidirectional ports on 64-pin devices, 9 bidirectional ports on 80-pin devices).
4. External Memory Bus, configurable for 8 and 16-bit operation, is available on PIC18F8527/8622/8627/8722 devices.

All other features for devices in this family are identical. These are summarized in Table 1-2 and Table 1-2.

The pinouts for all devices are listed in Table 1-3 and Table 1-4.

Like all Microchip PIC18 devices, members of the PIC18F8722 family are available as both standard and low-voltage devices. Standard devices with Enhanced Flash memory, designated with an "F" in the part number (such as PIC18F6627), accommodate an operating VDD range of 4.2V to 5.5V. Low-voltage parts, designated by "LF" (such as PIC18LF6627), function over an extended VDD range of 2.0V to 5.5V.

TABLE 1-1: DEVICE FEATURES (PIC18F6527/6622/6627/6722)

Features	PIC18F6527	PIC18F6622	PIC18F6627	PIC18F6722
Operating Frequency	DC – 40 MHz	DC – 40 MHz	DC – 40 MHz	DC – 40 MHz
Program Memory (Bytes)	48K	64K	96K	128K
Program Memory (Instructions)	24576	32768	49152	65536
Data Memory (Bytes)	3936	3936	3936	3936
Data EEPROM Memory (Bytes)	1024	1024	1024	1024
Interrupt Sources	28	28	28	28
I/O Ports	Ports A, B, C, D, E, F, G	Ports A, B, C, D, E, F, G	Ports A, B, C, D, E, F, G	Ports A, B, C, D, E, F, G
Timers	5	5	5	5
Capture/Compare/PWM Modules	2	2	2	2
Enhanced Capture/Compare/PWM Modules	3	3	3	3
Enhanced USART	2	2	2	2
Serial Communications	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART
Parallel Communications (PSP)	Yes	Yes	Yes	Yes
10-bit Analog-to-Digital Module	12 Input Channels	12 Input Channels	12 Input Channels	12 Input Channels
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT
Programmable High/Low-Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled
Packages	64-pin TQFP	64-pin TQFP	64-pin TQFP	64-pin TQFP

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่ควรออกให้ผู้อื่นโดยไม่ได้รับอนุญาต

PIC18F8722 FAMILY

TABLE 1-2: DEVICE FEATURES (PIC18F8527/8622/8627/8722)

Features	PIC18F8527	PIC18F8622	PIC18F8627	PIC18F8722
Operating Frequency	DC – 40 MHz	DC – 40 MHz	DC – 40 MHz	DC – 40 MHz
Program Memory (Bytes)	48K	64K	96K	128K
Program Memory (Instructions)	24576	32768	49152	65536
Data Memory (Bytes)	3936	3936	3936	3936
Data EEPROM Memory (Bytes)	1024	1024	1024	1024
Interrupt Sources	29	29	29	29
I/O Ports	Ports A, B, C, D, E, F, G, H, J	Ports A, B, C, D, E, F, G, H, J	Ports A, B, C, D, E, F, G, H, J	Ports A, B, C, D, E, F, G, H, J
Timers	5	5	5	5
Capture/Compare/PWM Modules	2	2	2	2
Enhanced Capture/Compare/PWM Modules	3	3	3	3
Enhanced USART	2	2	2	2
Serial Communications	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART
Parallel Communications (PSP)	Yes	Yes	Yes	Yes
10-bit Analog-to-Digital Module	16 Input Channels	16 Input Channels	16 Input Channels	16 Input Channels
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT
Programmable High/Low-Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled
Packages	80-pin TQFP	80-pin TQFP	80-pin TQFP	80-pin TQFP

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้ทำซ้ำโดยไม่ได้รับอนุญาต

PIC18F8722 FAMILY

TABLE 1-3: PIC18F6527/6622/6627/6722 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
RG5/MCLR/VPP RG5 MCLR VPP	7	I I P	ST ST 	Master Clear (input) or programming voltage (input). Digital input. Master Clear (Reset) input. This pin is an active-low Reset to the device. Programming voltage input.
OSC1/CLKI/RA7 OSC1 CLKI RA7	39	I I I/O	ST CMOS TTL	Oscillator crystal or external clock input. Oscillator crystal input or external clock source input. ST buffer when configured in RC mode, CMOS otherwise. External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.) General purpose I/O pin.
OSC2/CLKO/RA6 OSC2 CLKO RA6	40	O O I/O	— — TTL	Oscillator crystal or clock output. Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate. General purpose I/O pin.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels Analog = Analog input
I = Input O = Output
P = Power I²C™ = I²C/SMBus input buffer

Note 1: Default assignment for ECCP2 when configuration bit CCP2MX is set.
Note 2: Alternate assignment for ECCP2 when configuration bit CCP2MX is cleared.

PIC18F8722 FAMILY

TABLE 1-3: PIC18F6527/6622/6627/6722 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
RA0/AN0 RA0 AN0	24	I/O I	TTL Analog	PORTA is a bidirectional I/O port. Digital I/O. Analog input 0.
RA1/AN1 RA1 AN1	23	I/O I	TTL Analog	Digital I/O. Analog input 1.
RA2/AN2/VREF- RA2 AN2 VREF-	22	I/O I I	TTL Analog Analog	Digital I/O. Analog input 2. A/D reference voltage (low) input.
RA3/AN3/VREF+ RA3 AN3 VREF+	21	I/O I I	TTL Analog Analog	Digital I/O. Analog input 3. A/D reference voltage (high) input.
RA4/T0CKI RA4 T0CKI	28	I/O I	ST ST	Digital I/O. Timer0 external clock input.
RA5/AN4/HLVDIN RA5 AN4 HLVDIN	27	I/O I I	TTL Analog Analog	Digital I/O. Analog input 4. High/Low-Voltage Detect input.
RA6				See the OSC2/CLKO/RA6 pin.
RA7				See the OSC1/CLKI/RA7 pin.

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
I²C™ = I²C/SMBus input buffer

Note 1: Default assignment for ECCP2 when configuration bit CCP2MX is set.
2: Alternate assignment for ECCP2 when configuration bit CCP2MX is cleared.

PIC18F8722 FAMILY

TABLE 1-3: PIC18F6527/6622/6627/6722 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
RB0/INT0/FLT0	48	I/O	TTL	PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs. Digital I/O. External interrupt 0. PWM Fault input for ECCPx.
RB0		I	ST	
INT0		I	ST	
FLT0		I	ST	
RB1/INT1	47	I/O	TTL	Digital I/O. External interrupt 1.
RB1		I	ST	
INT1		I	ST	
		I	ST	
RB2/INT2	46	I/O	TTL	Digital I/O. External interrupt 2.
RB2		I	ST	
INT2		I	ST	
		I	ST	
RB3/INT3	45	I/O	TTL	Digital I/O. External interrupt 3.
RB3		I	ST	
INT3		I	ST	
		I	ST	
RB4/KBI0	44	I/O	TTL	Digital I/O. Interrupt-on-change pin.
RB4		I	TTL	
KBI0		I	TTL	
		I	TTL	
RB5/KBI1/PGM	43	I/O	TTL	Digital I/O. Interrupt-on-change pin. Low-Voltage ICSP™ Programming enable pin.
RB5		I	TTL	
KBI1		I	TTL	
PGM		I/O	ST	
RB6/KBI2/PGC	42	I/O	TTL	Digital I/O. Interrupt-on-change pin. In-Circuit Debugger and ICSP programming clock pin.
RB6		I	TTL	
KBI2		I	TTL	
PGC		I/O	ST	
RB7/KBI3/PGD	37	I/O	TTL	Digital I/O. Interrupt-on-change pin. In-Circuit Debugger and ICSP programming data pin.
RB7		I	TTL	
KBI3		I	TTL	
PGD		I/O	ST	

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels Analog = Analog input
I = Input O = Output
P = Power I²C™ = I²C/SMBus input buffer

Note 1: Default assignment for ECCP2 when configuration bit CCP2MX is set.
Note 2: Alternate assignment for ECCP2 when configuration bit CCP2MX is cleared.

PIC18F8722 FAMILY

TABLE 1-3: PIC18F6527/6622/6627/6722 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
RC0/T1OSO/T13CKI	30	I/O	ST	PORTC is a bidirectional I/O port. Digital I/O. Timer1 oscillator output. Timer1/Timer3 external clock input.
RC0		O	—	
T1OSO		I	ST	
T13CKI				
RC1/T1OSI/ECCP2/P2A	29	I/O	ST	Digital I/O. Timer1 oscillator input. Enhanced Capture 2 input/Compare 2 output/ PWM 2 output. ECCP2 PWM output A.
RC1		I	CMOS	
T1OSI		I/O	ST	
ECCP2 ⁽¹⁾		O	—	
P2A ⁽¹⁾				
RC2/ECCP1/P1A	33	I/O	ST	Digital I/O. Enhanced Capture 1 input/Compare 1 output/ PWM 1 output. ECCP1 PWM output A.
RC2		I/O	ST	
ECCP1		O	—	
P1A				
RC3/SCK1/SCL1	34	I/O	ST	Digital I/O. Synchronous serial clock input/output for SPI™ mode. Synchronous serial clock input/output for I²C™ mode.
RC3		I/O	ST	
SCK1		I/O	ST	
SCL1				
RC4/SDI1/SDA1	35	I/O	ST	Digital I/O. SPI data in. I²C data I/O.
RC4		I	ST	
SDI1		I/O	ST	
SDA1				
RC5/SDO1	36	I/O	ST	Digital I/O. SPI data out.
RC5		O	—	
SDO1				
RC6/TX1/CK1	31	I/O	ST	Digital I/O. EUSART1 asynchronous transmit. EUSART1 synchronous clock (see related RX1/DT1).
RC6		O	—	
TX1		I/O	ST	
CK1				
RC7/RX1/DT1	32	I/O	ST	Digital I/O. EUSART1 asynchronous receive. EUSART1 synchronous data (see related TX1/CK1).
RC7		I	ST	
RX1		I/O	ST	
DT1				

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
I²C™ = I²C/SMBus input buffer

Note 1: Default assignment for ECCP2 when configuration bit CCP2MX is set.
2: Alternate assignment for ECCP2 when configuration bit CCP2MX is cleared.

PIC18F8722 FAMILY

TABLE 1-3: PIC18F6527/6622/6627/6722 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
RD0/PSP0 RD0 PSP0	58	I/O I/O	ST TTL	PORTD is a bidirectional I/O port. Digital I/O. Parallel Slave Port data.
RD1/PSP1 RD1 PSP1	55	I/O I/O	ST TTL	Digital I/O. Parallel Slave Port data.
RD2/PSP2 RD2 PSP2	54	I/O I/O	ST TTL	Digital I/O. Parallel Slave Port data.
RD3/PSP3 RD3 PSP3	53	I/O I/O	ST TTL	Digital I/O. Parallel Slave Port data.
RD4/PSP4/SDO2 RD4 PSP4 SDO2	52	I/O I/O O	ST TTL —	Digital I/O. Parallel Slave Port data. SPI data out.
RD5/PSP5/SDI2/SDA2 RD5 PSP5 SDI2 SDA2	51	I/O I/O I I/O	ST TTL ST I ² C/SMB	Digital I/O. Parallel Slave Port data. SPI™ data in. I ² C™ data I/O.
RD6/PSP6/SCK2/SCL2 RD6 PSP6 SCK2 SCL2	50	I/O I/O I/O I/O	ST TTL ST I ² C/SMB	Digital I/O. Parallel Slave Port data. Synchronous serial clock input/output for SPI mode. Synchronous serial clock input/output for I ² C mode.
RD7/PSP7/SS2 RD7 PSP7 SS2	49	I/O I/O I	ST TTL TTL	Digital I/O. Parallel Slave Port data. SPI slave select input.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels Analog = Analog input
I = Input O = Output
P = Power I²C™ = I²C/SMBus input buffer

Note 1: Default assignment for ECCP2 when configuration bit CCP2MX is set.
2: Alternate assignment for ECCP2 when configuration bit CCP2MX is cleared.

PIC18F8722 FAMILY

TABLE 1-3: PIC18F6527/6622/6627/6722 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
RE0/RD/P2D RE0 RD P2D	2	I/O I O	ST TTL —	PORTC is a bidirectional I/O port. Digital I/O. Read control for Parallel Slave Port. ECCP2 PWM output D.
RE1/WR/P2C RE1 WR P2C	1	I/O I O	ST TTL —	Digital I/O. Write control for Parallel Slave Port. ECCP2 PWM output C.
RE2/CS/P2B RE2 CS P2B	64	I/O I O	ST TTL —	Digital I/O. Chip select control for Parallel Slave Port. ECCP2 PWM output B.
RE3/P3C RE3 P3C	63	I/O O	ST —	Digital I/O. ECCP3 PWM output C.
RE4/P3B RE4 P3B	62	I/O O	ST —	Digital I/O. ECCP3 PWM output B.
RE5/P1C RE5 P1C	61	I/O O	ST —	Digital I/O. ECCP1 PWM output C.
RE6/P1B RE6 P1B	60	I/O O	ST —	Digital I/O. ECCP1 PWM output B.
RE7/ECCP2/P2A RE7 ECCP2 ⁽²⁾ P2A ⁽²⁾	59	I/O I/O O	ST ST —	Digital I/O. Enhanced Capture 2 input/Compare 2 output/ PWM 2 output. ECCP2 PWM output A.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
 ST = Schmitt Trigger input with CMOS levels Analog = Analog input
 I = Input O = Output
 P = Power I²C™ = I²C/SMBus input buffer

Note 1: Default assignment for ECCP2 when configuration bit CCP2MX is set.
Note 2: Alternate assignment for ECCP2 when configuration bit CCP2MX is cleared.

PIC18F8722 FAMILY

TABLE 1-3: PIC18F6527/6622/6627/6722 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
RF0/AN5	18	I/O	ST	PORTF is a bidirectional I/O port. Digital I/O. Analog input 5.
RF0		I	Analog	
AN5				
RF1/AN6/C2OUT	17	I/O	ST	
RF1		I	Analog	
AN6		O	—	
C2OUT				
RF2/AN7/C1OUT	16	I/O	ST	
RF2		I	Analog	
AN7		O	—	
C1OUT				
RF3/AN8	15	I/O	ST	Digital I/O. Analog input 8.
RF3		I	Analog	
AN8				
RF4/AN9	14	I/O	ST	
RF4		I	Analog	
AN9				
RF5/AN10/CVREF	13	I/O	ST	
RF5		I	Analog	
AN10		O	Analog	
CVREF				
RF6/AN11	12	I/O	ST	Digital I/O. Analog input 11.
RF6		I	Analog	
AN11				
RF7/SS1	11	I/O	ST	
RF7		I	TTL	
SS1				

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels Analog = Analog input
I = Input O = Output
P = Power I²C™ = I²C/SMBus input buffer

Note 1: Default assignment for ECCP2 when configuration bit CCP2MX is set.
2: Alternate assignment for ECCP2 when configuration bit CCP2MX is cleared.

PIC18F8722 FAMILY

TABLE 1-3: PIC18F6527/6622/6627/6722 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
RG0/ECCP3/P3A RG0 ECCP3 P3A	3	I/O I/O O	ST ST —	PORTG is a bidirectional I/O port. Digital I/O. Enhanced Capture 3 input/Compare 3 output/ PWM 3 output. ECCP3 PWM output A.
RG1/TX2/CK2 RG1 TX2 CK2	4	I/O O I/O	ST — ST	Digital I/O. EUSART2 asynchronous transmit. EUSART2 synchronous clock (see related RX2/DT2).
RG2/RX2/DT2 RG2 RX2 DT2	5	I/O I I/O	ST ST ST	Digital I/O. EUSART2 asynchronous receive. EUSART2 synchronous data (see related TX2/CK2).
RG3/CCP4/P3D RG3 CCP4 P3D	6	I/O I/O O	ST ST —	Digital I/O. Capture 4 input/Compare 4 output/PWM 4 output. ECCP3 PWM output D.
RG4/CCP5/P1D RG4 CCP5 P1D	8	I/O I/O O	ST ST —	Digital I/O. Capture 5 input/Compare 5 output/PWM 5 output. ECCP1 PWM output D.
RG5				See RG5/MCLR/VPP pin.
Vss	9, 25, 41, 56	P	—	Ground reference for logic and I/O pins.
VDD	10, 26, 38, 57	P	—	Positive supply for logic and I/O pins.
AVss	20	P	—	Ground reference for analog modules.
AVDD	19	P	—	Positive supply for analog modules.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels Analog = Analog input
I = Input O = Output
P = Power I²C™ = I²C/SMBus input buffer

Note 1: Default assignment for ECCP2 when configuration bit CCP2MX is set.
2: Alternate assignment for ECCP2 when configuration bit CCP2MX is cleared.

PIC18F8722 FAMILY

TABLE 1-4: PIC18F8527/8622/8627/8722 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
RG5/MCLR/VPP	9	I	ST	Master Clear (input) or programming voltage (input). Digital input. Master Clear (Reset) input. This pin is an active-low Reset to the device. Programming voltage input.
RG5		I	ST	
MCLR		P		
VPP				
OSC1/CLKI/RA7	49	I	ST	Oscillator crystal or external clock input. Oscillator crystal input or external clock source input. ST buffer when configured in RC mode, CMOS otherwise. External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.) General purpose I/O pin.
OSC1		I	CMOS	
CLKI		I/O	TTL	
RA7				
OSC2/CLKO/RA6	50	O	—	Oscillator crystal or clock output. Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate. General purpose I/O pin.
OSC2		O	—	
CLKO		I/O	TTL	
RA6				

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
I²C™/SMB = I²C/SMBus input buffer

- Note 1:** Alternate assignment for ECCP2 when configuration bit CCP2MX is cleared (all operating modes except Microcontroller mode).
- 2:** Default assignment for ECCP2 in all operating modes (CCP2MX is set).
- 3:** Alternate assignment for ECCP2 when CCP2MX is cleared (Microcontroller mode only).
- 4:** Default assignment for P1B/P1C/P3B/P3C (ECCPMX is set).
- 5:** Alternate assignment for P1B/P1C/P3B/P3C (ECCPMX is clear).

PIC18F8722 FAMILY

TABLE 1-4: PIC18F8527/8622/8627/8722 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
RA0/AN0	30	I/O	TTL	PORTA is a bidirectional I/O port.
RA0		I	Analog	Digital I/O.
AN0				Analog input 0.
RA1/AN1	29	I/O	TTL	Digital I/O.
RA1		I	Analog	Analog input 1.
AN1				
RA2/AN2/VREF-	28	I/O	TTL	Digital I/O.
RA2		I	Analog	Analog input 2.
AN2		I	Analog	A/D reference voltage (low) input.
VREF-				
RA3/AN3/VREF+	27	I/O	TTL	Digital I/O.
RA3		I	Analog	Analog input 3.
AN3		I	Analog	A/D reference voltage (high) input.
VREF+				
RA4/T0CKI	34	I/O	ST/OD	Digital I/O. Open-drain when configured as output.
RA4		I	ST	Timer0 external clock input.
T0CKI				
RA5/AN4/HLVDIN	33	I/O	TTL	Digital I/O.
RA5		I	Analog	Analog input 4.
AN4		I	Analog	High/Low-Voltage Detect input.
HLVDIN				
RA6				See the OSC2/CLKO/RA6 pin.
RA7				See the OSC1/CLKI/RA7 pin.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels Analog = Analog input
I = Input O = Output
P = Power I²C™/SMB = I²C/SMBus input buffer

- Note 1:** Alternate assignment for ECCP2 when configuration bit CCP2MX is cleared (all operating modes except Microcontroller mode).
- 2:** Default assignment for ECCP2 in all operating modes (CCP2MX is set).
- 3:** Alternate assignment for ECCP2 when CCP2MX is cleared (Microcontroller mode only).
- 4:** Default assignment for P1B/P1C/P3B/P3C (ECCPMX is set).
- 5:** Alternate assignment for P1B/P1C/P3B/P3C (ECCPMX is clear).

PIC18F8722 FAMILY

TABLE 1-4: PIC18F8527/8622/8627/8722 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
RB0/INT0/FLT0	58	I/O	TTL	PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0		I	ST	Digital I/O.
INT0		I	ST	External interrupt 0.
FLT0		I	ST	PWM Fault input for ECCPx.
RB1/INT1	57	I/O	TTL	Digital I/O.
RB1		I	ST	External interrupt 1.
INT1		I	ST	
RB2/INT2	56	I/O	TTL	Digital I/O.
RB2		I	ST	External interrupt 2.
INT2		I	ST	
RB3/INT3/ECCP2/P2A	55	I/O	TTL	Digital I/O.
RB3		I	ST	External interrupt 3.
INT3		I	ST	Enhanced Capture 2 input/Compare 2 output/
ECCP2 ⁽¹⁾		O	—	PWM 2 output.
P2A ⁽¹⁾		O	—	ECCP2 PWM output A.
RB4/KBI0	54	I/O	TTL	Digital I/O.
RB4		I	TTL	Interrupt-on-change pin.
KBI0		I	TTL	
RB5/KBI1/PGM	53	I/O	TTL	Digital I/O.
RB5		I	TTL	Interrupt-on-change pin.
KBI1		I	TTL	
PGM		I/O	ST	Low-Voltage ICSP™ Programming enable pin.
RB6/KBI2/PGC	52	I/O	TTL	Digital I/O.
RB6		I	TTL	Interrupt-on-change pin.
KBI2		I	TTL	
PGC		I/O	ST	In-Circuit Debugger and ICSP™ programming clock pin.
RB7/KBI3/PGD	47	I/O	TTL	Digital I/O.
RB7		I	TTL	Interrupt-on-change pin.
KBI3		I	TTL	
PGD		I/O	ST	In-Circuit Debugger and ICSP programming data pin.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
 ST = Schmitt Trigger input with CMOS levels Analog = Analog input
 I = Input O = Output
 P = Power I²C™/SMB = I²C/SMBus input buffer

- Note 1:** Alternate assignment for ECCP2 when configuration bit CCP2MX is cleared (all operating modes except Microcontroller mode).
- 2:** Default assignment for ECCP2 in all operating modes (CCP2MX is set).
- 3:** Alternate assignment for ECCP2 when CCP2MX is cleared (Microcontroller mode only).
- 4:** Default assignment for P1B/P1C/P3B/P3C (ECCPMX is set).
- 5:** Alternate assignment for P1B/P1C/P3B/P3C (ECCPMX is clear).

PIC18F8722 FAMILY

TABLE 1-4: PIC18F8527/8622/8627/8722 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
C0/T1OSO/T13CKI	36	I/O	ST	PORTC is a bidirectional I/O port.
RC0		O	—	Digital I/O.
T1OSO		I	ST	Timer1 oscillator output.
T13CKI				Timer1/Timer3 external clock input.
C1/T1OSI/ECCP2/P2A	35	I/O	ST	Digital I/O.
RC1		I	CMOS	Timer1 oscillator input.
T1OSI		I/O	ST	Enhanced Capture 2 input/Compare 2 output/
ECCP2 ⁽²⁾				PWM 2 output.
P2A ⁽²⁾		O	—	ECCP2 PWM output A.
C2/ECCP1/P1A	43	I/O	ST	Digital I/O.
RC2		I/O	ST	Enhanced Capture 1 input/Compare 1 output/
ECCP1				PWM 1 output.
P1A		O	—	ECCP1 PWM output A.
C3/SCK1/SCL1	44	I/O	ST	Digital I/O.
RC3		I/O	ST	Synchronous serial clock input/output for SPI™ mode.
SCK1		I/O	ST	Synchronous serial clock input/output for I²C™ mode.
SCL1				
C4/SDI1/SDA1	45	I/O	ST	Digital I/O.
RC4		I	ST	SPI data in.
SDI1		I/O	ST	I²C data I/O.
SDA1				
C5/SDO1	46	I/O	ST	Digital I/O.
RC5		O	—	SPI data out.
SDO1				
C6/TX1/CK1	37	I/O	ST	Digital I/O.
RC6		O	—	EUSART1 asynchronous transmit.
TX1		I/O	ST	EUSART1 synchronous clock (see related RX1/DT1).
CK1				
C7/RX1/DT1	38	I/O	ST	Digital I/O.
RC7		I	ST	EUSART1 asynchronous receive.
RX1		I/O	ST	EUSART1 synchronous data (see related TX1/CK1).
DT1				

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
I²C™/SMB = I²C/SMBus input buffer

- Note 1:** Alternate assignment for ECCP2 when configuration bit CCP2MX is cleared (all operating modes except Microcontroller mode).
- 2:** Default assignment for ECCP2 in all operating modes (CCP2MX is set).
- 3:** Alternate assignment for ECCP2 when CCP2MX is cleared (Microcontroller mode only).
- 4:** Default assignment for P1B/P1C/P3B/P3C (ECCPMX is set).
- 5:** Alternate assignment for P1B/P1C/P3B/P3C (ECCPMX is clear).

PIC18F8722 FAMILY

TABLE 1-4: PIC18F8527/8622/8627/8722 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number TQFP	Pin Type	Buffer Type	Description
PORTD is a bidirectional I/O port.				
RD0/AD0/PSP0	72	I/O	ST	Digital I/O.
RD0		I/O	TTL	External memory address/data 0.
AD0		I/O	TTL	Parallel Slave Port data.
PSP0				
RD1/AD1/PSP1	69	I/O	ST	Digital I/O.
RD1		I/O	TTL	External memory address/data 1.
AD1		I/O	TTL	Parallel Slave Port data.
PSP1				
RD2/AD2/PSP2	68	I/O	ST	Digital I/O.
RD2		I/O	TTL	External memory address/data 2.
AD2		I/O	TTL	Parallel Slave Port data.
PSP2				
RD3/AD3/PSP3	67	I/O	ST	Digital I/O.
RD3		I/O	TTL	External memory address/data 3.
AD3		I/O	TTL	Parallel Slave Port data.
PSP3				
RD4/AD4/PSP4/SDO2	66	I/O	ST	Digital I/O.
RD4		I/O	TTL	External memory address/data 4.
AD4		I/O	TTL	Parallel Slave Port data.
PSP4		I/O	TTL	SPI™ data out.
SDO2		O	—	
RD5/AD5/PSP5/ SDI2/SDA2	65	I/O	ST	Digital I/O.
RD5		I/O	TTL	External memory address/data 5.
AD5		I/O	TTL	Parallel Slave Port data.
PSP5		I/O	TTL	SPI data in.
SDI2		I	ST	I²C™ data I/O.
SDA2		I/O	I²C/SMB	
RD6/AD6/PSP6/ SCK2/SCL2	64	I/O	ST	Digital I/O.
RD6		I/O	TTL	External memory address/data 6.
AD6		I/O	TTL	Parallel Slave Port data.
PSP6		I/O	TTL	Synchronous serial clock input/output for SPI mode.
SCK2		I/O	ST	Synchronous serial clock input/output for I²C mode.
SCL2		I/O	I²C/SMB	
RD7/AD7/PSP7/SS2	63	I/O	ST	Digital I/O.
RD7		I/O	TTL	External memory address/data 7.
AD7		I/O	TTL	Parallel Slave Port data.
PSP7		I/O	TTL	SPI slave select input.
SS2		I	TTL	

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
 ST = Schmitt Trigger input with CMOS levels Analog = Analog input
 I = Input O = Output
 P = Power I²C™/SMB = I²C/SMBus input buffer

- Note 1:** Alternate assignment for ECCP2 when configuration bit CCP2MX is cleared (all operating modes except Microcontroller mode).
- 2:** Default assignment for ECCP2 in all operating modes (CCP2MX is set).
- 3:** Alternate assignment for ECCP2 when CCP2MX is cleared (Microcontroller mode only).
- 4:** Default assignment for P1B/P1C/P3B/P3C (ECCPMX is set).
- 5:** Alternate assignment for P1B/P1C/P3B/P3C (ECCPMX is clear).

PIC18F8722 FAMILY

TABLE 1-4: PIC18F8527/8622/8627/8722 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
RE0/AD8/RD/P2D	4	I/O	ST	PORTC is a bidirectional I/O port.
RE0		I/O	TTL	Digital I/O.
AD8		I	TTL	External memory address/data 8.
RD		O	—	Read control for Parallel Slave Port.
P2D		O	—	ECCP2 PWM output D.
RE1/AD9/WR/P2C	3	I/O	ST	Digital I/O.
RE1		I/O	TTL	External memory address/data 9.
AD9		I	TTL	Write control for Parallel Slave Port.
WR		O	—	ECCP2 PWM output C.
P2C		O	—	
RE2/AD10/CS/P2B	78	I/O	ST	Digital I/O.
RE2		I/O	TTL	External memory address/data 10.
AD10		I	TTL	Chip select control for Parallel Slave Port.
CS		O	—	ECCP2 PWM output B.
P2B		O	—	
RE3/AD11/P3C	77	I/O	ST	Digital I/O.
RE3		I/O	TTL	External memory address/data 11.
AD11		O	—	ECCP3 PWM output C.
P3C ⁽⁴⁾		O	—	
RE4/AD12/P3B	76	I/O	ST	Digital I/O.
RE4		I/O	TTL	External memory address/data 12.
AD12		O	—	ECCP3 PWM output B.
P3B ⁽⁴⁾		O	—	
RE5/AD13/P1C	75	I/O	ST	Digital I/O.
RE5		I/O	TTL	External memory address/data 13.
AD13		O	—	ECCP1 PWM output C.
P1C ⁽⁴⁾		O	—	
RE6/AD14/P1B	74	I/O	ST	Digital I/O.
RE6		I/O	TTL	External memory address/data 14.
AD14		O	—	ECCP1 PWM output B.
P1B ⁽⁴⁾		O	—	
RE7/AD15/ECCP2/P2A	73	I/O	ST	Digital I/O.
RE7		I/O	TTL	External memory address/data 15.
AD15		I/O	ST	Enhanced Capture 2 input/Compare 2 output/
ECCP2 ⁽³⁾		O	—	PWM 2 output.
P2A ⁽³⁾		O	—	ECCP2 PWM output A.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels Analog = Analog input
I = Input O = Output
P = Power I²C™/SMB = I²C/SMBus input buffer

- Note 1:** Alternate assignment for ECCP2 when configuration bit CCP2MX is cleared (all operating modes except Microcontroller mode).
- 2:** Default assignment for ECCP2 in all operating modes (CCP2MX is set).
- 3:** Alternate assignment for ECCP2 when CCP2MX is cleared (Microcontroller mode only).
- 4:** Default assignment for P1B/P1C/P3B/P3C (ECCPMX is set).
- 5:** Alternate assignment for P1B/P1C/P3B/P3C (ECCPMX is clear).

PIC18F8722 FAMILY

TABLE 1-4: PIC18F8527/8622/8627/8722 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
RF0/AN5 RF0 AN5	24	I/O I	ST Analog	PORTF is a bidirectional I/O port. Digital I/O. Analog input 5.
RF1/AN6/C2OUT RF1 AN6 C2OUT	23	I/O I O	ST Analog —	Digital I/O. Analog input 6. Comparator 2 output.
RF2/AN7/C1OUT RF2 AN7 C1OUT	18	I/O I O	ST Analog —	Digital I/O. Analog input 7. Comparator 1 output.
RF3/AN8 RF3 AN8	17	I/O I	ST Analog	Digital I/O. Analog input 8.
RF4/AN9 RF4 AN9	16	I/O I	ST Analog	Digital I/O. Analog input 9.
RF5/AN10/CVREF RF5 AN10 CVREF	15	I/O I O	ST Analog Analog	Digital I/O. Analog input 10. Comparator reference voltage output.
RF6/AN11 RF6 AN11	14	I/O I	ST Analog	Digital I/O. Analog input 11.
RF7/SS1 RF7 SS1	13	I/O I	ST TTL	Digital I/O. SPI™ slave select input.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels Analog = Analog input
I = Input O = Output
P = Power I²C™/SMB = I²C/SMBus input buffer

- Note 1:** Alternate assignment for ECCP2 when configuration bit CCP2MX is cleared (all operating modes except Microcontroller mode).
2: Default assignment for ECCP2 in all operating modes (CCP2MX is set).
3: Alternate assignment for ECCP2 when CCP2MX is cleared (Microcontroller mode only).
4: Default assignment for P1B/P1C/P3B/P3C (ECCPMX is set).
5: Alternate assignment for P1B/P1C/P3B/P3C (ECCPMX is clear).

PIC18F8722 FAMILY

TABLE 1-4: PIC18F8527/8622/8627/8722 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
RG0/ECCP3/P3A RG0 ECCP3 P3A	5	I/O I/O O	ST ST —	PORTG is a bidirectional I/O port. Digital I/O. Enhanced Capture 3 input/Compare 3 output/ PWM 3 output. ECCP3 PWM output A.
RG1/TX2/CK2 RG1 TX2 CK2	6	I/O O I/O	ST — ST	Digital I/O. EUSART2 asynchronous transmit. EUSART2 synchronous clock (see related RX2/DT2).
RG2/RX2/DT2 RG2 RX2 DT2	7	I/O I I/O	ST ST ST	Digital I/O. EUSART2 asynchronous receive. EUSART2 synchronous data (see related TX2/CK2).
RG3/CCP4/P3D RG3 CCP4 P3D	8	I/O I/O O	ST ST —	Digital I/O. Capture 4 input/Compare 4 output/PWM 4 output. ECCP3 PWM output D.
RG4/CCP5/P1D RG4 CCP5 P1D	10	I/O I/O O	ST ST —	Digital I/O. Capture 5 input/Compare 5 output/PWM 5 output. ECCP1 PWM output D.
RG5				See RG5/MCLR/VPP pin.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels Analog = Analog input
I = Input O = Output
P = Power I²C™/SMB = I²C/SMBus input buffer

- Note 1:** Alternate assignment for ECCP2 when configuration bit CCP2MX is cleared (all operating modes except Microcontroller mode).
2: Default assignment for ECCP2 in all operating modes (CCP2MX is set).
3: Alternate assignment for ECCP2 when CCP2MX is cleared (Microcontroller mode only).
4: Default assignment for P1B/P1C/P3B/P3C (ECCPMX is set).
5: Alternate assignment for P1B/P1C/P3B/P3C (ECCPMX is clear).

PIC18F8722 FAMILY

TABLE 1-4: PIC18F8527/8622/8627/8722 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
RH0/A16	79	I/O	ST	PORTH is a bidirectional I/O port.
RH0		I/O	TTL	Digital I/O.
A16				External memory address/data 16.
RH1/A17	80	I/O	ST	Digital I/O.
RH1		I/O	TTL	External memory address/data 17.
A17				
RH2/A18	1	I/O	ST	Digital I/O.
RH2		I/O	TTL	External memory address/data 18.
A18				
RH3/A19	2	I/O	ST	Digital I/O.
RH3		I/O	TTL	External memory address/data 19.
A19				
RH4/AN12/P3C	22	I/O	ST	Digital I/O.
RH4		I	Analog	Analog input 12.
AN12		O	—	ECCP3 PWM output C.
P3C ⁽⁵⁾				
RH5/AN13/P3B	21	I/O	ST	Digital I/O.
RH5		I	Analog	Analog input 13.
AN13		O	—	ECCP3 PWM output B.
P3B ⁽⁵⁾				
RH6/AN14/P1C	20	I/O	ST	Digital I/O.
RH6		I	Analog	Analog input 14.
AN14		O	—	ECCP1 PWM output C.
P1C ⁽⁵⁾				
RH7/AN15/P1B	19	I/O	ST	Digital I/O.
RH7		I	Analog	Analog input 15.
AN15		O	—	ECCP1 PWM output B.
P1B ⁽⁵⁾				

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels Analog = Analog input
I = Input O = Output
P = Power I²C™/SMB = I²C/SMBus input buffer

- Note 1:** Alternate assignment for ECCP2 when configuration bit CCP2MX is cleared (all operating modes except Microcontroller mode).
- 2:** Default assignment for ECCP2 in all operating modes (CCP2MX is set).
- 3:** Alternate assignment for ECCP2 when CCP2MX is cleared (Microcontroller mode only).
- 4:** Default assignment for P1B/P1C/P3B/P3C (ECCPMX is set).
- 5:** Alternate assignment for P1B/P1C/P3B/P3C (ECCPMX is clear).

PIC18F8722 FAMILY

TABLE 1-4: PIC18F8527/8622/8627/8722 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
RJ0/ALE	62	I/O	ST	PORTJ is a bidirectional I/O port.
RJ0		O	—	Digital I/O.
ALE				External memory address latch enable.
RJ1/ $\overline{\text{OE}}$	61	I/O	ST	Digital I/O.
RJ1		O	—	External memory output enable.
$\overline{\text{OE}}$				
RJ2/ $\overline{\text{WRL}}$	60	I/O	ST	Digital I/O.
RJ2		O	—	External memory write low control.
$\overline{\text{WRL}}$				
RJ3/ $\overline{\text{WRH}}$	59	I/O	ST	Digital I/O.
RJ3		O	—	External memory write high control.
$\overline{\text{WRH}}$				
RJ4/BA0	39	I/O	ST	Digital I/O.
RJ4		O	—	External memory byte address 0 control.
BA0				
RJ5/ $\overline{\text{CE}}$	40	I/O	ST	Digital I/O.
RJ5		O	—	External memory chip enable control.
$\overline{\text{CE}}$				
RJ6/ $\overline{\text{LB}}$	41	I/O	ST	Digital I/O.
RJ6		O	—	External memory low byte control.
$\overline{\text{LB}}$				
RJ7/ $\overline{\text{UB}}$	42	I/O	ST	Digital I/O.
RJ7		O	—	External memory high byte control.
$\overline{\text{UB}}$				
Vss	11, 31, 51, 70	P	—	Ground reference for logic and I/O pins.
VDD	12, 32, 48, 71	P	—	Positive supply for logic and I/O pins.
AVss	26	P	—	Ground reference for analog modules.
AVDD	25	P	—	Positive supply for analog modules.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
 ST = Schmitt Trigger input with CMOS levels Analog = Analog input
 I = Input O = Output
 P = Power I²C™/SMB = I²C/SMBus input buffer

Note 1: Alternate assignment for ECCP2 when configuration bit CCP2MX is cleared (all operating modes except Microcontroller mode).

- 2: Default assignment for ECCP2 in all operating modes (CCP2MX is set).
- 3: Alternate assignment for ECCP2 when CCP2MX is cleared (Microcontroller mode only).
- 4: Default assignment for P1B/P1C/P3B/P3C (ECCPMX is set).
- 5: Alternate assignment for P1B/P1C/P3B/P3C (ECCPMX is clear).

VS1002d - MP3 AUDIO CODEC

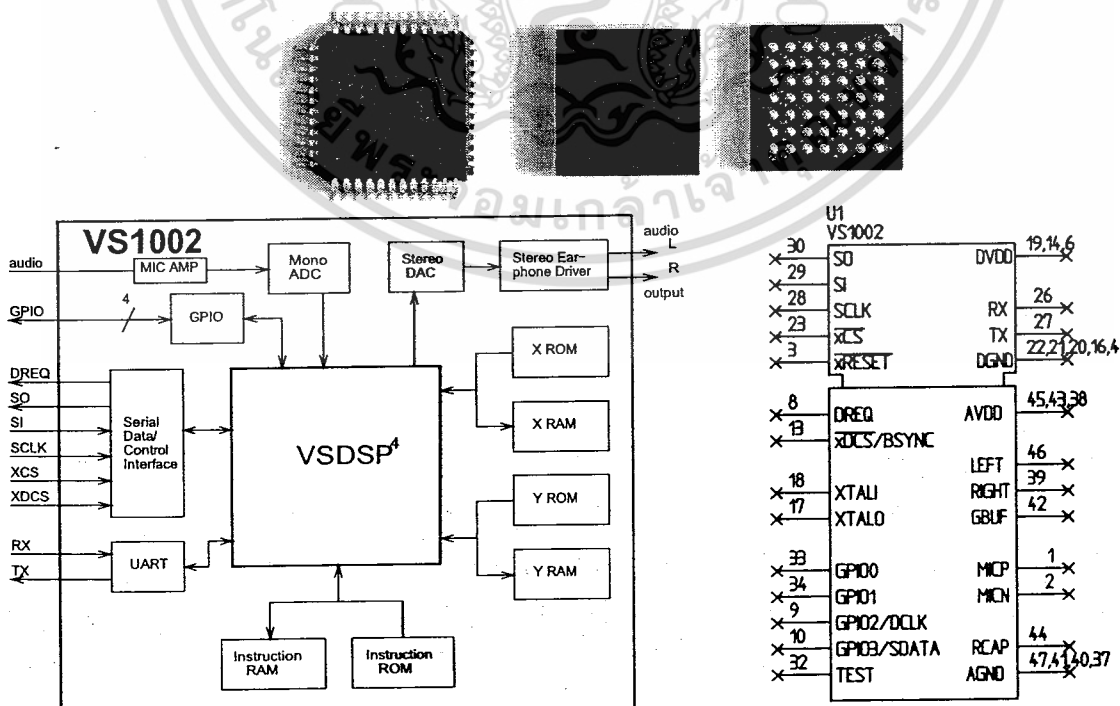
Features

- Decodes MPEG 1.0 & 2.0 audio layer III (CBR + VBR); WAV and PCM files
- Encodes ADPCM from microphone input
- Streaming support for MP3 and WAV
- Bass control
- Operates with single a clock 12..13 MHz or 24..26 MHz.
- Internal clock doubler
- Low-power operation
- High-quality on-chip stereo DAC with no phase error between channels
- Stereo earphone driver capable of driving a 30Ω load
- Separate 2.5 V..3.6 V operating voltages for analog and digital
- 7.5 KiB On-chip RAM for user code / data
- Serial control and data interfaces
- Can be used as a slave co-processor
- SPI flash boot for special applications
- UART for debugging purposes
- New functions may be added with software and 4 GPIO pins
- Lead-free RoHS-compliant packages

Description

VS1002d is a single-chip MP3 audio decoder. It contains a high-performance, low-power DSP processor core VS_DSP⁴, working data memory, 5 KiB instruction RAM and 2.5 KiB data RAM for user applications, serial control and input data interfaces, 4 general purpose I/O pins, an UART, as well as a high-quality variable-sample-rate mono ADC and stereo DAC, followed by an earphone amplifier and a ground buffer.

VS1002d receives its input bitstream through a serial input bus, which it listens to as a system slave. The input stream is decoded and passed through a digital volume control to an 18-bit over-sampling, multi-bit, sigma-delta DAC. The decoding is controlled via a serial control bus. In addition to the basic decoding, it is possible to add application specific features, like DSP effects, to the user RAM memory.



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4 Characteristics & Specifications

4.1 Absolute Maximum Ratings

Parameter	Symbol	Min	Max	Unit
Analog Positive Supply	AVDD	-0.3	3.6	V
Digital Positive Supply	DVDD	-0.3	3.6	V
Current at Any Digital Output			±50	mA
Voltage at Any Digital Input ²		-0.3	DVDD+0.3 ¹	V
Operating Temperature		-40	+85	°C
Storage Temperature		-65	+150	°C

¹ Must not exceed 3.6 V

² Current must be limited to ±50 mA

4.2 Recommended Operating Conditions

Parameter	Symbol	Min	Typ	Max	Unit
Ambient Operating Temperature		-40	.	+85	°C
Analog and Digital Ground ¹	AGND DGND		0.0		V
Positive Analog	AVDD	2.5	2.7	3.6	V
Positive Digital	DVDD	2.3	2.7	3.6	V
Input Clock Frequency	XTALI	24	24.576	26	MHz
Input Clock Frequency, with clock doubler	XTALI	12	12.288	13	MHz
Internal Clock Frequency	CLKI	24 ²	24.576	26	MHz
Internal Clock Frequency, DVDD ≥ 2.7V	CLKI	24 ²		28.636	MHz
Master Clock Duty Cycle		40~	50	60	%

¹ Must be connected together as close to the device as possible for latch-up immunity.

² The maximum sample rate that can be played with correct speed is CLKI/512.

Thus, if CLKI is 24 MHz, 48 kHz is played 2.5% off-pitch.

4.3 Analog Characteristics

Unless otherwise noted: AVDD=2.5..3.6V, DVDD=2.3..3.6V, TA=-40..+85°C, XTALI=12..13MHz, internal Clock Doubler active. DAC tested with 1307.894 Hz full-scale output sinewave, measurement bandwidth 20..20000 Hz, analog output load: LEFT to GBUF 30Ω, RIGHT to GBUF 30Ω. Microphone test amplitude 100 mVpp, f=1 kHz.

Parameter	Symbol	Min	Typ	Max	Unit
DAC Resolution			18		bits
Total Harmonic Distortion	THD		0.1	0.2	%
Dynamic Range (DAC unmuted, A-weighted)	IDR		90		dB
S/N Ratio (full scale signal)	SNR	70	85		dB
Interchannel Isolation (Cross Talk), AC-coupled		50	75		dB
Interchannel Isolation (Cross Talk), with GBUF			40		dB
Interchannel Gain Mismatch		-0.5		0.5	dB
Frequency Response 20 Hz..15000 Hz		-0.2		0.2	dB
Frequency Response 15000 Hz..19000 Hz		-1.0		-0.2	dB
Full Scale Output Voltage (Peak-to-peak)		1.4	1.6 ¹	2.0	Vpp
Deviation from Linear Phase				5	°
Analog Output Load Resistance	AOLR	16	30 ²		Ω
Analog Output Load Capacitance				100	pF
Microphone input impedance			100		kΩ
Microphone input amplitude			100 ³	280	mVpp AC
Microphone Total Harmonic Distortion	MTHD		0.03	0.10	%
Microphone S/N Ratio	MSNR	70	82		dB

¹ 3.2 volts can be achieved with +-to-+ wiring for mono difference sound.

² AOLR may be much lower, but below *Typical* distortion performance may be compromised.

³ 100 mVpp is optimum level. Above typical amplitude the Harmonic Distortion increases.

4.4 Power Consumption

Average current tested with an MPEG 1.0 Layer III 128 kbit/s sample and generated sine, output at full volume, XTALI = 12.288 MHz, internal clock doubler enabled, DVDD = 2.7 V, AVDD = 2.7 V.

Parameter	Min	Typ	Max	Unit
Power Supply Consumption AVDD, Reset		0.6	5.0	μA
Power Supply Consumption DVDD, Reset		3.7	10.0	μA
Power Supply Consumption AVDD, sine test, 30Ω + GBUF		22	30	mA
Power Supply Consumption DVDD, sine test		9	18	mA
Power Supply Consumption AVDD, no load		6		mA
Power Supply Consumption AVDD, output load 30Ω		10		mA
Power Supply Consumption AVDD, 30Ω + GBUF		16		mA
Power Supply Consumption DVDD		19		mA

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.5 Digital Characteristics

Parameter	Symbol	Min	Typ	Max	Unit
High-Level Input Voltage		$0.7 \times DVDD$		$DVDD + 0.3^1$	V
Low-Level Input Voltage		-0.2		$0.3 \times DVDD$	V
High-Level Output Voltage at $I_O = -2.0$ mA		$0.7 \times DVDD$			V
Low-Level Output Voltage at $I_O = 2.0$ mA				$0.3 \times DVDD$	V
Input Leakage Current		-1.0		1.0	μA
SPI Input Clock Frequency ²				$\frac{CLKI}{6}$	MHz
Rise time of all output pins, load = 50 pF				50	ns

¹ Must not exceed 3.6V

² Value for SCI reads. SCI and SDI writes allow $\frac{CLKI}{4}$.

4.6 Switching Characteristics - Boot Initialization

Parameter	Symbol	Min	Max	Unit
XRESET active time		2		XTALI
XRESET inactive to software ready			50000 ¹	XTALI
Power on reset, rise time of DVDD		10		V/s

¹ DREQ rises when initialization is complete. You should not send any data or commands before that.

5 Packages and Pin Descriptions

5.1 Packages

Both LPQFP-48 and BGA-49 are lead (Pb) free and also RoHS compliant packages. RoHS is a short name of *Directive 2002/95/EC on the restriction of the use of certain hazardous substances in electrical and electronic equipment*.

5.1.1 LQFP-48

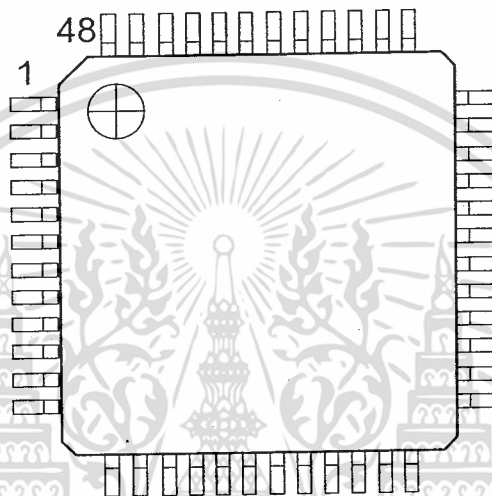


Figure 1: Pin Configuration, LQFP-48.

LQFP-48 package dimensions are at <http://www.vlsi.fi/>.

5.1.2 BGA-49

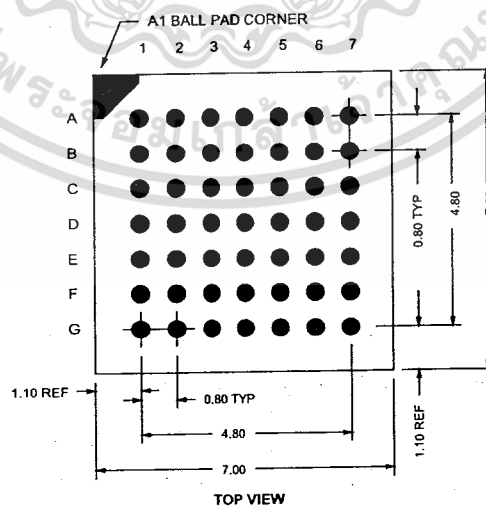


Figure 2: Pin Configuration, BGA-49.

BGA-49 package dimensions are at <http://www.vlsi.fi/>.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.2 LQFP-48 and BGA-49 Pin Descriptions

Pin Name	LQFP-48 Pin	BGA49 Ball	Pin Type	Function
MICP ²	1	C3	AI	microphone input, use pull-down resistor if not used
MICN ²	2	C2	AI	microphone input, use pull-down resistor if not used
XRESET	3	B1	DI	active low asynchronous reset
DGND0	4	D2	PWR	digital ground
DVDD0	6	D3	PWR	digital power supply
DREQ	8	E2	DO	data request output
GPIO2 ² / DCLK ¹	9	E1	DI	general purpose IO 2 / serial input data bus clock, use pull-down resistor if not used
GPIO3 ² / SDATA ¹	10	F2	DI	general purpose IO 3 / serial data input, use pull-down resistor if not used
XDCS / BSYNC ¹	13	E3	DI	data chip select / byte sync
DVDD1	14	F3	PWR	digital power supply
DGND1	16	F4	PWR	digital ground
XTALO	17	G3	AO	crystal output
XTALI	18	E4	AI	crystal input
DVDD2	19	F5	PWR	digital power supply
DGND2	20	F6	PWR	digital ground (in BGA-49, DGND2, 3, 4 conn. together)
DGND3	21	F6	PWR	digital ground
DGND4	22	F6	PWR	digital ground
XCS	23	G6	DI	chip select input (active low)
RX	26	E6	DI	UART receive, use pull-up resistor if not used
TX	27	F7	DO	UART transmit
SCLK	28	D6	DI	clock for serial bus
SI	29	E7	DI	serial input
SO	30	D5	DO3	serial output
TEST	32	C6	DI	reserved for test, connect to DVDD
SPIBOOT / GPIO0 ³	33	C7	DIO	general purpose IO 0, use 100 k Ω pull-down resistor
GPIO1 ²	34	B6	DIO	general purpose IO 1, use pull-down resistor if not used
AGND0	37	C5	PWR	analog ground, low-noise reference
AVDD0	38	B5	PWR	analog power supply
RIGHT	39	A6	AO	right channel output
AGND1	40	B4	PWR	analog ground
AGND2	41	A5	PWR	analog ground
GBUF	42	C4	AO	virtual ground for audio output, 1.23 V nominal
AVDD1	43	A4	PWR	analog power supply
RCAP	44	B3	AIO	filtering capacitance for reference
AVDD2	45	A3	PWR	analog power supply
LEFT	46	B2	AO	left channel output
AGND3	47	A2	PWR	analog ground

¹ First pin function is active in New Mode, latter in Compatibility Mode.

² If not used, use 100 k Ω pull-down resistor.

³ Unless pull-down resistor is used, SPI Boot is tried. See Chapter 9.4 for details.

Type	Description	Type	Description
DI	Digital input, CMOS Input Pad	AI	Analog input
DO	Digital output, CMOS Input Pad	AO	Analog output
DIO	Digital input/output	AIO	Analog input/output
DO3	Digital output, CMOS Tri-stated Output Pad	PWR	Power supply pin

In BGA-49, no-connect balls are A1, A7, B7, C1, D1, D4, D7, E5, F1, G1, G2, G7.

In LQFP-48, no-connect pins are 5, 7, 11, 12, 15, 24, 25, 31, 35, 36, 48.

เอกสารนี้เป็นเอกสารลิขสิทธิ์ของมหาวิทยาลัยสงขลานครินทร์ ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6 Connection Diagram, LQFP-48

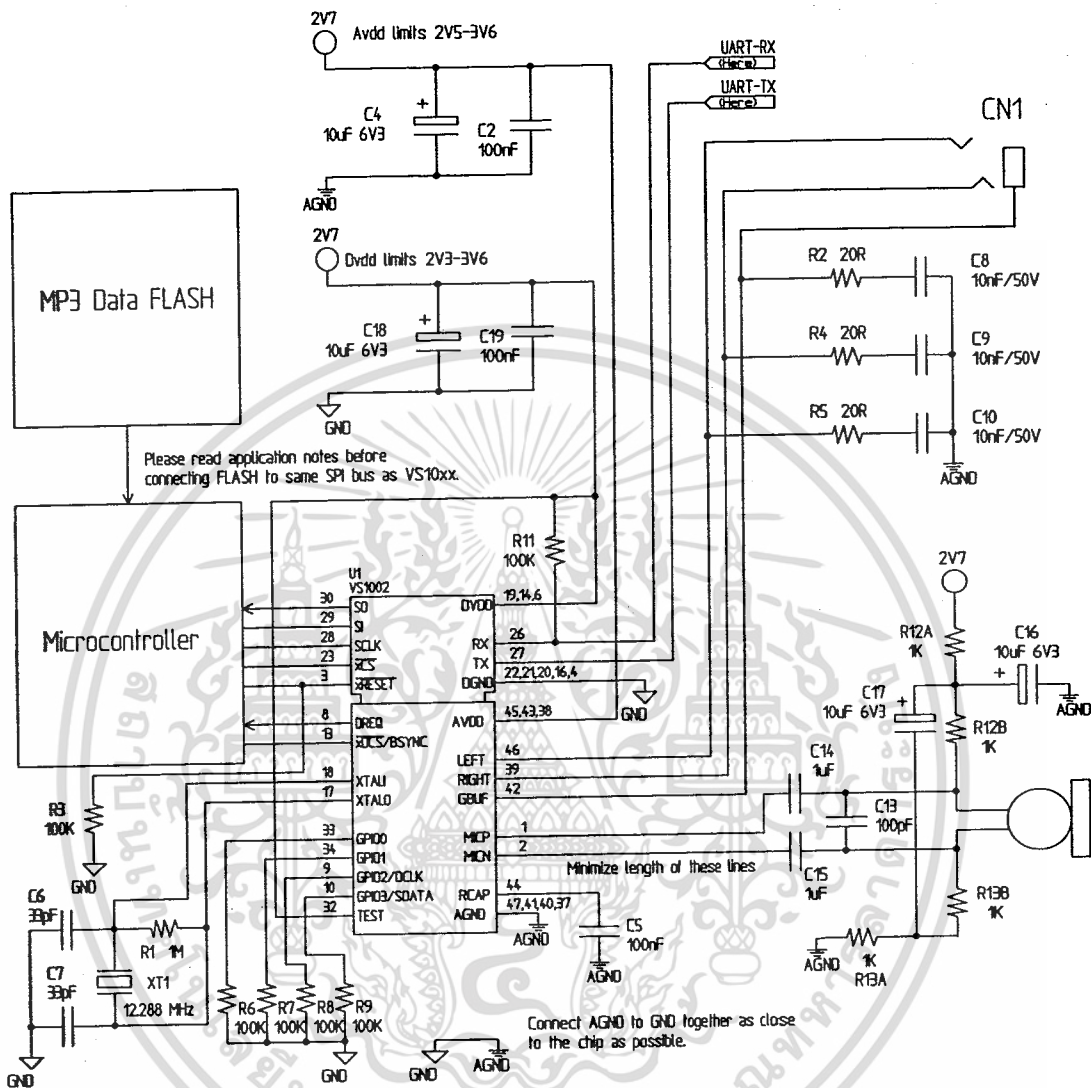


Figure 3: Typical Connection Diagram Using LQFP-48.

The ground buffer GBUF can be used for common voltage (1.23 V) for earphones. This will eliminate the need for large isolation capacitors on line outputs, and thus the audio output pins from VS1002d may be connected directly to the earphone connector.

If GBUF is not used, LEFT and RIGHT must be provided with 1-100 μF capacitors depending load resistance.

If UART is not used, RX should connect to DVDD and TX be unconnected.

Note: This connection assumes SM_SDINew is active (see Chapter 8.6.1). If also SM_SDISHARE is used, xDCS should have a pull-up resistor (see Chapter 7.2.1).

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7 SPI Buses

7.1 General

The SPI Bus - that was originally used in some Motorola devices - has been used for both VS1002d's Serial Data Interface SDI (Chapters 7.4 and 8.4) and Serial Control Interface SCI (Chapters 7.5 and 8.5).

7.2 SPI Bus Pin Descriptions

7.2.1 VS1002 Native Modes (New Mode)

These modes are active when SM_SDINEW is set to 1 (default at startup). DCLK, SDATA and BSYNC are replaced with GPIO2, GPIO3 and XDCS, respectively.

SDI Pin	SCI Pin	Description
XDCS	XCS	Active low chip select input. A high level forces the serial interface into standby mode, ending the current operation. A high level also forces serial output (SO) to high impedance state. If SM_SDISHARE is 1, pin XDCS is not used, but the signal is generated internally by inverting XCS.
SCK		Serial clock input. The serial clock is also used internally as the master clock for the register interface. SCK can be gated or continuous. In either case, the first rising clock edge after XCS has gone low marks the first bit to be written.
SI		Serial input. If a chip select is active, SI is sampled on the rising CLK edge.
-	SO	Serial output. In reads, data is shifted out on the falling SCK edge. In writes SO is at a high impedance state.

7.2.2 VS1001 Compatibility Mode

This mode is active when SM_SDINEW is set to 0. In this mode, DCLK, SDATA and BSYNC are active.

SDI Pin	SCI Pin	Description
-	XCS	Active low chip select input. A high level forces the serial interface into standby mode, ending the current operation. A high level also forces serial output (SO) to high impedance state. There is no chip select for SDI, which is always active.
BSYNC	-	SDI data is synchronized with a rising edge of BSYNC.
DCLK	SCK	Serial clock input. The serial clock is also used internally as the master clock for the register interface. SCK can be gated or continuous. In either case, the first rising clock edge after XCS has gone low marks the first bit to be written.
SDATA	SI	Serial input. SI is sampled on the rising SCK edge, if XCS is low.
-	SO	Serial output. In reads, data is shifted out on the falling SCK edge. In writes SO is at a high impedance state.

7.3 Data Request Pin DREQ

The DREQ pin/signal is used to signal if VS1002d's FIFO is capable of receiving data. If DREQ is high, VS1002d can take at least 32 bytes of SDI data or one SCI command. When these criteria are not met, DREQ is turned low, and the sender should stop transferring new data.

Because of the 32-byte safety area, the sender may send upto 32 bytes of SDI data at a time without checking the status of DREQ, making controlling VS1002d easier for low-speed microcontrollers.

Note: DREQ may turn low or high at any time, even during a byte transmission. Thus, DREQ should only be used to decide whether to send more bytes. It should not abort a transmission that has already started.

7.4 Serial Protocol for Serial Data Interface (SDI)

7.4.1 General

The serial data interface operates in slave mode so the DCLK signal must be generated by an external circuit.

Data (SDATA signal) can be clocked in at either the rising or falling edge of DCLK (Chapter 8.6).

VS1002d assumes its data input to be byte-synchronized. SDI bytes may be transmitted either MSb or LSb first, depending of contents of SCI_MODE (Chapter 8.6.1).

The firmware is able to accept the maximum bitrate the SDI supports.

7.4.2 SDI in VS1002 Native Modes (New Mode)

In VS1002 native modes, byte synchronization is achieved by XDSCS (or XCS if SM_SDISHARE is 1). The state of XDSCS (or XCS) may not change while a data byte transfer is in progress. To always maintain data synchronization even if there may be glitches in the boards using VS1002d, it is recommended to turn XDSCS (or XCS) every now and then, for instance once after every flash data block or a few kilobytes, just to keep sure the host and VS1002d are in sync.

If SM_SDISHARE is 1, the XDSCS signal is internally generated by inverting the XCS input.

For new designs, using VS1002 native modes are recommended.

7.4.3 SDI in VS1001 Compatibility Mode

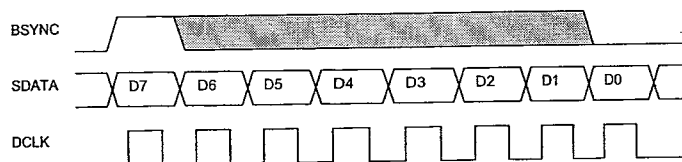


Figure 4: BSYNC Signal - one byte transfer.

When VS1002d is running in VS1001 compatibility mode, a BSYNC signal must be generated to ensure correct bit-alignment of the input bitstream. The first DCLK sampling edge (rising or falling, depending on selected polarity), during which the BSYNC is high, marks the first bit of a byte (LSB, if LSB-first order is used, MSB, if MSB-first order is used). If BSYNC is '1' when the last bit is received, the receiver stays active and next 8 bits are also received.

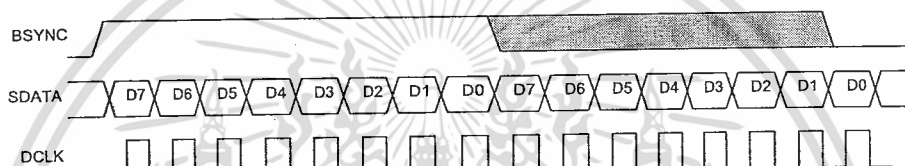


Figure 5: BSYNC Signal - two byte transfer.

7.5 Serial Protocol for Serial Command Interface (SCI)

7.5.1 General

The serial bus protocol for the Serial Command Interface SCI (Chapter 8.5) consists of an instruction byte, address byte and one 16-bit data word. Each read or write operation can read or write a single register. Data bits are read at the rising edge, so the user should update data at the falling edge. Bytes are always send MSb first.

The operation is specified by an 8-bit instruction opcode. The supported instructions are read and write. See table below.

Instruction		
Name	Opcode	Operation
READ	0b0000 0011	Read data
WRITE	0b0000 0010	Write data

Note: After sending an SCI command, it is not allowed to send SCI or SDI data for 5 microseconds.

7.5.2 SCI Read

VS1002d registers are read by the following sequence, as shown in Figure 6. First, XCS line is pulled low to select the device. Then the READ opcode (0x3) is transmitted via the SI line followed by an 8-bit word address. After the address has been read in, any further data on SI is ignored. The 16-bit data corresponding to the received address will be shifted out onto the SO line.

XCS should be driven high after data has been shifted out.

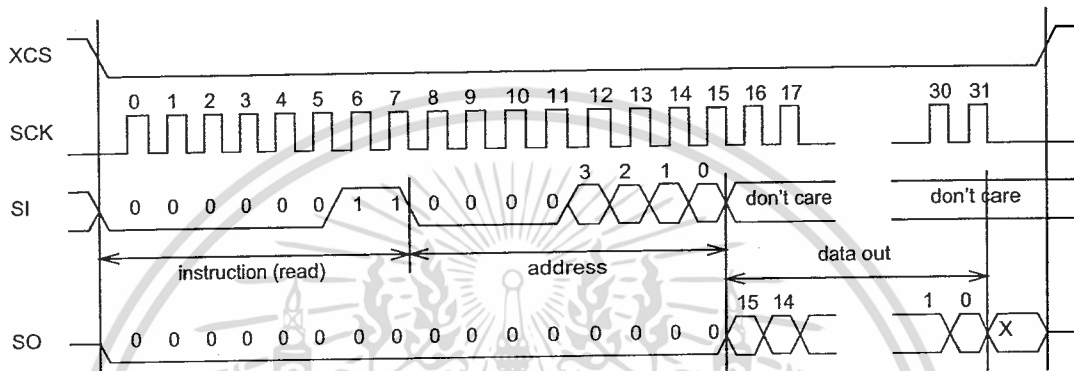


Figure 6: SCI Word Read

7.5.3 SCI Write

VS1002d registers are written to using the following sequence, as shown in Figure 7. First, XCS line is pulled low to select the device. Then the WRITE opcode (0x2) is transmitted via the SI line followed by an 8-bit word address.

After the word has been shifted in and the last clock has been sent, XCS should be pulled high to end the WRITE sequence.

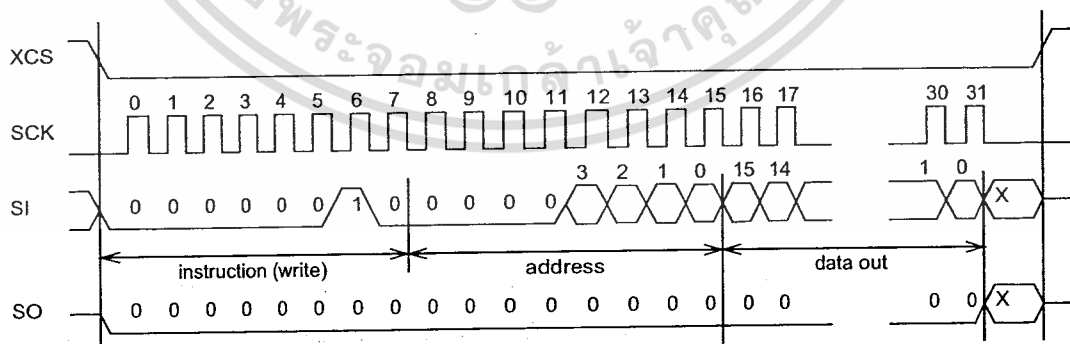


Figure 7: SCI Word Write

7.6 SPI Timing Diagram

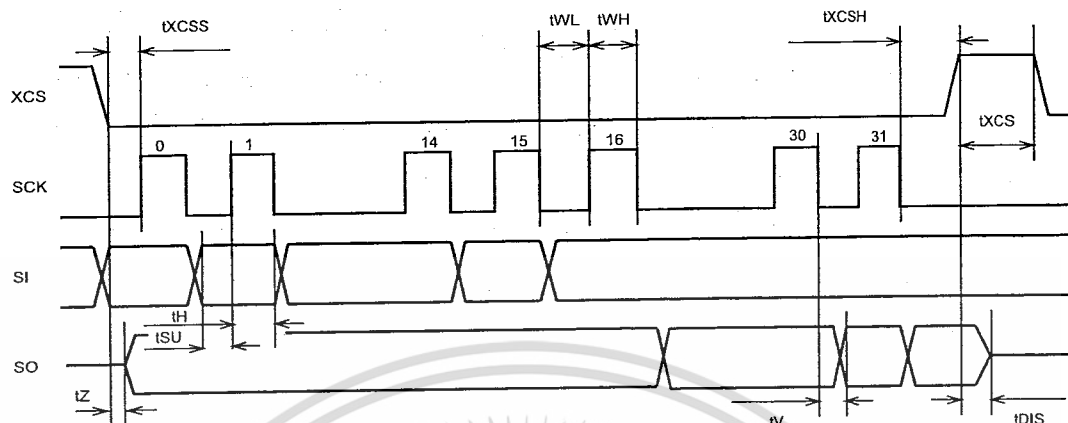


Figure 8: SPI Timing Diagram.

Symbol	Min	Max	Unit
tXCSS	5		ns
tSU	-26		ns
tSH	2		XTALI cycles
tSZ	0		ns
tWL	2		XTALI cycles
tWH	2		XTALI cycles
tV		2 (+ 25ns ¹)	XTALI cycles
tXCSH	-26		ns
tXCS	2		XTALI cycles
tDIS		10	ns

¹ 25ns is when pin loaded with 100pF capacitance. The time is shorter with lower capacitance.

Note: As tWL and tWH, as well as tSH require at least 2 clock cycles, the maximum speed for the SPI bus that can easily be used is 1/6 of VS1011's external clock speed XTALI. Slightly higher speed can be achieved with very careful timing tuning. For details, see Application Notes for VS10XX.

Note: Negative numbers mean that the signal can change in different order from what is shown in the diagram.

7.7 SPI Examples with SM_SDINEW and SM_SDISHARED set

7.7.1 Two SCI Writes

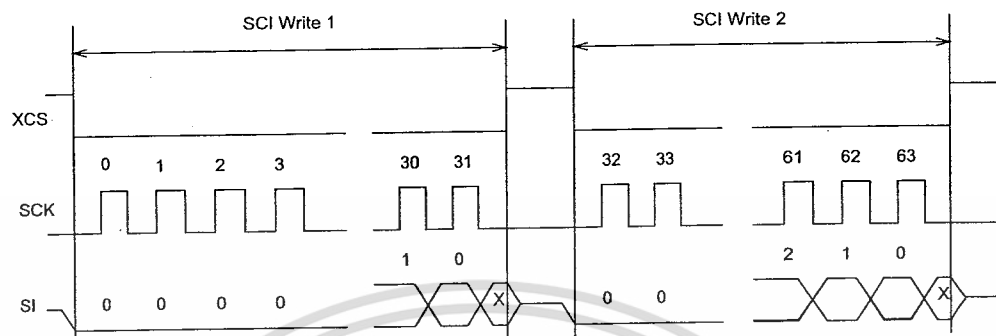


Figure 9: Two SCI Operations.

Figure 9 shows two consecutive SCI operations. Note that xCS *must* be raised to inactive state between the writes.

7.7.2 Two SDI Bytes

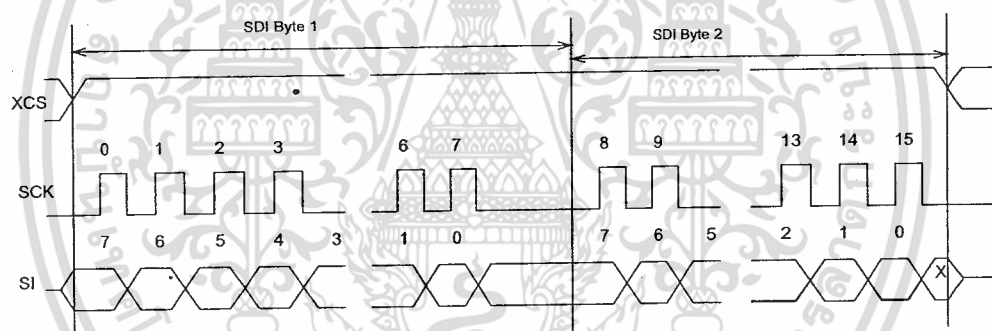


Figure 10: Two SDI Bytes.

SDI data is synchronized with a raising edge of xCS as shown in Figure 10. However, every byte doesn't need separate synchronization.

7.7.3 SCI Operation in Middle of Two SDI Bytes

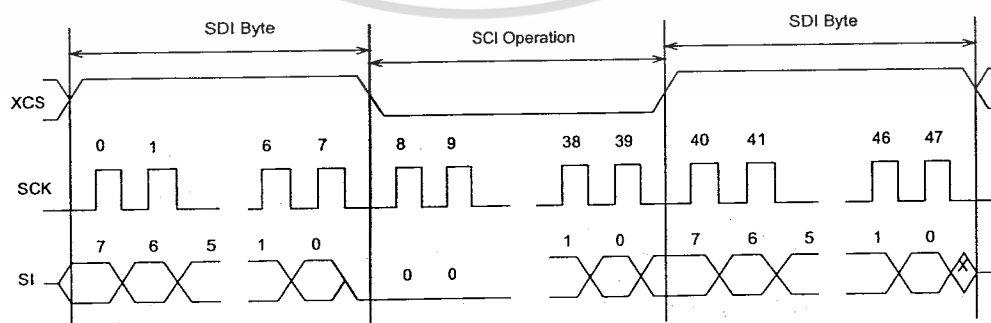


Figure 11: Two SDI Bytes Separated By an SCI Operation.

Figure 11 shows how an SCI operation is embedded in between SDI operations. The changes in xCS are used to synchronize both SDI and SCI.

เอกสารนี้เป็นเอกสารที่เผยแพร่ในนามของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

โครงการชิ้นนี้จะไม่สามรถเสร็จสมบูรณ์ได้ หากขาดการสนับสนุนของอาจารย์ที่ปรึกษา
รศ.ดร.สมศักดิ์ ชุมช่วย ที่ให้ความรู้ให้การสนับสนุนด้านข้อมูลและคำปรึกษาที่ดีตลอดมา รวมถึง
จากอาจารย์ทุกท่านในภาควิชาอิเล็กทรอนิกส์ รุ่นพี่ปริญญาโท และรุ่นพี่ที่จบไปแล้วทั้งทางด้าน
ข้อมูลและวัสดุอุปกรณ์ต่างๆ และที่ต้องให้ความขอบคุณที่สุดคือคุณพ่อ คุณแม่ ที่เลี้ยงเรามาจนมี
วันนี้ จึงใคร่ขอขอบพระคุณทุกท่านผู้มีพระคุณมา ณ ที่นี้

คณะผู้จัดทำ



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารอ้างอิง

1. http://www.arip.co.th/2006/forum.php?forum_id=Hardware&g1=4&p_id=18519
2. http://www.alliedcomp.co.th/index.php?option=com_mamboboard&func=view&catid=20&id=29
3. <http://www.zone-it.com/66173>
4. <http://www.com-th.net/webboard/index.php?topic=13214.0>
5. <http://www.leksound.net/forum/index.php?PHPSESSID=ccd0db5e78aea9531eadddc1dc1a8984&topic=11920.0>
6. <http://www.asiamediasoft.net/content/newboard/WHAT-MP3>
7. <http://www.bloggang.com/viewdiary.php?id=pingpo&month=07-2005&date=05&group=8&gblog=13>
8. <http://learners.in.th/blog/mp3/58659>
9. <http://cannot.info/encode>